Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Новосибирский государственный технический университет»

Кафедра: Автоматики

ОТЧЕТ ПО ПРАКТИКЕ

Производственная (преддипломная) практика: практика по получению

профессиональных умений и опыта профессиональной деятельности
(сосредоточенная)

Направление подготовки: 270304 – Управление в технических системах

Выполнил:

Студент
Гаевский Илья Валерьевич
(Ф. И. О.)

Группа АА-47

Факультет АВТ

_____
подпись

«___» _____ 20__ г.

Проверил:

Руководитель от НГТУ Колкер Алексей Борисович
(Ф. И. О.)
Балл: _____, ECTS_____,

Оценка _____

_____
подпись

«___» _____ 20__ г.

Новосибирск 2018

Задание:

Разработать систему стабилизации мультироторными летательными аппаратами и математическую модель квадрокоптера.

Ход практики:

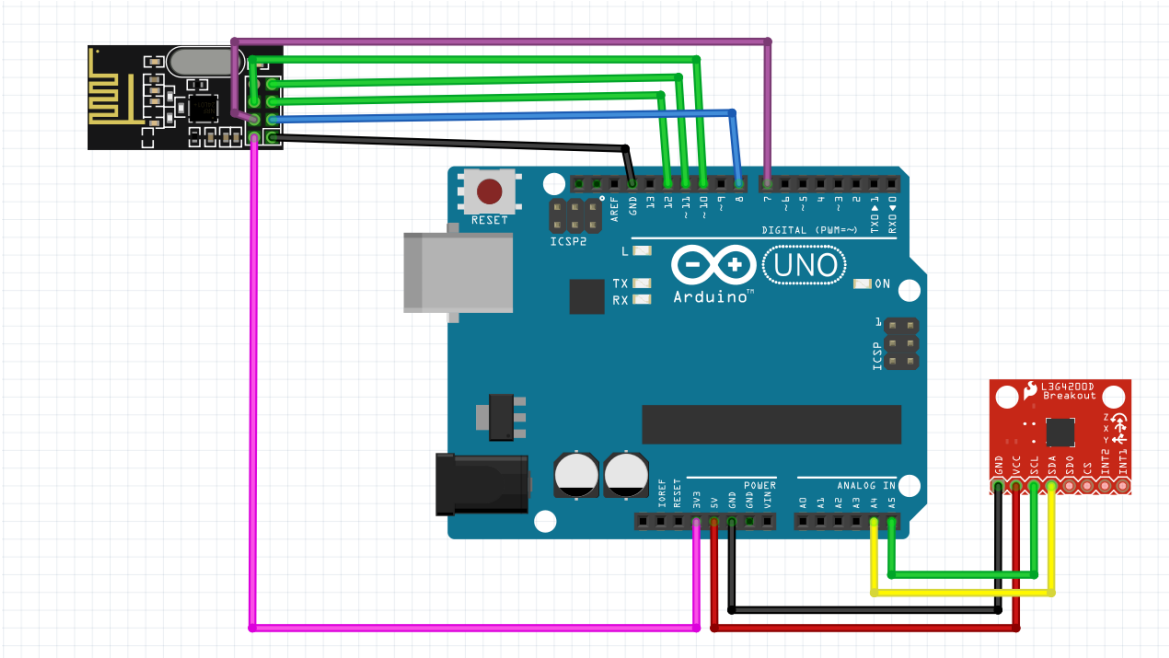

Рис.1 Структурная схема системы.

Компоненты системы:

1. nRF24L01+;
2. MPU9250;
3. Arduino Uno.

В курсовой работе был создан полетный контроллер, принимающий значения от пульта дистанционного управления на основе модуля nRF24L01+, передающий информацию по частоте 2,4 ГГц, сопряженный с Arduino по протоколу SPI. Так же полетный контроллер принимает значения от модуля MPU9250, содержащего в себе акселерометр, гироскоп, магнитометр, барометр, подключенный к Arduino по протоколу I2C. Микроконтроллер, принимает значения от пульта управления о требуемой мощности, тангажу, крену, рысканье; от гироскопа и акселерометра данные о тангаже, крене и рысканьи. На основе этих данных, микроконтроллер высчитывает правило управления и отправляет требуемые значения на моторы. Теперь же, помимо этого надо разработать математическую модель коптера.

Разработка модели

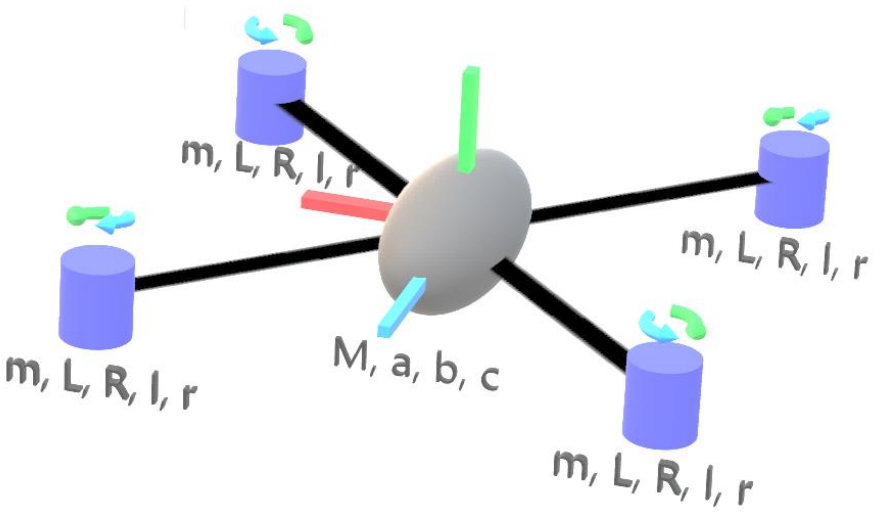

Рис.2. Общий вид коптера

Где ось Ox – голубая, Oy- красная, Oz – зеленая; M – масса тела коптера; a, b, c – параметры элипсоида (по оси Ox – b, Oy – c, Oz – a); m -масса одного двигателя, L – высота двигателя, R – радиус двигателя, 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.

Моменты инерции по осям:

Ox

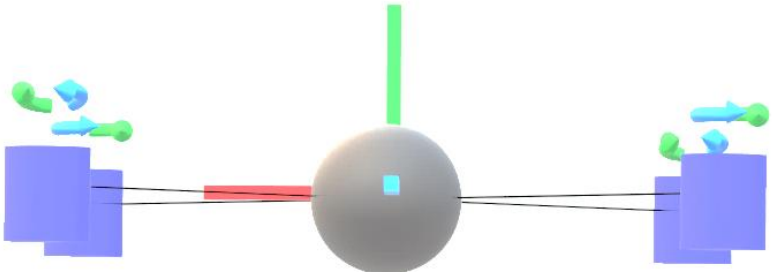

Рис. 3. Модель коптера вдоль оси Ox

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

Oy

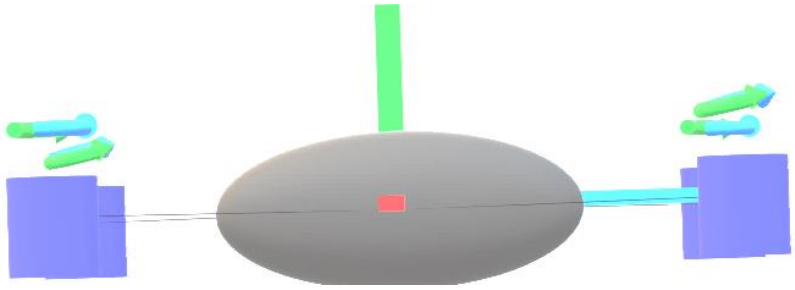

Рис. 4. Модель коптера вдоль оси Oy

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

Oz

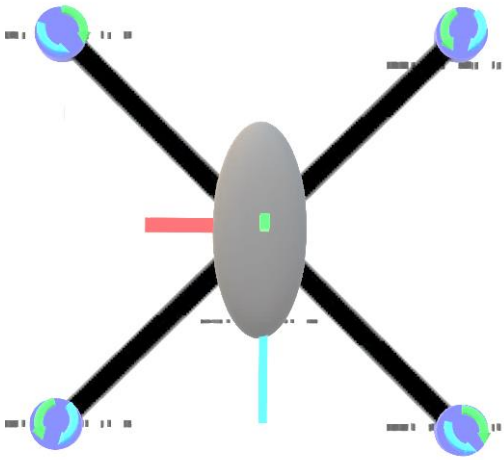

Рис. 5. Модель коптера вдоль оси Oz

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

Матрицы сил и вращения твердого тела.

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

$$\begin{cases} P_k = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ P \end{bmatrix} \\ P = \sum_{i=1}^{4} P_i = \sum_{i=1}^{4} c_{p_i} w_i^2 \\ c_{p_i} = \frac{\rho c_a S_i r_i^2}{2} \\ G = \begin{bmatrix} 0 \\ 0 \\ -(M+4m)g \end{bmatrix} \end{cases}$$	(4)-(7)

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

$$\begin{cases} \mathcal{J}_x \ddot{\phi} = M_{R_x} \\ \mathcal{J}_y \ddot{\theta} = M_{R_y} \\ \mathcal{J}_z \ddot{\psi} = M_{R_z} \end{cases}$$	(8)
$$M_R = \begin{bmatrix} M_{R_x} \\ M_{R_y} \\ M_{R_z} \end{bmatrix} = [M_q]$$	(9)
$$M_q = \begin{bmatrix} M_{q_x} \\ M_{q_y} \\ M_{q_z} \end{bmatrix} = \begin{bmatrix} (P_2 + P_3 - P_1 - P_4) l \sin 45^\circ \\ (P_2 + P_3 - P_1 - P_4 P_4) l \cos 45^\circ \\ M_2 + M_4 - M_1 - M_3 \end{bmatrix}$$	(10)
$$M_i = c_{p_i} r w^2$$	(11)
$$w_i = k V_i U_i$$	(12)

Где, M_R – матрица результирующего момента, M_q – матрица моментов от винтов kV – коэффициент преобразования напряжения в обороты для напряжений от 1В до 4.2В, где, согласно документации на мотор, идет линейная зависимость.

Реализация модели в Matlab

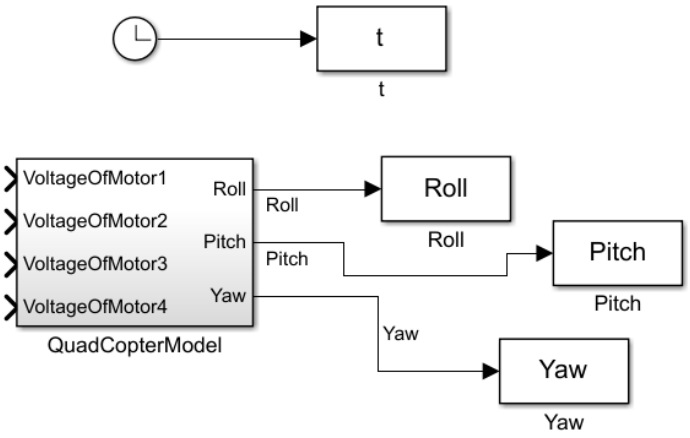

Рис.6. Модель коптера в Simulink

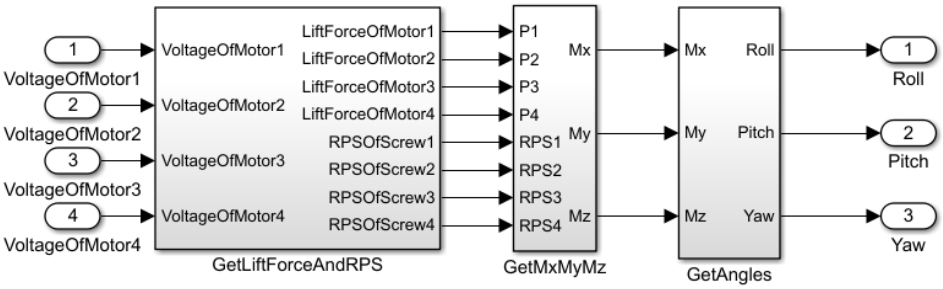

Рис.7. Структурная схема модели коптера

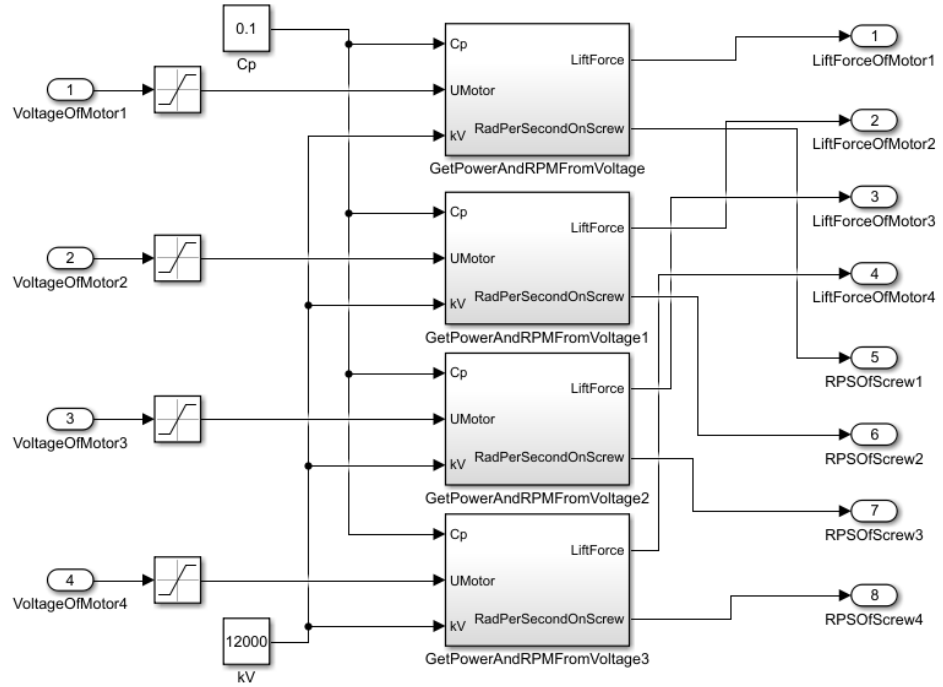

Рис.8. Блок GetLiftForceAndRPS

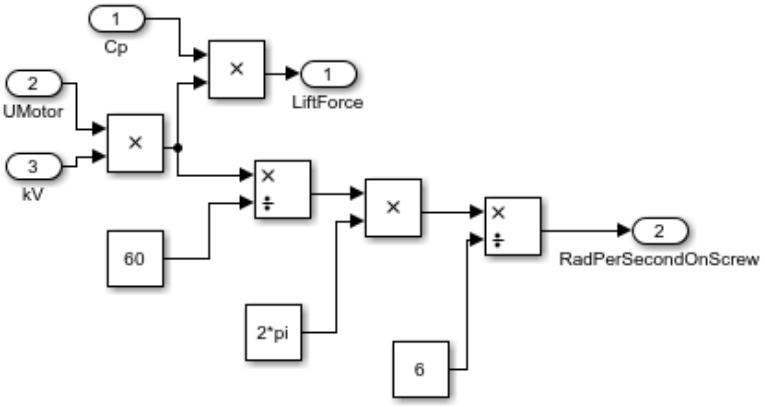

Рис.9. Блок GetPowerAndRPMFromVoltage

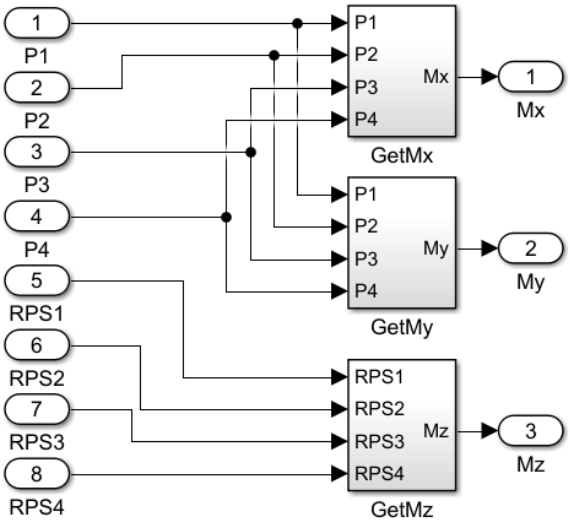

Рис.10. Блок GetMxMyMz

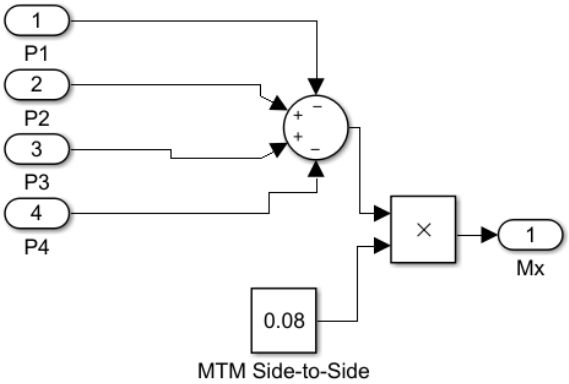

Рис.11. Блок GetMx

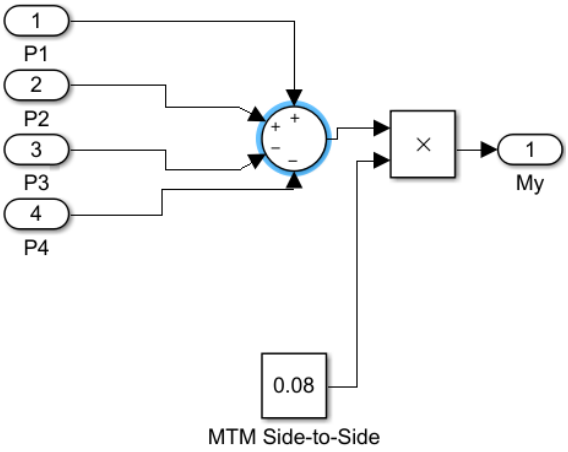

Рис.12. Блок GetMy

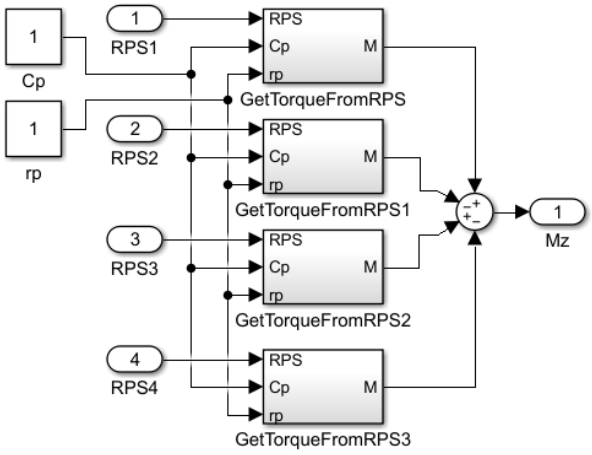

Рис.12. Блок GetMz

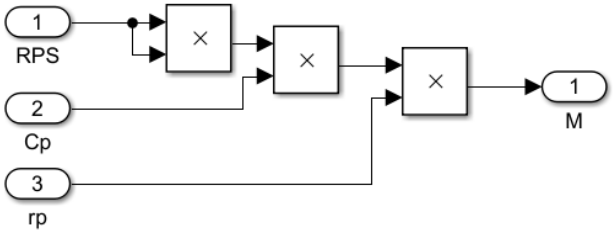

Рис.13. Блок GetTorqueFromRPS

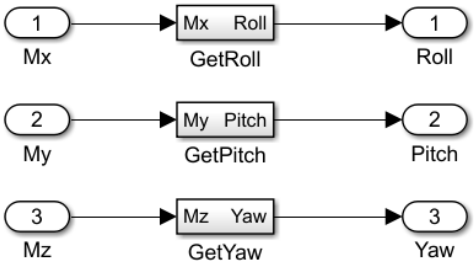

Рис.14. Блок GetAngles

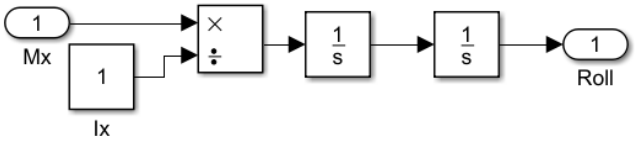

Рис.15. Блок GetRoll

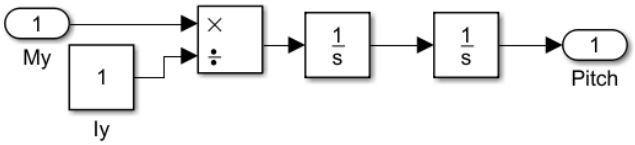

Рис.16. Блок GetPitch

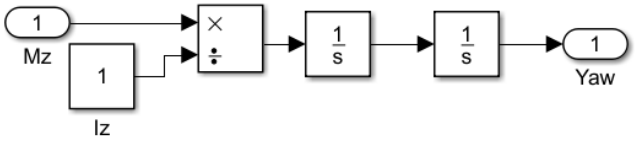

Рис.17. Блок GetYaw

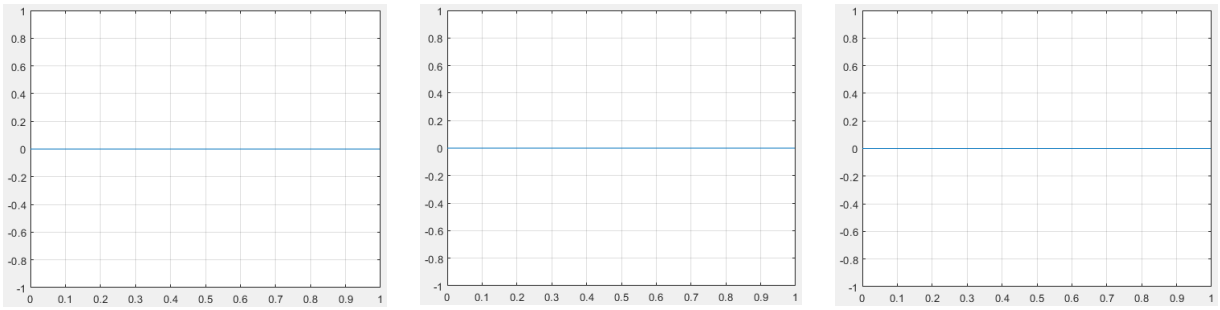

Рис.18-21. Переходный процесс при одинаковом напряжении на всех моторах (взлет)

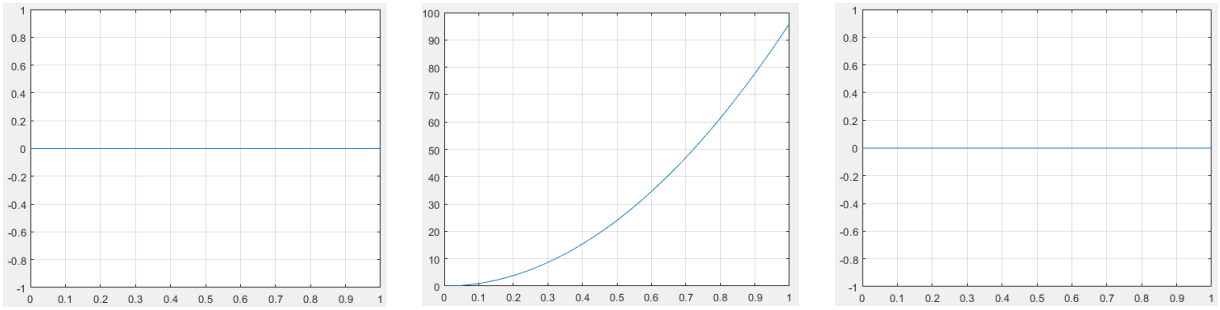

Рис.22-24. Переходный процесс при большей мощности на передних моторах (тангаж)

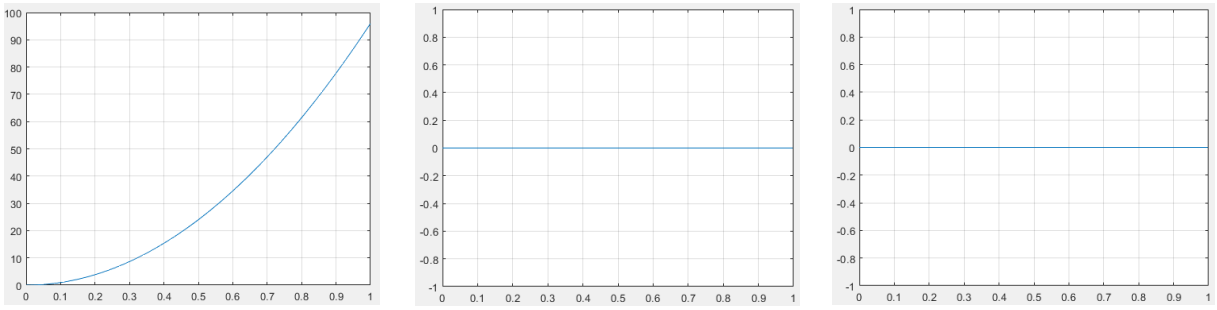

Рис.25-27. Переходный процесс при большей мощности на левых моторах (крен)

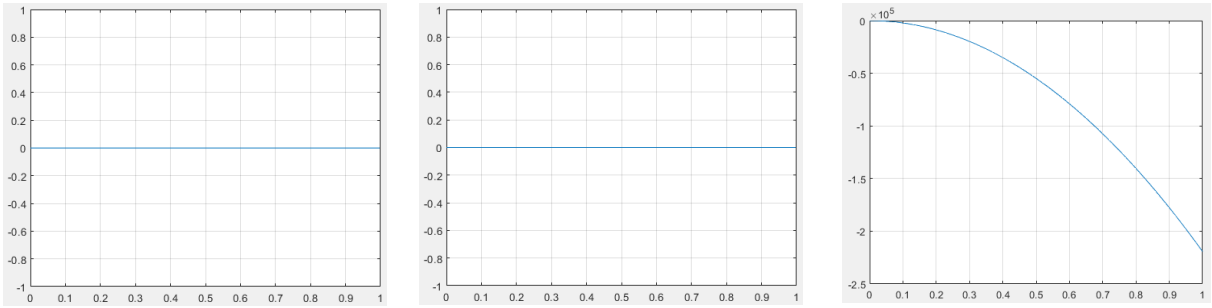

Рис.28-30. Переходный процесс при большей мощности на диагональных моторах (рысканье)

Как видно из графиков, закон управления для коптера работает оптимально.

Вычисление физических параметров коптера EACHINE e708

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

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

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

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 (7).

$(M+4m)A = P + G$	(13)

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

$A = \begin{bmatrix} A_x \\ A_y \\ A_z \end{bmatrix}$	(14)

Для того, чтобы определить подъемную силу P, выразим ее из формулы (13)

$$P = (M + 4m)A - G$$	(15)
$$\begin{bmatrix}0\\0\\P\end{bmatrix} = (M + 4m)\begin{bmatrix}A_x\\A_y\\A_z\end{bmatrix} - \begin{bmatrix}0\\0\\-(M + 4m)g\end{bmatrix}$$	(16)

При подъеме коптера вверх, результирующее ускорение направлено вдоль оси Oz, следовательно, перезапишем формулу (16).

$$P = (M + 4m)(A_z + g)$$	(17)

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.

$$c_p = \frac{P}{4w^2} = \frac{(M + 4m)(A_z + g)}{4w^2}$$	(18)

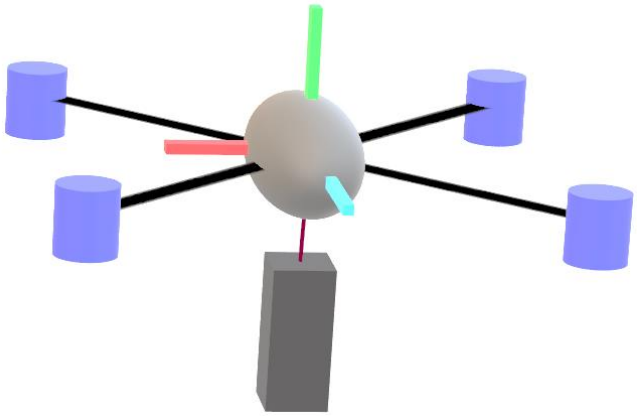

Рис.31. Схема экспериментальной установки. Черный параллелепипед – безмен, прикрепленный к столу.

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

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

Заключение:

В ходе данной работы был разработан полетный контроллер с использованием гироскопа, акселерометра, изучена работа с несколькими интерфейсами, построена математической модели квадрокоптера.

Листинг программы:

```
#include <printf.h>

#include <Servo.h>

#include <MPU6050_tockn.h>

#include <Wire.h>

#include <RF24.h>
#include <RF24_config.h>
#include <nRF24L01.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;

struct ANGLE_COMMAND
{
  COORD coord;
  float angle;
};

RF24 radio(7, 8);

Servo leftFront, rightFront, leftRear, rightRear;

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)
*/

//pipes adresses
byte addresses[6] = "1Node";

byte message[4] = {0, 0, 0, 0};

//delaytime
int dT = 2;

//PID's coefficients
float KpYaw = 1.0, KiYaw = 1.0, KdYaw = 1.0;       //YAW   РЫСКАНЬЕ - Z
float KpRoll = 1.0, KiRoll = 1.0, KdRoll = 1.0;    //ROLL  КРЕН      - Y
```

```
float KpPitch = 1.0, KiPitch = 1.0, KdPitch = 1.0; //PITCH ТАНГАЖ   - X

//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
int lastYaw = 0, lastRoll = 0, lastPitch = 0;

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

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

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

//Input raw values from Gyro
float inGyroX = 0.0, inGyroY = 0.0, inGyroZ = 0.0;

//Input raw values from Accel
float inAccX = 0.0, inAccY = 0.0, inAccZ = 0.0;

//Input raw values from AccelAngle
float inAccAngleX = 0.0, inAccAngleY = 0.0;

//Input raw values from GyroAngle
float inGyroAngleX = 0.0, inGyroAngleY = 0.0, inGyroAngleZ = 0.0;

//Input raw values from Angle
      float inAngleX = 0.0, inAngleY = 0.0, inAngleZ = 0.0;

      //Input filtered values from Gyro
      float filteredGyroYaw = 0.0, filteredGyroRoll = 0.0, filteredGyroPitch = 0.0;

      //Input filtered values from Accel
      float filteredAccYaw = 0.0, filteredAccRoll = 0.0, filteredAccPitch = 0.0;

      //Variables for Kalman's filter
      float deviationGyroYaw = 0.0, deviationGyroRoll = 0.0, deviationGyroPitch = 0.0;
    //middle deviation
      float speedGyroYaw = 0.0, speedGyroRoll = 0.0, speedGyroPitch = 0.0;
    //speed of working

      float PcGyroYaw = 0.0, PcGyroRoll = 0.0, PcGyroPitch = 0.0;
      float GGyroYaw = 0.0, GGyroRoll = 0.0, GGyroPitch = 0.0;
      float PGyroYaw = 0.0, PGyroRoll = 0.0, PGyroPitch = 0.0;

      float deviationAccYaw = 0.0, deviationAccRoll = 0.0, deviationAccPitch = 0.0;
      float speedAccYaw = 0.0, speedAccRoll = 0.0, speedAccPitch = 0.0;

      float PcAccYaw = 0.0, PcAccRoll = 0.0, PcAccPitch = 0.0;
      float GAccYaw = 0.0, GAccRoll = 0.0, GAccPitch = 0.0;
      float PAccYaw = 0.0, PAccRoll = 0.0, PAccPitch = 0.0;

      void writeMotors()
      {
        leftFront.write(LF);
        rightFront.attach(RF);
        leftRear.attach(LR);
        rightRear.attach(RR);
      }

      void getGyro()
      {
        inGyroX = mySensor.getGyroX();
```

```
134.          inGyroY = mySensor.getGyroY();
135.          inGyroZ = mySensor.getGyroZ();
136.        }
137.
138.        void getAccel()
139.        {
140.          inAccX = mySensor.getAccX();
141.          inAccY = mySensor.getAccY();
142.          inAccZ = mySensor.getAccZ();
143.        }
144.
145.        void getAccAngles()
146.        {
147.          inAccAngleX = mySensor.getAccAngleX();
148.          inAccAngleY = mySensor.getAccAngleY();
149.        }
150.
151.        void getGyroAngles()
152.        {
153.          inGyroAngleX = mySensor.getGyroAngleX();
154.          inGyroAngleY = mySensor.getGyroAngleY();
155.          inGyroAngleZ = mySensor.getGyroAngleZ();
156.        }
157.
158.        void getAngles()
159.        {
160.          inAngleX = mySensor.getAngleX();
161.          inAngleY = mySensor.getAngleY();
162.          inAngleZ = mySensor.getAngleZ();
163.        }
164.
165.        void getData()
166.        {
167.          radio.read(&message, sizeof(message));
168.          inThrottle = message[0];
169.          inYaw = message[1] - 512;
170.          inRoll = message[2] - 512;
171.          inPitch = message[3] - 512;
172.        }
173.
174.        void filterGyro()
175.        {
176.          PcGyroYaw = PGyroYaw + speedGyroYaw;
177.          GGyroYaw = PcGyroYaw / (PcGyroYaw + deviationGyroYaw);
178.          PGyroYaw = (1 - GGyroYaw) * PcGyroYaw;
179.          filteredGyroYaw = GGyroYaw * (inGyroZ - filteredGyroYaw) + filteredGyroYaw;
180.
181.          PcGyroRoll = PGyroRoll + speedGyroRoll;
182.          GGyroRoll = PcGyroRoll / (PcGyroRoll + deviationGyroRoll);
183.          PGyroRoll = (1 - GGyroRoll) * PcGyroRoll;
184.          filteredGyroRoll = GGyroRoll * (inGyroX - filteredGyroRoll) + filteredGyroRoll
    ;
185.
186.          PcGyroPitch = PGyroPitch + speedGyroPitch;
187.          GGyroPitch = PcGyroPitch / (PcGyroPitch + deviationGyroPitch);
188.          PGyroPitch = (1 - GGyroPitch) * PcGyroPitch;
189.          filteredGyroPitch = GGyroPitch * (inGyroY - filteredGyroPitch) + filteredGyroP
    itch;
190.        }
191.
192.        void filterAccel()
193.        {
194.          PcAccYaw = PAccYaw + speedAccYaw;
195.          GAccYaw = PcAccYaw / (PcAccYaw + deviationAccYaw);
196.          PAccYaw = (1 - GAccYaw) * PcAccYaw;
197.          filteredAccYaw = GAccYaw * (inAccZ - filteredAccYaw) + filteredAccYaw;
198.
199.          PcAccRoll = PAccRoll + speedAccRoll;
200.          GAccRoll = PcAccRoll / (PcAccRoll + deviationAccRoll);
```

```
201.          PAccRoll = (1 - GAccRoll) * PcAccRoll;
202.          filteredAccRoll = GAccRoll * (inAccX - filteredAccRoll) + filteredAccRoll;
203.
204.          PcAccPitch = PAccPitch + speedAccPitch;
205.          GAccPitch = PcAccPitch / (PcAccPitch + deviationAccPitch);
206.          PAccPitch = (1 - GAccPitch) * PcAccPitch;
207.          filteredAccPitch = GAccPitch * (inAccY - filteredAccPitch) + filteredAccPitch;

208.        }
209.
210.        void PIDs()
211.        {
212.          PIDYaw = (int)(KpYaw * errorYaw + KdYaw * (errorYaw - lastYaw) + KiYaw * (inte
    gratedYaw + errorYaw));
213.          integratedYaw += errorYaw;
214.          lastYaw = errorYaw;
215.
216.          PIDRoll = (int)(KpRoll * errorRoll + KdRoll * (errorRoll - lastRoll) + KiRoll
    * (integratedRoll + errorRoll));
217.          integratedRoll += errorRoll;
218.          lastRoll = errorRoll;
219.
220.          PIDPitch = (int)(KpPitch * errorPitch + KdPitch * (errorPitch - lastPitch) + K
    iPitch * (integratedPitch + errorPitch));
221.          integratedPitch += errorPitch;
222.          lastPitch = errorPitch;
223.        }
224.
225.        void calculateErrors()
226.        {
227.          errorYaw = inYaw - filteredGyroYaw;
228.          errorRoll = inRoll - filteredGyroRoll;
229.          errorPitch = inPitch - filteredGyroPitch;
230.        }
231.
232.        void countMotors()
233.        {
234.          LF = inThrottle - PIDPitch - PIDYaw + PIDRoll;
235.          RF = inThrottle - PIDPitch + PIDYaw - PIDRoll;
236.          LR = inThrottle + PIDPitch + PIDYaw + PIDRoll;
237.          RR = inThrottle + PIDPitch - PIDYaw - PIDRoll;
238.        }
239.
240.        void readCommandFromSerial()
241.        {
242.          float angle = 0.0;
243.          COORD inputCoord;
244.          char message[MAX_SIZE_OF_SERIAL];
245.          int i = 0;
246.
247.          for (i = 0; i < MAX_SIZE_OF_SERIAL; i++)
248.          {
249.            message[i] = '\0';
250.          }
251.
252.          i = 0;
253.          while (Serial.available() > 0 && i < MAX_SIZE_OF_SERIAL)
254.          {
255.            message[i] = Serial.read();
256.            i++;
257.          }
258.          if (i == 0)
259.            return;
260.
261.          if (strncmp(message, "Z", 1) == 0 || strncmp(message, "z", 1) == 0)
262.            inputCoord = Z;
263.          else if (strncmp(message, "Y", 1) == 0 || strncmp(message, "y", 1) == 0)
264.            inputCoord = Y;
265.          else if (strncmp(message, "X", 1) == 0 || strncmp(message, "x", 1) == 0)
```

```
266.           inputCoord = X;
267.
268.         char *pch = strtok(message, " ");
269.         pch = strtok(NULL, " ");
270.
271.         Serial.print("\nDEBUG READED:");
272.         Serial.println("\tmessage");
273.         Serial.println(message);
274.
275.         Serial.println("\tpch");
276.         Serial.println(pch);
277.
278.         Serial.print("\tinputCoord:");
279.         Serial.print(inputCoord);
280.         Serial.print("\tAngle:");
281.         Serial.print(angle);
282.         Serial.flush();
283.
284.         // switch (inputCoord)
285.         // {
286.         // case Z:
287.         //   inYaw = angle;
288.         //   break;
289.         // case Y:
290.         //   inRoll = angle;
291.         //   break;
292.         // case X:
293.         //   inPitch = angle;
294.         //   break;
295.         // default:
296.         //   break;
297.         // }
298.       }
299.
300.       ANGLE_COMMAND parseCommandFromSerial()
301.       {
302.         struct ANGLE_COMMAND temp;
303.         temp.coord = Z;
304.         temp.angle = 10;
305.         return temp;
306.       }
307.
308.       void setup()
309.       {
310.         Serial.begin(BAUD_RATE);
311.         // put your setup code here, to run once:
312.         // setup pins (digital: 2 in for axelerometer and gyro, 2 in transmitter, 4 ou
    t for motors)
313.         leftFront.attach(2);
314.         rightFront.attach(3);
315.         leftRear.attach(4);
316.         rightRear.attach(5);
317.
318.       #ifdef _ESP32_HAL_I2C_H_        // For ESP32
319.         Wire.begin(SDA_PIN, SCL_PIN); // SDA, SCL
320.       #else
321.         Wire.begin();
322.       #endif
323.
324.         mySensor.begin();
325.
326.         if (DEBUG)
327.           mySensor.calcGyroOffsets(true);
328.         else
329.         {
330.           mySensor.calcGyroOffsets();
331.           mySensor.setGyroOffsets(calculatedGyroOffsetX, calculatedGyroOffsetY, calcul
    atedGyroOffsetZ);
332.         }
```

```
333.
334.          radio.begin();
335.          radio.openReadingPipe(1, addresses);
336.          radio.startListening();
337.        }
338.
339.        void debugOutput(const char *param)
340.        {
341.          if (strcmp(param, "angle") == 0)
342.          {
343.            Serial.print("\nangleX:");
344.            Serial.print(inAngleX);
345.
346.            Serial.print("\tangleY:");
347.            Serial.print(inAngleY);
348.
349.            Serial.print("\tangleZ:");
350.            Serial.print(inAngleZ);
351.            return;
352.          }
353.        }
354.
355.        void loop()
356.        {
357.          // put your main code here, to run repeatedly:
358.          //Get Datas
359.
360.          if (radio.available())
361.            getData();
362.
363.          //if (DEBUG)
364.          //{
365.          readCommandFromSerial();
366.          //}
367.
368.          mySensor.update();
369.          getGyro();
370.          getAccel();
371.          getGyroAngles();
372.          getAccAngles();
373.          getAngles();
374.
375.          //debugOutput("angle");
376.
377.          //filter in
378.          filterGyro();
379.          filterAccel();
380.          //calculate calculateErrors
381.          calculateErrors();
382.          //PID
383.          PIDs();
384.          //count motors
385.          countMotors();
386.          //write motors
387.          writeMotors();
388.
389.          delay(dT);
390.        }
```

Список литературы:

1) https://www.lucidar.me/en/inertial-measurement-unit/mpu-9250-and-arduino-9-axis-imu/
   Описание работы с модулем MPU9250
2) http://docs.cntd.ru/document/gost-20058-80
   Основы динамики летательных аппаратов в атмосфере
3) https://geektimes.ru/post/258196/
   Создание квадрокоптера