MATLAB Drone Code Competition

Overview

For this competition, we will be using MATLAB's Simulink with the inbuilt parrot drone package. This package allows us to build upon a pre existing framework of a drone that was originally made for educational purposes. MATLAB allows us to design the logic for the drone within Simulink itself, making logic design and error correction easy, as well as compile it to C++ and then Machine level code for uploading it to their physical drone. Flexibility is also provided to make changes to the compiled C++ code.

In this competition, we will be aiming on building the logic within the environment of Simulink as well as visualizing it on Simulink 3D Animation.

Aim

We will be focusing on making a basic position controller for the parrot drone on MATLAB Simulink. For this round we will be looking to achieve main objectives, namely-
-Altitude control
-Angle control
-Position control

As an example, we have already completed the altitude controller. What's left for you is

- Motor Mixing Algorithm
- Yaw Controller
- Attitude Controller
- Position Controller
- Optional - Earth to body frame converter.

Motor Mixing Algorithm

We take the inputs as the Thrust, Roll, Pitch, and Yaw. We require the outputs to be the individual commands to each motor such that we are able to decouple the commands of roll, pitch, yaw and thrust (Basically make it such that the commands can create motion strictly along only one of the mentioned axes.)

Yaw Controller

We take in the inputs of where our ideal yaw and what our yaw currently is. Our task is to output a function to minimize the error in these two values in the most optimal and stable way.

Attitude Controller

Attitude refers to the angular difference between an aircrafts axes and the earths horizon. In our case, we need our multirotor to be as level with the horizon as possible. Your task will be to apply control logic to make sure that the angular difference is minimum.

Position Controller

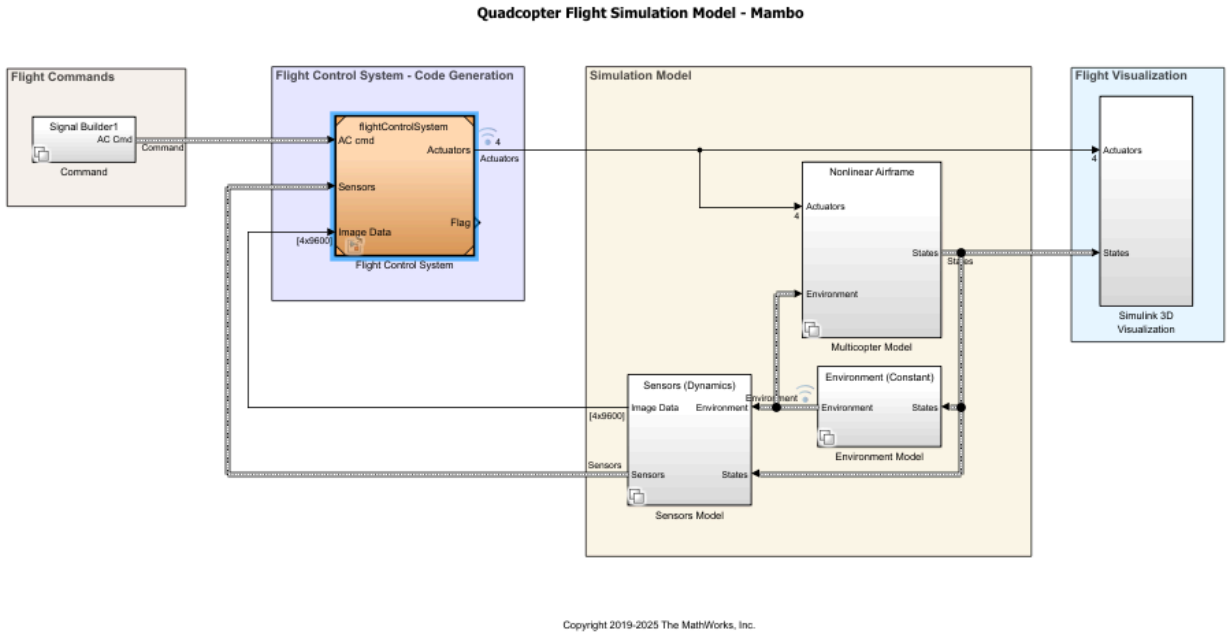
You might notice that after applying just the attitude controller, there is still a drift in position. This is because when we apply only the attitude controller, we only correct any changes in the attitude of the drone. This means that any disturbance in the angle will inadvertently cause a corresponding drift in the position. To counter this, we must apply a controller on the position as well ensuring that the position error is minimized. Your task is to input the actual location of the drone, as well as the current location of the drone, and output angles of pitch and roll such that the error in position of the drone is minimized.

Earth to Body Frame Converter

This is a basic rotation of frames that takes into account any angles in yaw can converts ground coordinate inputs to the body coordinates

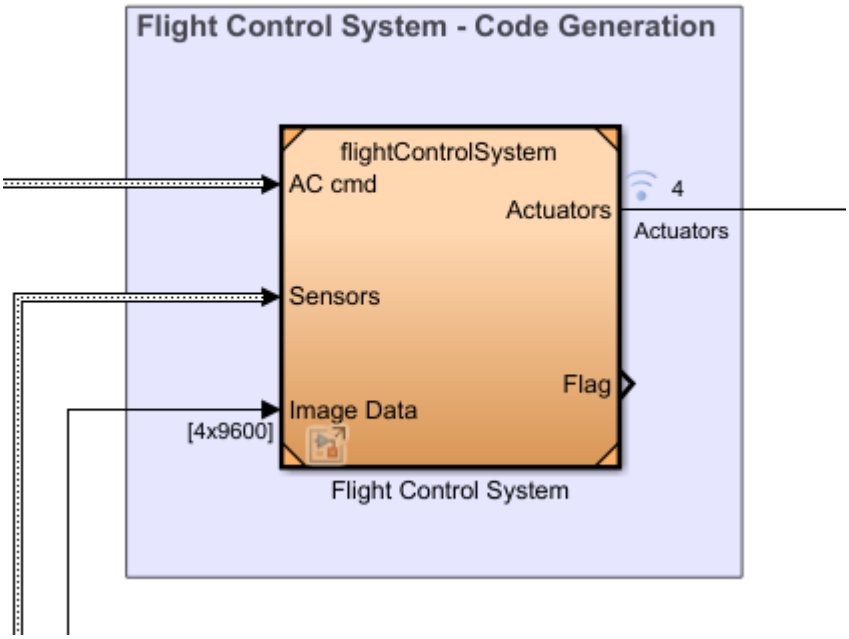
Brief Introduction to the Model

On opening the model in Simulink, the first thing you will see is this
(Middle mouse to pan, scroll to zoom)



Now this might look intimidating, but let's break it down to understand that all these parts are.

Flight Control System (FCS)



This is the only part that we will be focusing on for this competition. It is responsible for all the signal processing and control systems that we will be implementing.

Input

Sensors-

This contains the values of the sensors as well as some noise to simulate what a real sensor on the drone might output.

Image Data-

The parrot drone has a onboard downward facing camera that we can use to perform some onboard image processing for applications like line following as well as for functioning as an optical flow sensor.

AC CMD-

These are the input commands that are responsible for telling the drone what we want to do like when to take off and land.

Output

Actuators-

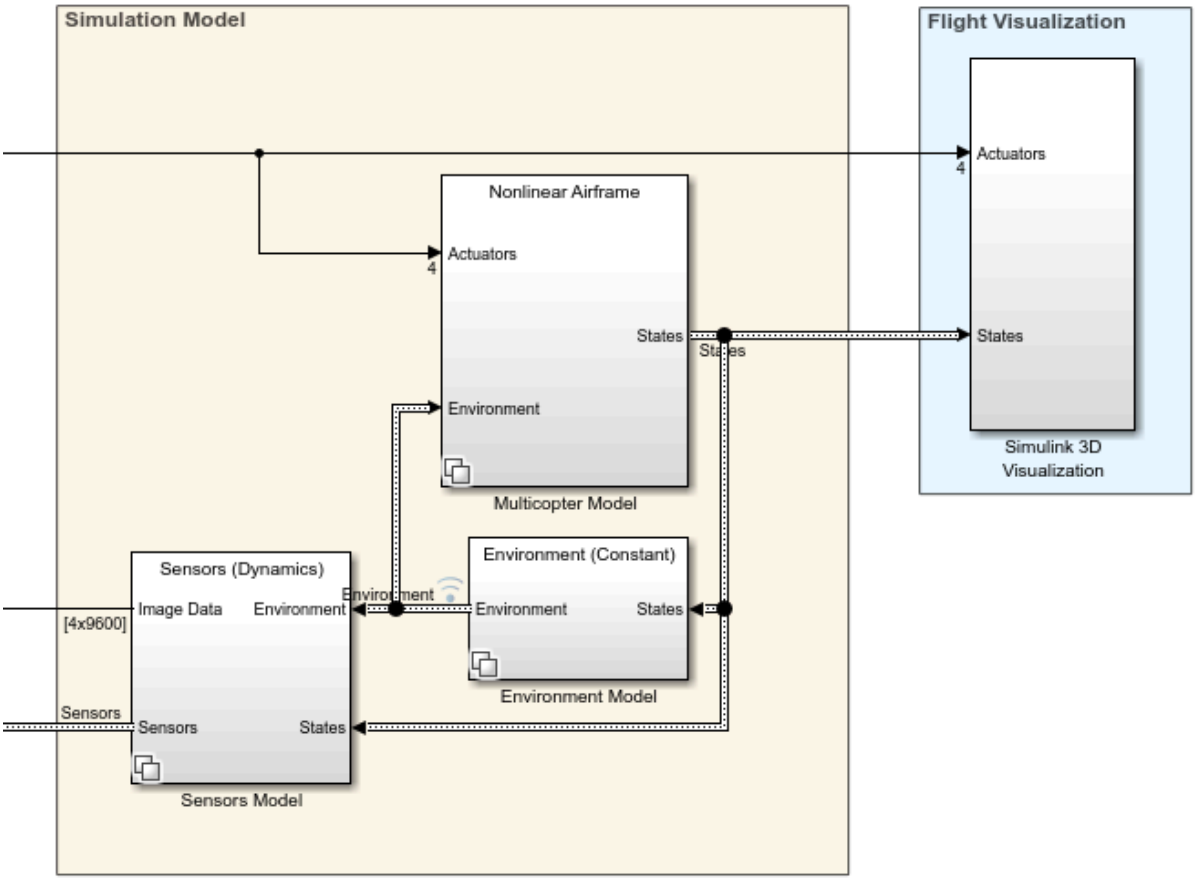
These are the input for the motors which we will be developing the logic for.

Flag-

This was initially a failsafe that shuts down the drone in case it experiences an unstable orientation (high acceleration, high angles etc.). This has been put in place to prevent the hardware and/or the surroundings from getting too damaged when we upload the code onto an actual drone and fly it in an environment. For now, we have disabled this allowing us to experiment with different tuning parameters without the simulation stopping unless the drone hits the ground.

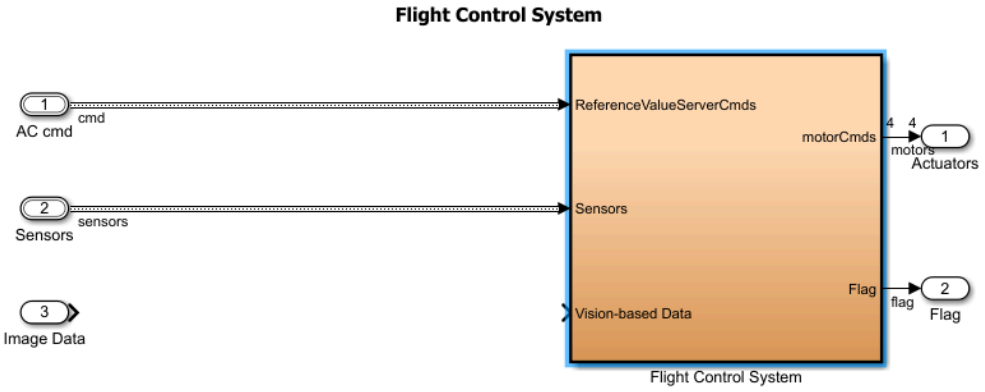
Simulation and Flight Visualization

It is important to not make any changes to this.

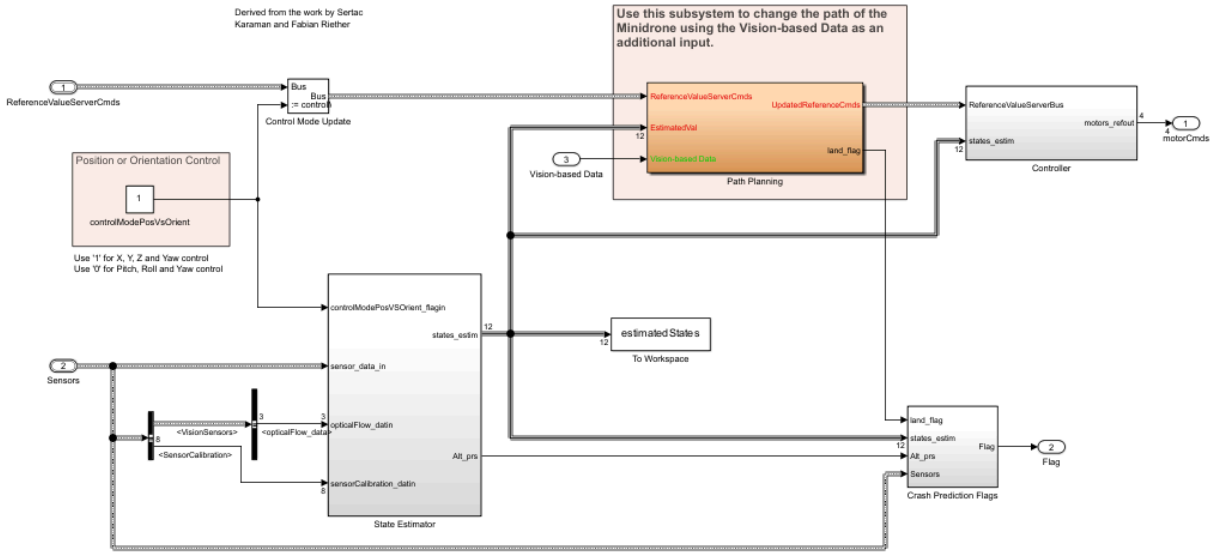


The simulation block is responsible for the simulating the physics, and also providing us with the signal outputs of the sensors.
The Visualization block is responsible for the graphical visualization of the drone in an environment.

We proceed into the FCS block by double clicking on it.
We encounter another FCS block, we enter this one too.



We now see another set of blocks-



Controller

This is the main block in which we will be focusing most of our logic within and where we make our control systems.

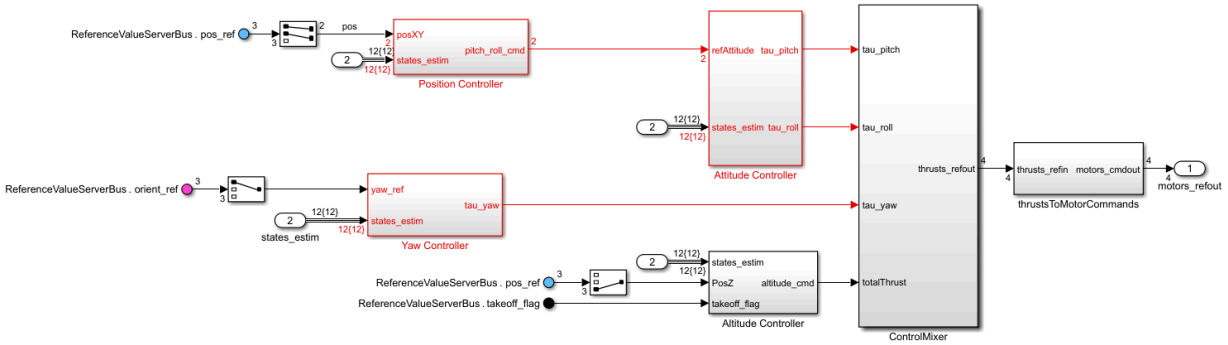
State Estimator

This takes the noisy signal as an input, filters it using filters such as the Kalman Filter and generates a filtered signal as an output. Although this outputted signal is much better than the noisy signal, it is not perfect, meaning that there will still be some small disturbances which might cause some instabilities that we must deal with as we move along.

Path Planning

This block allows us to tell the drone which coordinates we would like it to go to. This does not as such have any control logic, but acts as an input for our controller.

We now have a closer look at our Controller block-

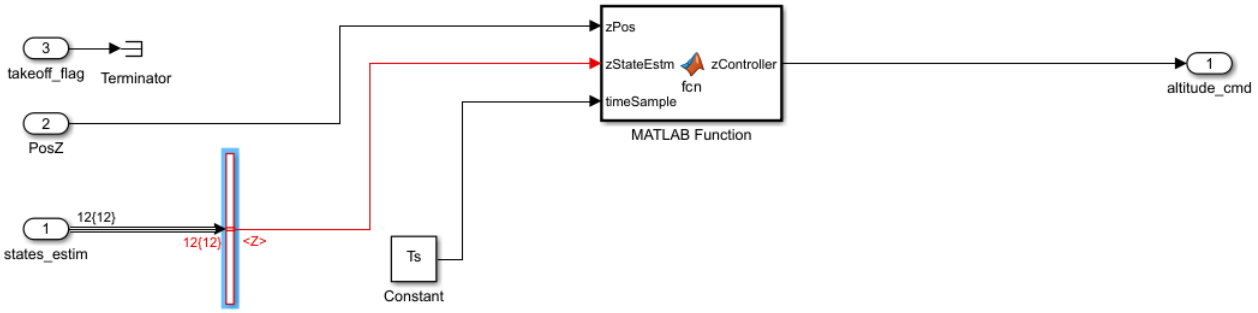


Here we can find some similarities to what our logic actually is.

Altitude Controller

To make understanding Simulink a bit easier, we have completed the Altitude Controller with the help of a single PID loop.

On opening the altitude controller, we see some logic like this-



Here the highlighted rectangular block, is a bus selector. This is used to select one signal from a bus of signal. We use this selected signal to calculate the error for our PID loop. We also take PosZ, which corresponds to the ideal value (The position that the drone is requested to fly to). This is the other part that we use to calculate the error with. The Ts block is the block that is used to find the time Sample of the simulation. The time sample is basically how often the calculations are done. For example if our time sample is set to 0.2, then majority of all calculations are done every 0.2 seconds. The use of this will become apparent when we open the MATLAB Function block.

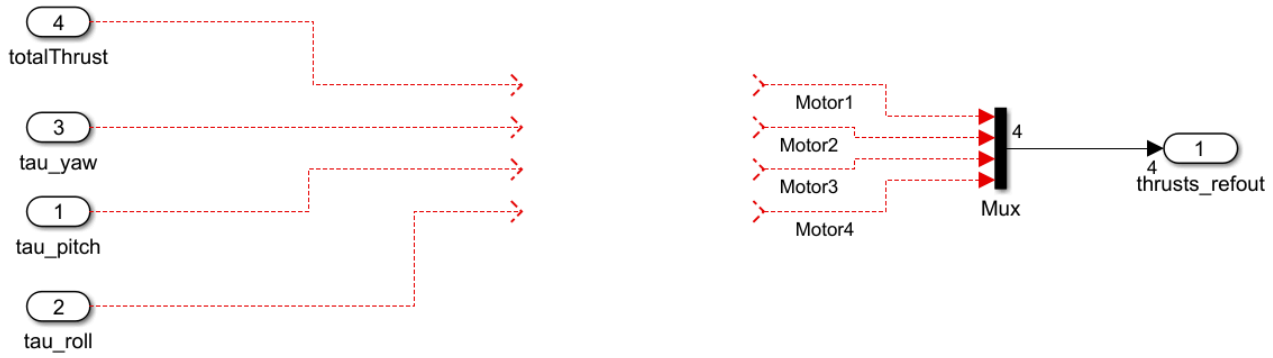
```
function zController  = fcn(zPos, zStateEstm, timeSample)

p = 0.8;
i = 0.25;
d = 0.1;
dt = timeSample;

persistent error
if isempty(error)
    error = single(0);
end

persistent errorInt
if isempty(errorInt)
    errorInt = single(0);
end

lastError = error;
error = single(zPos - zStateEstm);
errorDiff = single((error - lastError) / dt);
errorInt = single(errorInt + error * dt);

zController = single(p * error + i * errorDiff + d * errorInt);
```

Here we see how the function is formed-

```
function zController  = fcn(zPos, zStateEstm, timeSample)
p = 0.8;
i = 0.25;
d = 0.1;
dt = timeSample;

persistent error
if isempty(error)
    error = single(0);
end

persistent errorInt
if isempty(errorInt)
    errorInt = single(0);
end

lastError = error;
error = single(zPos - zStateEstm);
errorDiff = single((error - lastError) / dt);
errorInt = single(errorInt + error * dt);

zController = single(p * error + i * errorDiff + d * errorInt);
```

Similar functions will have to me made for the other controllers. If you want, you could also use block to make the entire logic as add, subtract, constant, discrete integral and derivative exist.

In the Control Mixer-

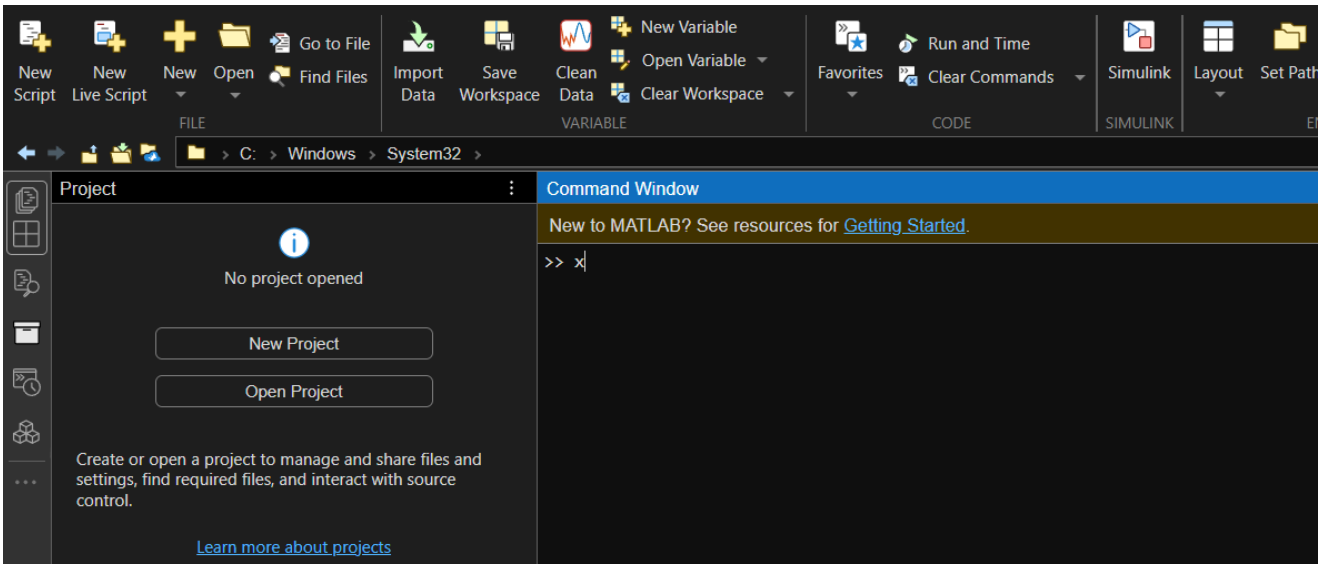


We have Motor1 as our back left motor
We have Motor2 as our Back right motor
We have Motor3 as our Front Right motor
and finally, we have Motor4 as our front Left Motor.

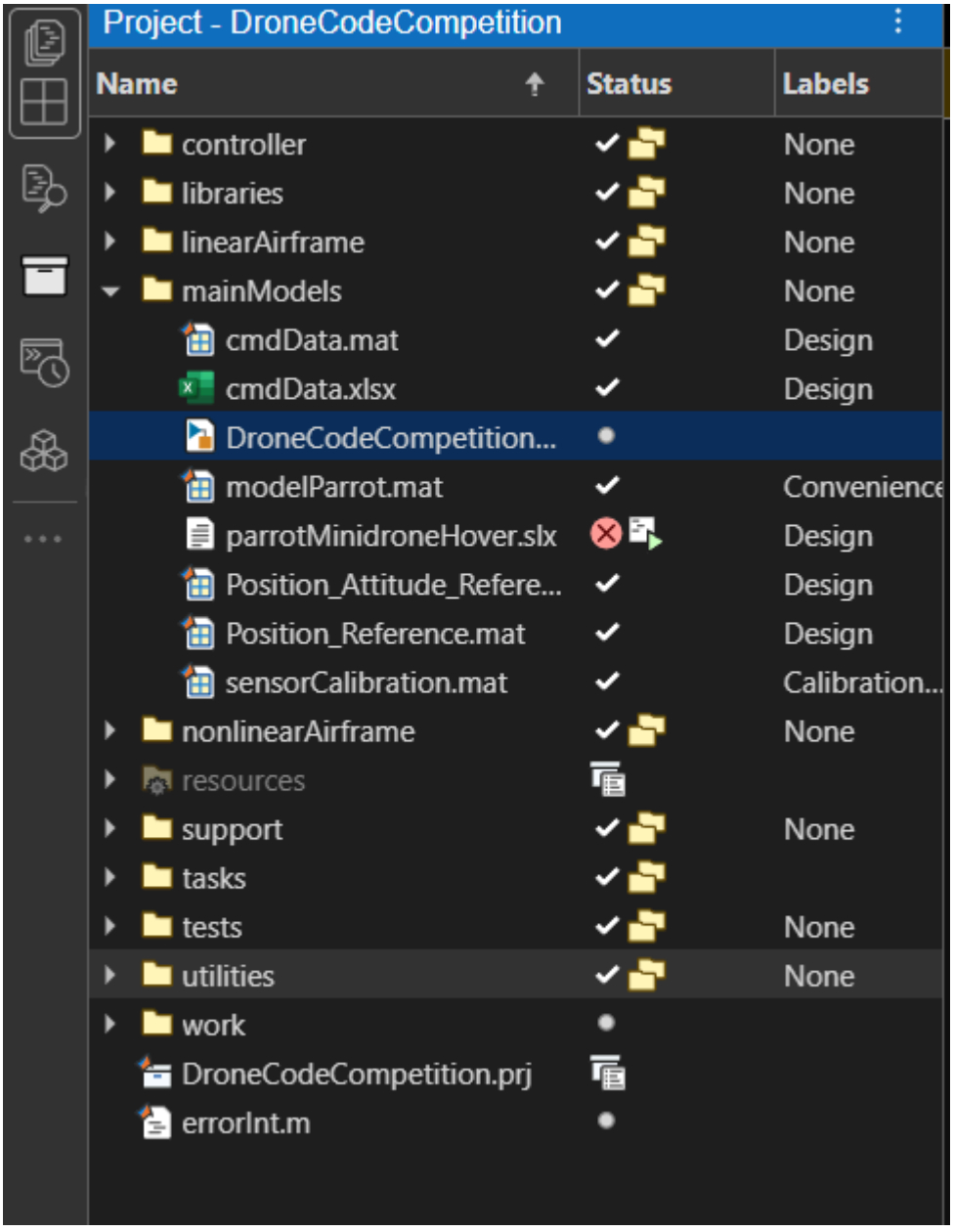
You task will be to make a function taking inputs as tau_pitch, tau_roll, tau_yaw and totalThrust, and generating outputs as motor inputs. Now there is no need to multiply the equations with any constants as this part will be handled later in the logic.

Once we have completed out control mixer block, we can connect the altitude controller to this and run the simulation

Opening the Project Files



Open the project files (DroneCodeCompetition.prj) from the MATLAB Files

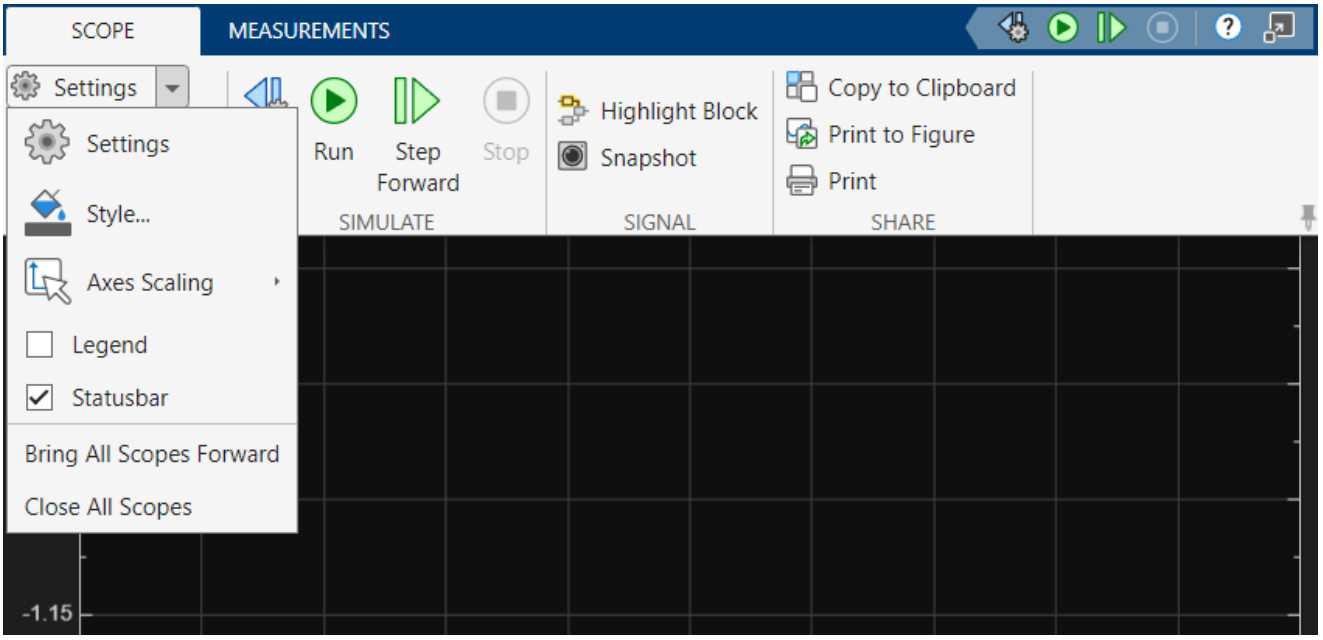


Locate and open the DroneCodeCompetition.slx file from the explorer.

Some Features of MATLAB to Use

Adding Scope

You can use the help of a scope block to have a look at what a value is as the simulation goes on graphically. We just add the value we want to look at into the scope and double click on it before or during the simulation to view the graph.

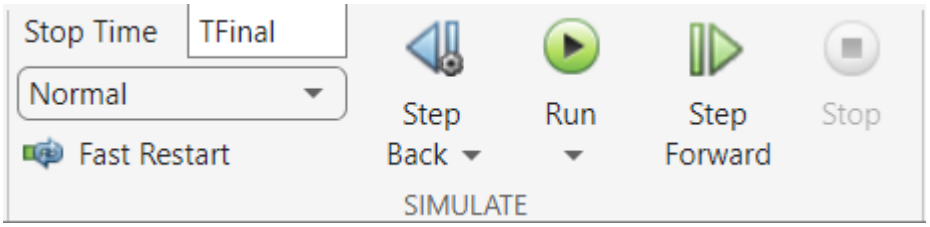


You can also enable the legend in the scope in the case when you would like to view multiple inputs.

Adding Blocks

You can double click anywhere in the Simulink workspace to add any block.

Running The Simulation



Further Resources-

You may be able to complete the competition without these, but if you wish to gain a better understanding, or if you are stuck in a certain part, then you may use these resources.

Understanding more about PID Controllers - PID Control - A brief introduction, Simple Examples of PID Control.
PID Tuning - How to Tune a PID Controller - Made Simple!

MATLAB Basics (Syntax and operations) - MATLAB Resources

For further help, you can refer to MATLABS online courses

MATLAB's official courses for Simulink - Simulink Onramp | Self-Paced Online Courses - MATLAB & Simulink