Оглавление

Введение

Мультироторные летательные беспилотные аппараты используются во многих сферах. Они используются для съемок, доставок, перебрасывания кабелей и многих других областей. Квадрокоптер – мультироторный беспилотный летательный аппарат, имеющий 4 ротора, является самым популярным среди прочих вертолетноподобных летательных аппаратов. В своей основе имеют крестообразную раму, равной длины плеч, в центре которой располагается полетный контроллер, аккумулятор, а на концах креста располагаются моторы с прикрепленными на них винтами. Управляются они, в основном, человеком-оператором с пультом дистанционного управления. Так же имеются варианты автоматического управления, например, по бортовой камере, или по заранее заданному маршруту, который построен по системам навигации.

Задача стабилизации квадрокоптера является основной и обязательной для летательного аппарата. Благодаря наклонам по осям квадрокоптер может перемещаться вперед, назад, влево, вправо и менять направление курса, поворачиваясь по часовой и против часовой стрелки. Благодаря поддерживаемому углу наклона, регулируется скорость перемещения квадрокоптера в пространстве. Изменение угла крена позволяет коптеру двигаться влево-вправо. Изменение угла тангажа позволяет коптеру двигаться вперед-назад. Изменение угла рысканья позволяет коптеру вращаться около своей оси, изменяя направление курса.

Базовым звеном для системы управления является регулятор. Данное устройство позволяет рассчитывать входные воздействия для объекта управления относительно заданных величин и выходных величин объекта управления, измеренных при помощи датчиков.

Глава 1. Техническое задание

1. Цели и задачи

   1.1. Назначение системы

   Данная система должна стабилизировать квадрокоптер в заданных пределах по углам рысканья, тангажа и крена в пространстве.

   1.2. Цели создания системы

   Данная система создается для решения задачи стабилизации квадрокоптера в пространстве.

2. Характеристика объекта

   2.1. Краткие сведения об объекте

   Квадрокоптер Eachine e708 имеет:

   - ДПТ CL-0720-12
   - редуктор с передаточным числом 6:1
   - винты, диаметром 0.138 м
   - расстояние между осями моторов по диагонали 0.225м
   - расстояние между осями моторов по стороне 0.159 м
   - высота цилиндра, в котором находится двигатель 0.058 м
   - радиус цилиндра, в котором находится двигатель 0.015 м
   - масса цилиндра, в котором находится двигатель 0.005 кг
   - ширина центрального корпуса 0.044 м
   - высота центрального корпуса 0.034 м
   - длина центрального корпуса 0.104 м
   - масса центрального корпуса 0.083 кг

3. Требования к системе

   3.1. Требования к системе в целом

   Система управления должна отработать входящее воздействие в течение 3 секунд, перерегулирование в пределах 10%, обеспечить статическую ошибку в пределах 5%. Входящее воздействие по крену и тангажу ограничено ±1 рад.

4. Сроки

Дата начала	Дата конца	Перечень работ
1.05.18	14.05.18	Описание физических аспектов модели квадрокоптера
15.05.18	16.05.18	Вычисление физических параметров квадрокоптера и создание модели квадрокоптера в среде моделирования VisSim
17.05.18	21.05.18	Создание системы управления
22.05.18	23.05.18	Создание принципиальной схемы
24.05.18	2.06.18	Создание ПО для микроконтроллера
3.06.18	10.06.18	Оформление ВКР

5. Компоненты модуля стабилизации
   5.1. MPU6050;
   5.2. Arduino nano
   5.3. IRLR8726

Глава 2. Разработка модели

2.1. Описание модели квадрокоптера

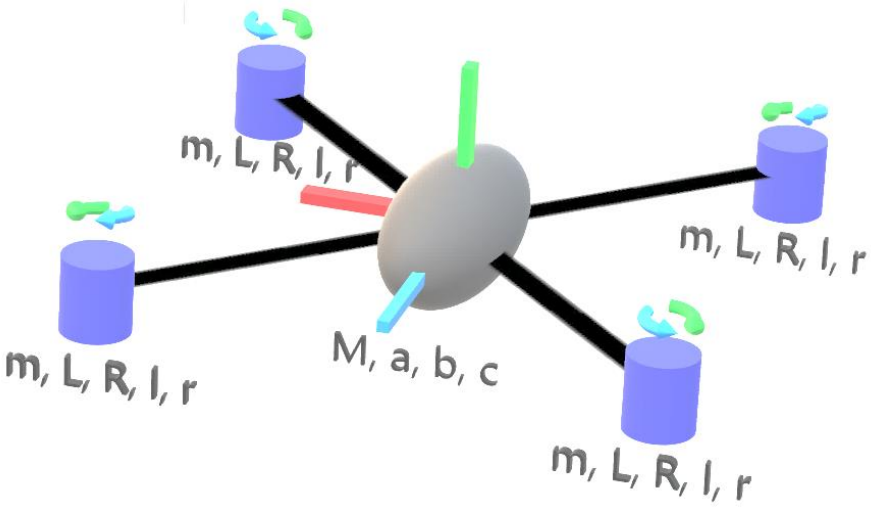


Рис.2. Общий вид модели коптера

Где ось Ox – голубая, Oy- красная, Oz – зеленая; M – масса тела коптера; a, b, c – параметры эллипсоида (по оси Ox – b, Oy – c, Oz – a); m -масса цилиндра, в котором находится двигатель, L – высота цилиндра, в котором находится двигатель, R – радиус цилиндра, в котором находится двигатель, l – расстояние от центра коптера до двигателя, r – радиус винта; зелеными стрелками указано направление вращения двигателя, синими – направления момента сопротивления винта, плечи соединения тела и двигателей и тела.

2.2. Определение моментов инерции

Ox

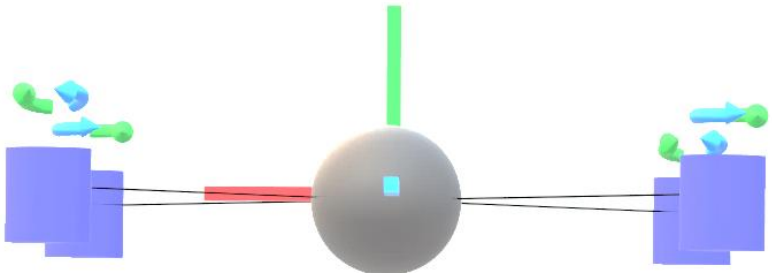


Рис. 3. Модель коптера вдоль оси Ox

$$\mathcal{J}_x = M\frac{a^2 + c^2}{5} + 4m\left(\frac{R^2}{4} + \frac{L^2}{12} + l^2\right) \quad (1)$$

Oy

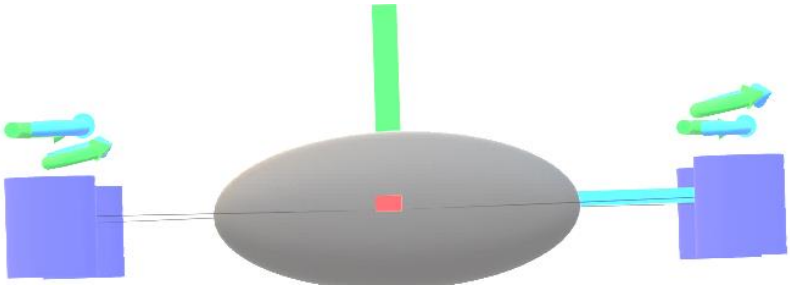


Рис. 4. Модель коптера вдоль оси Oy

$$\mathcal{J}_y = M\frac{a^2 + b^2}{5} + 4m\left(\frac{R^2}{4} + \frac{L^2}{12} + l^2\right) \quad (2)$$

Oz

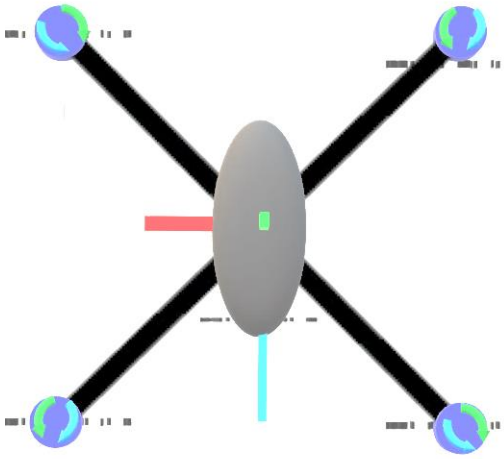


Рис. 5. Модель коптера вдоль оси Oz

$$\mathcal{J}_z = M\frac{c^2 + b^2}{5} + 4m\left(\frac{R^2}{2} + l^2\right) \quad (3)$$

2.3. Матрицы сил и вращения твердого тела

Подъемная сила каждого винта с модулем P направлена вверх, сонаправлено оси Oz, относительно самого коптера.

$$\begin{cases} P_k = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ P \end{bmatrix} \\ P = \sum_{i=1}^{4} P_i = \sum_{i=1}^{4} c_{p_i} w_i^2 \\ G = \begin{bmatrix} 0 \\ 0 \\ -(M+4m)g \end{bmatrix} \end{cases} \quad (4)\text{-}(6)$$

Где, P_k – матрица подъемной силы, разбитой по осям, P – суммарная тяга, P_i – сила тяги i-го винта, c_{p_i} – коэффициент силы тяги i-го винта, ρ – плотность воздуха, c_a – коэффициент подъемной силы, S_i – площадь отметаемой i-ым винтом поверхности, r_i- радиус i-го винта, f – матрица сил сопротивления воздуха, G – матрица сил притяжения, g – ускорение свободного падения.

Допускаем, что коптер симметричен, центр масс коптера расположен строго в точке начала координат, тогда

$$\begin{cases} \mathcal{J}_x \ddot{\phi} = M_{R_x} \\ \mathcal{J}_y \ddot{\theta} = M_{R_y} \\ \mathcal{J}_z \ddot{\psi} = M_{R_z} \end{cases} \quad (7)$$

$$M_R = \begin{bmatrix} M_{R_x} \\ M_{R_y} \\ M_{R_z} \end{bmatrix} = M_q \quad (8)$$

$$M_q = \begin{bmatrix} M_{q_x} \\ M_{q_y} \\ M_{q_z} \end{bmatrix} = \begin{bmatrix} (P_2 + P_3 - P_1 - P_4) l \sin 45^\circ \\ (P_2 + P_3 - P_1 - P_4) l \cos 45^\circ \\ M_2 + M_4 - M_1 - M_3 \end{bmatrix} \quad (9)$$

$$M_i = c_{p_i} r w^2 \quad (10)$$

$$w_i = k V_i U_i \quad (11)$$

Где, M_R – матрица результирующего момента, M_q – матрица моментов от винтов, kV – коэффициент преобразования напряжения в обороты для

напряжений от 0В до 4.2В, где, согласно документации на мотор, идет линейная зависимость.

Глава 3. Реализация модели в VisSim

3.1. Структура модели коптера

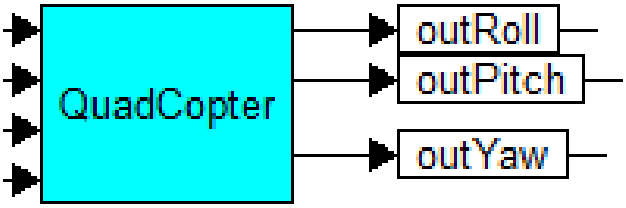


Рис.6. Модель коптера в VisSim

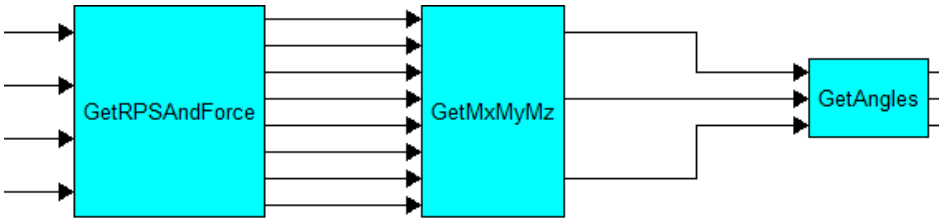


Рис.7. Структурная схема модели коптера

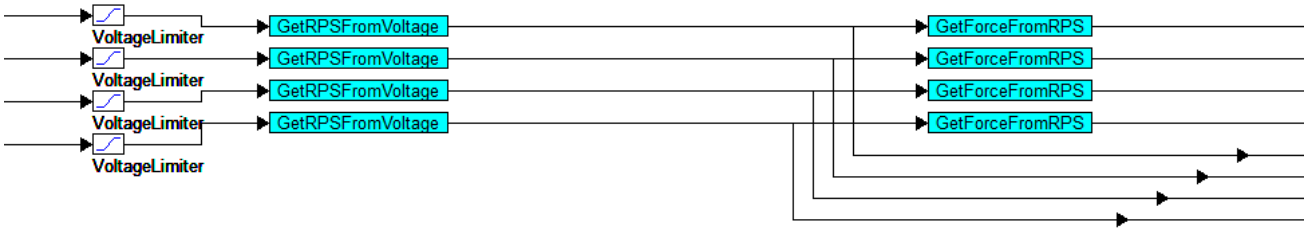


Рис.8. Блок GetRPSAndForce

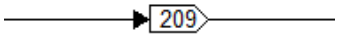


Рис.9. Блок GetRPMFromVoltage

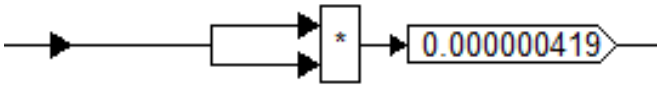


Рис.10. Блок GetForceFromRPS

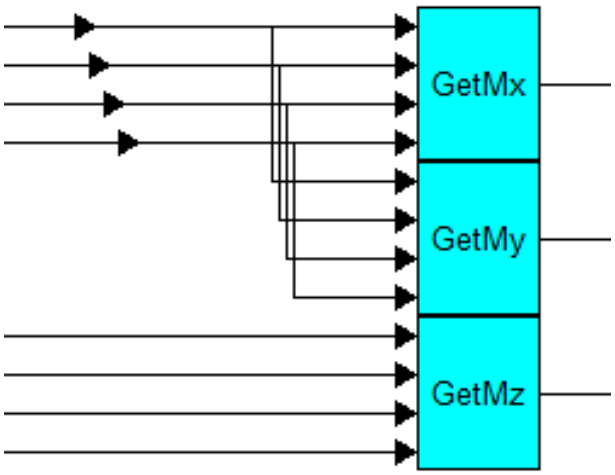


Рис.11. Блок GetMxMyMz

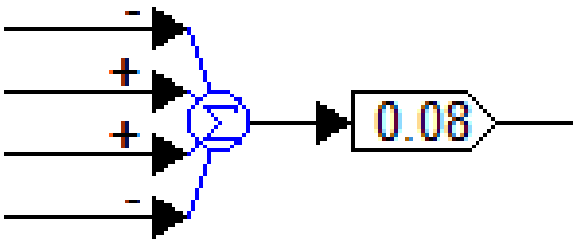


Рис.12. Блок GetMx

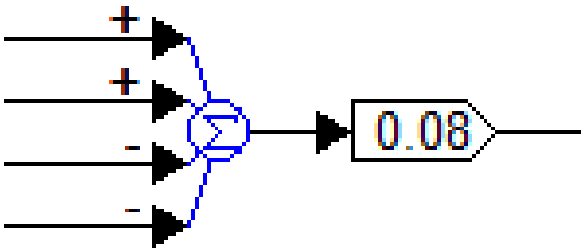


Рис.13. Блок GetMy

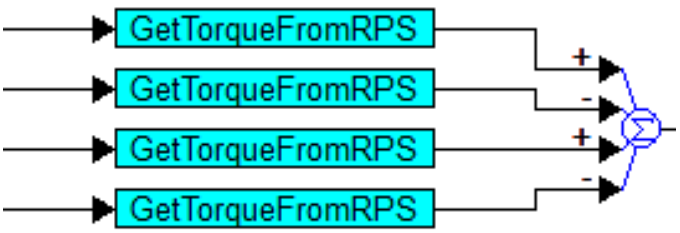


Рис.14. Блок GetMz

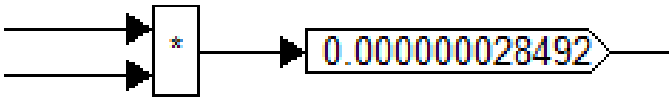


Рис.15. Блок GetTorqueFromRPS

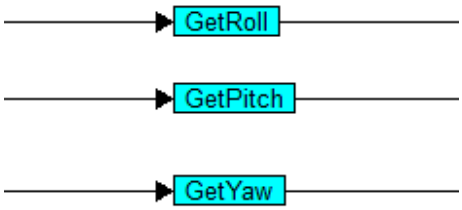


Рис.16. Блок GetAngles

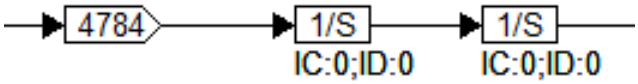


Рис.17. Блок GetRoll

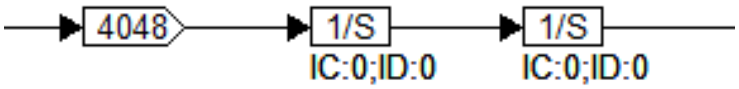


Рис.18. Блок GetPitch

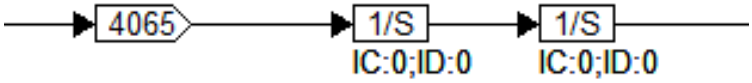


Рис.19. Блок GetYaw

3.2. Результаты моделирования

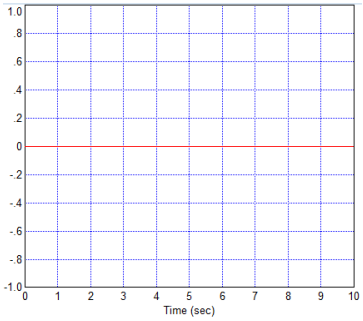


Рис.20. Выходной угол крена при одинаковом напряжении на всех моторах (взлет)

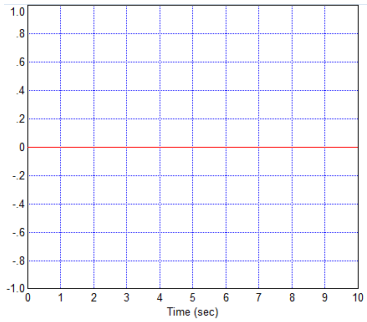


Рис.21. Выходной угол тангажа при одинаковом напряжении на всех моторах (взлет)

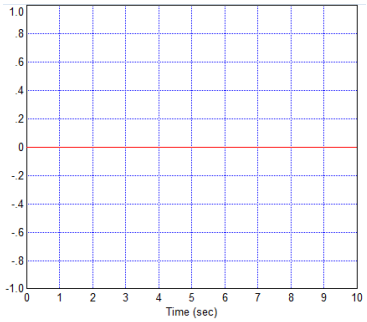


Рис.22. Выходной угол рысканья при одинаковом напряжении на всех моторах (взлет)

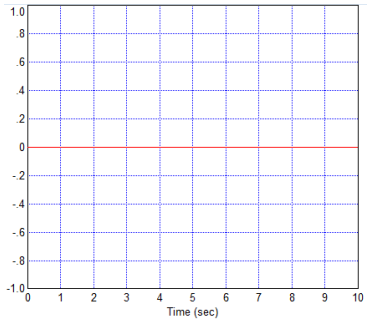


Рис.23. Выходной угол крена при большей мощности на передних моторах (тангаж)

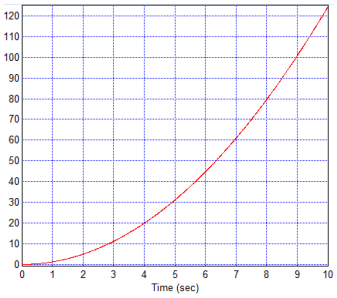


Рис.24. Выходной угол тангажа при большей мощности на передних моторах (тангаж)

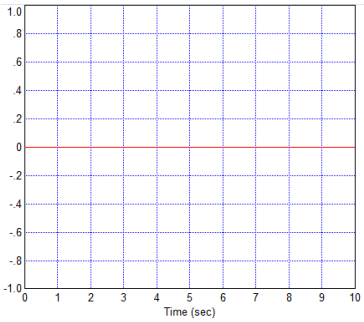


Рис.25. Выходной угол рысканья при большей мощности на передних моторах (тангаж)

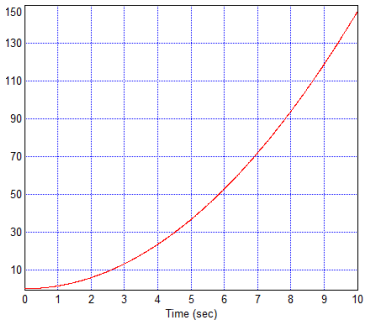


Рис.26. Выходной угол крена при большей мощности на левых моторах (крен)

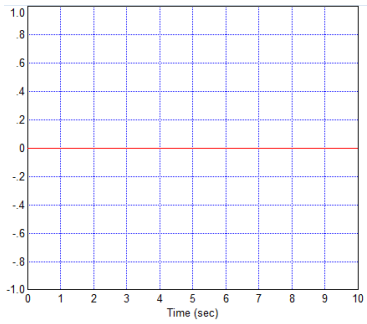


Рис.27. Выходной угол тангажа при большей мощности на левых моторах (крен)

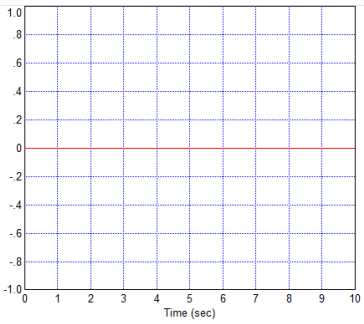


Рис.28. Выходной угол рысканья при большей мощности на левых моторах (крен)

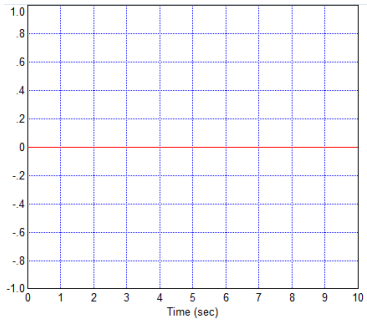


Рис.29. Выходной угол крена при большей мощности на диагональных моторах (рысканье)

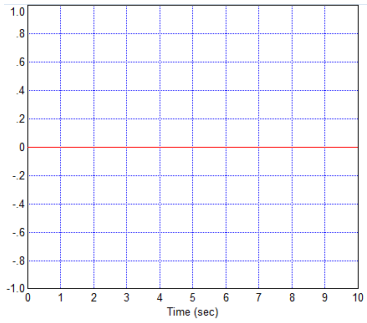


Рис.30. Выходной угол тангажа при большей мощности на диагональных моторах (рысканье)

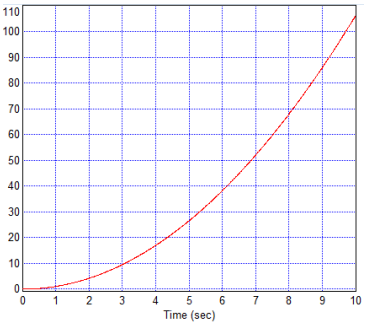


Рис.31. Выходной угол рысканья при большей мощности на диагональных моторах (рысканье)

Результаты моделирования показывают, что объект неустойчив и для его стабилизации требуется система автоматического управления.

Глава 4. Вычисление физических параметров коптера EACHINE e708

Как модель коптера для моделирования, был выбран коптер EACHINE e708.

4.1. Расчет моментов инерции по осям

Ox

$$\mathcal{J}_x = M\frac{a^2+c^2}{5} + 4m\left(\frac{R^2}{4}+\frac{L^2}{12}+l^2\right) =$$

$$= 0.083\frac{0.017^2 + 0.022^2}{5} + 4\times 0.005\left(\frac{0.015^2}{4}+\frac{0.058^2}{12}+0.0975^2\right) =$$

$$= 0.000209 \text{ кг} \times \text{м}^2$$

Oy

$$\mathcal{J}_y = M\frac{a^2+b^2}{5} + 4m\left(\frac{R^2}{4}+\frac{L^2}{12}+l^2\right) =$$

$$= 0.083\frac{0.017^2 + 0.0525^2}{5} + 4\times 0.005\left(\frac{0.015^2}{4}+\frac{0.058^2}{12}+0.0975^2\right) =$$

$$= 0.000247 \text{ кг} \times \text{м}^2$$

Oz

$$\mathcal{J}_z = M\frac{c^2+b^2}{5} + 4m\left(\frac{R^2}{2}+l^2\right) =$$

$$= 0.083\frac{0.022^2 + 0.0525^2}{5} + 4\times 0.005\left(\frac{0.015^2}{2}+0.0975^2\right) = 0.000246 \text{ кг} \times \text{м}^2$$

4.2. Определение коэффициента силы тяги винта

Для того, чтобы определить значение коэффициента силы тяги винта, нужно вычислить значение подъемной силы винта. Запишем второй закон Ньютона из матриц (4) и (6).

$$(M + 4m)A = P + G \tag{12}$$

где матрица A – матрица результирующего ускорения по осям связной системы координат.

$$A = \begin{bmatrix} A_x \\ A_y \\ A_z \end{bmatrix} \tag{13}$$

Для того, чтобы определить подъемную силу P, выразим ее из формулы (12)

$$P = (M + 4m)A - G \tag{14}$$

$$\begin{bmatrix} 0 \\ 0 \\ P \end{bmatrix} = (M + 4m) \begin{bmatrix} A_x \\ A_y \\ A_z \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ -(M + 4m)g \end{bmatrix} \tag{15}$$

При подъеме коптера вверх, результирующее ускорение направлено вдоль оси Oz, следовательно, перезапишем формулу (15).

$$P = (M + 4m)(A_z + g) \tag{16}$$

Вычислять ускорение будем при помощи безмена.

После вычисления суммарной подъемной силы, надо его разделить на количество винтов, в данном случае 4, и разделить на квадрат скорости вращения винтов. Для этого нужно получить скорость вращения двигателя и разделить его на передаточное число понижающего редуктора. Скорость вращения мотора получаем из напряжения, подаваемого на моторы. Оно равно максимальному напряжению на батарее коптера 4.2 В.

$$c_p = \frac{P}{4w^2} = \frac{(M + 4m)(A_z + g)}{4w^2} \tag{17}$$

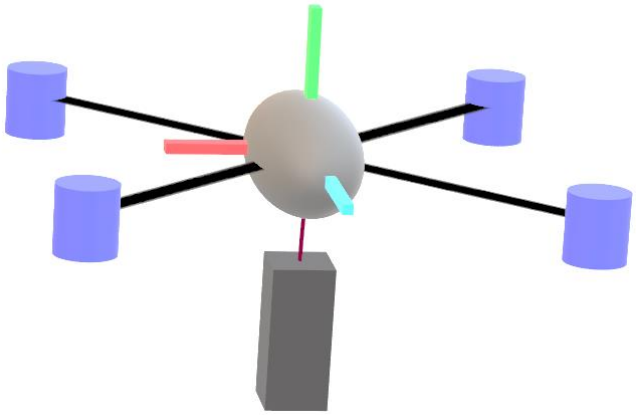


Рис.32. Схема экспериментальной установки. Черный параллелепипед – безмен, прикрепленный к столу.

Безменом будет измеряться произведение суммы масс на ускорение. Значение с безмена будет умножено на 9.8 $^{м}/_{с^2}$, чтобы получить значение силы и поделено на полную сумму массы коптера. Полученное значение массы с безмена равно 0.03 кг. Значение ускорения 2,85 $^{м}/_{с^2}$. Суммарная подъемная сила при взлете 1.3 Н. Скорость оборотов двигателя равна $4.2 \times 12000 = 50400\ ^{об}/_{мин} = 840\ ^{об}/_{с} = 5275\ ^{рад}/_{с}$. На винте имеем скорость вращения $\frac{5275}{6}\ ^{рад}/_{с} = 880\ ^{рад}/_{с}$.

$$c_p = \frac{1.3}{4 \times 880^2} = 0.000000419\ \text{Н} \times \text{с}^2/_{\text{рад}^2}$$

Глава 5. Разработка системы управления

5.1. Выбор регулятора

Во многих публикациях рассказывается об использовании 3 ПИД регуляторов для коррекции напряжений моторов. Поэтому в данной работе будет использован такой же подход.

Александр Дербенёв в своей статье [3] пишет: «Простейшая задача стабилизации — контролировать угол наклона, но из законов динамики вращательного движения следует, что непосредственно управлять можно лишь его второй производной. Самый легкий способ повлиять на нужную величину — использовать ПИД.
На вход регулятора подаётся обобщённая координата (в нашем случае угол), на выходе мы получаем момент сил (вторая производная угла). Каждый ПИД-регулятор стабилизирует значение одной обобщённой координаты. Мы используем три ПИД с постоянными коэффициентами»

5.2. Распределение выхода регуляторов и газа на моторы

Распределение выхода регуляторов и газа на моторы зависит от компоновки, красной стрелкой указано направление квадрокоптера вперед.

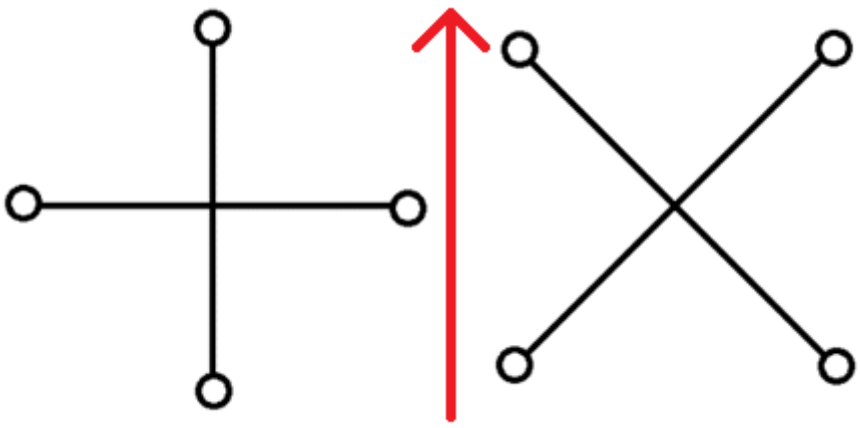


Рис.33. + и X образные квадрокоптеры.

Так как Eachine e708 по своей структуре X-образный, закон управления будем составлять исходя их его конструкции. Для начала нужно пронумеровать моторы.

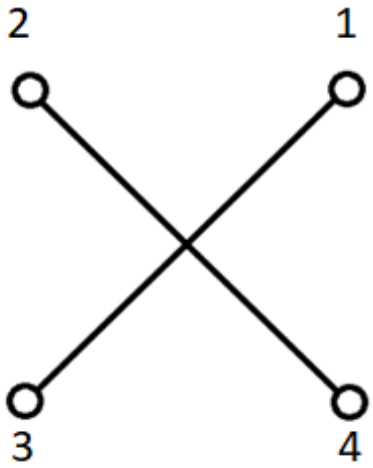


Рис.34. X-образный квадрокоптер с пронумерованными моторами

Газ добавляется ко всем моторам с положительным знаком.

Выход регулятора по крену отнимается от значений газа у моторов 1 и 4, и прибавляется к значениям газа моторов 2 и 3.

Выход регулятора по тангажу отнимается от значений газа у моторов 1 и 2, и прибавляется к значениям газа моторов 3 и 4.

Выход регулятора по рысканью отнимается от значений газа у моторов 2 и 4, и прибавляется к значениям газа моторов 1 и 3.

Итого, матрица управляющих значений выглядит так:

$$\begin{cases} u_1 = T - R - P + Y \\ u_2 = T + R - P - Y \\ u_3 = T + R + P + Y \\ u_4 = T - R + P - Y \end{cases} \quad (18)$$

где, u_i – управление, подаваемое на i-й мотор, T – значение газа, R – значение ПИД регулятора, высчитанное по ошибке крена, P – значение ПИД регулятора, высчитанное по ошибке тангажа, Y – значение ПИД регулятора, высчитанное по ошибке рысканья.

Так же нужно учесть, что, максимальное значение, которое можно записать в порт ШИМ микроконтроллера равно 256, а на выходе регулятора нужно получить напряжение, поэтому выходное значение будем ограничивать от 0 до 256, а потом умножать на 0.016.

5.3. Разработка регулятора в VisSim

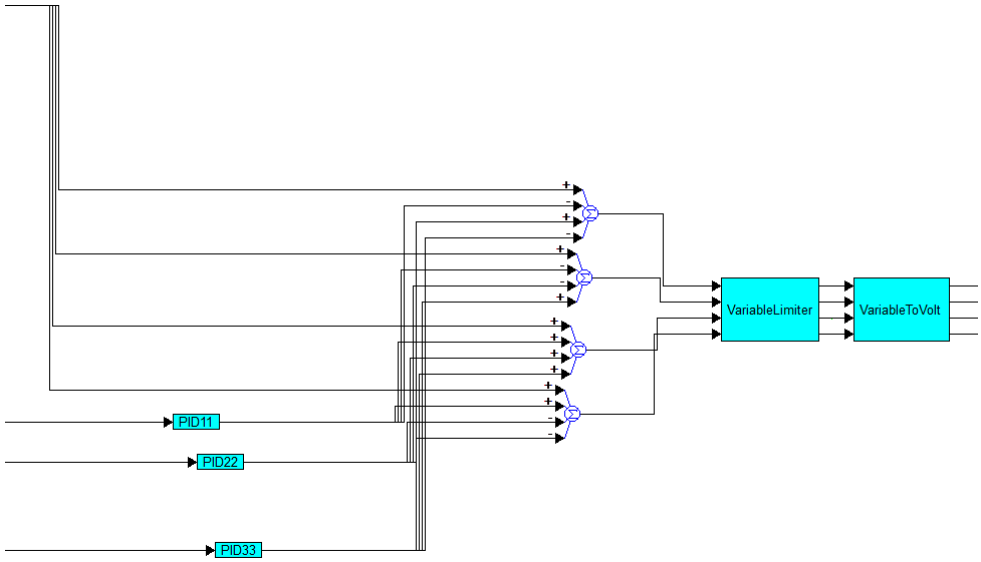


Рис.35. Структура блока PIDRegulator

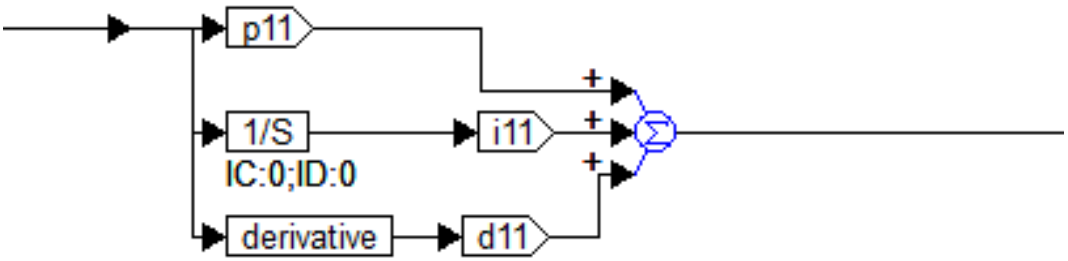


Рис.36. Структура блока PID

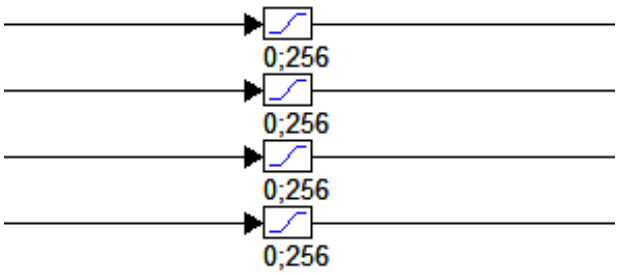


Рис.37. Структура блока VariableLimiter

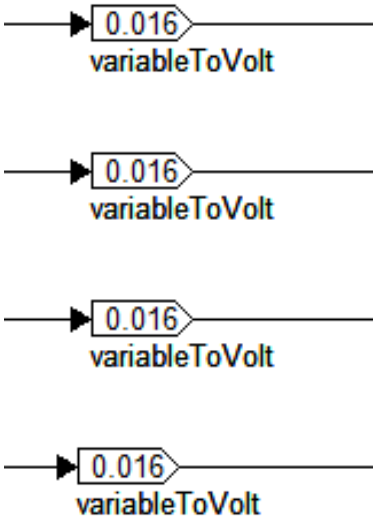


Рис.38. Структура блока VariableToVolt

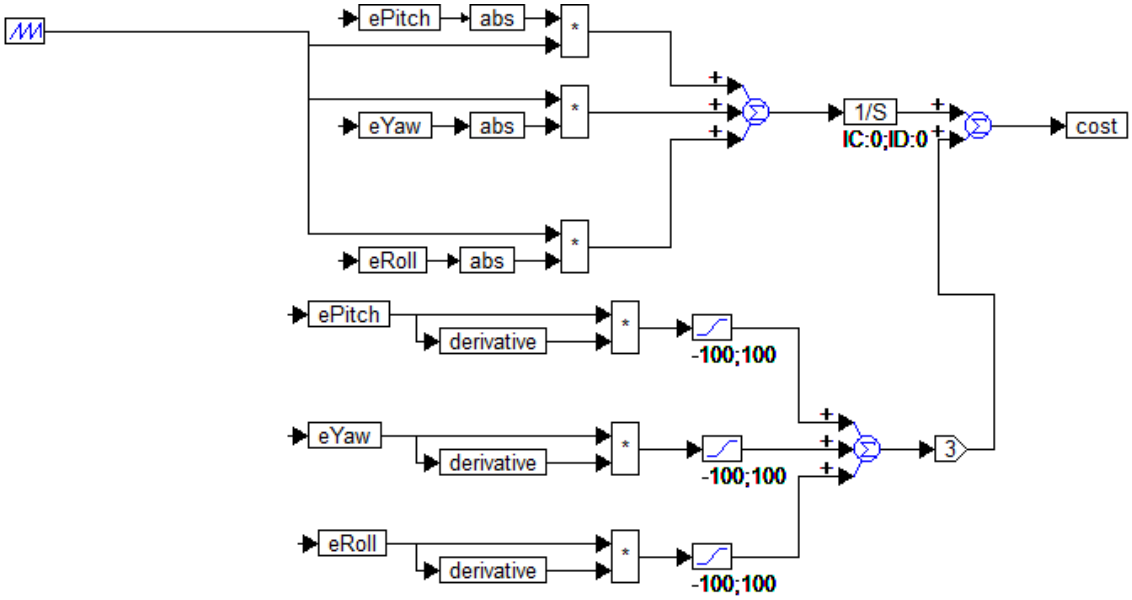


Рис.39. Стоимостная функция

5.4. Результаты оптимизации

Таблица коэффициентов

	Крен	Тангаж	Рысканье
P	15.20834	23.07135	19.04041
I	-0.00005	-0.01472	-0.07566
D	38.19664	44.36351	36.17238

Табл.1. Таблица оптимизированных коэффициентов

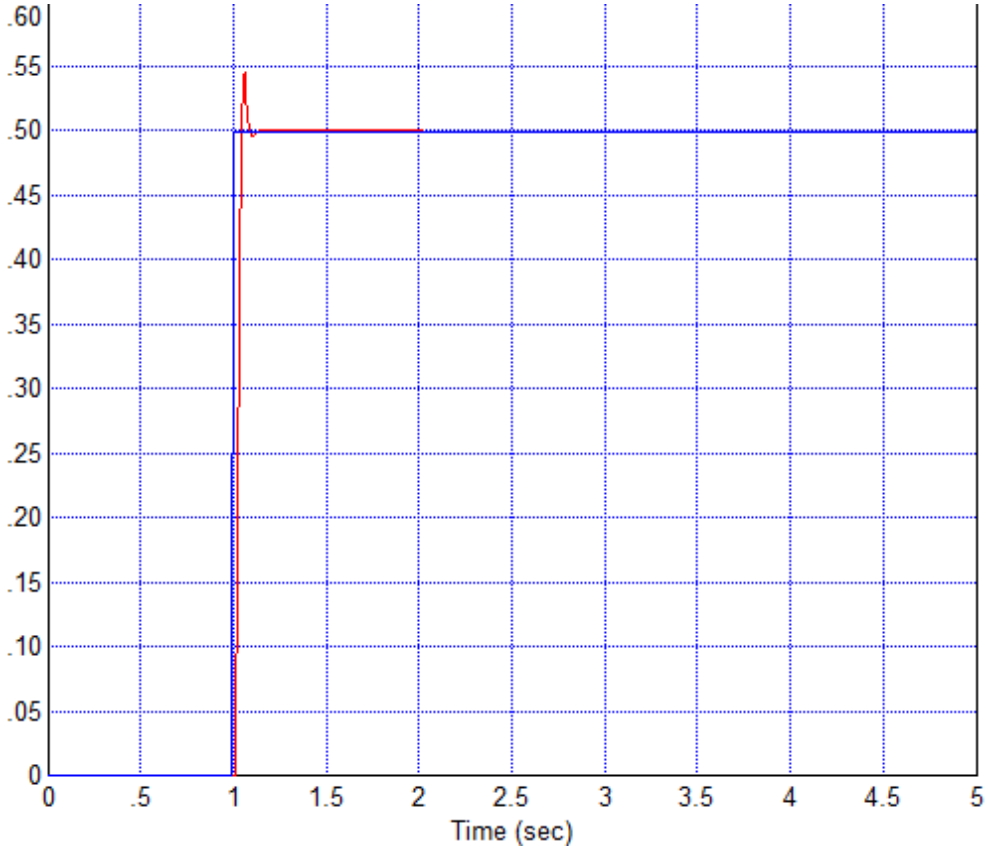


Рис.40. Переходный процесс угла крена

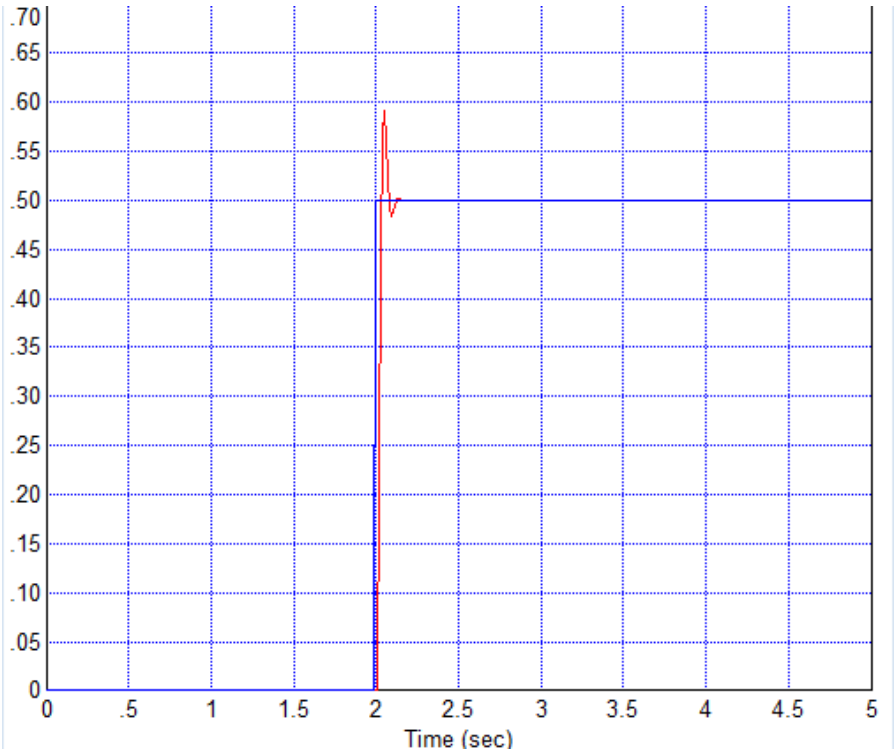


Рис.41. Переходный процесс угла тангажа

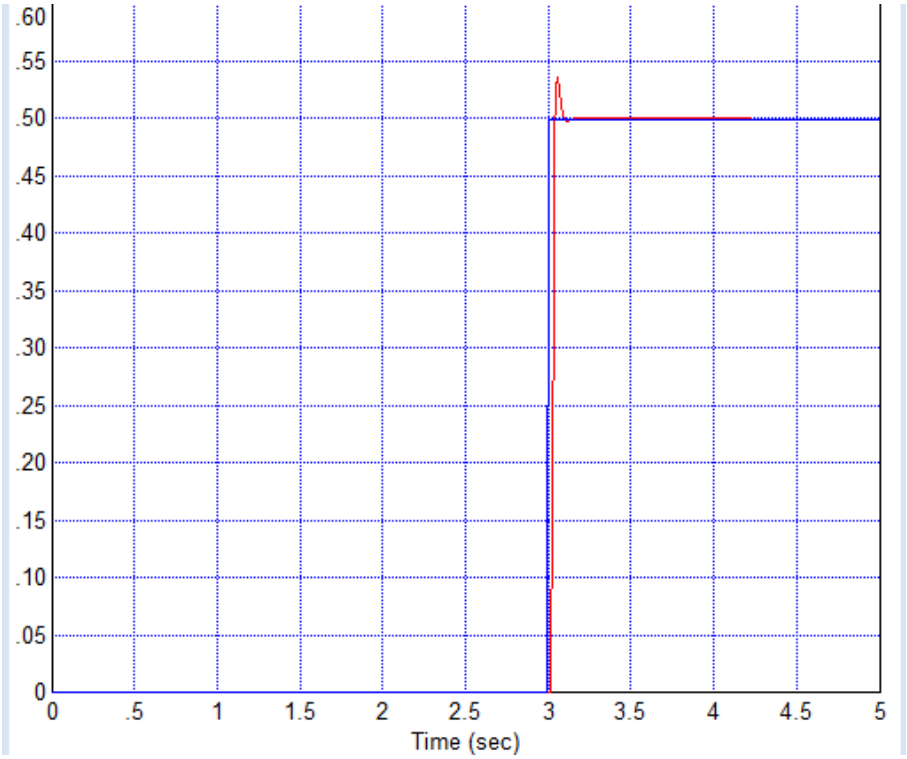


Рис.42. Переходный процесс угла рысканья

5.5. Проверка робастности

Таблица коэффициентов для проверки робасности

	Крен	Тангаж	Рысканье
P	15	23	19
I	-1	-1	-1
D	38	44	36

Табл.2. Таблица коэффициентов для проверки регулятора на робасность.

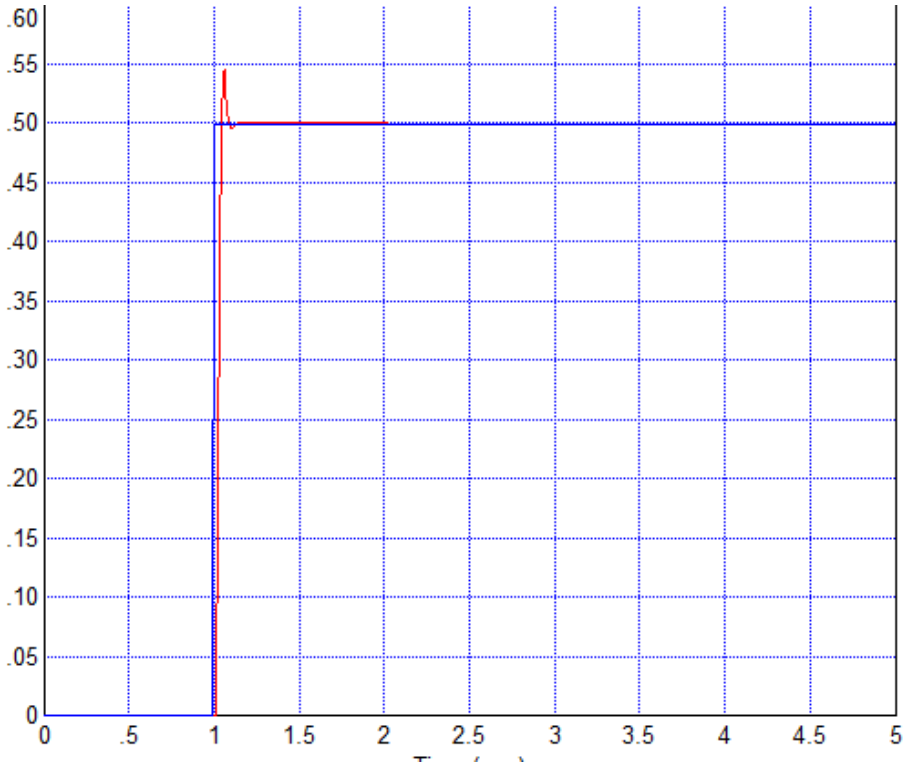


Рис.43. Переходный процесс угла крена

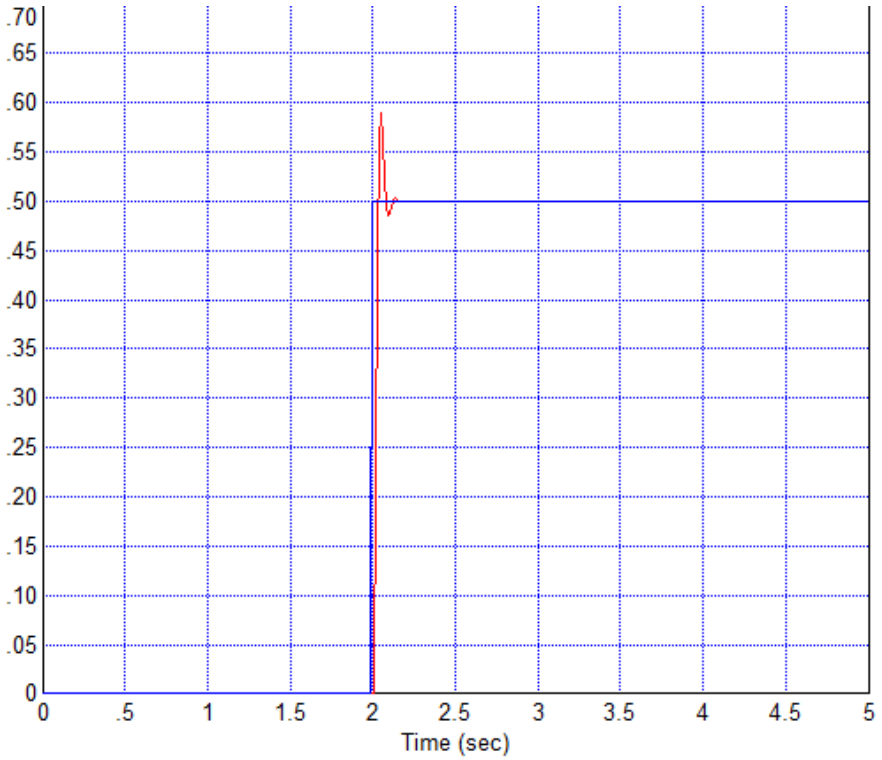


Рис.44. Переходный процесс угла тангажа

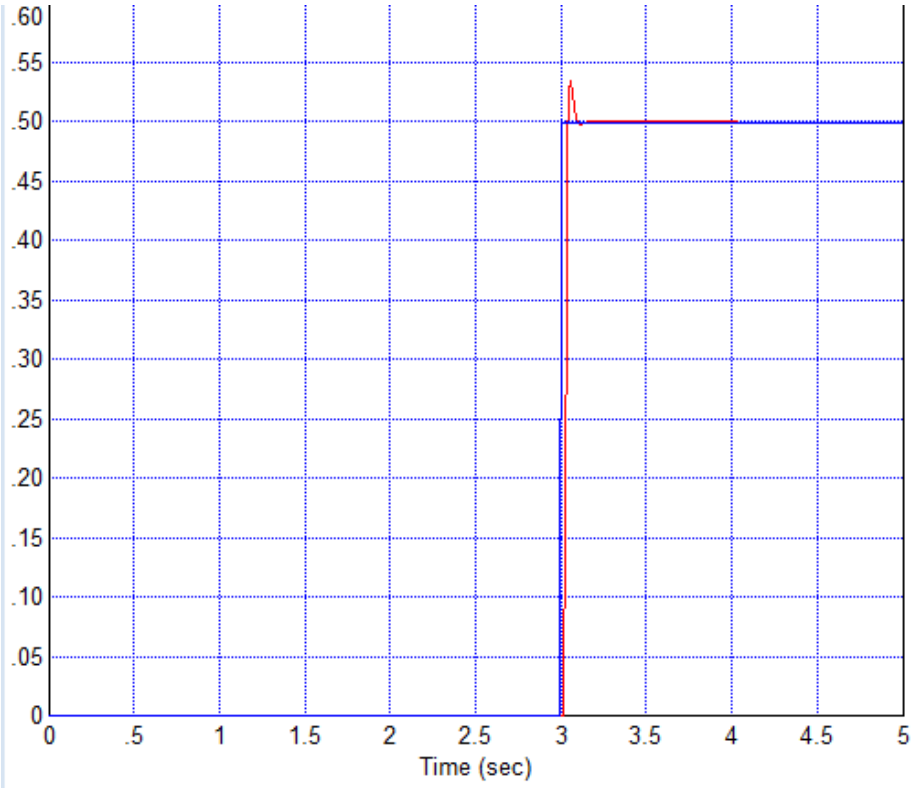


Рис.45. Переходный процесс угла рысканья

Судя по переходным процессам, можно утверждать, что регулятор получился робастным.

Глава 6. Разработка функциональной и принципиальной схем

6.1. Введение

В качестве модуля стабилизации, в которой будет реализовываться алгоритм управления была выбрана платформа Arduino Nano, имеющая микроконтроллер Atmel atmega328p, модуль гироскопа и акселерометра GY521, имеющая чип MPU6050, совмещающая в себе 3-х осевой гироскоп и 3-х осевой акселерометр, для управления ДПТ CL-0720-12, используются МОП-транзисторы логического уровня с индуцированным N-каналом IRLR8726. Выбор на платформу Arduino Nano пал из-за ее компактных размеров, наличия на плате преобразователя интерфейса USB-UART FT232R, стабилизатора напряжения AMS1117CD-5.0 5 В, а также наличия библиотек для работы с модулем GY-521. Так как чип MPU6050 имеет поддержку шины I2C, Arduino Nano будет получать данные о положениях угла по данной шине.

Функционально, микроконтроллер будет в начале опрашивать модуль гироскопа и акселерометра, получив данные, микроконтроллер будет вычислять управляющее воздействие и передавать на моторы

Транзисторы будут подключены к пинам Arduino Nano, которые поддерживают ШИМ. Питание будет идти от идущего в комплекте с квадрокоптером Li-Po аккумулятора 752540 емкостью 550 мАЧ, имеющего одну ячейку с максимально выходным напряжением 4.2 В и номинальным 3.7 В.

Пины, поддерживающие ШИМ, D3, D5, D6, D9, D10, D11. Пины D3, D5, D6 принадлежат порту PD, а D9, D10, D11 принадлежат порту PB. Так как максимальный выходной ток на один порт ограничен 40 мА, нужно равномерно разделить нагрузку между портами для более эффективного управления моторами. Пины, поддерживающие шину I2C это A4 (SDA) и A5 (SCL).

Контакты аккумулятора будут присоединяться к пинам Vin и COM Arduino Nano, где AMS1117CD-5.0 стабилизирует напряжение до 5 В и подаст питание на микроконтроллер.

В приложении указаны функциональная и принципиальная схемы.

6.2. Описание функциональной схемы

В функциональной схеме имеются 4 блока.

1. Микроконтроллер
2. Датчики
3. Управление моторами
4. Моторы

Микроконтроллер в начале цикла опрашивает датчики, после высчитывает управляющее воздействие, передает нужные воздействия на моторы через блок управления моторами.

6.3. Описание принципиальной схемы

В принципиальной схеме имеется:

1. Микроконтроллер Arduino Nano (DD1)
2. Акселерометр и гироскоп (DD2)
3. Аккумуляторная батарея Li-Po (G1)
4. Моторы CL-0720-12 (M1, M2, M3, M4)
5. Транзисторы IRLR8726 (VT1, VT2, VT3, VT4)

Плюсовой контакт аккумуляторной батареи G1 подключается к контакту Vin на плате Arduino Nano DD1, минусовой контакт подключается к контакту COM, так же плюсовой контакт аккумуляторной батареи подключается к плюсовым контактам моторов M1, M2, M3, M4. Минусовой контакт моторов M1, M2, M3, M4 подключается к стокам транзисторов VT1, VT2, VT3, VT4 соответственно. Истоки транзисторов VT1, VT2, VT3, VT4 подключаются к контакту COM микроконтроллера Arduino Nano DD1. Затвор транзистора VT1 подключается к контакту D2 микроконтроллера, VT2 – D3, VT3 – D10, VT4 – D11. Контакт VCC модуля GY-521 подключается к контакту +5V микроконтроллера, контакт GND модуля GY-521 подключается к

контакту COM микроконтроллера, контакты SCL и SDA модуля GY-521 подключаются к контактам A5 и A4 микроконтроллера соответственно.

Глава 7. Разработка программного обеспечения

Для работы с модулем GY-521 нужно подключить библиотеки Wire и MPU6050_tockn.

Процедура void readCommandFromSerial() позволяет устанавливать значения газа, требуемого угла крена, тангажа или рысканья, чтобы управлять коптером. При помощи процедуры void getAngles(), в глобальные переменные записываются значения углов, полученные от гироскопа и акселерометра. Так как модуль выдает значения в градусах, а модель была оптимизирована для радиан, нужно преобразовать значения, этим занимается процедура void getRadFromDegrees(). Процедура void calculateErrors() вычисляет ошибки по углам крена, тангажа, рысканья, основываясь на требуемых значениях и значениях, полученных от датчиков. Процедура PIDs() выполняет вычисление значений ПИД-регуляторов на основе полученных ошибок. Далее, выходные значения регуляторов используются для вычисления выходных значений. Это делает процедура void countMotors(), которая так же устанавливает пределы для выходных значений. Для записи полученных управляющих значений для моторов, используется процедура void writeMotors().

Заключение

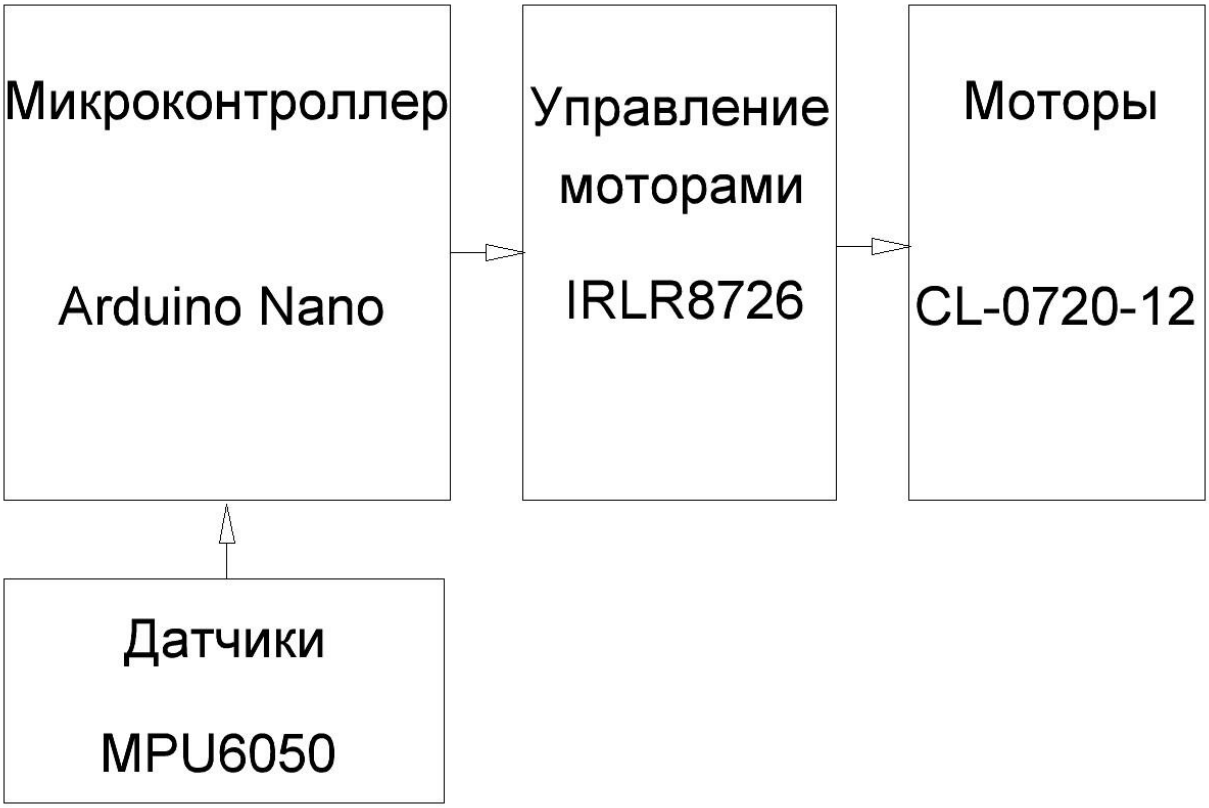
1. Математическая модель квадрокоптера была разработана и воссоздана в программе VisSim с учетом ее физических параметров.
2. Была разработана система управления, позволяющая квадрокоптеру стабилизировать углы крена, тангажа и рысканья на основе ПИД-регуляторов.
3. Была разработана принципиальная схема модуля стабилизации для мультироторных летательных аппаратов, в частном случае, для четырех роторного.
4. Была написана программа для модуля стабилизации.

Предполагается дальнейшие исследования и разработка в сторону контроля высоты при помощи барометра, использования систем навигации для ориентации и отслеживания перемещения в пространстве, использования других схем управления, других регуляторов.

Список использованных материалов

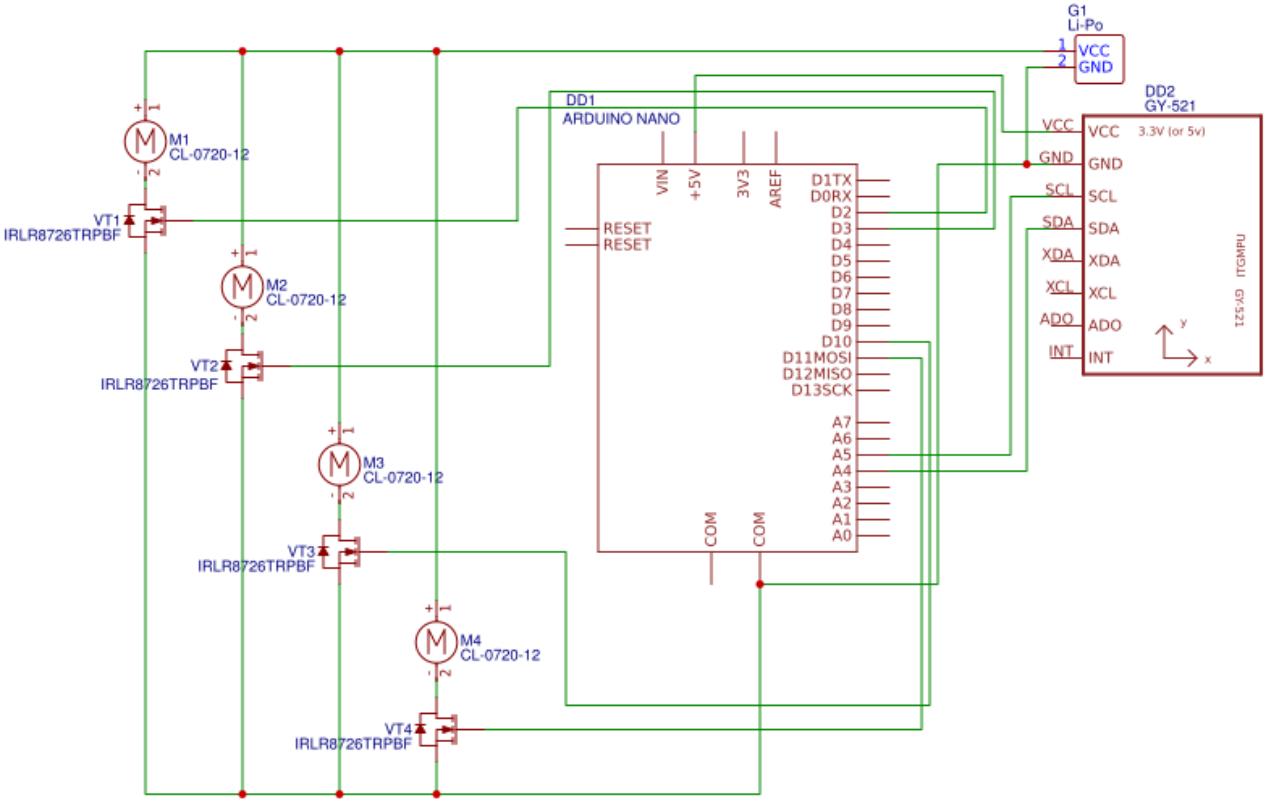
1) ГОСТ 20058-80, Основы динамики летательных аппаратов в атмосфере
2) Oscar Liang, Quadcopter PID Explaned, https://oscarliang.com/quadcopter-pid-explained-tuning/, дата обращения (16.05.18)
3) Александр Дербенёв, Автономный квадрокоптер с нуля: PID и грабли, URL https://habr.com/company/technoworks/blog/216437/, дата обращения (16.05.18)
4) Иван Ефимов, Программируем квадрокоптер на Arduino (Часть 1). URL https://habr.com/post/227425/, дата обращения (16.05.18)
5) Viswanadhapalli Praveen, Anju S. Pillai, Modeling and Simulation of Quadcopter using PID Controller, International Journal of Computer Technology and Applications, 9(15), 2016. URL

Приложение 1



Приложение 1. Функциональная схема модуля стабилизации

Приложение 2



Приложение 2. Принципиальная схема модуля стабилизации

Приложение 3

```
#include <MPU6050_tockn.h>

#include <Wire.h>

#ifdef _ESP32_HAL_I2C_H_
#define SDA_PIN 4
#define SCL_PIN 5
#endif

#define calculatedGyroOffsetX -6.65
#define calculatedGyroOffsetY -1.62
#define calculatedGyroOffsetZ -1.62

#define BAUD_RATE 19200

#define MAX_SIZE_OF_SERIAL 64

#define DEBUG false

#define ACCELEROMETER_COEF 0.1
#define GYRO_COEF 0.1

typedef enum
{
  X,
  Y,
  Z
} COORD;

int leftFront = 3, rightFront = 2, leftRear = 10, rightRear = 11;

MPU6050 mySensor(Wire, ACCELEROMETER_COEF, GYRO_COEF);
/*
  Trottle - высота  (Левый стик, вертикаль)
  Pitch - Тангаж    (Правый стик, вертикаль)
  Roll -  Крен      (Правый стик, горизонталь)
  Yaw -   Рысканье  (Левый стик, горизонталь)
*/

//delaytime
int dT = 2;

//PID's coefficients
float KpYaw =   19.04041, KiYaw =   -0.07566, KdYaw =   36.17238; //YAW   РЫСКАНЬЕ - Z
float KpRoll =  15.20834, KiRoll =  -0.00005, KdRoll =  38.19664; //ROLL  КРЕН     - X
float KpPitch = 23.07135, KiPitch = -0.01472, KdPitch = 44.36351; //PITCH ТАНГАЖ   - Y

//PID's values to storage
int PIDYaw = 0, PIDRoll = 0, PIDPitch = 0;

//Errors to PID
int errorYaw = 0, errorRoll = 0, errorPitch = 0;

//Last values of calculateErrors for derivation
float lastYaw = 0.0, lastRoll = 0.0, lastPitch = 0.0;

//Integrator values for inegration;
int integratedYaw = 0, integratedRoll = 0, integratedPitch = 0;

//Values for motors
int LF, RF, LR, RR;

//Input values from serial
int inTrottle = 0, inYaw = 0, inRoll = 0, inPitch = 0;

//Input raw values from MPU60560
```

```
float inAngleX = 0.0, inAngleY = 0.0, inAngleZ = 0.0;

void getRadFromDegrees()
{
   inAngleX *= 0.0174;
   inAngleY *= 0.0174;
   inAngleZ *= 0.0174;
}

void writeMotors()
{
  digitalWrite(leftFront,LF);
  digitalWrite(rightFront, RF);
  digitalWrite(leftRear, LR);
  digitalWrite(rightRear, RR);
}

void getAngles()
{
  inAngleX = mySensor.getAngleX();
  inAngleY = mySensor.getAngleY();
  inAngleZ = mySensor.getAngleZ();
}

void PIDs()
{
  PIDYaw = (int)(KpYaw * errorYaw + KdYaw * (errorYaw - lastYaw) / dT / 0.001 + KiYaw * (integratedYaw + errorYaw) * dT * 0.001);
  integratedYaw += errorYaw;
  lastYaw = errorYaw;

  PIDRoll = (int)(KpRoll * errorRoll + KdRoll * (errorRoll - lastRoll) / dT / 0.001 + KiRoll * (integratedRoll + errorRoll) * dT * 0.001);
  integratedRoll += errorRoll;
  lastRoll = errorRoll;

  PIDPitch = (int)(KpPitch * errorPitch + KdPitch * (errorPitch - lastPitch) / dT / 0.001 + KiPitch * (integratedPitch + errorPitch) * dT * 0.001);
  integratedPitch += errorPitch;
  lastPitch = errorPitch;
}

void calculateErrors()
{
  errorYaw = inYaw - inAngleZ;
  errorRoll = inRoll - inAngleX;
  errorPitch = inPitch - inAngleY;
}

void countMotors()
{
  LF = constrain(inTrottle - PIDPitch - PIDYaw + PIDRoll, 0, 255);
  RF = constrain(inTrottle - PIDPitch + PIDYaw - PIDRoll, 0, 255);
  LR = constrain(inTrottle + PIDPitch + PIDYaw + PIDRoll, 0, 255);
  RR = constrain(inTrottle + PIDPitch - PIDYaw - PIDRoll, 0, 255);}

void readCommandFromSerial()
{
  float angle = 0.0;
  COORD inputCoord;
  char message[MAX_SIZE_OF_SERIAL];
  int i = 0;

  for (i = 0; i < MAX_SIZE_OF_SERIAL; i++) {
    message[i] = '\0';
  }

  i = 0;
  while (Serial.available() > 0 && i < MAX_SIZE_OF_SERIAL) {
```

```
    message[i] = Serial.read();
    i++;
  }
  if (i == 0)
    return;

  if (strncmp(message, "Z", 1) == 0 || strncmp(message, "z", 1) == 0)
    inputCoord = Z;
  else if (strncmp(message, "Y", 1) == 0 || strncmp(message, "y", 1) == 0)
    inputCoord = Y;
  else if (strncmp(message, "X", 1) == 0 || strncmp(message, "x", 1) == 0)
    inputCoord = X;
  else if (strncmp(message, "T", 1) == 0 || strncmp(message, "t", 1) == 0) {
    char *pch = strtok(message, " ");
    pch = strtok(NULL, " ");
    inTrottle = atoi(pch);
    return;
  }

  char *pch = strtok(message, " ");
  pch = strtok(NULL, " ");

  angle = atof(pch);

  switch (inputCoord)
  {
  case Z:
    inYaw = angle;
    break;
  case Y:
    inRoll = angle;
    break;
  case X:
    inPitch = angle;
    break;
  default:
   break;
  }
}

void setup()
{
  Serial.begin(BAUD_RATE);
  // put your setup code here, to run once:
  // setup pins (digital: 2 in for accelerometer and gyro, 4 out for motors)
  pinMode(leftFront, OUTPUT);
  pinMode(rightFront, OUTPUT);
  pinMode(leftRear, OUTPUT);
  pinMode(rightRear, OUTPUT);

#ifdef _ESP32_HAL_I2C_H_        // For ESP32
  Wire.begin(SDA_PIN, SCL_PIN); // SDA, SCL
#else
  Wire.begin();
#endif

  mySensor.begin();

  if (DEBUG)
    mySensor.calcGyroOffsets(true);
  else
  {
    mySensor.calcGyroOffsets();
    mySensor.setGyroOffsets(calculatedGyroOffsetX, calculatedGyroOffsetY, calculatedGyroOffsetZ);
  }
}

void debugOutput(const char *param)
{
```

```
202.   if (strcmp(param, "angle") == 0)
203.   {
204.     Serial.print("\nangleX:");
205.     Serial.print(inAngleX);
206.
207.     Serial.print("\tangleY:");
208.     Serial.print(inAngleY);
209.
210.     Serial.print("\tangleZ:");
211.     Serial.print(inAngleZ);
212.     return;
213.   }
214. }
215.
216. void loop()
217. {
218.   //get input
219.   readCommandFromSerial();
220.   //get datas
221.   mySensor.update();
222.   getAngles();
223.   getRadFromDegrees();
224.   if (DEBUG){
225.     debugOutput("angle");
226.   }
227.   //calculate calculateErrors
228.   calculateErrors();
229.   //PID
230.   PIDs();
231.   //count motors
232.   countMotors();
233.   //write motors
234.   writeMotors();
235.
236.   delay(dT);
237. }
```