Project name:- FCND-Controls-CPP
Submitted by:- Bishwajit Kumar

Objective: - To build a controller in C++ along with implementing and tuning these controllers in multiple steps for 5 scenarios(for 5 different controllers).

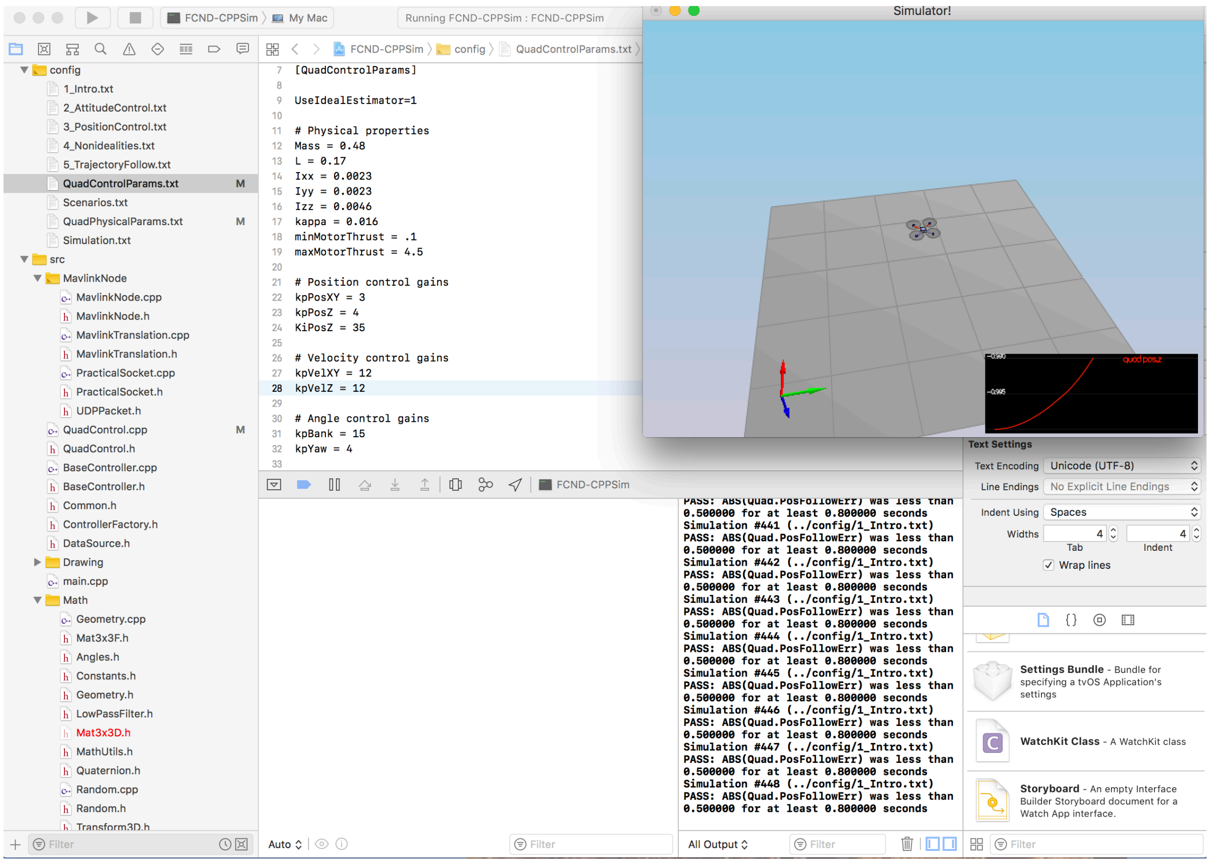
1) Body rate control
2) Roll pitch control
3) Altitude control
4) Lateral control
5) Yaw control

Mainly modification in 2 file was required to complete the project:-

1) /config/QuadControlParams.txt: Contains all control gains and other desired tuning parameters.
2) /src/QuadControl.cpp: Contains all of the code foe different controllers.

Implementation:-

Scenario 1: - Inro_1
Simulator provided for intro scenario required changes in mass value of drone to make it stable. Changed mass to 0.48g to make it stable in QuadControlParams.txt file.



```
[QuadControlParams]

UseIdealEstimator=1

# Physical properties
Mass = 0.48
L = 0.17
Ixx = 0.0023
Iyy = 0.0023
Izz = 0.0046
kappa = 0.016
minMotorThrust = .1
maxMotorThrust = 4.5

# Position control gains
kpPosXY = 3
kpPosZ = 4
KiPosZ = 35

# Velocity control gains
kpVelXY = 12
kpVelZ = 12

# Angle control gains
kpBank = 15
kpYaw = 4
```

```
PASS: ABS(Quad.PosFollowErr) was less than
0.500000 for at least 0.800000 seconds
Simulation #441 (../config/1_Intro.txt)
PASS: ABS(Quad.PosFollowErr) was less than
0.500000 for at least 0.800000 seconds
Simulation #442 (../config/1_Intro.txt)
PASS: ABS(Quad.PosFollowErr) was less than
0.500000 for at least 0.800000 seconds
Simulation #443 (../config/1_Intro.txt)
PASS: ABS(Quad.PosFollowErr) was less than
0.500000 for at least 0.800000 seconds
Simulation #444 (../config/1_Intro.txt)
PASS: ABS(Quad.PosFollowErr) was less than
0.500000 for at least 0.800000 seconds
Simulation #445 (../config/1_Intro.txt)
PASS: ABS(Quad.PosFollowErr) was less than
0.500000 for at least 0.800000 seconds
Simulation #446 (../config/1_Intro.txt)
PASS: ABS(Quad.PosFollowErr) was less than
0.500000 for at least 0.800000 seconds
Simulation #447 (../config/1_Intro.txt)
PASS: ABS(Quad.PosFollowErr) was less than
0.500000 for at least 0.800000 seconds
Simulation #448 (../config/1_Intro.txt)
PASS: ABS(Quad.PosFollowErr) was less than
0.500000 for at least 0.800000 seconds
```

Scenario 2:- Body rate and roll/pitch control

- Implement calculating the motor commands given commanded thrust and moments in C++
  Implemented **GenerateMotorCommands** method in QuadControl.cpp as below: -

```
float l = L / sqrtf(2.f);
float f1 = momentCmd.x / l;
float f2 = momentCmd.y / l;
float f3 = - momentCmd.z / kappa;
float f4 = collThrustCmd;
cmd.desiredThrustsN[0] = (f1 + f2 + f3 + f4)/4.f;
cmd.desiredThrustsN[1] = (-f1 + f2 - f3 + f4)/4.f;
cmd.desiredThrustsN[2] = (f1 - f2 - f3 + f4)/4.f ;
cmd.desiredThrustsN[3] = (-f1 - f2 + f3 + f4)/4.f;
```

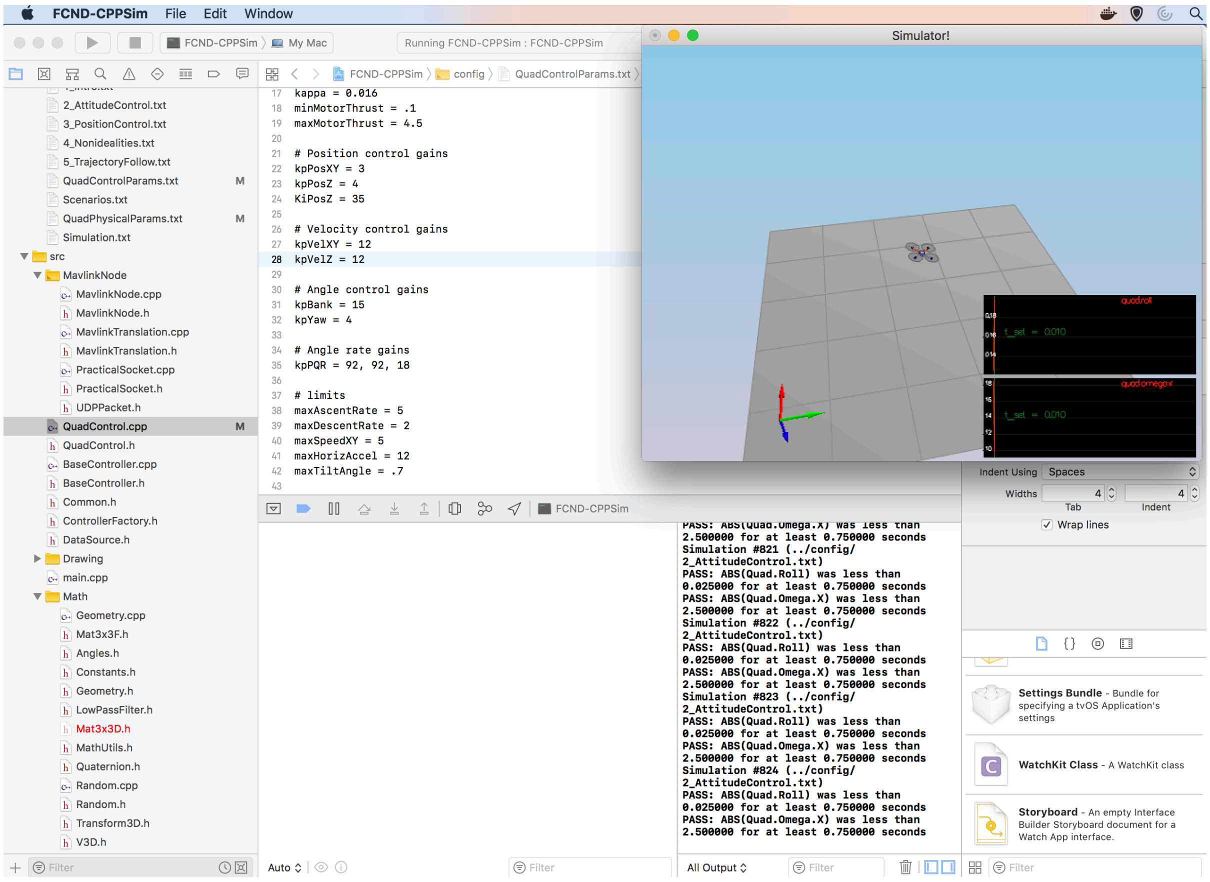
- Implement body rate control in C++ :-
  This is a single order P controller and controls roll,pitch and yaw in 3 axes.
  Modified **BodyRateControl** method to return desired moments for all 3 axes(X,Y,Z).

```
V3F MI;
MI.x = Ixx;
MI.y = Iyy;
MI.z = Izz;
momentCmd = MI * kpPQR * ( pqrCmd - pqr );
```

- Implement roll pitch control in C++:-

  This is again a single order P controller and works by receiving commanded acceleration in x and y axes. It basically converts rotation matrix from world frame to body frame. It returns a V3F(pqrCmd below) containing desired pitch and roll rates. Modified **RollPitchControl** method to implement the same.

```
float R11 = R(0, 0);
float R12 = R(0, 1);
float R21 = R(1, 0);
float R22 = R(1, 1);
float R33 = R(2, 2);

float c = collThrustCmd / mass;
V3F b = V3F(R(0, 2), R(1, 2), 0.f);
V3F b_c = V3F(accelCmd.x / -c, accelCmd.y / -c, 0.f);
b_c.constrain(-maxTiltAngle, maxTiltAngle);

V3F b_error = b_c - b;
V3F b_c_dot = kpBank * b_error;

pqrCmd.x = (R21 * b_c_dot.x - R11 * b_c_dot.y) / R33;
pqrCmd.y = (R22 * b_c_dot.x - R12 * b_c_dot.y) / R33;
pqrCmd.z = 0.f;
```



```
kappa = 0.016
minMotorThrust = .1
maxMotorThrust = 4.5

# Position control gains
kpPosXY = 3
kpPosZ = 4
KiPosZ = 35

# Velocity control gains
kpVelXY = 12
kpVelZ = 12

# Angle control gains
kpBank = 15
kpYaw = 4

# Angle rate gains
kpPQR = 92, 92, 18

# limits
maxAscentRate = 5
maxDescentRate = 2
maxSpeedXY = 5
maxHorizAccel = 12
maxTiltAngle = .7
```

```
PASS: ABS(Quad.Omega.X) was less than
2.500000 for at least 0.750000 seconds
Simulation #821 (../config/
2_AttitudeControl.txt)
PASS: ABS(Quad.Roll) was less than
0.025000 for at least 0.750000 seconds
PASS: ABS(Quad.Omega.X) was less than
2.500000 for at least 0.750000 seconds
Simulation #822 (../config/
2_AttitudeControl.txt)
PASS: ABS(Quad.Roll) was less than
0.025000 for at least 0.750000 seconds
PASS: ABS(Quad.Omega.X) was less than
2.500000 for at least 0.750000 seconds
Simulation #823 (../config/
2_AttitudeControl.txt)
PASS: ABS(Quad.Roll) was less than
0.025000 for at least 0.750000 seconds
PASS: ABS(Quad.Omega.X) was less than
2.500000 for at least 0.750000 seconds
Simulation #824 (../config/
2_AttitudeControl.txt)
PASS: ABS(Quad.Roll) was less than
0.025000 for at least 0.750000 seconds
PASS: ABS(Quad.Omega.X) was less than
2.500000 for at least 0.750000 seconds
```

FCND-CPPSim
File
Edit
Window
Running FCND-CPPSim : FCND-CPPSim
Simulator!
quad.roll
quad.omega.x
1_set = 0.010
Indent Using Spaces
Widths 4 4
Tab Indent
Wrap lines
Settings Bundle - Bundle for specifying a tvOS Application's settings
WatchKit Class - A WatchKit class
Storyboard - An empty Interface Builder Storyboard document for a Watch App interface.
All Output

Scenario 3:- Position/velocity and yaw angle control

- Implement lateral position control in C++:-

  This control returns a desired acceleration in world frame. Implemented **LateralPositionControl** method to return a V3F with desired horizontal acceleration.

```
const V3F error = posCmd - pos;
velCmd += kpPosXY * error;

velCmd.x = CONSTRAIN(velCmd.x, -maxSpeedXY, maxSpeedXY);
velCmd.y = CONSTRAIN(velCmd.y, -maxSpeedXY, maxSpeedXY);

const V3F error_dot = velCmd - vel;
accelCmd += kpVelXY * error_dot;

accelCmd.x = CONSTRAIN(accelCmd.x, -maxAccelXY, maxAccelXY);
accelCmd.y = CONSTRAIN(accelCmd.y, -maxAccelXY, maxAccelXY);
accelCmd.z = 0.0F;
```

- Implement altitude control in C++:-

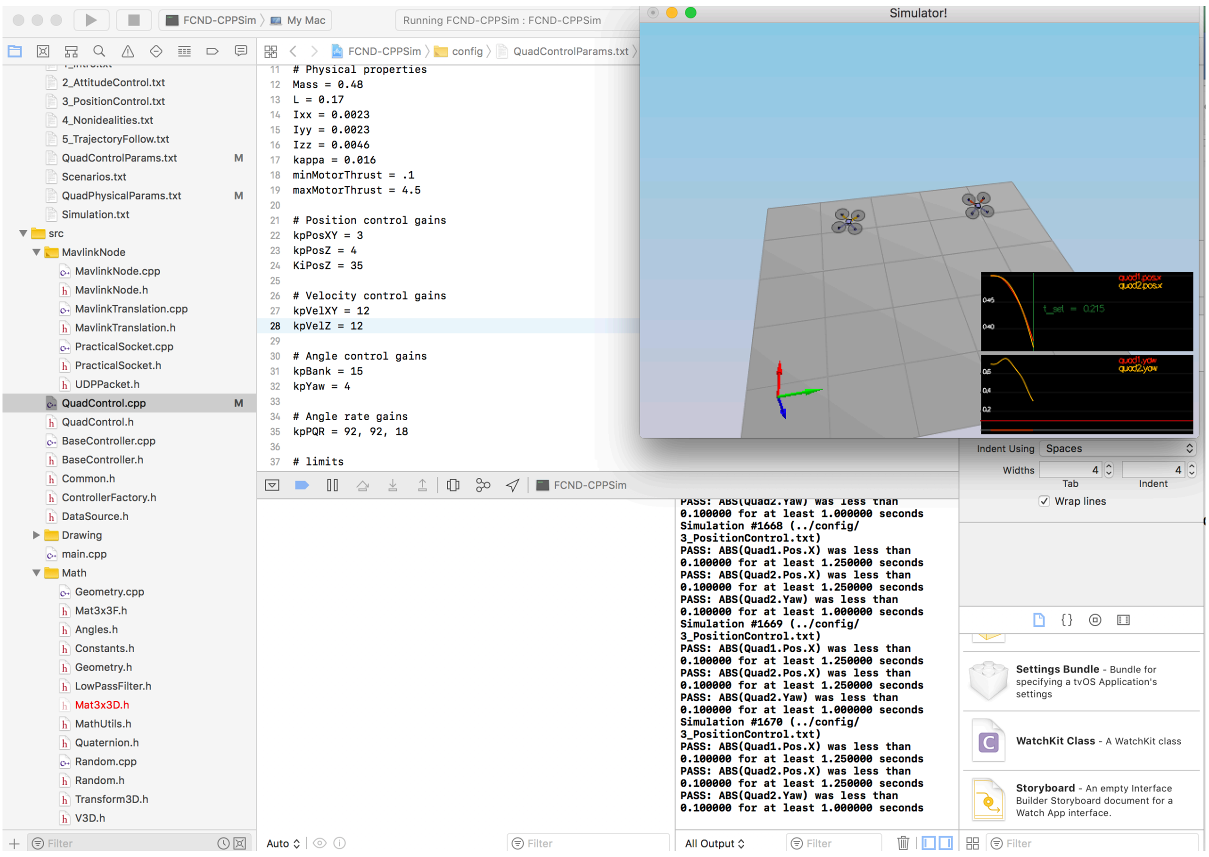
  This control returns thrust to control the acceleration in z direction.Implemented **AltitudeControl** method which returns overall thrust (in N).

  ```
  float z_target = posZCmd;
  float z_dot_target = velZCmd;
  float z_actual = posZ;
  float z_dot_actual = velZ;
  integratedAltitudeError = integratedAltitudeError + ((z_target - z_actual) * dt);
  float z_dot_dot_target = accelZCmd;

  float u_bar_1 = (kpPosZ * (z_target - z_actual)) + (kpVelZ * (z_dot_target - z_dot_actual)) +
  (KiPosZ * integratedAltitudeError) + z_dot_dot_target;
  float b = R(2,2);
  float c = (u_bar_1 - CONST_GRAVITY) / b;
  float c_const = CONSTRAIN(c, -maxAscentRate / dt, maxAscentRate / dt);
  thrust = - c_const * mass;
  ```

- Implement Yaw control in C++:-

  This control returns desired yaw rate. Implemented **AltitudeControl** method which returns taw rate (in rad/s)

  ```
  float yaw_error = yawCmd - yaw;
  yaw_error = fmodf(yaw_error, 2.*F_PI);
  yawRateCmd = kpYaw * yaw_error;
  ```

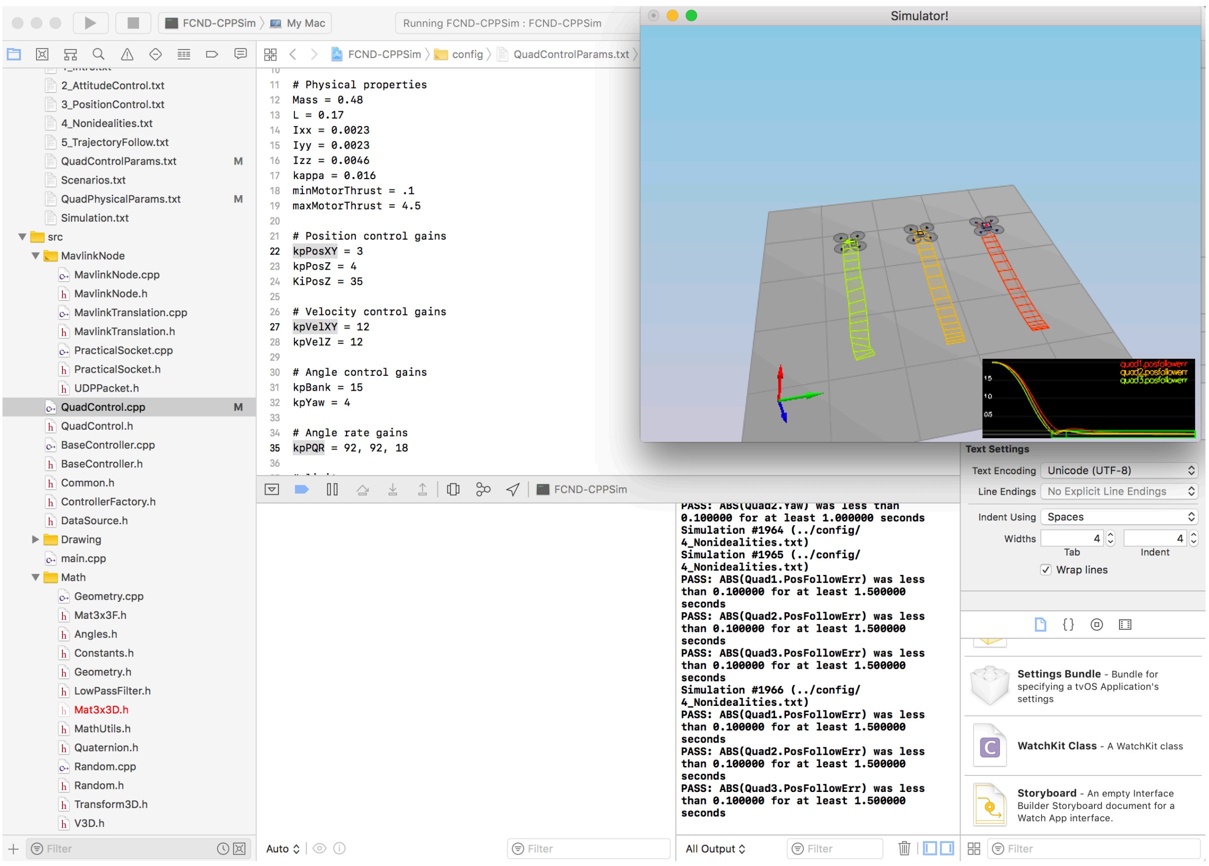


```
# Physical properties
Mass = 0.48
L = 0.17
Ixx = 0.0023
Iyy = 0.0023
Izz = 0.0046
kappa = 0.016
minMotorThrust = .1
maxMotorThrust = 4.5

# Position control gains
kpPosXY = 3
kpPosZ = 4
KiPosZ = 35

# Velocity control gains
kpVelXY = 12
kpVelZ = 12

# Angle control gains
kpBank = 15
kpYaw = 4

# Angle rate gains
kpPQR = 92, 92, 18

# limits
```

```
PASS: ABS(Quad2.Yaw) was less than
0.100000 for at least 1.000000 seconds
Simulation #1668 (../config/
3_PositionControl.txt)
PASS: ABS(Quad1.Pos.X) was less than
0.100000 for at least 1.250000 seconds
PASS: ABS(Quad2.Pos.X) was less than
0.100000 for at least 1.250000 seconds
PASS: ABS(Quad2.Yaw) was less than
0.100000 for at least 1.000000 seconds
Simulation #1669 (../config/
3_PositionControl.txt)
PASS: ABS(Quad1.Pos.X) was less than
0.100000 for at least 1.250000 seconds
PASS: ABS(Quad2.Pos.X) was less than
0.100000 for at least 1.250000 seconds
PASS: ABS(Quad2.Yaw) was less than
0.100000 for at least 1.000000 seconds
Simulation #1670 (../config/
3_PositionControl.txt)
PASS: ABS(Quad1.Pos.X) was less than
0.100000 for at least 1.250000 seconds
PASS: ABS(Quad2.Pos.X) was less than
0.100000 for at least 1.250000 seconds
PASS: ABS(Quad2.Yaw) was less than
0.100000 for at least 1.000000 seconds
```

Scenario 4:- Non idealities and robustness

Edited **AltitudeControl** method and tuned kpPosXY and kpVelXY parameters along with kpPQR to achieve desired result

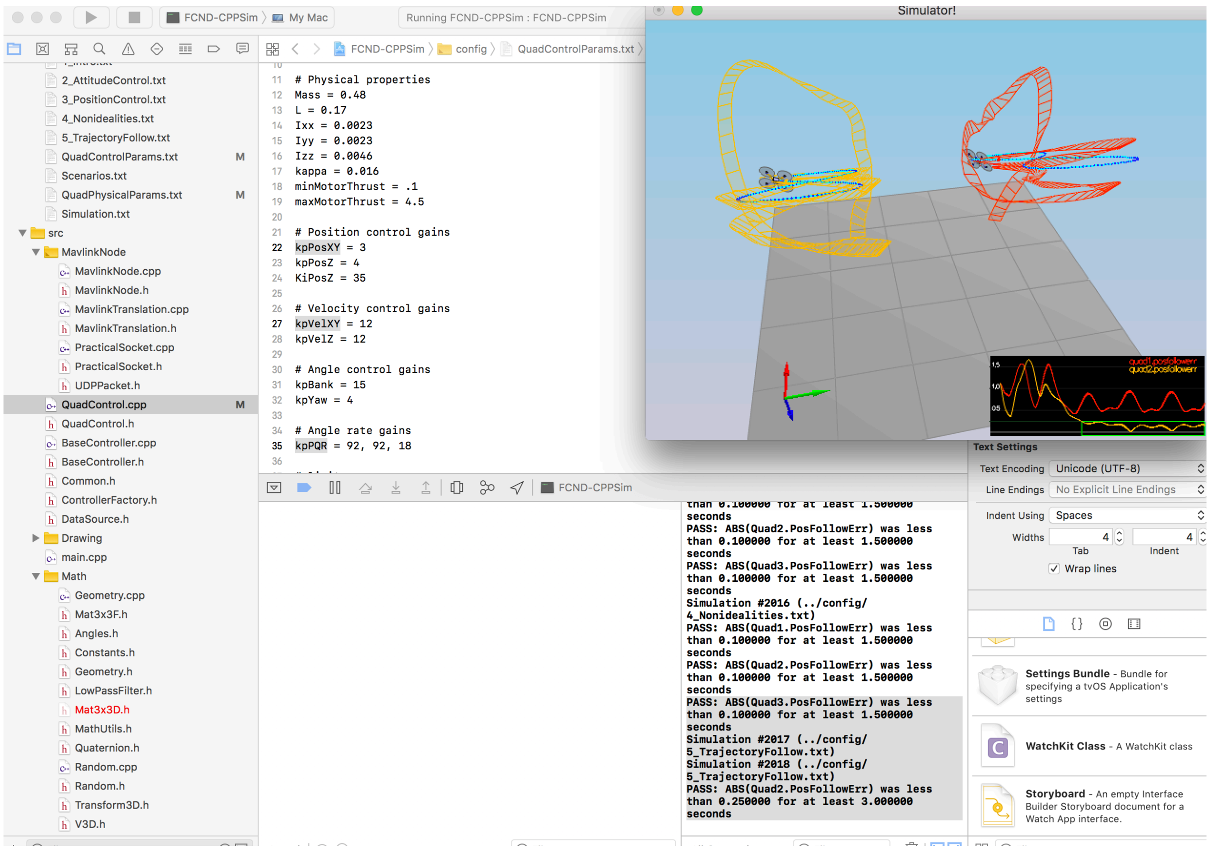


```
# Physical properties
Mass = 0.48
L = 0.17
Ixx = 0.0023
Iyy = 0.0023
Izz = 0.0046
kappa = 0.016
minMotorThrust = .1
maxMotorThrust = 4.5

# Position control gains
kpPosXY = 3
kpPosZ = 4
KiPosZ = 35

# Velocity control gains
kpVelXY = 12
kpVelZ = 12

# Angle control gains
kpBank = 15
kpYaw = 4

# Angle rate gains
kpPQR = 92, 92, 18
```

```
PASS: ABS(Quad2.Yaw) was less than
0.100000 for at least 1.000000 seconds
Simulation #1964 (../config/
4_Nonidealities.txt)
Simulation #1965 (../config/
4_Nonidealities.txt)
PASS: ABS(Quad1.PosFollowErr) was less
than 0.100000 for at least 1.500000
seconds
PASS: ABS(Quad2.PosFollowErr) was less
than 0.100000 for at least 1.500000
seconds
PASS: ABS(Quad3.PosFollowErr) was less
than 0.100000 for at least 1.500000
seconds
Simulation #1966 (../config/
4_Nonidealities.txt)
PASS: ABS(Quad1.PosFollowErr) was less
than 0.100000 for at least 1.500000
seconds
PASS: ABS(Quad2.PosFollowErr) was less
than 0.100000 for at least 1.500000
seconds
PASS: ABS(Quad3.PosFollowErr) was less
than 0.100000 for at least 1.500000
seconds
```

Scenario 5:- Testing it all together to make an 8 shape.

Able to make 8 shape with pass results(Though seems some more fine tuning is required in the provided parameters as it is not working as provided in sample example)



```
# Physical properties
Mass = 0.48
L = 0.17
Ixx = 0.0023
Iyy = 0.0023
Izz = 0.0046
kappa = 0.016
minMotorThrust = .1
maxMotorThrust = 4.5

# Position control gains
kpPosXY = 3
kpPosZ = 4
KiPosZ = 35

# Velocity control gains
kpVelXY = 12
kpVelZ = 12

# Angle control gains
kpBank = 15
kpYaw = 4

# Angle rate gains
kpPQR = 92, 92, 18
```

```
than 0.100000 for at least 1.500000
seconds
PASS: ABS(Quad2.PosFollowErr) was less
than 0.100000 for at least 1.500000
seconds
PASS: ABS(Quad3.PosFollowErr) was less
than 0.100000 for at least 1.500000
seconds
Simulation #2016 (../config/
4_Nonidealities.txt)
PASS: ABS(Quad1.PosFollowErr) was less
than 0.100000 for at least 1.500000
seconds
PASS: ABS(Quad2.PosFollowErr) was less
than 0.100000 for at least 1.500000
seconds
PASS: ABS(Quad3.PosFollowErr) was less
than 0.100000 for at least 1.500000
seconds
Simulation #2017 (../config/
5_TrajectoryFollow.txt)
Simulation #2018 (../config/
5_TrajectoryFollow.txt)
PASS: ABS(Quad2.PosFollowErr) was less
than 0.250000 for at least 3.000000
seconds
```

Could not try extra challenges and will do the same later.

Please find Project3.mov video in the attached folder for all scenarios video.