# La Battaglia delle Urla

Dimostra la tua maestria nella programmazione con questa sfida finale. Gestisci stati complessi per creare un 'urlometro' che misura la durata delle tue urla.

## Corsi

- Classi 9-12

## Materiali

- Telefono e computer
- Connessione a Internet
- Cartone, forbici e un paio di elastici per fissare il telefono al cartone

## Descrizione

Questo è il progetto finale e più complesso, pensato perché gli studenti mettano insieme tutto ciò che hanno imparato. Dovranno costruire una "macchina a stati" funzionante per un gioco, gestendo le transizioni tra gli stati "in attesa", "conteggio" e "terminato". Questo richiede una comprensione solida di come eventi, loop, variabili e condizionali annidati lavorano insieme per creare un'applicazione interattiva robusta.

## Obiettivi Educativi

- Riunire in un unico progetto complesso tutti i concetti di programmazione appresi.
- Progettare e realizzare un programma con più stati e transizioni logiche (una macchina a stati).
- Gestire il flusso del programma usando una combinazione sinergica di logica guidata dagli eventi e di logica guidata dai loop.
- Fare il debug e risolvere i problemi in un programma con una logica di stato complessa.

## Inizio (10 minuti) - La Macchina a Stati

1. Dai il benvenuto alla classe: **"Siete arrivati in cima alla montagna. Oggi costruirete il vostro progetto più intelligente di sempre: un gioco che misura la durata delle vostre urla."**

2. Introduci il concetto finale: **"Pensate a un videogioco. Un personaggio può essere 'in esplorazione', 'in battaglia' o 'in un menu'. Ognuno è uno 'stato' diverso. Anche il nostro gioco di oggi ha degli stati: 'pronto a partire', 'conteggio del tempo' e 'visualizzazione del risultato'. Imparare a gestire questi stati è la chiave della programmazione avanzata."**
3. Lancia la sfida di progettazione: **"Come faremo? Ci servirà un po' di tutto: un evento per iniziare, un loop per contare e delle variabili per ricordare in che stato ci troviamo."**

## Macchine a Stati: le "Modalità" di un Programma

Quasi tutti i programmi complessi possono essere descritti come una **macchina a stati**. Questo significa che il programma può trovarsi in "modalità" o "stati" diversi, e che esistono regole chiare per passare da uno stato all'altro. Il nostro urlometro ha tre stati principali: 1. **Stato di Attesa:** Il gioco è pronto a partire, gli serve solo il segnale di via. La variabile `partitaInCorso` è falsa. 2. **Stato di Conteggio:** La barra di LED avanza. La variabile `partitaInCorso` è vera. 3. **Stato Terminato (Cooldown):** La partita è finita e mostra il risultato con la musica per 5 secondi. Durante questo tempo `partitaInCorso` resta vera, perciò non si può iniziare una nuova partita. Dopo i 5 secondi, `partitaInCorso` torna falsa e il gioco è di nuovo pronto per ricominciare. Attenzione: un urlo lanciato durante questo raffreddamento va perso senza avviso — bisogna aspettare che lo schermo si spenga e urlare di nuovo.

## La Sinergia tra Eventi e Loop

Nei progetti complessi spesso non si usa *solo* un evento o *solo* un loop. Si usa una combinazione intelligente dei due, dove ciascuno fa quello che sa fare meglio: * **Eventi:** Sono perfetti per **avviare** azioni o cambiare stato in modo istantaneo. Nel nostro caso, un evento di rumore forte è il colpo di pistola che dà il via alla partita. * **Loop:** Sono perfetti per **gestire** processi continui che dipendono da uno stato. Il nostro loop principale non avvia il gioco. Quando partitaInCorso è vera, si occupa di tenere il gioco attivo finché il livello di rumore è maggiore o uguale a 5. Quando il rumore scende sotto quel valore, il loop termina e il programma passa alla fase finale della partita.

### La Logica della Nostra Battaglia

La nostra architettura mette insieme tutto quello che abbiamo imparato: * Usiamo una variabile booleana, `partitaInCorso`, per controllare se si può iniziare una nuova partita (gestisce gli stati di **Attesa** e di **Cooldown**). * Verifichiamo se il gioco è iniziato e, finché il livello di rumore è maggiore o uguale a 5, la barra continua ad avanzare di **un LED ogni mezzo secondo** (gestisce lo stato di **Conteggio**, quello che accende lo schermo LED). * Un **evento** di rumore forte mette in moto il gioco attivando `partitaInCorso`. L'evento scatta sempre; è il **loop principale** che gli dà retta solo quando siamo in **Attesa**. * Il **loop principale** si occupa del resto: aggiorna lo schermo se stiamo **Contando** e, quando il rumore si ferma, passa allo stato **Terminato**.

## Sviluppo (20-30 minuti) - Costruzione dell'Urlometro

1. Questo è il progetto più complesso di tutti. Incoraggia gli studenti a costruirlo passo dopo passo e a provarlo spesso.
2. Guidali attraverso **la logica di gestione degli stati, spiegando il ruolo di ogni variabile e come l'evento e il loop collaborano**, come dettagliato qui sotto.

## Chiusura (5-10 minuti) - Vi Siete Diplomati!

1. Una volta che gli urlometri funzionano e gli studenti si stanno sfidando, festeggia il loro traguardo.
2. Guida una riflessione finale: **"Guardate il codice che avete costruito. Ha eventi, loop, variabili, condizionali... ha tutto. Descrivete come funziona, come passa da uno stato all'altro. Congratulazioni, avete programmato una macchina a stati completa e avete portato a termine questo viaggio di introduzione al pensiero computazionale!"**

### Rifletti

Questo programma gestisce diversi "stati". Sai dire quali sono i vari stati e quali variabili li controllano?

Descrivi lo scopo dell'evento e lo scopo del loop principale. Perché sono necessari entrambi affinché questo gioco funzioni correttamente?

Se volessi rendere il gioco più difficile (dover urlare più a lungo perché i LED si accendano), quale valore cambieresti nel codice?