



Università degli studi di Verona
Dipartimento di Informatica

Corso di Arduino

Centralina ArtNet-DMX

Studenti

Michele Cenzato (VR492099)
Angelo Marano Massimiliano (VR456447)

Docenti

Nicola Dall'Ora
Enrico Fraccaroli
Sebastiano Gaiardelli

AA 2023/2024

Indice

1	Introduzione	3
1.1	Motivazione progetto	3
1.2	Protocollo DMX	3
1.3	Protocollo ArtNet	3
2	Hardware	4
2.1	Scelte progettuali	4
2.2	ESP32	4
2.3	Arduino Nano	5
2.4	Scheda MAX485	6
2.5	Schema dei collegamenti	7
3	Software	8
3.1	Librerie utilizzate	8
3.2	Connessione al cloud	8
3.2.1	Dati caricati	8
3.2.2	Invio dei dati	10
3.3	Interfaccia Web	11
4	Immagini	12
4.1	Centralina	12
4.2	Strumentazione	13
4.3	Software di controllo	14

1. Introduzione

1.1 Motivazione progetto

L'intento è quello di realizzare una centralina che permetta il controllo di apparecchi elettronici utilizzati nel mondo dello spettacolo. Ne esistono già in commercio, ma spesso sono vincolate ad applicazioni specifiche o proprietarie, rendendo l'uso del dispositivo dipendente da scelte implementative altrui. Si vuole progettare un apparecchio che possa essere comandato da tutti i software che implementano lo stesso protocollo, garantendo così una maggiore flessibilità e compatibilità.

1.2 Protocollo DMX

Il **protocollo DMX (Digital Multiplex)** è uno standard di comunicazione digitale ampiamente utilizzato nell'industria dello spettacolo e dell'illuminazione per controllare luci, effetti e dispositivi correlati, come macchine del fumo, teste mobili, dimmer e LED RGB. È definito dallo standard **DMX512**, dove "512" indica il numero massimo di canali che possono essere controllati su una singola linea.

Per avere un'idea più chiara: [Protocollo DMX - Wikipedia](#)

1.3 Protocollo ArtNet

Il **protocollo Art-Net** è uno standard di comunicazione basato su **UDP/IP** che è stato progettato per funzionare su reti Ethernet e per semplificare la trasmissione di dati **DMX** su lunghe distanze e attraverso reti complesse. In pratica, invia dati **DMX** come pacchetti **UDP** sulla rete. Questi pacchetti, successivamente, vengono riconvertiti in segnale **DMX** grazie a un **Nodo Art-Net**, ossia un dispositivo come quello che stiamo andando a realizzare in questo progetto.

Per avere un'idea più chiara: [Protocollo ArtNet - Wikipedia](#)

2. Hardware

2.1 Scelte progettuali

Si è scelto di utilizzare due microcontrollori invece che uno solo perché:

- Non esistono librerie compatibili con l'ESP32 in questione riguardo il DMX
- Si è voluta provare la comunicazione seriale
- Si ha voluto avere a che fare con due schede diverse per scoprirle entrambe

2.2 ESP32

L'**ESP32** è un microcontrollore versatile e potente dotato di connettività Wi-Fi e Bluetooth integrata. Grazie alle sue capacità di elaborazione e comunicazione wireless, è ideale per progetti che richiedono una gestione dei dati in tempo reale e la connessione a reti Internet o Cloud. In questo progetto, l'ESP32 viene utilizzato per:

- Gestire la comunicazione wireless con il sistema Cloud, permettendo il monitoraggio e il controllo remoto
- Inviare i dati ad Arduino Nano tramite seriale
- Semplificare l'integrazione del sistema grazie alla sua versatilità e al supporto di numerose librerie



Figura 2.1: Scheda ESP32 Wroom32

2.3 Arduino Nano

L'**Arduino Nano** è un microcontrollore compatto e a basso consumo, basato sul chip ATmega328, molto diffuso per la sua semplicità d'uso e la sua facilità di integrazione in progetti di piccole dimensioni. Viene utilizzato in questo progetto per:

- Leggere i dati inviati su porta seriale dalla scheda ESP32
- Inviare i dati alla scheda MAX485 per essere convertiti in DMX
- Offrire una soluzione che non vada a sovraccaricare l'ESP32 rendendolo più lento nel ricevere dati dalla rete

L'integrazione di entrambi i componenti permette di sfruttare i punti di forza di ciascun microcontrollore, garantendo una gestione efficiente sia della parte di comunicazione wireless (ESP32), sia del controllo diretto e preciso degli hardware periferici (Arduino Nano).

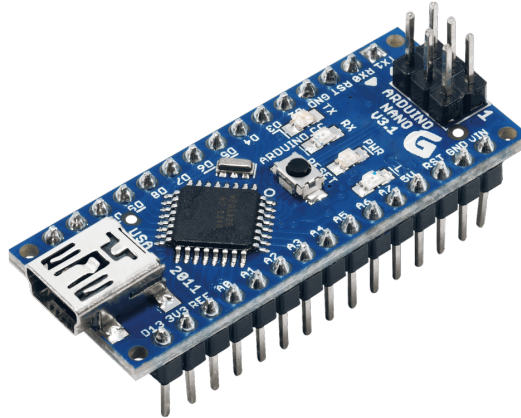


Figura 2.2: Scheda Arduino Nano

2.4 Scheda MAX485

La scheda basata sul chip **MAX485** funge da convertitore tra il segnale inviato da Arduino (**TTL**) e quello utilizzato dal DMX (**RS-485**). Questo componente è fondamentale nel nostro progetto perché:

- Consente di convertire il segnale digitale proveniente dal microcontrollore in un segnale compatibile con lo standard DMX. Da un segnale **single ended** si passa ad uno **differenziale**, adatto a coprire lunghe distanze ad alta velocità di trasmissione.
- Facilita l'interfacciamento tra dispositivi con livelli logici differenti, assicurando che il segnale DMX venga trasmesso correttamente agli apparecchi di illuminazione e agli altri dispositivi dello spettacolo.

In sintesi, l'uso della scheda **MAX485** permette di mantenere la qualità e l'integrità del segnale DMX, rendendo il sistema più affidabile e performante.

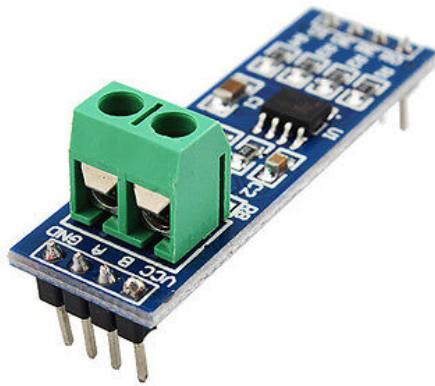


Figura 2.3: Scheda MAX485

2.5 Schema dei collegamenti

Di seguito è riportato lo **schema dei collegamenti** realizzato con **Fritzing**.

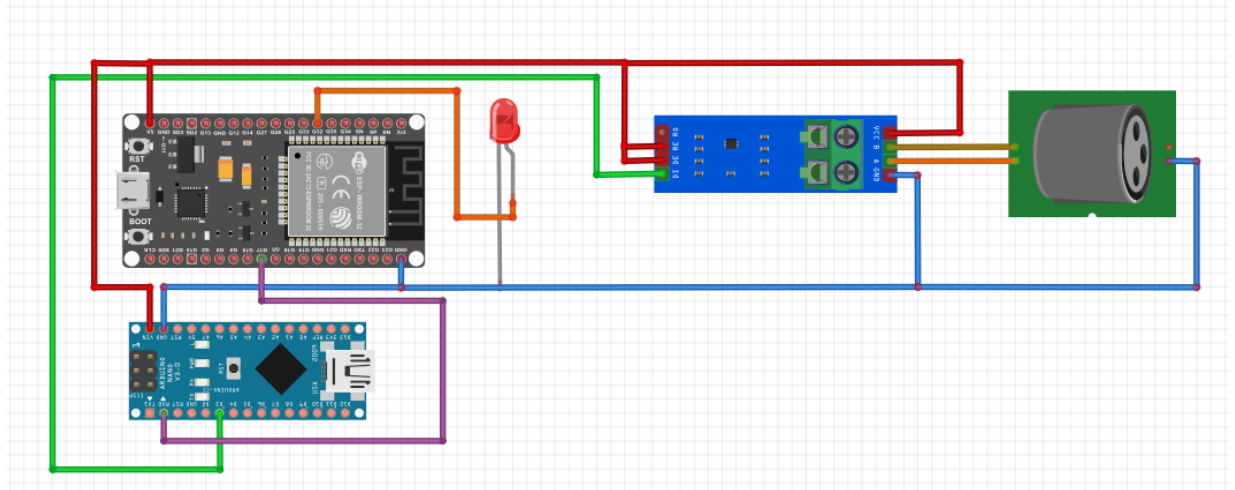


Figura 2.4: Schema dei collegamenti hardware.

Più nello specifico:

- L'alimentazione viene presa dalla porta **MicroUSB** della scheda **ESP32**
- La scheda **ESP32** alimenta le altre due schede
- Pin 17 ESP -> Pin RX0 Nano: comunicazione seriale
- Pin 32 ESP -> LED: led di stato
- Pin 3 Nano -> Pin DI MAX485: segnale digitale dati
- Pin DE/RE MAX485: sempre a livello 5V
- Pin A/B/GND MAX485: segnale differenziale DMX con massa (a porta XLR)

Per connettere fisicamente i cavi alle varie schede è stata utilizzata la tecnica della **saldatura a stagno**.

3. Software

3.1 Librerie utilizzate

Per la gestione della comunicazione con il cloud, dei pacchetti ArtNet, del WiFi e di tutto il resto, nel progetto vengono utilizzate le seguenti librerie:

- `#include "ThingSpeak.h"`: per interfacciarsi con la piattaforma ThingSpeak e inviare dati in tempo reale.
- `#include <WiFiUdp.h>`: per gestire i pacchetti ArtNet.
- `#include <Preferences.h>`: per salvare le impostazioni in memoria.
- `#include <WebServer.h>`: per generare l'interfaccia web.
- `#include <WiFi.h>`: per gestire tutto ciò che riguarda la rete.
- `#include <DmxSimle.h>`: per gestire l'output DMX, solo su Arduino Nano.

3.2 Connessione al cloud

La connessione al cloud è gestita interamente dall'**ESP32**. Grazie alla sua connettività Wi-Fi integrata, l'ESP32 si occupa di inviare dati al servizio cloud, consentendo il monitoraggio in tempo reale e l'elaborazione delle informazioni acquisite dal sistema.

3.2.1 Dati caricati

I dati caricati sul cloud sono i seguenti:

- `averageIntensity`: intensità media di tutti i canali attivi. Può essere utilizzata come statistica per misurare il consumo medio di energia elettrica.
- `activeChannels`: numero di canali attivi. Utile per avere un'idea di quanti dispositivi sono accesi.

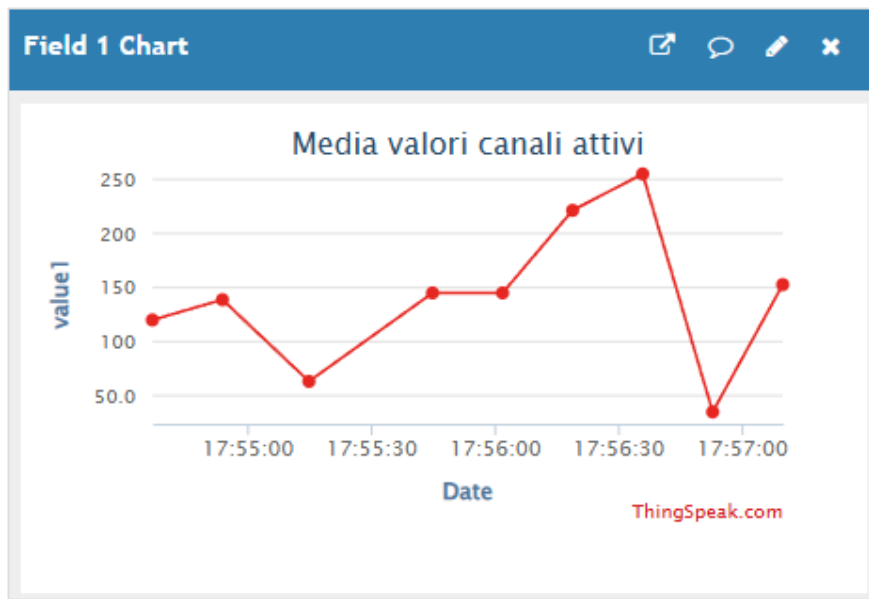


Figura 3.1: Grafico media canali attivi ThingSpeak

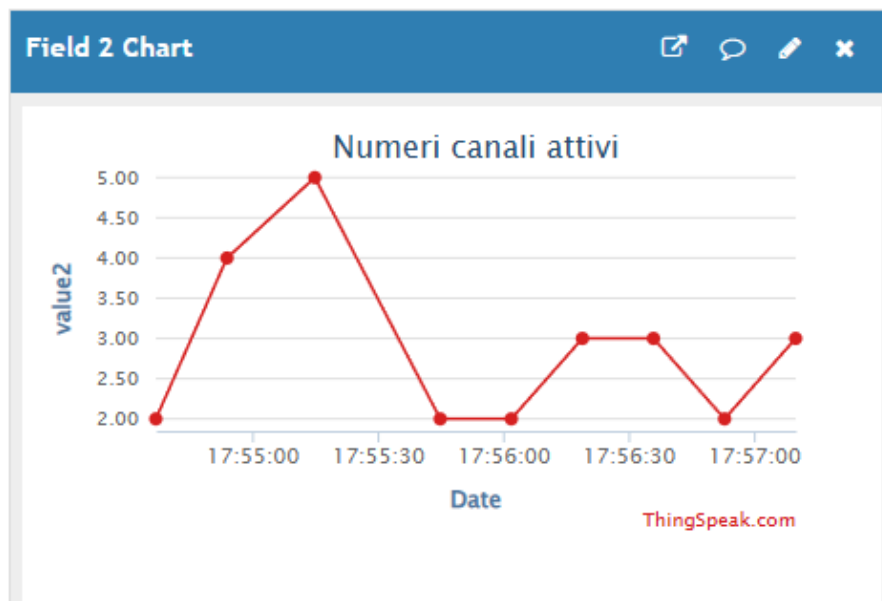


Figura 3.2: Grafico numero canali attivi ThingSpeak

3.2.2 Invio dei dati

Il seguente frammento di codice mostra come viene gestito l'invio periodico dei dati al Cloud utilizzando ThingSpeak:

```
1 void loop() {
2
3     ...
4
5     currentMillis = millis();
6     if (currentMillis - previousMillis >= interval) {
7         previousMillis = currentMillis;
8         updateCloudInfo();
9     }
10
11 }
12
13 void updateCloudInfo() {
14
15     int activeChannels = 0;
16     int totalIntensity = 0;
17
18     for (int i = 0; i < length; i++) {
19         if (data[i] > 0) {
20             activeChannels++;
21             totalIntensity += data[i];
22         }
23     }
24
25     float averageIntensity = (activeChannels > 0) ? (float)
        totalIntensity / activeChannels : 0;
26
27     ThingSpeak.setField(1, averageIntensity);
28     ThingSpeak.setField(2, activeChannels);
29
30     int res = ThingSpeak.writeFields(channel_ID, ApiKeyR);
31     if (res == 200)
32         Serial.println("Dati inviati al cloud!");
33     else
34         Serial.print("Errore nell'invio dei dati al cloud!");
35
36 }
```

Listing 3.1: Invio dati a ThingSpeak

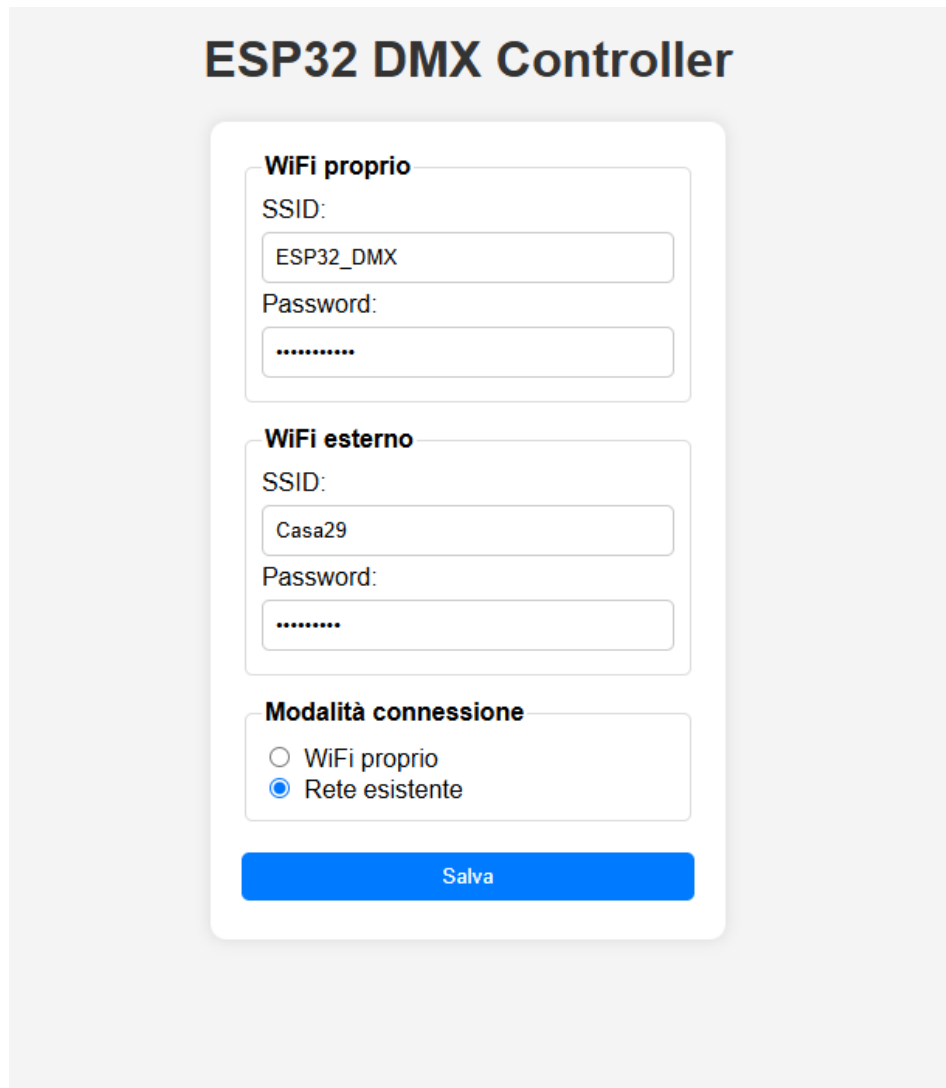
In questo esempio, il codice controlla se è trascorso un intervallo di tempo definito per inviare nuovi dati. In caso affermativo, aggiorna le variabili di tempo e scrive i campi sul canale ThingSpeak. Il risultato dell'operazione viene poi verificato e stampato sulla Serial Monitor.

3.3 Interfaccia Web

Vi è la possibilità di collegarsi al dispositivo tramite un'interfaccia web, navigando all'indirizzo IP del microcontrollore. Questa interfaccia è sempre disponibile, sia in presenza di una rete esistente, sia in modalità di connessione diretta.

Sulla pagina è possibile:

- Configurare la modalità di funzionamento:
 - Modalità WiFi connesso
 - Modalità WiFi Stand-Alone
- Modificare SSID e Password della rete a cui collegarsi
- Modificare SSID e Password della rete autogenerata



The image shows a web interface titled "ESP32 DMX Controller". It contains three main sections for configuration:

- WiFi proprio**: Fields for SSID (containing "ESP32_DMX") and Password (masked with dots).
- WiFi esterno**: Fields for SSID (containing "Casa29") and Password (masked with dots).
- Modalità connessione**: Two radio buttons, "WiFi proprio" (unselected) and "Rete esistente" (selected).

At the bottom of the form is a blue button labeled "Salva".

Figura 3.3: Interfaccia Web

4. Immagini

Di seguito alcune immagini per comprendere meglio cosa si è andato a realizzare.

4.1 Centralina

La centralina è composta esternamente da: connettore **MicroUSB**, porta **DMX** femmina, led di stato, antenna **WiFi**.



Figura 4.1: Centralina vista dall'esterno

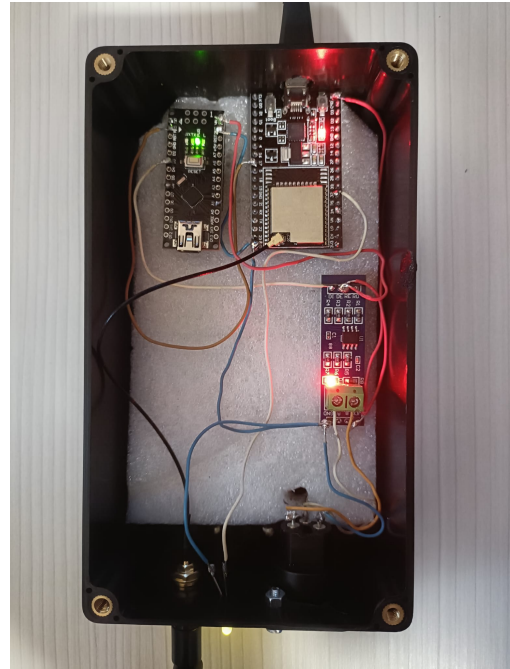


Figura 4.2: Centralina vista dall'interno

4.2 Strumentazione

Qualche immagini della tipica strumentazioni utilizzati nel campo dell'illuminotecnica.



Figura 4.3: Faro LED RGBW



Figura 4.4: Testa mobile



Figura 4.5: Centralina in commercio



Figura 4.6: Connettore XLR per il DMX

4.3 Software di controllo

Sono disponibili in rete diversi software gratuiti in grado di inviare segnale di tipologia **DMX** e **ArtNet**. In questo progetto per la fase di testing è stato utilizzato **QLC+**, software open source adatto allo scopo.



Figura 4.7: Software QLC+ - Virtual Console

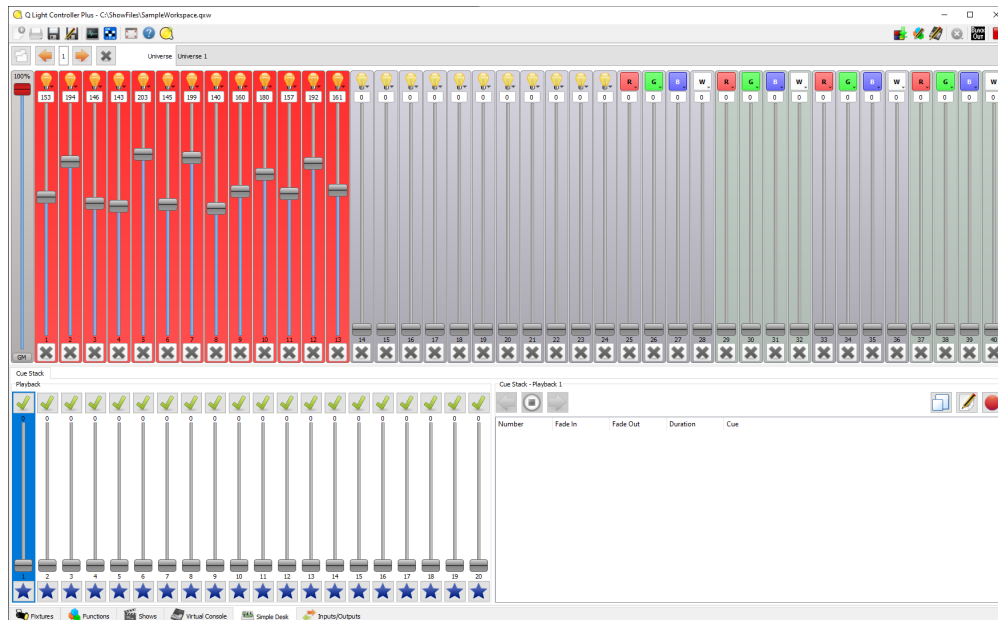


Figura 4.8: Software QLC+ - Comandi canali