# Progetto Real-Time Embedded Systems
## Assembly Chain

**di**

Carlo Nonato
Lorenzo Fiorani

### Requisiti

Il progetto prevede la simulazione di una catena di montaggio composta da un nastro sul quale vengono trasportati diversi tipi di oggetti e due robot i quali hanno il compito, rispettivamente, di rimuovere gli oggetti non conformi e di montare un componente sopra l'oggetto trasportato.

### Moduli

Il progetto è stato suddiviso nei seguenti moduli:

- **Simulation**: rappresenta il mondo con cui si può interagire.
- **Controller**: rappresenta il circuito integrato in grado di pilotare i robot per eseguire i rispettivi compiti.

### Simulation

Questo modulo posiziona gli elementi grafici sulla scena, gestisce la creazione di un oggetto casuale ogni 5 secondi, permette la cattura di un frame dalla singola telecamera posta al di sopra dei robot stessi e, infine, permette l'accesso ai robot e al nastro trasportatore.

L'oggetto è scelto tra tre possibili forme: rettangolo, ellisse e ottagono. Esso viene generato con dimensioni e colori casuali.

Una particolare attenzione merita l'implementazione del robot. Questa mette a disposizione dei comandi che permettono di compiere delle semplici azioni: afferrare l'oggetto (se ve n'è uno) posto al di sotto del braccio, programmare una rotazione dell'oggetto afferrato che verrà portata a termine in un secondo momento, ruotare il braccio verso il punto di partenza o di arrivo e rilasciare l'oggetto. Tutti i comandi sono bloccanti per il chiamante e, sfruttando il framework dei segnali e degli slot forniti da Qt,

il codice del comando è eseguito dal thread principale (quello che controlla la parte di GUI) e non dal thread chiamante, ottenendo così una simulazione più realistica.

## Controller

Il controller attua la propria logica attraverso tre thread:

- Thread che richiede un frame alla telecamera ogni 100 ms.
- Thread che gestisce il robot delle anomalie.
- Thread che gestisce il robot del montaggio.

I thread sono creati tramite la libreria **pthread** in modo "detached". Per via del funzionamento della chiamata **`pthread_create()`**, i thread sono funzioni statiche della classe **`Controller`**.

## Thread gestore della telecamera

Questo thread si occupa di richiedere un nuovo frame alla telecamera ogni 100 ms e di copiarlo, applicando appositi tagli, in due variabili condivise con i thread gestori dei due robot. Le due immagini sono visibili nei riquadri laterali dopo l'accensione del controller. Per evitare race-conditions, la scrittura di queste variabili è protetta da due critical sections dei semafori privati dei due thread gestori dei robot che, in quanto lettori, potrebbero accedervi nel momento in cui la variabile viene sovrascritta.

Altro compito del thread è quello di segnalare che è disponibile un nuovo frame agli altri thread e ciò è implementato tramite due semafori di sincronizzazione (uno per thread) sui quali viene eseguita la chiamata **`sem_post()`** dopo la fase di acquisizione.

## Thread gestore del robot delle anomalie

Questo thread aspetta la notifica del nuovo frame disponibile dal thread gestore della telecamera, dopodiché entra nella crical section del proprio semaforo privato e copia in una variabile locale il proprio frame.

Sfruttando le funzioni della libreria **OpenCV**, vengono riconosciuti i contorni delle figure presenti nel frame e nel caso questi non descrivano dei rettangoli o dei quadrati, l'oggetto in questione viene scartato inviando gli appositi comandi al robot.

## Thread gestore del robot di montaggio

Il funzionamento è identico al precedente thread nella fase di sincronizzazione, dopo la quale viene calcolata la rotazione dell'oggetto di passaggio e la medesima viene applicata all'elemento da montare. Quando l'oggetto si trova in posizione, cioè quando è allineato

con il centro del braccio del robot, l'elemento viene rilasciato e il robot viene riposizionato al proprio zero.