Idea progetto - appendice

**Asincronismo**

per garantire safty sarà necessario dimensionare il raggio di comunicazione in maniera opportuna.

La dimensione del raggio dipenderà da:

* velocità massima veicolo
* tempi di aggiornamento Ti
* spazio di frenata.

Volendo avere un raggio costante nel tempo e uguale per tutti i veicolo l’idea sarebbe quella di valutare il caso peggiore possibile e dimensionare di conseguenza il raggio.

(raggio sarà una cosa tipo: r = 2\*Ti + Tfrenata )

**Sezioni critiche troncate**

Nell’approccio distribuito non viene trasmessa tutta la traiettoria ma solo una porzione. Può verificarsi il caso in cui l’ultimo pezzo trasmesso appartenga ad una sezione critica,

in questo caso il secondo veicolo non conoscerebbe il tempo di uscita dalla sezione critica e vedrebbe questa sezione occupata per un tempo infinito e quindi nel caso non avesse precedenza dovrebbe aspettare. Tale situazione si risolverà in automatico nelle iterazioni successive in quanto per ipotesi nessun robot può finire il suo percorso all’interno di un intersezione.

**Traiettoria trasmessa**

Dobbiamo ancora definire quanta traiettoria trasmettere, per il momento pensiamo di trasmettere il pezzo di traiettoria contenuto nel raggio di comunicazione, tuttavia una volta passati alla fase simulativa potremo vedere se tale soluzione si rivelerà funzionale.

**Corridoi**

se è presente un corridoio di lunghezza tale da non permettere a due robot che si trovano ai due capi di comunicare fra loro allora sarà necessario introdurre dei ripetitori/semafori, in maniera tale da far aspettare uno dei due nel caso in cui la sezione critica sia già occupata.

**Memoria**

AL momento l’algoritmo pensato prevede di ricalcolare sezioni critiche e precedenze ad ogni iterazione.

Stiamo pensando anche a possibili soluzioni in cui le informazioni ( o solo alcune) di precedenze e sc vengano sfruttate anche al passo successivo.

Suddivisione FIle

classe Vehicle : definiamo variabili di stato del robot e le funzioni necessarie per accedervi

private int robotID = -1;

private int priority = -1;

private Pose pose = null;

private int pathIndex = -1;

private double velocity = 0.0;

private double velMax = 0.0;

private double accMax = 0.0;

private int criticalPoint = -1;

private boolean csTooClose = 0;

private TrajectoryEnvelope trjEnv = null;

private String category = "";