

## Sistemi di Elettronica Digitale – A.A. 2022-23

# Relazione di Progetto

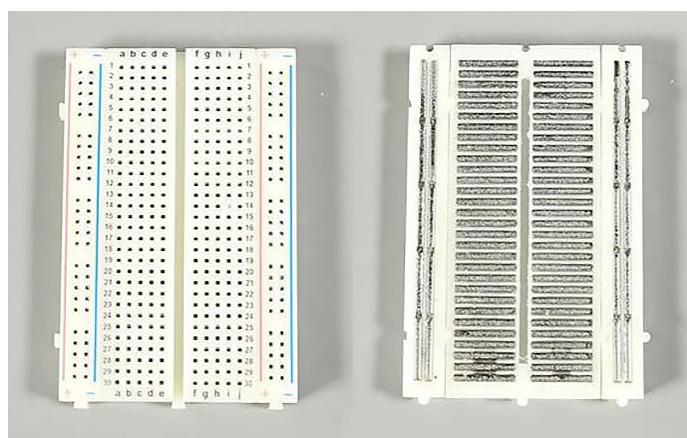
**A**RDUINO è nato nel 2005 come progetto open source, molto prima del progetto Raspberry, a Ivrea. In Italia si stava sviluppando l'elettronica di sistema, che non sviluppa componenti, si occupa di sistemi specifici, basati su microcontrollori, che semplifica la professione dell'elettronico, perché ha già dentro tutto, e sposta il target dall'hardware al firmware. L'idea di Arduino, seppur nasca nell'elettronica italiana, è sociale. Lo scopo è la community e la cultura, cioè la diffusione di persone che permetta di creare elettronica di sistema, ma con pochissimi prerequisiti elettronici, quindi più veloce ed agevole, ma soprattutto facile e alla portata di tutti. Il programma, detto *sketch*, è articolato in una parte di inizializzazione *setup()* e una parte *loop()* che viene eseguita ciclicamente. Il progetto Arduino è stato creato per permettere ad artisti creativi, designer e progettisti di prototipare e progettare le loro idee senza dover disporre di molte conoscenze tecniche. Questa relazione tratta la possibilità di usare Arduino per riprodurre musica e giochi di luce coordinati alla musica, il tutto grazie a pochi componenti.

## I componenti utilizzati

**A**RDUNIO si serve di componenti elettronici che hanno tutti un specifico ruolo e funzionamento. In questo progetto, ho utilizzato alcuni dei componenti forniti dal kit, che ora andrò a spiegare.

Ponti: piccoli cavi elettrici di varie lunghezze e colori.

Breadboard<sup>1</sup>: permette di testare e costruire il circuito. Essa consente il passaggio di elettricità, poiché dotata di strisce metalliche conduttrive parallele sotto uno strato di plastica dotato di fori in cui incastrare i vari elementi. La basetta qui in dotazione è contraddistinta da due strisce laterali, rispettivamente rossa per +, ossia il polo positivo, e blu per -, che rappresenta il polo negativo, per connettere, rispettivamente, l'alimentazione e la massa.



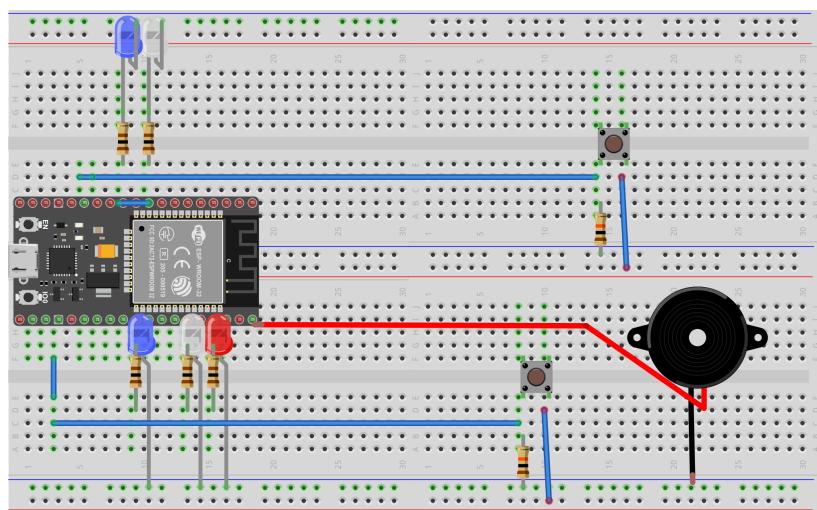
<sup>1</sup> Immagine presa da <https://www.digikey.com/en/maker/projects/learn-to-breadboard/43bde9f32dad493182278f0441466a6e>. Specifico che, data la forma dell'ESP32, era necessario utilizzare due breadboard.

Diodo: dispositivo che permette il passaggio di corrente in una sola direzione. In questo progetto, ho usato cinque diodi LED, che sono emettitori di luce. I diodi LED lavorano in polarizzazione diretta, in quanto emettono fotoni a seguito dell'energia rilasciata dalla ricombinazione di una coppia p-n. Quando l'elettrone fa salti quantici, tipicamente scambia energia con l'ambiente — ci sono vari modi, ma anche tramite fotoni grazie all'effetto fotoelettrico. Il diodo led, quindi, ha una tensione diversa in base al colore. I LED, per emettere fotoni, hanno bisogno di una certa energia e la luce che emettono dipende, ma non in modo lineare, dalla corrente che li attraversa. Si parla, infatti, di illuminazione led a bassa potenza proprio perché non è lineare. Quindi, con correnti relativamente piccole, il led riesce ad emettere un quantitativo significativo di fotoni. Si accendono con circa 10mA, ciò comporta elevata efficienza. Il diodo LED ha una tensione di polarizzazione diretta che varia a seconda della lunghezza d'onda della luce che emettono, ed emettono tanta più luce quanta più corrente li attraversa.

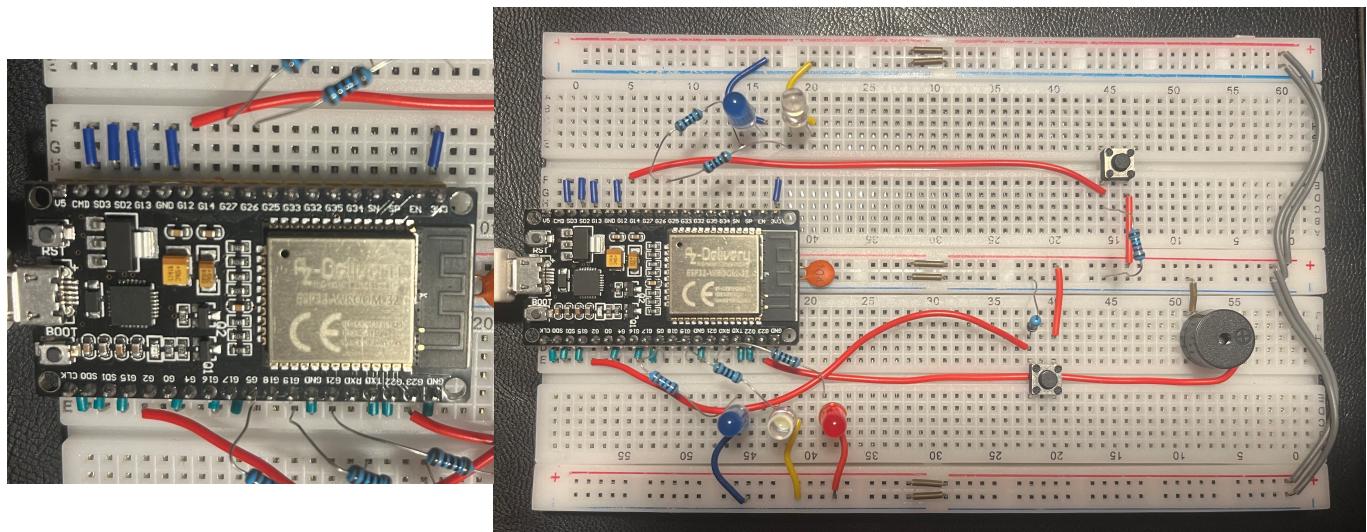
Buzzer: è uno dispositivo elettronico che permette di emettere un suono, un «buzz», se correttamente alimentato. In questo progetto, uso un buzzer attivo, cioè un dispositivo che, se alimentato con una tensione opportuna, riproduce un tono ad una frequenza pre-impostata.

Resistenza: dispositivo che oppone resistenza al passaggio di corrente, cioè limita la corrente passante da un componente elettronico. Nel progetto, ne ho utilizzate di grandezze differenti, in base alla necessità. Infatti, per i diodi, ho utilizzato resistenze da  $100\Omega$ , poiché è assolutamente necessario interporre in serie all'alimentazione una resistenza che limiti il passaggio della corrente, in quanto il valore ottimale per il corretto funzionamento del led è sopra i 10mA, per quanto detto nel paragrafo precedente. Correnti superiori a 25mA romperebbero il LED, mentre con correnti non inferiori a 3mA si avverrebbe un corretto funzionamento, tuttavia con luminosità minore. Per avere un'alta luminosità, ho deciso di usare  $100\Omega$ , consapevole che in base al colore del LED cambia la tensione necessaria — il blu è meno efficiente del rosso, in quanto si parla di efficienza se, a parità di tensione, uno assorbe più corrente dell'altro, mentre il bianco è più efficiente del blu. Per quanto riguarda il buzzer, invece, per limitare la corrente è sempre meglio avere una resistenza da  $100\Omega$  in serie al buzzer. Infine, per i bottoni ho usato una resistenza da  $10k\Omega$ , che serve per portare il pin in INPUT che ha un'alta impedenza, non deve mai essere lasciato flottante, ad uno stato ben preciso, in questo caso pull-down = LOW, quando il bottone non è premuto — quando si preme il bottone, deve iniziare a funzionare la parte di programma che gestisce un determinato brano e i led ad esso associati.

Cavo micro-USB: alimenta il circuito e trasferisce il codice dal computer ad Arduino.



Schema collegamenti — in senso orario dall'alto i pin sono 14, 26, 25, 23, 21, 19, 5, 2.



## Obiettivo

L'obiettivo principale di questo progetto è quello di comprendere le basi della conversione da nota musicale a frequenza. Infatti, l'udito riesce a produrre delle sensazioni sonore solamente per un intervallo di frequenze compreso da 20Hz, sotto questa soglia sono infrasuoni, fino a 20kHz, sopra sono ultrasuoni: quest'intervallo di frequenze determina la banda udibile. Tuttavia, quest'intervallo con l'avanzare degli anni tende a restringersi da entrambi i limiti.

In questo caso, i brani scelti sono due: la Marsigliese, inno ufficiale della Francia, e l'inno dell'Argentina. Chiaramente, i compositori utilizzano le melodie attraverso le note, ma che cosa sono le note? Sono le varie altezze dei suoni, ma quindi qual è la relazione fra le note e le frequenze, che servono per poter far suonare al buzzer i due brani? La frequenza, usata come standard internazionale di riferimento, corrispondente al La del corista indicato con la dizione anglosassone A4, che è fissata a 440 Hz. In base a questa convenzione, per calcolare la frequenza delle altre note, considerando il *temperamento equabile*<sup>2</sup>, si utilizza la seguente formula

$$f = 2^{\frac{N}{12}} \cdot f_{\text{rif}}$$

dove  $f_{\text{rif}}$  sono i 440Hz della nota A4, mentre  $N$  il numero di semitonni di distanza dalla nota di riferimento.

Note	ottave									
	0	1	2	3	4	5	6	7	8	9
Do	16,35	32,70	65,41	130,8	261,6	523,3	1047	2093	4186	8372
Do#-Reb	17,32	34,65	69,30	138,6	277,2	554,4	1109	2217	4435	8870
Re	18,35	36,71	73,42	146,8	293,7	587,3	1175	2349	4699	9397
Re#-Mib	19,45	38,89	77,78	155,6	311,1	622,3	1245	2489	4978	9956
Mi	20,60	41,20	82,41	164,8	329,6	659,3	1319	2637	5274	10548
Fa	21,83	43,65	87,31	174,6	349,2	698,5	1397	2794	5588	11175
Fa#-Solv	23,12	46,25	92,50	185,0	370,0	740,0	1480	2960	5920	11840
Sol	24,50	49,00	98,00	196,0	392,0	784,0	1568	3136	6272	12544
Sol#-Lab	25,96	51,91	103,8	207,7	415,3	830,6	1661	3322	6645	13290
La	27,50	55,00	110,0	220,0	440,0	880,0	1760	3520	7040	14080
La#-Sib	29,14	58,27	116,5	233,1	466,2	932,3	1865	3729	7459	14917
Si	30,87	61,74	123,5	246,9	493,9	987,8	1976	3951	7902	15804

Frequenze corrispondenti a tutte le note del temperamento equabile

<sup>2</sup> Il temperamento equabile è il sistema musicale per la costruzione di una scala musicale fondato sulla suddivisione dell'ottava in intervalli tra di loro uguali.  
<https://musyance.com/che-cos-e-il-sistema-temperato-equabile-la-storia-la-sua-diffusione>

Per poter far partire un inno piuttosto che un altro, ci sono due bottoni, ai quali è stato associato un singolo inno e, una volta terminato, se premuto nuovamente farà partire un'altra volta l'inno di quel bottone. In aggiunta, per la Francia ci sono tre diodi LED — rosso, bianco e blu — e per l'Argentina due — blu e bianco. A ciascuna nota del singolo inno è associata una combinazione luminosa dei LED, che possono essere accesi oppure spenti. Una volta terminato l'inno scelto, i LED associati si spegneranno, per poi ripartire quando si vorrà nuovamente riprodurre l'inno in questione.

## Il codice

**A** seguire, riporto il codice usato per programmare la scheda AZ-delivery ESP32 Dev Kit C V2, basata sul modulo ESP32-WROOM. Le due parti principali sono `setup()`, dove ho inizializzato i singoli componenti, e `loop()`, nella quale sono indicati le funzioni che si occupano di riprodurre, rispettivamente, l'inno francese ed argentino, mentre all'interno di questi ultimi ci sono le funzioni che si occupano dell'illuminazione dei LED, attraverso l'utilizzo di switch-case. Le funzioni all'interno di `loop()`, ma anche quelle annidate nelle funzioni in `loop()`, continueranno a funzionare finché l'ESP32 è alimentata.

Le variabili sono aree di memoria di Arduino dove si possono registrare dati. Le variabili possono essere cambiate tutte le volte che vogliamo: le uso per indicare i pin a cui ho collegato le varie componenti e gli array per note e tempi degli inni. Le costanti, invece, le definisco come macro con `#define`, nello specifico sono le frequenze delle note e i tempi delle pause.

```
/**
 * Progetto di Sistemi di Elettronica Digitale – A.A. 2022–23
 * Martina Contestabile
 * Matricola 731044
 */

#include <ESP32Servo.h> // fondamentale per poter usare la funzione tone()

/*
 * Frequenza delle note, notazione Anglosassone.
 */
#define NOTE_B0 31
#define NOTE_C1 33
#define NOTE_CS1 35
#define NOTE_D1 37
#define NOTE_DS1 39
#define NOTE_E1 41
#define NOTE_F1 44
#define NOTE_FS1 46
#define NOTE_G1 49
#define NOTE_GS1 52
#define NOTE_A1 55
#define NOTE_AS1 58
#define NOTE_B1 62
#define NOTE_C2 65
#define NOTE_CS2 69
#define NOTE_D2 73
#define NOTE_DS2 78
#define NOTE_E2 82
#define NOTE_F2 87
#define NOTE_FS2 93
```

```
#define NOTE_G2 98
#define NOTE_GS2 104
#define NOTE_A2 110
#define NOTE_AS2 117
#define NOTE_B2 123
#define NOTE_C3 131
#define NOTE_CS3 139
#define NOTE_D3 147
#define NOTE_DS3 156
#define NOTE_E3 165
#define NOTE_F3 175
#define NOTE_FS3 185
#define NOTE_G3 196
#define NOTE_GS3 208
#define NOTE_A3 220
#define NOTE_AS3 233
#define NOTE_B3 247
#define NOTE_C4 262
#define NOTE_CS4 277
#define NOTE_D4 294
#define NOTE_DS4 311
#define NOTE_E4 330
#define NOTE_F4 349
#define NOTE_FS4 370
#define NOTE_G4 392
#define NOTE_GS4 415
#define NOTE_A4 440
#define NOTE_AS4 466
#define NOTE_B4 494
#define NOTE_C5 523
#define NOTE_CS5 554
#define NOTE_D5 587
#define NOTE_DS5 622
#define NOTE_E5 659
#define NOTE_F5 698
#define NOTE_FS5 740
#define NOTE_G5 784
#define NOTE_GS5 831
#define NOTE_A5 880
#define NOTE_AS5 932
#define NOTE_B5 988
#define NOTE_C6 1047
#define NOTE_CS6 1109
#define NOTE_D6 1175
#define NOTE_DS6 1245
#define NOTE_E6 1319
#define NOTE_F6 1397
#define NOTE_FS6 1480
#define NOTE_G6 1568
#define NOTE_GS6 1661
#define NOTE_A6 1760
#define NOTE_AS6 1865
#define NOTE_B6 1976
#define NOTE_C7 2093
#define NOTE_CS7 2217
#define NOTE_D7 2349
#define NOTE_DS7 2489
#define NOTE_E7 2637
#define NOTE_F7 2794
#define NOTE_FS7 2960
#define NOTE_G7 3136
#define NOTE_GS7 3322
#define NOTE_A7 3520
#define NOTE_AS7 3729
#define NOTE_B7 3951
#define NOTE_C8 4186
#define NOTE_CS8 4435
#define NOTE_D8 4699
#define NOTE_DS8 4978
```

```

/*
 * Tempi delle note musicali e pause, servono per dare il corretto ritmo ai brani.
 */
#define PAUSE 0
#define FOUR 4
#define TWO 2
#define THIRTEEN_EIGHTHS 13/8
#define ONE 1
#define THREE_EIGHT 8/3
#define SEVEN_QUARTER 7/4
#define HALF 2
#define A_QUARTER 4
#define SEVEN_EIGHTHS 8/7
#define ONE_1
#define THREE_HALF 3/2

/*
 * Setup dei pin assegnati sull'Arduino.
 * I buttoni sono pull-down, cioè sono settati a low.
 */
int buzzer = 23;
int buttonFr = 15;
int bFr = 5;
int wFr = 19;
int rFr = 21;

int buttonAg = 12;
int bAg = 27;
int wAg = 26;

/*
 * Array che rappresenta l'insieme delle note dell'inno francese.
 */
int marseillaise[] = {
    NOTE_D4, NOTE_D4, NOTE_D4,
    NOTE_G4, NOTE_G4, NOTE_A4, NOTE_A4,
    NOTE_D5, NOTE_B4, NOTE_G4, NOTE_G4, NOTE_B4, NOTE_G4,
    NOTE_E4, NOTE_C5, NOTE_A4, NOTE_FS4,
    NOTE_G4, PAUSE, NOTE_G4, NOTE_A4,
    NOTE_B4, NOTE_B4, NOTE_C5, NOTE_B4,
    NOTE_B4, NOTE_A4, PAUSE, NOTE_A4, NOTE_B4,
    NOTE_C5, NOTE_C5, NOTE_D5, NOTE_C5,
    NOTE_B4, PAUSE, NOTE_D5, NOTE_D5,
    NOTE_D5, NOTE_B4, NOTE_G4, NOTE_D5, NOTE_B4, NOTE_G4,
    NOTE_D4,
    PAUSE
};

/*
 * Durata di ciascuna nota dell'inno francese.
 */
int durationsFrance[] = {
    FOUR, THIRTEEN_EIGHTHS, FOUR,
    ONE, ONE, ONE, ONE,
    THIRTEEN_EIGHTHS, TWO, THIRTEEN_EIGHTHS, FOUR, THIRTEEN_EIGHTHS, FOUR,
    ONE, TWO, THIRTEEN_EIGHTHS, FOUR,
    TWO, ONE, THIRTEEN_EIGHTHS, FOUR,
    ONE, ONE, ONE, THIRTEEN_EIGHTHS, FOUR,
    ONE, ONE, ONE, THIRTEEN_EIGHTHS, FOUR,
    ONE, ONE, ONE, THIRTEEN_EIGHTHS, FOUR,
    TWO, ONE, THIRTEEN_EIGHTHS, FOUR,
    ONE, THIRTEEN_EIGHTHS, FOUR, ONE, THIRTEEN_EIGHTHS, FOUR,
    TWO,
    THREE_EIGHT
};

```

```

/*
 * Array che rappresenta l'insieme delle note dell'inno argentino.
 */
int argentinaAnthem[] = {
    NOTE_AS4, NOTE_D5, NOTE_AS4, NOTE_F5, NOTE_DS5, NOTE_C5, NOTE_A4, NOTE_A4,
    NOTE_A4, NOTE_A4, NOTE_A4, NOTE_AS4, NOTE_C4, NOTE_AS4, PAUSE, PAUSE, NOTE_F4,
    NOTE_G4, NOTE_A4, NOTE_AS4, NOTE_AS4, NOTE_D5, NOTE_C5, NOTE_AS4, NOTE_A4,
    NOTE_G4, NOTE_F4, NOTE_D4, NOTE_AS3, NOTE_D4, NOTE_F4, NOTE_DS4, NOTE_DS4,
    NOTE_G4, NOTE_F4, NOTE_G4, NOTE_A4, NOTE_AS4, NOTE_AS3, NOTE_C4, NOTE_D4,
    NOTE_D4, NOTE_D4, NOTE_D4, NOTE_F4, NOTE_DS4, NOTE_G4, NOTE_G4,
    NOTE_G4, NOTE_F4, NOTE_F4, NOTE_A4, NOTE_G4, NOTE_G4, NOTE_AS4, NOTE_CS4,
    NOTE_D4, NOTE_G4, NOTE_F4, NOTE_G4, NOTE_A4
};

/*
 * Durata di ciascuna nota dell'inno argentino.
 */
int durationsArgentina[] = {
    TWO, SEVEN_QUARTER, A_QUARTER, SEVEN_QUARTER, A_QUARTER, ONE, ONE, ONE, ONE,
    THREE_HALF, A_QUARTER, A_QUARTER, TWO, TWO, HALF, HALF, ONE, ONE,
    ONE, THREE_HALF, HALF, ONE, ONE, THREE_HALF, HALF, HALF, HALF, ONE, THREE_HALF,
    HALF, ONE, THREE_HALF, HALF, ONE, THREE_HALF, HALF, TWO, HALF, THREE_HALF, HALF,
    ONE, THREE_HALF, HALF, ONE, ONE, ONE, ONE, ONE, THREE_HALF, HALF, ONE,
    THREE_HALF, HALF, ONE, THREE_HALF, HALF, TWO, HALF, ONE, ONE
};

void setup() {
    // Imposto il buzzer come output, in quanto è deve solo riprodurre un suono, non
    // si compiono azioni sul buzzer.
    pinMode(buzzer, OUTPUT);

    // Componenti associate alla Francia.
    pinMode(buttonFr, INPUT);
    pinMode(bFr, OUTPUT);
    pinMode(wFr, OUTPUT);
    pinMode(rFr, OUTPUT);

    // Componenti associate all'Argentina.
    pinMode(buttonAg, INPUT);
    pinMode(bAg, OUTPUT);
    pinMode(wAg, OUTPUT);
}

void loop() {
    /*
     * Bisogna controllare i bottoni: se non si fa nulla su di loro, allora il
     * programma non fa nulla, resta in attesa. Invece, se si preme uno dei
     * bottoni, si fa partire il rispettivo inno. Una volta premuto il bottone,
     * questo fa partire l'inno e torna LOW, in attesa di future azioni su di esso.
     */
    int buttonFrState = digitalRead(buttonFr);
    if (buttonFrState == HIGH) {
        franceAnthem();
    }

    int buttonAgState = digitalRead(buttonAg);
    if (buttonAgState == HIGH) {
        argentinaAnthem();
    }
}

/*
 * Metodo che si occupa di gestire l'inno francese.
 */
void franceAnthem() {
    for (int thisNote = 0; thisNote < 45; thisNote++) {
        int note = marseillaise[thisNote];
        int noteDuration = 500 / durationsFrance[thisNote];
        int pauseBetweenNotes = noteDuration * 1.00;
}

```

```

ledshowFr(note);

if (note == 0) {
    while (noteDuration--) {
        delayMicroseconds(50);
    }
    continue;
} else {
    tone(buzzer, marseillaise[thisNote], noteDuration);
    delay(pauseBetweenNotes);
    noTone(buzzer);
}
}

// Terminato il brano, devo spegnere i led.
digitalWrite(bFr, LOW);
digitalWrite(wFr, LOW);
digitalWrite(rFr, LOW);
}

/*
 * Metodo che si occupa di gestire i led dell'inno francese.
 */
void ledshowFr(int currentNote) {
    switch (currentNote) {
        case NOTE_D4:
            digitalWrite(bFr, HIGH);
            digitalWrite(wFr, HIGH);
            digitalWrite(rFr, HIGH);
            break;
        case NOTE_G4:
            digitalWrite(bFr, HIGH);
            digitalWrite(wFr, LOW);
            digitalWrite(rFr, HIGH);
            break;
        case NOTE_A4:
            digitalWrite(bFr, LOW);
            digitalWrite(wFr, HIGH);
            digitalWrite(rFr, LOW);
            break;
        case NOTE_D5:
            digitalWrite(bFr, HIGH);
            digitalWrite(wFr, HIGH);
            digitalWrite(rFr, LOW);
            break;
        case NOTE_B4:
            digitalWrite(bFr, LOW);
            digitalWrite(wFr, HIGH);
            digitalWrite(rFr, HIGH);
            break;
        case NOTE_E4:
            digitalWrite(bFr, LOW);
            digitalWrite(wFr, LOW);
            digitalWrite(rFr, LOW);
            break;
        case NOTE_C5:
            digitalWrite(bFr, HIGH);
            digitalWrite(wFr, HIGH);
            digitalWrite(rFr, HIGH);
            break;
        case NOTE_FS4:
            digitalWrite(bFr, LOW);
            digitalWrite(wFr, HIGH);
            digitalWrite(rFr, LOW);
            break;
        case PAUSE:
            digitalWrite(bFr, HIGH);
            digitalWrite(wFr, LOW);
            digitalWrite(rFr, HIGH);
    }
}

```

```

        break;
    default:
        break;
    }
}

/*
 * Metodo che si occupa di gestire l'inno argentino.
 */
void argentinaAnthem() {
    for (int thisNote = 0; thisNote < 62; thisNote++) {
        int note = argentinaAnthem[thisNote];
        int noteDuration = 500 / durationsArgentina[thisNote];
        int pauseBetweenNotes = noteDuration * 1.00;
        ledshowAg(note);

        if (note == 0) {
            while (noteDuration--) {
                delayMicroseconds(5);
            }
            continue;
        } else {
            tone(buzzer, argentinaAnthem[thisNote], noteDuration);
            delay(pauseBetweenNotes);
            noTone(buzzer);
        }
    }

    digitalWrite(bAg, LOW);
    digitalWrite(wAg, LOW);
}

/*
 * Metodo che si occupa di gestire i led dell'inno argentino.
 */
void ledshowAg(int currentNote) {
    switch (currentNote) {
        case NOTE_A4:
            digitalWrite(bAg, HIGH);
            digitalWrite(wAg, HIGH);
            break;
        case NOTE_C5:
            digitalWrite(bAg, LOW);
            digitalWrite(wAg, LOW);
            break;
        case NOTE_D5:
            digitalWrite(bAg, HIGH);
            digitalWrite(wAg, LOW);
            break;
        case NOTE_B4:
            digitalWrite(bAg, LOW);
            digitalWrite(wAg, HIGH);
            break;
        case NOTE_G4:
            digitalWrite(bAg, HIGH);
            digitalWrite(wAg, HIGH);
            break;
        case NOTE_E4:
            digitalWrite(bAg, LOW);
            digitalWrite(wAg, HIGH);
            break;
        case NOTE_F4:
            digitalWrite(bAg, LOW);
            digitalWrite(wAg, LOW);
            break;
        case NOTE_C4:
            digitalWrite(bAg, HIGH);
            digitalWrite(wAg, HIGH);
    }
}

```

```
        break;
    case PAUSE:
        digitalWrite(bAg, HIGH);
        digitalWrite(wAg, LOW);
        break;
    case NOTE_A3:
        digitalWrite(bAg, LOW);
        digitalWrite(wAg, HIGH);
        break;
    case NOTE_D4:
        digitalWrite(bAg, HIGH);
        digitalWrite(wAg, LOW);
        break;
    case NOTE_CS4:
        digitalWrite(bAg, LOW);
        digitalWrite(wAg, LOW);
        break;
    case NOTE_B3:
        digitalWrite(bAg, HIGH);
        digitalWrite(wAg, HIGH);
        break;
    default:
        break;
}
```