SISTEMA OCHO

OCHOGLASS & OCHOSTICK

Visione Artificiale e Telemetria Laser per l'Assistenza alla Mobilità

Liceo Scientifico Internazionale STEM Virginia Centurione Bracelli - Biennio

1. Visione del Progetto e Obiettivi Scientifici

Il progetto **OchoGlass & OchoStick** rappresenta una soluzione di frontiera nel campo delle tecnologie assistive. L'obiettivo è fornire all'utente ipovedente una "percezione aumentata" del contesto circostante attraverso un sistema di analisi intelligente.

Mentre il bastone tradizionale si limita a rilevare ostacoli fisici tramite contatto, il sistema **Ocho** integra:

- **Analisi Semantica:** Identificazione degli oggetti (sedie, porte, persone) tramite algoritmi di Deep Learning.
- **Telemetria Laser Infrarossa:** Misurazione della distanza esatta tramite sensori Time-of-Flight (ToF), garantendo un monitoraggio senza contatto.
- **Eco-sistema Wireless:** Dialogo costante tra gli occhiali (unità centrale) e il bastone (unità di rilevamento al suolo) per una protezione coordinata.

2. Architettura Hardware: Componenti e Specifiche

Il sistema è stato progettato seguendo un approccio "No-Solder" per facilitare la prototipazione rapida e la manutenzione didattica, utilizzando componenti di alta precisione integrati su piattaforme ESP32.

2.1 Elenco Componenti OchoGlass (Occhiali)

➔ **Microcontrollore Principale:** *Seeed Studio XIAO ESP32-S3 Sense*. Dotato di CPU dual-core fino a 240MHz, 8MB di PSRAM (fondamentale per i buffer delle immagini) e

supporto hardware per il calcolo vettoriale (NPU).

- ➔ **Modulo Camera:** *OV2640*. Sensore da 2 Megapixel integrato sulla XIAO, utilizzato per catturare i frame destinati all'inferenza dell'IA.
- ➔ **Sensore di Distanza (ToF):** *VL53L0X*. Un sensore laser miniaturizzato che utilizza il tempo di volo della luce infrarossa per misurare distanze fino a 2 metri con precisione millimetrica.
- ➔ **Modulo Audio:** *DFPlayer Mini*. Un lettore MP3 compatto controllato via seriale UART, che decodifica i file salvati su memoria esterna.
- ➔ **Memoria:** *MicroSD (16GB)*. Contiene i campioni audio pre-registrati (es. "Sedia rilevata", "Ostacolo basso").
- ➔ **Alimentazione:** *Batteria LiPo 3.7V*. Gestita dal circuito di ricarica integrato nella XIAO.
- ➔ **Interfaccia Fisica:** *Interruttore* per l'accensione generale e *Mini Breadboard (170 punti)* per i collegamenti.
- ➔ **Supporto:** *Montatura per occhiali protettivi* con stanghette larghe per ospitare l'elettronica.

2.2 Elenco Componenti OchoStick (Bastone)

- ➔ **Microcontrollore:** *ESP32 DevKit V1 (Diymore)*. Scelto per la robustezza e la facilità di collegamento ai sensori di terra, funge da trasmettitore wireless via ESP-NOW.
- ➔ **Sensore di Distanza (ToF):** *VL53L0X*. Montato in posizione angolata verso il suolo per intercettare dislivelli (gradini) o ostacoli bassi.
- ➔ **Modulo motore a vibrazione:** motore DC5V 9000 giri/min
- ➔ **Alimentazione:** *Batteria LiPo 3.7V*.
- ➔ **Struttura:** *Bastone di legno* con breadboard fissata per l'elettronica.

3. Organizzazione del Lavoro e Metodologia

Il progetto è stato diviso per le due classi affidando OchoGlass al II liceo e OchoStick al I liceo. OchoGlass è stato sviluppato attraverso una struttura a gruppi di lavoro specializzati:

- **Team Hardware & Chassis:** Responsabile dell'ergonomia e del bilanciamento dei pesi sulla montatura. Ha curato il fissaggio dei componenti e l'ingegnerizzazione dei supporti.
- **Team Telemetria & Audio:** Ha gestito il protocollo **I2C** per i sensori laser e la comunicazione **UART** per il modulo audio.
- **Team AI (Edge Impulse):** Ha curato la raccolta del dataset fotografico e l'addestramento del modello **FOMO** (Faster Objects, More Objects).
- **Team Integration (System Leads):** Responsabile dell'architettura del software e della "fusione" dei sistemi tramite una macchina a stati.

4. Logica del Software e Implementazione Dettagliata

L'architettura software rappresenta l'intelligenza del sistema, orchestrando il flusso di dati tra sensori, comunicazione wireless e modelli di visione artificiale in un ambiente a risorse limitate.

4.1 Gestione delle Priorità e Multitasking

Il software deve elaborare dati provenienti da fonti eterogenee con latenze diverse. Per garantire la sicurezza dell'utente, è stata implementata una logica di **priorità asimmetrica** basata su una "macchina a stati" non bloccante:

1. **Livello 0 - Sicurezza Critica (Sensore ToF):** È il thread prioritario. Il laser VL53L0X viene interrogato ogni 50 ms. Se la distanza rilevata è inferiore alla soglia di sicurezza (400 mm), il sistema interrompe immediatamente ogni altra attività (inclusa l'inferenza IA) per azionare l'allarme vocale.
2. **Livello 1 - Eventi Wireless (ESP-NOW):** Il modulo ESP32 rimane in ascolto passivo dei pacchetti provenienti dall'OchoStick. Al ricevimento di un "interrupt" wireless, viene attivata una funzione di callback che inserisce il messaggio d'avviso nella coda audio.
3. **Livello 2 - Analisi Semantica (Edge AI):** L'inferenza IA è l'attività computazionalmente più onerosa. Viene eseguita ciclicamente solo se i livelli di priorità superiore sono in stato "IDLE", garantendo che l'utente riceva informazioni sugli oggetti solo quando non vi sono pericoli immediati.

4.2 Protocolli di Comunicazione e Integrazione Moduli

- **I2C (Inter-Integrated Circuit):** Utilizzato per il dialogo con i sensori ToF a una frequenza di 400kHz.
- **UART (Universal Asynchronous Receiver-Transmitter):** Dedicato al DFPlayer Mini a 9600 baud per la riproduzione dei messaggi vocali.
- **ESP-NOW:** Protocollo wireless a 2.4GHz a bassa latenza (<10 ms) per il link Occhiali-Bastone.

4.3 Gestione della Memoria e Elaborazione Immagini (GrayScale)

L'integrazione di un modello di Deep Learning su un microcontrollore richiede un'ottimizzazione estrema della memoria.

- **Buffer Immagine:** La camera OV2640 cattura frame allocati nella **PSRAM**.
- **Conversione GrayScale:** L'immagine viene convertita da RGB565 a scala di grigi per ridurre l'uso della RAM del 66% e dimezzare i tempi di calcolo della rete neurale.
- **Cropping & Scaling:** Ritaglio e riscalatura a 96x96 pixel per il modello FOMO.

4.4 Implementazione Software: OchoStick (Trasmettitore)

L'unità installata sul bastone monitora il terreno e invia pacchetti rapidi agli occhiali in caso di pericolo.

```
// --- CODICE OCHOSTICK (ESP32 DevKit V1) ---
#include <esp_now.h>
```

```
#include <WiFi.h>
#include "Adafruit_VL53L0X.h"

// Indirizzo MAC degli OchoGlass (da personalizzare)
uint8_t broadcastAddress[] = {0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX};
Adafruit_VL53L0X lox = Adafruit_VL53L0X();

void setup() {
  WiFi.mode(WIFI_STA); // Modalità Station per ESP-NOW
  if (esp_now_init() != ESP_OK) return; // Inizializzazione protocollo

  // Registrazione del destinatario (OchoGlass)
  esp_now_peer_info_t peerInfo;
  memcpy(peerInfo.peer_addr, broadcastAddress, 6);
  peerInfo.channel = 0;
  peerInfo.encrypt = false;
  esp_now_add_peer(&peerInfo);

  lox.begin(); // Avvio sensore ToF
}

void loop() {
  VL53L0X_RangingMeasurementData_t measure;
  lox.rangingTest(&measure, false);

  // Se rileva un ostacolo basso o un gradino (< 30cm)
  if (measure.RangeStatus != 4 && measure.RangeMilliMeter < 300) {
    uint8_t alertSignal = 1; // Messaggio di allerta
    esp_now_send(broadcastAddress, &alertSignal, sizeof(alertSignal));
    delay(1000); // Evita l'invio compulsivo di messaggi
  }
}
```

4.5 Implementazione Software: OchoGlass (Ricevitore e IA)

Il codice principale gestisce l'integrazione tra i dati wireless, i sensori locali e l'inferenza Edge Impulse.

```
// --- CODICE OCHOGLASS (XIAO ESP32-S3) ---
#include <esp_now.h>
#include "DFRobotDFPlayerMini.h"
#include "ei_run_classifier.h" // Libreria esportata da Edge Impulse
volatile bool stickAlert = false; // Flag per allerta dal bastone

// Callback eseguita alla ricezione di dati dal bastone
void OnDataRecv(const uint8_t * mac, const uint8_t *incomingData, int len) {
  if (incomingData[0] == 1) stickAlert = true;
}

void loop() {
  // 1. Controllo immediato allerta bastone (ESP-NOW)
  if (stickAlert) {
```

```
    myDFPlayer.play(1); // "Attenzione: ostacolo al suolo"
    stickAlert = false;
    return;
  }

  // 2. Controllo Sensore ToF Locale (VL53L0X)
  if (readLocalLaser() < 400) {
    myDFPlayer.play(2); // "Ostacolo frontale ravvicinato"
    return;
  }

  // 3. Esecuzione Inferenza IA (Solo se la strada è libera)
  ei::signal_t signal;
  signal.total_length = EI_CLASSIFIER_INPUT_WIDTH * EI_CLASSIFIER_INPUT_HEIGHT;
  signal.get_data = &raw_feature_get_data; // Prende i dati della camera (GrayScale)
  ei_impulse_result_t result = { 0 };
  EI_IMPULSE_ERROR res = run_classifier(&signal, &result, false);

  // Analisi dei risultati (FOMO)
  for (size_t ix = 0; ix < result.bounding_boxes_count; ix++) {
    auto bb = result.bounding_boxes[ix];
    if (bb.value > 0.8) { // Se confidenza > 80%
      if (strcmp(bb.label, "Sedia") == 0) myDFPlayer.play(3);
      if (strcmp(bb.label, "Porta") == 0) myDFPlayer.play(4);
    }
  }
}
```

5. Il Processo di Machine Learning (Edge AI)

L'implementazione dell'intelligenza artificiale su un microcontrollore richiede un approccio metodologico rigoroso, noto come **TinyML**. Il cuore del riconoscimento visivo di OchoGlass è stato sviluppato tramite la piattaforma **Edge Impulse**, seguendo un flusso di lavoro che ottimizza le reti neurali per l'esecuzione "at the edge" (direttamente sull'hardware, senza cloud).

5.1 Data Acquisition e Dataset Diversity

La qualità di un modello IA dipende direttamente dai dati su cui viene addestrato.

- **Campionamento Reale:** Le immagini non sono state scaricate da internet, ma acquisite direttamente tramite la fotocamera del telefono connesso direttamente al cloud di Edge Impulse.
- **Classi di Riconoscimento:** Abbiamo definito tre classi principali: `Sedia`, `Persona`, `Porta`, oltre a una classe `Background` (ambiente vuoto) per ridurre i falsi positivi.
- **Variabilità:** Per ogni oggetto sono state scattate oltre 100 foto variando angolazione, distanza e condizioni di illuminazione (luce artificiale calda/fredda e luce naturale).

5.2 Labelling e Algoritmo FOMO

A differenza dei modelli standard di *Object Detection* (come YOLO) che sono troppo pesanti per un microcontrollore, abbiamo utilizzato **FOMO (Faster Objects, More Objects)**.

- **Centroidi vs Bounding Boxes:** FOMO non cerca di disegnare un rettangolo perfetto attorno all'oggetto, ma ne identifica il centroide. Questo semplifica drasticamente il calcolo matematico, trasformando il problema della rilevazione in una griglia di classificazione.
- **Efficienza:** FOMO è fino a 30 volte più veloce di MobileNet SSD, permettendo alla XIAO di analizzare l'ambiente in tempo quasi reale.

5.3 Impulse Design e Pre-processing

L' "Impulso" è la pipeline di elaborazione dei dati:

1. **Image Data:** Le immagini originali vengono ridimensionate a **96x96 pixel**.
2. **Color Depth (GrayScale):** Come specificato precedentemente, abbiamo forzato la conversione in scala di grigi. Poiché le sedie o le porte sono riconoscibili principalmente dalla loro forma (bordi e contrasti) piuttosto che dal colore, eliminare i canali R, G, B ha permesso di risparmiare memoria preziosa senza degradare l'accuratezza.
3. **Image Blocks:** Estraggono le caratteristiche salienti dell'immagine riducendo la complessità spaziale.

5.4 Training e Ottimizzazione (Quantizzazione)

Durante la fase di addestramento sulla rete neurale **MobileNetV2 0.35**, abbiamo monitorato costantemente la *Loss Curve* per evitare l'overfitting (ovvero quando l'IA "impara a memoria" le foto del dataset ma non sa riconoscere nuovi oggetti).

- **Quantizzazione a 8 bit (INT8):** Dopo il training, il modello è stato convertito da pesi in virgola mobile (float32) a numeri interi a 8 bit. Questo processo riduce la dimensione del modello di 4 volte e accelera l'inferenza sulla CPU ESP32-S3, che è ottimizzata per operazioni su interi.

5.5 Deployment e Integrazione Firmware

Una volta ottenuta una precisione accettabile nel test set, il modello è stato esportato come **Libreria C++ Arduino**. L'integrazione nel codice finale permette alla XIAO di caricare i tensori nella **PSRAM** all'avvio e di chiamare la funzione `run_classifier()` ogni volta che il sensore laser conferma che la strada è libera, chiudendo il cerchio tra percezione sensoriale e intelligenza semantica.

6. Conclusioni e Analisi Critica

Il progetto **OchoGlass & OchoStick** ha dimostrato la fattibilità dell'integrazione di sistemi di Intelligenza Artificiale e telemetria laser in un dispositivo indossabile a basso costo. Tuttavia, come ogni prototipo di ricerca, l'analisi dei risultati mette in luce sia i traguardi raggiunti che i margini di miglioramento necessari per un'eventuale produzione su larga scala.

6.1 Valutazione dei Risultati

Il sistema è riuscito a soddisfare i requisiti primari di sicurezza e informazione:

- **Affidabilità del Laser:** Il sensore ToF si è rivelato estremamente preciso nel rilevamento degli ostacoli frontali, garantendo una risposta immediata che compensa i tempi di elaborazione più lenti dell'IA.
- **Accuratezza dell'IA:** L'uso della scala di grigi e del modello FOMO ha permesso di ottenere un riconoscimento abbastanza stabile delle classi addestrate (sedie, persone, porte).
- **Sincronizzazione Wireless:** Il protocollo ESP-NOW ha garantito che gli avvisi provenienti dal bastone (OchoStick) arrivassero agli occhiali senza ritardi percepibili, creando un ecosistema integrato.

6.2 Limiti Tecnici Rilevati

Durante la fase di test, sono state identificate alcune criticità strutturali:

- **Gestione Energetica:** Il consumo combinato della camera, del processore S3 a pieno carico e del modulo audio è elevato. Con le attuali batterie LiPo, l'autonomia è limitata, rendendo necessaria una gestione più aggressiva del risparmio energetico.
- **Latenza della Catena Audio:** Il passaggio tra il rilevamento dell'oggetto e la riproduzione del file MP3 dal DFPlayer Mini presenta un ritardo totale di circa 500-700ms. In contesti urbani dinamici, questo ritardo potrebbe risultare critico.
- **Sensibilità Ambientale:** I sensori ToF basati su infrarossi mostrano limiti in presenza di superfici molto scure (che assorbono il segnale) o in condizioni di luce solare zenitale che interferisce con il raggio laser.

6.3 Prospettive Future e Sviluppi

Il percorso evolutivo del progetto si orienta verso tre direzioni principali:

1. **Feedback Aptico:** Integrare piccoli motori a vibrazione (simili a quelli degli smartphone) nelle stanghette degli occhiali per fornire indicazioni direzionali silenziose e immediate.
2. **Miniaturizzazione (Custom PCB):** Passare dalla breadboard a un circuito stampato (PCB) progettato su misura. Questo ridurrebbe drasticamente il volume e il peso del

dispositivo, permettendo di integrare i componenti direttamente all'interno di una montatura ergonomica stampata in 3D.

3. **Integrazione Large Language Models (LLM):** Con l'evoluzione dell'Edge AI, l'obiettivo è permettere all'utente di interagire vocalmente con gli occhiali per chiedere dettagli sul contesto (es. *"Quante persone ci sono nella stanza?"*), passando da un sistema di allerta a un vero assistente cognitivo.