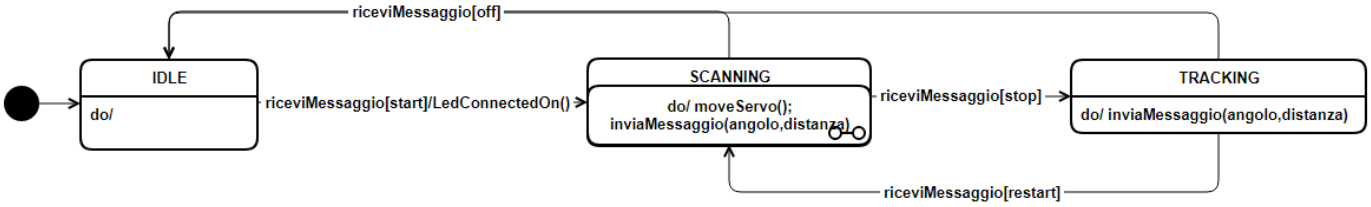
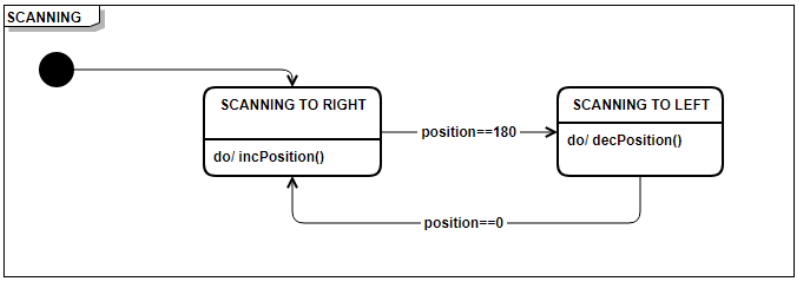
RELAZIONE CONSEGNA N° 3
CORSO SISTEMI EMBEDDED E IOT

Riccardo Marchi, Marco Modigliani, Simone Venturi

Il progetto Smart Radar consiste nel realizzare un radar intelligente utilizzando due sistemi embedded: un Raspberry Pi e un Arduino. La logica dell'applicazione e il controllo di essa, vengono gestiti interamente tramite il microprocessore mentre il microcontrollore si occupa solamente di analizzare l'ambiente circostante e inviare i dati rilevati al sistema di controllo. Questa interazione fra i due sistemi avviene mediante seriale.



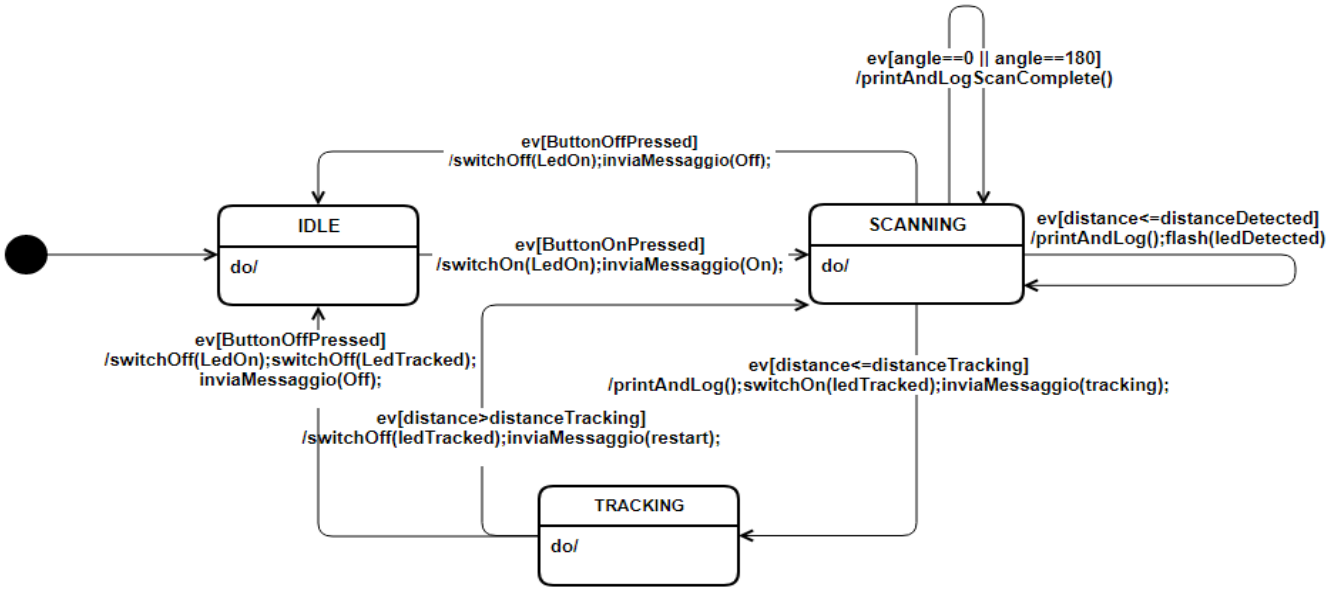


ANALISI COMPORTAMENTO ARDUINO

Come osservato in precedenza il compito svolto da Arduino è molto basilare. In fase di progettazione si era pensato di adottare una soluzione senza scheduler né task in quanto il compito del microcontrollore si può definire con la sola macchina a stati riportata sopra. Durante la fase di testing sono state notate interferenze fra il servomotore e il sensore di prossimità, infatti quest'ultimo non rilevava oggetti oltre i 70 cm. Cambiando soluzione e adottando uno scheduler, separando così il task che pilota servomotore e sensore di prossimità da quello che monitora la seriale, è stato risolto il problema anche se alcune interferenze rimangono. La libreria "Servo.h", usata per pilotare il servomotore, e lo scheduler utilizzano entrambe TIMER1 creando così un conflitto. È stata adottata la libreria "ServoTimer2.h" la quale, come descritto già dal nome, per pilotare il servo fa uso del TIMER2. A questo punto il sensore lavora dentro il proprio range di specifica, ma a distanze oltre il metro fornisce dati inconsistenti e imprecisi. A questo ultimo problema viene posto rimedio scartando, a livello di logica su Raspberry, valori oltre i XXX. Tornado al comportamento di Arduino, nella fase di setup viene letto il valore di un potenziometro usato per definire la velocità angolare del servo motore, questo dato va ad influire sul tempo di

scheduling del task del servomotore aumentandolo per rendere la velocità minore. Una volta ricevuto il messaggio di partenza tramite seriale, si passa allo stato SCANNING accendendo il led di connessione. Qui il microcontrollore non fa altro che muovere il servomotore, prendere il valore del sensore di prossimità, creare una stringa di messaggio (formata dall'angolo del servo e dalla distanza rilevata) e inviarla al Raspberry, il quale in base al contenuto prenderà delle decisioni. Se arriva il messaggio di tracking, Arduino ferma il servomotore continuando però a rilevare la distanza e a inviare un messaggio al microprocessore. In qualsiasi stato in cui si trovi, se viene ricevuto il messaggio "Off", il microcontrollore ferma ogni sua attività e si riporta nello stato IDLE.

ANALISI COMPORTAMENTO RASPBERRY PI



L'architettura dell'applicazione che gira su Raspberry si basa su una macchina ad eventi. In questo caso vengono considerati solo due tipi di eventi: generati dai bottoni On/Off oppure da messaggi. Ogni volta che viene rilevato un evento viene inserito in una coda bloccante, in modo tale che, in caso di assenza di eventi, il sistema rimane fermo e non continua a ciclare per controllare se è successo qualcosa. Questo porta a diversi vantaggi: non vengono persi eventi, il sistema può essere al più in ritardo sulla gestione di essi nel caso arrivassero più messaggi durante una fase di calcolo, inoltre, non si spreca CPU inutilmente per gestire un while(true){}. Ricevuto l'evento scatenato dal bottone On, il sistema sveglia Arduino, accende il led ON e transita in un stato SCANNING nel quale aspetta tutti i messaggi, considerati eventi, che il microcontrollore gli invia tramite seriale. Ogni messaggio, che si trova nella coda degli eventi, viene prima di tutto decodificato e poi analizzato. Dall'analisi del messaggio vengono prese decisioni riguardo tutti gli aspetti forniti da specifica. Si analizza l'angolo per capire se è stato effettuato un giro completo, ma soprattutto si esamina il valore della distanza. Prima di tutto si guarda che questa non rientri in un range di valori troppo alto, cosa che ne determinerebbe l'inconsistenza, poi si effettuano controlli

per determinare se si è rilevato un oggetto (in caso verrà fatto un flash sul led DETECTED) e se questo è vicino a tal punto da passare allo stato TRACKING. Per determinare se due oggetti rilevati sono diversi si guarda se fra loro c'è almeno un angolo dove non viene rilevato alcun oggetto. In questo modo qualsiasi oggetto disposto in obliquo non verrà identificato come un insieme di oggetti, ma nel caso di due oggetti attaccati con distanze dal radar differenti questi verranno identificati come un unico oggetto. Se viene avvistato un corpo con distanza minore o uguale a tracking, viene mandato un messaggio ad Arduino, acceso l'apposito led e vengono di nuovo controllati i messaggi per capire che comportamento tenere. Se la distanza rimane uguale non viene fatto nulla, se questa cambia rimanendo dentro il range di tracking viene solo stampata a video, se è fuori questo range si ritorna allo stato SCANNING e si invia un messaggio ad Arduino. In ogni stato diverso da IDLE con l'evento di pressione del tasto Off si ritorna allo stato iniziale avvisando il microcontrollore.