Progetto 2 - IoT

Analisi

Nella prima fase del progetto abbiamo preso in considerazione tutti gli aspetti e requisiti posti dalle specifiche per valutarne le priorità, dipendenze e relazioni.

Infatti, il progetto dovrà basarsi su una struttura a macchine a stati finiti sincrone utilizzando un approccio a Task.

Design

Nella progettazione della soluzione abbiamo posto come punto di partenza l'utilizzo di uno scheduler sincrono il cui compito è quello di temporizzare l'esecuzione dei diversi task che compongono il progetto.

Iniziando ad analizzare più nel dettaglio il comportamento richiesto dalle specifiche, incominciamo a definire i macro stati che potremo identificare come Task. Di questi, tre si riferiscono rispettivamente alle modalità *Single*, *Manual* e *Auto*, mentre un quarto si propone all'implementazione della funzionalità di cambio modalità.

Ogni macro stato si compone di microstati in cui il sistema di può trovare per quella specifica modalità. Per esempio, la modalità *Manual* si compone degli stati: RECEIVING, MOVE, SCAN. All'interno di ogni task il microcontrollore più cambiare stato in base alle condizioni attuali dell'ambiente d'esecuzione, garantendo sempre almeno un punto di entrata e almeno un punto di uscita.

Nella progettazione dell'interazione con la porta seriale di Arduino ci siamo preposti dell'utilizzo di un applicativo grafico sviluppato in Java. Questo permetterà al suo utilizzatore di cambiare la velocità di scansione, cambiare la direzione del servomotore, cambiare la modalità corrente e analizzare i dati trasmessi dal componente sonar.

Sviluppo

Le scelte implementative più rilevanti che attuate ci hanno portate alla soluzione preposta sono definite di seguito.

Scelta dei task e progressione tra stati

In una prima versione si era pensato si avere un quinto task, il lampeggio del led, che dopo aver

Ogni task è progettato in modo tale che tra uno stato e l'altro intercorre un tick dello scheduler, in questo modo si è potuto implementare direttamente il lampeggio del led senza l'utilizzo di ulteriore task. Infatti analizzando il comportamento del lampeggio questo viene utilizzato anche per riempire i tick "vuoti" effettuati per raggiungere il tempo di completamento totale della scansione.

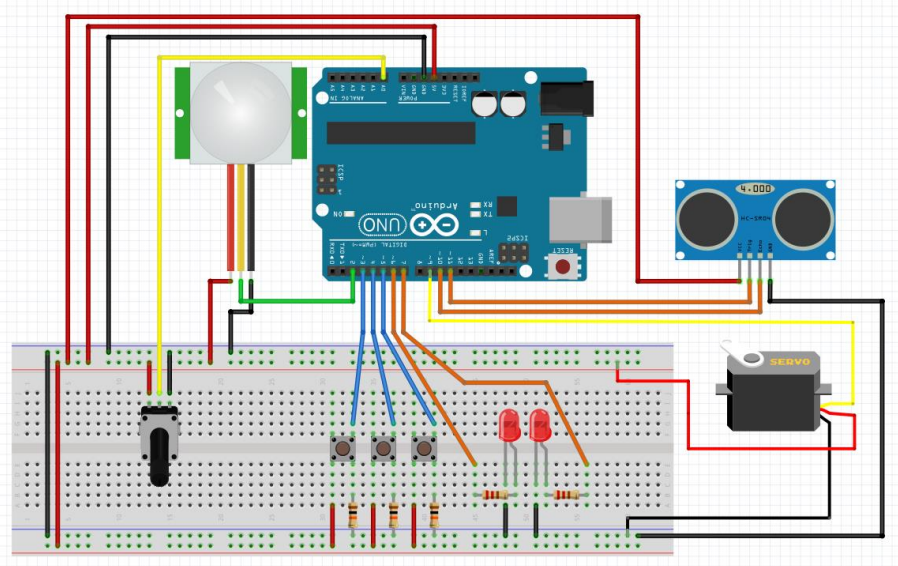
Risveglio dalla modalità risparmio energetico

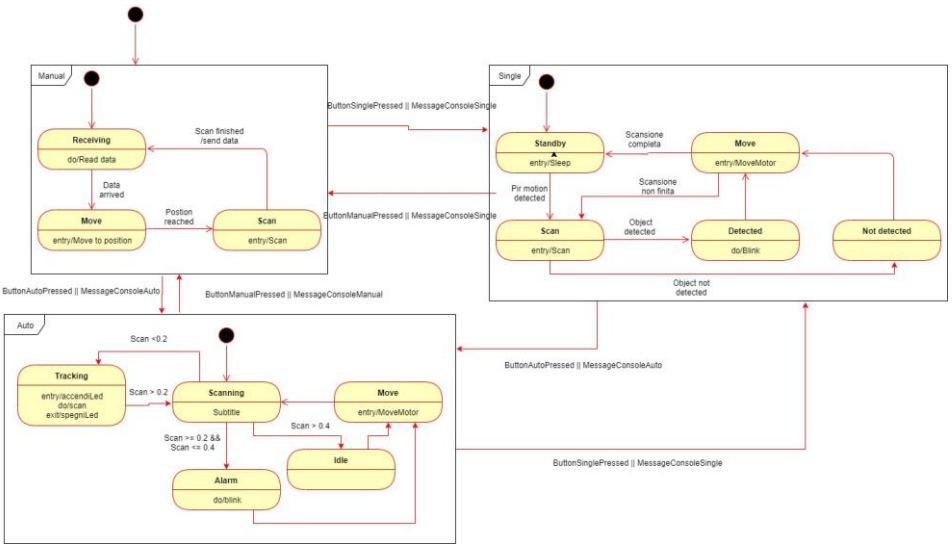
Per ottenere il maggior risparmio energetico è stata fatta la scelta di utilizzare la "sleep mode" più impattante per il sistema, la "SLEEP_MODE_PWR_DOWN". Infatti da questo

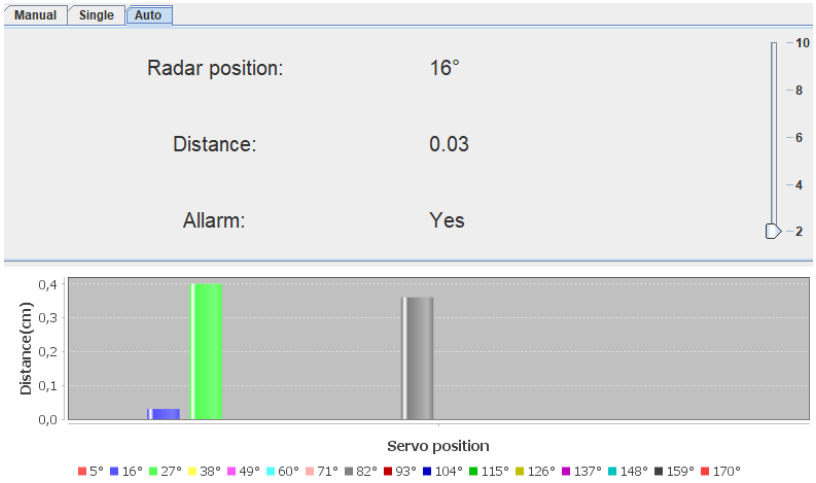
stato che si attiva nello stato di Standby della modalità Single ci si può risvegliare tramite interrupt. Abbiamo utilizzato la libreria “EnableInterrupt” per riuscire appunto ad abilitare le interruzioni sui pin su cui abbiamo collegati i bottoni.

Gestione dei componenti durante gli stati

Abbiamo prestato particolare attenzione al fatto che componenti come il servomotore durante i punti morti del funzionamento delle modalità rimanessero accessi e quindi continuamente in tensione, abbiamo ovviato a questo particolare problema aggiungendo un comportamento generale a livello di task, in cui se questo viene attivato allora attiva anche i componenti necessari, viceversa alla disattivazione.




Manual
Receiving
do/Read data
Scan finished
/send data
Data
arrived
Move
entry/Move to position
Postion
reached
Scan
entry/Scan
ButtonSinglePressed || MessageConsoleSingle
ButtonManualPressed || MessageConsoleSingle
Single
Standby
entry/Sleep
Scansione
completa
Move
entry/MoveMotor
Pir motion
detected
Scansione
non finita
Scan
entry/Scan
Object
detected
Detected
do/Blink
Not detected
Object not
detected
ButtonAutoPressed || MessageConsoleAuto
ButtonManualPressed || MessageConsoleManual
Auto
Scan <0.2
Tracking
entry/accendiLed
do/scan
exit/spegniLed
Scan > 0.2
Scanning
Subtitle
Move
entry/MoveMotor
Scan > 0.4
Scan >= 0.2 &&
Scan <= 0.4
Idle
Alarm
do/blink
ButtonAutoPressed || MessageConsoleAuto
ButtonSinglePressed || MessageConsoleSingle


Manual
Single
Auto
Radar position:
16°
Distance:
0.03
Allarm:
Yes
10
8
6
4
2
0,4
0,3
0,2
0,1
0,0
Distance(cm)
Servo position
5° 16° 27° 38° 49° 60° 71° 82° 93° 104° 115° 126° 137° 148° 159° 170°


Manual
Single
Auto
Move:
<=
180°
=>
Object:
Yes
Distance:
0.49
2,0
1,5
1,0
0,5
0,0
Distance(cm)
Servo position
82° 90° 100° 110° 120° 130° 140° 150° 160° 170°