# DEBI TEAM 92


## ABSTRACT

**This is the system specified for DEBI competition 202 which Autonomous Competition with two teams opponent to each other, by shooting different color of balls in opponent field. The team with minimum balls at the end of the game (which is 10 min) will when the match.**

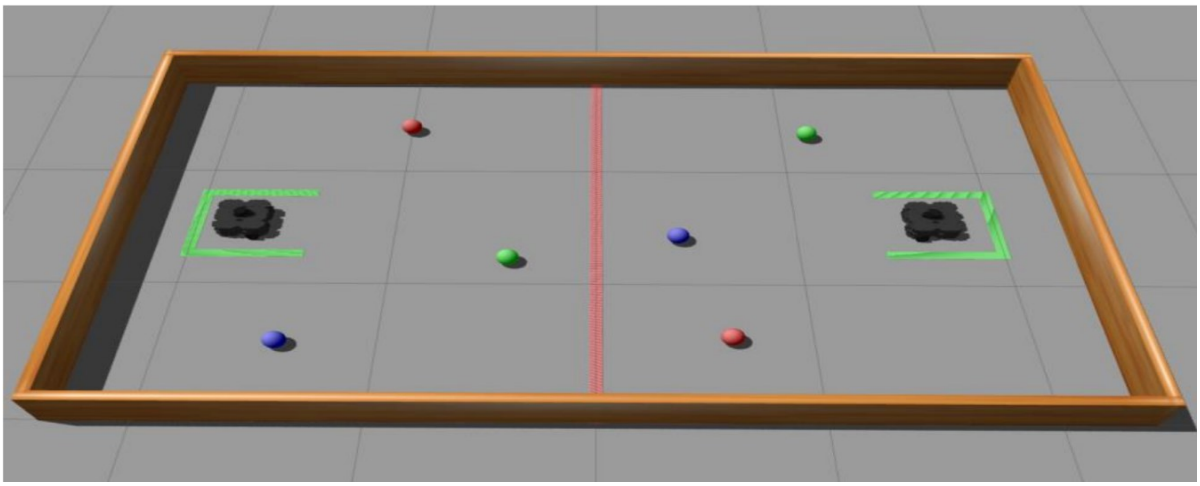
**We exploit LIDAR, mono camera, and wheel encoders sensors to achieve autonomy in different sections such as Perception we use camera model to perform object detection with state-of-the-art model YOLOv8. We train model on large dataset to obtain robust detection.**

**Robot also localize it self by Particle filter which use LIDAR scans to match with the predefined map, and robot navigate through this map with DWA (dynamic widow approach) Local planner.**

**We also Use PID to perform position control in different section in the map.**


## CONTENT

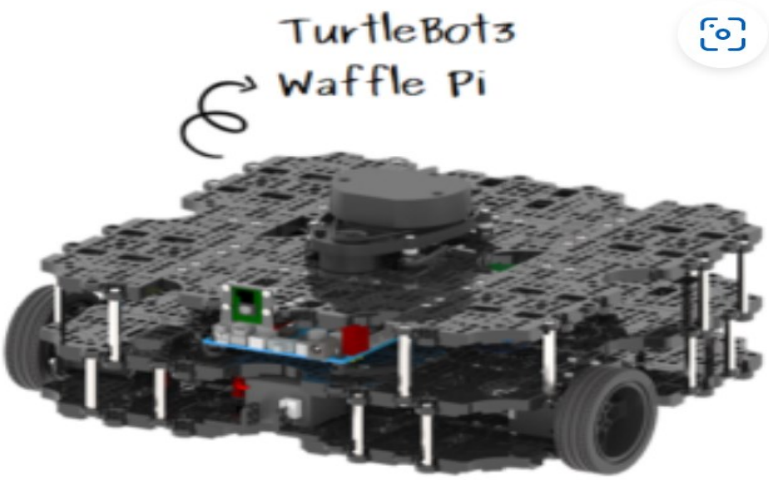




# 1-INTRODUCTION TO SYSTEM

The system is totally deployed on ROS and most component such as localization and path planning is a open-source implementation by ROS community.

The theme of the competition use Turtlebot3 Waffle pi which is open source robot supported by ROS. It contain great sensors to achieve autonomous system such us, LIDAR, mono-camera, IMU, and Encoder. All this powered by Raspberry-pi 3 and MCU called OpenCR . We could connect with Raspberry by SSH connection and stream ROS topics on our PC to achieve intensive computation such as object detection model YOLOv8.




One of the treat of using ROS system is it could use a lot of ROS tools such as data visualizer RViz and 3D simulator Gazebo which the main tool for testing the competition theme because the turtlebot3 is not present with us so we depend totally on simulator

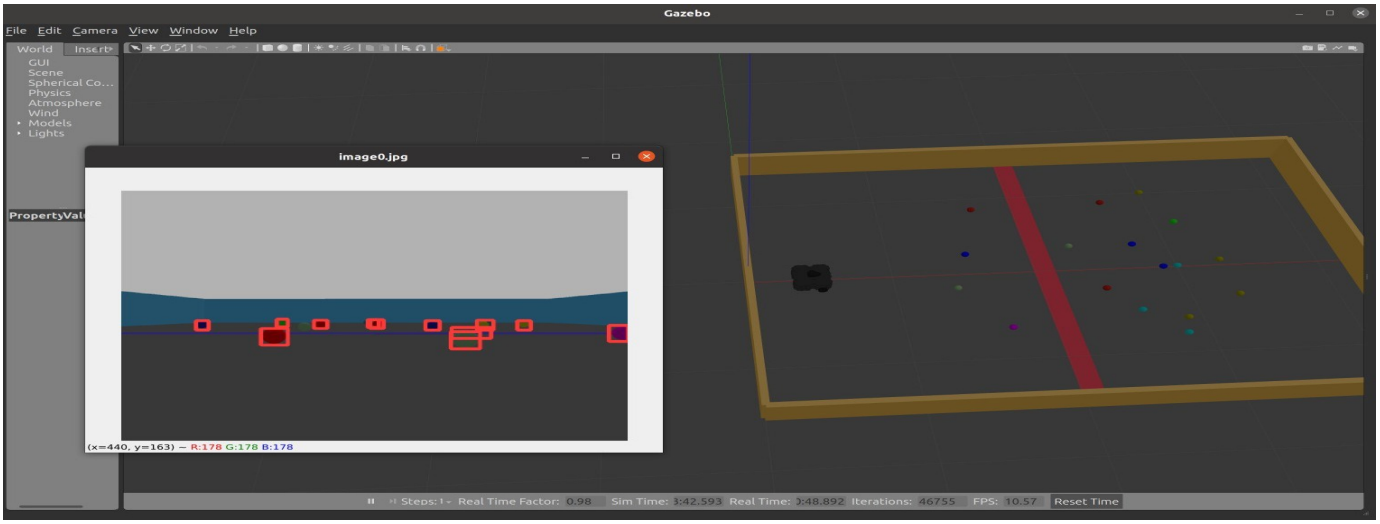
# 2-PERCEPTION

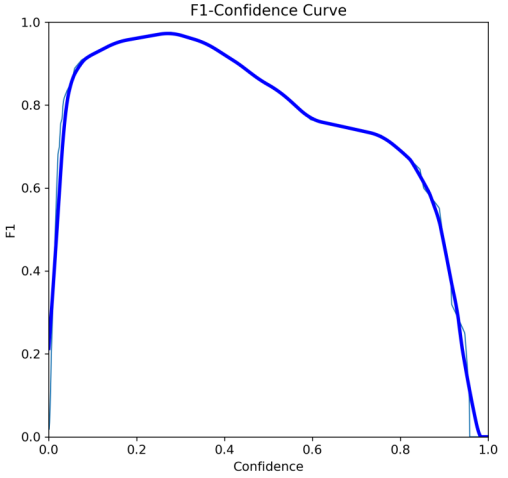
We need perception module for detect the balls that we need to shoot in another field we also need to localize it's position in the field to take action with it.

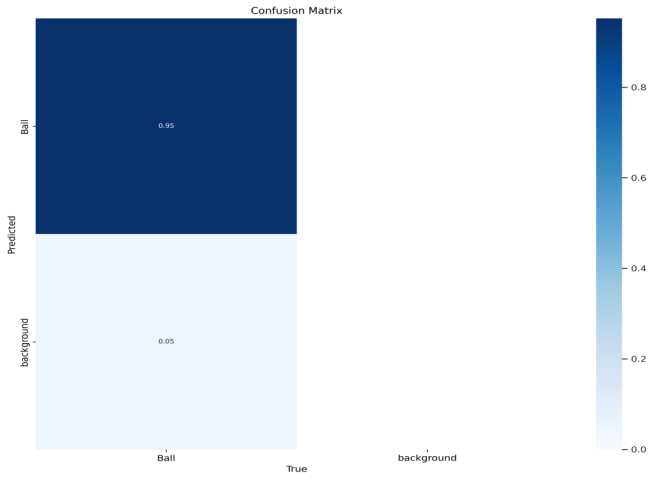
## 2.1– Object Detection

We use Yolov8 object detection to detect and localize the ball position in camera frame we Train a yolov8n model on a large dataset that comprise 2,000 training image of solid balls with different colors we also preprocess this data (Data augment) to add more variety on this dataset by augment Hue of images to make model less sensitive to the colors.

We validated the model with 200 images and give as great Precision and Recall which is suitable for us (we actually care about recall more than precision for DEBI theme).



## 2.2– Compute position

After we detect the ball and get it's position with respect to camera frame we need to know it's position with respect to the map, we know latter that we know from localization module that we know the robot position in the map.

We could compute Xr, Yr which is the xy position with respect to base frame of the robot.

$$VFOV = 48.8$$

$$HFOV = 62.0$$

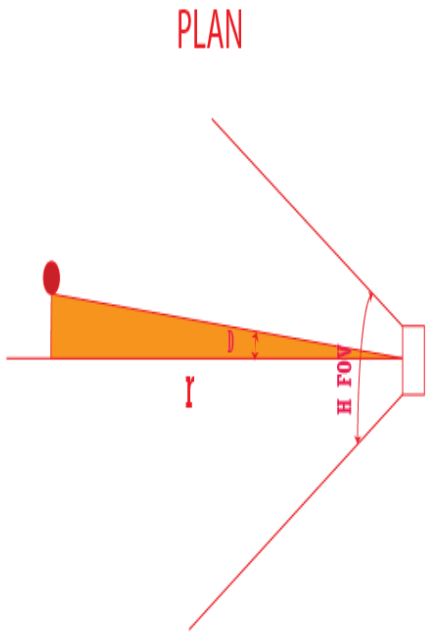
$$H = 12cm$$

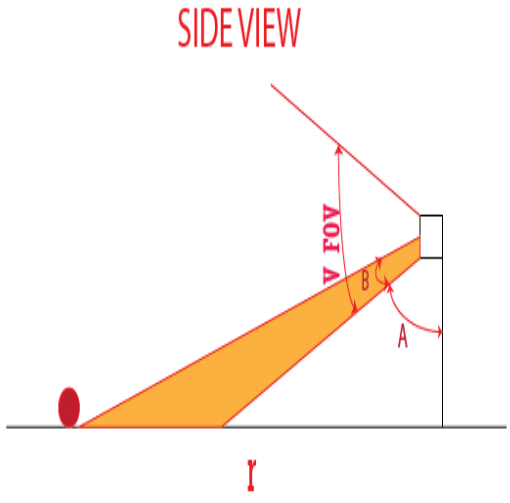
$$A = \frac{(180 - 48.8)}{2} = 65.5$$

$$B = \frac{(pixels - total)}{total} * VFOV$$

$$r = \frac{H}{tan(A + B)}$$

$$D = \frac{pixels - \frac{total}{2}}{\frac{total}{2}} * HFOV$$



# 3-LOCALIZATION

In Localization module we use AMCL package which is developed by ROS community it's just implementation of Particle filter which define multiple particle as a belief distribution when robot not sure about It's position the distribution is wide, and if the robot confident about it's position the distributions narrowed.

AMCL is just localizer not a SLAM algorithm so we need to give AMCL a map of the game field and the initial position of this map. We draw the map with gmapping SLAM and save this map with map_saver package.

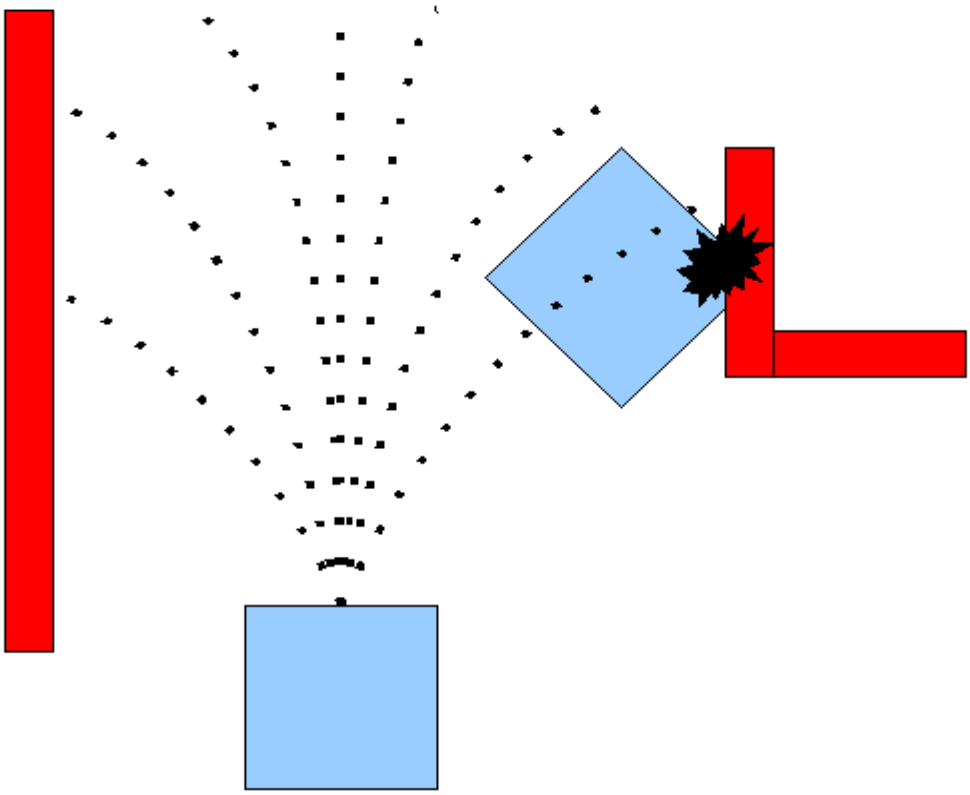
AMCL return to us a (x, y, theta) of the robot on the map.


One of the problem of the AMCL is retrieval frequency of the pose though we need odometry to get high frequency estimation of the position of robot, this is important for the control system and it's responsivity of the system at all.

We could correct this high frequency odometry by the low frequency update step with high accuracy to achieve robust localization system.

# 4-NAVIGATION

We use DWA local planner which stand for Dynamic Window Approach local planner. The dwa_local_planner package provides a controller that drives a mobile base in the plane. This controller serves to connect the path planner to the robot. Using a map, the planner creates a kinematic trajectory for the robot to get from a start to a goal location. Along the way, the planner creates, at least locally around the robot, a value function, represented as a grid map. This value function encodes the costs of traversing through the grid cells. The controller's job is to use this value function to determine dx,dy,dtheta velocities to send to the robot.



The basic idea of the Dynamic Window Approach (DWA) algorithm is as follows:

1. Discretely sample in the robot's control space (dx,dy,dtheta)
2. For each sampled velocity, perform forward simulation from the robot's current state to predict what would happen if the sampled velocity were applied for some (short) period of time.
3. Evaluate (score) each trajectory resulting from the forward simulation, using a metric that incorporates characteristics such as: proximity to obstacles, proximity to the goal, proximity to the global path, and speed. Discard illegal trajectories (those that collide with obstacles).
4. Pick the highest-scoring trajectory and send the associated velocity to the mobile base.
5. Rinse and repeat.

# 5-CONTROL

We Use PID controller to make position control on x, y, or  yaw we prefer to use PID on path planning in the very short plans because of complexity of path planning.

# 6-STRATIGY

- We observe the balls by object detector and  estimate it's position in the map, then after that we categorize the balls in the map in two categories Home and Away, home is the balls in our field and away is the balls in the opponent field.
- If the seen balls contain Home balls we sort them by distance and aim to the nearest ball.
- After Shooting the ball we return back to better position to preserve the map in better way, we do that by plan a path.
- If the seen balls don't contain Home balls and the number of seen balls is equal to total number of balls in the game-field, we deduce that all balls in the opponent field and we still steady for any action from the opponent.
- If the seen balls don't contain Home balls and the number of seen balls is not equal to total number of balls in the game-field, the we will start searching for balls in our field by rotating to ourself with small speeds.
- If we don't found any balls after a one cycle of searching we start translate our position to another place to find the missing balls.

# 6-FUTURE PLANS

## i) Path Planner

We need to try different path planning algorithm because DWA is not stable and sometimes stucks. We maybe try TEP, and MPC.

## ii) Object Detector

We need to increase the dataset to work very will in the competition, and we will try different architectures and deploy the best of them, we may implement one on Tensorflow or Pytorch.

## iii) Strategies

We will try different strategies with different cases of balls and different organization of them in the field.

## iv) Safety

We will try to make our software to be safe as possible and this may done in different levels, such as in planner we could increase inflations of the obstacles to ensure that the trajectory never be so close from the obstacles. We could also add constraints on the speed of the robot and acceleration to be more safe and never happens any accident to the turtlebots or even the map. We also will check that robot in the specified field hardcoded and continuously check for this and if something wrong happened we shut down the speeds.

## iv) Clean Code

We will enhance our software by increase it’s modularity and add comments to be more friendly to track and we will make it public for anyone need to use after that.