UNIVERSITÀ DEGLI STUDI DI TRENTO

Dipartimento di Ingegneria e Scienza dell'Informazione

Corso di Laurea in
Informatica

Elaborato finale

Realizzazione di un sistema IoT basato su Android per l'acquisizione multimediale di lezioni universitarie

Supervisore

Marco Ronchetti

Laureando

Matteo Contrini

Anno accademico 2018/2019

Ringraziamenti

La presente tesi è stata redatta nell'ambito di un tirocinio formativo svolto presso l'azienda AXIA Studio. Grazie a Tiziano, Chloe e al prof. Ronchetti per avermi seguito nel percorso.

Indice

Sommario

L'oggetto di questa tesi è lo sviluppo di un sistema di cattura e registrazione multimediale di lezioni universitarie, conferenze e seminari. Il sistema prodotto si basa su un dispositivo fisico dotato di apposito hardware per la cattura video, sul quale viene eseguito un apposito software su sistema operativo Android.

Il sistema è stato realizzato nell'ambito del progetto LODE (Lectures On DEmand) del professor Marco Ronchetti (Università di Trento), ed è stato elaborato durante un tirocinio formativo presso l'azienda AXIA Studio, con l'obiettivo finale di affinare la soluzione e di immetterla sul mercato delle soluzioni IoT accademiche.

Il progetto LODE, nel cui contesto questo lavoro si inserisce, è un progetto con l'obiettivo di fornire una soluzione low cost per l'acquisizione video delle lezioni universitarie. L'ultima versione del sistema prevede la possibilità di registrare flussi multimediali multipli, in particolare il video in uscita dal computer del docente, il video acquisito da una videocamera IP che riprende il docente e/o la lavagna, e l'audio di un microfono indossato dal docente. I contenuti sono poi elaborati e pubblicati su una pagina web accessibile dagli studenti del corso.

Una funzionalità del sistema prevede anche che durante le lezioni lo studente possa catturare degli screenshot di ciò che viene proiettato in quel momento, e inserire delle annotazioni testuali o disegnare sulle catture.

Nella sua ultima iterazione, LODE prevede l'uso di un dispositivo fisico, soprannominato LodeBox, che incorpora un "mini-computer" Raspberry Pi. Tramite apposite estensioni hardware, in particolare un modulo "HDMI to CSI-2", il dispositivo è in grado di acquisire un input HDMI come se si trattasse del video di una videocamera, e di sfruttare funzionalità come l'anteprima su schermo e la registrazione H.264 tramite encoder hardware. Il dispositivo prevede anche la possibilità di collegare un proiettore tramite uscita HDMI, un microfono tramite dongle USB, e una videocamera IP/RTSP (Real Time Streaming Protocol) tramite la rete locale.

Questa soluzione si scontra però con delle difficoltà che rendono difficile la realizzazione di un sistema affidabile e distribuibile su larga scala. Tra le problematiche principali si evidenziano la difficoltà nel gestire situazioni "plug and play", come la disconnessione e la riconnessione del cavo HDMI, la mancanza di storage integrato duraturo e ad alte prestazioni, le scarse garanzie sulla disponibilità del modulo hardware di conversione HDMI.

Ci si è orientati quindi verso la ricerca di SBC (Single-Board Computer) alternative più adatte per la realizzazione di applicazioni multimediali. Tra le soluzioni più accessibili dal punto di vista economico spiccano molte board basate sul sistema operativo Android, che offre i benefici di avere una piattaforma ben nota, documentata e con la possibilità di sostituire l'hardware riusando gran parte del codice scritto.

Gran parte del mio lavoro si è quindi concentrato sullo sviluppo di alcune applicazioni Java/Kotlin, con lo scopo di verificare la fattibilità di una versione di LodeBox con piattaforma Android.

La peculiarità fondamentale di molte board Android con input HDMI è che permettono di sfruttare direttamente le API di Android per l'acquisizione di video e immagini, permettendo di scrivere codice che non sia strettamente legato all'hardware. La piattaforma Android prevede due versioni delle API per l'accesso alla fotocamera, in particolare la classe `android.hardware.Camera` e le classi contenute in `android.hardware.camera2.*`. A partire da Android 5.0 (livello API 21), la classe `Camera` è deprecata ed è invece consigliato l'uso delle API `camera2`, più avanzate ma anche più complesse da usare. Ciò non ostante, non è garantito che ogni dispositivo con Android 5 o superiore supporti a pieno le API `camera2`.

È stato quindi necessario approfondire il funzionamento di entrambe le versioni delle API, per determinare se entrambe permettono di soddisfare i requisiti di LODE e quale versione conviene usare a seconda dell'hardware che si ha a disposizione.

Oltre all'input HDMI, si ha la necessità di registrare anche una videocamera esterna, che per contenere i costi è una videocamera IP raggiungibile nella rete locale tramite il protocollo di streaming RTSP. Di conseguenza, il flusso può essere registrato direttamente sulla board, utilizzando una versione della libreria `ffmpeg` compilata per funzionare su Android.

Un altro punto importante dello sviluppo di un'applicazione embedded per Android è la realizzazione di una modalità *kiosk*, cioè una situazione in cui l'utilizzatore del sistema può utilizzare solo ed esclusivamente una singola applicazione, senza poter accedere alle altre parti del sistema operativo o ad altre funzioni. Questo ha portato a sperimentare diverse soluzioni per assicurarsi che l'applicazione resti sempre a schermo intero, tra cui la modalità "lock task" di Android e l'avvio automatico come applicazione launcher.

Il risultato di questa fase è stato un insieme di applicazioni-prototipo che sono state sperimentate sul campo durante alcune lezioni presso l'Università di Trento, permettendo di raccogliere importanti riscontri sul funzionamento del sistema.

A questo punto la fattibilità delle funzionalità di base che il sistema deve fornire sono state verificate, e i passi successivi hanno riguardato altri aspetti per migliorare il funzionamento del software.

Uno di questi è la possibilità di mettere in pausa e riprendere la registrazione della lezione. L'idea considerata è stata quella di non sospendere la registrazione ma di escludere in post-produzione i segmenti di video corrispondenti alle pause. Questo è agevolmente ottenibile grazie a `ffmpeg` e con l'ausilio di uno script che generi il comando (potenzialmente lungo) per ritagliare i segmenti. Nel caso in cui non fosse possibile avere una registrazione unica, una tecnica simile può essere adottata per unire i segmenti video in fase di post-elaborazione.

Un'altra questione approfondita riguarda la sincronizzazione dei flussi, con lo scopo principale di evitare che l'audio risulti troppo sfasato rispetto al video, ed evitare che sia visibile lo sfasamento tra voce registrata e labiale del relatore. Il problema sorge principalmente per via della presenza del video RTSP, la cui latenza è difficilmente stimabile[1]. La soluzione sperimentale proposta prevede di incorporare l'audio nella registrazione video HDMI (in modo da ottenere una naturale sincronizzazione tra i due flussi acquisiti via cavo), e di occuparsi invece di sincronizzare video HDMI e RTSP. L'implementazione sperimentale è stata ottenuta mostrando su schermo un marcatore visivo che permetta di individuare con precisione un instante comune tra i due flussi video, e di conseguenza sincronizzarli.

Come accennato, un'altra funzione del sistema LODE prevede che lo studente possa prendere appunti in tempo reale durante lo svolgimento delle lezioni, catturando degli screenshot di ciò che vede in quel momento. Per evitare di inviare inutilmente screenshot al server nel caso in cui non ci sia stato nessun cambiamento percepibile nell'immagine, un sistema intelligente può rilevare le differenze tra i fotogrammi per determinare se l'immagine è nuova o invariata. Dopo aver appurato empiricamente che il confronto di tutti i pixel di due fotogrammi risulta computazionalmente dispendioso, un'alternativa considerata è quella di scegliere con un fattore di casualità un numero limitato di pixel "salienti". A livello intuitivo, il metodo sviluppato realizza una griglia con un numero oscillante di righe e colonne, le cui intersezioni determinano una quantità limitata di pixel che possono essere utilizzati per rilevare rapidamente le differenze tra i fotogrammi.

I capitoli di questa tesi approfondiscono più dettagliatamente le problematiche e le soluzioni che sono state sintetizzate in questo sommario.

1 Introduzione

Questo capitolo introduce il progetto LODE sviluppato presso l'Università di Trento, nel cui contesto questa tesi si inserisce. Sono approfonditi motivazioni e vantaggi, ed è introdotta l'ultima versione del sistema, cioè un dispositivo hardware dal nome LodeBox. Sono infine presentate le principali problematiche legate a LodeBox e l'hardware alternativo considerato per l'evoluzione del progetto.

[1] La latenza è influenzata dalla rete, dai tempi di codifica e dai buffer di trasmissione e ricezione. Non è quindi facile trovare un intervallo di tempo abbastanza preciso per sincronizzare il video con gli altri flussi.

1.1 LODE: introduzione e motivazioni

LODE (Lectures On DEmand) è un progetto realizzato dal professor Marco Ronchetti e collaboratori presso l'Università di Trento. Si presenta come una soluzione per l'acquisizione in formato video e audio di lezioni universitarie, con la particolarità di essere una soluzione a basso costo e facilmente manovrabile.[1]

Le lezioni registrate possono poi essere consultate tramite una pagina web appositamente generata, che combina il video del computer del docente, il video ripreso da una videocamera posizionata in aula e l'audio catturato da un microfono.

I vantaggi che questo sistema offre sono molteplici, tra cui: la possibilità per gli studenti-lavoratori di seguire le lezioni in remoto a qualsiasi orario; la possibilità di recuperare le lezioni in caso di assenze non volontarie (es. malattia); supporto per gli studenti che non comprendono bene la lingua del corso; la possibilità di rivedere porzioni specifiche di qualsiasi lezione in qualsiasi momento, e di verificare quindi la qualità dei propri appunti e il livello di comprensione.

Le versioni più recenti di LODE prevedono anche un'interfaccia web utilizzabile dagli studenti durante lo svolgimento della lezione. Questo strumento permette di catturare degli screenshot di quanto proiettato dal docente in quell'istante, e di scrivere o disegnare annotazioni sulle catture. In questo modo gli studenti sono coinvolti in modo meno passivo e seguono la lezione con più attenzione.

1.2 La soluzione LodeBox

L'ultima versione di LODE prevede l'utilizzo di una board Raspberry Pi per eseguire il software di acquisizione. Il dispositivo è inserito in una piccola scatola che espone dei connettori verso l'esterno. Tra questi sono presenti un ingresso HDMI per collegare un computer e un'uscita HDMI per collegare un proiettore o uno schermo. Le porte USB permettono di collegare un telecomando e un "dongle" per l'acquisizione dell'audio di un microfono con jack 3,5 mm.

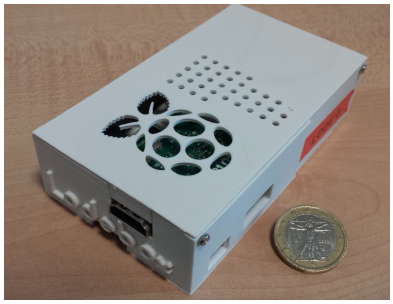

Figura 1.1: La scatola di LodeBox.

La scatola include un modulo "HDMI to MIPI CSI-2", che permette di convertire il segnale HDMI nel formato seriale della fotocamera. Il video HDMI è infatti acquisito utilizzando la libreria `PiCamera` per Python, che permette di acquisire il segnale come se si trattasse del video di una fotocamera. La libreria semplifica di molto lo sviluppo, perché espone delle funzioni di alto livello per abilitare l'anteprima del video a schermo intero, la registrazione con un encoder H.264[1] con accelerazione hardware e la cattura di screenshot.

Sempre in ottica di riduzione dei costi, LodeBox prevede la ripresa del docente tramite una qualsiasi videocamera IP (wireless o cablata) che supporti il protocollo RTSP (Real Time Streaming Protocol). La registrazione avviene sul dispositivo utilizzando `ffmpeg`/`avconv`[2] in modalità copia (senza codifica), richiedendo quindi un uso molto basso di risorse.

Raspberry Pi non prevede spazio di archiviazione interno, e richiede quindi di utilizzare una scheda microSD per memorizzare le registrazioni. Le memorie di tipo flash hanno però vita limitata, a seconda di quanto intensivamente sono utilizzate, e questo riduce di conseguenza l'affidabilità a lungo termine del sistema.

Un altro svantaggio di questa soluzione è il livello di "fault tolerance" in caso di eventi come la disconnessione (volontaria o meno) del cavo HDMI in ingresso. Dato che l'input HDMI è mappato sull'interfaccia della fotocamera, non è previsto che la fotocamera possa essere improvvisamente scollegata. Secondo le prove effettuate, risulta molto difficile rilevare in modo affidabile la disconnessione del cavo HDMI. Nei casi in cui è possibile, non risulta invece fattibile il recupero dell'applicazione,

[1] H.264, conosciuto anche come AVC o MPEG-4 Part-10, è il più popolare codec di compressione video. Nato nel 2003, è ancora oggi lo standard de facto in numerosi ambiti, tra cui lo streaming video.

[2] *ffmpeg* è uno strumento estremamente popolare per l'elaborazione di video e audio tramite linea di comando. Supporta numerosi formati, codec e protocolli, e permette di effettuare operazioni quali il muxing, transmuxing, transcoding e altre funzioni più avanzate. *avconv* è un fork di ffmpeg nato per via di divergenze sui metodi di sviluppo, ed è stato incluso per un breve periodo in alcune distribuzioni Linux in sostituzione di ffmpeg.

perché le operazioni sulla `PiCamera` sollevano eccezioni o bloccano indefinitamente l'esecuzione del codice.

Infine, un altro problema è posto dalla presenza del modulo di conversione HDMI-CSI, la cui disponibilità e compatibilità a lungo termine non sono garantite.

1.3 Hardware alternativo

Per poter evolvere LodeBox in una soluzione più affidabile e adatta alla produzione e distribuzione, si sono quindi cercate SBC (Single-Board Computer) alternative, preferibilmente pensate per la realizzazione di applicazioni multimediali. I principali vincoli per la scelta di una nuova scheda erano la presenza dell'input HDMI, la possibilità di collegare un microfono, e il costo accessibile.

Molte board con sistema operativo Android soddisfano questi requisiti, e consentono anche di avere a disposizione una piattaforma nota, documentata, di facile sviluppo e per cui sono disponibile molte risorse. In molti casi l'input HDMI è reso disponibile tramite l'interfaccia della fotocamera CSI, da cui deriva la possibilità di acquisire l'input tramite le API di Android per l'accesso alla fotocamera.

2 Acquisizione video HDMI

Lorem ipsum dolor sit amet.

2.1 Le API `android.hardware.Camera`

Lorem ipsum dolor sit amet.

2.2 Le API `android.hardware.camera2.*`

Lorem ipsum dolor sit amet.[2]

3 Acquisizione video RTSP

Lorem ipsum dolor sit amet.

3.1 Il protocollo RTSP

Lorem ipsum dolor sit amet.

3.2 La libreria MobileFFmpeg

Lorem ipsum dolor sit amet.

4 Realizzazione della modalità "kiosk"

Lorem ipsum dolor sit amet.

4.1 Il tema a schermo intero

Lorem ipsum dolor sit amet.

4.2 L'overscan di sistema

The Android WindowManager is a system service, which is responsible for managing the z-ordered list of windows, which windows are visible, and how they are laid out on screen. Among other things, it automatically performs window transitions and animations when opening or closing an app or rotating the screen.

4.3 La modalità "lock task"

Lorem ipsum dolor sit amet.

4.4 L'animazione di avvio

Lorem ipsum dolor sit amet.

5 Sospensione della registrazione

Lorem ipsum dolor sit amet.

6 Sincronizzazione di video e audio

Lorem ipsum dolor sit amet.

6.1 Definizione del problema

Lorem ipsum dolor sit amet.

6.2 Proposta di soluzione

Lorem ipsum dolor sit amet.

6.3 Implementazione e sperimentazione

Lorem ipsum dolor sit amet.

7 Rilevamento delle differenze tra fotogrammi

Lorem ipsum dolor sit amet.

7.1 In post-produzione

Lorem ipsum dolor sit amet.

7.2 In tempo reale

Lorem ipsum dolor sit amet.

7.2.1 Implementazione e sperimentazione

Lorem ipsum dolor sit amet.

8 Acquisizione audio (forse)

Lorem ipsum dolor sit amet.

9 Conclusione

Lorem ipsum dolor sit amet.

Bibliografia

[1] Ronchetti Marco. Strumenti per favorire l'apprendimento nei primi anni di università. Parte 2: Il sistema LodeBox. In Raffone Alessandra, editor, *La città educante. Metodologie e tecnologie innovative a servizio delle smart communities*, pages 137–146. Liguori, 2018.

[2] Wahltinez Oscar. Understanding Android camera capture sessions and requests. https://medium.com/androiddevelopers/understanding-android-camera-capture-sessions-and-requests-4e54d9150295, settembre 2018. Ultimo accesso 25/06/2019.

Allegato A Titolo primo allegato

Lorem ipsum dolor sit amet.