

IOTheater

Elisa Simoni – 922134

Embedded Systems and IoT - 2022-2023

Introduzione

Il mio progetto consiste nella creazione di un teatro smart che sia integrato con un'applicazione mobile e un web server node.js express. L'applicazione mobile consentirà agli utenti di programmare routine settando durata, colore luci, durata luci, luminosità, canzone, durata canzone e volume. L'app comunica con arduino attraverso un modulo bluetooth. Il web server comunica tramite protocollo MQTT con un ESP8226 e via seriale con Arduino stesso. Nel web server sarà possibile vedere cosa sta funzionando nell'Arduino e se i sensori dell'ESP rilevano una presenza o meno.

- Ecco alcune delle funzionalità principali dell'applicazione:
- Durata della routine: è possibile settare la durata totale della routine, una volta inviata i componenti si “muoveranno” da soli senza bisogno di intervenire.
- Durata di accensione delle luci: gli utenti potranno impostare un timer per decidere quanto tempo le luci dovranno rimanere accese durante gli spettacoli. Sarà possibile anche regolare l'intensità della luce e il colore.
- Rilevamento dell'artista: l'applicazione sarà dotata di un sensore "occhio di bue" che rileverà la presenza dell'artista sul palco. Quando l'artista verrà rilevato, il faretto si accenderà automaticamente
- Gestione della musica: gli utenti potranno scegliere le canzoni (Tetris, Trono di Spade e Star Wars) da riprodurre durante lo spettacolo, decidere la loro durata e regolare il volume secondo le loro preferenze.
- Apertura e chiusura del sipario: all'avvio della routine il sipario si aprirà e alla fine si chiuderà.

Resistenza 220 Ohm



IOTheater HARDWARE

Jumper



LED RGB 5mm



Micro servo 9g



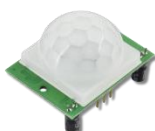
Mini Piezo Buzzer



Modulo HC-06



PIR Sensor



ESP8266



Led 5mm

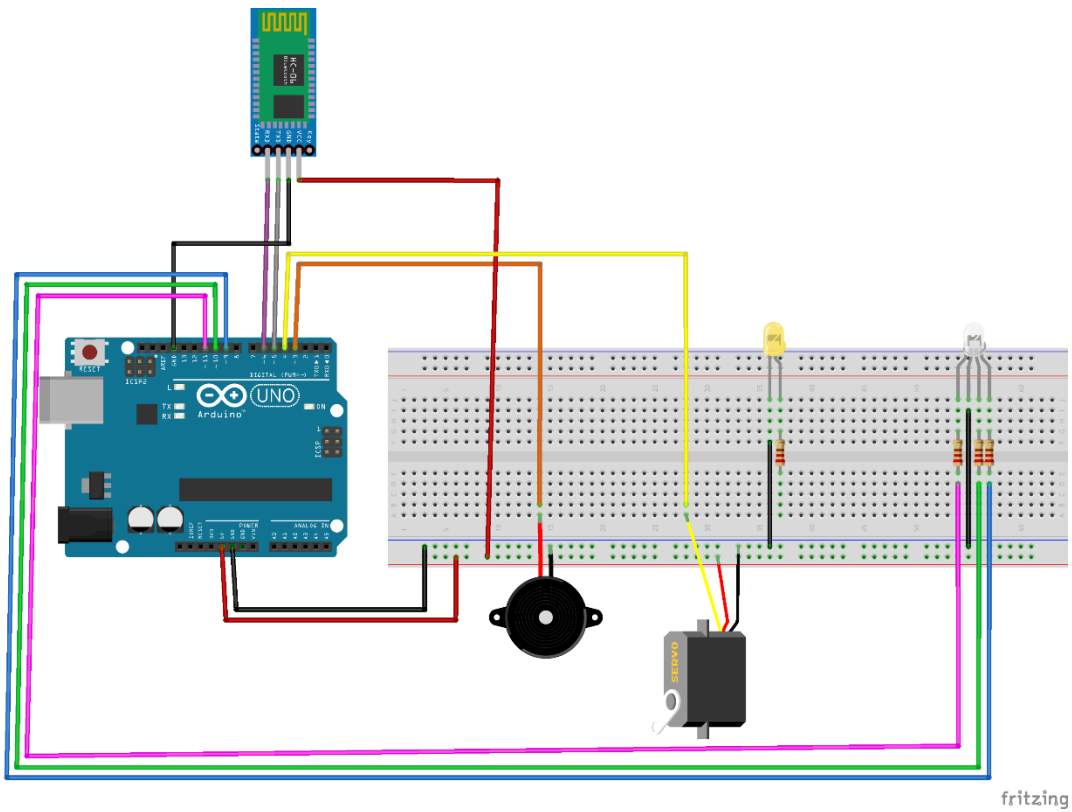


Arduino UNO RV3

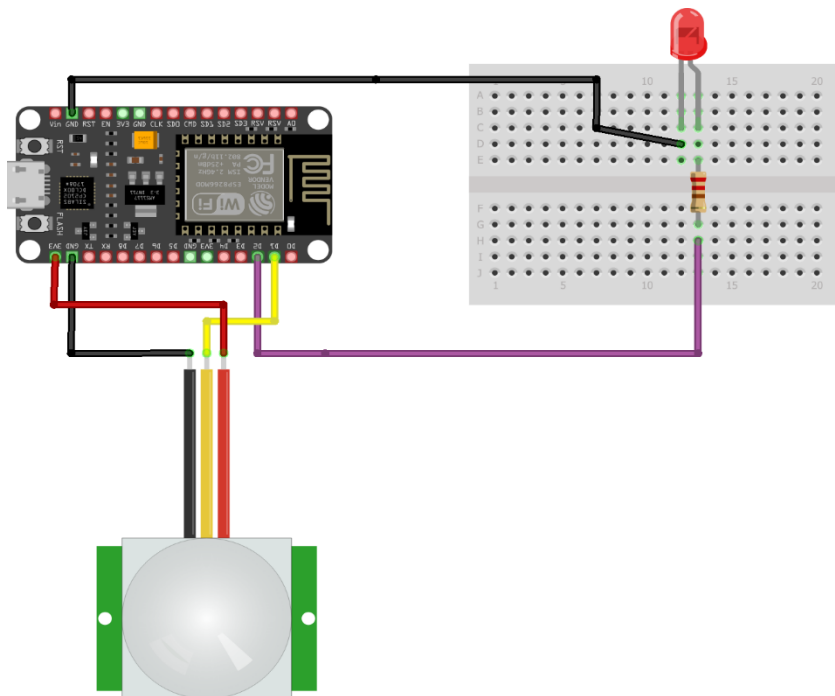


Circuiti

Arduino UNO



ESP8266



Architettura software

IOTheater Arduino

Linguaggio di programmazione: C++

Cartelle:

- ↳ Kernel (MsgService, Scheduler, Task ecc...)
- ↳ Devices (Hardware)
- ↳ Model (SmartTheater, Spotlight, Music, Curtains, Light ecc...)
- ↳ Task (Tutte le task)

Breve spiegazione: Il fulcro del codice è SmartTheater che può assumere due stati (IS_SHOW, NO_SHOW). Il codice parte in NO_SHOW le “AudienceLight” sono accese, appena avviene la connessione bluetooth le luci si spengono e si entra nella fase IS_SHOW dove avremo per prima cosa la lettura del JSON inviato dall’applicazione mobile. In base alla durata della routine i componenti hardware cambieranno stato, per prima cosa si accenderà il micro servo, si aprirà a 180° e poi si spegnerà in attesa della fine della routine per poi riaccendersi e tornare alla posizione di partenza (simulazione apertura chiusura sipario). Musica e luci entreranno nelle loro fasi di ON e OFF in base alla loro “StartTime” ed “EndTime” passata dall’utente.

Librerie: TimerOne, ArduinoJson, Pitches (per la musica)

IOTheater ESPBoard

Linguaggio di programmazione: C++

Cartelle:

- ↳ Kernel (Task.h)
- ↳ Devices (Hardware)
- ↳ Model (ESPBoard)
- ↳ Task (ActorPresenceTask)

Breve spiegazione: Il fulcro del codice è ESPBoard. Il funzionamento del codice è il seguente: All’inizio del programma viene effettuata una connessione WiFi a una rete a 2,4Ghz, la connessione al server e al broker MQTT. Il PIR effettua continue rilevazioni e manda un JSON al web Server contenente la rilevazione o meno dell’attore.

Librerie: ArduinoJson

P.S. Per la programmazione sui microcontrollori ho usato come IDE VSCode supportato dal plugin PlatformIO.

IOTheater WEBServer

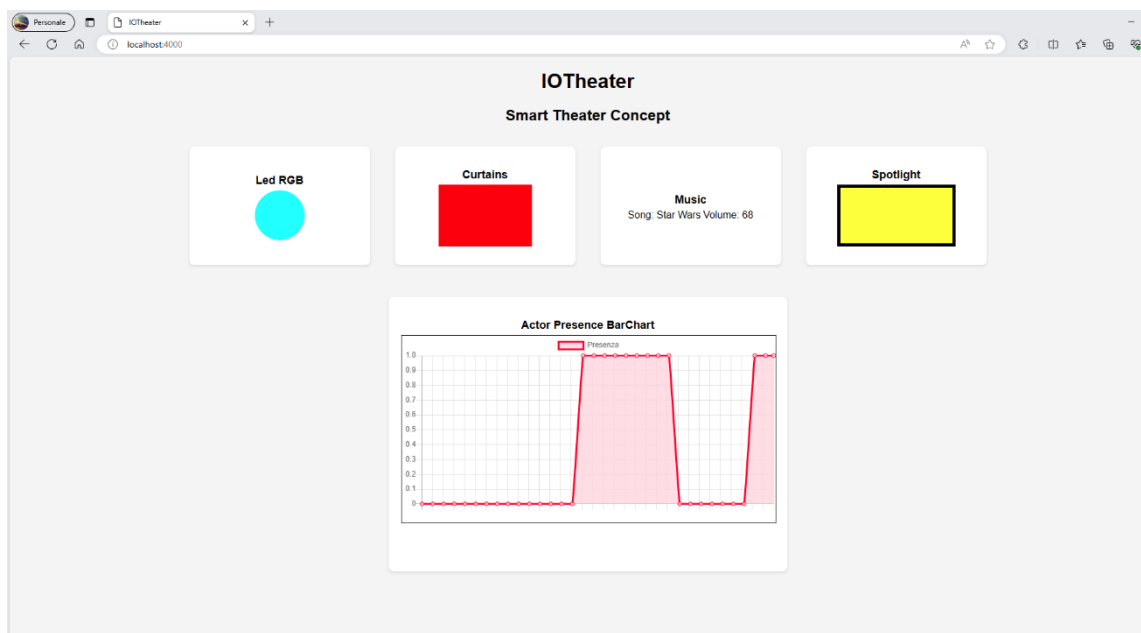
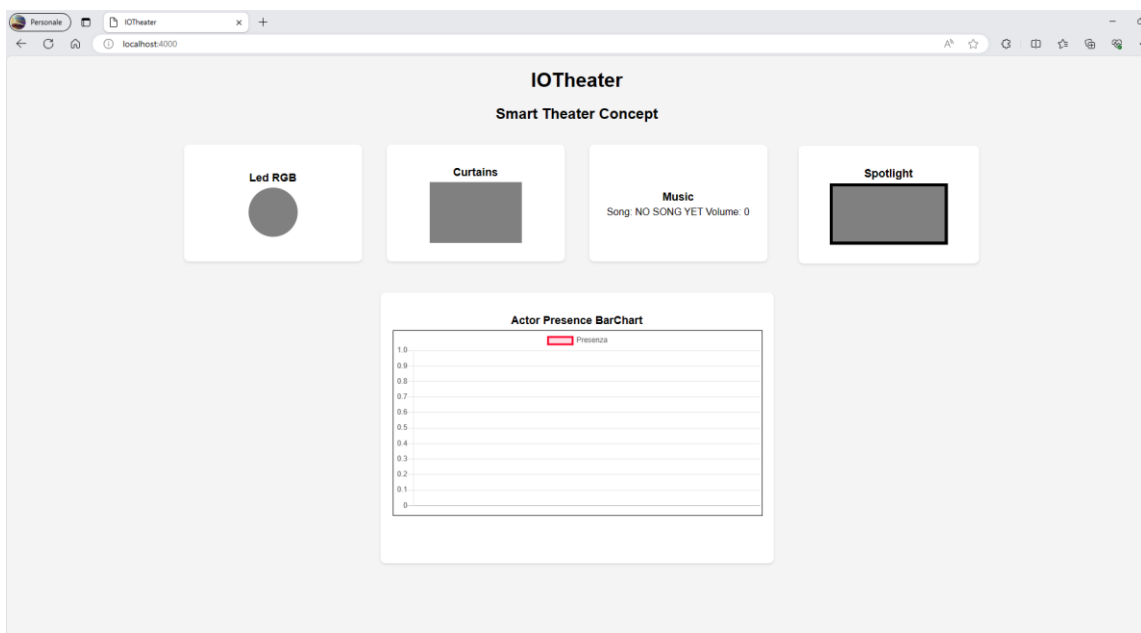
Linguaggio di programmazione: Javascript, HTML, CSS, node.js express

File:

- ↗ index.html
- ↗ style.css
- ↗ script.js

Breve spiegazione: Il server viene creato con node.js express e aperto sulla porta 4000. Vi è una conseguente connessione al broker MQTT (broker.emqx.io). Il web Server legge messaggi in ingresso sia da Arduino che dall'ESP. La pagina index.html contiene una rappresentazioni schematica delle componenti in funzione su Arduino e un grafico del periodo di presenza dell'attore registrato dal PIR. Se lo stato è NO_SHOW Spotlight non registra, se lo stato è SHOW allora può rilevare.

Librerie: mqtt, express, socket.io, chart.js, serialport



IOTheaterApp

Linguaggio di programmazione: Kotlin

Breve spiegazione: L'app si collega via bluetooth ad Arduino e trasferisce un JSON contenente le informazioni riguardo alla routine. La modalità bluetooth triggera il IS_SHOW di SmartTheater. (p.s. prima di effettuare qualsiasi invio bisogna connettersi bluetooth).

Librerie: Material

The image displays two screenshots of the 'ROUTINE' app interface, connected by a large black arrow pointing from left to right, indicating a transition or configuration change.

Left Screenshot (Initial State):

- ROUTINE** (Title)
- Set duration of routine (minutes): Input field with 'Duration (minutes)' placeholder.
- Choose what light will be On: Three buttons labeled RED, GREEN, and BLUE.
- Light timer (seconds): Two input fields labeled 'Turn On (seconds)' and 'Turn Off (seconds)'.
- Light intensity: A horizontal slider.
- Select Song: A dropdown menu showing '1'.
- Volume: A horizontal slider.
- Song Timer (seconds): Two input fields labeled 'Play Start (seconds)' and 'Play Stop(seconds)'.
- START ROUTINE** (Orange button)
- CONNECT** (Purple button) and **READY** (Orange button) at the bottom.

Right Screenshot (Configured State):

- ROUTINE** (Title)
- Set duration of routine (minutes): Input field with '1'.
- Choose what light will be On: Three buttons labeled RED, GREEN, and BLUE. The RED button is highlighted.
- Light timer (seconds): Two input fields with '2' and '50'.
- Light intensity: A horizontal slider.
- Select Song: A dropdown menu showing '3'.
- Volume: A horizontal slider.
- Song Timer (seconds): Two input fields with '2' and '2'.
- START ROUTINE** (Orange button)
- CONNECT** (Purple button) and **READY** (Orange button) at the bottom.

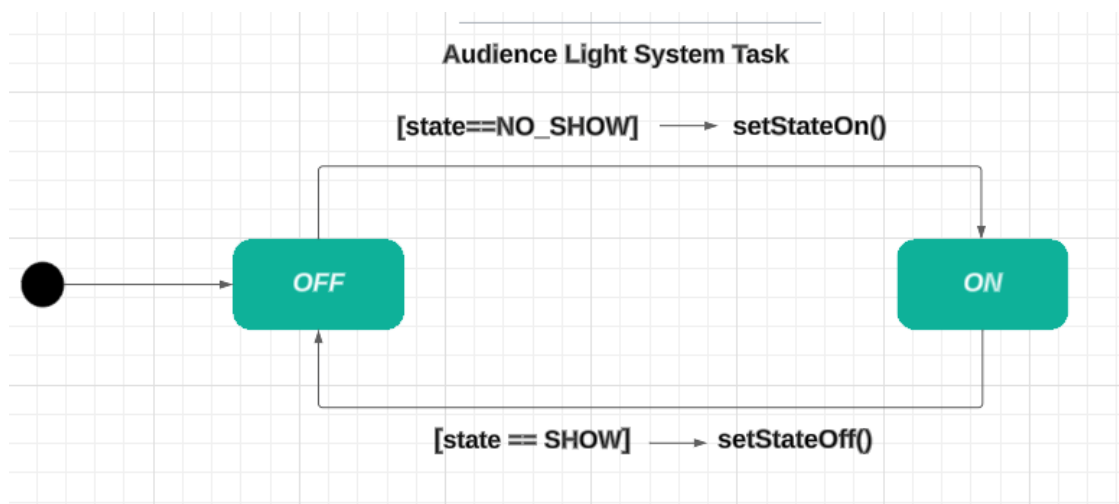
TASK

IOTheater Arduino

È composto da 7 tasks:

- ↳ CurtainsTask
- ↳ AudienceLightSystemTask
- ↳ StageLightSystemTask
- ↳ MusicTask
- ↳ BluetoothMsgTask
- ↳ SerialMsgTask

Audience Light System Task

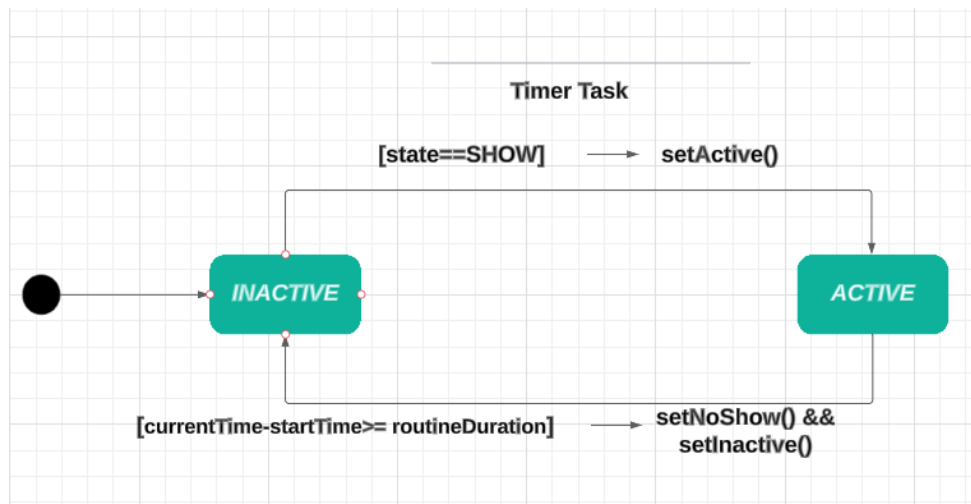


La classe AudienceLightSystemTask è responsabile della gestione dello stato del sistema di illuminazione per il pubblico.

- Lo stato iniziale del sistema di illuminazione per il pubblico viene impostato su OFF.
- Se lo stato è OFF e non ci sono spettacoli in corso (`theater->isNoShow()` restituisce true), il metodo `setStateOn` viene chiamato per accendere il sistema di illuminazione.
- Se lo stato è ON e c'è uno spettacolo in corso (`theater->isShow()` restituisce true), il metodo `setStateOff` viene chiamato per spegnere il sistema di illuminazione.

In sintesi, la classe AudienceLightSystemTask gestisce lo stato del sistema di illuminazione per il pubblico in un teatro, accendendolo quando non ci sono spettacoli in corso e spegnendolo quando c'è uno spettacolo in corso.

Timer Task

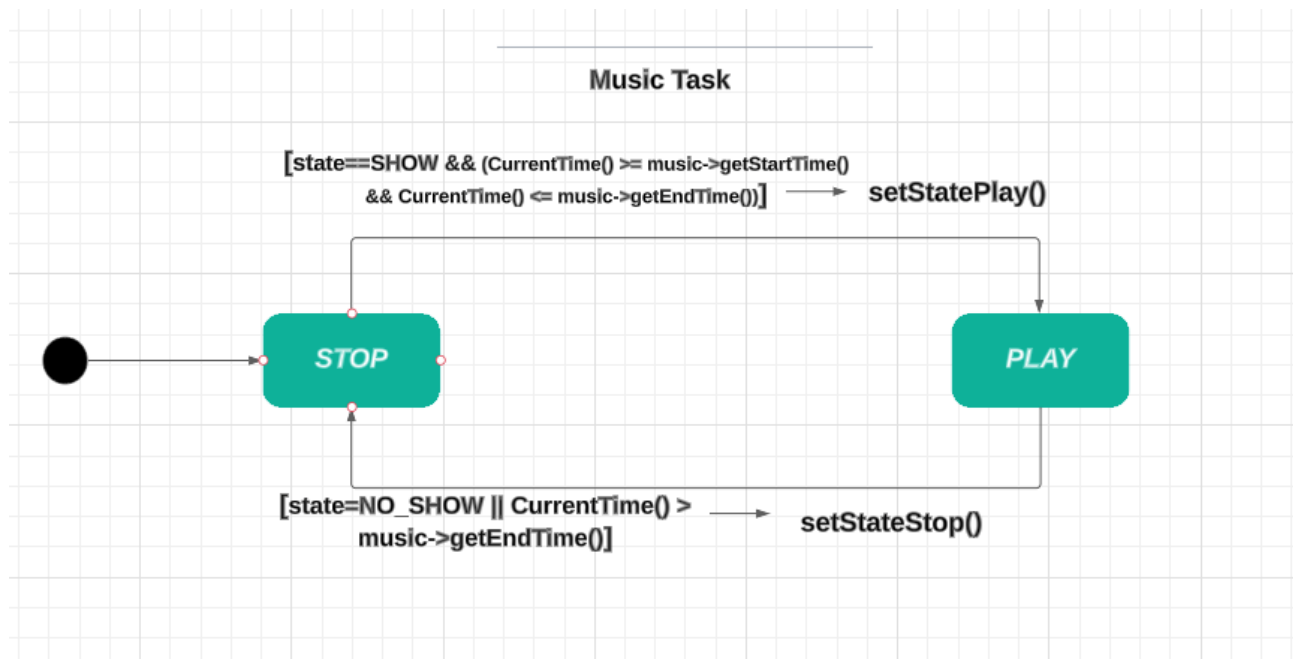


La classe `TimerTask` viene utilizzata per gestire il tempo e lo stato di uno spettacolo teatrale.

- Se lo stato è **INACTIVE** e lo spettacolo è attivo (`theater->isShow()` restituisce `true`), il metodo `setActive` viene chiamato per passare allo stato **ACTIVE**.
- Se lo stato è **ACTIVE**, il tempo trascorso viene calcolato come differenza tra `currentTime` e `startTime`, e viene aggiornato il tempo corrente dello spettacolo teatrale.
- Se il tempo trascorso supera la durata dello spettacolo, viene chiamato il metodo `setNoShow` per indicare che lo spettacolo è finito, e lo stato viene impostato su **INACTIVE**.

In sintesi, la classe `TimerTask` gestisce il tempo e lo stato di uno spettacolo teatrale, passando tra gli stati **INACTIVE** e **ACTIVE** in base alla presenza o all'assenza dello spettacolo, e tiene traccia del tempo trascorso dello spettacolo teatrale.

Music Task

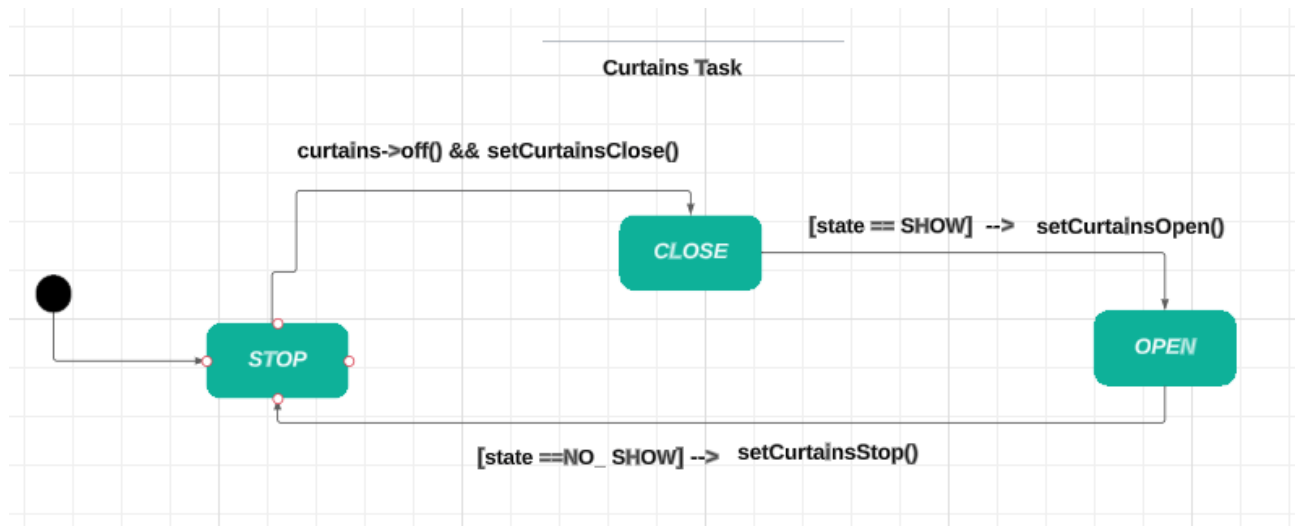


Lo stato STOP rappresenta lo stato iniziale dell'oggetto MusicTask. Qui la musica è ferma.

- Se lo spettacolo è in corso e il tempo corrente è compreso tra l'orario di inizio e l'orario di fine della musica, l'oggetto passa allo stato PLAY.
- Lo stato PLAY indica che la musica è attiva. Viene chiamato il metodo per avviare la riproduzione della musica.
- Se lo spettacolo non è più in corso o il tempo corrente supera l'orario di fine della musica, l'oggetto ritorna allo stato STOP, spegne la musica e reimposta lo stato iniziale.

In breve, l'oggetto MusicTask controlla se la musica deve essere attivata durante lo spettacolo in base all'orario corrente e passa tra gli stati STOP e PLAY a seconda delle condizioni specificate.

Curtains Task



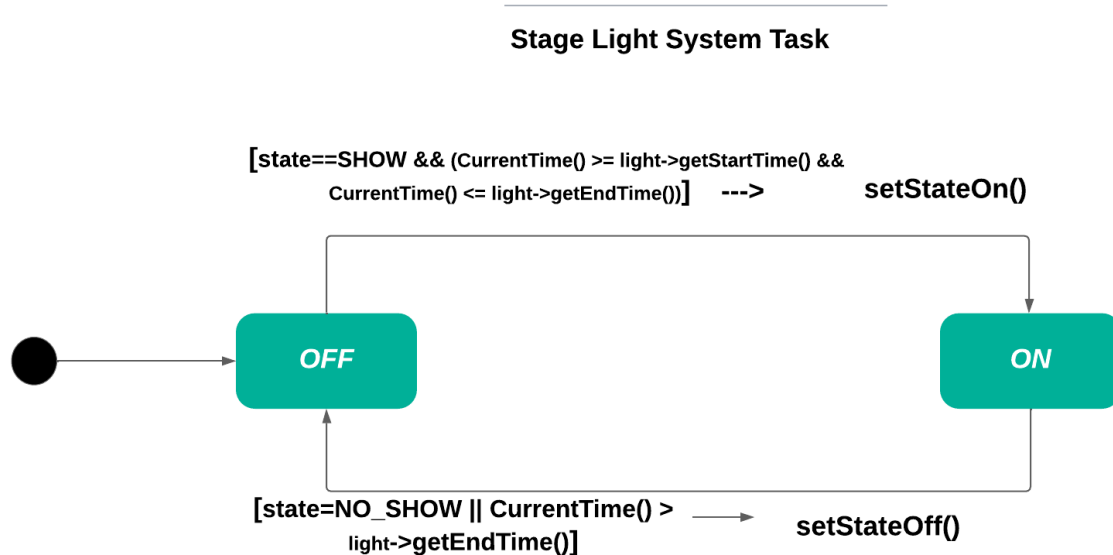
Lo stato iniziale è STOP, che rappresenta le tende ferme. Dalla condizione iniziale, le transizioni di stato sono le seguenti:

- CLOSE: Le tende sono chiuse. Vengono aperte quando lo spettacolo inizia.
- OPEN: Le tende sono aperte. Vengono chiuse quando lo spettacolo termina.
- STOP: Le tende sono ferme. Rimangono chiuse fino a quando lo stato non viene cambiato.
- L'uso dello `curtains-> off` e `curtains->on` mi permette di chiudere e aprire il motore del servo e in modo da non tenerlo in tensione quando non utilizzato.

BluetoothMsgTask e SerialMsgTask

Queste due task gestiscono la comunicazione bluetooth con l'applicazione e la comunicazione seriale con il web server. Il metodo `Handle content` gestisce i messaggi e il metodo `notify` spedisce i messaggi.

Stage Light System Task



La classe `StageLightSystemTask` è responsabile del controllo del sistema di illuminazione del palcoscenico in base allo spettacolo corrente e all'orario. La classe gestisce due stati: **OFF** (sistema di illuminazione spento) e **ON** (sistema di illuminazione acceso). Durante l'esecuzione del metodo `tick()`, vengono eseguite diverse azioni in base allo stato corrente:

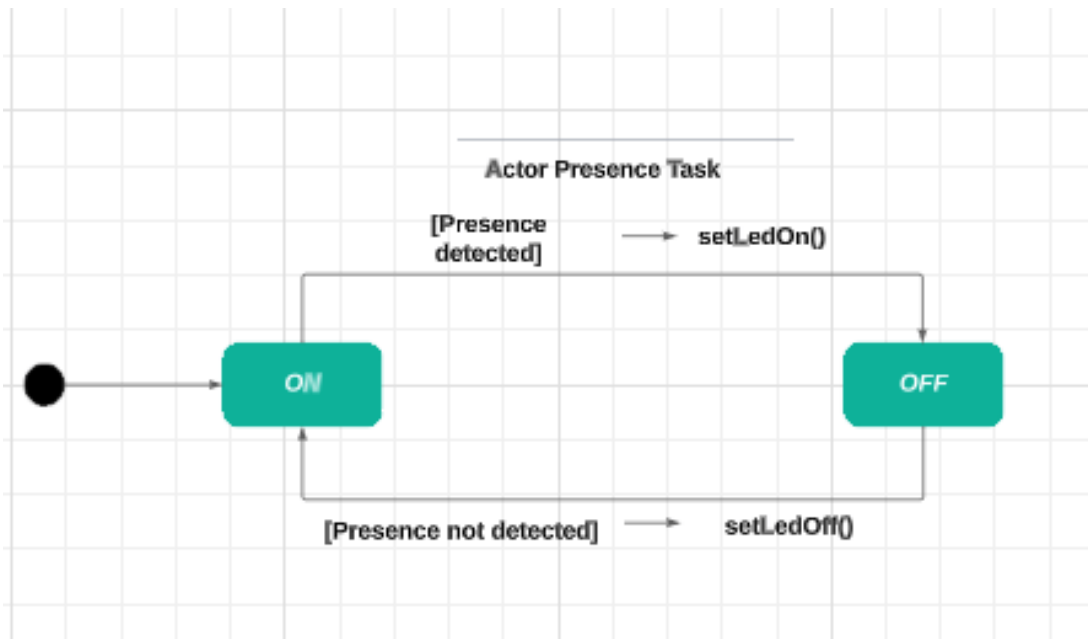
- **OFF:** Se lo spettacolo è in corso e `currentTime` rientra nell'intervallo di tempo del sistema di illuminazione del palcoscenico (specificato dalle proprietà `startTime` e `endTime`), il sistema di illuminazione viene acceso chiamando il metodo `turnOn()`. Lo stato viene quindi cambiato in **ON**.
- **ON:** Se lo spettacolo non è in corso e `currentTime` supera l'ora di fine del sistema di illuminazione del palcoscenico, il sistema di illuminazione viene spento chiamando il metodo `turnOff()`.

IOTheater ESP

É composto da 1 tasks:

↳ ActorPresenceTask

Actor Presence Task



Semplicemente se il PIR rileva una presenza viene settato su On il led che rappresenta il faretto e se invece non viene rilevata si setta su off.