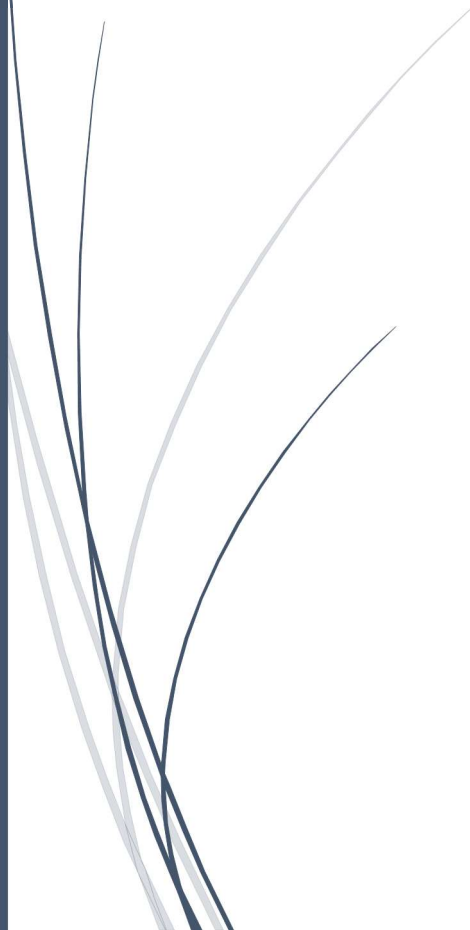




07/2022

Sample piano Arduino



Mattia Burato VR445864
Simone Pavanello VR446236
Marco Signorini VR446578

Sommario

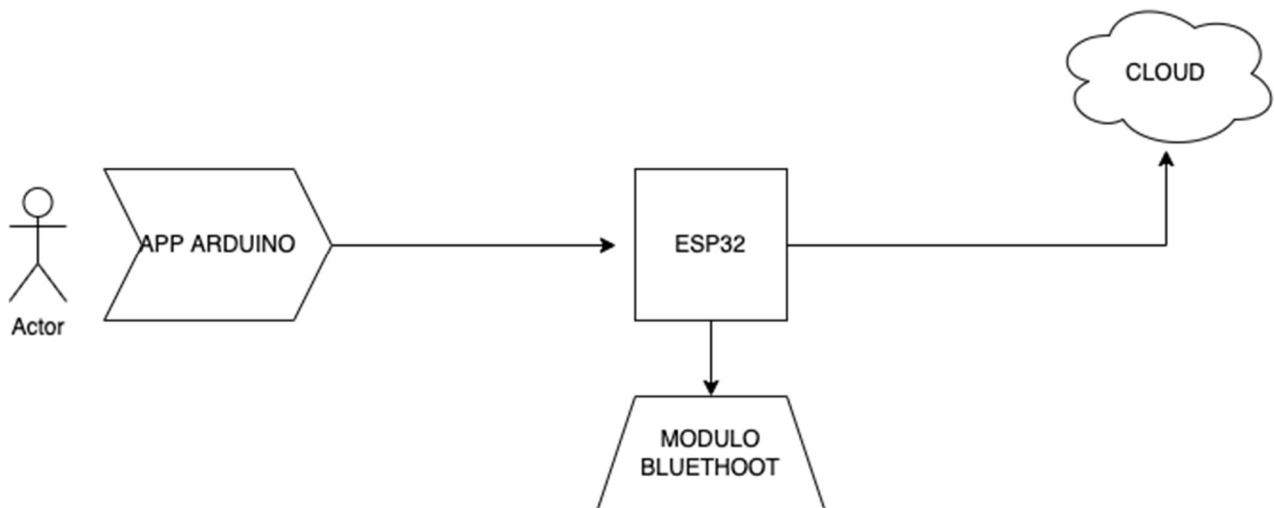
Descrizione generale	2
Componenti	4
Funzionamento	5
Codice	6
Cloud	7
Schema di Fritzing	8

Descrizione generale

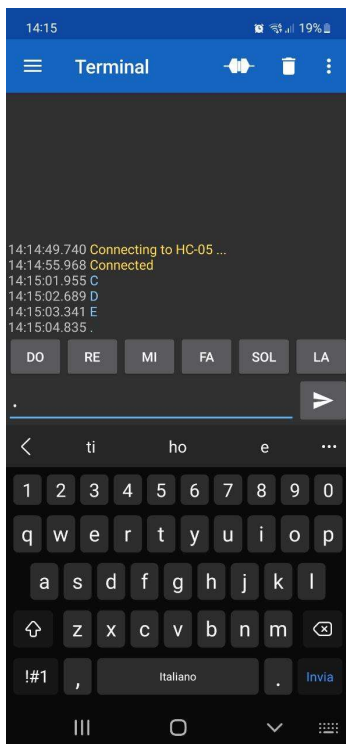
Il progetto consiste in un simulatore di piano con tre modalità.

- Live: suona direttamente la nota ricevuta
- Non live: tiene in memoria le note e le suona al ricevimento del carattere terminatore
- Modalità Star: suona il motivo di Star Wars.

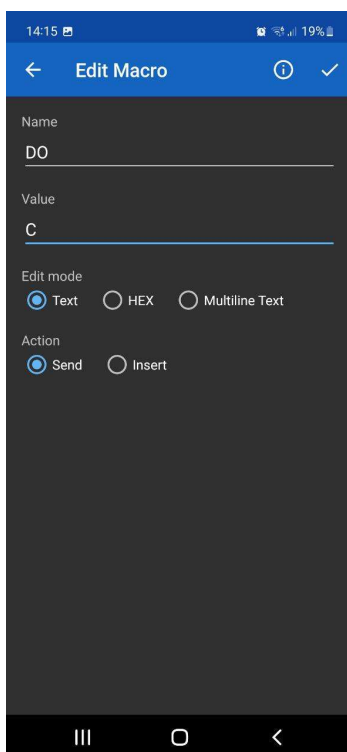
Una volta terminata la modalità (carattere terminatore) tutte le note vengono salvate su cloud. Le note vengono inviate attraverso un terminale Bluetooth connesso al modulo collegato all'ESP32.



Use case connessione dispositivi



In questa immagine si può un vedere una fase di connessione e di invio di tre note in un terminale Bluetooth (DO, RE, MI + carattere terminatore)



Nella seconda immagine si può vedere come è stato configurato il tasto DO: da applicazione viene mostrato DO, ma in realtà al modulo bluetooth viene inviato il carattere C, che è la notazione internazionale che rappresenta il DO.

Componenti

Per il progetto abbiamo selezionato i seguenti componenti

- Esp32
- Buzzer
- 5 resistenze (2 da 1K, 2 da 270 Ohm e 1 da 100 Ohm)
- Modulo Bluetooth HC-05
- 13 cavi maschio/femmina
- Led RGB
- Breadboard
- Terminale bluetooth

Scelte progettuali

Abbiamo scelto di usare l'ESP32 in quanto al suo interno è dotato di un modulo Wi-Fi. ESP32 monta anche un modulo Bluetooth ma noi disponendo di uno esterno abbiamo deciso di usarlo a scopo didattico.

Funzionamento

Il modulo Bluetooth rimane in attesa fino a quando non viene ricevuto un input dall'applicazione. Nel momento in cui viene ricevuto un carattere, questo viene valutato: si può ricevere ogni tipo di carattere e in base ad essi si decide il comportamento.

- "L" Indica che devo suonare "Live"
- "." Indica che devo terminare la riproduzione live (se stavo suonando live) se non stavo suonando live, riproduco tutte le note ricevute fino a quel momento. Inoltre, al ricevimento di questo carattere inizia il salvataggio sul cloud
- "\$" E' il carattere speciale che consente di suonare il tema musicale di "Star Wars".

Una volta ricevuta la nota la valutiamo e se corretta, ossia se è nell' insieme internazionale delle note (A, B, C, D, E, F, G) allora viene convertita alla frequenza da riprodurre con il buzzer. A questo punto viene illuminato il led in base all' intensità della frequenza. In caso di nota non valida non viene effettuata alcuna operazione. Le note vengono convertite attraverso una nostra libreria "pitches.h" dove verifichiamo in base alla lettera quale frequenza suonare, nella stessa libreria sono dichiarate le note con la frequenza per suonare il motivo di star wars. Una volta terminata la modalità salviamo su cloud tutte le note ricevute.

Codice

In questa sezione abbiamo inserito alcuni pezzi di codice importanti per il funzionamento.

Setup

In questa funzione inizializziamo i pin dei led e del buzzer, inoltre instauriamo la connessione con ThingSpeak (cloud) e la rete Wi-Fi. Il codice all'interno di questa funzione viene eseguito una sola volta all'avvio dell'applicazione.

```
// Operazioni iniziali, eseguite una sola volta all'avvio di arduino
void setup() {
  // Inizializza connessione wifi e connessione a cloud, e definisce i pin di output
  ledcAttachPin(BUZZER, TONE_PWM_CHANNEL);
  pinMode(LED_RED, OUTPUT);
  pinMode(LED_GREEN, OUTPUT);
  pinMode(LED_BLUE, OUTPUT);
  Serial.begin(9600);
  InitConnection();
  ThingSpeak.begin(client);
}
```

Loop

Nella funzione loop dopo aver verificato la connessione con il Bluetooth, iniziamo a ricevere gli input, e con la funzione "elaboraDatoRicevutoConBluetooth" controlliamo il carattere e lo riproduciamo. Il codice all'interno della funzione loop viene eseguito all'infinito.

```
// Operazione ripetuta all'infinito
void loop() {
  message = "";

  // Fino a quando non viene ricevuto alcun dato da bluetooth non viene eseguita alcuna elaborazione
  if (Serial.available() > 0) {
    // Se ho ricevuto un dato bluetooth, riempio la stringa message con ogni carattere ricevuto (finché c'è qualcosa da leggere)
    while (Serial.available()) {
      message += char(Serial.read());
    }

    // Elaboro il messaggio ricevuto
    elaboraDatoRicevutoConBluetooth();
  }
}
```

Cloud

Come cloud abbiamo deciso di utilizzare ThingSpeak di Matlab in quanto lo avevamo già visto a lezione. Per poter comunicare abbiamo utilizzato alcune funzioni messe a disposizione dalla libreria "ThingSpeak.h", di seguito un pezzo di codice in cui le utilizziamo.

```
void writeToCloud() {
    Serial.println("Preparo a scrivere su cloud");
    // Prepara l'array da scrivere su cloud

    // Ogni nota da scrivere occuperà 4 caratteri (3 per la nota e 1 per la virgola, Es.: "DO ," oppure "SOL,")
    int numeroNoteDaScrivere = contaNumeroNoteDaScrivere();
    char* melodiaString = (char *) malloc(sizeof(char) * (numeroNoteDaScrivere * 4));
    strcpy(melodiaString, "");

    for (int i = 0; i < NUMERO_MASSIMO_NOTE && melodia[i] > 0; i++) {
        strcat(melodiaString, convertiFrequenzaInNota(melodia[i]));
        strcat(melodiaString, ",");
    }

    // Scrive su cloud l'intero array
    int x = ThingSpeak.writeField(Channel_ID, Field_Number_1, melodiaString, myWriteAPIKey);
    if (x == 200) {
        Serial.println("Channel update successful.");
    }
    else {
        Serial.println("Problem updating channel. HTTP error code " + String(x));
    }
}
```

In questa funzione andiamo ad inserire all'interno di una stringa tutte le note che abbiamo suonato prima del ricevimento del carattere terminatore e tramite la funzione "ThingSpeak.writeField" inviamo le note al cloud.

Schema di Fritzing

In questa immagine è possibile vedere lo schema con tutti i dispositivi collegati con le varie resistenze. I nomi dei dispositivi e i valori delle resistenze sono visibili nella sezione “Componenti”.

