CONTROL PROJECT:

1. Generate motor thrust

```
////////////////////////////// BEGIN STUDENT CODE //////////////////////////
float l = L / sqrt(2.f);
float a = collThrustCmd / 4.f;
float b = momentCmd.x / (l * 4.f);
float c = momentCmd.y / (l * 4.f);
float d = -momentCmd.z / (kappa * 4.f);

cmd.desiredThrustsN[0] = a + b + c + d; // front left
cmd.desiredThrustsN[1] = a - b + c - d; // front right
cmd.desiredThrustsN[2] = a + b - c - d; // rear left
cmd.desiredThrustsN[3] = a - b - c + d; // rear right

////////////////////////////// END STUDENT CODE //////////////////////////
```

2. Body Rate control

```
////////////////////////////// BEGIN STUDENT CODE //////////////////////////
V3F Inertia;
Inertia.x = Ixx;
Inertia.y = Iyy;
Inertia.z = Izz;

momentCmd = Inertia * kpPQR * (pqrCmd - pqr);


////////////////////////////// END STUDENT CODE //////////////////////////
```

3. Pitch and Roll control

```
V3F pqrCmd;
Mat3x3F R = attitude.RotationMatrix_IwrtB();

////////////////////////////// BEGIN STUDENT CODE //////////////////////////
float c = -collThrustCmd / mass;

float b_x_c = CONSTRAIN(accelCmd.x / c, -maxTiltAngle, maxTiltAngle);
float b_x_dot = kpBank * (b_x_c - R(0, 2));
pqrCmd.x = (R(1, 0) * b_x_dot - R(0, 0) * b_x_dot / R(2, 2));

float b_y_c = CONSTRAIN(accelCmd.y / c, -maxTiltAngle, maxTiltAngle);
float b_y_dot = kpBank * (b_y_c - R(1, 2));
pqrCmd.y = (R(1, 1) * b_y_dot - R(0, 1) * b_y_dot / R(2, 2));

pqrCmd.z = 0;

////////////////////////////// END STUDENT CODE //////////////////////////
```

4. Altitude Control

```
Mat3x3F R = attitude.RotationMatrix_IwrtB();
float thrust = 0;

////////////////////////////// BEGIN STUDENT CODE ////////////////////////////
float z_err = posZCmd - posZ;
integratedAltitudeError += z_err * dt;
float z_dot_err = velZCmd - velZ;

float p_term = kpPosZ * z_err;
float i_term = KiPosZ * integratedAltitudeError;
float d_term = kpVelZ * z_dot_err + velZ;

float u_1_bar = p_term + i_term + d_term;

float b_z = R(2, 2);

float acceleration = (u_1_bar - 9.81) / b_z;
float cons_acceleration = CONSTRAIN(acceleration, -maxAscentRate / dt, maxDescentRate / dt);

thrust = -mass * cons_acceleration;
```

5. Lateral Position Control

```
accelCmdFF.z = 0;
velCmd.z = 0;
posCmd.z = pos.z;

// we initialize the returned desired acceleration to the feed-forward value.
// Make sure to _add_, not simply replace, the result of your controller
// to this variable
V3F accelCmd = accelCmdFF;

////////////////////////////// BEGIN STUDENT CODE ////////////////////////////
V3F p_term = posCmd - pos;
V3F d_term = velCmd - vel;

V3F accel = kpPosXY * p_term + kpVelXY * d_term + accelCmdFF;

accelCmd.constrain(-maxAccelXY, maxAccelXY);
accelCmd.z = 0;
```

6. Yaw Control

```
float yawRateCmd=0;
////////////////////////////// BEGIN STUDENT CODE ////////////////////////////
float yaw_err = yawCmd - yaw;
yaw_err = fmodf(yawCmd, 2.f * F_PI);
if (yaw_err > F_PI) {
    yaw_err -= 2.f * F_PI;
}
else if(yaw_err < -F_PI) {
    yaw_err += -2.f * F_PI;
}
yawRateCmd = kpYaw * yaw_err;
```