Team 8

Methodology:

Robot Control	**Node**: ControlSubscribed topics: None **Published topics:** /robot/robotnik_base_contol/cmd_vel **Topic stucture :** **Message Type:** Twist() **queue size :** 1 **Parameters:** MAX_LIN_VEL MAX_LIN_VEL LIN_VEL_STEP_SIZE ANG_VEL_STEP_SIZE **Methodology:** • function getKey() → key using tty stdin to read the input as a list of all keys have been pressed process the fist key. • Main() → publish call the getkey at beginning. based on the key returned, do a check to increase linear or angular velocities but within the range. If no key pressed for short time which we simulate with status if status reached 10 then both linear and angular velocities decayes to 0.
Sensor Incorporating and Alignment	**External Package (ira_laser_tools)** **This package used for combine the bose laser front and rear data in one topic only.** **Node:** ira_laser_tools **Subscribed topics:** /robot/front_laser/scan /robot/rear_laser/scan **Published topics:** /scan_multi , /merged_cloud **Topic stucture : LaserScan() , PointCloud()** **Parameters:** destination_frame angle_min angle_min angle_max angle_increment scan_time range_min range_max

	Custom Message: → **Name :** custom_msg **structure:** Header, LaserScan , Odometry **Combiner class:** use message_filters library in python to combine the odom and laser scan data and then publish to one topic called laser_odom. **Node :** Combine_lasers_and_odom **Subscribed topics:** /scan_multi /robot/robotnik_base_control/odom **Published topics:** /laser_odom **Topic stucture :** **Message Type:** custom_msg() **queue size :** 1 **Parameters:** None
Mapping with known poses	**Node**: mapping **Subscribed topics:** /scan_multi **Published topics:** /Map_known_Pose **Topic stucture :** **Message Type:** Ocupancy grid **queue size :** 1 **Parameters:** sensor_model_p_occ value="0.75 sensor_model_p_free" value="0.45" sensor_model_p_prior" value="0.5" robot_frame" value="robot_base_link" map_frame" value="robot_map" map_center_x" value="-50.0" map_center_y" value="-50.0" map_size_x" value="100.0" map_size_y" value="100.0" map_resolution" value="0.08" map_publish_freq" value="2" update_movement" value="0.03" **Methodology: Occupancy grid mapping with some modifications** The above are the parameters that algorithm needs to work. 1- we get the robot position from the ground truth we have in the simulation which is the location of the robot base frame "robot_base_link" and the fixed frame in the environment "robot_map" by publishing tf transfom between the two frames we get the robot true position and

	orientation. 2- every callback from the laser topic, get the true robot position and orientation then scan all the laser data which in the form of (angle,distance) - update the angle with the information you know about the robot orientation. - Get the x , y associated with each laser beam (object location in meters) - convert object location to indices, call bresenham to construct line from robot cell to object cell. - for each point in the returned line pixels : use the same algorithm taught in the lectures for update the cells values. cell += inverse(free_probability) – inverse(prior_probability) - except for the last cell (hit cell) cell += inverse(occ_probability) – inverse(prior_probability) 3- after processing all laser data: retrieve P for all cells values and multiply by 100 and store in list int8 which is the type of the occupancy grid 4- publich the map.
Mapping with Unknown Pose fastSlam	**Node**: fastSlam **Subscribed topics:** /laser_odom **Published topics:** /fastslam **Topic stucture :** **Message Type:** Ocupancy grid **queue size :** 1 **Parameters:** sensor_model_p_occ value="0.75 sensor_model_p_free" value="0.45" sensor_model_p_prior" value="0.5" robot_frame" value="robot_base_link" map_frame" value="robot_map" map_center_x" value="-50.0" map_center_y" value="-50.0" map_size_x" value="100.0" map_size_y" value="100.0" map_resolution" value="0.08" map_publish_freq" value="2" update_movement" value="0.03" particles" value="3" **Methodology:** **grid based fastslam**

	will mention the modifications to this part: 1- we get the odom data (x,y,qx,qy,qz,qw) from odometry sensor. 2- get the covariance matrix from the sensor. 3- sample error from gaussian distribution with the mean and variance we get from the sensor of each variable. np.random.normal(loc = x,scale=covariance_x,size=*self*.particles) sample n = *self*.particles samples. 4- for each particle scan its map with the current observations (laser data) , for each match increase the the probability, the particle with highest probability (hits) is the more likely to be the real robot pose. self.hit_miss = [e/count for e in self.hit_miss] idx = np.argmax(self.hit_miss) 5- publish the map of the particle with highest probability. 6- selective sampling : from a uniform distribution (min probability , max probability) sample n = num_particles samples, then get the indices and those are the new particles.
Mapping with Unknown Pose slam_kalman_filter	**Node**: slam_kf **Subscribed topics:** /laser_odom **Published topics:** /slam_kf **Topic structure :** **Message Type:** Ocupancy grid **queue size :** 1 **Parameters:** sensor_model_p_occ value="0.75 sensor_model_p_free" value="0.45" sensor_model_p_prior" value="0.5" robot_frame" value="robot_base_link" map_frame" value="robot_map" map_center_x" value="-50.0" map_center_y" value="-50.0" map_size_x" value="100.0" map_size_y" value="100.0" map_resolution" value="0.08" map_publish_freq" value="2" update_movement" value="0.03" #calculate kalman gain kX value=”0.8” kY value=”0.8” kTheta value=”0.8”

	Methodology: slam with KF for correction (grid based) will mention the modifications to this part: 1- we get the initial robot position and orientation from the sensor. (Only for the first reading). 2- get velocity readings from the odom sensor vx = odom.twist.twist.linear.x vy = odom.twist.twist.linear.y vTheta = odom.twist.twist.angular.z dt = (odom.header.stamp -self.prevTime).to_sec() 3- get the predicted position and orientation based on the velocities → linear equation : Xnew = Xold + V * dt 4- correct the robot data using the observations and kalman gain : X corrected = X pred + K (difference)

Team Contribution:

Name	Sec	BN	Contribution
Donia Abdel Fattah	1	28	Mapping , Robot Control
Raghad Khaled	1	30	SLAM,Sensor Incorporating and Alignment
Menna Allah Ahmed	2	29	Mapping , Robot Control
Nada El-sayed	2	32	SLAM,Sensor Incorporating and Alignment