REPORT 21/04/2018

FRA501 Principles of Model-based Design in Robotics

อาจารย์ธนัชชา ชูพจน์เจริญ

Quadrotor Dynamic Modelling

การโมเดลระบบของอากาศยานสี่ใบพัด

Member

นายจุฬภัทร จิรชัย	57340500013
นายเจตนันท์ หอมจันทนากุล	57340500015
นายวุฒิภัทร โชคอนันตทรัพย์	57340500067

Insitude of Field Robotics, KMUTT

สารบัญ

เรื่อง หน้า

บทที่ 1
Introduction

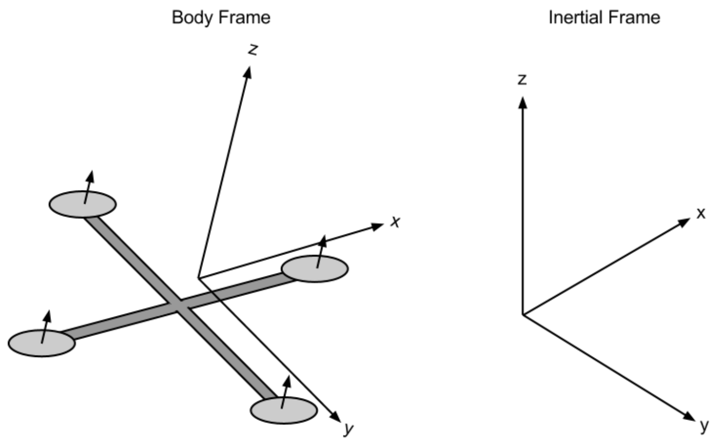
อากาศยานสี่ใบพัด (quadrotor) คือเฮลิคอปเตอร์หลายใบพัด (multi-helicopter) ที่มีลักษณะใบพัดเรียงตัวกันเป็นรูปสี่เหลี่ยมมุมฉาก หรือเป็นรูปทรงที่มีความสมมาตร เพื่อใช้ในการพยุงพาหนะและควบคุมทิศทางได้อย่างเป็นอิสระมากขึ้นเมื่อเทียบกับเฮลิคอปเตอร์ทั่วไป โดยในปัจจุบัน อากาศยานสี่ใบพัดได้ถูกนำมาประยุกต์ใช้งานในหลากหลายรูปแบบ ยกตัวอย่างเช่น ด้านเกษตรกรรม(ใช้ในการฉีดยาป้องกันศัตรูพืชหรือตรวจสอบผลิตผลเพื่อช่วยเกษตรกร) ด้านอุตสาหรรม(ใช้สำรวจรอยรั่วที่หลังคาของโรงงาน บ้านเรือน) ด้านการทหาร(ใช้ในการสำรวจพื้นที่ ตรวจดูที่ตั้งของศัตรู) ด้านกู้ภัยฉุกเฉิน(ใช้ในการหาพื้นที่ประสบภัย ไฟไหม้) ด้านการบันเทิง(ใช้แปรอักษรเป็นรูปภาพและเคลื่อนที่ในรูปแบบต่างๆ) เป็นต้น ซึ่งจะเห็นได้ว่าอากาศยานสี่ใบพัดนั้นสามารถนำไปใช้ได้ในหลากหลายทาง แต่ด้วยจำนวนของใบพัดที่มีมาก ทำให้ระบบมีกลไกและกระบวนการในการควบคุมที่มีความซับซ้อนมากขึ้น ทางคณะผู้จัดทำจึงสนใจที่จะทำโครงงานเกี่ยวกับอากาศยานสี่ใบพัด เพื่อศึกษาพลศาสตร์และออกแบบระบบควบคุม รวมไปถึงการจำลองการเคลื่อนที่ของอากาศยานโดยทำผ่านโปรแกรมจำลอง และนำความรู้ทั้งหมดไปใช้ในการพัฒนาการทำงานของอากาศยานสี่ใบพัดต่อไปในอนาคต

บทที่ 2
System Description

2.1 Quadroter mathematical model

2.1.1 Preliminar notions

โดรนเป็นอากาศยานชนิดหนึ่งที่มีมอเตอร์เป็นตัวขับเคลื่อนอยู่ 4 ตัว มีบอร์ดควบคุมและสั่งการอยู่ตรงกลางระหว่างมอเตอร์ทั้งสี่ ก่อนจะหาสมการคณิตศาสตร์ของโดรนนั้น จะต้องเริ่มจากรู้จักเฟรมอ้างอิงก่อน เฟรมอ้างอิงนี้จะช่วยบอกได้ว่าตัวโดรนนั้นอยู่ที่ตำแหน่งไหน เอียงอย่างไรอยู่ เฟรมอ้างอิงของโดรนนั้นใช้เพียง 2 เฟรมด้วยกัน คือเฟรมที่อยู่กับที่ไม่ขยับไปไหน และเฟรมที่เคลื่อนที่ไปมา เฟรมแรกคือเฟรมที่อยู่กับที่นั้นมีชื่อว่า “Inertial frame” เฟรมนี้เอาไว้สำหรับใช้ในการอ้างอิงพื้นโลก โดยที่ทิศทางของแรงโน้มถ่วง (gravity) จะมีทิศทางไปทางแกน -Z ของ “Inertial frame” ส่วนเฟรมอีกอันคือ “Body frame” เฟรมนี้เอาไว้ใช้สำหรับในการบอกตำแหน่งและการเอียงของตัวโดรน โดยการบอกนั้นจะบอกเทียบกับ “Inertial frame” การตั้งเฟรมนี้นั้นจะให้แกน +Z ชี้ไปตามทิศของมอเตอร์ทั้งสี่ ดังรูปที่ 2.1

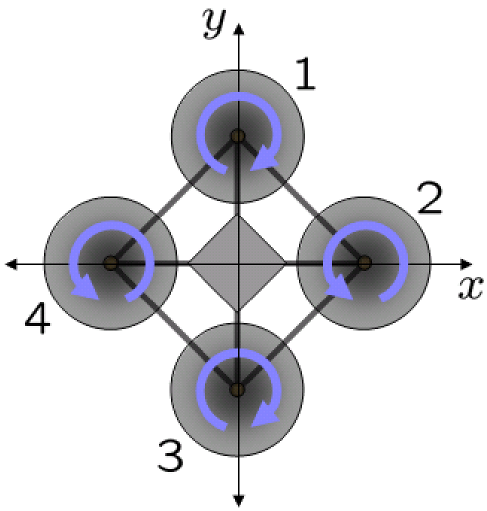


รูปที่ 2.1: เฟรมอ้างอิงของโดรน

การควบคุมตำแหน่งและท่าทางของโดรนนั้น เราสามารถควบคุมได้โดยการสั่งให้มอเตอร์ทั้งสี่ตัวมีความเร็วที่แตกต่างกัน เมื่อมอเตอร์หมุนด้วยความเร็วไม่เท่ากันจะทำให้เกิดแรงและโมเมนต์ขึ้นที่ตัวโดรน แรงยกนั้นจะเกิดขึ้นจากการหมุนมอเตอร์ทั้งหมด ส่วนการเอียง Pitch (หมุนรอบแกน Y ของ Body frame), Roll (หมุนรอบแกน X ของ Body frame) จะเกิดจากความแตกต่างระหว่างแรงยกของมอเตอร์ทั้งสี่ตัว แรงโน้มถ่วงของโลก และ Yaw (หมุนรอบแกน Z ของ Body frame) เกิดจากการที่มอเตอร์หมุนด้วยความเร็วที่ไม่สมดุลกัน โดรนจะไม่หมุน Yaw หากมีมอเตอร์หมุนไปในทิศทางตรงกันข้ามกัน ดังนั้นทำให้เราสามารถแบ่งใบพัดของโดรนออกเป็น 2 กลุ่ม แต่ละกลุ่มจะมีทิศทางการหมุนตรงข้ามกันและอยู่ฝั่งตรงกันข้ามกัน

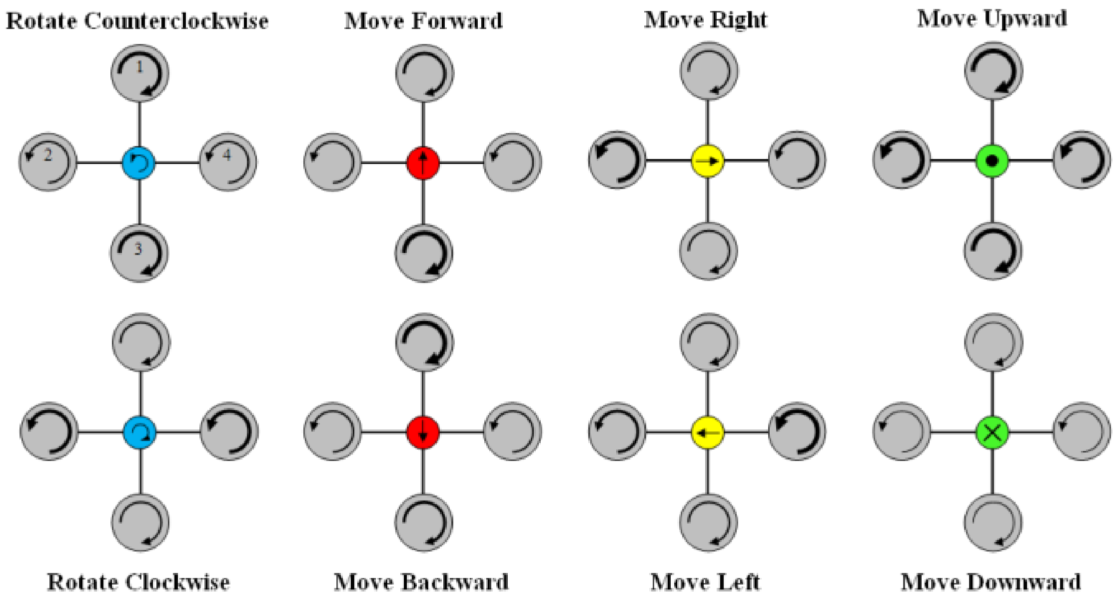
- ใบพัด หน้า และ หลัง (เลข 2 และเลข 4 ในรูปที่ 2.2) จะหมุนทวนเข็มนาฬิกา CCW
- ใบพัด ซ้าย และ ขวา (เลข 1 และเลข 3 ในรูปที่ 2.2) จะหมุนตามเข็มนาฬิกา CW

การเคลื่อนที่ในปริภูมิของโดรนนั้นเราสามารถแบ่งได้ออกเป็น 2 ส่วนคือ การเคลื่อนที่ตามแนวแกน และการเคลื่อนที่หมุนรอบแกน ในการอธิบายการเคลื่อนที่ของโดรนนั้น หากนับองศาอิสระจะได้ทั้งหมดเป็น 6 องศาอิสระ โดย 6 องศาอิสระนี้คือ การเคลื่อนที่ตามแนวแกน 3 แกน (X Y Z) และการเคลื่อนที่หมุนรอบแกน (Roll Pitch Yaw) การควบคุมการเคลื่อนที่ใน 6 องศาอิสระนั้นสามารถทำได้โดยปรับความเร็วการหมุนของมอเตอร์ให้มีความแตกต่างกัน การเคลื่อนที่ไปข้างหน้า ถอยหลัง ไปด้านข้าง ขึ้นลง หมุนรอบ Roll Pitch Yaw, ที่โดรนสามารถหมุนรอบแกน Yaw ได้นั้นเกิดจากทอร์ค



รูปที่ 2.2: ทิศทางการหมุนของแต่ละใบพัด

ของมอเตอร์ทั้งสี่ ผลรวมของทอร์คจะส่งผลต่อความเร็วในการหมุนรอบแกน Yaw หากมอเตอร์ทุกตัวหมุนด้วยความเร็วเท่ากัน ผลรวมทอร์คจะมีค่าเท่ากับศูนย์ทำให้โดรนไม่หมุน แต่ถ้ามอเตอร์หมุนด้วยความเร็วไม่เท่ากัน จะทำให้ผลรวมทอร์คมีค่าไม่เท่ากับศูนย์จะส่งผลให้โดรนเกิดการหมุนรอบแกน Yaw ได้ ถ้ามอเตอร์ทุกตัวหมุนด้วยความเร็วเพิ่มขึ้นหรือลดลงพร้อมกันจะทำให้โดรนเคลื่อนที่ขึ้นหรือลงตามแนวดิ่ง และด้วยการที่เรามี 4 Inputs แต่มี 6 Outputs นั้นทำให้การควบคุมโดรนเป็นแบบ Underactuated ซึ่งมีความซับซ้อนพอสมควร เพื่อที่จะคอนโทรลโดรนได้นั้น ให้สมมุติว่าตัวโดรนเป็นวัตถุชิ้นเดียว (Rigid body) โครงสร้างมีความสมมาตร (Symmetric) ความเร็วของแต่ละมอเตอร์ จะทำให้ใบพัดมีแรงยก เราสามารถบอกลักษณะการเคลื่อนที่ของโดรนได้ ดังรูปที่ 2.3



รูปที่ 2.3: ทิศทางการเคลื่อนที่ของโดรนเมื่อมอเตอร์หมุนด้วยความเร็วต่างๆ

2.1.2 Euler angles and Quaternions

Euler angles

มุมออยเลอร์เป็นมุม 3 มุมที่คิดโดย Leonhard Euler เพื่อที่จะเอาไว้ใช้อธิบายการเอียงของวัตถุในปริภูมิ ใช้ตัวแปรเพียงแค่ 3 ตัวเท่านั้น การบอกมุมออยเลอร์สามารถบอกได้หลายวิธี ในที่นี้เราใช้ ZYX Euler angles ในการอธิบายมุมเอียงของเฟรมอ้างอิงที่เราสนใจเทียบกับเฟรมอ้างอิงเฟรมอื่น มุมออยเลอร์เกิดจากการหมุนเฟรมรอบแกนสามแกน มาหมุนเรียงต่อกัน ต่อไปจะเป็นการรวมแมทริกการหมุนสามแกนเข้าด้วยกัน

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c(\phi) & -s(\phi) \\ 0 & s(\phi) & c(\phi) \end{bmatrix} \tag{2.1}$$

$$R_y(\theta) = \begin{bmatrix} c(\theta) & 0 & s(\theta) \\ 0 & 1 & 0 \\ -s(\theta) & 0 & c(\theta) \end{bmatrix} \tag{2.2}$$

$$R_z(\psi) = \begin{bmatrix} c(\psi) & -s(\psi) & 0 \\ s(\psi) & c(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

โดยที่ $c(\phi) = cos(\phi)$, $s(\phi) = sin(\phi)$, $c(\theta) = cos(\theta)$, $s(\theta) = sin(\theta)$, $c(\psi) = cos(\psi)$, $s(\psi) = sin(\psi)$ จากแมทริกการหมุนแสดงให้เห็นว่า “Inertial frame” และ “Body frame” มีความสัมพันธ์กันเป็นแมทริกการหมุนคือ $R_{zyx}(\phi, \theta, \psi)$

$$\begin{gathered} R_{zyx}(\phi,\theta,\psi) = R_z(\psi)R_y(\theta)R_x(\phi) \\ = \begin{bmatrix} c(\theta)c(\psi) & s(\phi)s(\theta)c(\psi) - c(\phi)s(\psi) & c(\phi)s(\theta)c(\psi) + s(\phi)s(\psi) \\ c(\theta)s(\psi) & s(\phi)s(\theta)c(\psi) + c(\phi)s(\psi) & c(\phi)s(\theta)c(\psi) - s(\phi)c(\psi) \\ -s(\theta) & s(\phi)c(\theta) & c(\phi)c(\theta) \end{bmatrix} \end{gathered} \tag{2.4}$$

Quaternions

ในบางครั้งหรือบางกรณี Euler angles อาจจะทำให้เกิด Gimbal lock ได้ ซึ่งจะทำให้การควบคุมเสียไป 1 องศาอิสระ เพื่อที่จะหลีกเลี่ยงเหตุการณ์นี้จึงได้นำ quaternions มาใช้ quaternions ใช้เพื่อที่จะบอกมุมเอียงของวัตถุใดๆ ได้เหมือนกับ Euler angles โดยที่เราสามารถแปลง Euler angles ให้เป็น quaternions ได้จากสมการที่ 2.5

$$\begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} = \begin{bmatrix} c(\frac{\phi}{2})c(\frac{\theta}{2})c(\frac{\psi}{2}) + s(\frac{\phi}{2})s(\frac{\theta}{2})s(\frac{\psi}{2}) \\ s(\frac{\phi}{2})c(\frac{\theta}{2})c(\frac{\psi}{2}) + c(\frac{\phi}{2})s(\frac{\theta}{2})s(\frac{\psi}{2}) \\ c(\frac{\phi}{2})s(\frac{\theta}{2})c(\frac{\psi}{2}) + s(\frac{\phi}{2})c(\frac{\theta}{2})s(\frac{\psi}{2}) \\ c(\frac{\phi}{2})c(\frac{\theta}{2})s(\frac{\psi}{2}) + s(\frac{\phi}{2})s(\frac{\theta}{2})c(\frac{\psi}{2}) \end{bmatrix} \tag{2.5}$$

และเราสามารถที่จะเขียนแมทริกการหมุนให้อยู่ในรูปของ quaternions ได้ดังสมการที่ 2.6

$$R_{zyx}(\phi,\theta,\psi) = \begin{bmatrix} q_1^2+q_2^2-q_3^2-q_4^2 & 2(q_2q_3-q_1q_4) & 2(q_1q_3+q_2q_4) \\ 2(q_2q_3+q_1q_4) & q_1^2-q_2^2+q_3^2-q_4^2 & 2(q_3q_4-q_1q_2) \\ 2(q_2q_4-q_1q_3) & 2(q_1q_2-q_3q_4) & q_1^2-q_2^2-q_3^2+q_4^2 \end{bmatrix} \tag{2.6}$$

2.1.3 Qaudroter mathematical model

ในส่วนนี้จะเป็นการอธิบายสมการการเคลื่อนที่ของโดรน โดยใช้สมการของ Newton และ Euler มาช่วยในการอธิบายพลวัต (Dynamics) ของโดรน เพื่อใช้ทำแบบจำลอง (Simulating) และควบคุม (Controlling) ท่าทางของโดรนด้วย เริ่มจากให้ $[x \quad y \quad z \quad \phi \quad \theta \quad \psi]^T$ เป็นเวกเตอร์ที่บอกตำแหน่งและมุม (linear/angular position) ของโดรนโดยเทียบจากเฟรมโลก (Inertial frame) และ $[u \quad v \quad w \quad p \quad q \quad r]^T$ เป็นเวกเตอร์ที่บอกความเร็วเชิงเส้นและความเร็วเชิงมุม (linear/angular velocity) ของโดนโดยเทียบจากเฟรมโดรน (Body frame) พลวัตของโดรนจะเปิดจากเฟรมอ้างอิงสองเฟรมนี้มีความสัมพันธ์กัน

$$\begin{aligned} \nu &= R\nu_B \\ \omega &= T\omega_B \end{aligned} \tag{2.7}$$

โดยที่ $\nu = [\dot{x} \quad \dot{y} \quad \dot{z}]^T, \omega = [\dot{\phi} \quad \dot{\theta} \quad \dot{\psi}]^T, \nu_B = [u \quad v \quad w]^T, \omega_B = [p \quad q \quad r]^T$ และ T เป็นแมทริกการแปลงมุมการหมุน (angular transformation)

$$T = \begin{bmatrix} 1 & s(\phi)t(\theta) & c(\phi)t(\theta) \\ 0 & c(\phi) & -s(\phi) \\ 0 & \dfrac{s(\phi)}{c(\theta)} & \dfrac{c(\phi)}{c(\theta)} \end{bmatrix} \tag{2.8}$$

โดยที่ $t(\theta) = tan(\theta)$ ดังนั้นเราจะได้สมการจลศาสตร์ (Kinematic model) ของโดรนเป็น

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ u \\ v \\ w \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \\ p \\ q \\ r \end{bmatrix} = \begin{bmatrix} w[s(\phi)s(\psi) + c(\phi)c(\psi)s(\theta)] - v[c(\phi)s(\psi) - c(\psi)s(\phi)s(\theta)] + u[c(\psi)c(\theta)] \\ v[c(\phi)c(\psi) + s(\phi)s(\psi)s(\theta)] - w[c(\psi)s(\phi) - c(\phi)s(\psi)s(\theta)] + u[c(\theta)s(\psi)] \\ w[c(\phi)c(\theta)] - u[s(\theta)] + v[c(\theta)s(\phi)] \\ rv - qw - gs(\theta) \\ pw - ru + gc(\theta)s(\theta) \\ qu - pv + gc(\theta)s(\theta) - F/m \\ p + r[c(\phi)t(\theta)] + q[s(\phi)t(\theta)] \\ q[c(\phi)] - r[s(\phi)] \\ r\dfrac{c(\phi)}{c(\theta)} + q\dfrac{s(\phi)}{c(\theta)} \\ \frac{I_y - I_z}{I_x}qr + \frac{\tau_\phi}{I_x} \\ \frac{I_z - I_x}{I_y}pr + \frac{\tau_\theta}{I_y} \\ \frac{I_x - I_y}{I_z}pq + \frac{\tau_\psi}{I_z} \end{bmatrix} \tag{2.9}$$

จากกฎของนิวตันระบุความสัมพันธ์ของแรงรวมที่กระทำต่อโดรนดังเมทริกซ์ต่อไปนี้

$$m(\omega_B \wedge \nu_B + \dot{\nu_B}) = \mathsf{f}_B \tag{2.10}$$

โดย m คือน้ำหนักของโดรน , $\wedge$ คือ cross product และ $\mathsf{f}_B = \begin{bmatrix} f_x & f_y & f_z \end{bmatrix}^T \in \mathsf{R}^3$ คือแรงรวม จากสมการออยเลอร์ ให้แรงบิดรวมที่ใช้กับโดรน เป็นไปดังนี้

$$I.\dot{\omega}_B + \omega_B \wedge (I.\omega_B) = \mathsf{m}_B \tag{2.11}$$

โดย $\mathsf{m}_B = \begin{bmatrix} m_x & m_y & m_z \end{bmatrix}^T \in \mathsf{R}^3$ เป็นแรงบิดรวม และ I เป็นเมทริกซ์ของความเฉื่อย :

$$I = \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix} \wedge \mathsf{R}^{3\times3} \tag{2.12}$$

ดังนั้น จะได้ dynamic model ของโดรนโดยอ้างอิง body frame ดังนี้

$$\begin{bmatrix} f_x \\ f_y \\ f_z \\ m_x \\ m_y \\ m_z \end{bmatrix} = \begin{bmatrix} m(\dot{u} + qw - rv) \\ m(\dot{v} - pw + ru) \\ m(\dot{w} + pv - qu) \\ \dot{p}I_x - qrI_y + qrI_z \\ \dot{q}I_y + prI_x - prI_z \\ \dot{r}I_z - pqI_x + pqI_y \end{bmatrix} \tag{2.13}$$

2.1.4 Motor model

ในแต่ละใบพัดของ quadrotor จะให้แรงที่มีทิศทางตรงข้ามกับแกน Z ของ body frame แรงที่เกิดจากการหมุนจะทำให้เกิดแรงบิดรอบแนวแกน roll pitch yaw โดยแรงและแรงบิดจะขึ้นอยู่กับความเร็วของใบพัดและค่าคงที่ค่าหนึ่ง ซึ่งสามารถแสดงความสัมพันธ์ของแรงได้ดังสมการที่ 2.14 และความสัมพันธ์ของแรงบิดได้ดังสมการที่ 2.15

$$F_i = k_f \Omega_i^2 \tag{2.14}$$

$$T_i = -k_t sgn(i) \Omega_i^2 \tag{2.15}$$

โดยที่ตัวห้อย i จะแสดงถึงลำดับของใบพัดแต่ละตัว Ω_i แสดงถึงความเร็วการหมุนของใบพัด i k_f และ k_t เป็นค่าคงที่การคูณของแรงและแรงบิด sgn เป็นฟังก์ชันที่ใช้บอกทิศทางการหมุนของใบพัดที่มีความแตกต่างกันตามการติดตั้ง แรงทั้งหมดที่เกิดบน quadrotor สามารถที่จะเขียนสมการให้อยู่ในรูปของความเร็วการหมุนของใบพัดคูณกับแมทริกการแปลง M ดังสมการที่ 2.16

$$\begin{bmatrix} F \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = M \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix} \tag{2.16}$$

โดยที่ แมทริกการแปลง M มีการกำหนดไว้ดังสมการที่ 2.17

$$M = \begin{bmatrix} k_f & k_f & k_f & k_f \\ \frac{k_f d}{\sqrt{2}} & -\frac{k_f d}{\sqrt{2}} & -\frac{k_f d}{\sqrt{2}} & \frac{k_f d}{\sqrt{2}} \\ \frac{k_f d}{\sqrt{2}} & \frac{k_f d}{\sqrt{2}} & -\frac{k_f d}{\sqrt{2}} & -\frac{k_f d}{\sqrt{2}} \\ k_t & -k_t & k_t & -k_t \end{bmatrix} \tag{2.17}$$

โดยที่ Ω_i^2 เป็นความเร็วการหมุนของใบพัดกำลังสอง และ d คือระยะทางจากจุดศูนย์กลางมวลของ quadrotor ไปถึงจุดกึ่งกลางของใบพัด

2.2 Linear Quadratic Regulator

ในส่วนนี้จะกล่าวถึงตัวควบคุม LQR ซึ่งเป็นส่วนที่ใช้ในการควบคุม quadrotor โดยจะแบ่งออกเป็นสองส่วนควบคุมตำแหน่งและทิศทางการหมุนสามารถอ้างอิงและควบคุมได้โดยตรง ส่วน Roll และ Pitch ไม่สามารถควบคุมได้โดยตรงแต่จะควบคุมผ่านตำแหน่ง x และ y

ตัวควบคุม LQR ถูกพัฒนามาเพื่อใช้กับระบบที่มีการควบคุม inputs และ output เพื่อที่จะหาตัวควบคุม state feedback ที่มีประสิทธิ์ภาพมากที่สุด

สมมุติให้ Linear state spcae model ของ quadrotor เป็นดังสมการที่ 2.18

$$\dot{x} = Ax + Bu \tag{2.18}$$

โดยที่ A เป็น state matrix, B เป็น input matrix, x เป็น state vector ที่มีจำนวนตัวเท่ากับ n ตัว, และ u เป็น input vector ที่มีจำนวนตัวเท่ากับ m ตัว โดยทั้ง A, B จะต้องตรวจสอบก่อนว่าระบบเราสามารถที่จะควบคุมได้หรือไม่

ตัวควบคุม LQR สามารถออกแบบให้มีประสิทธิ์ภาพสูงสุดได้โดยที่ยังควบคุมระบบได้ เป้าหมายหลักก็คือการหาค่า feedback gain K, ผลลัพธ์การควบคุมที่ดีที่สุดโดยการ minimizes cost function ตามสมการที่ 2.19

$$J = \int_0^\infty (x^T Q x + u^T R u) dt \tag{2.19}$$

โดยที่ Q และ R เป็นนำหนักของ input และ state ถ้าหาก Q มีค่ามากหมายความว่าความถูกต้องของ state มีความสำคัญมาก และควรจะมีค่าความผิดพลาดน้อยเพื่อที่จะทำให้ cost ของ J น้อย และ R ก็คล้ายๆกันหากมีค่ามากหมายความว่าการใช้ input มีความสำคัญมาก

ผลลัพธ์ของ control input จาก feedback gain ดังสมการที่ 2.20

$$u = -Kx \tag{2.20}$$

โดยที่ K เป็น feedback gain และสามารถหาได้จากสมการที่ 2.21

$$K = R^{-1} B^T P \tag{2.21}$$

โดยที่ P เป็นสิ่งที่เราต้องหาเพื่อใช้ในการหา K จากการแก้สมการของ Riccati ดังสมการที่ 2.22

$$A^T P + PA + Q - PBR^{-1} B^T P = 0 \tag{2.22}$$

2.3 Hector Quadrotor ROS Stack

Hector Quadrotor เป็น ROS stacks (stack เป็นการรวบรวม packages ที่มีหลากหลายฟังก์ชันเอาไว้ด้วยกัน) โดยภายในประกอบไปด้วย ROS packages ดังต่อไปนี้

1) hector_quadrotor_description
2) hector_quadrotor_gazebo
3) hector_quadrotor_teleop
4) hector_quadrotor_gazebo_plugin

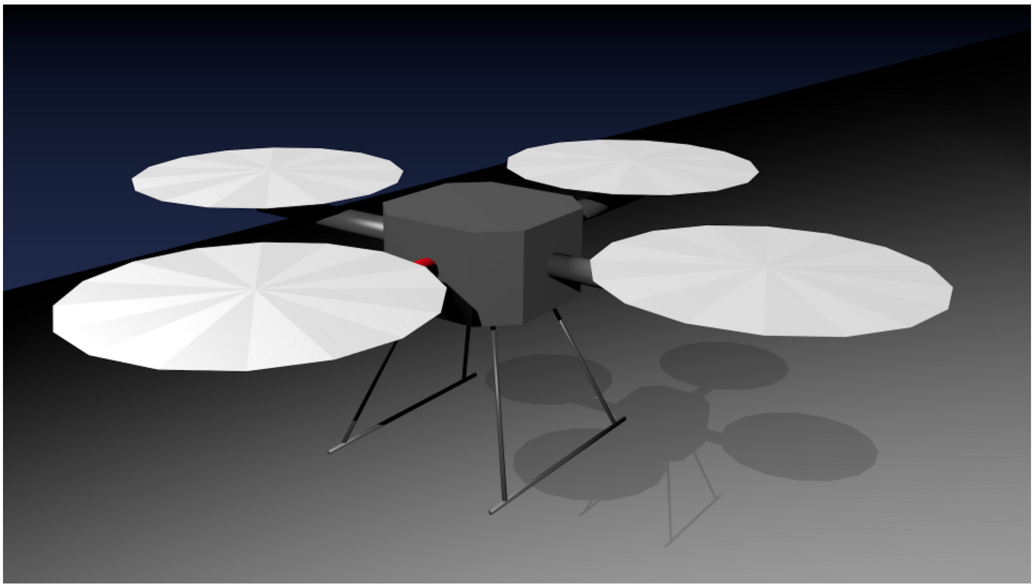
2.3.1 hector_quadrotor_description

package นี้ให้ URDF model ของ quadrotor มี visual geometry เป็นไฟล์ COLLADA และ collision geometry เป็นไฟล์ STL โดยจะเป็นส่วนเสริมที่ทำให้เราสามารถที่จะใช้ model ในโปรแกรม Gazebo ได้[1]

quadroter_base.urdf.xacro เป็นไฟล์ xacro ของโมเดลขั้นพื้นฐานของ quadroter

quadrotor.urdf.xacro เป็นไฟล์ xacro สำหรับการแสดงโมเดลขั้นพื้นฐานโดยไม่มีเซนเซอร์เพิ่มเติม

quadrotor_hokuyo_utm30lx.urdf.xacro เป็นไฟล์ xacro สำหรับการแสดงโมเดลขั้นพื้นฐานโดยเพิ่มเติมเซนเซอร์ Hokuyo เข้าไปสำหรับแสกนสิ่งแวดล้อม



รูปที่ 2.4: ภาพ quadrotor ที่ render ด้วย Blender

2.3.2 hector_quadrotor_gazebo

package นี้ให้ quadroter model ที่มาจากไฟล์ urdf และมาแสดงผลในโปรแกรม Gazebo โดย features[2] ที่มีคือ

1) Colored COLLADA quadrotor mesh model (.dae)
2) URDF description
3) Publishing of ground truth pose and simulated imu data
4) Spawnable in Gazebo
5) Controller สำหรับใช้ใน Gazebo โดยสามารถควบคุมผ่าน geometry_msgs/Twist message ใน topic 'cmd_vel'

[1] http://wiki.ros.org/hector_quadrotor_description
[2] http://wiki.ros.org/hector_quadrotor_gazebo

2.3.3 hector_quadrotor_teleop

package นี้จะทำให้เราสามารถที่จะควบคุม quadrotor ผ่าน gamepad ได้ โดยจะมี node ที่ publish geometry_msgs/Twist message ไปที่ topic ’cmd_vel’ ปัจจุบันมี gamepad 2 ชนิดที่นำมาใช้ได้ก็คือ

1) logitech_gamepad.launch สำหรับ Logitech gamepads
2) xbox_controller.launch สำหรับ Xbox controller gamepads

แต่หากต้องการใช้งานนอกเหนือจากนี้ สามารถที่จะสร้าง node ขึ้นมาใหม่ได้

2.3.4 hector_quadrotor_gazebo_plugin

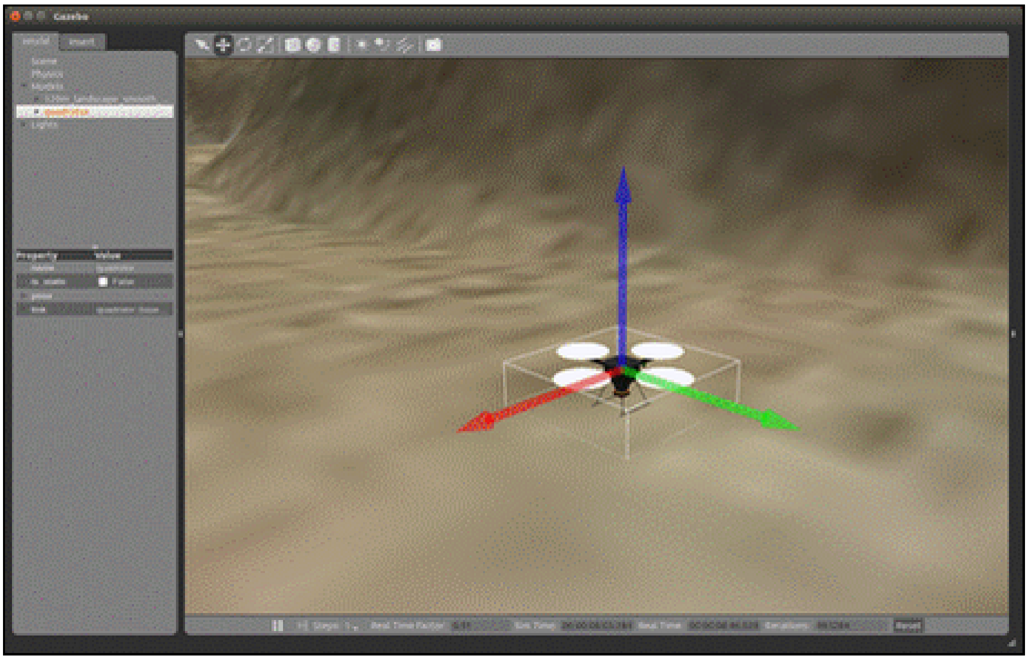
package นี้เป็นส่วนเสริมของ Gazebo ที่จะทำให้มีเซนเซอร์ต่างๆเข้ามาในระบบได้

Barometer Plugin เป็นส่วนเสริมที่จำลองเซนเซอร์ที่ใช้วัดความสูงของ quadrotor โดยมีพื้นฐานมาจากการวัด barometric pressure

Simple Controller Plugin package นี้เป็นส่วนเสริมของ Gazebo ที่จะทำให้เราสามารถควบคุม Velocities และ Yaw rate ได้โดยการคำนวณ แรงและแรงบิด โดยในปัจจุบันยังไม่สามารถที่จะควบคุมตำแหน่ง ทิศทางการหันหัว และความสูงได้ แต่สามารถที่จะเขียนทับเข้าไปได้

Quadrotor Propulsion Plugin / Quadrotor Aerodynamics Plugin package นี้เป็นส่วนเสริมของ Gazebo ที่จะทำให้เราสามารถควบคุมการหมุนของใบพัด quadroter ได้ โดยการควบคุม voltages ของมอเตอร์ และยังจำลอง wind vector ได้อีกด้วย

Matlab Compilation ในการเขียนสมการต่างๆของระบบ quadroter มักจะนิยมใช้ Matlab script และคอมไพล์เป็น C libraries โดยใช้ Matlab Coder ส่วนเสริมตัวนี้จะช่วยทำให้สามารถเชื่อมต่อกับ Matlab ได้



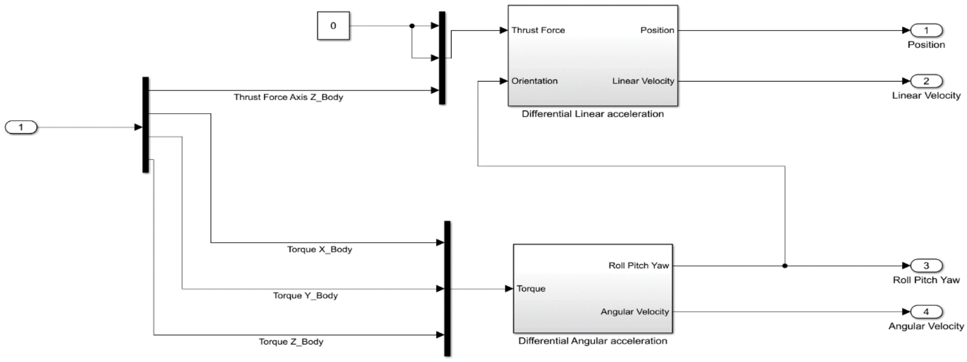
รูปที่ 2.5: ภาพ quadrotor ในโปรแกรม Gazebo

บทที่ 3
Simulation

3.1 Quadrotor simulation model

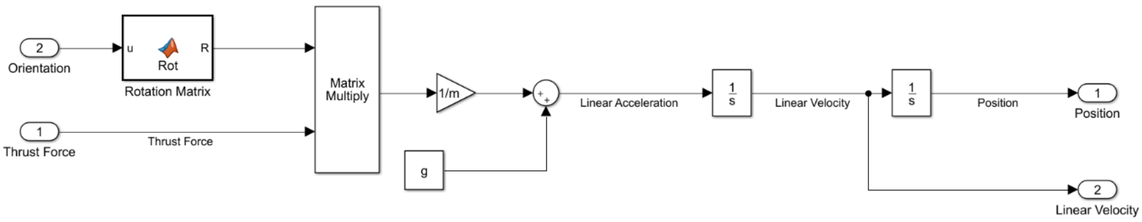
เราได้ใช้โปรแกรม Simulink จำลองการทำงานของ quadrotor โดยจะแบ่งออกเป็นสองส่วน คือ

1) อัตราการเปลี่ยนแปลงความเร่งเชิงเส้นโดยถูกเขียนขึ้นจากทฤษฎีที่กล่าวไปในบทที่ 2
2) อัตราการเปลี่ยนแปลงความเร่งเชิงมุมโดยถูกเขียนขึ้นจากทฤษฎีที่กล่าวไปในบทที่ 2



รูปที่ 3.1: แบบจำลอง quadrotor ในโปรแกรม simulink

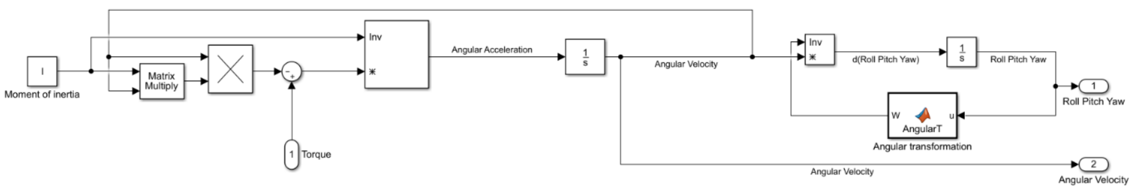
3.1.1 อัตราการเปลี่ยนแปลงความเร่งเชิงเส้น



รูปที่ 3.2: อัตราการเปลี่ยนแปลงความเร่งเชิงเส้น

Input เป็นแรงขับที่กระทำกับ quadrotor และทิศทางการหมุนของ quadrotor
Output เป็นตำแหน่งและความเร็วเชิงเส้นของ quadrotor

3.1.2 อัตราการเปลี่ยนแปลงความเร่งเชิงมุม



รูปที่ 3.3: อัตราการเปลี่ยนแปลงความเร่งเชิงมุม

Input เป็นแรงบิดที่กระทำกับ quadrotor
Output เป็นทิศทางการหมุนของ quadrotor และความเร็วเชิงมุมของ quadrotor

3.1.3 ผลการทดลอง

กำหนดค่าเริ่มต้นต่างๆของ quadrotor ที่ใช้กับการทดลองดังนี้

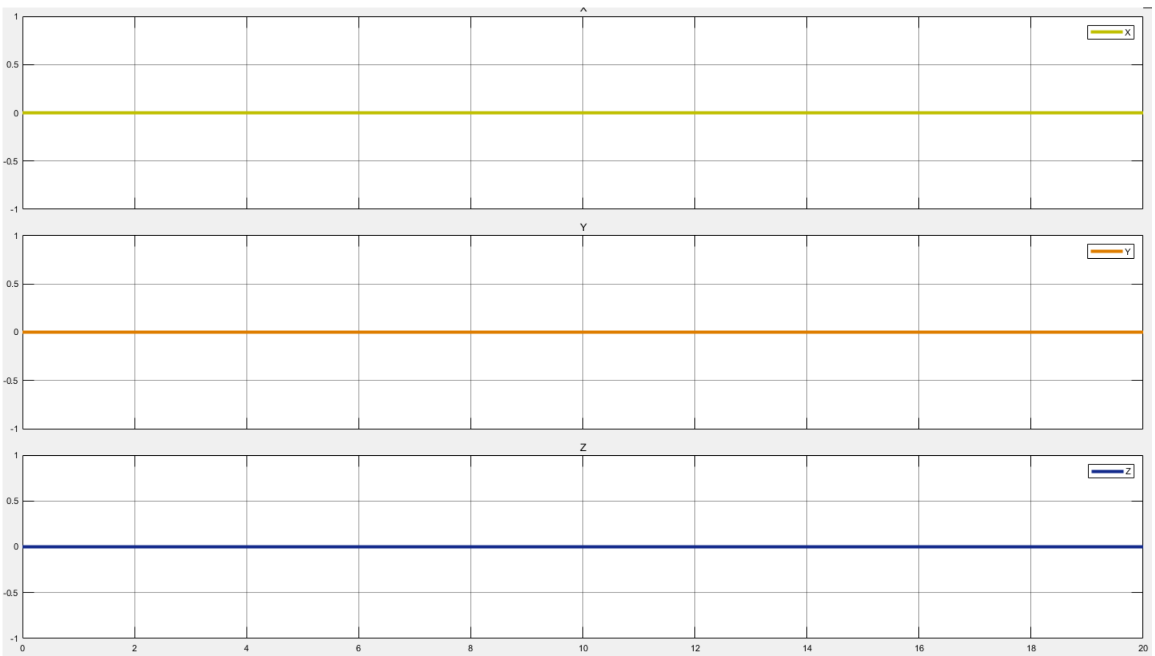
ตัวแปร	ค่า	หน่วย
I_x	0.01152	kgm^2
I_y	0.01152	kgm^2
I_z	0.0218	kgm^2
g	9.80665	m/s^2
m	1.477	kg
L	0.26	m

3.1.3.1 Hover quadrotor

ทดลอง Simulation โดยการใส่แรงขับที่กระทำให้ quadrotor สามารถลอยตัวอยู่ได้โดยไม่ร่วงลงตามแรงโน้มถ่วงของโลก

$$\begin{gathered} F = ma \\ F = 1.477 \times 9.80665 \end{gathered} \tag{3.1}$$

Thrust force Z	14.48442205	Nm
Torque X	0	Nm
Torque Y	0	Nm
Torque Z	0	Nm



รูปที่ 3.4: กราฟแสดงตำแหน่งของ quadrotor

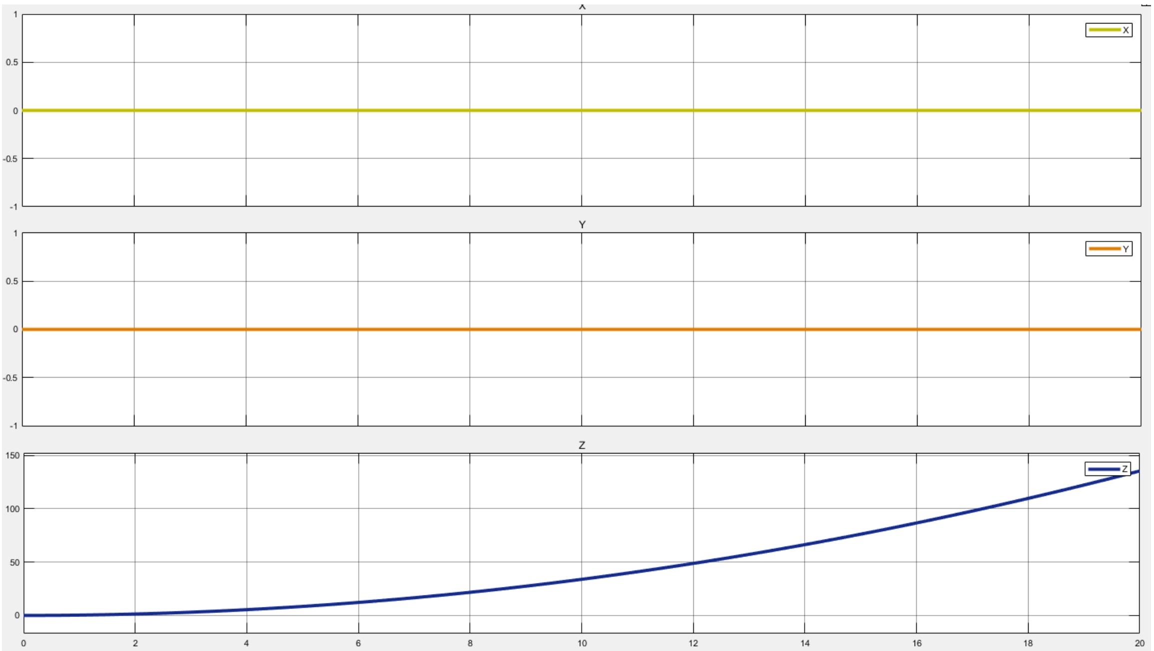
จากผลการทดลอง quadrotor สามารถลอยอยู่ได้ที่ตำแหน่ง $x = 0$, $y = 0$ และ $z = 0$ โดยไม่ร่วงลงมาตามแรงโน้มถ่วงของโลก

3.1.3.2 Climb up quadrotor

ทดลอง Simulation โดยการใส่แรงขับที่กระทำให้ quadrotor สามารถลอยตัวขึ้นไปในอากาศได้

$$F > ma$$
$$F > 1.477 \times 9.80665 \quad (3.2)$$

Thrust force Z	15.48442205	Nm
Torque X	0	Nm
Torque Y	0	Nm
Torque Z	0	Nm



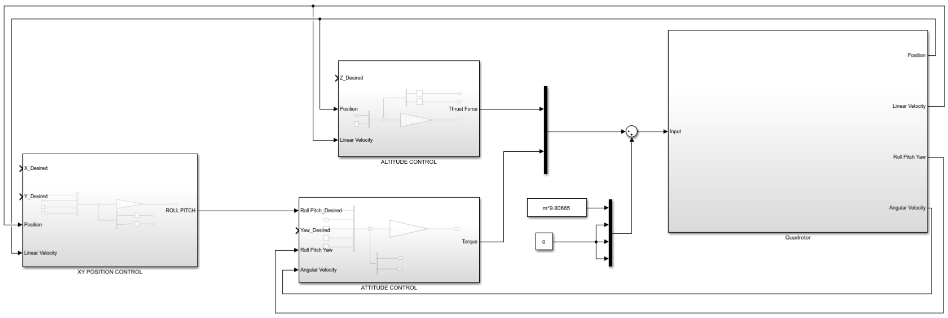
รูปที่ 3.5: กราฟแสดงตำแหน่งของ quadrotor

จากผลการทดลอง quadrotor สามารถเคลื่อนที่ลอยขึ้นไปในทิศทางตามแนวแกน Z ได้

3.2 Simulation กับตัวควบคุม Linear Quadratic Regulator

ในส่วนของการทดลองตัวควบคุมจะแบ่งออกเป็นสามส่วนคือ

1) Altitude control
2) Attitude control
3) X Y Position control



รูปที่ 3.6: แบบจำลองตัวควบคุม LQR ของ quadrotor ในโปรแกรม simulink

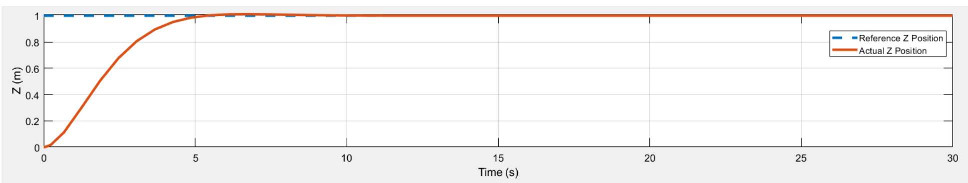
3.2.1 Altitude control

มี state และ input คือ

$$\vec{x} = \begin{bmatrix} z \\ v_z \end{bmatrix}, u = F_z \tag{3.3}$$

การทดลองครั้งที่ 1 กำหนดค่าน้ำหนักเมทริก Q กับ R ให้กับ Altitude control ดังนี้

$$\begin{gathered} Q_{Altitude} = diag(\begin{bmatrix} 1 & 1 \end{bmatrix}) \\ R_{Altitude} = 1 \\ K_{Altitude} = \begin{bmatrix} 1 & 1.9885 \end{bmatrix} \end{gathered} \tag{3.4}$$



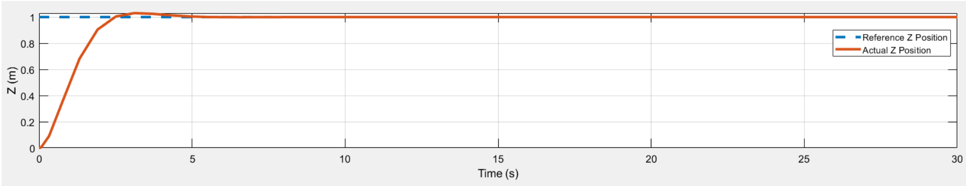
รูปที่ 3.7: ผลการทดสอบ Altitude control ครั้งที่ 1

จากกราฟจะเห็นได้ว่าตำแหน่งตามแนวแกน Z ของ quadrotor สามารถลู่เข้าตำแหน่งที่ต้องการได้และใช้เวลาไปประมาณ 5 วินาที

การทดลองครั้งที่ 2 ต้องการให้ตำแหน่งตามแนวแกน Z ของ quadrotor ลู่เข้าสู่ตำแหน่งที่ต้องการได้เร็วขึ้น จึงได้กำหนดค่าน้ำหนักเมทริก Q กับ R ขึ้นใหม่ดังนี้

$$\begin{gathered} Q_{Altitude} = diag(\begin{bmatrix} 10 & 1 \end{bmatrix}) \\ R_{Altitude} = 1 \\ K_{Altitude} = \begin{bmatrix} 3.1623 & 3.2158 \end{bmatrix} \end{gathered} \tag{3.5}$$

จากกราฟจะเห็นได้ว่าตำแหน่งตามแนวแกน Z ของ quadrotor สามารถลู่เข้า ณ ตำแหน่งที่ 1 เมตรเร็วขึ้น แต่มีการเกิด Overshoot ขึ้นเล็กน้อย



รูปที่ 3.8: ผลการทดสอบ Altitude control ครั้งที่ 2

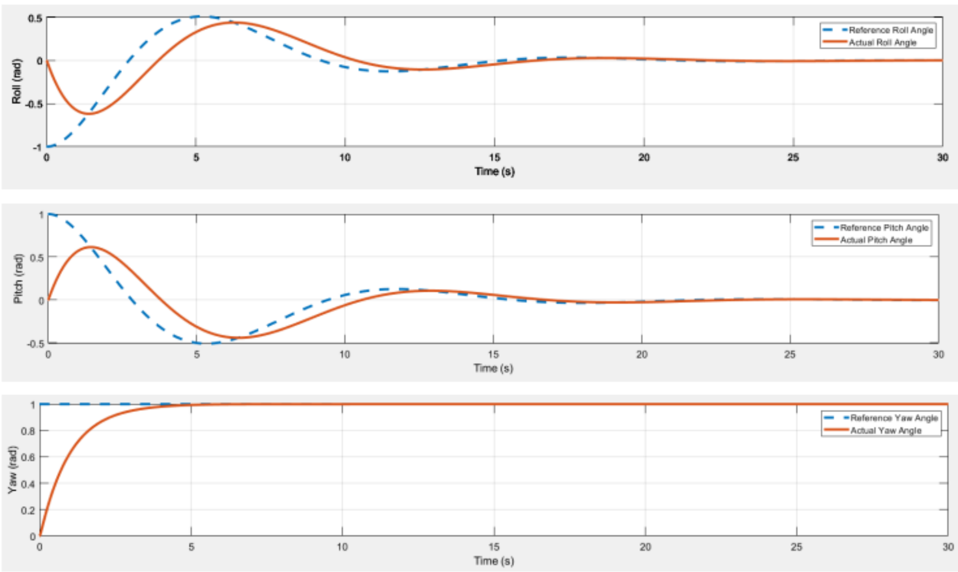
3.2.2 Attitude control

มี state และ input คือ

$$\vec{x} = \begin{bmatrix} \phi \\ \theta \\ \psi \\ p \\ q \\ r \end{bmatrix}, \vec{u} = \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} \tag{3.6}$$

การทดลองครั้งที่ 1 กำหนดค่าน้ำหนักเมทริก Q กับ R ให้กับ Attitude control ดังนี้

$$\begin{gathered} Q_{Attitude} = diag(\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}) \\ R_{Attitude} = diag(\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}) \\ K_{Attitude} = \begin{bmatrix} 1 & 0 & 0 & 1.0115 & 0 & 0 \\ 0 & 1 & 4.31e^{-17} & 0 & 1.0115 & 4.16e^{-17} \\ 0 & 2.42e^{-15} & 1 & 0 & 2.19e^{-17} & 1.0216 \end{bmatrix} \end{gathered} \tag{3.7}$$

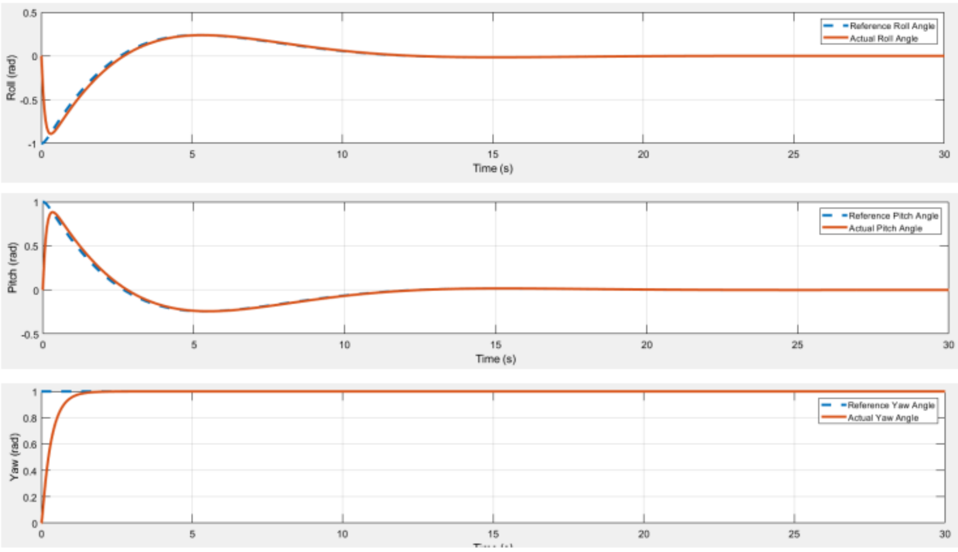


รูปที่ 3.9: ผลการทดสอบ Attitude control ครั้งที่ 1

จากกราฟจะเห็นได้ว่าทิศทางการหมุนทั้งสามนั้นสามารถลู่เข้าตำแหน่งที่ต้องการได้แต่ จะมี Roll กับ Pitch ที่จะไม่สามารถลู่เข้าสู่ตำแหน่งได้ทันก่อนที่ตำแหน่งจะเปลี่ยนไป

การทดลองครั้งที่ 2 ต้องการให้ทิศทางการหมุน Roll กับ Pitch ลู่เข้าสู่ตำแหน่งได้ทันก่อนที่ตำแหน่งจะเปลี่ยน จึงได้ทำการกำหนดค่าน้ำหนักเมทริก Q กับ R ขึ้นมาใหม่ดังนี้

$$
\begin{gathered}
Q_{Attitude} = diag(\begin{bmatrix}100 & 100 & 10 & 1 & 1 & 1\end{bmatrix}) \\
R_{Attitude} = diag(\begin{bmatrix}1 & 1 & 1\end{bmatrix}) \\
K_{Attitude} = \begin{bmatrix} 10 & 0 & 0 & 1.1092 & 0 & 0 \\ 0 & 1 & 3.23e^{-17} & 0 & 1.1092 & -2.16e^{-17} \\ 0 & 1.562e^{-15} & 3.1623 & 0 & -1.1433e^{-17} & 1.0667 \end{bmatrix}
\end{gathered} \tag{3.8}
$$



รูปที่ 3.10: ผลการทดสอบ Attitude control ครั้งที่ 2

3.2.3 X Y Position control

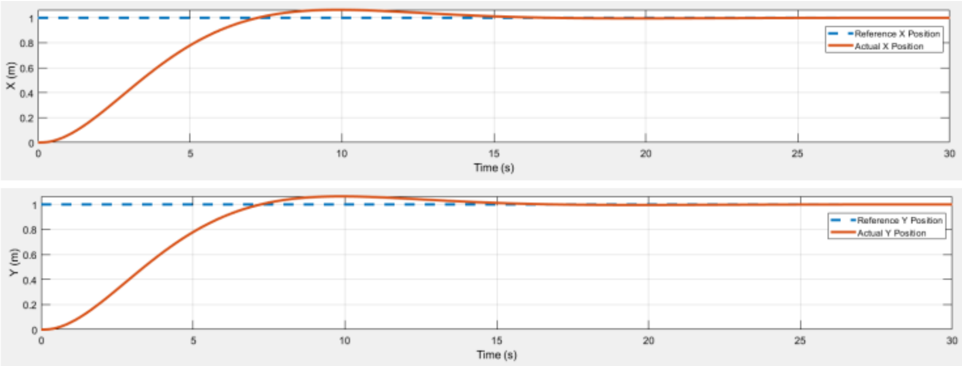
มี state และ input คือ

$$
\vec{x} = \begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix}, \vec{u} = \begin{bmatrix} \phi \\ \theta \end{bmatrix} \tag{3.9}
$$

การทดลองครั้งที่ 1 กำหนดค่าน้ำหนักเมทริก Q กับ R ให้กับ X Y Position control ดังนี้

$$
\begin{gathered}
Q_{XY} = diag(\begin{bmatrix}1 & 1 & 10 & 10\end{bmatrix}) \\
R_{XY} = diag(\begin{bmatrix}1 & 1\end{bmatrix}) \\
K_{XY} = \begin{bmatrix} 0 & -1 & 0 & -3.1944 \\ 1 & 0 & 3.1944 & 0 \end{bmatrix}
\end{gathered} \tag{3.10}
$$

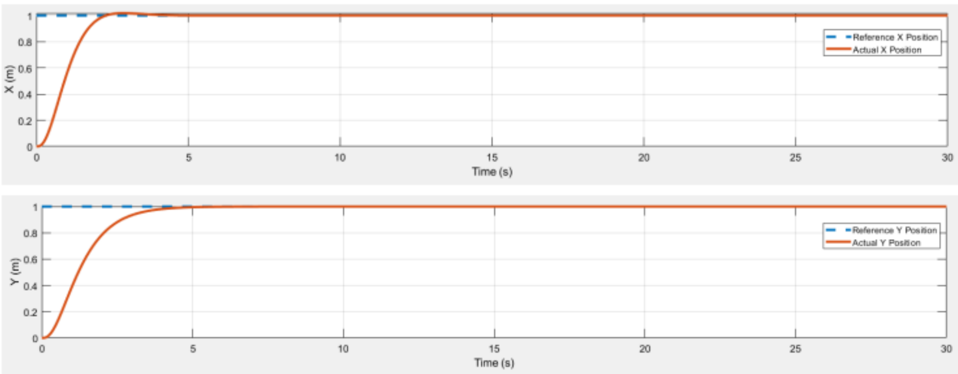
จากกราฟจะเห็นได้ว่าตำแหน่งของแกน X และ Y สามารถลู่เข้าสู่ตำแหน่งที่ต้องการได้แต่ใช้เวลานานถึง 15 วินาที นอกจากนี้ยังมี Overshoot อยู่เล็กน้อยอีกด้วย



รูปที่ 3.11: ผลการทดสอบ XY Position control ครั้งที่ 1

การทดลองครั้งที่ 2 ต้องการให้ทิศทางการหมุน Roll กับ Pitch ลู่เข้าสู่ตำแหน่งได้ทันก่อนที่ตำแหน่งจะเปลี่ยน จึงได้ทำการกำหนดค่าน้ำหนักเมทริก Q กับ R ขึ้นมาใหม่ดังนี้

$$\begin{gathered} Q_{XY} = diag(\begin{bmatrix} 200 & 100 & 200 & 200 \end{bmatrix}) \\ R_{XY} = diag(\begin{bmatrix} 1 & 1 \end{bmatrix}) \\ K_{XY} = \begin{bmatrix} 0 & -10 & 0 & -14.2141 \\ 14.1421 & 0 & 14.2437 & 0 \end{bmatrix} \end{gathered} \tag{3.11}$$



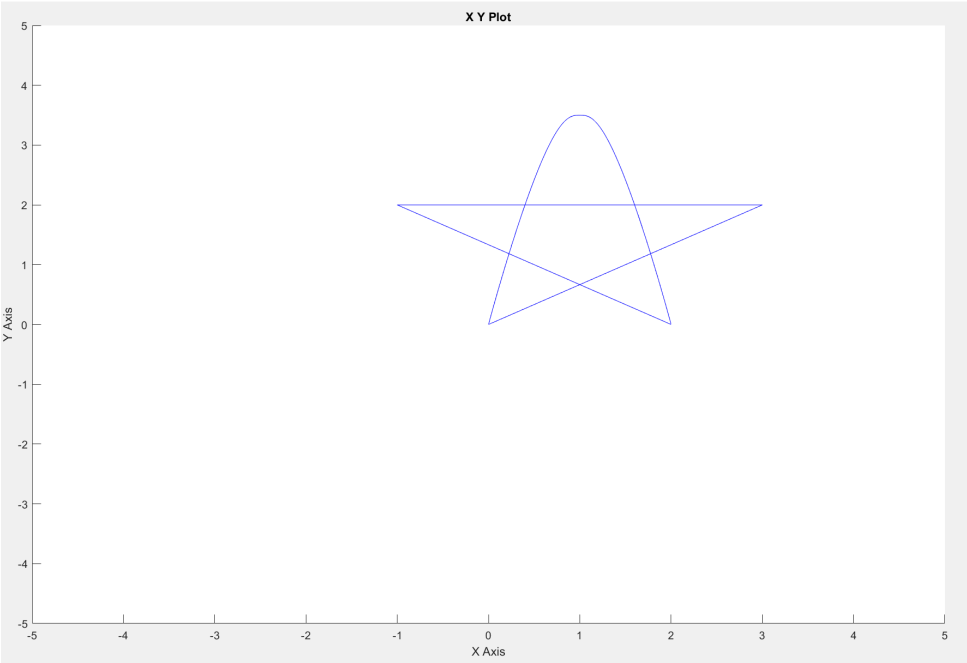
รูปที่ 3.12: ผลการทดสอบ XY Position control ครั้งที่ 2

จากกราฟจะเห็นได้ว่าตำแหน่ง X และ Y ของ Quadrotor นั้นสามารถลู่เข้าสู่ตำแหน่งที่ต้องการได้เร็วขึ้นและยังลด Overshoot ลงได้อีกด้วย

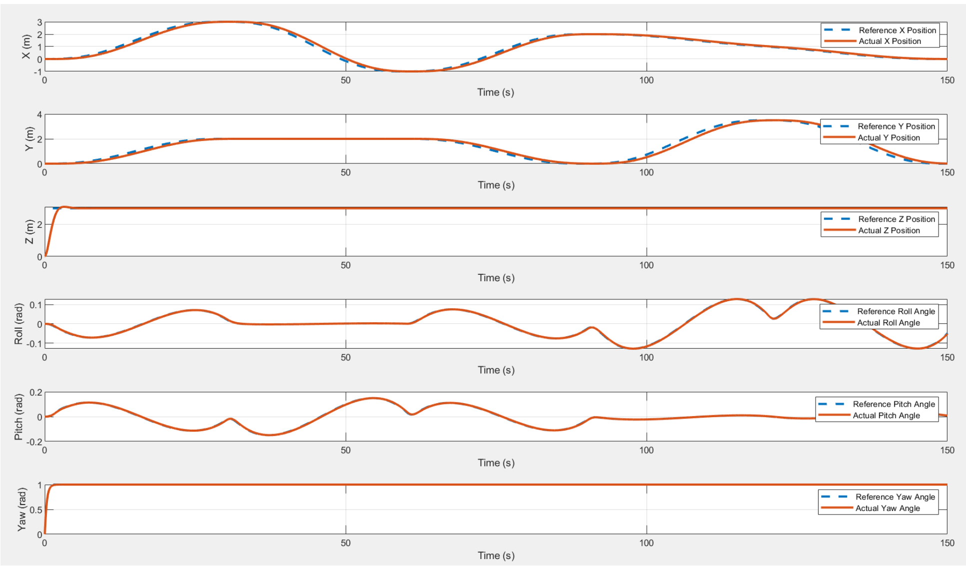
3.3 Linear Quadratic Regulator กับ Path Planning

ในการทดลองครั้งนี้จะนำตัวควบคุม LQR ที่ได้ทำการทดลองหาค่าน้ำหนักที่ Q กับ R เหมาะสมจากการทดลองในบทก่อนหน้านี้มาทำการรวมกับ Path planning เพื่อทำทดลองว่า Quadrotor สามารถบินไปตามตำแหน่งต่างๆตามที่ Path planning สั่งได้

ทำการสร้างเส้นทางที่เป็นรูปดาวด้วย Path Planning



รูปที่ 3.13: เส้นทางของการวิ่งรูปดาว



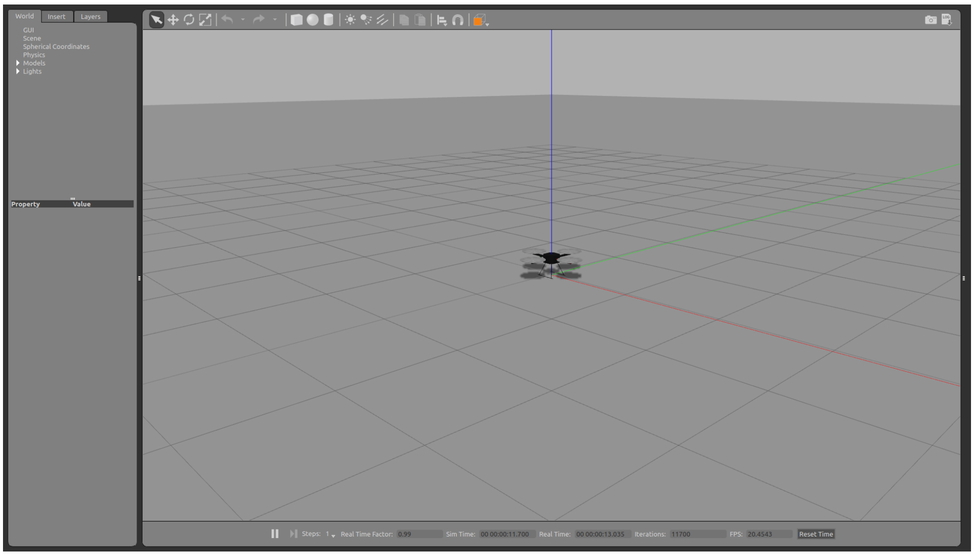
รูปที่ 3.14: สถานะQuadrotor ของการบินเป็นรูปดาว

จากกราฟจะได้เห็นว่า Quadrotor สามารถบินไปตามตำแหน่ง และ ทิศทางการหมุนตามที่ต้องการได้

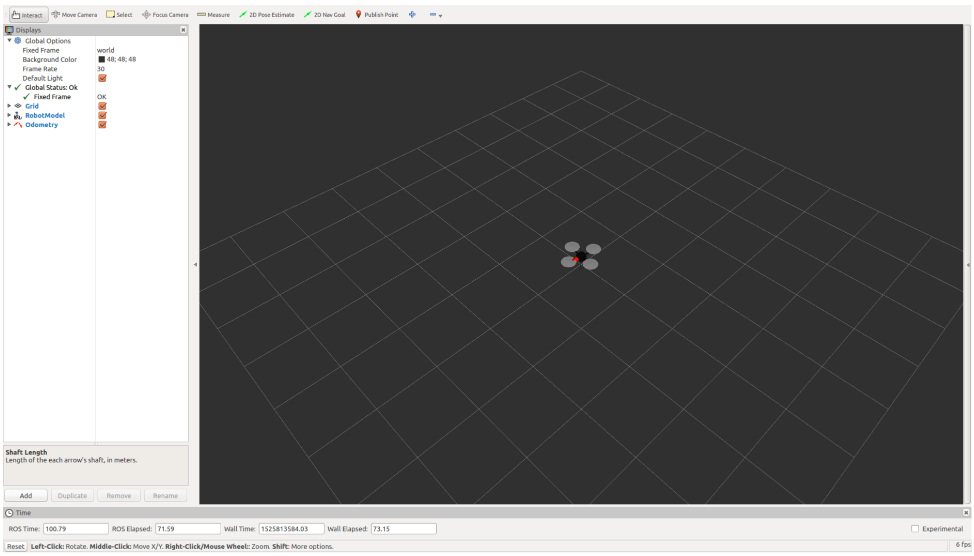
3.4 Gazebo and Rviz

ทดสอบการทำงานของโปรแกรม Matlab โดยจำลองในโปรแกรม Gazebo และแสดงผลภาพใน Rviz การทดสอบแบ่งเป็น 3 ตัวอย่าง

1) สั่งให้ quadrotor บินขึ้นเหนือพื้น
2) สั่งให้ quadrotor บินวนเป็นเกลียว
3) สั่งให้ quadrotor บินเป็นรูปดาว



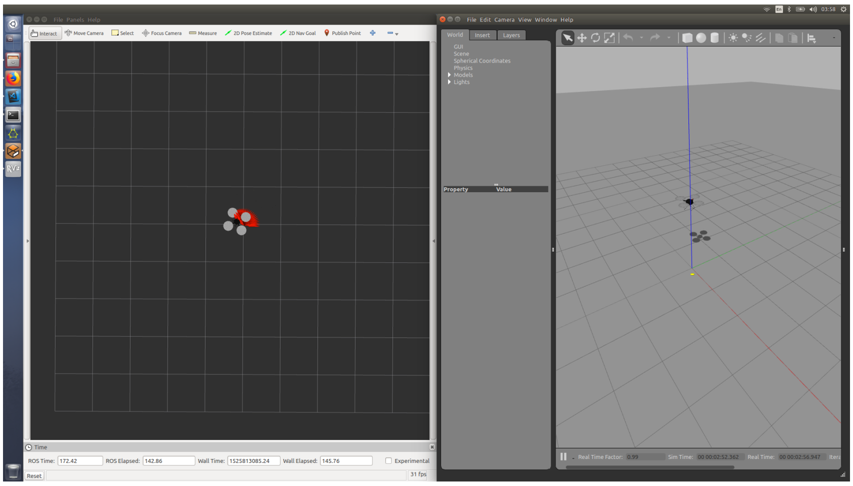
(ก) การจำลอง quadrotor ด้วยโปรแกรม Gazebo



(ข) การแสดงผลภาพ quadrotor ด้วยโปรแกรม Rviz

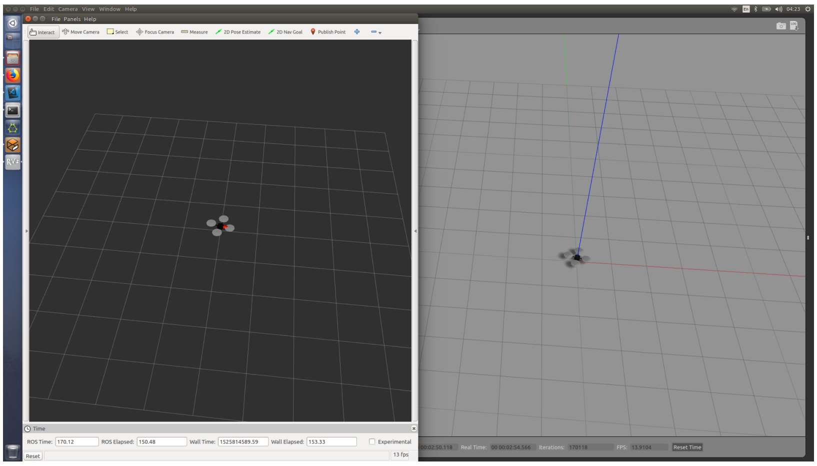
รูปที่ 3.15: การจำลองและการแสดงผลภาพของ quadrotor

3.4.1 สั่งให้ quadrotor บินขึ้นเหนือพื้น

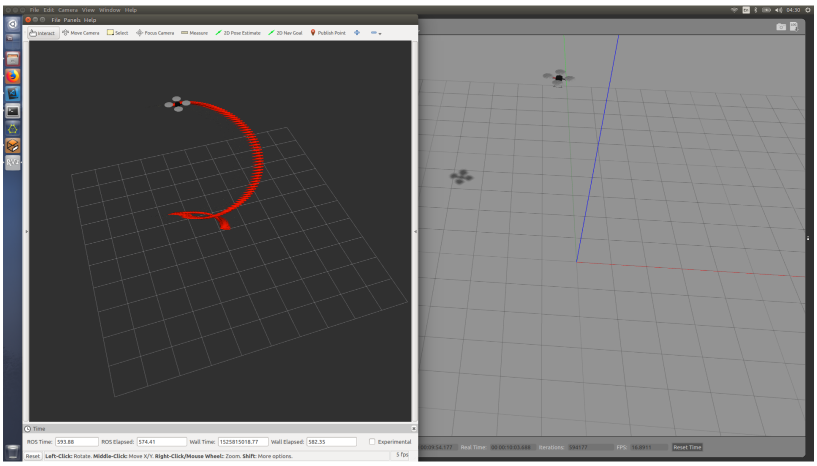


รูปที่ 3.16: ผลการสั่งให้ quadrotor บินขึ้นเหนือพื้น 2 เมตร

3.4.2 สั่งให้ quadrotor บินวนเป็นเกลียว



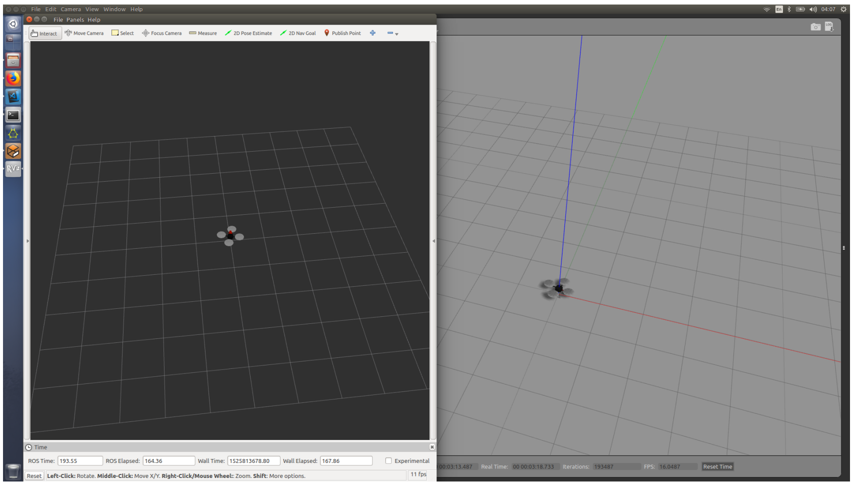
(ก) quadrotor บินวนเป็นเกลียว ขั้นที่ 1



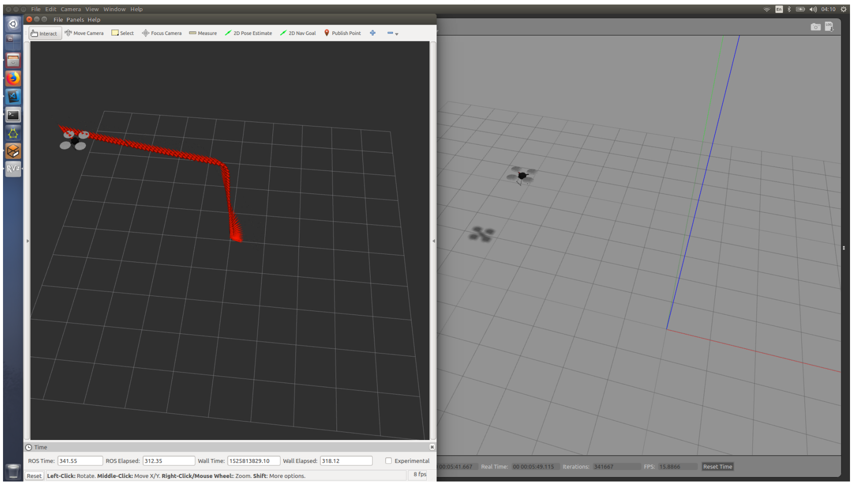
(ข) quadrotor บินวนเป็นเกลียว ขั้นที่ 2

รูปที่ 3.17: ผลการสั่งให้ quadrotor บินวนเป็นเกลียว

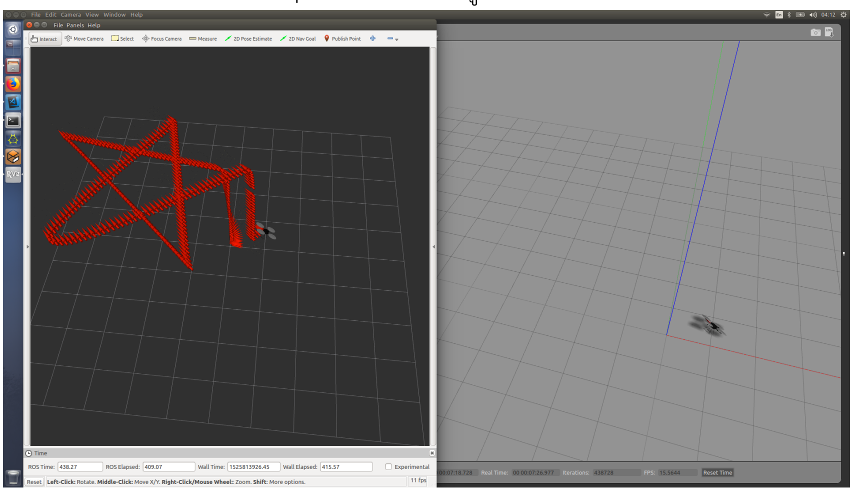
3.4.3 สั่งให้ quadrotor บินเป็นรูปดาว



(ก) quadrotor บินเป็นรูปดาว ขั้นที่ 1



(ข) quadrotor บินเป็นรูปดาว ขั้นที่ 2



(ค) quadrotor บินเป็นรูปดาว ขั้นที่ 3

รูปที่ 3.18: ผลการสั่งให้ quadrotor บินเป็นรูปดาว

บทที่ 4
Conclusion

เนื่องจาก Quadrotor เป็นอากาศยานที่มีความซับซ้อนและน่าสนใจ ดังนั้นถ้าหากเรารู้จักระบบของมันได้เป็นอย่างดี จะทำให้เราสามารถใช้ประโยชน์จากศักยภาพของด้วยได้อย่างเต็มประสิทธิภาพ จากการทดลองที่ผ่านมาทั้งหมด สามารถสรุปว่าแบบจำลอง Quadrotor ที่ออกแบบนั้นสามารถนำมาใช้ในการควบคุมได้, ตัวควบคุม Quadrotor ที่ใช้มีประสิทธิ์ภาพเพียงพอต่อการใช้งานและสามารถควบคุมสถานะต่างๆของ Quadrotor ได้ และสุดท้าย Path planning สามารถสร้างเส้นทางการบินของ Quadrotor ได้ การทำงานทุกส่วนสามารถนำมาใช้งานร่วมกันได้ ทำให้เราได้ระบบจำลองและระบบควบคุมเพื่อทดสอบระบบก่อนนำไปประยุกต์ใช้งานกับระบบจริงต่อในอนาคต ซึ่งทั้งหมดเป็นผลที่เกิดมาจากการเรียนรายวิชา Principles of Model-based Design ที่ทำให้ได้ศึกษาและเข้าใจอากาศยานสี่ใบพัด

เอกสารอ้างอิง

[1] E. Johnson and D. Schrage. *The georgia tech unmanned aerial research vehicle: Gtmax*. AIAA Guidance, Navigation, and Control Conference, 2003.

[2] X. Z. J. Qi and Z. Jiang. *Design and implement of a rotorcraft uav testbed*. EEE International Conference on Robotics and Biomimetics, 2006.