Robotics and Automation

ECE – 2008

PROJECT REPORT

PID Controlled Quadcopter and ROS Implementation

Under the Guidance of

Prof. Sanjay Kumar Singh

School of Electronics and Communication Engineering

Vellore Institute of Technology (VIT)

Vellore, TamilNadu - 632014

Team Members:

- Ayushman Mookherjee – 18BEC0888
- A Raahul – 18BEC0883
- Vineet Rajpal – 18BEC0908
- Rahul BR – 18BEC0984

Abstract:

Quadcopter, also known as quadrotor, is a helicopter with four rotors. Stability is a very important factor when it comes to any UAV. Given all the dynamics in an environment it becomes difficult to adjust the stability and hence the flight of the quadcopter. The purpose of our project is to present the mathematical model of quadcopter dynamics and to develop proper methods for stabilization and control. At the same time to help validate our findings and efforts simulation of a quadcopter has been conducted using ROS.

Our project includes the stabilization of a quadcopter by using PID controllers to regulate its four basic movements: roll, pitch, yaw angles, and altitude. We provide differential equations of the quadcopter dynamics derived from the Newton-Euler equations. Control method and stabilization is done using PID controller.

For the simulation our system comprises of IMU which consists of accelerometer and gyro sensors to determine the system orientation and speed control of four BLDC motors to enable the quadcopter fly in six directions. The quadcopter in put into a gazebo world and the pitch, roll and yaw responses of quadcopter is obtained and PID controller is used to stabilize the system response.

The simulation holds good and is able to reflect our approach and findings. We have been able to stabilize the drone to a certain extent and will definitely help in long distance flight paths. The quadcopter is responsive and control is decent.

List of Symbols, Abbreviations and Nomenclature:

PID: Proportional, Integral, Derivative

IMU: Inertial Measurement Unit

BLDC: Brushless DC

DC: Direct current

UAV: Unmanned Aerial Vehicle

ROS: Robotics Operating System

VTOL: Vertical Take Off and Landing

SDF: Simulation Description Format

URDF: Unified Robotic Description Format

XML: Extensible Markup Language

Introduction:

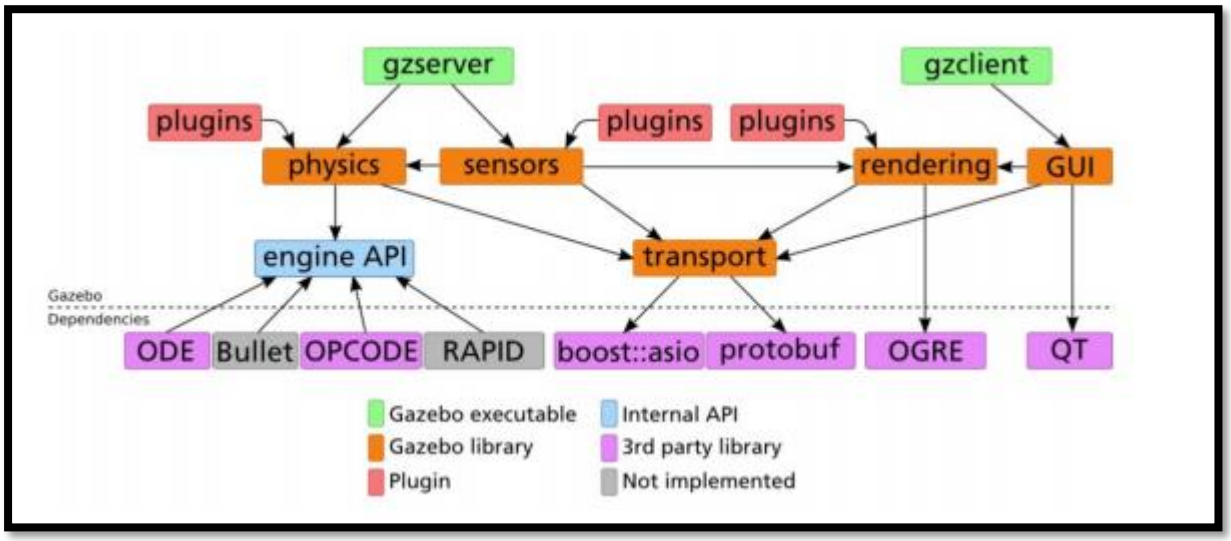
Quadcopter, also known as quadrotor, is a helicopter with four rotors. Stability is a very important factor when it comes to any UAV. Given all the dynamics in an environment it becomes difficult to adjust the stability and hence the flight of the quadcopter. The purpose of our project is to present the mathematical model of quadcopter dynamics and to develop proper methods for stabilization and control. At the same time to help validate our findings and efforts simulation of a quadcopter has been conducted using ROS. Our project includes the stabilization of a quadcopter by using PID controllers to regulate its four basic movements: roll, pitch, yaw angles, and altitude. PID control provides a continuous variation of output within a control loop feedback mechanism to accurately control the process, removing oscillation and increasing process efficiency.

We saw the modelling of a four rotor UAV known as the quadrotor aircraft. This paper explains the developments of a PID control method to obtain stability in flying the Quad-rotor flying object. The model has four input forces which are basically the thrust provided by each propeller connected to each rotor with fixed angle.

The methodology gave us further insight into stabilization with the help of PID controller, however we do not intend to follow the Gesture Control or HIL aspect of the paper.

Methodology:

- **Gazebo:**
  Gazebo is a 3D dynamic simulator with the ability to accurately and efficiently simulate populations of robots in complex indoor and outdoor environments Gazebo is a 3D simulator, while ROS serves as the interface for the robot. Combining both results in a powerful robot simulator. With Gazebo you are able to create a 3D scenario on your computer with robots, obstacles and many other objects. Gazebo also uses a physical engine for illumination, gravity, inertia, etc.



- **Robot Modelling:**
  SDF, is an XML format that describes objects and environments for robot simulators, visualization, and control. Originally it was developed as part of the Gazebo robot simulator with scientific robot applications in mind. For ROS we would be using an URDF model, however since we are using an online simulator, we are sticking with the SDF model created for the Gazebo worlds. Gazebo has 5 type of plugins out of which 4, namely world, model, sensor and visual are specified in the SDF file.
  For the development of the project we have created a xacro file to create the links of our quadcopter which is basically an XML file. After creating the code we have launched the rviz file to visualise our robot. Next we have opened the gazebo simulation which is basically the robot simulation in an environment so for that we have created a launch file where we have added the code of our robot with the initial positions and all that is the position where the robot is launched on the environment and to control the robot in the gazebo simulation we have also added a gazebo plug in code in the xacro(XML Macros) file.

- **Flight Control and Modeling:**
  To help model the flight we use various ROS plugins. The most important being the libplugin_drone. This helps us dictate the initial parameters of the model and flight parameters. It helps set poles, inertia and masses. This is the key to the control of the quadcopter. This plugin unlike the lift drag plugin that simulates the conditions of the wind collisions which lead to the lift of the object, this plugin emulates the physics behind it and the simulations that thus take place implements the motion of equations. The IMU sensor plugin provides us with the values from the simulation that give us further insight for our project.

  Equations of motion:

  Φ = Roll angle u1= force on the propellers u2= moment

  m= mass = 0.2Kg i= moment of inertia = 0.1 units

  $Z'' = g - (u1/m).\cos\Phi$ $Y'' = (u1/m).\sin\Phi$ $\Phi'' = u2/i$

  Upon linearization :

  $Z'' = g - (u1/m).\cos\Phi \rightarrow g - (u1/m)$

  $Y'' = (u1/m).\sin\Phi \rightarrow g\,\Phi$

  $''\Phi = u2/i \rightarrow$ remains the same

  The system is linearized around the hover point, where the properties are:

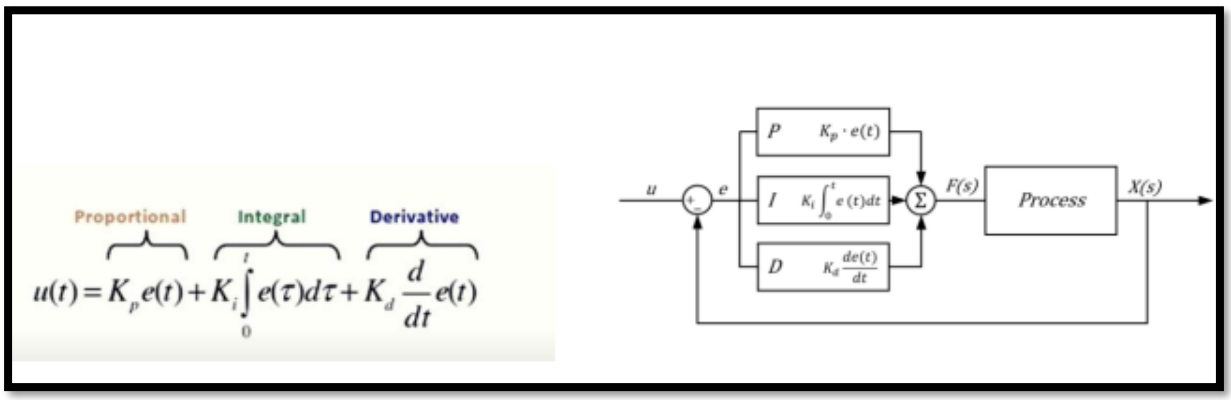
  $u1 = mg,\ u2 = 0,\ \Phi = 0,\ Y = Yo,\ Z = Zo$

  $u1 = mg + \Delta u1,\ u2 = \Delta u2,\ \Phi = 0 + \Delta\Phi,\ Y = Yo + \Delta Y,\ Z = Zo + \Delta Z$

- **PID Controller**

A PID controller is an instrument used in industrial control applications to regulate temperature, flow, pressure, speed and other process variables. PID controllers use a control loop feedback mechanism to control process variables and are the most accurate and stable controller.

A PID controller's made of 3 parts: the proportional term, the integral term and the derivative term, which all get added together to produce a single output. Every second the PID gets input from the sensors and using the desired position set by whatever control method the drone uses (transmitter, CV, etc), finds the difference(error) between it and that desired position. That error value gets fed into all 3 equations and eventually multiplied by each's respective constants to get the output



P-Controller:

Proportional or P- controller gives output which is proportional to current error e (t). It compares desired or set point with actual value or feedback process value. The resulting error is multiplied with proportional constant to get the output. If the error value is zero, then this controller output is zero.

I-Controller:

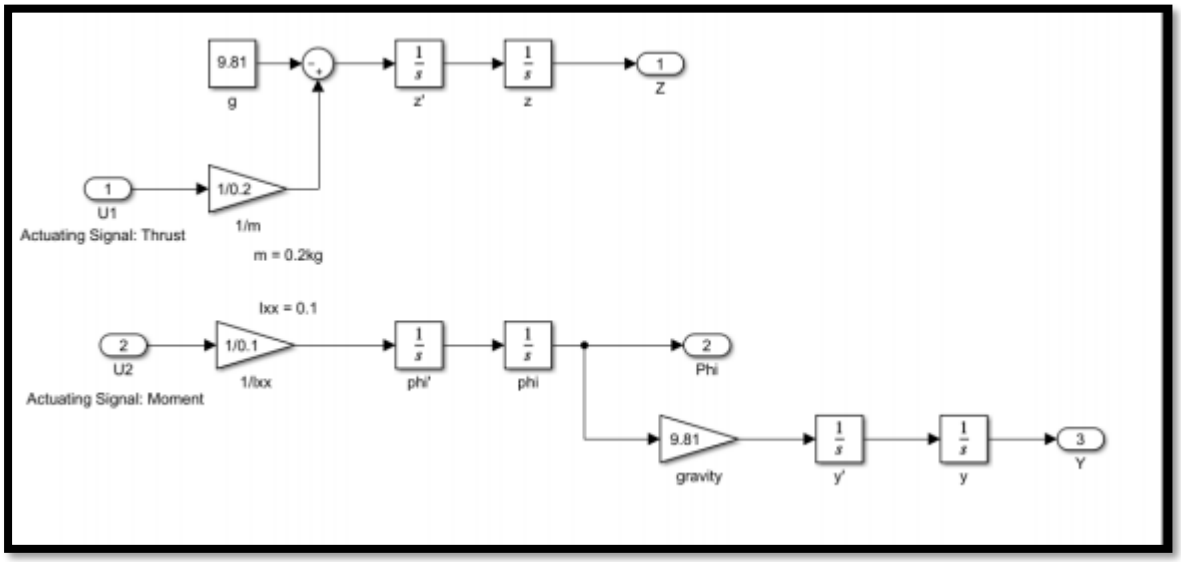
Due to limitation of p-controller where there always exists an offset between the process variable and set point, I-controller is needed, which provides necessary action to eliminate the steady state error. It integrates the error over a period of time until error value reaches to zero. It holds the value to final control device at which error becomes zero. Integral control decreases its output when negative error takes place. It limits the speed of response and affects stability of the system. Speed of the response is increased by decreasing integral gain Ki

D-Controller:

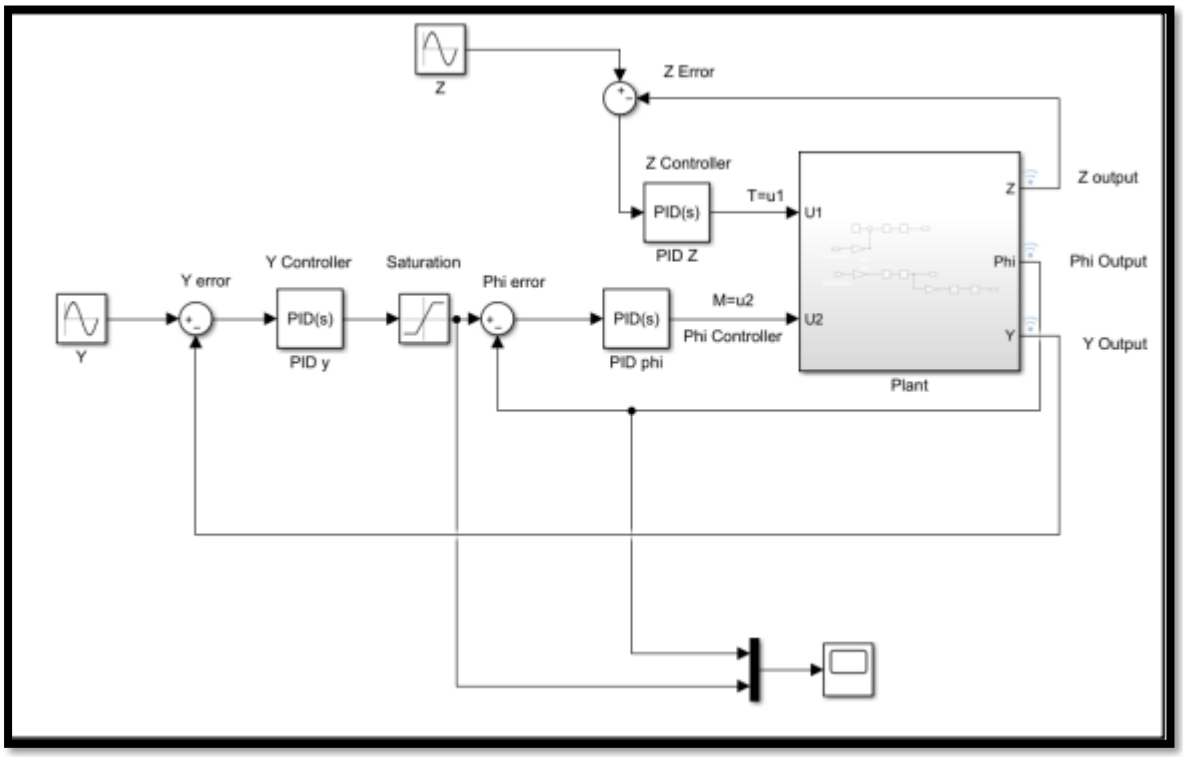
I-controller doesn't have the capability to predict the future behavior of error. So it reacts normally once the set point is changed. D-controller

overcomes this problem by anticipating future behavior of the error. Its output depends on rate of change of error with respect to time, multiplied by derivative constant. It gives the kick start for the output thereby increasing system response.

- **Simulink Plant Design:**



- **Controller Block Diagram:**



Results and Discussions:

The best part about our project is that we were able to make a PID Flight controller that gave us a stabilized flight with a given path taken and able to fly a real- life model of a Quadcopter on entirely virtual platform, while experimenting the drone, we had the drone get stuck to a wall and the drone would not come out unless we had given the command for it to do so, this gave us an opportunity to learn various new concepts and implement them.

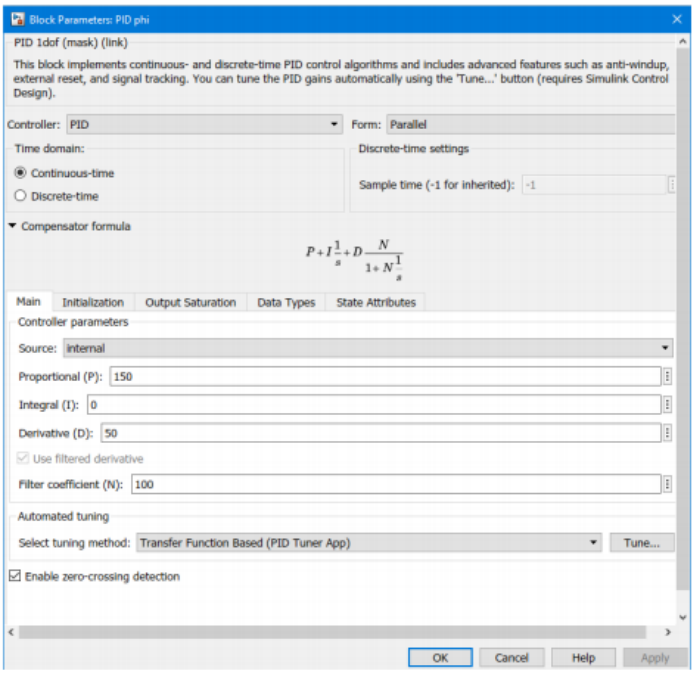


Fig 3.1

PID ϕ

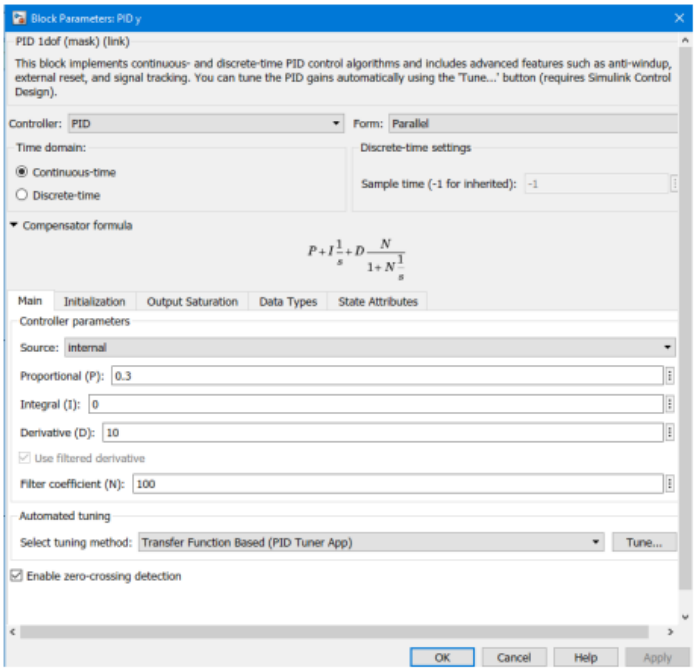


Fig 3.2

PID y

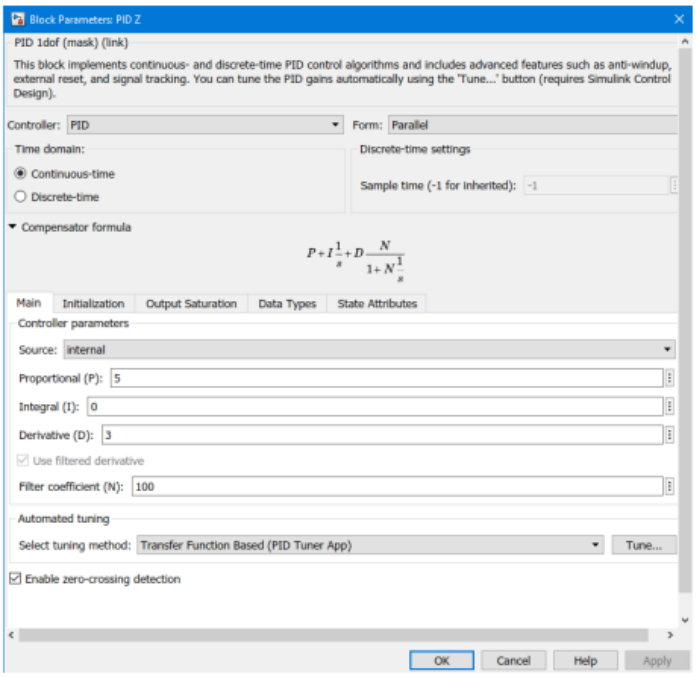


Fig 3.3

PID z

Performance Analysis

Fig 3.4

-----> Z position, Reference and actual

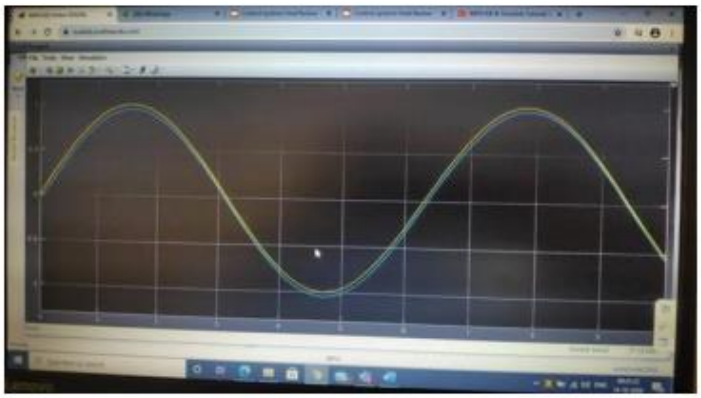


Fig 3.5

-----> Y position, Reference and actual

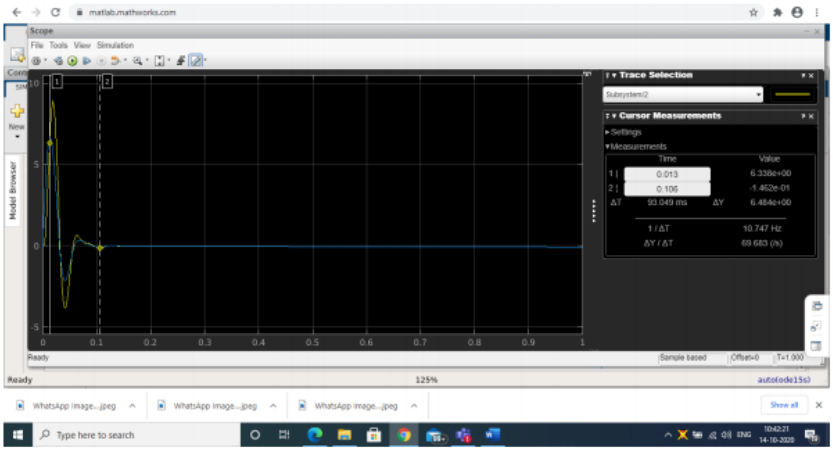


Fig 3.6

-----> Steady state reached at 0.101sec

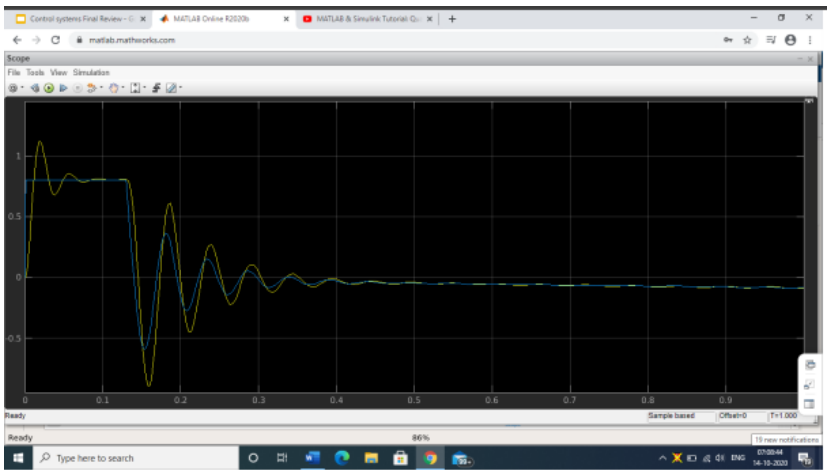


Fig 3.7

P = 50, I = 0, D = 30

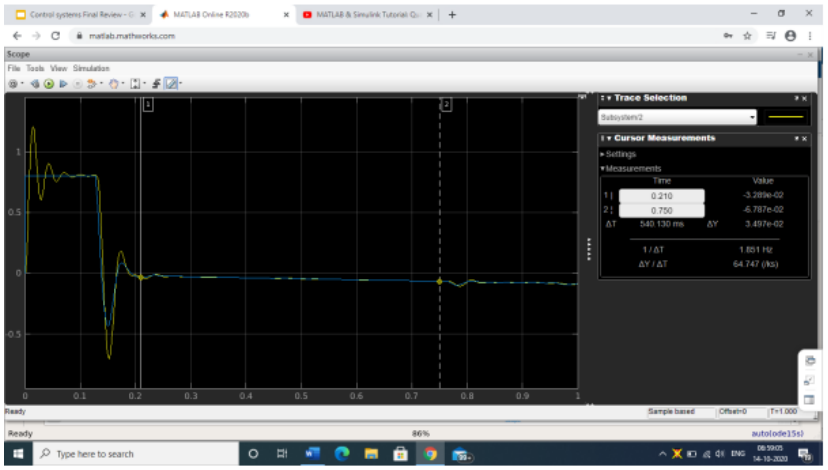


Fig 3.8

P = 150, I = 0, D = 5

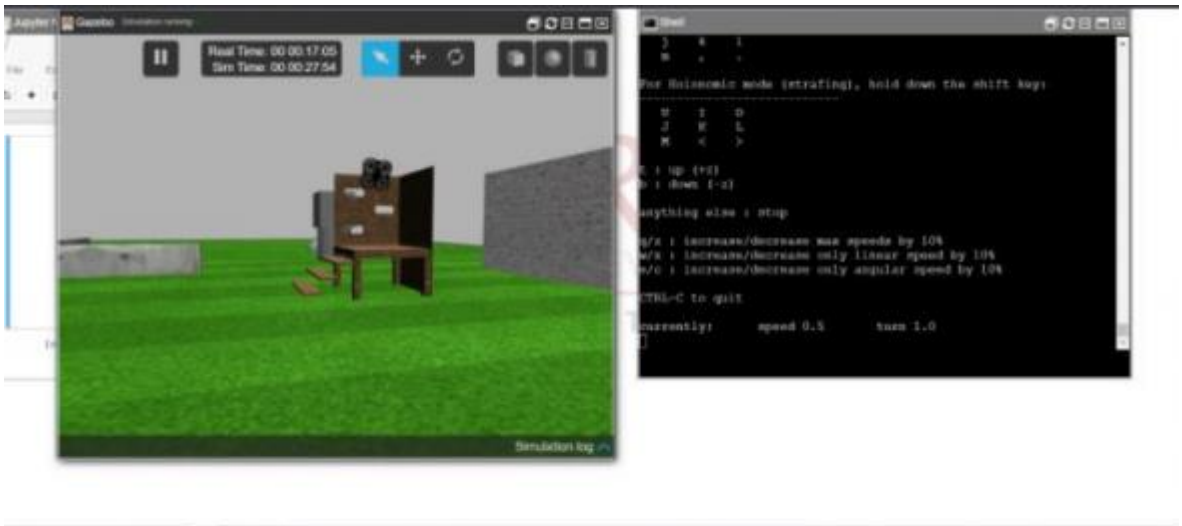
The graph of angle v/s time to check roll angle stablization was analyzed. The settling time was found to be around 0.101 sec. As three PID controllers were used graphs for z and y positions were also successfully obtained and observed. On comparison with the input signal, we see that the PID controlled signal has lesser peak variations and greater stability.

1.Upon simulation an angle v/s time graph is obtained that gives the analysis for roll angle stabilization.

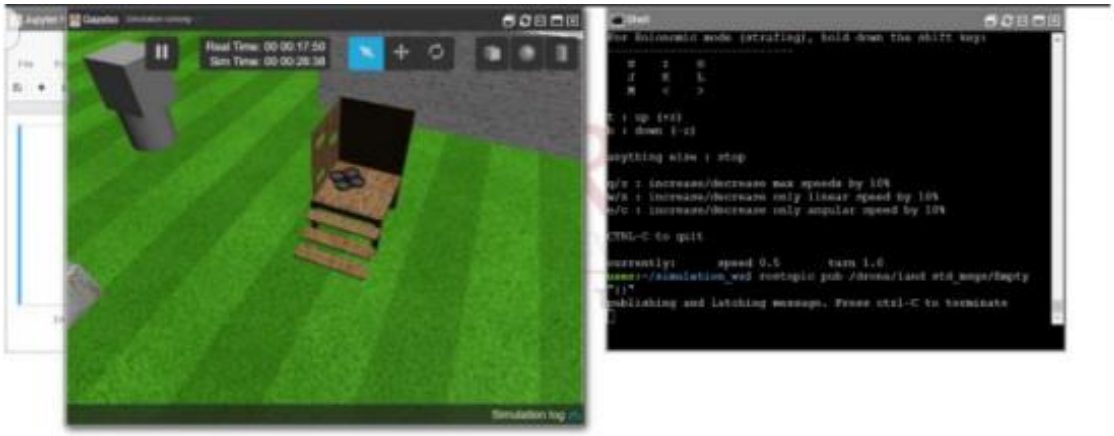
2. Time to reach stability is lesser when PID is used than when there is no PID used.

3.Using the inner loop P value considerably higher than the outer loop yielded a stable system.

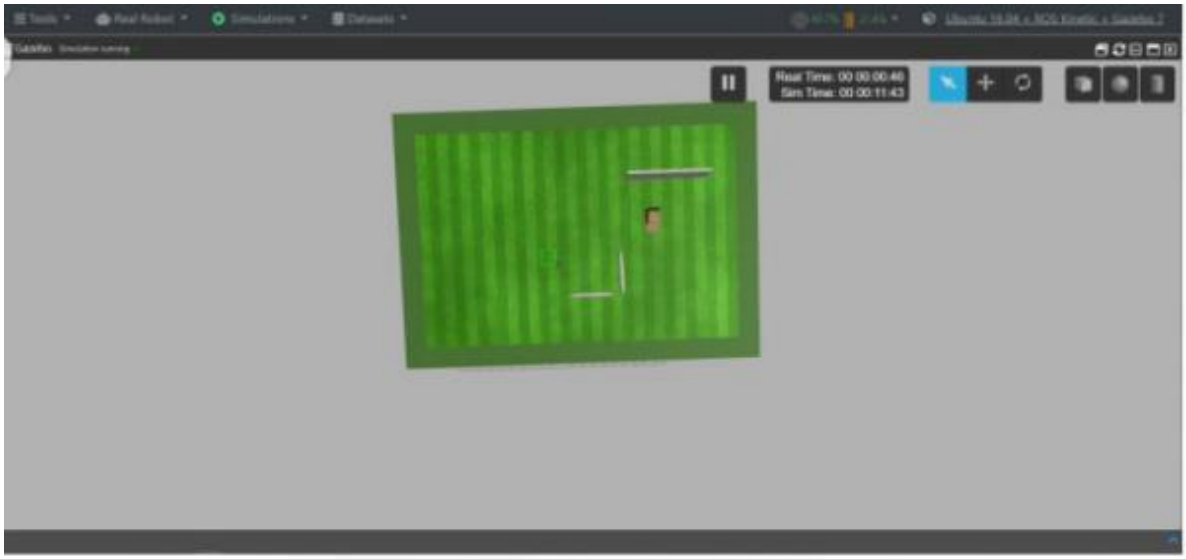
In these figures below, the world is simulated in the gazebo world the world is like a real time example and has commodities that replicate the real world this helps in visualizing the project and the movement of the robot. The spawning of the world file is very important in the gazebo since it helps in picturing how the project will function and later also helps in visualizing various scenarios.

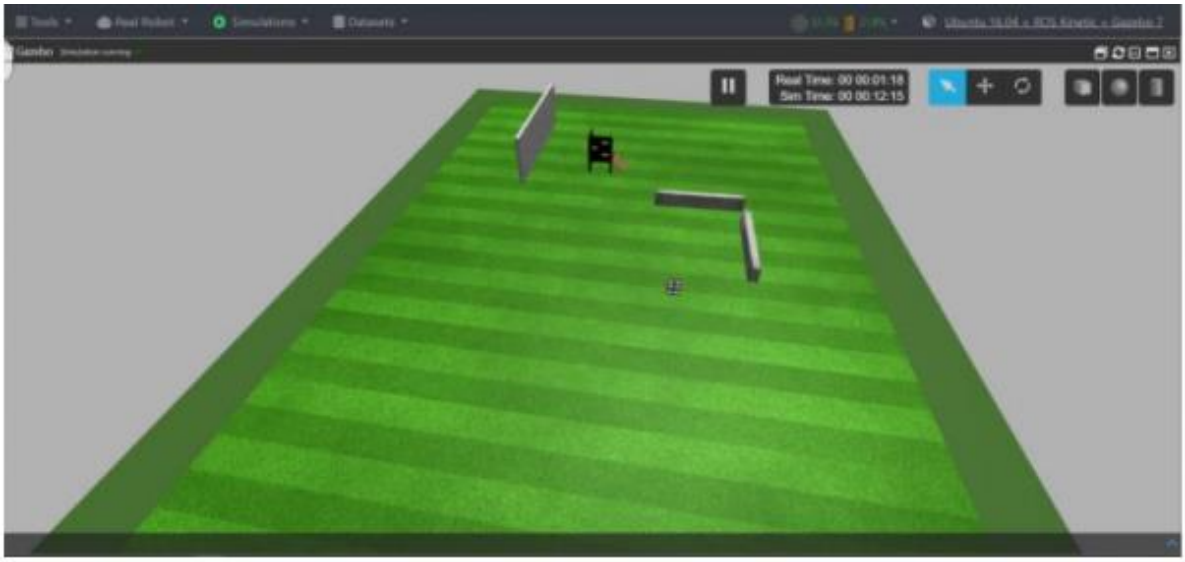


drone stuck



landed





Conclusions:

We have learnt a completely new software the ROS software, we worked on a virtual operating system, and developed a product we are extremely proud of

Our quadcopter serves the following purpose: Quadcopter is a nice project for learning PCB design. Including power, motor driving, signal processing. Design schematic and layout PCB require detail work, need to pay attentions to datasheet for each chip, it is easier than physical work (soldering) since PC software could point out most of problems. Fabrication and soldering are challenges, a little careless would result solder touch each other and it is hard to notice. But all these are solvable once we have more experience on PCB design, like math problems, do more practice.

Since the quad rotor is in a state where it can fly, there are several areas in the hardware, software, controller, and model that can be improved in order to get the quad rotor flying to be useful in the defense sector. Of all the changes that can and need to be made, it looks like the hardware issues are currently most limiting to achieving flight.

Recommendations:

http://docs.ros.org/en/melodic/api/moveit_tutorials/html/doc/getting_started/getting_started.html

http://udel.edu/~sagarptl/QuadcopterFinalReport.pdf

http://depts.washington.edu/soslab/mw/images/archive/2/2b/20100612072345!Group3_finalreport.pdf

https://github.com/NishanthARao/ROS-Quadcopter-Simulation

https://github.com/NishanthARao/ROS-Quadcopter-Simulation

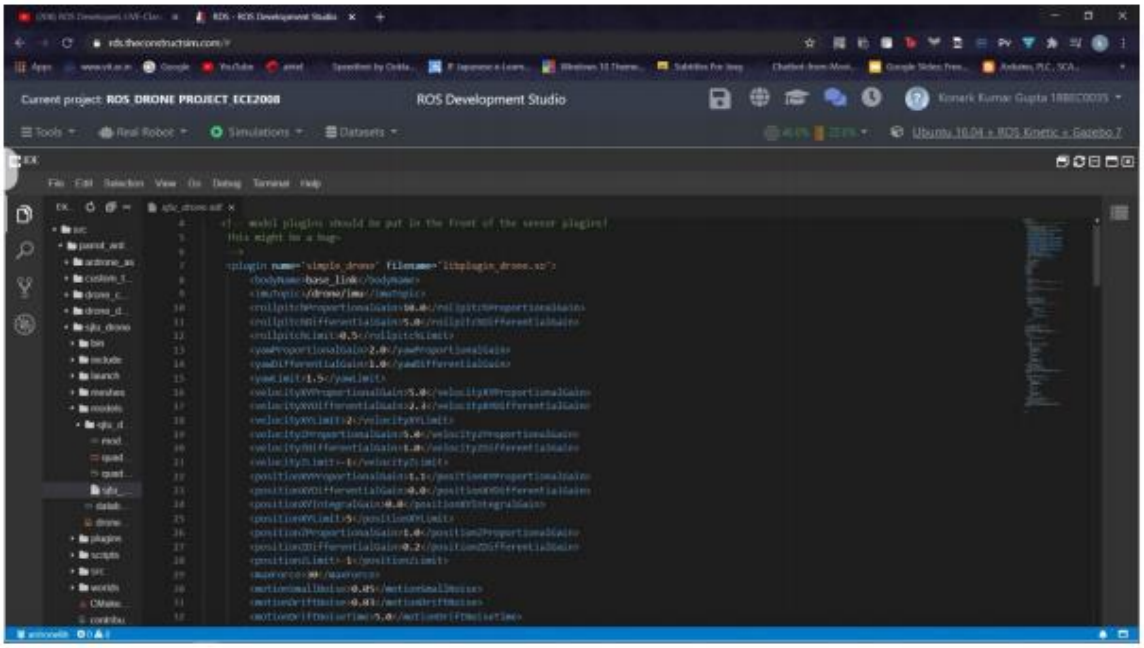
References:

1. Atheer L. Salih, M. Moghavvemi; Haider A. F. Mohamed, Khalaf Sallom Gaeid, “Modelling and PID controller design for a quadrotor unmanned air vehicle”, 2010 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)
2. Vitul Varshney, Melvin Wilson, Sakthivel Sivaraman, “PID based Stabilization of Gesture Controlled Drones using HIL Simulation”, International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958, Volume-3 Issue-6, August 2014

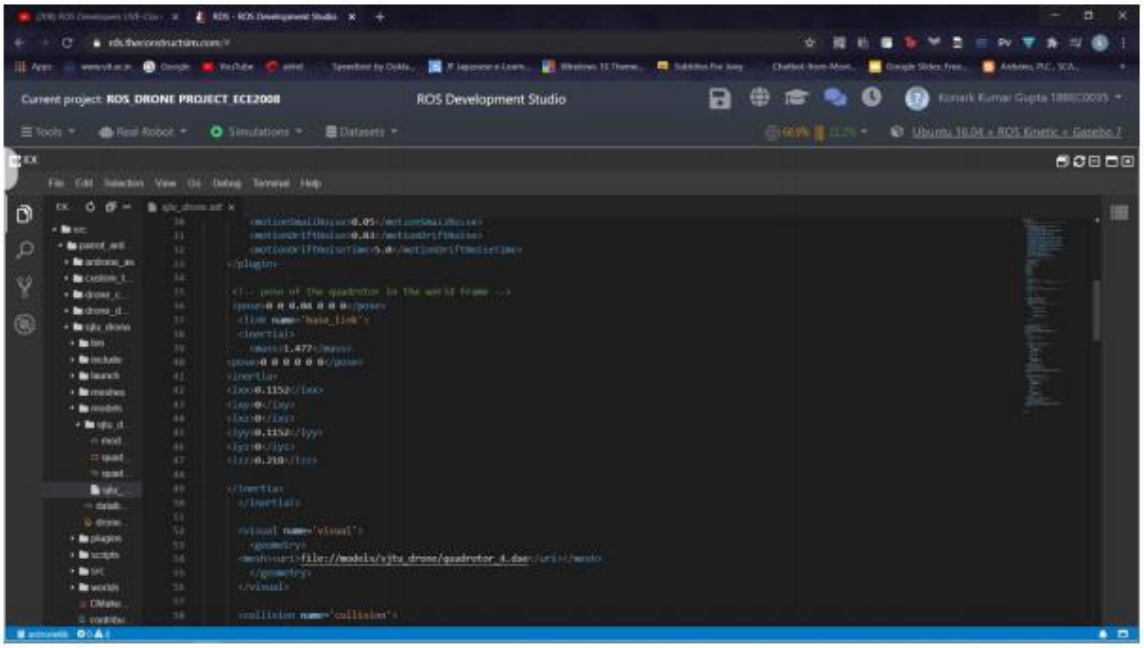
3. Viswanadhapalli Praveen, Alexander Sendobry, Stefan Kohlbrecher, Uwe Klingauf, Oskar Von Stryk “Comprehensive Simulation of Quadrotor UAVs Using ROS and Gazebo”, I J C T A, 9(15), 2016, pp. 7151-7158.
4. Johannes Meyer, Anju S. Pillai, “Modeling and Simulation ofQuadcopter using PID Controller”, DOI: 10.1007/978-3-642-34327-8_36, Proceedings of the Third international conference on Simulation, Modeling, and Programming for Autonomous Robots, November 2012.
5. C. Sciortino and A. Fagiolini, "ROS/Gazebo-Based Simulation of Quadcopter Aircrafts," 2018 IEEE 4th International Forum on Research and Technology for Society and Industry (RTSI), Palermo, 2018, pp. 1-6, doi: 10.1109/RTSI.2018.8548411.
6. H. S. Paing, A. V. Schagin, K. S. Win and Y. H. Linn, "New Designing Approaches for Quadcopter Using 2D Model Modelling a Cascaded PID Controller," 2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering
7. Modelling and control of quadcopter-Teppo Luukkonen,Independent research project in applied mathematics August 22, 2011 Aalto University
8. Quadcopter stabilization by using PID controllers-Luis E. Romero, David F. Pozo(IEEE member), Jorge A. Rosales(IEEE member)

Appendix:

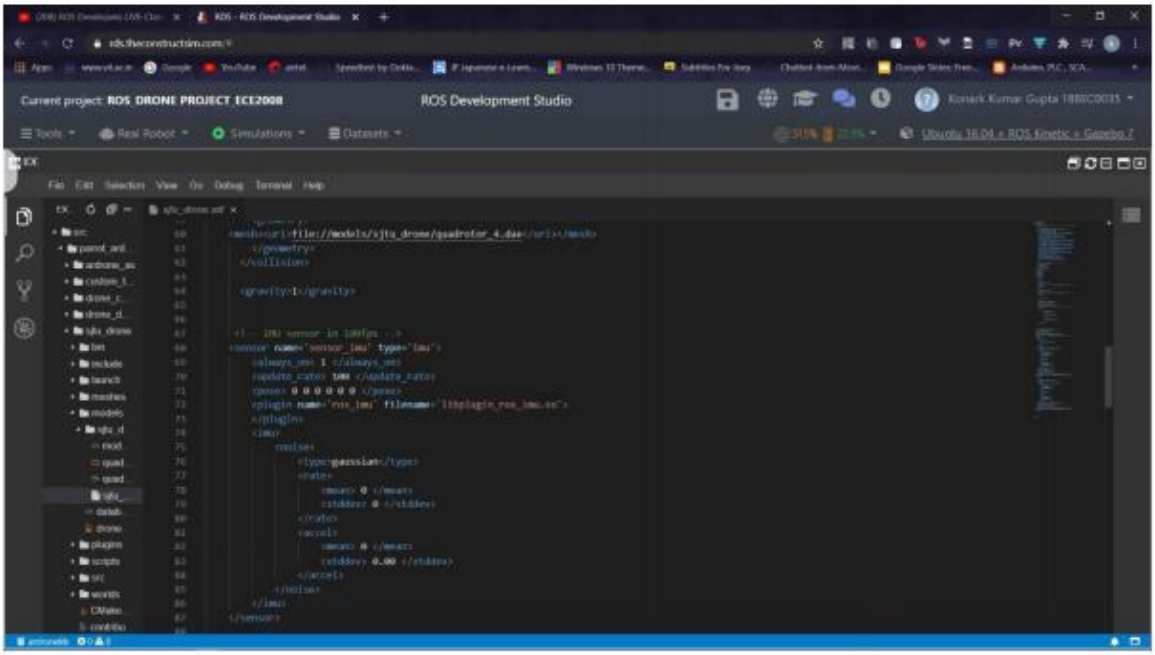
Drone Plugin



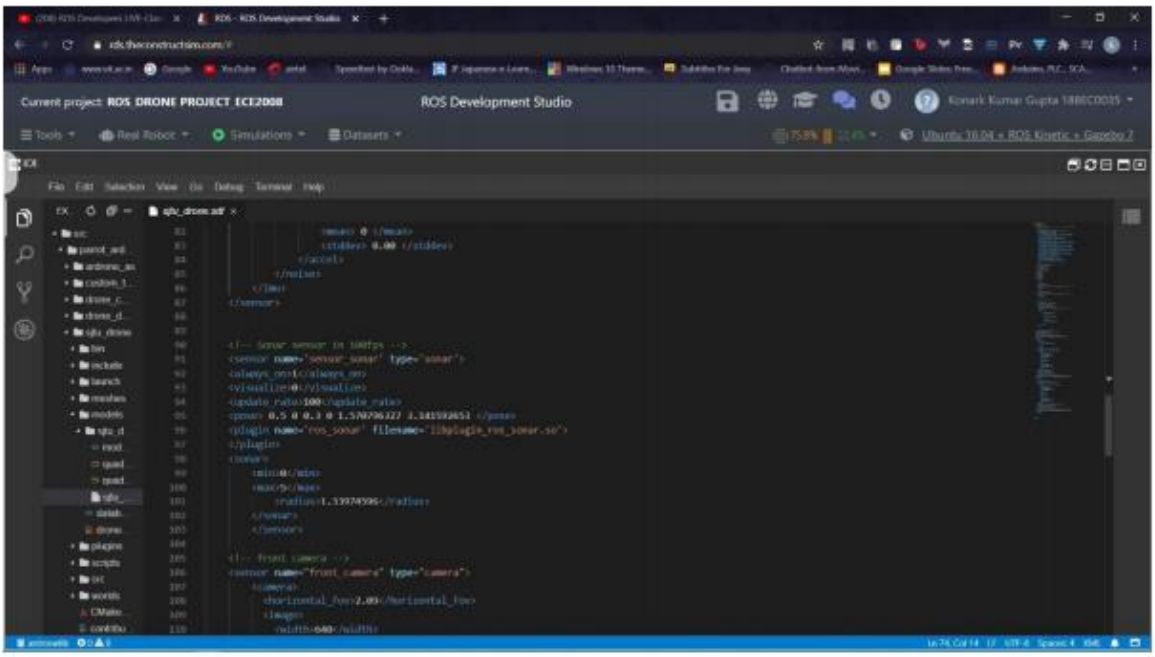
Position Vectors for copter



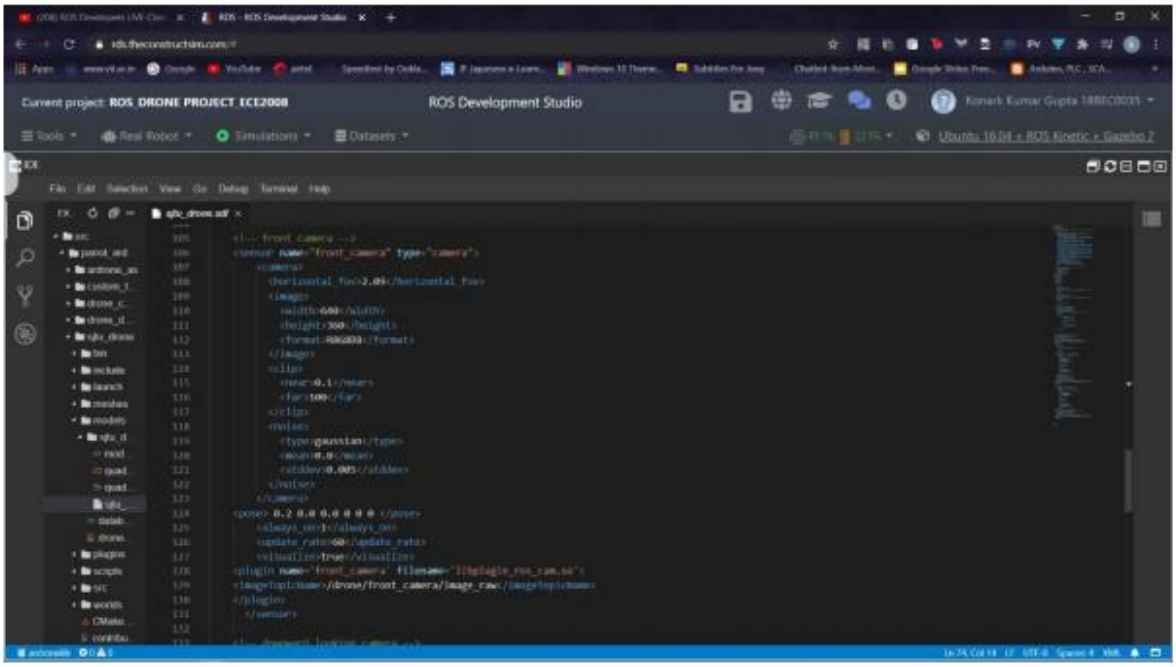
IMU sensor



Sonar



Front camera



Downward camera