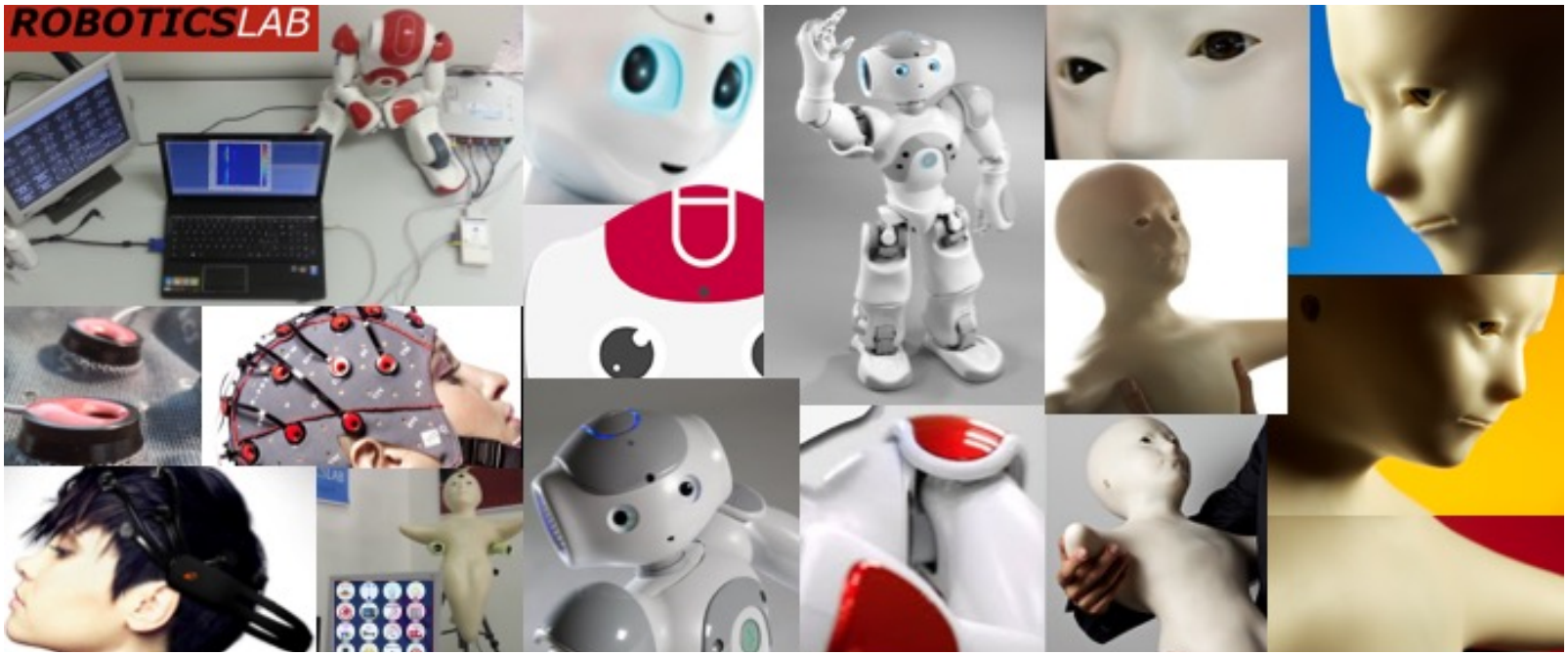


SONAR, LASER E MAPPA IN TEAMBOTS

PROF. ROSARIO SORBELLO

4° LEZIONE – PALERMO 19 OTTOBRE 2022

ROBOT MULTIFORAGEN150EXPLORE

- INTRODUZIONE DI SENSORISTICA + EFFICIENTE

VERSIONE BASE MULTIFORAGEN150:

PER OGNI OSTACOLO RILEVO SOLO IL PUNTO IN CUI QUESTO RISULTA MENO DISTANTE DAL ROBOT STESSO

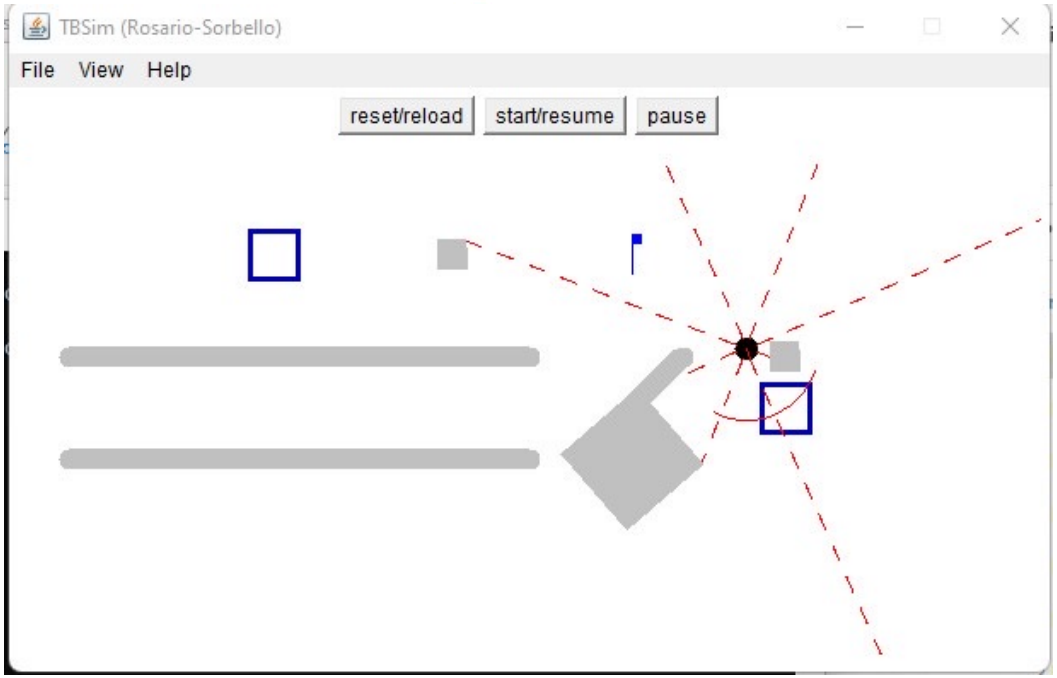
ESEMPIO: 4 PARETI SOLO 4 PUNTI UNO PER PARETE

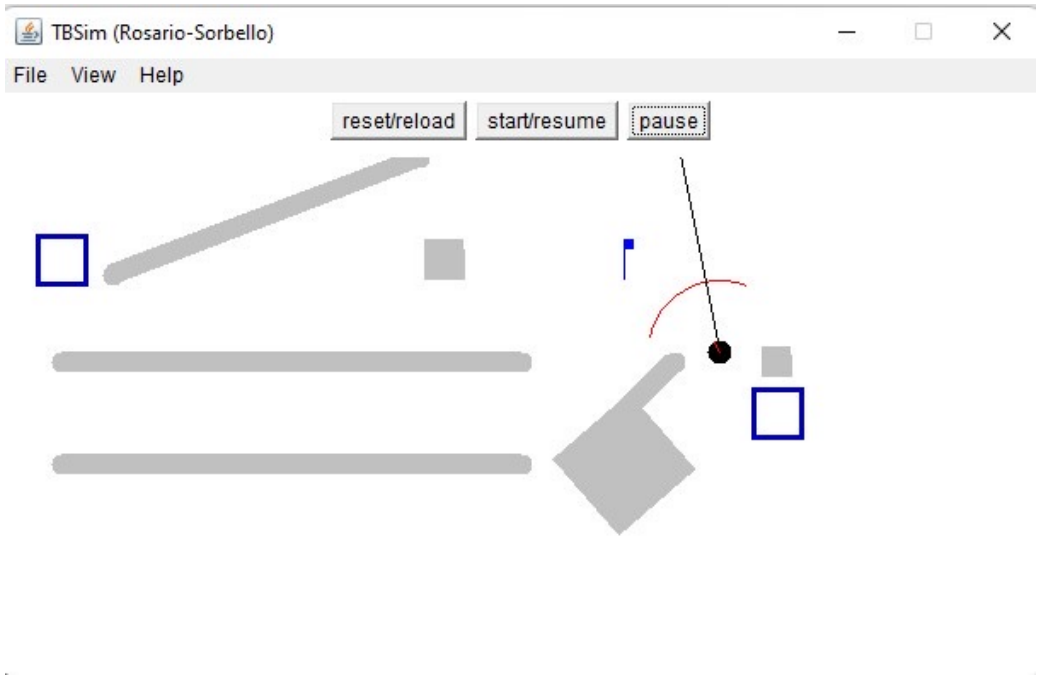
PROBLEMA: VICOLI CIECHI, CORRIDOI CHIUSI O OSTACOLI A V

ROBOT MULTIFORAGEN150EXPLORE

Le caratteristiche che contraddistinguono il **MultiForageN150Explore** sono:

- una cintura di 16 sonar, disposti circolarmente a 360° lungo tutto il corpo del robot,
- un sensore laser ottico, che esegue una scansione mantenendosi all'interno di un angolo di 45° rispetto alla direzione del robot.





SONAR

I sonar disposti sul corpo del **MultiForageN150Explore** permettono di rilevare qualunque ostacolo che si trovi nel loro raggio di azione. È possibile utilizzare tutti i sonar contemporaneamente, una parte o nessuno. Soltanto i sonar accesi vengono utilizzati per individuare gli ostacoli.

Le rilevazioni fatte dal sonar non sono mai precise al 100% ma presentano un certo errore che simula i malfunzionamenti dei sistemi reali che utilizzano i sonar.

Sono comunque presenti tre livelli di precisione (*Low, Medium, High*) che permettono di far variare il valore dell'errore massimo, il quale è influenzato anche una da una componente che è direttamente proporzionale alla distanza dal robot dell'oggetto rilevato.

LASER

Oltre alla cintura di sonar, il **MultiForageN150Explore** è dotato di un laser ottico che effettua una scansione dell'ambiente mantenendosi all'interno di un angolo di 45° rispetto alla direzione del robot. Il laser può essere acceso o spento.

La precisione delle rilevazioni eseguite con il laser è di gran lunga maggiore rispetto a quelle ottenute con il sonar e non dipende dalla distanza dell'oggetto individuato.

C'è da sottolineare il fatto che sia il laser che il sonar possono esclusivamente rilevare la presenza di un ostacolo, ma non possono dire nulla riguardo la natura di tale ostacolo.

IMPIEGO DELLA SENSORISTICA

Le rilevazioni dei sonar e del laser ottico possono essere utilizzate in vari modi:

- per ricostruire l'ambiente in cui il robot si trova a partire dalla posizione degli ostacoli,
- per effettuare un calcolo più preciso delle traiettorie da effettuare per raggiungere un goal evitando efficacemente gli ostacoli (utilizzando il metodo dei campi magnetici per trovare tali traiettorie),
- per cercare eventuali porte o punti di uscita da stanze o labirinti.

Innanzi tutto va detto che sia sonar che laser hanno un indice di errore che dipende dalle proprietà fisiche del sistema stesso. Il laser ha un indice di errore minore del sonar.

L'altra componente dell'errore, per ogni singola rilevazione, è direttamente proporzionale alla distanza dell'oggetto rilevato dal robot.

In questo modo si potrebbe pensare di attribuire un peso ad ogni rilevazione che dipenda appunto dal tipo di sensore utilizzato e dalla distanza alla quale si trova l'oggetto identificato.

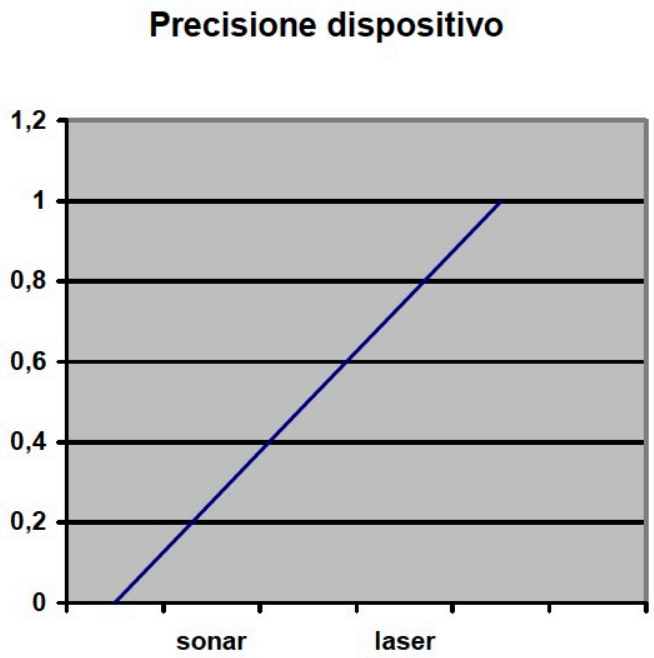
Inoltre, si supponga di voler rappresentare la conoscenza tramite una rappresentazione a griglia del mondo, e che ogni cella relativa ad una zona della mappa contenga una variabile intera `occupied` con valore compreso tra 0 e 1 che quantifica la certezza con cui un ostacolo si trovi in tale cella (il valore 1 corrisponde al fatto certo).

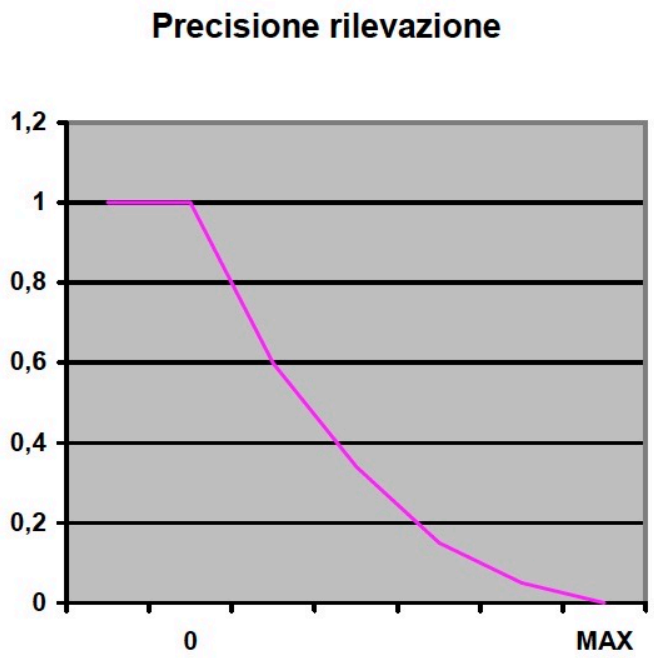
A questo punto, ogni qual volta un ostacolo viene rilevato in una posizione relativa ad una cella, il valore della sua variabile `occupied` viene incrementato del peso assegnato alla rilevazione. Non appena il valore di `occupied` di una cella **X** supera il valore 1, si può affermare che un ostacolo si trova in posizione **X**, e di conseguenza si aggiorna la conoscenza che il robot ha del mondo.

PRECISIONE DISPOSITIVO E RILEVAZIONE

Il primo ha come ascissa il valore associato ai diversi dispositivi sonar e laser, mentre il secondo la distanza relativa dell'oggetto rilevato.

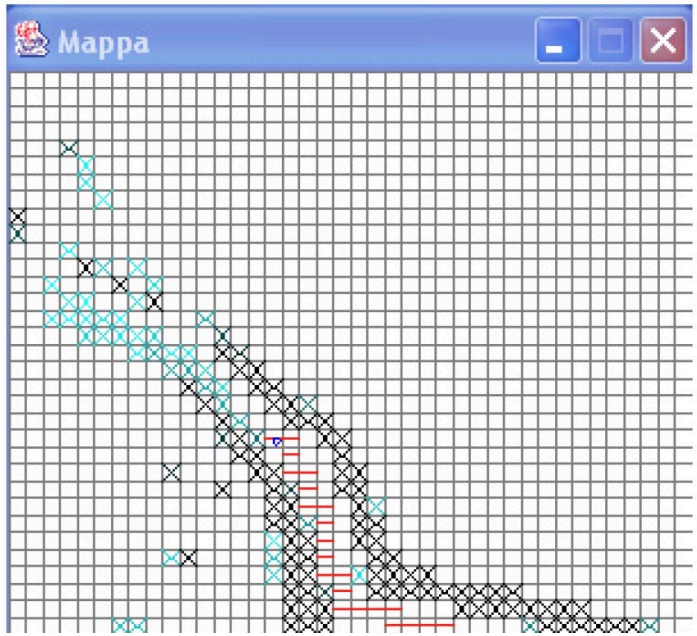
A questo punto possiamo ad esempio supporre di moltiplicare i risultati relativi ai due insiemi *fuzzy*, per ottenere il peso associato ad ogni singola rilevazione.





MAPPA

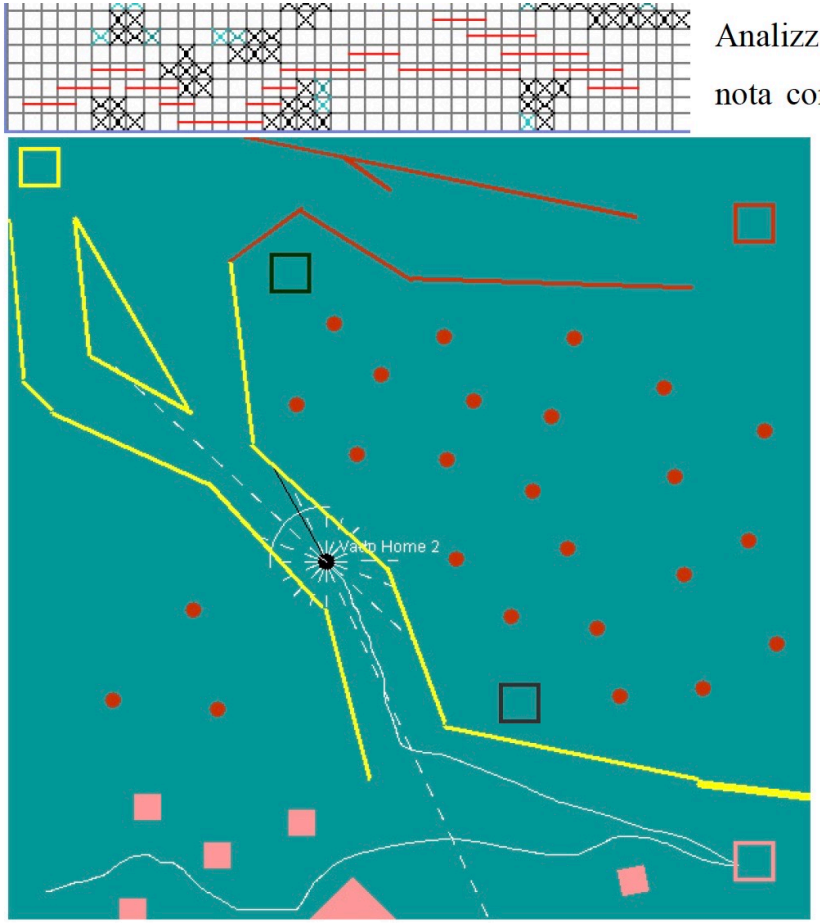
All'interno del package **CSAI.unipa.knowledgment** sono fornite le classi per implementare e gestire una mappa. Il modo in cui usare tale mappa può essere visto dall'esempio fornito all'interno della cartella \\Esempi\gara in cui un robot **MultiForageN150Explore** viene immesso in un campo di gara di cui conosce soltanto la posizione di alcune basi che deve conquistare una dietro l'altra, ed aumenta la sua conoscenza sul mondo man mano che procede nella sua esplorazione, generando una mappa che utilizza la logica *fuzzy* descritta in precedenza.



Un ostacolo "certo" viene disegnato nella mappa come una **x** di colore nero che occupa la cella in cui l'ostacolo si trova, mentre ostacoli non ancora del tutto "certi" vengono disegnati come una x di colore blu con intensità che aumenta man mano che aumenta la certezza che quell'ostacolo effettivamente sia presente (e cioè man mano che viene segnalato dai sensori).

ROBOTICSLAB

MAPPA



Analizzando la mappa ricostruita in figura si nota come la certezza di un ostacolo dipenda dalla sua vicinanza al robot (il percorso del robot è individuato da dei trattini orizzontali rossi, mentre la sua posizione da un cerchietto blu); gli ostacoli più lontani sono quelli meno certi, ancora da verificare.

Una volta effettuata la costruzione della mappa si potrebbe pensare di addestrare una rete neurale per effettuare il riconoscimento di stanze, di porte, di ostacoli pericolosi, vicoli ciechi e quant'altro possa servire ai fini della simulazione.

MAPPA

L'uso dei sonar può portare a notevoli vantaggi anche quando i rilevamenti vengono utilizzati per il calcolo di traiettorie efficienti nel raggiungimento di un prefissato punto, evitando tutti gli ostacoli che si trovano davanti. È possibile vedere come nell'esempio precedentemente citato \\Esempi\Gara il robot si muova con una certa facilità all'interno di ambienti molto complessi, semplicemente utilizzando tutti i suoi sensori per rilevare il maggior numero di ostacoli e quindi pianificare traiettorie che tengano conto di una situazione più generale possibile.

Si rende molto efficiente l'uso dei sonar in particolare quando il robot si trova a passare lungo stretti corridoi, in quanto il numero elevato di ostacoli individuati lo obbliga a muoversi mantenendosi sempre a equa distanza dalle pareti, anche se il punto di attrazione si trova al di la di una delle due pareti.

5.1. Importare le classi

Innanzi tutto deve essere inserita la cartella CSAI all'interno della directory \\tb\src\.

Si importano i seguenti package:

```
import CSAI.unipa.clay.*
import CSAI.unipa.knowledgment.NodeMap
import CSAI.unipa.abstractrobot.*
```

e deve essere dichiarata la classe del robot come estensione della classe:

```
ControlSystemMFN150Explore
```

5.2. Dichiarare le variabili

Tra le variabili globali si dichiarano le seguenti:

```
...
      private NodeBoolean         sonar_configuration
      private NodeBoolean         laser_configuration
      private NodeVec2Array       PS_SONAR
      private NodeVec2Array       PS_LASER
      private NodeMap             MAP
...
```

Le variabili `sonar_configuration` e `laser_configuration` vengono utilizzate rispettivamente per definire il comportamento dei sonar e del laser nei vari stati dell'FSA.

Le variabili `PS_SONAR` e `PS_LASER` contengono il risultato delle rilevazioni del sonar e del laser.

La variabile `MAP` è la classe che gestisce la mappa.

5.3. Metodo configure()

Il **MultiForageN150Explore** di *default* ha tutti i sonar accesi, utilizza il livello di precisione *MEDIUM* per la rilevazione dei dati, ed ha il laser ottico attivato.

È possibile modificare questi valori iniziali utilizzando nel *configure()* i metodi:

```
...
      public void turnOffSonar(int item)
      public void turnOnSonar(int item)
      public void turnOffAllSonar()
      public void turnOnLaser()
      public void turnOffLaser()
      public void setSonarPrecision(double prec)
...
```

I valori di precisione del sonar sono contenuti all'interno della classe *SonarObjectSensor*:

```
...
      SonarObjectSensor.LOW
      SonarObjectSensor.MEDIUM
      SonarObjectSensor.HIGH
...
```

Sempre all'interno del metodo *configure()* va instanziata la mappa invocandone il costruttore a cui vanno passati come parametro: i limiti (in ordine left, right, top, bottom) che definiscono la mappa, la dimensione delle celle (che sono quadrate), il raggio di appartenenza ad una cella (indica quale è la distanza massima a partire dal centro di una cella, per cui un punto debba essere considerato appartenente a questa), il nome della mappa, e il robot che la sta creando, ad esempio:

```
...
      MAP = new NodeMap(-10, 10, 10, -10, 0.5, 0.4, "Mappa ",
                        true, abstract_robot);
...
```

A questo punto si associa il rilevamento di sonar e laser alle variabili *NodeVec2Array* appropriate, e si fondono insieme in un'unica variabile `PS_ALL_OBS` anch'essa di tipo *NodeVec2Array*:

```
...
      PS_OBS = new va_Obstacles_r(abstract_robot);

      PS_SONAR = new va_SonarSensed_r(abstract_robot);

      PS_LASER = new va_LaserSensed_r(abstract_robot);

      NodeVec2Array
      PS_TEMP = new va_Merge_vava(PS_LASER, PS_SONAR);

      NodeVec2Array
      PS_ALL_OBS = new va_Merge_vava(PS_TEMP, PS_OBS);
...
```

La variabile `PS_ALL_OBS` contiene al suo interno tutti gli ostacoli che sono stati rilevati per mezzo del laser, del sonar, e del vecchio sistema di rilevamento, e va usata per calcolare le traiettorie che evitano gli ostacoli per mezzo della classe *v_Avoid_Sonar* contenuta nel package **CSAI.unipa.clay** invocandone il costruttore in questo modo:

```
...
      NodeVec2
      MS_AVOID_OBSTACLES = new v_AvoidSonar_va(3.0,abstract_robot.RADIUS+0.1,
                                               PS_ALL_OBS, PS_GLOBAL_POS);
...
```

A questo punto vanno definiti i comportamenti utilizzando la `MS_AVOID_OBSTACLES` sopra definita. L'ultimo passo da fare è quello di definire il comportamento del sonar e del laser relativo ad ogni stato dell'FSA. Si supponga per esempio di avere un automa formato da 3 stati.

Si associa il comportamento del sonar all'FSA in questo modo:

```
...
      d_SonarControl_ir
      SONAR_CONFIGURATION=new d_SonarControl_ir(STATE_MACHINE,abstract_robot);
      SONAR_CONFIGURATION.sonarActived[0] = new
                                    int[]{0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15};
      SONAR_CONFIGURATION.sonarPrecision[0] = SonarObjectSensor.MEDIUM;
      SONAR_CONFIGURATION.sonarActived[1] = new
                                    int[]{0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15};
```

```
      SONAR_CONFIGURATION.sonarPrecision[1] = SonarObjectSensor.MEDIUM;
      SONAR_CONFIGURATION.sonarActived[2] = new int[]{4,12};
      SONAR_CONFIGURATION.sonarPrecision[2] = SonarObjectSensor.HIGH;
      SONAR_CONFIGURATION.sonarActived[3] = new int[]{0};
      SONAR_CONFIGURATION.sonarPrecision[3] = SonarObjectSensor.HIGH;
...
```

Per quanto riguarda il laser, se ne associa il comportamento all'FSA in questo modo:

```
...
      b_Select_ir
      LASER_CONFIGURATION = new b_Select_ir(STATE_MACHINE, abstract_robot);

      LASER_CONFIGURATION.embedded[0] = true;
      LASER_CONFIGURATION.embedded[1] = false;
      LASER_CONFIGURATION.embedded[2] = true;
      LASER_CONFIGURATION.embedded[3] = false;
...
```

Infine si associano questi comportamenti alle variabili globali che selezioneranno il comportamento adeguato dipendentemente dallo stato in cui il robot si trova:

```
...
      laser_configuration = LASER_CONFIGURATION
      sonar_configuration = SONAR_CONFIGURATION
...
```

All'interno del *takeStep()* bisogna innanzi tutto gestire la mappa, invocando i metodi che effettuano i controlli sulla posizione del robot, e sui rilevamenti degli ostacoli.
I metodi da invocare sono i seguenti:

```
...
      MAP.setCellVisited(curr_time);
      MAP.setObstacle(PS_SONAR.Value(curr_time),
                              abstract robot.getPrecisionRivelation(), curr time);
```

```
      MAP.setObstacle(PS_LASER.Value(curr_time),
                                        LaserFinderObjectSensor.FUZZY, curr_time);
...
```

Il metodo *setCellVisited()* serve per settare come visitata la cella in cui il robot si trova in quel momento (escludendo quindi la possibilità che un ostacolo vi si possa trovare sopra).
Il metodo *setObstacle()* va invocato passandogli come parametri:

- il risultato delle rilevazioni del sonar (o laser),
- *l'array* che contiene i pesi delle singole rilevazioni. Nel caso del sonar viene restituito invocando il metodo *abstract_robot.getPrecisionRivelation()* mentre per il laser (l'indice di errore non dipende dalla distanza ma soltanto dalle caratteristiche del dispositivo stesso) è contenuto all'interno della variabile statica `LaserFinderObjectSensor.FUZZY`,
- l'istante di tempo in cui si sono effettuate le rilevazioni.

Infine deve essere aggiornato il comportamento del sonar e del laser a seconda dello stato in cui ci si trova scrivendo le seguenti righe di codice:

```
...
      //SONAR
      bresult = sonar_configuration.Value(curr_time);

      //LASER
      bresult = laser_configuration.Value(curr_time);
...
```

Essendo stata `bresult` definita precedentemente come una variabile *booleana*.

I metodi *Value()* relativi a `sonar_configuration` e `laser_configuration` ritornano valore *true* ogni volta che rispettivamente le impostazioni del sonar o del laser vengono modificate (cioè ad ogni passaggio di stato, se sono stati definiti comportamenti per ogni stato).