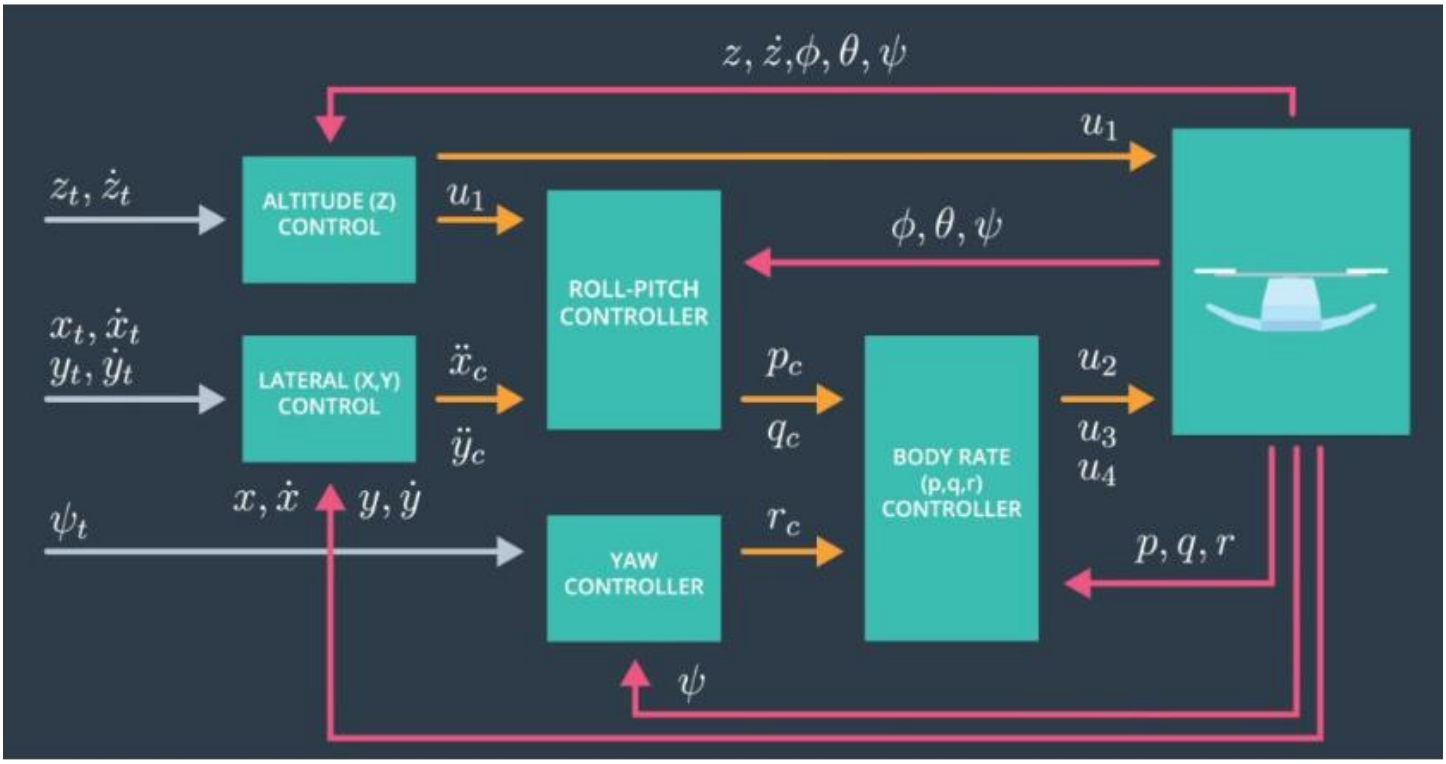
# Controller of Quadcopter




The above diagram is from Udacity lesson on control. This is all what I have implemented in by controller program.

## First of all, for I design the body rate controller and the code I wrote for is as follows.

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

then I wrote code for the roll pitch controller which is as follows

```
if ( collThrustCmd > 0 ) {
   float c = - collThrustCmd / mass;
   float b_x_cmd = CONSTRAIN(accelCmd.x / c, -maxTiltAngle, maxTiltAngle);
   float b_x_err = b_x_cmd - R(0,2);
   float b_x_p_term = kpBank * b_x_err;
```

```
  float b_y_cmd = CONSTRAIN(accelCmd.y / c, -maxTiltAngle, maxTiltAngle);
  float b_y_err = b_y_cmd - R(1,2);
  float b_y_p_term = kpBank * b_y_err;

  pqrCmd.x = (R(1,0) * b_x_p_term - R(0,0) * b_y_p_term) / R(2,2);
  pqrCmd.y = (R(1,1) * b_x_p_term - R(0,1) * b_y_p_term) / R(2,2);
 } else {
  pqrCmd.x = 0.0;
  pqrCmd.y = 0.0;
 }

 pqrCmd.z = 0;
```

**for the design of Altitude controller, I implemented following code**.

```
float z_err = posZCmd - posZ;
 float p_term = kpPosZ * z_err;

 float z_dot_err = velZCmd - velZ;
 integratedAltitudeError += z_err * dt;


 float d_term = kpVelZ * z_dot_err + velZ;
 float i_term = KiPosZ * integratedAltitudeError;
 float b_z = R(2,2);

 float u_1_bar = p_term + d_term + i_term + accelZCmd;

 float acc = ( u_1_bar - CONST_GRAVITY ) / b_z;

 thrust = - mass * CONSTRAIN(acc, - maxAscentRate / dt, maxAscentRate / dt);
```

## For the implementation of yaw control the code I wrote was

```
float yaw_cmd_2_pi = 0;
 if ( yawCmd > 0 ) {
  yaw_cmd_2_pi = fmodf(yawCmd, 2 * F_PI);
 } else {
  yaw_cmd_2_pi = -fmodf(-yawCmd, 2 * F_PI);
 }
 float err = yaw_cmd_2_pi - yaw;
 if ( err > F_PI ) {
  err -= 2 * F_PI;
 } if ( err < -F_PI ) {
  err += 2 * F_PI;
 }
 yawRateCmd = kpYaw * err;
```

## For the lateral position control the was applied with the following code

```
 V3F kpPos;
 kpPos.x = kpPosXY;
 kpPos.y = kpPosXY;
 kpPos.z = 0.f;

 V3F kpVel;
 kpVel.x = kpVelXY;
 kpVel.y = kpVelXY;
 kpVel.z = 0.f;

 V3F capVelCmd;
 if ( velCmd.mag() > maxSpeedXY ) {
  capVelCmd = velCmd.norm() * maxSpeedXY;
 } else {
```

```
  capVelCmd = velCmd;
 }

 accelCmd = kpPos * ( posCmd - pos ) + kpVel * ( capVelCmd - vel ) + accelCmd;

 if ( accelCmd.mag() > maxAccelXY ) {
  accelCmd = accelCmd.norm() * maxAccelXY;
 }
```

## finally, with the commanded thrust and moments from the above code the motor control code is as follows

```
float l = L / sqrtf(2.f);
 float t1 = momentCmd.x / l;
 float t2 = momentCmd.y / l;
 float t3 = - momentCmd.z / kappa;
 float t4 = collThrustCmd;

 cmd.desiredThrustsN[0] = (t1 + t2 + t3 + t4)/4.f;  // front left  - f1
 cmd.desiredThrustsN[1] = (-t1 + t2 - t3 + t4)/4.f; // front right - f2
 cmd.desiredThrustsN[2] = (t1 - t2 - t3 + t4)/4.f ; // rear left   - f4
 cmd.desiredThrustsN[3] = (-t1 - t2 + t3 + t4)/4.f; // rear right  - f3
```

After design the above controller for passing the different scenarios I tuned the parameter in the config file of all different scenarios. The tuned values are in the list of controls parameters.

**The controller passed all the scenarios in the simulation and results are as follows.**

Scenario 1

```
PASS: ABS(Quad.PosFollowErr) was less than 0.500000 for at least 0.800000 seconds
```

Scenario 2

```
PASS: ABS(Quad.Omega.X) was less than 2.500000 for at least 0.750000 seconds
```

Scenario 3

```
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
```

Scenario 4

```
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

```
PASS: ABS(Quad2.PosFollowErr) was less than 0.250000 for at least 3.000000 seconds
```