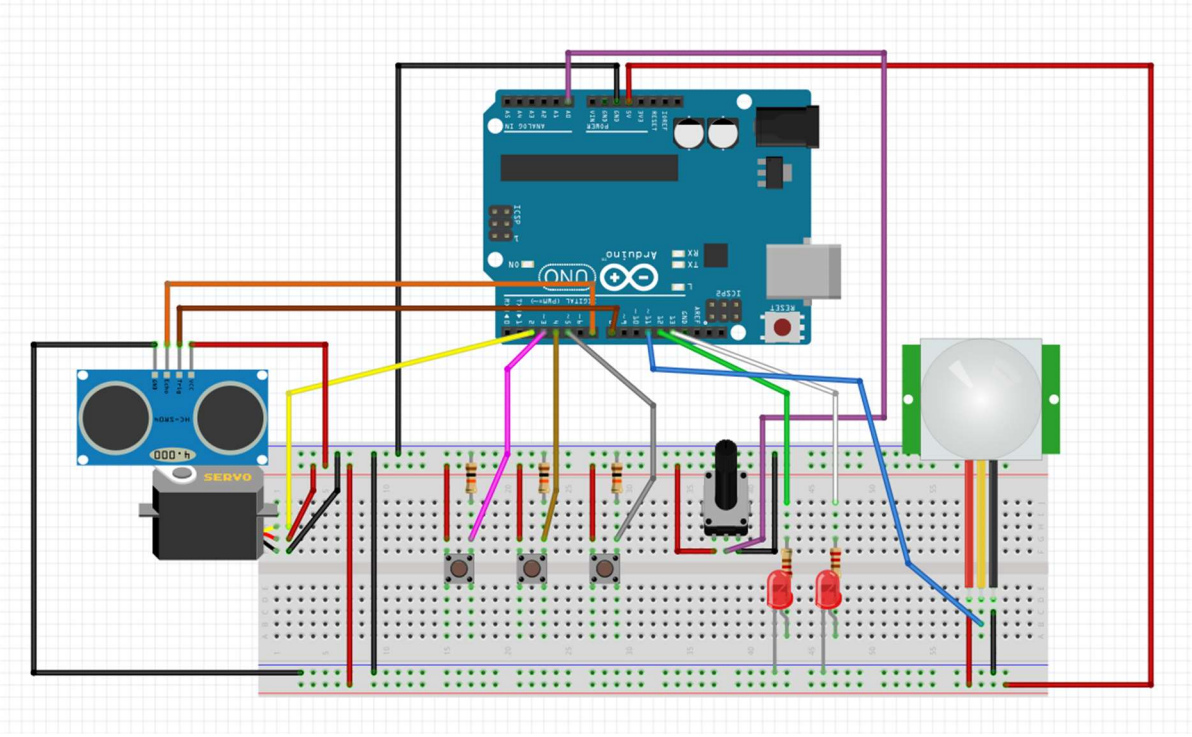
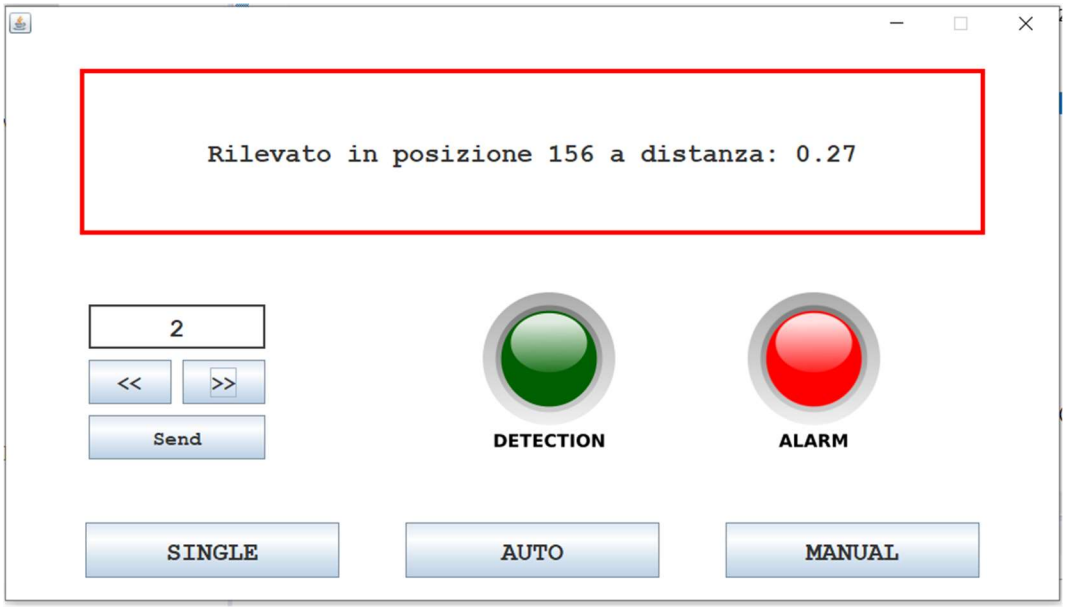
RELAZIONE Progetto #2 - *Smart Radar*

Autori: Matteo Ragazzini, Marta Spadoni

Le specifiche del progetto richiedono di realizzare un sistema embedded che implementi uno smart radar il quale può operare in tre diverse modalità: manual, single e auto. L'insieme dei sensori e degli attuatori utilizzati sono: un sonar montato su un servo motore, un pir, tre interruttori tattili, un potenziometro e due led rossi. Riportiamo di seguito lo schema del circuito realizzato in Fritzing.



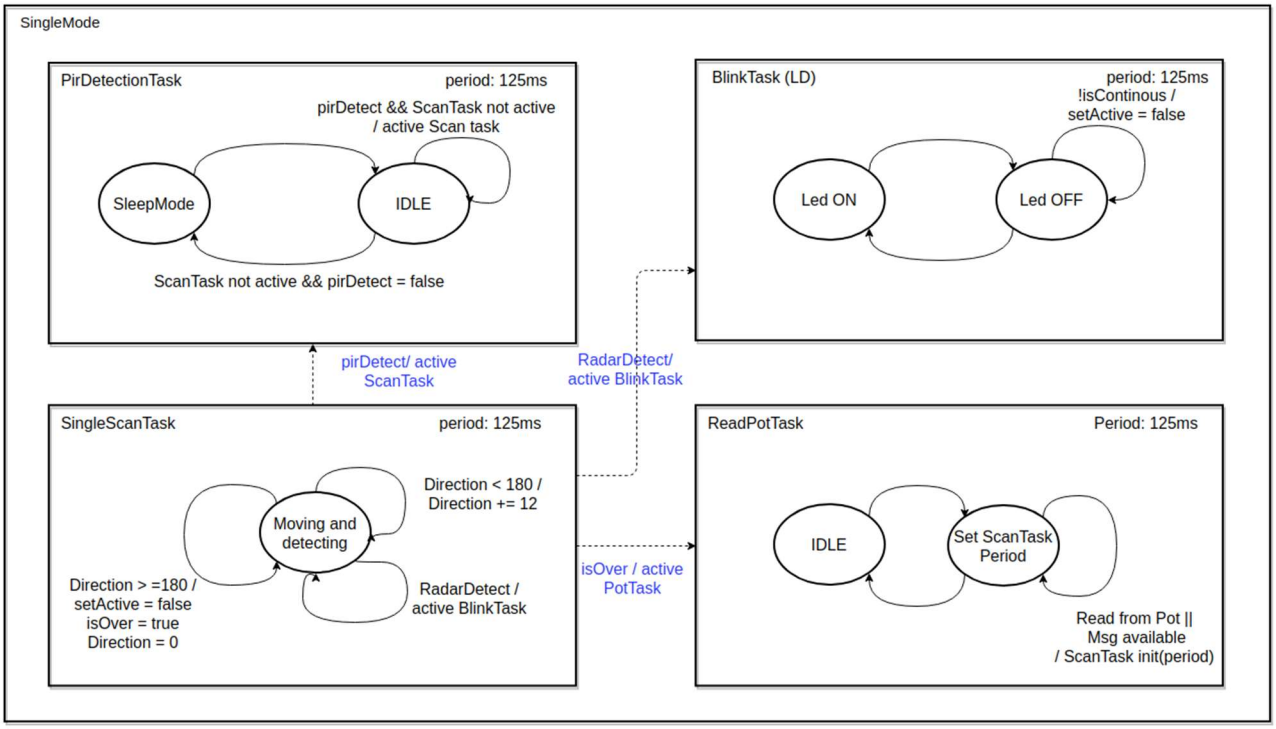
Il sistema è inoltre collegato via Seriale al PC dove è in esecuzione Console, un programma Java dal quale è possibile: scegliere la modalità di funzionamento, direzionare il servo motore se il sistema è in modalità manual, scegliere la velocità di scansione nelle modalità single e auto e infine visualizzare i risultati dei rilevamenti effettuati dal sonar e l'eventuale stato di allarme.



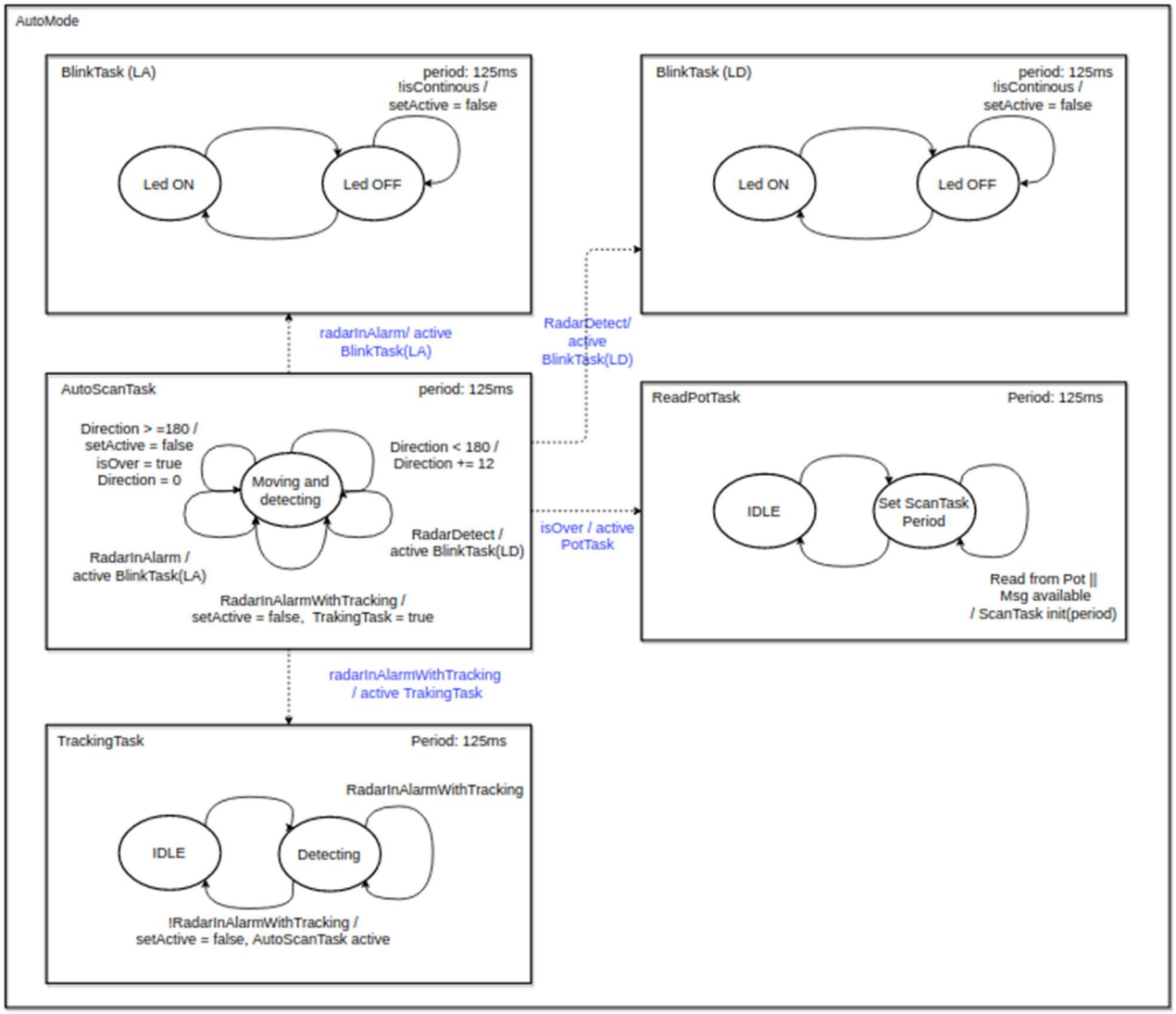
Nelle specifiche viene richiesto di utilizzare un approccio a Task e di modellare il sistema con macchine a stati finiti, abbiamo quindi iniziato la fase di progettazione esaminando ogni modalità e individuando nello specifico i vari task e le condizioni per passare da un task all'altro.

Nella modalità single è inizialmente attivo solo il Pir, quando questo rileva un movimento il Radar, cioè l'accoppiata servo-sonar, si attiva e inizia la *scansione*. La scansione consiste nell'individuare nell'arco di 180° l'eventuale presenza di oggetti. Da specifiche l'arco di spostamento del motore deve essere suddiviso in N direzioni, nel nostro caso 16, e per ognuna di esse il sonar deve effettuare un rilevamento. Decidendo di avere 16 step in ogni scansione il motore si sposta ogni volta di 12° ($\lceil\frac{180}{16}\rceil$).
Il Radar per ogni step della scansione riporta su Console il risultato del rilevamento, in caso positivo riporta sia la distanza che la direzione in gradi a cui è stato individuato l'oggetto e inoltre attiva il blink del led LD.
Il tempo impiegato dal sistema per effettuare una scansione è variabile, in particolare può essere impostato tramite potenziometro o via Console. Nella nostra implementazione il tempo minimo per effettuare una scansione totale è di 2 secondi mentre il tempo massimo è 10 secondi. Come scelta progettuale decidiamo che il nostro radar ha 5 tempi di scansione (2, 4, 6, 8 e 10 secondi) quindi, nel leggere il potenziometro mappiamo il valore rilevato in un range da 1 a 5. A fronte di questa analisi lo stato della scansione è rappresentato dalla posizione del servo e dovendo effettuare, nel caso più veloce, 16 rilevamenti in 2 secondi il periodo base con cui si effettua uno step della scansione è $\frac{2}{16} = 0{,}125s$. Avendo individuato dei tempi di scansione multipli di 2, la variazione del periodo della stessa, per avere scansioni più lunghe, diventa semplicemente il valore del potenziometro mappato nel range per il periodo base.
In questa modalità è inoltre richiesto di implementare un risparmio energetico, così da ridurre il consumo fino a quando il pir non rileva un movimento. Per implementare questa parte di specifiche abbiamo pensato a due diverse soluzioni che bilanciano diversamente l'effettivo risparmio energetico ottenuto e la reattività del sistema. Nella prima soluzione abbiamo determinato la possibilità di mandare il microcontrollore nella sleep mode Power Down, questa è appunto la modalità di sleep che comporta un risparmio energetico più elevato potendo essere disattivata solo da interruzioni esterne, ciò porta però alla perdita della possibilità di switchare fra le modalità tramite GUI e pulsanti. Nella seconda soluzione, quella poi effettivamente implementata, abbiamo deciso di mandare l'ATMEGA in Idle Mode, in questo modo si ha sì un risparmio energetico minore ma il sistema rimane assolutamente reattivo venendo risvegliato dall'interruzione generata da Timer1, lo stesso timer che determina l'attività di scheduling svolta dallo scheduler.
La nostra implementazione della SingleMode è quindi caratterizzata da tre task principali: PirDetectionTask dove implementiamo la sleep mode e verifichiamo se il pir abbia rilevato un movimento o meno, PotTask dove verifichiamo se o tramite seriale o dal potenziometro stesso è stato impostato un nuovo tempo di scansione e infine SingleScanTask dove vengono effettuati i 16 step della scansione.

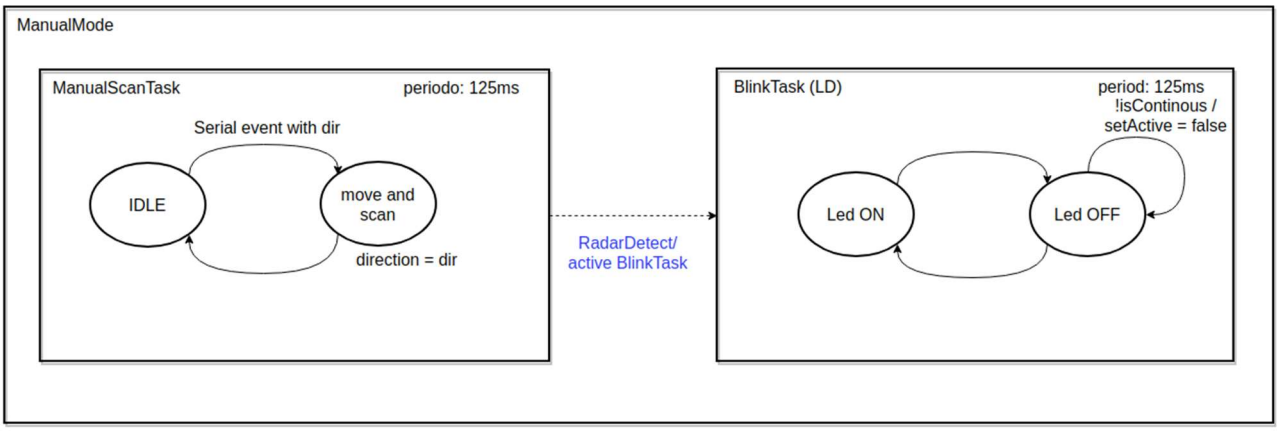
Descriviamo la Single Mode con la seguente macchina a stati finiti.



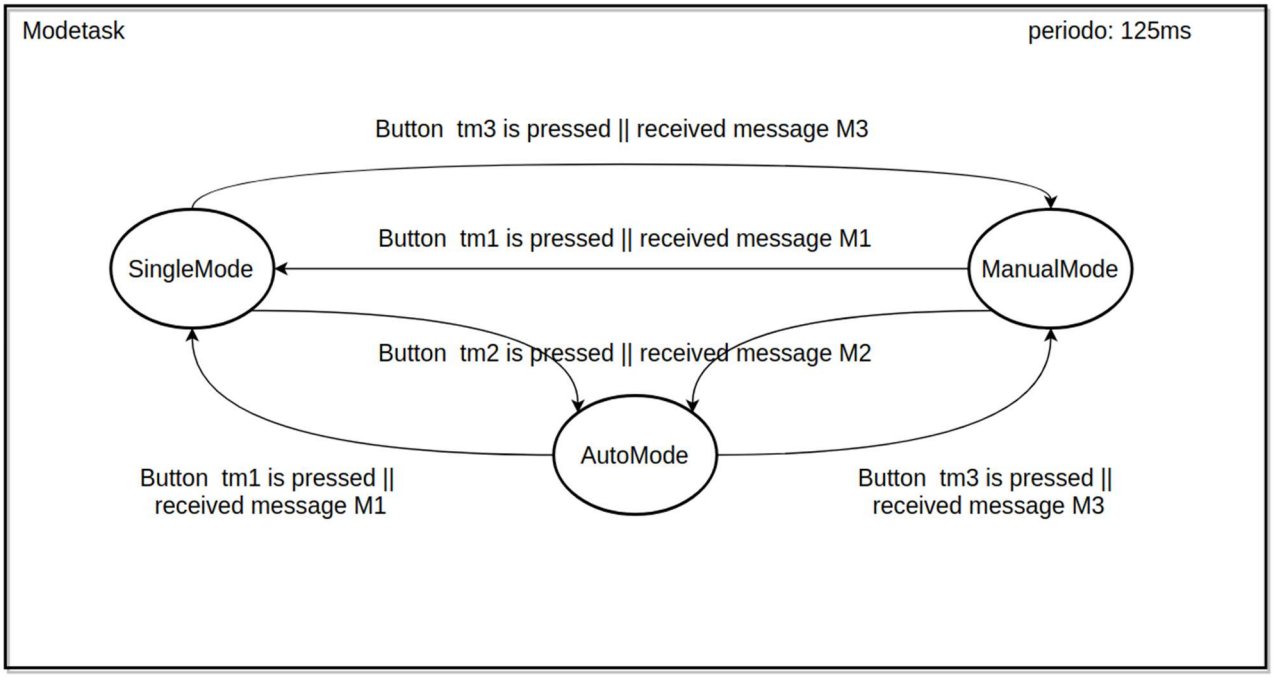
Passando ad esaminare la modalità Auto, indentifichiamo da subito un comportamento comune alla fase descritta precedentemente ovvero quello della scansione con il relativo settaggio della velocità. In questa fase, infatti, si hanno scansioni eseguite in maniera continua ma quando il radar rileva un oggetto ad una distanza compresa tra D_{near} e D_{far}, per noi rispettivamente 0,20 e 0,40 metri, si entra in uno stato di allarme, questo viene segnalato sul circuito attraverso il blink continuo del led LA mentre su Console dal lampeggio continuo del led rosso; durante lo stato di allarme le scansioni continuano e si esce dallo stato solo quando una scansione successiva non rileva un oggetto alla distanza sopracitata. Nel nostro sistema, inoltre, il radar è in grado di entrare in uno stato di allarme con Tracking. Quando il sonar rileva un oggetto a distanza minore di D_{near} la scansione segnala lo stato di allarme e ferma la rotazione del radar per effettuare rilevamenti continui in tale posizione. La fase di Tracking termina quando il radar non rileva più l'oggetto ad una distanza pericolosa.
La modalità AutoMode presenta quindi una sua versione della scansione, AutoScanTask, il TrackingTask, attivato nella fase di Tracking, e un ulteriore BlinkTask continuo per il led dell'allarme.



Nella modalità Manual invece il concetto di scansione cambia perché in questa fase è l'utente, tramite Console, a determinare in quale direzione il motore deve posizionarsi per effettuare un rilevamento.



Il sistema parte in modalità Manual ma è in grado di switchare da una modalità all'altra quando il bottone corrispondente ad una certa fase viene premuto. Per questo motivo, dopo la fase di progettazione, ci è venuto naturale definire il concetto di Modalità (*Mode*) come oggetto con associati l'insieme di task da attivare e disattivare e il bottone che ne determina l'attivazione. Inoltre, abbiamo definito un task ModeTask nel quale facciamo il check se o tramite seriale o tramite la pressione di uno degli interruttori tattili è stato dato l'input per cambiare modalità, in tal caso la modalità corrente viene spenta e quella selezionata viene attivata,



Passando al livello architetturale del nostro sistema abbiamo deciso di utilizzare uno scheduler di tipo "dinamico" che ci desse la possibilità di evitare la schedulazione dei Task non attivi in quel momento. Inoltre abbiamo cercato di rendere il nostro codice il più riusabile ed estendibile possibile, in particolare, avendo già notato una forte somiglianza tra la scansione effettuata in modalità single e quella in modalità auto abbiamo deciso di utilizzare il pattern Template Method, realizzando la classe astratta Scan dove il metodo tick() viene utilizzato come template, in questo ultimo vengono richiamati i metodi astratti checkAlarm() e scanOver(), il primo è utile per inserire un'eventuale fase di allarme ed è quindi utilizzato in particolar modo dalla scansione automatica mentre il secondo viene richiamato quando la scansione è terminata, ciò permette a ciascuna modalità di stabilire le operazioni da effettuare nella fase finale. Implementiamo i due metodi nelle classi concrete: SingleScanTask e AutoScanTask.

Per quanto riguarda il collegamento via seriale tra Arduino e Console abbiamo determinato un semplice protocollo di comunicazione così che, sia dal lato Java che da Arduino, attraverso semplici check sulla lettera iniziale del messaggio fosse possibile capire il destinatario dello stesso per poi procedere con diverse decodificazioni. In particolare, in Arduino, esaminando le classi per la comunicazione via Seriale messe a disposizione, abbiamo notato la presenza di metodi che usano un Pattern, un'interfaccia funzionale, questi ci hanno facilitato la messa in atto di tale sistema di comunicazione. Riportiamo un riassunto del protocollo utilizzato.

Messaggio	Significato
C	Fase di calibrazione terminata
MindiceModalità	Switch in modalità (0-Single, 1-Auto, 2-Manual)
Sangolo	Spostamento del servo
R:angolo:distanza	Rilevamento
PvaloreRange	Velocità (potenziometro via console)
A1/A0	Fase di allarme on/off