[Sistemi Embedded e IoT - a.a. 2019-2020]

Progetto #2 - Smart Radar

Alpi Davide 831238

Foschini Francesco 826200

Penazzi Paolo 825618

Descrizione soluzione

Model

Abbiamo sviluppato per prima cosa le classi di model, rappresentanti ogni componente fisico (pulsanti, led, potenziometro, servo, sonar, pir).
Tutti questi elementi sono incapsulati dentro la classe Radar.

Tale classe contiene inoltre importanti informazioni sullo stato del sistema:

- la modalità {SINGLE, MANUAL, AUTO}
- la velocità {ULTRASLOW, SLOW, NORMAL, FAST}
- i comandi ricevuti {MOVE_LEFT, MOVE_RIGHT, MODE_SINGLE, MODE_MANUAL, MODE_AUTO, SPEED_ULTRASLOW, SPEED_NORMAL, SPEED_FAST}
- lo stato di allarme
- l’ultima misurazione effettuata

Tasks

Ogni task possiede un puntatore alla classe Radar, che viene istanziata nel setup.

La scelta del periodo base dello scheduler possedeva 2 vincoli principali. Il primo è che doveva essere sufficientemente alto da permettere un timeout ragionevole alla funzione pulseIn() del sonar. Il secondo è che doveva essere sufficientemente breve per non perdersi pressioni di pulsanti (gestiti sempre tramite task, quindi polling). Abbiamo optato per 50ms, ampiamente sufficienti per misurare diversi metri di distanza col sonar e tali da non perdere mai eventi di pressione.
I nostri pulsanti rimbalzano fino a 100ms, nel caso se ne possiedano di più precisi, si può ridurre tranquillamente il periodo base a 25ms.

I task sono:

- 3 per il polling sui pulsanti, che inviano al Radar il messaggio relativo al cambio di modalità quando rilevano una pressione. **input**
- 1 per il blinking dei led di allarme e di rilevazione. **output**
- 1 per la misurazione della distanza dal sonar **input**
- 1 per la lettura della seriale **input**
- 1 per la scrittura sulla seriale **output**
- 1 task per ogni modalità, che si occupano del movimento del servo e delle logiche proprie di ogni modalità. **output**

I task di input sono schedulati prima dei task di movimento del servo, per evitare comportamenti indesiderati. Così abbiamo anche il vantaggio che si misura col sonar prima di muovere il servo, evitando quindi errori fisici derivati da letture in movimento.
Alcuni task all'interno del proprio tick reimpostano dinamicamente il proprio periodo, in base alla "speed" del Radar. Ad ogni valore dell'enum Speed è associato infatti un intero rappresentante il periodo in ms.

Tenendo come riferimento per uno sweep completo Tmin = 2s e Tmax=10s, e considerando che le posizioni sono 16, abbiamo calcolato i seguenti periodi:

ULTRASLOW = 600ms (Tsweep = 600 * 16 = 9600ms)
SLOW = 450ms (Tsweep = 450 * 16 = 7200ms)
NORMAL = 300ms (Tsweep = 300 * 16 = 4800ms)
FAST = 150ms (Tsweep = 150 * 16 = 2400ms)

Console

La console si occupa di visualizzare tutti i messaggi ricevuti da arduino e di attivare/disattivare lo stato di allarme (contorno rosso sulla text area).

Sono presenti poi pulsanti autoesplicativi che permettono di selezionare modalità e velocità del radar.

Con freccia sx e freccia dx da tastiera è possibile inoltre pilotare il radar in modalità manual.

Siamo a conoscenza di un bug per il quale a volte si freeza la text area e non si aggiorna più con i messaggi che arrivano. In tal caso chiudere e riaprire l'applicazione.

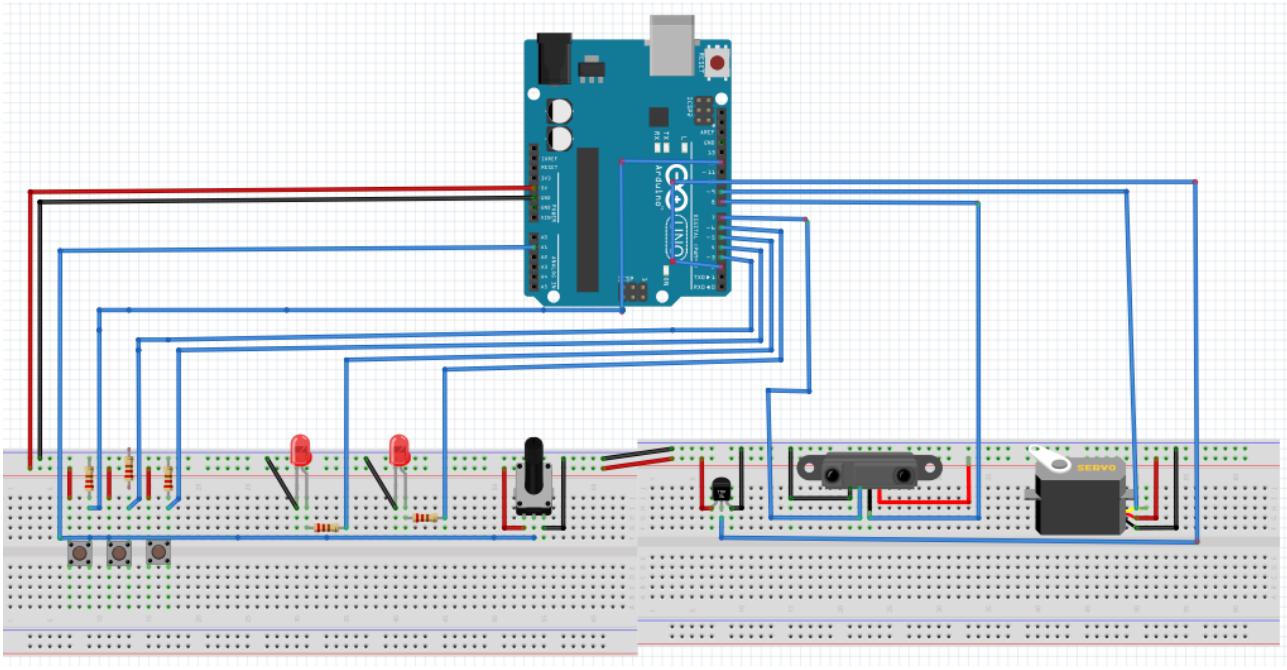
Note sulle specifiche

In modalità *single*, il sistema **effettua una singola scansione e va in modalità risparmio energetico (sleep mode). Si risveglia solo quando viene rilevato** movimento dal pir P. A fronte di questo evento, il sistema esegue una *scansione*.
Notare che mentre il sistema è in sleep, l'unico modo per svegliarlo è tramite il pir.

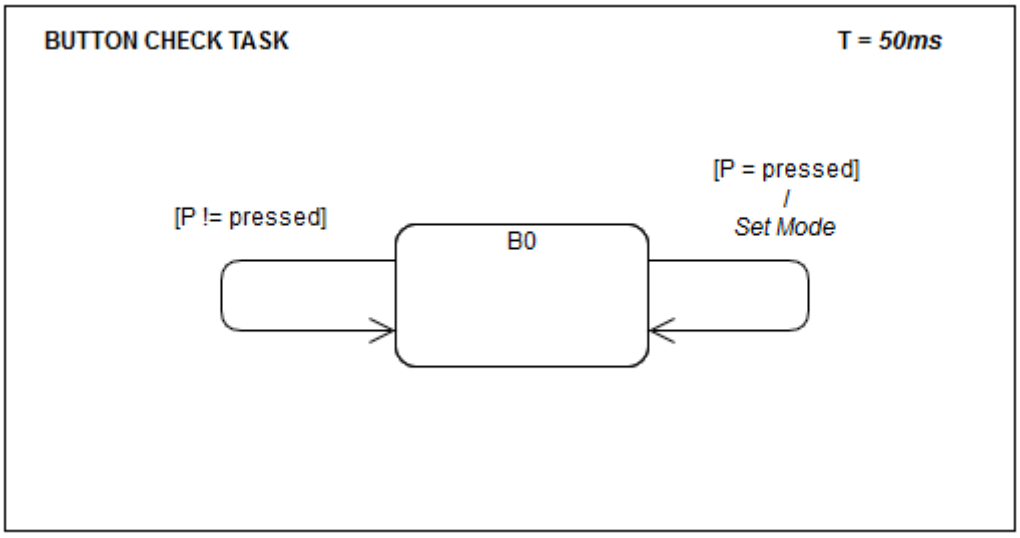
Il led LD si accende solo in modalità *single* quando l'ultima misurazione effettuata ha rilevato un oggetto ad una qualsiasi distanza.

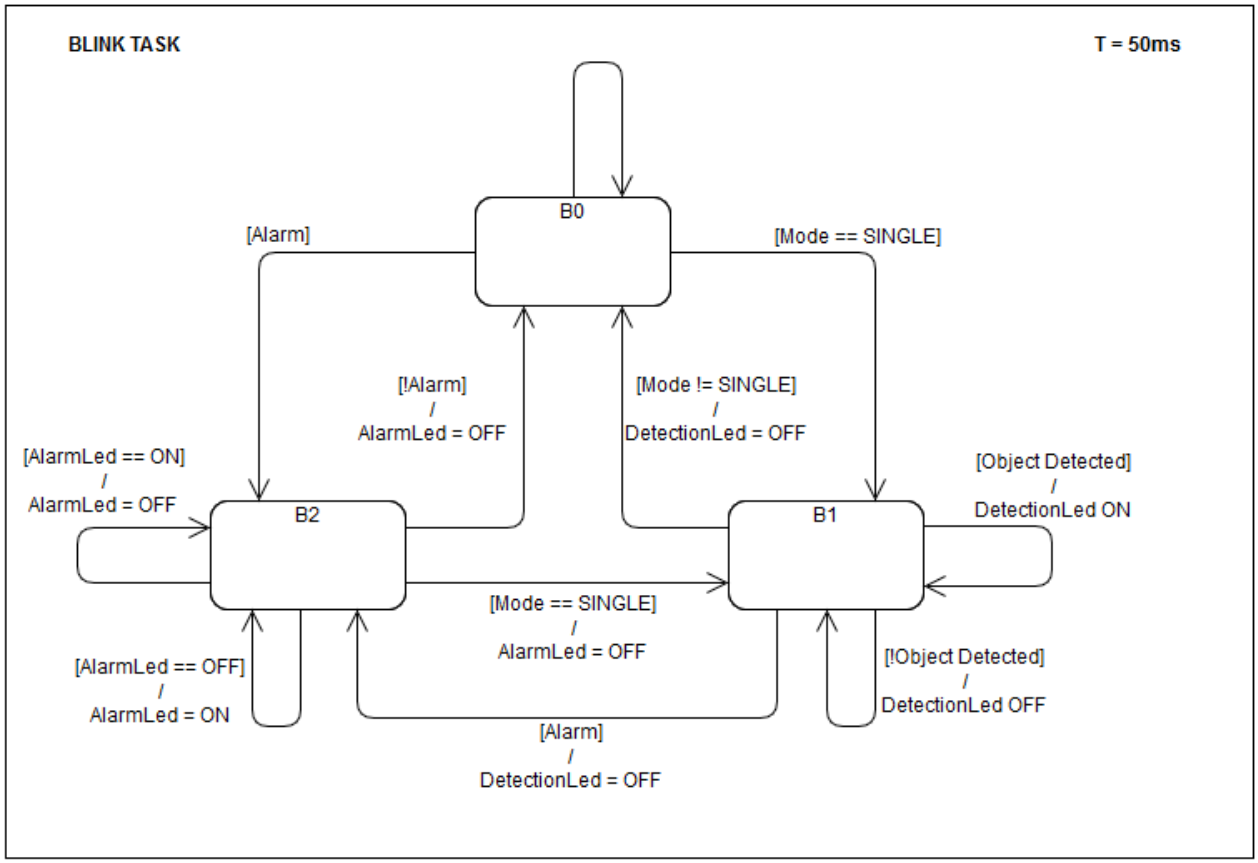
La velocità impostata influisce allo stesso modo sul tempo impiegato per effettuare uno sweep completo in ognuna delle 3 modalità.

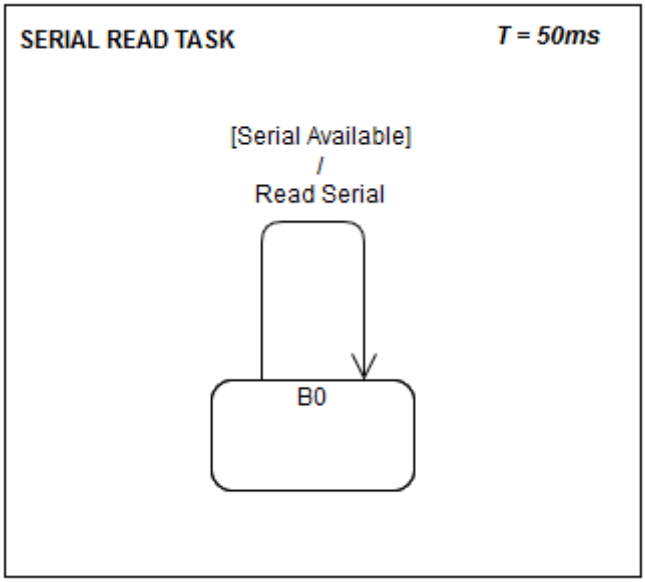
Schema circuito

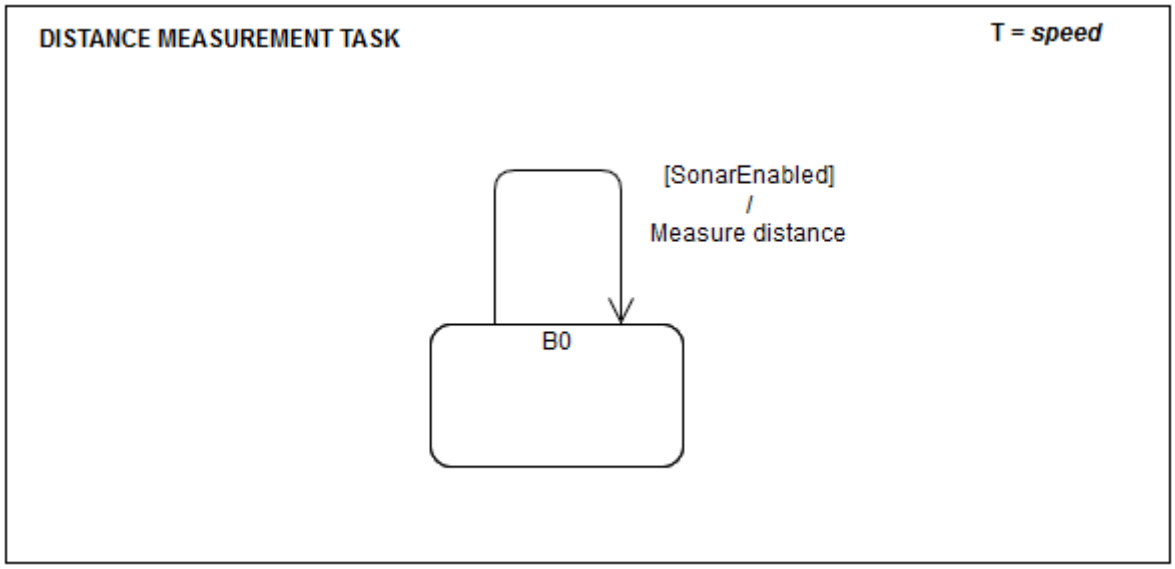


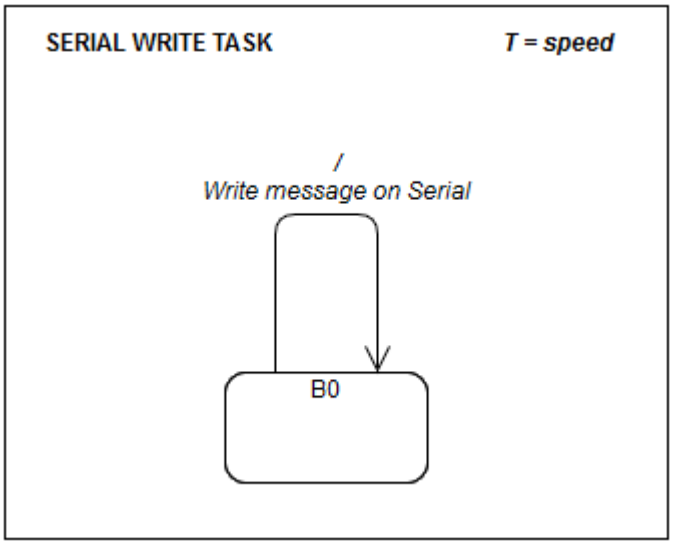
Diagrammi macchine a stati finiti dei singoli task

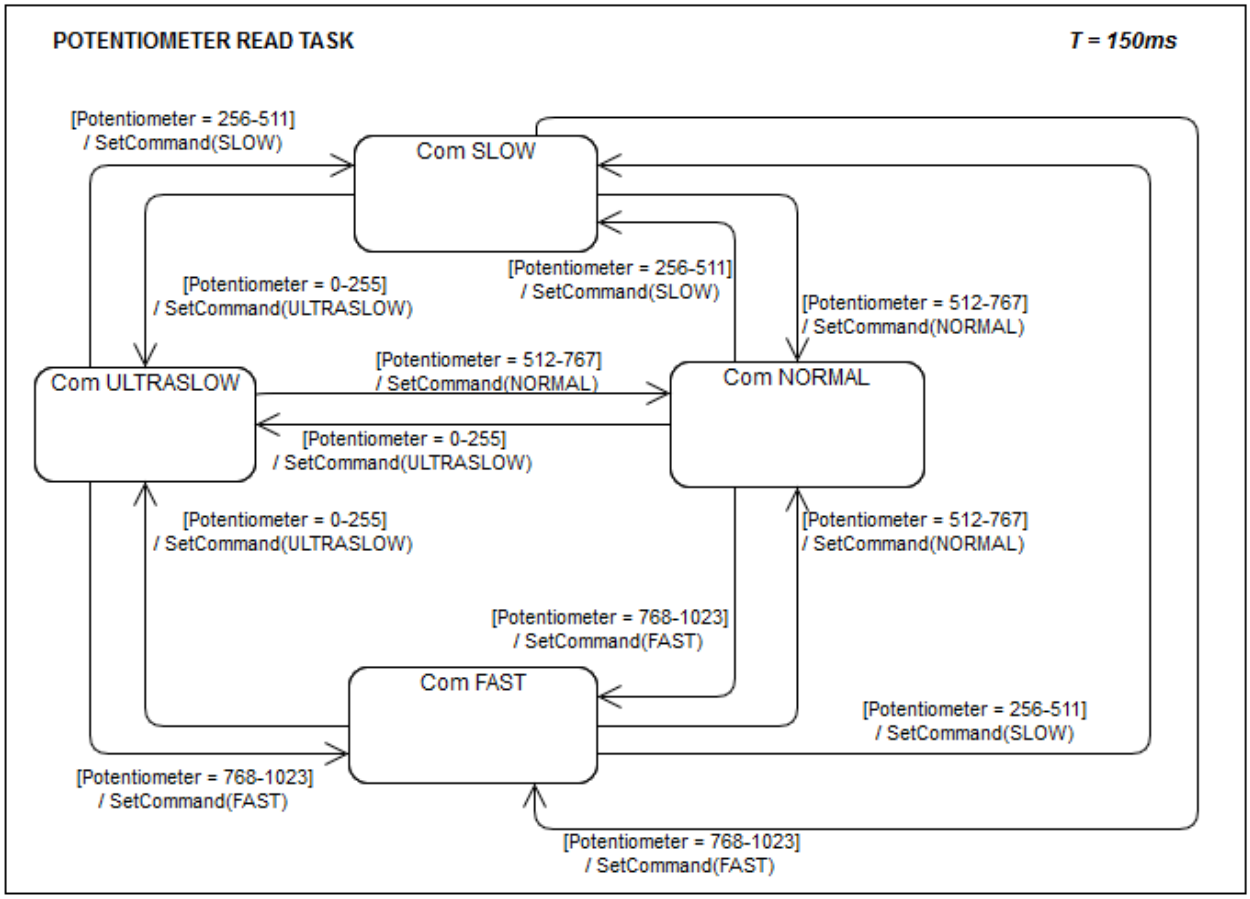


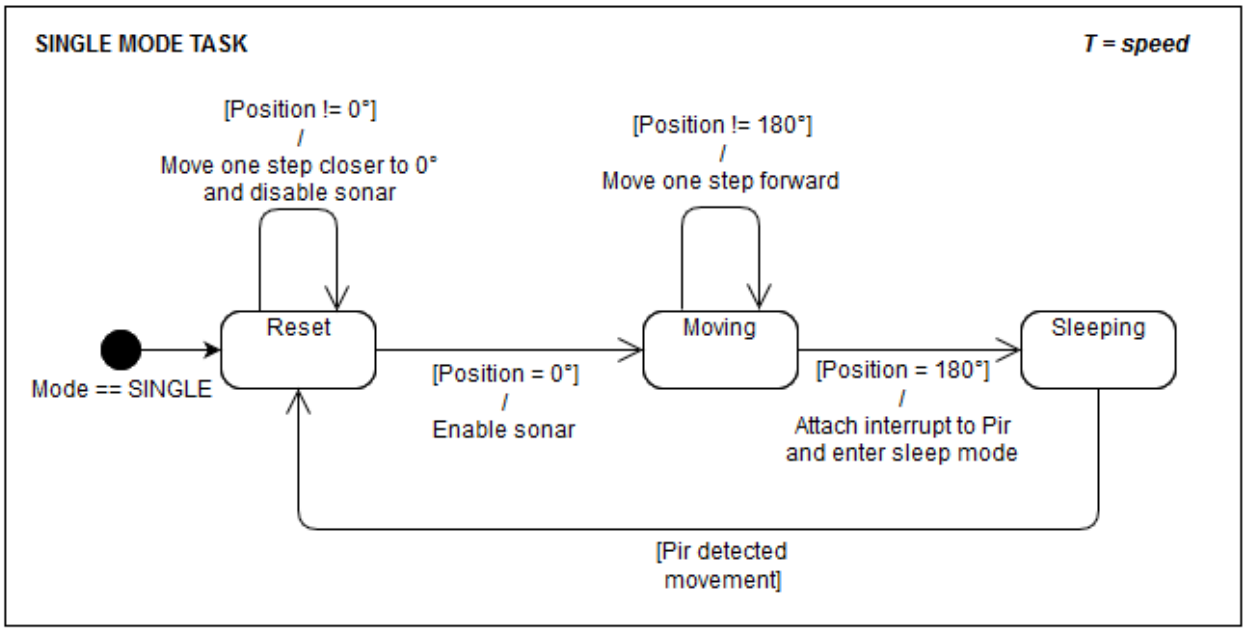


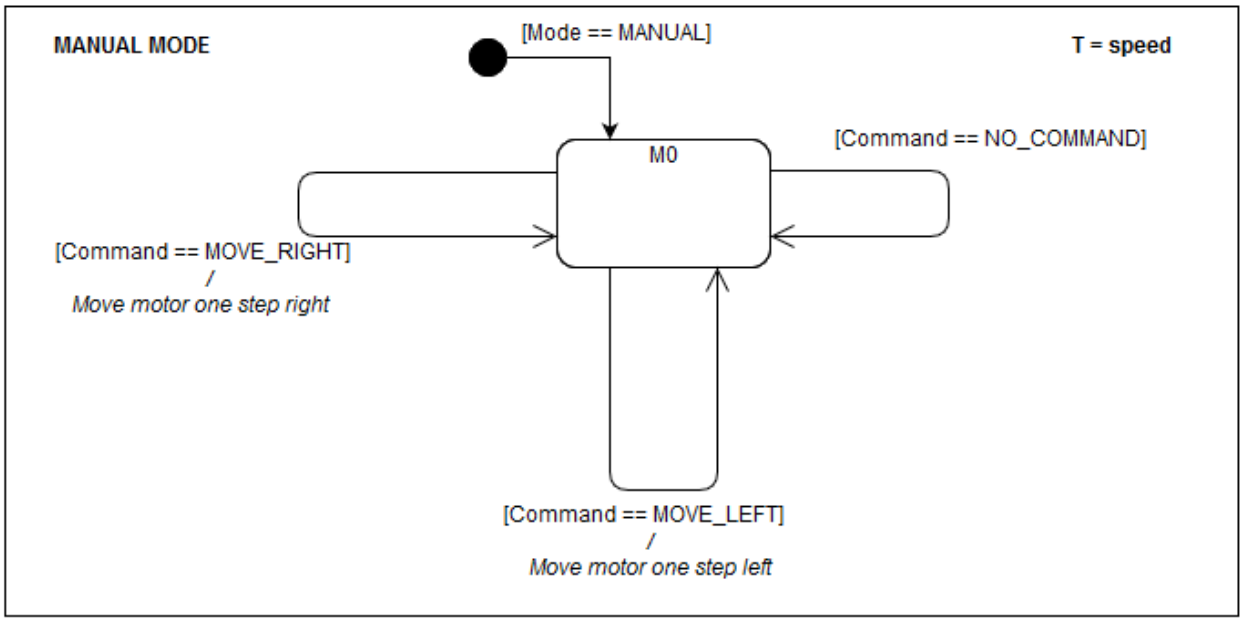

SERIAL READ TASK
T = 50ms
[Serial Available]
/
Read Serial
B0

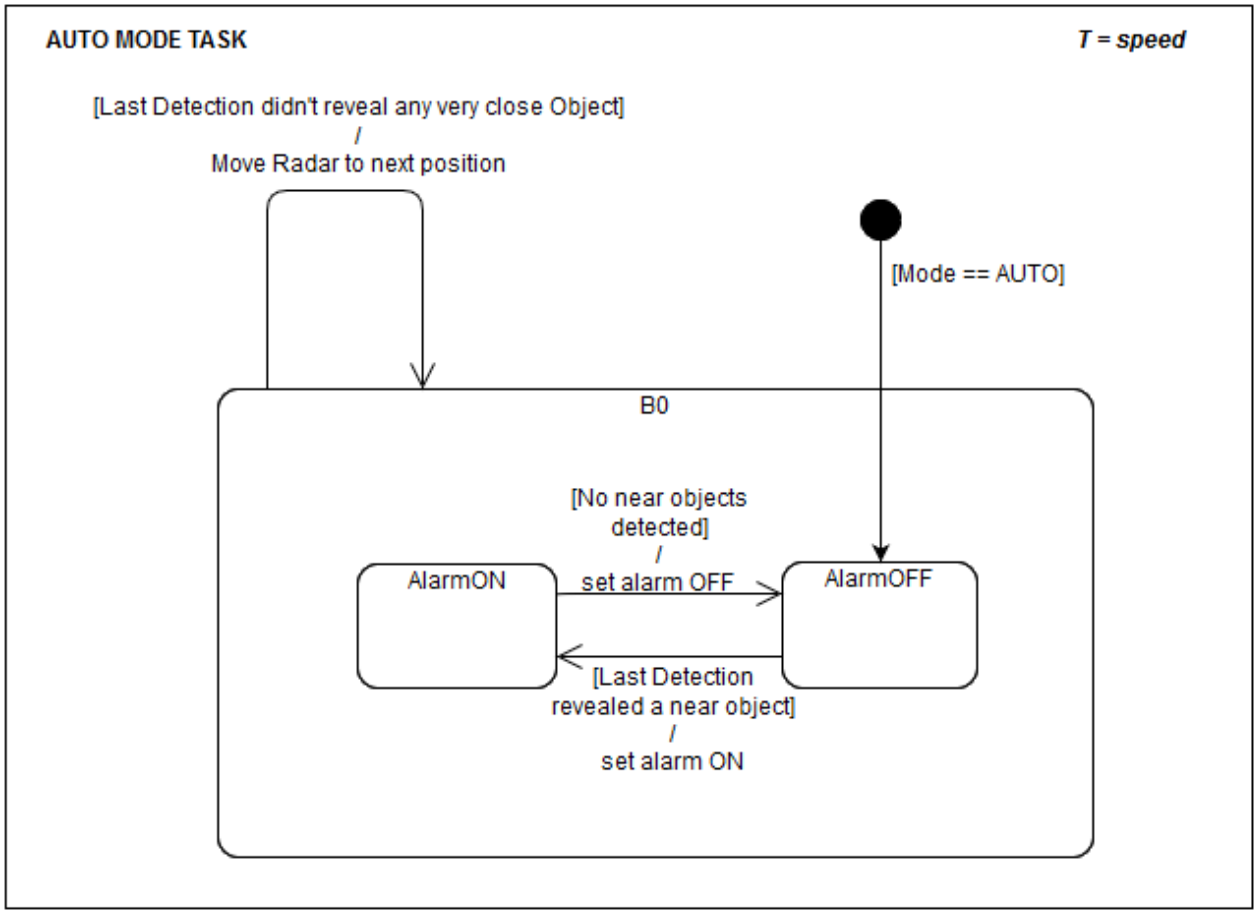

DISTANCE MEASUREMENT TASK
T = speed
[SonarEnabled]
/
Measure distance
B0


SERIAL WRITE TASK
T = speed
/
Write message on Serial
B0


POTENTIOMETER READ TASK
T = 150ms
[Potentiometer = 256-511]
/ SetCommand(SLOW)
Com SLOW
[Potentiometer = 256-511]
/ SetCommand(SLOW)
[Potentiometer = 0-255]
/ SetCommand(ULTRASLOW)
[Potentiometer = 512-767]
/ SetCommand(NORMAL)
Com ULTRASLOW
[Potentiometer = 512-767]
/ SetCommand(NORMAL)
Com NORMAL
[Potentiometer = 0-255]
/ SetCommand(ULTRASLOW)
[Potentiometer = 0-255]
/ SetCommand(ULTRASLOW)
[Potentiometer = 512-767]
/ SetCommand(NORMAL)
[Potentiometer = 768-1023]
/ SetCommand(FAST)
Com FAST
[Potentiometer = 256-511]
/ SetCommand(SLOW)
[Potentiometer = 768-1023]
/ SetCommand(FAST)
[Potentiometer = 768-1023]
/ SetCommand(FAST)


SINGLE MODE TASK
T = speed
[Position != 0°]
/
Move one step closer to 0°
and disable sonar
[Position != 180°]
/
Move one step forward
Reset
Moving
Sleeping
Mode == SINGLE
[Position = 0°]
/
Enable sonar
[Position = 180°]
/
Attach interrupt to Pir
and enter sleep mode
[Pir detected
movement]


MANUAL MODE
[Mode == MANUAL]
T = speed
M0
[Command == NO_COMMAND]
[Command == MOVE_RIGHT]
/
Move motor one step right
[Command == MOVE_LEFT]
/
Move motor one step left


AUTO MODE TASK
T = speed
[Last Detection didn't reveal any very close Object]
/
Move Radar to next position
[Mode == AUTO]
B0
[No near objects detected]
/
AlarmON
set alarm OFF
AlarmOFF
[Last Detection revealed a near object]
/
set alarm ON