Fondazione ITS Academy per la Mobilità Sostenibile Aerospazio/Meccatronica e Servizi alle Imprese - Piemonte

Corso ITS 15ª Edizione

A.F. 2025-2026 - Torino

Tecnico Superiore per l'Automazione e la Robotica Industriale

Embedded System e AI per l'Automazione – EMBT03

Data:	24/01/2026	Rev: 00
File:	RELAZIONE_ARDUINO_NATALE.pdf	
Redatto da:	Roberto Andrea Marchetti, Yassine Sakhoune, Michele Cabutto, Davide Sargiacomo	

INDICE

1 OBBIETTIVO....3

2 COMPONENTI E STRUMENTI....3

3 PROCEDIMENTO....3

3.1 Ideazione....4

3.2 Assemblaggio....4

3.3 Realizzazione delle animazioni....5

3.3.1 Sviluppo dei frame....5

3.3.2 Implementazione nel codice....5

3.4 Realizzazione del programma....6

3.4.1 Campionamento....6

3.4.2 Definizione dei livelli del volume....7

3.4.3 Selezione e visualizzazione frame....7

4 CONCLUSIONE....8

GLOSSARIO....9

Visualizzatore Musicale

Rappresentare le musiche natalizie su una matrice LED

1 Obbiettivo

Lo scopo del progetto è la realizzazione di un dispositivo in grado di fornire un feedback visivo in tempo reale basato sull'intensità sonora, utilizzando lo stile grafico della "Pixel Art" a tema natalizio.

Il sistema si basa sull'utilizzo della scheda Arduino Uno R4 WiFi (di seguito Arduino R4 o Arduino), sfruttando la sua matrice LED (Light Emitting Diode) integrata (8x12) per l'output visivo. L'input audio viene acquisito tramite un microfono basato sull'amplificatore operazionale (OpAmp) MAX4466.

Il risultato finale deve essere un'animazione che reagisce dinamicamente al volume della musica (nello specifico brani natalizi come "Jingle Bells"): al variare dei decibel (dB) percepiti, l'immagine sulla matrice deve evolversi, creando un effetto di "VU-meter" grafico.

2 Componenti e Strumenti

Hardware:

- Arduino Uno R4 WiFi,
- sensore MAX4466,
- cavi jumper,
- cavo USB-C.

Software:

- Arduino IDE,
- libreria Arduino_LED_Matrix.h esterna.

3 Procedimento

Data la necessità di ottimizzare i tempi di sviluppo, il gruppo di lavoro (composto da 4 membri) ha adottato una metodologia parallela, dividendo il progetto in due macro-aree:

- **Sottogruppo Animazione e Design**: Responsabile dell'ideazione grafica, della creazione dei frame e della logica visiva.
- **Sottogruppo Software e Hardware:** Responsabile della gestione del segnale microfonico, del cablaggio e del codice di controllo.

3.1 Ideazione

La prima fase del lavoro ha riguardato il brainstorming per individuare un soggetto grafico che fosse sia esteticamente gradevole sia funzionale alla rappresentazione di una scala di valori (volume basso - alto).

Sono state valutate le seguenti proposte:

- una stella che rotante a tempo di musica,
- una stella pulsante (ingrandimento/rimpicciolimento),
- una pallina di Natale che si riempie progressivamente,
- un albero di Natale statico a cui si aggiungono addobbi all'aumentare dell'intensità sonora,
- un albero di Natale con riempimento progressivo.

La scelta è ricaduta sull'albero di Natale con riempimento progressivo. Questa opzione è stata ritenuta la più efficace poiché:

1. offre un feedback visivo intuitivo (simile alle barre del volume dei vecchi stereo),
2. si adatta perfettamente alla griglia della matrice LED,
3. era l'opzione realizzabile nel tempo a disposizione, garantendo un risultato pulito e riconoscibile.

3.2 Assemblaggio

Per assemblare l'OpAmp all'Arduino, abbiamo utilizzato i cavi jumper collegandoli ai seguenti pin:

- 5V per alimentare l'OpAmp,
- `GND` per chiudere il circuito,
- `A0` (pin analogico) per permettere all'Arduino di ricevere il segnale analogico prodotto dall'OpAmp.

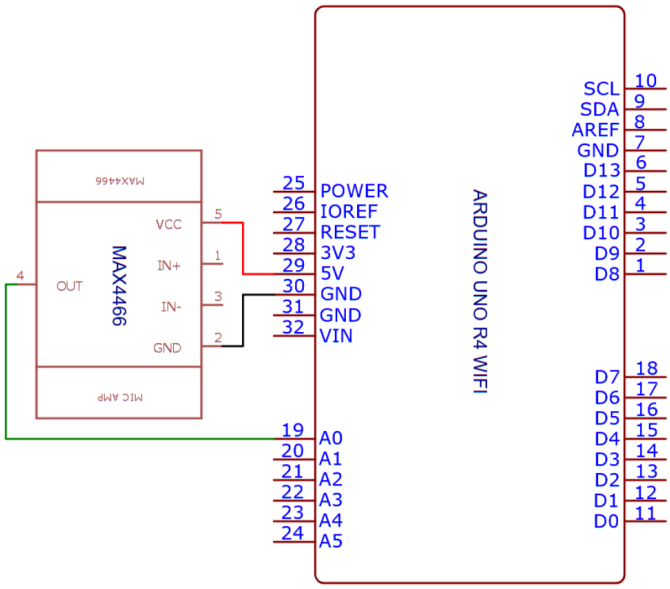

Immagine 1: Schema di collegamento Arduino - OpAmp

3.3 Realizzazione delle animazioni

Il lavoro si è concentrato sulla traduzione dell'idea in dati interpretabili da Arduino. La matrice LED dell'Arduino R4 è gestita come una griglia di dimensioni `[8][12]` (8 righe per 12 colonne, gestite come byte).

3.3.1 Sviluppo dei frame

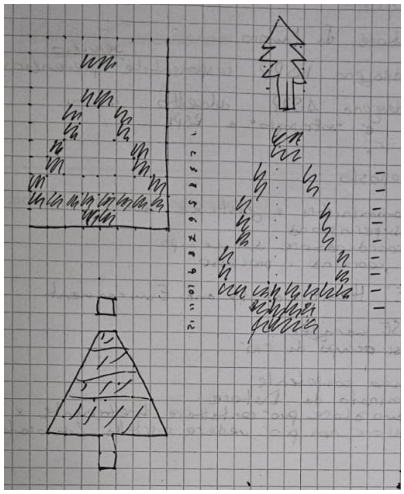

Immagine 2: Schizzi su carta del design dell'albero

Dopo aver realizzato vari schizzi su carta per definire la forma ottimale dell'albero, si è proceduto alla digitalizzazione dei disegni. È stata creata una sequenza di 10 frame totali (numerati da 0 a 9):

- Frame 0 (stato di quiete): visualizza solo il contorno dell'albero. Questo stato corrisponde all'assenza di suono o a un volume molto basso.
- Frame 1-9 (stato attivo): a ogni incremento del frame corrisponde l'aggiunta di una riga di pixel interna o di un livello di riempimento, fino ad arrivare all'albero completamente illuminato (Frame 9), che corrisponde al picco massimo di intensità sonora.

Questa suddivisione in 10 step ha permesso di comunicare al sottogruppo di programmazione un dato preciso: il range di volume rilevato dal microfono deve essere "mappato" in 10 fasce distinte per richiamare l'immagine corretta.

3.3.2 Implementazione nel codice

I disegni sono stati convertiti in matrici di byte (dove 1 rappresenta un LED acceso e 0 un LED spento) e inseriti all'interno di un file header dedicato, denominato Animazioni_Albero.h, per mantenere il codice principale pulito e ordinato.

Di seguito è riportato un esempio della struttura dati utilizzata per il "Frame 0" (contorno dell'albero vuoto):

```
byte disegno0[8][12] = {
  {0,0,0,0,0,0,0,1,1,1,0,0},
  {0,0,0,0,1,1,1,0,0,1,0,0},
  {0,0,1,1,0,0,0,0,0,1,1,1},
  {1,1,0,0,0,0,0,0,0,0,0,1},
  {1,1,0,0,0,0,0,0,0,0,0,1},
  {0,0,1,1,0,0,0,0,0,1,1,1},
  {0,0,0,0,1,1,1,0,0,1,0,0},
  {0,0,0,0,0,0,0,1,1,1,0,0},
};
```

Il file Animazioni_Albero.h contiene quindi l'array completo di tutti e 10 i frame, pronti per essere richiamati dal ciclo principale del programma in base ai valori letti dal microfono.

```
#include "Animazione_Albero.h"
```

3.4 Realizzazione del programma

Per sviluppare il programma e caricarlo sull'Arduino R4 (collegato al PC tramite cavo USB-C) è stato utilizzato l'ambiente di sviluppo integrato (IDE) fornito dall'azienda stessa, che permette la comunicazione seriale e il monitoraggio dei segnali in tempo reale, utile per il debugging.

3.4.1 Campionamento

All'interno di una finestra temporale di 50 ms, abbiamo rilevato i valori massimi e minimi del segnale elettrico per calcolare l'ampiezza picco-picco (P2P), inoltre abbiamo inserito delle variabili per calibrare le intercettazioni ('`sogliaSilenzio`' e '`sensibilita`'), che servono per filtrare il rumore di fondo ambientale e fare in modo che i LED reagiscano ai suoni effettivamente significativi.

Definizione delle variabili:

```
const int sampleWindow = 50;

// Parametri di calibrazione
int sogliaSilenzio = 50;     // valore minimo per ignorare il rumore di fondo
int sensibilita = 200;       // soglia per raggiungere il livello massimo
int reference = 100;         // offset rumore
```

Calcolo del P2P all'interno della finestra di campionamento:

```
// Campionamento
while (millis() - startMillis < sampleWindow) {
  int sample = analogRead(micPin);
  if (sample < 1024) {
    if (sample > signalMax) signalMax = sample;
    else if (sample < signalMin) signalMin = sample;
  }
}

// Calcolo ampiezza P2P
int diff = (int)signalMax - (int)signalMin - reference;
if (diff < 0) diff = 0;
unsigned int peakToPeak = (unsigned int)diff;

if(peakToPeak > 5000) peakToPeak = 0;

// Intensità luminosa
int intensitaLed = map(peakToPeak, sogliaSilenzio, sensibilita, 0, 255);
intensitaLed = constrain(intensitaLed, 0, 255);
analogWrite(ledPin, intensitaLed);
```

3.4.2 Definizione dei livelli del volume

Nella seconda parte del codice, abbiamo trasformato il valore numerico del volume in un'immagine dinamica. Utilizzando la funzione '`map`', il volume viene calcolato a partire da un indice 0 fino a 9, il quale corrisponde ai dieci livelli di animazione che ci eravamo preimpostati.

```
    // Mappatura del valore su 10 LIVELLI (da 0 a 9)
    // Invece di mappare su 96 LED, mappiamo sull'indice dell'array di disegni

    int indiceLivello = 0;

    #ifdef LOGARITMIC
      if (peakToPeak > 2) {
        logVal = log10((float)peakToPeak);
        indiceLivello = map((logVal - 2.2) * 3000, 0, 600, 0, 9);
      }
    #else
      if (peakToPeak > sogliaSilenzio) {
          indiceLivello = map(peakToPeak, sogliaSilenzio, sensibilita, 1, 9);
      } else {
          indiceLivello = 0;
      }
    #endif

      indiceLivello = constrain(indiceLivello, 0, 9);

      aggiornaMatrice(indiceLivello);
```

3.4.3 Selezione e visualizzazione frame

Attraverso la funzione '`aggiornaMatrice`', il programma seleziona il disegno bitmap corrispondente e lo copia all'interno di un array.

```
    void aggiornaMatrice(int livelloSelezionato) {
    // Scansione per righe e colonne
      for (int r = 0; r < 8; r++) {
        for (int c = 0; c < 12; c++) {
          // pixel dal disegno salvato in Animazione_Albero.h al frame corrente
          frame[r][c] = livelliAnimazione[livelloSelezionato][r][c];
        }
      }
    }
```

Infine il comando '`matrix.renderBitmap`' invia i dati alla matrice 8x12 creando l'immagine definitiva in cui l'albero si riempie a in modo proporzionale alle onde sonore che intercetta il sensore.

```
    // Invio del frame alla matrice
    matrix.renderBitmap(frame, 8, 12);
```

4 Conclusione

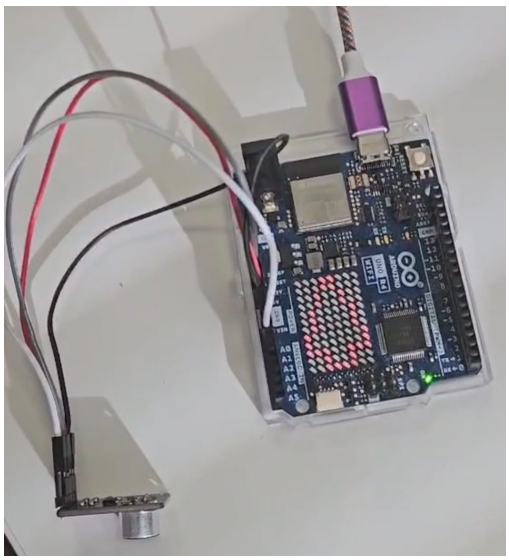

Immagine 3: Visualizzatore completo e funzionante

Il progetto ha portato alla realizzazione di un visualizzatore musicale pienamente funzionante e rispondente agli obiettivi prefissati. L'integrazione tra l'hardware (Arduino R4 e sensore MAX4466) e il software di gestione ha permesso di ottenere un feedback visivo fluido e dinamico, capace di reagire con precisione alle variazioni di intensità sonora dei brani natalizi.

La scelta dell'albero di Natale con riempimento progressivo è risultata vincente, poiché ha permesso di mappare i 10 livelli di intensità in modo leggibile sulla matrice 12x8.

La finestra di campionamento di 50 ms si è dimostrata un compromesso ottimale tra velocità di aggiornamento e stabilità del segnale.

L'implementazione di parametri di calibrazione ha consentito di isolare efficacemente il rumore di fondo, garantendo che l'animazione si attivasse solo in presenza di musica.

In conclusione, l'esperienza ha confermato l'efficacia della metodologia di lavoro parallela adottata dal gruppo, permettendo di consegnare un prototipo completo e testato nei tempi previsti.

Glossario

- **Arduino Uno R4 WiFi**: nel testo è richiamato anche come Arduino R4 o Arduino.
- **Debug / Debugging**: processo sistematico di individuazione, analisi e rimozione di errori (bug) all'interno di un software o di un sistema elettronico.
- **IDE (Integrated Development Environment / Ambiente di Sviluppo Integrato)**: software che fornisce strumenti completi per lo sviluppo di programmi (compilatore, editor di testo, debugger). Nel contesto di questo progetto, ci si riferisce tipicamente all'Arduino IDE, utilizzato per scrivere, compilare e caricare il codice sorgente (sketch) sulla scheda Arduino.
- **LED (Light Emitting Diode)**: componente optoelettronico a semiconduttore che emette luce fredda quando attraversato da corrente elettrica.
- **OpAmp (Operational Amplifier / Amplificatore Operazionale)**: circuito integrato analogico ad alto guadagno, utilizzato per elaborare segnali elettrici. Nel progetto viene impiegato per elevare l'ampiezza del segnale in uscita dal microfono (dell'ordine dei millivolt) a un livello leggibile dai pin analogici di Arduino.
- **P2P (peak-to-peak)**: in ambito elettronico e fisico, indica il valore picco-picco di una grandezza variabile (come una tensione elettrica o un'onda sonora). Rappresenta la differenza assoluta tra il valore massimo positivo e il valore minimo negativo di una forma d'onda.