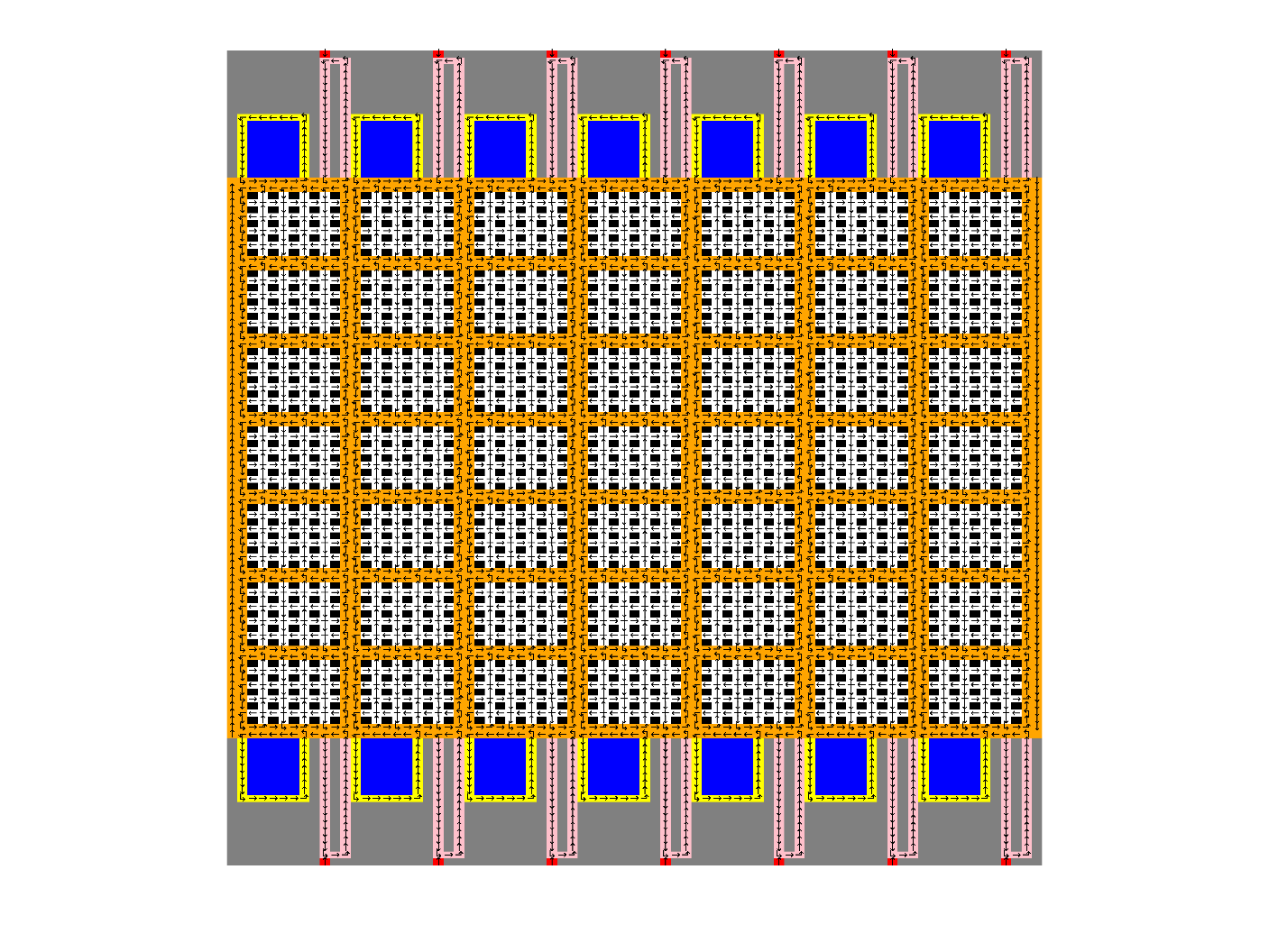
**Sorting Robot**

Sorting Robot sorts the parcel into their respective bins depending on the locations using barcodes. Map for sorting robots includes parcel bin, streets and highways for robot movements, robot charging points, parcel pickup points where human will place parcel conveyor belt to robot top. Once parcel is placed on robot top, robot should place the parcel to its respective bin by following best possible path and avoiding the congestion as much as possible.

**Navigation Graph**

Charging

****

Pickup Queue

Charging Queue

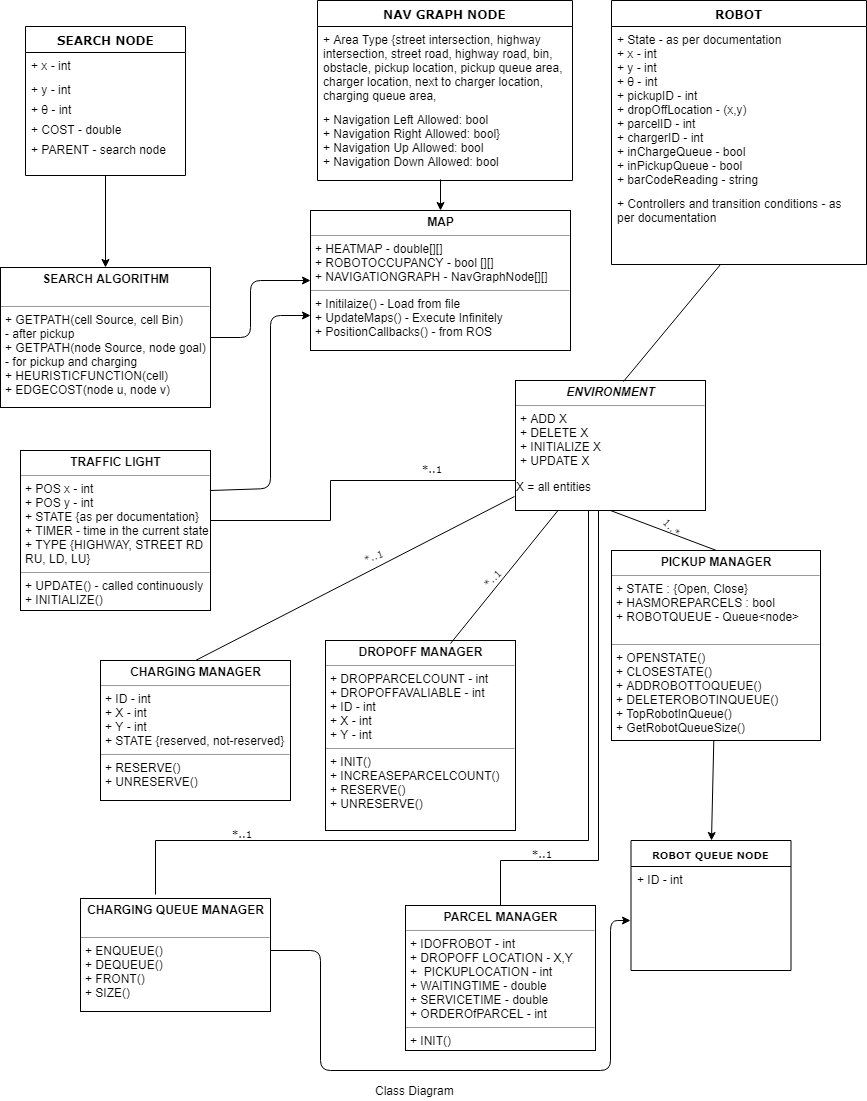
Pickup

Bin

Streets

Highways

CLASS DIAGRAM:



**Heat Map:** Heatmap is used for conges tion avoidance. Search Algorithm avoids area of high heat value. A star algorithm tries to schedule robot to go through area of lower cost thus distributing map to avoid congestion at particular place.

HeatMap(t)= ηΔt HeatMap(t-Δt)+ OccupancyMap(t)

η = evaporation rate (typically 0.75 per millisecond)

**Search:**

A\* Algorithm. It operates on (x,y,θ) as state vector.

There are two variants of the algorithm:

1. Shortest cost path to a bin in which case destination can be anyone of four adjoining edges.



1. It is the case of charging /pickup queue, where goal is available (xG, yG, θG) representing location of starting of the queue of chosen charging station / pickup point.
2. Edge Cost:

v=maximum linear speed, ω=maximum angular speed, penalty is applied to penalize stopping and turning (≈1.2), α penalizes heat map (to experiment)

1. Heuristic Cost for bin search:

Here (*G*x,*G*y) stands for the bin location

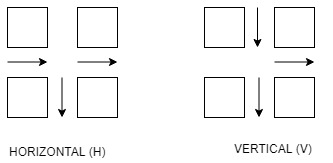
1. Heuristic Cost for goal search:

**Traffic Light**

Purpose of Traffic Light Management System is to ensure the robots don’t collide **sideways** which is important given that there is no proximity sensor that tells robots on the sides.

The intersection management policy is that at any point of time only one robot will be allowed to center the intersection thus avoiding collision. The traffic light ensures adherence of the policy.

1. Traffic light related to the street intersection has 2 states:
   1. Traffic originating from Horizontal Lane can flow
   2. Traffic originating from Vertical Lane can flow



Traffic light change is modeled as FSM where vertices represent state of traffic light and the edges represent condition meeting which the state will change.

The FSM representing Traffic Light change is given below:

Case RD (right down):

(no robot at horizontal k blocks and robot at vertical k blocks) or (after T time and robot at vertical k blocks)

(O(x,y-i)=0 ∀ 0≤i≤k ∧ ∃ i: O(x-i,y)=0 0≤i≤k) ∨ (Timer>T ∧ ∃ i: O(x-i,y)=0 0≤i≤k)

Initial

(after T2 time or no robot at horizontal k blocks)

(Timer>T2 ∨ O(x,y-i)=0 ∀ 0≤i≤k)

HG, VR: Horizontal Green/Vertical Red

HY, VR: Horizontal Yellow, Vertical Red

HR, VG: Horizontal Red, Vertical Green

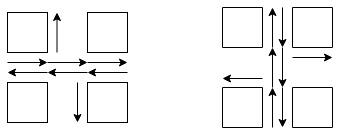
HR, VY: Horizontal Red, Vertical Yellow

RD : Right Down

O : Occupancy Map

Cases Right Up (RU), Left Down (LD), Left Up (LU) can be similarly enumerated.

1. Traffic Light for Highways: It operates in two states:



Here also Traffic Light condition is exactly same as for Street Traffic Lights.

**Coordinate Axis Systems**

The navigation map, heat map, search and traffic light use the cell coordinate axis system where origin is top-left, X axis is the vertical axis, Y axis is the horizontal axis and the coordinate system is discretized into cells.

The robot positions and simulator use the real coordinate axis system.

A central utility exists to convert the world coordinate axis system into cell coordinate axis system.

**Robot BFSM:**

Overall Architecture to control Robot a behavioral FSM. There vertices represent different types of controllers. Robot at any state means the corresponding controller is loaded on the robot for motion. The edges represent the transition condition which are continuously monitored. Upon reaching the transition condition the controller is changed.

**Behaviors:**

**Select Pickup Behaviour**

pickup = arg min w1 CostToReachPickup(i) + w2 PickupQueueSize(i) : pickup(i) has more parcels

**Select Charger**

pickup = arg min w1 CostToReachCgarger(i) + w2 ChargerQueueSize(i)

**Robot Controller Callbacks to Ensure:**

i) Callback when robot enters the pickup area, if operational (robotInPickupQueue = False and RobotInPickupArea=true and Pickup(i) has more parcels). If the condition is true, call “enqueue pickup queue” and set robotInPickupQueue = True

ii) Callback when robot enters the charging area (robotInChargeQueue = False and RobotInChargeArea=true). If the condition is true, call “enqueue charge queue” and set robotInChargeQueue = True

iii) Callback when robot leaves the pickup area (pickup cell only) (robotInPickupQueue = True and RobotInPickupArea=false and Pickup(i) has more parcels). If the condition is true, call “dequeue pickup queue”, set robotInPickupQueue = False, set pickup state = open

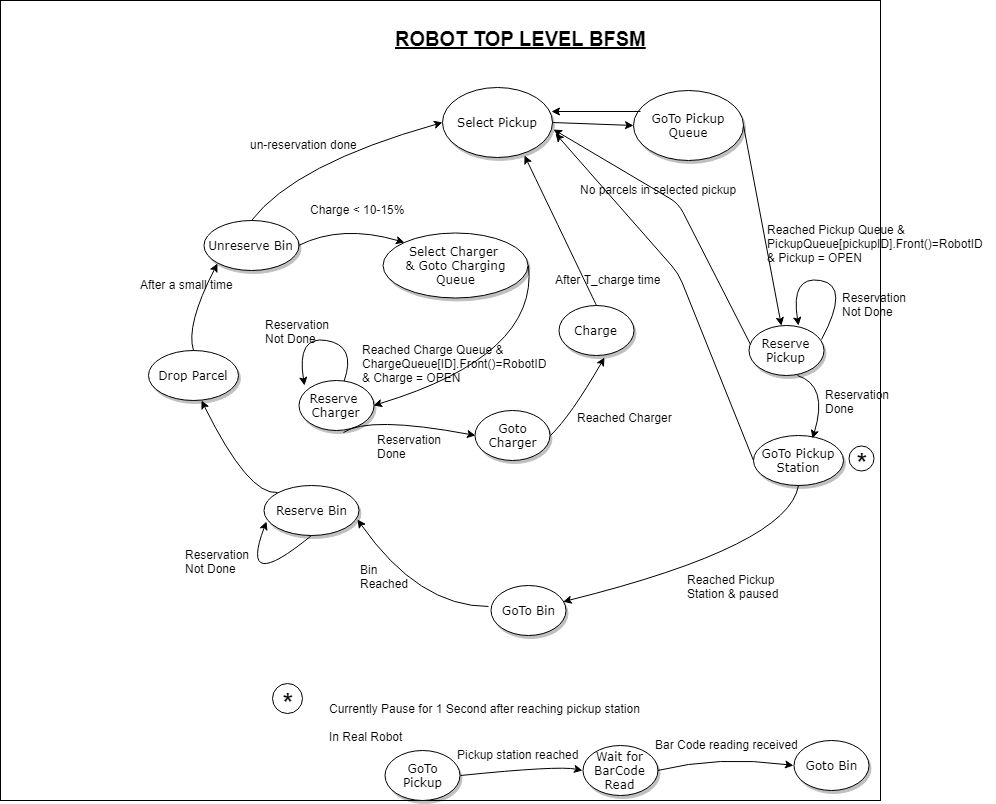
iv) Callback when robot leaves the charge area (charge cell and the next cell) (robotInChargeQueue = True and RobotInChargeArea=false). If the condition is true, call “dequeue charge queue”, set robotInChargeQueue = False, set charger state = open

|  |  |
| --- | --- |
| ↓ |  |
| ↓ | → |
| ↓↑ |  |

Charging Point

v) Callback when the robot is in a pickup queue, but the pickup queue has no parcels. Change state as per BFSM.

In all cases, the information from area is available from the annotated navigation map and the charger/pickup ID selected by the robot



Charge>95% (from plugin)

**Robot “Goto Goal” Behavior**

**Level 1a: Goal is forward ahead a certain distance**

Assume that the robot senses dF from the front proximity sensor. The cell corresponding to the physical location of the robot at front can be calculated. The robot must never stop/aim to stop at an intersection. Therefore, before dF, the first non-intersection road is calculated. Let it be at a distance of dI. This is only when the behavior is non-turn. Note that, unlike previously, the world coordinate axis system is now in place.

Bins

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  | dF |  |
|  | dI |  |  |  |
|  |  |  |  |  |

Last Non-Intersection

v=KP min(d(q,goal), dI)

θG=atan2(Gy-y, Gx-x)

ω= KP angleDifference(θ,θG)

d(q,goal) = Eucledian distance between current position and sub-sub-goal

dF= distance from front proximity sensor

dI = distance to the previous intersection with no robot

**Level 1b: Goal is at the same place with a π/2 orientation**

v=KP (d(q,goal)) : iff goal ahead of robot

θG=desired goal angle

ω= KP angleDifference(θ,θG)

**Level 2a: Traffic Light Aware Motion Planner**

Read the current robot position and current a sub-goal directly ahead

Set sub-sub-goal in the path as the last point where traffic light is yellow/red or last point where a turn needs to be performed.

Publish sub-sub-goal

**Level 3: Sequencer**

Get path from search algorithm

Simply the path as sequences of straight motion and turn only

*(e.g. (1,1), (2,1), (3,1), (4,1), (4,2), (4,3), (4,4) becomes (1,1) (4,1) turn (4,4))*

while path is not null

select sub-goal as next turn/straight path at a maximum distance k

publish sub-goal and do not wait for action server

plan a new path from sub-goal

simply the path as sequences of straight motion and turn only

wait for action server

/robot{i}/sub-sub-goal

/robot{i}/goal

/robot{i}/sub-goal

/robot{i}/sub-sub-goal

RQT-GRAPH: Rqt Graph showing nodes, topics and date to be passed between nodes is shown below:

