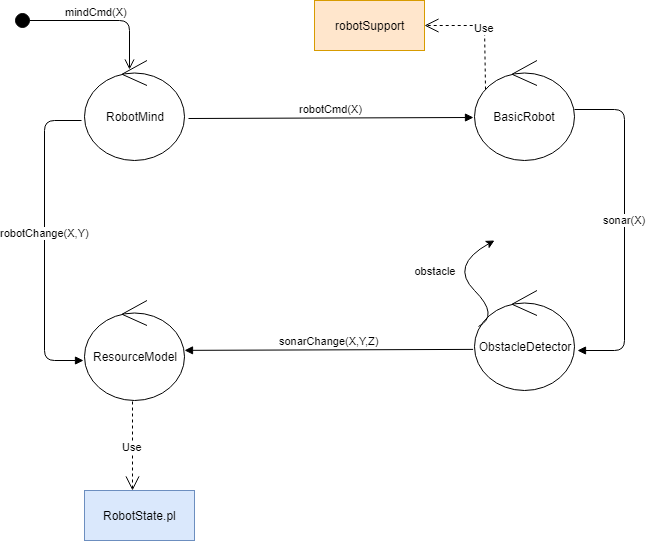
SPRINT1

In questo sprint ci si concentra sulla parte di movimento e detecting di ostacoli del robot

* Realizzazione supporto Robot
* Ricezione comandi Robot
* Riconoscimento ed evasione di ostacoli

Architettura Finale

Spiegazione

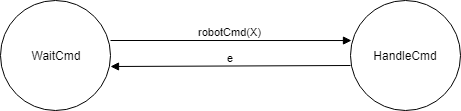
Per la parte di supporto al movimento è possibile riutilizzare un componente già presente chiamato **robotSupport**  che ci offre il funzionamento base di un robot DDR con gestione di un sonar HC-SR04.

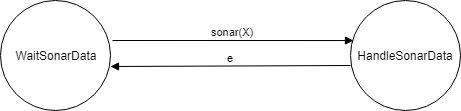
L’attore *basicRobot* si occupa quindi di ricevere dei comandi “semplici” e di eseguirli tramite il supporto sottostante, inoltre genererà anche eventi relativi ai dati che vengono rilevati dal sonar nel formato *sonarRobot:sonar(X)*

Un altro attore *obstacleDetector* che si comporta come un observer nei confronti dei dati provenienti dal sonar allo scopo di rilevare eventuali ostacoli e notificarli al sistema tramite un evento *obstacle*

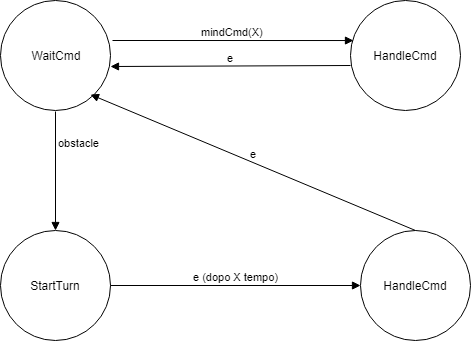
Per quanto riguarda l’evasione degli ostacoli è bene che non sia basicrobot ad occuparsene ma un altro componente che lavori ad un livello di astrazione più alto. *RobotMind*  si occupa sia di inoltrare i movimenti al robot che decidere il modo in cui evitare gli ostacoli.

Comportamento

**BasicRobot**

**ObstacleDetector**

**RobotMind**



NOTA:  
Per migliorare il setUp dei parametri viene aggiunto un file Prolog *sonarConfig.pl*  contenente la distanza dopo la quale un oggetto viene considerato ostacolo. Il formato del file avrà una riga del tipo *detection(MISURA).*

Testing…

Al fine di testare quanto sviluppato ci servirebbe sapere due cose:

* Se la mind ha analizzato ed inoltrato bene il comando inviatogli
* Se la mind reagisce bene agli ostacoli
* Se l’obstacleDetector rileva correttamente gli ostacoli

…verso l’MVC

Per quanto detto in fase di analisi iniziamo ad introdurre il concetto di modello e di resourceModelcome “centro di raccolta” delle informazioni del robot. Introduciamo quindi un nuovo attore *resourceModel* che si occupa di aggiornare in maniera consistente le informazioni all’interno di un file *RobotState.pl* in maniera tale che interrogandolo potremo sempre avere uno stato aggiornato del robot.

Due linee:

* state (robot, X) X:= movingForward,movingBackward,stop,turnLeft,turnRight
* state (sonar, X, Y) X:= misura rilevata Y:=ostacle/clear

Sia robotmind che obstacle detector hanno il compito di andare a modificare le relative righe per mantenere uno stato consistente