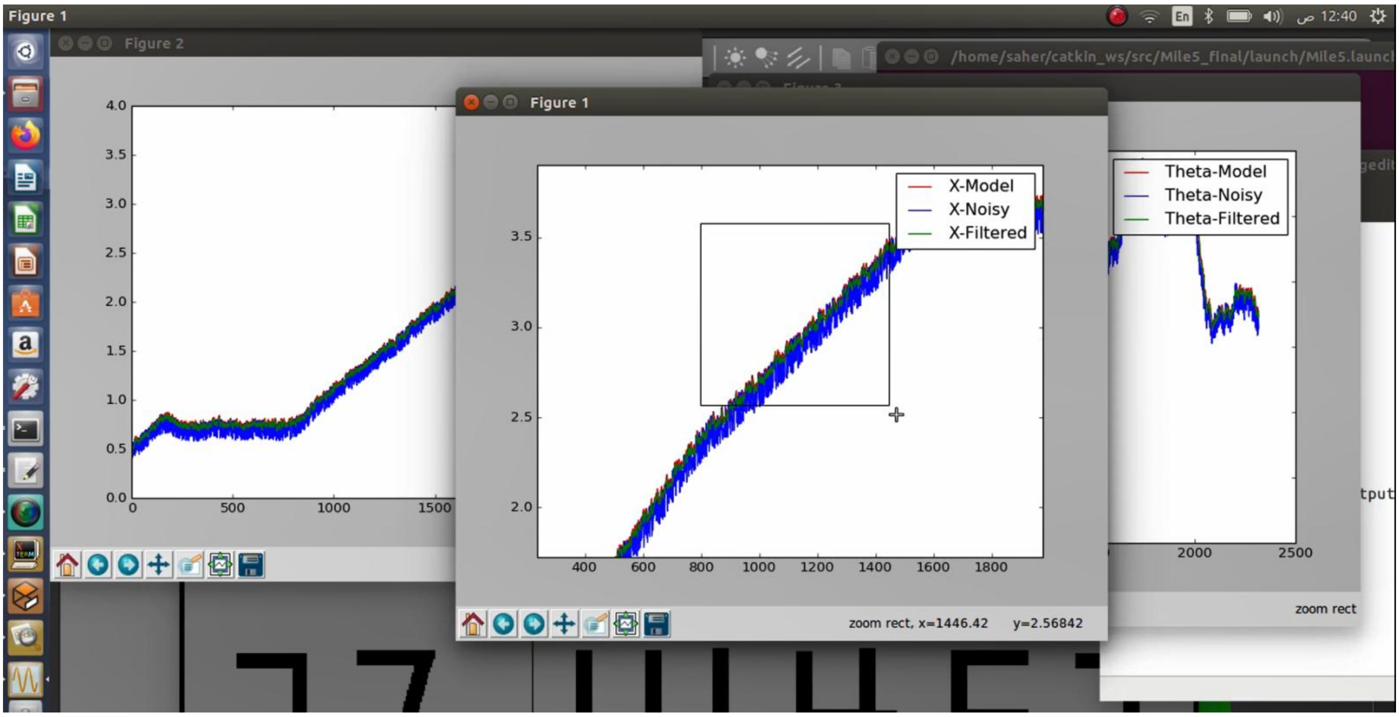
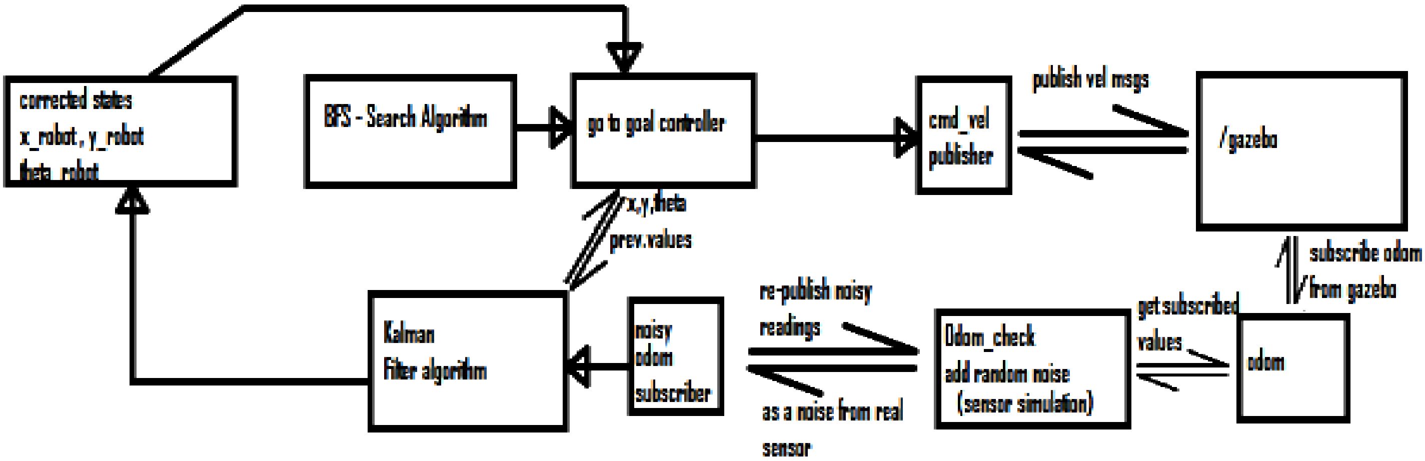
in this report :

the milestone was about making a filtration for sensor readings to get an estimated corrected reading from the sensor , and this can be done by (even simulating a sensor and it`s noise) , or by reading values from real position sensor , then implementing a filtration technique to avoid the noise from the sensor in reality , for example we have some techniques that i learned (Complementary filter : this takes a noisy readings about the linear accelerations (x,y) and by help of (/odom) it implements a filtration technique to get the corrected position ,& the other is kalman filter for only linear systems , and also extended kalman filter (EKF) , the tutorial codes given was only taking the sensor reading (x_p) about the x-axis , thus i implemented it 3 times and this succeeded to implement filtration techniques as shown



and then by implementing a closed loop control with the previous milestones work , the Target was changed as following :



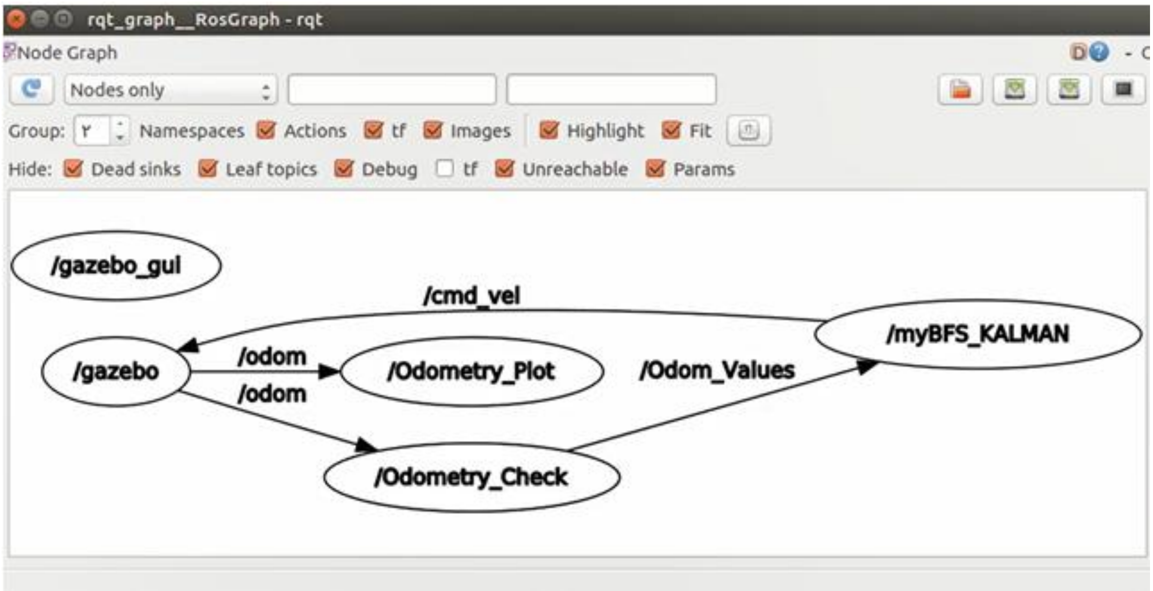
saher mohamed

as shown the symbol ⇌ , means that there`s a connection between blocks , but they are separate files , means that "odom_check" is separate file that is also launched from launch file automatically , and the other arrow symbol means that the blocks are implemented in same python file "for example all of { BFS + kalman filter + goto goal controller are implemented in one python file} and also launched from the same launch file that launched

saher mohamed

"odom_check" file , it was very difficult to implement it all in one as this , but it`s done instead of making larger numbers of publishers and subscribers .

The rqt_graph was somehow similar to the schematic i draw in my head ,



rqt_graph__RosGraph - rqt

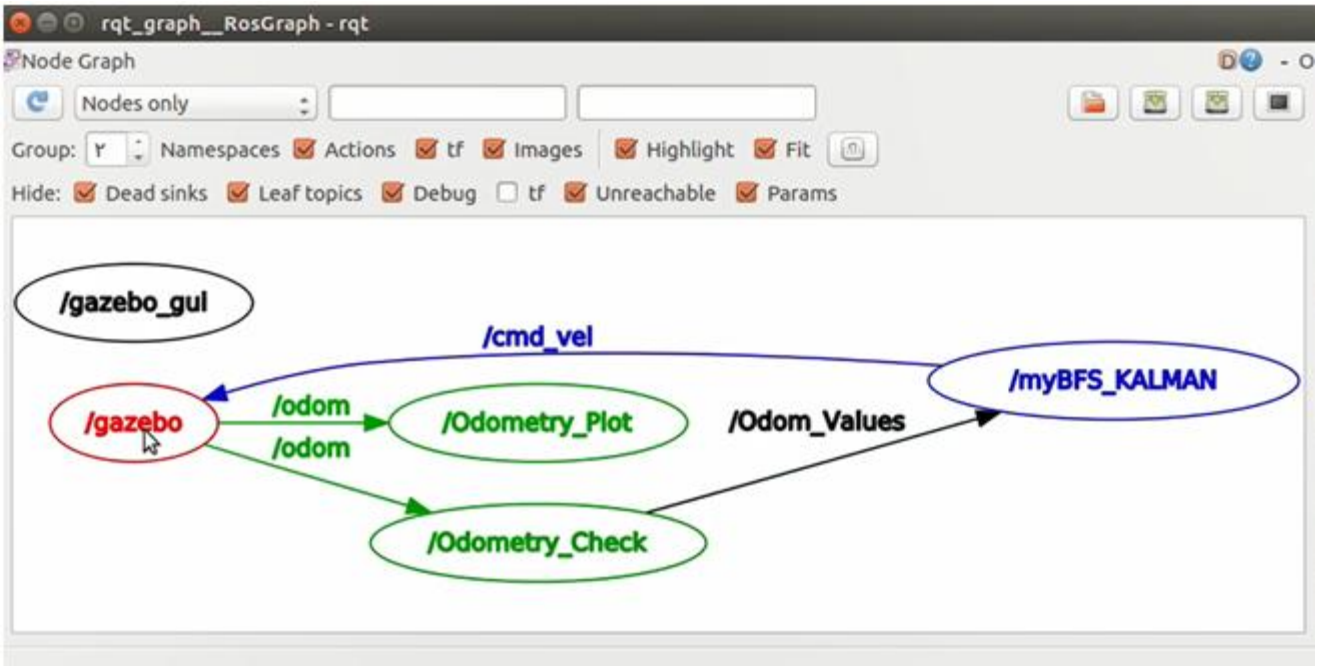
Node Graph

Nodes only

Group: ٢ Namespaces ☑ Actions ☑ tf ☑ Images ☑ Highlight ☑ Fit

Hide: ☑ Dead sinks ☑ Leaf topics ☑ Debug □ tf ☑ Unreachable ☑ Params

and of course by hovering it will get then differnet colors , means that they are influencing each other



rqt_graph__RosGraph - rqt

Node Graph

Nodes only

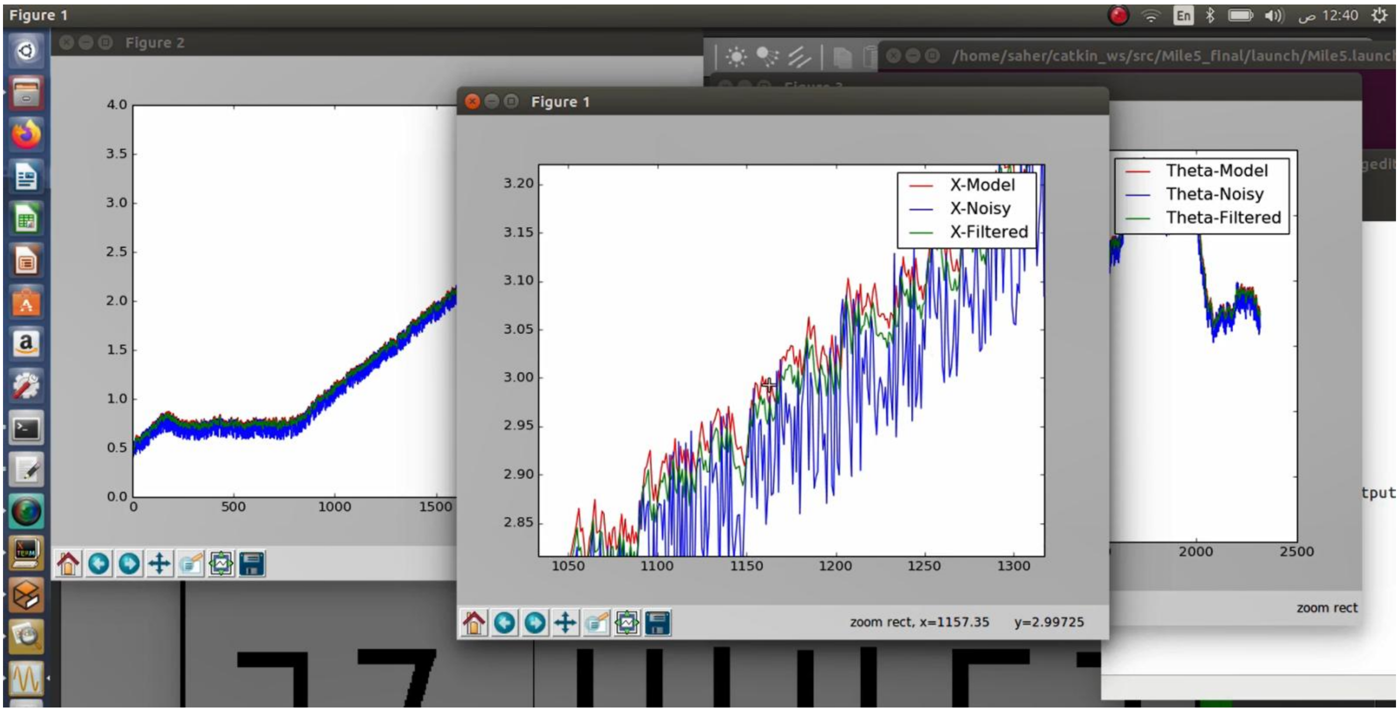
Group: ٢ Namespaces ☑ Actions ☑ tf ☑ Images ☑ Highlight ☑ Fit

Hide: ☑ Dead sinks ☑ Leaf topics ☑ Debug □ tf ☑ Unreachable ☑ Params

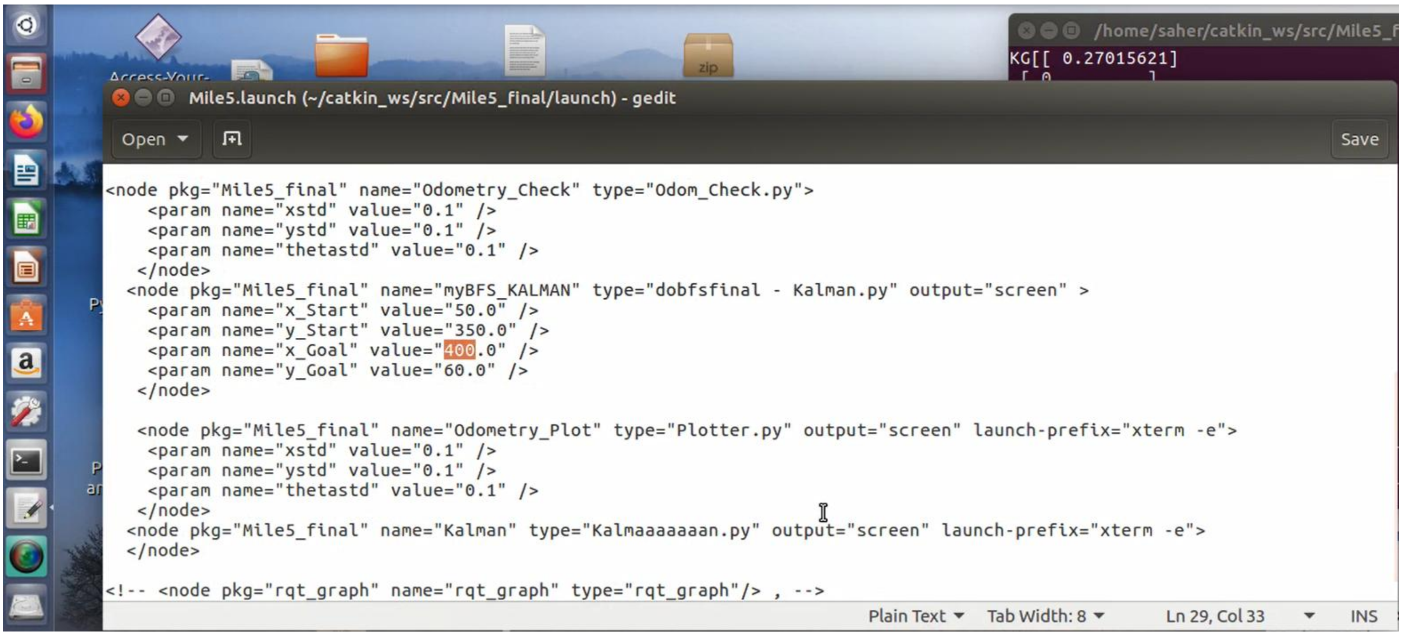
as the code in "myBFS_KALMAN" is ordered as well first to get the start and goal points from launch file , which first is to initialize ros node " myBFS_KALMAN " and get ros parameters , then opens input image into 2D binarry array of integers , then start to implement the BFS algorithm , and then juming into creating subscriber for the noisy data from odometry_check which simulates the sensor noise ,then get into the kalman part which initialize it , and in the Main loop (while the robot not reach it`s goal) which is last (re-scaled) element in the output BFS path array , it will go inside it .

inside the loop starts as kalman filter as well , but first it check if it reached first temp goal point "which inside the BFS path array" and if it not , it will then get the noisy x , y , theta and then get their filtered values , by calling the prediction and filteration methods , then sets (curr_x,y,theta) to the controller and calculates the "rho" value and muliplies it in the gain and implement rest steps of normal goto goal controller , and if the robot reached first temp goal in array , it will increment the counter (BFS array pointer) to reach to it , then re-enters the main loop again , then re-do the steps that explained above , and while doint these steps it puts (model / predicted values) and (Noisy valuys) and (Filtered values) for each x,y,theta , inside some arrays for plotting .

after it exits the main loop (reached the last temp goal in BFS array path) it sets robot angular and linears velocities to zero then shutdown the ros publisher , and starts plotting the arrays that was filled by "(model / predicted values) and (Noisy valuys) and (Filtered values) for each x,y,theta" and shows up the final figures , and by zooming into one of the output figures it shows up (the model / predicted state , noisy values , and filtered),as shown below.



and the image below shows the launch file that responsible for operating the previous files , and it was explained in the video the scale trick for setting start and goal points , and the "odom_check" python file also takes ros parameters which is the standard deviation of the (x,y,theta) to add the noise according to them , as when the variance of the "Gaussian / 'Normal Distribution' curve increases the noise will increase ,which means the curve will be wider ", and the "narrower the curve , the less variance/noise" .


Mile5.launch (~/catkin_ws/src/Mile5_final/launch) - gedit
Open
Save
/home/saher/catkin_ws/src/Mile5_f
Plain Text
Tab Width: 8
Ln 29, Col 33
INS

```
KG[[ 0.27015621]
```

```
<node pkg="Mile5_final" name="Odometry_Check" type="Odom_Check.py">
    <param name="xstd" value="0.1" />
    <param name="ystd" value="0.1" />
    <param name="thetastd" value="0.1" />
   </node>
  <node pkg="Mile5_final" name="myBFS_KALMAN" type="dobfsfinal - Kalman.py" output="screen" >
    <param name="x_Start" value="50.0" />
    <param name="y_Start" value="350.0" />
    <param name="x_Goal" value="400.0" />
    <param name="y_Goal" value="60.0" />
   </node>

   <node pkg="Mile5_final" name="Odometry_Plot" type="Plotter.py" output="screen" launch-prefix="xterm -e">
    <param name="xstd" value="0.1" />
    <param name="ystd" value="0.1" />
    <param name="thetastd" value="0.1" />
   </node>
  <node pkg="Mile5_final" name="Kalman" type="Kalmaaaaaaan.py" output="screen" launch-prefix="xterm -e">
  </node>

<!-- <node pkg="rqt_graph" name="rqt_graph" type="rqt_graph"/> , -->
```