Содержание.

Страница

1 Постановка задачи. 2

2. Квадрокоптер, как объект управления. 3

3. Уравнения ЛА. Возможности управления ЛА четырехроторной схемы. 4

4. Задача плоского движения. Передаточные функции. 7

4.1 Каналы крена, тангажа высоты, . 7

4.1.1 Уравнения движения. 7

4. 1.2 Линеаризация уравнений движения . 8

4. 1.3 Показания датчиков. 9

4. 1.4 Линеаризация уравнений датчиков. 9

4. 1.5 Построение системы стабилизации. 10

4.2. Курс 13

4.3. Выводы по системе стабилизации. 14

Применение решения к системе пространственного движения.

5. Интегрирование уравнений движения. Применение кватернионов. 15

6. Схема системы стабилизации углового положения. 17

7. Макета ЛА. 18

7.1 Параметры макета. 19

7.2. Общая схема системы. 19

7.3 Програмные средства 19

7.4 Описание контрольной системы 20

7.5 Текст программы 21

8. Итоги работы и выводы, направление дальнейшей работы. 28

Приложения. 29

Список использованной литературы. 29

**1. Постановка задачи.**

В текущей работе решается задача построения системы стабилизации четырехроторного ЛА вертолетного типа (квадрокоптер), а также применения таковой системы к действующему макету ЛА.

Для решения задачи стабилизации предполагается, что на борту ЛА находится микромеханические датчики кажущихся ускорений и угловых скоростей. Также возможно применение дополнительных датчиков.

Практическая часть состоит в сборке, и применении и доводке полученных уравнений СУ к реальной СУ макета ЛА.

**2.Квадрокоптер, как объект управления.**

Квадрокоптер – это летательный аппарат с четырьмя роторами, вращающимися диагонально в противоположных направлениях (рис. 1).

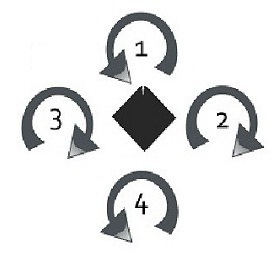


Рис.2

Каждый ротор приводится в движение собственным двигателем, что позволяет управлять тягой каждого двигателя по отдельности.

Преимуществом четырехроторной схемы является простота управления. По сути, все эволюции ЛА можно описать при помощи четырёх простых манёвров: манёвра «взлёта и снижения», двух маневров «наклона» (иначе говоря, по крену и тангажу) и маневра «поворота»(курс). (см. рис.2).

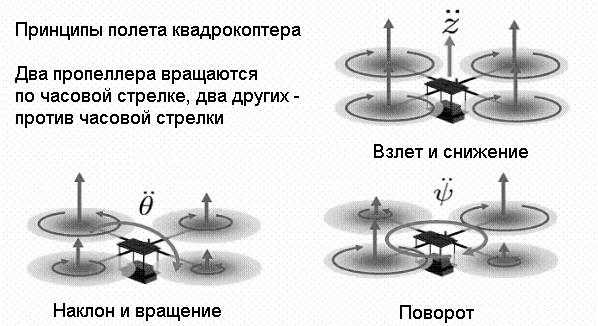


Рис.3

Как в дальнейшем будет видно из анализа уравнений, эти манёвры находятся в суперпозиции друг относительно друга, в том смысле, что задавать ускорения аппарату через систему управления можно независимо по каждому из четырех каналов.

Как правило и в настоящей работе, четырёхроторная схема рассматривается как схема для радиоуправляемого или беспилотного ЛА относительно небольших размеров, имеющего СУ на основе БИНС на борту.

**3. Уравнения ЛА. Возможности управления ЛА четырехроторной схемы.**

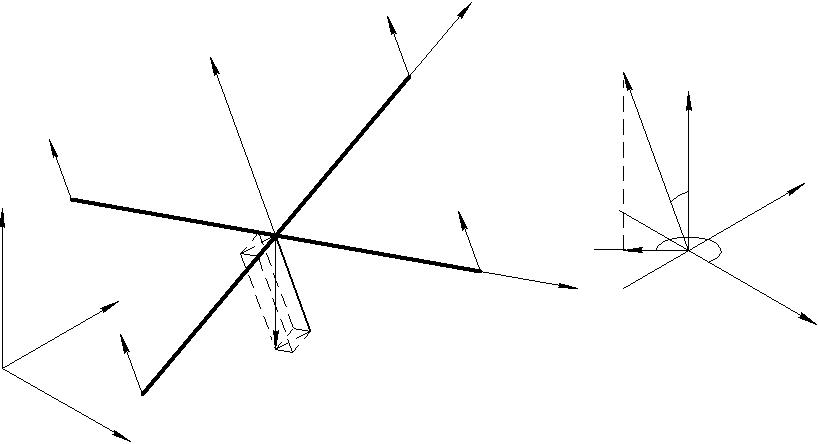


Рис.4

В связанной СК уравнения движения могут быть записаны как

, где – коэффициент связи силы тяги пропеллера с создаваемым им боковым моментом,

А силы и моменты обусловлены влиянием аэродинамических эффектов (в основном, сопротивление воздуха).

При этом в земной системе уравнения линейных координат запишутся как:

, где углы задают угловое положение вертикали аппарата.

Варьируя эту систему уравнений в соответствии с рассматриваемыми режимами (см.Рис.3) от положения равновесия,

и пренебрегая сопротивлением воздуха, получаем:

Режим «Взлет, снижение» :

Крен, тангаж (вокруг оси x, аналогично для у):

Курс :

Откуда видно, что СУ имеет возможность вносить ускорения по любой из осей независимо от других, что существенно упрощает исследование ЛА и управление им. Фактически, вносить ускорение по нескольким каналам одновременно можно простым суммированием управляющих сигналов.

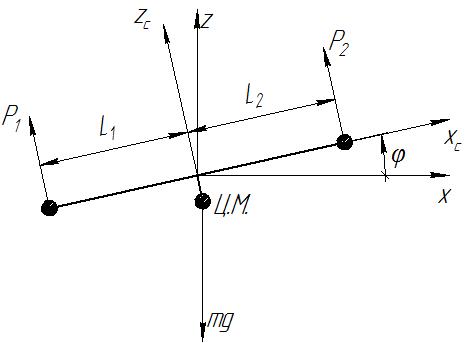
**4.Задача плоского движения. Передаточные функции.**

**4.1 Крен, тангаж.**

Основной проблемой данной задачи является решение задачи стабилизации углового положения вокруг горизонтальных осей по каналам крена и тангажа (момент вокруг произвольной горизонтальной оси задается суммой моментов по каналам крена и тангажа). Чтобы понять возможности по стабилизации аппарата в этих каналах рассмотрим упрощенную плоскую задачу. Также, применительно к этой задаче рассмотрим вопрос стабилизации высоты.

**4.1.1 Уравнения движения.**

Рассмотрим задачу плоского движения:

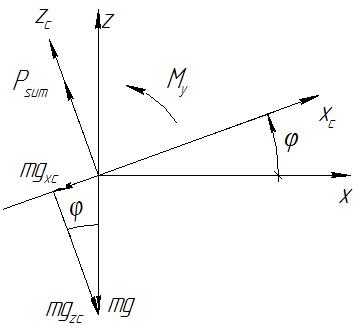


Приводя вектора сил к центру масс. Получаем:

Введем обозначения:

**(Земная система координат)**:

**(Собственная система координат)**:



**Матрицы пересчета:**

**4.1.2 Линеаризация уравнений объекта.**

Линеаризация уравнений объекта проводится вокруг балансового режима:

Из первого условия также следует, что

Рассматривая систему в предположении малости угла , уравнения динамики принимают вид:

Ось :

Запишем тягу через отклонение:

Рассматривая уравнение отклонения от балансового режима, можно записать:

Угол :

Ось :

**4.1.3 Показания датчиков:**

Кажущиеся ускорения (показания акселерометра):

Подставляя уравнения динамики, получаем:

**Уравнение датчика расстояния:**

В силу того, что ось датчика расстояния выставлена по , а не по .

**4.1.4 Линеаризация уравнений датчиков**:

Из уравнения , в силу малости углов:

Записав уравнения акселерометра в отклонениях, от балансового значения , получаем

Уравнение гироскопа остаётся таким, как и для нелинеаризованной системы:

Показания акселерометра по оси (уравнение (4.2)) также не требуют линеаризации:

Имеет смысл рассмотреть , как функцию угла , так как показания , как выясняется, позволяют уточнить угол.

Да**тчик расстояния.**

Линеаризованное уравнение датчика расстояния можно записать как.

, т.к. после линеаризации .

**4.1.5.** **Построение системы стабилизации.**

Т.к. . При построении системы стабилизации эти величины полагаются равными, а индексы собственной системы опущены.

Тогда, для построения системы стабилизации интересны передаточные функции:

,а также уравнения кинематики:

,и уравнения опорного режима:

Задача системы стабилизации – поддержание заданных переменных:

Замыкание СУ выполняется в виде:

При этом

, где в и учитывается динамика исполнительных органов.

При анализе, эти функции можно принять равными 1 (без учета динамики двигателей).

Из уравнений видно, что каналы управления по высоте и углу наклона не связаны друг с другом и могут рассматриваться отдельно.

**Канал стабилизации высоты.**

Схема СУ по каналу z выглядит следующим образом, где – ПФ регулятора.



Пусть ,

Тогда передаточные функции системы:

Из второй и третьей передаточных функции видно наличие ненужной статической ошибки. Для ее ликвидации можно ввести интегральную коррекцию в главную обратную связь.

Пусть

Тогда

Статическая ошибка ликвидирована.

Для системы можно записать условие устойчивости по критерию Гурвица ():

Или

Если же

То есть постоянная времени интегрирования должна превышать постоянную времени системы.

Коэффициент , находясь при старшей степени знаменателя позволяет влиять на скорость переходного процесса.

**Канал стабилизации углового положения.**

Система стабилизации по каналу углового положения.

Коррекция

*\_*



, где – момент связанный с неравностью плеч тяг двигателей и характеристик движков.

В системе явно прослеживаются 2 контура – быстрой коррекции и медленной доводки.

Пусть , (быстрое движение)

Тогда передаточные функции системы:

Передаточная функция имеет астатизм в первой степени. Чтобы избавится от него и статической ошибки, необходимо в главной обратной связи ввести двойную интегральную коррекцию.

, тогда

Из условия устойчивости:

Передаточные функции при подключении получаются сложные и затруднены для анализа, однако в силу того, что это движение медленное, его можно анализ можно опустить.

Интегральные составляющие имеют свойство ухода и накопления ложной информации, т.к. не имеют обратной связи непосредственно по контролируемому углу.

В силу этого, вместо непосредственного суммирования предлагается коррекция интегральных составляющих W1 через комплементарный фильтр:

(величины взвешенные и безразмерные, )

**4.2 Угол курса. Плоская задача.**

Движение при решении плоской задачи описывается уравнением

В отклонениях от балансового режима:

, уравнения имеют вид (с учетом уравнений маневра )

Уравнение гироскопа

Выполним замыкание в виде



Вводя , получаем

Наличие статической ошибки для данного канала несущественно.

**4.3 Выводы по системе стабилизации. Применение результатов к общей задаче движения.**

Анализ уравнений движения при решении плоской задачи показывает, что для стабилизации углового положения по каналам наклона (в частности, тангажа и крена) необходимо знать угловые скорости, углы наклона аппарата, а также интегралы углов. При решении плоской задачи углы наклона можно было высчитывать непосредственно путем интегрирования угловых скоростей. Однако, при решении задачи пространственной навигации, углы, в общем случае, не являются интегралами угловых скоростей (Попытка применения непосредственно интегрирования приводит к существенному уходу в течении нескольких секунд). Интегрирование уравнений движения требует применения методов интегрирования кинематических уравнений. В текущей работе для решения этой задачи привлекается формализм кватернионов.

**5. Интегрирование уравнений движения:**

Стабилизация положения ЛА требует знания текущих углов наклона аппарата относительно плоскости горизонта и координат. При использовании БИНС классическая схема интегрирования уравнений движения для БИНС выглядит следующим образом:

(схема)

Рис.6

Блок интегрирования кинематических уравнений может быть выполнен при помощи разных математических методов. Для текущей работы выбран кватернионный метод, как вычислительно простой и отвечающей физике ЛА.

**Уравнения углового движения в кватернионах:**

*, где –угловые скорости в проекции на оси координат, а -кватернионные параметры.*

Или, в дискретном варианте,

С последующей нормализацией уравнений.

Переход от параметров в кватернионах к угловым параметрам :

Помня, что . Можно в предположении малости углов отклонения выполнить линеаризацию .

Тогда, кватернионные параметры по сути превращаются в проекции угла поворота на соответствующие им оси связанной системы, то есть именно те параметры, которые требуются для системы стабилизации ЛА.

Такой переход, однако ухудшает характеристики системы, т.к. при больших углах система начнет терять чувствительность.

**Применение результатов к действующей модели.**

Применение мат модели к реальной системе требует уточнений в связи с неидеальностью гироскопических датчиков.

Шум гироскопов проходит через канал стабилизации по угловой скорости, влияя на работу регуляторов и двигателей. В силу не идеальности системы, это приводит к потере чувствительности и колебаниям. Чтобы не допустить последнего, обратная связь по угловой скорости должна быть подвергнута фильтрации. Вводится фильтр нижних частот, отсекающий шумовую составляющую.

Введения фильтра показаний гироскопа по остальным каналам не приводит к существенному улучшению параметров (интегральные составляющие сами по себе имеют свойства фильтров), и более того уменьшают время стабильной работы, так как уход системы увеличивается в силу потери информации в системе фильтрации.

**6.Схема системы стабилизации углового положения.**



***=****(1,0,0,0)*

*Коррекция*

Блок суммирования сигналов (S –сигналы на управляющие органы):

Исполнительные органы:

Ранее управление задавалось непосредственно в виде сил и моментов, но на практике это недостижимо.

Исследуем систему регулятор-двигатель-пропеллер в первом приближении.

*,где – количество оборотов, при этом.*

*Аппроксимируя апериодическим звеном и линеаризуя зависимость тяги от скорости получаем:*

Таким образом, исполнительные органы обладают инерционностью и непостоянным коэффициентом усиления, зависящим от параметров атмосферы и выбранного режима.

**7. Макет ЛА.**

Для исследования и демонстрации работы СУ собран макет ЛА.



Состав макета:

- Рама четырехлучевая (материал нэйлон) x1

-Двигатели A28M 1100Kv Brushless Outrunner Motor x4

-Регуляторы Marcus point 20 x4

- контроллер KK Controller Board Version 5.5 (переработан) x1

- Блок ИИНС MPU-6050 Module 3 Axis Gyroscope+Acce​​lerometer. x1

- Wifi роутер TPlink MP3020 x1

- Ультразвуковой дальномер x1

- Плата разводки питания x1

Контроль и управление осуществляются при помощи:

Персонального компьютера (ОС Fedora v18.0) и геймпада.

**7.1 Параметры макета.**

Пренебрегая массой всех частей, кроме рамы и двигателей получаем.

Существенна также масса аккумулятора, но аккумуляторы могут меняться.

**7.2 Общая схема системы.**



**7.3 Программные средства***.*

Для исследования системы составлена тестовая программа, решающая задачи угловой стабилизации и передачи телеметрии. Также протестированы возможности проведения плавной посадки и проведены другие тесты.

Управляющая программа полетного контроллера выполнена на кооперативном диспетчере библиотеки «MirmOS.h».

Библиотека MirmOS построена на базе функций Arduino IDE, использует службу времени и некоторые другие функции и процедуры из библиотек Arduino.

В текущей работе используются функции кооперативного диспетчера, службы времени Arduino. Запланированно использование интерпретатора ОС для передачи команд от контролирующих систем или для связи вычислительных устройств.

Программа работает в режиме кооперативной могозадачности.

Объявление задач выполняется функцией SH.newtask((void\*)task[,int delay]);

Это процедура добавляет задачи в стек кооперативного диспетчера и позволяет вводить задержку перед вызовом процедуры.

***7.4 Описание контрольной системы.***

Главный контроллер (ГлК) поддерживает связь с ПК по каналу Wifi сети.Связь выполнена на основе протокола IP. Передача данных телеметрии от контроллера к ПК ведется по протоколу UDP, а команд управления от ПК к ГлК по каналу TCP.

ПК работает под управлением операционной системы Linux (Fedora 18). Тексты программ приводятся в приложении.

Проработана возможность подключения пульта управления к системе.

***7.5 Текст программы.*** *(Контроллер KKboard работает на чипе atmega168) компилятор avr-gcc , тексты библиотечных функций в приложении на CD.*

Функция инициализации setup().Функция входа в программу programm(). (см. файл main\_prog.cpp библиотеки MirmOS в приложении).

Объявление библиотек, переменных:

#define KK

#include "MirmOS.h" // Библиотека ОС

#include "UltrasonicKKmod.h" // Библиотека взаимодействия с дальномером

#include "Wire.h" //

#include "I2Cdev.h" // Библиотеки для работы с щиной I2c

#include "MPU6050.h" // и блоком акселерометров.

MPU6050 accelgyro;

int16\_t ax, ay, az;

int16\_t gx, gy, gz,gy2;

#define startKvat 0x0FFF

long l0=startKvat,l1=0,l2=0,l3=0;

long l0n,l1n,l2n,l3n;

unsigned int temp[4];

char servN,servNflag;

unsigned int serv[4];

long intax,intaz,intdist,intgy,intgx,intgz;

int servsumm=0,servsumm2;

int servdiff\_y=0,servdiff\_x=0,servcomp\_z=0;

int startgx,startgy,startgz,startaz,startax,startay;

long intphi\_x,intphi\_y,intphi\_z;

long timerestart;

char flag\_restart\_gyr=1;

#define ALTITUDE 230

#define TIMCORRECT 2000

int servnum[4];

int tempserv;

char tempnum;

unsigned int s[4];

#define MOTOR1 5

#define MOTOR2 6

#define MOTOR3 7

#define MOTOR4 8

#define TIMESTOP 20000

#define STARTMINUS 250\*16

#define Kc 1.4

#define Kd 1.00

#define TKK 0.5

#define Kax 0.0005\*4

int dist=0,edist;

int intazop=14500;

coopShedulerType SH;

long req;

// sensor connected to:

// Trig - 2, Echo - 3

Ultrasonic ultrasonic(2, 3);

int flag=0;

**Функция инициализации (вызывается первой)**

1. void setup() {
2. TCCR1A=(0<<COM1A1) | (0<<COM1A0) | (0<<COM1B1) | (0<<COM1B0) | (0<<WGM11) | (0<<WGM10);
3. TCCR1B=(0<<ICNC1) | (0<<ICES1) | (0<<WGM13) | (0<<WGM12) | (0<<CS12) | (0<<CS11) | (1<<CS10);
4. TIMSK1=1;
5. Wire.begin();
6. accelgyro.initialize();
7. Serial.begin(38400);
8. attachInterrupt(1, startdur, HIGH);
9. //Serial.write(accelgyro.testConnection() ? "MPU6050 connection successful\n" : "MPU6050 connection failed\n");
10. }

2-10 – инициализация таймера 1.

5-9 – инициализация библиотек.

8 Назначение прерывания, работающего с сонаром. (В качестве обработчика функция startdur)

**Точка старта основной программы:**

1. void programm()
2. {
3. SH.start(idle,intask);
4. }

3 – Запуск кооперативного диспетчера с функцией простоя ядра idle и функцией инициализации intask.

**Функция простоя ядра.**

Содержит подпрограмму, принимающую команды от главного контроллера.

Тестовый вариант.

1. char data;
2. char strrest[]="gyrcorrect";
3. char strdvstart[]="startdv";
4. char strdvstop[]="stop";
5. char nchargyr=0;
6. char ncharstart=0;
7. char ncharstop=0;
8. void idle()
9. {
10. if (Serial.available()) {
11. data=Serial.read();
12. if (data == strrest[nchargyr]) {nchargyr++;}
13. else {nchargyr=0;}
14. if (data == strdvstart[ncharstart]) {ncharstart++;}
15. else {ncharstart=0;}
16. if (data == strdvstop[ncharstop]) {ncharstop++;}
17. else {ncharstop=0;}
18. if (nchargyr==10) SH.newTask(restart);
19. if (ncharstart==7) SH.newTask(startdv);
20. if (ncharstop==4) SH.newTask(stoptask2);
21. }
22. }

10-11 считывание символов

12-20 проверка на соответствие принимаемой информации имеющимся командам.

Можно запускать задачи:

18 – тарировки гироскопов.

19 - включения тестовой программы

20 – отключения двигателей, завершение программы

**Функция, запускающая торировку гироскопов:**

1. void restart()
2. {timerestart=millis();
3. PORTB=PORTB&(~(1<<6));
4. delay(200);
5. PORTB=PORTB|(1<<6);
6. SH.newTask(gyrostart);
7. flag\_restart\_gyr=1;
8. }

3-5 - мигнуть светодиодом

6 - функция старта гироскопов

7 - установить флаг режима торировки

**Старт тестовой программы:**

1. void startdv ()
2. {
3. intdist=0;
4. edist=0;
5. intaz=0;
6. flag=1;
7. t1();
8. SH.newTask(task1);
9. req=millis();
10. SH.newTask(stoptask,TIMESTOP);
11. }
12. // Мигание светодиодом
13. void t1(){PORTB=PORTB|(1<<6);if (flag==1) SH.newTask(t2,100\*TKK);}
14. void t2(){PORTB=PORTB&(~(1<<6));SH.newTask(t1,100\*TKK);}

3-5 - сброс переменных

6 - установить флаг работы основной программы

7 - функция, производящая мигание светодиодом, пока установлен флаг работы

8 - Вызов функции выдачи сигналов на двигатели.

10 - Остановить работу тестовой программы через TIMESTOP милисекунд.

**Функции остановки:**

1. void stoptask2()
2. {flag=0;}
3. void stoptask()
4. {if ((millis()-req)>TIMESTOP-1000) flag=0;}

2 - отключить работу тестовой программы.

4 - Тоже самое, но с проверкой, не допускающей срабатывание подпрограммы остановки на неродном вызове процесса.

**Функция инициализации диспетчера:**

1. void intask()
2. {
3. SH.newTask(restart);
4. SH.newTask(taskPrint);
5. SH.newTask(sonar);
6. }

3 – Запуск торировки гироскопов.

4 – Запуск сервера телеметрии.

5 – Запуск подпрограммы работы с сонаром.

**Подпрограмма для работы с сонаром.**

1. void sonar()
2. { ultrasonic.Timing();
3. edist=150;
4. intdist=intdist+edist;
5. SH.newTask(sonar,10);
6. }

2. Подать сигнал на сонар.

3-4 Интегрирование расстояния, полученного сонаром (в текущей версии расстояние задано константой)

5 Задержка на повторный вызов.

**Обработчик прерывания сигнала сонара.**

1. void startdur()
2. {int dur=(unsigned long)TCNT1 - (unsigned long)ultrasonic.starttimer;
3. if (!(dur<5000)) ultrasonic.Duration(dur);
4. }

Задача обработчик вызывается по фронту соответствующего импульса, сохраняет показания сонара в переменной ultrasonic.duration

**Функция инициализации гироскопов и системы интегралов. Задание НУ.**

1. // Гироскоп. Исходная, начальная, текущая информация. Время коррекции (TIMCORRECT) определено в define.
2. int br=1;
3. int k=1;
4. int n=0;
5. void gyrostart()
6. {PORTB=PORTB|(1<<6);
7. flag=0;
8. accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
9. intgx=0;
10. intgy=0;
11. intgz=0;
12. intaz=0;
13. intphi\_x=-70/Kintphi\_x;
14. intphi\_y=-0/Kintphi\_y;
15. intphi\_z=-150/Kintphi\_z;
16. n=0;
17. SH.newTask(gyrcorrect,100);
18. }

6-16 задание НУ интегралам системы.

17 – вызов функции поиска нуля гироскопов.

**Функция тарировки гиродатчиков.**

void gyrcorrect()

1. {accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
2. intgx=intgx+gx;
3. intgy=intgy+gy;
4. intgz=intgz+gz;
5. intaz=intaz+az;
6. n++;
7. if (millis()-timerestart<TIMCORRECT) SH.newTask(gyrcorrect,1);
8. else{
9. startgx=intgx/n;
10. startgy=intgy/n;
11. startgz=intgz/n;
12. startaz=intaz/n;
13. l0=startKvat;
14. l1=0;
15. l2=0;
16. l3=0;
17. intgx=0;intgy=0;intgz=0;intaz=0;
18. SH.newTask(kvaternion);
19. flag\_restart\_gyr=0;}
20. }

1-7 – интегрирует показание датчиков в течении некоторого времени.

8-19 – По истечении некоторого времени находит среднее, определяя его за ноль датчиков (полагается, что в течении процедуры аппарат стоит на земле).

18 – запуск кватернионной системы отслеживания углового положения.

**Функция, отвечающая за интегрирование кинематических уравнений.**

1. long w1;long w2; long w3;
2. double filtrgx,filtrgy,filtrgz,filtraz;
3. #define Kvatnorm 1/21278/2 \*10 //?
4. #define h 0.002
5. long z;
6. double norm;
7. void kvaternion()
8. {accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
9. filtraz=az-(az-filtraz)\*0.75;
10. filtrgx=gx-(gx-filtrgx)\*Kf;
11. filtrgy=gy-(gy-filtrgy)\*Kf;
12. filtrgz=gz-(gz-filtrgz)\*Kf;
13. az=filtraz;
14. gx=filtrgx;
15. gy=filtrgy;
16. gz=filtrgz;
17. w1=gx-startgx;w2=gy-startgy;w3=gz-startgz;
18. l0n=l0+(-l1\*w1-l2\*w2-l3\*w3)\*Kvatnorm\*h;
19. l1n=l1+( l0\*w1-l3\*w2+l2\*w3)\*Kvatnorm\*h;
20. l2n=l2+( l3\*w1+l0\*w2-l1\*w3)\*Kvatnorm\*h;
21. l3n=l3+(-l2\*w1+l1\*w2+l0\*w3)\*Kvatnorm\*h;
22. norm=sqrt(l0n\*l0n+l1n\*l1n+l2n\*l2n+l3n\*l3n)/startKvat;
23. l0=l0n/norm;
24. l1=l1n/norm;
25. l2=l2n/norm;
26. l3=l3n/norm;
27. intaz=intaz\*0.975+(az-startaz);
28. //z=z+intaz;
29. intgx=l1;
30. intphi\_x=intphi\_x+intgx;
31. intgy=l2;
32. intphi\_y=intphi\_y+intgy;
33. intgz=l3;
34. intphi\_z=intphi\_z+intgz;
35. if (!flag\_restart\_gyr) SH.newTask(kvaternion,2);
36. }

9-12 уравнения фильтров.

18-21 уравнения приращений кватернионов.

22-26 нормирование кватернионов

27-34 Вычисление второго интеграла, задающегоо поправку на момент возмущения.

35 повторный вызов процедуры, если не поставлен флаг новой торировки

**Обработчик таймера точной службы времени.**

1. ISR(TIMER1\_COMPA\_vect)
2. { TIMSK1=0;
3. digitalWrite(servN+5,0);
4. servN++;
5. if (servN!=4) SH.newTask(impulse);
6. }

Задача таймера – контроль за длительностью выдаваемых сервоимпульсов.

4-5 смена контролируемого импульса.

Импульсы выдаются по очереди.

**Функция настройки таймера 1 на выдачу импульса необходимой длительности.**

1. void impulse (void)
2. {
3. digitalWrite(servN+5,1);
4. OCR1A=TCNT1+serv[servN];
5. TIFR1=TIFR1 |(1<<OCF1A); // Очистка флага прерывания.
6. TIMSK1=(0<<ICIE1) | (0<<OCIE1B) | (1<<OCIE1A) | (0<<TOIE1);
7. }

Установка сигнала по каналу servN и настройка параметров таймера, прерывание которого отключит необходимы сигнал.

**Таблица коэффициентов регуляторов системы.**

1. #define K\_idist (0.02\*3.86\*0.05)\*16
2. #define K\_pdist (0.01\*3.86)\*16
3. #define K\_iaz (-2\*0.001)\*16
4. #define Kf 0.2
5. #define K\_y 0.05 \*0.6\*4
6. #define Kp\_y (K\_y /8 \*3 )
7. #define Ki\_y (K\_y \*0.002 \*1000 \*0.45 \* 1.8 )
8. #define Kintphi\_y (K\_y\*0.000004 \*1000 \*1.2 \*5 /2/2 )
9. #define K\_x K\_y
10. #define Kp\_x Kp\_y
11. #define Ki\_x Ki\_y
12. #define Kintphi\_x Kintphi\_y
13. #define K\_z (0.00960 \*2)
14. #define Kp\_z (K\_z \*3 \*1.2 )
15. #define Ki\_z (K\_z \*0.002 \*3 \*0.45 \*5 \*1000 )
16. #define Kintphi\_z (K\_z\*0.000004 \*1.6 \*1000 )
17. #define Kax 0.0003 \*16
18. //#define LIMITSERV 100
19. #define LIMITSERV 100\*16
20. #define LIMITDIFF 100\*16
21. //130 - polet 2 motora 1047

**Расчет управляющих сигналов.**

1. #define Ki\_az 0.003
2. void task1()
3. {
4. servdiff\_y=-(gy-startgy)\*Kp\_y -intgy\*Ki\_y -intphi\_y\*Kintphi\_y;
5. servdiff\_x=-(gx-startgx)\*Kp\_x -intgx\*Ki\_x -intphi\_x\*Kintphi\_x;
6. servcomp\_z= - Kp\_z\*(gz-startgz) - Ki\_z \* intgz -intphi\_z\*Kintphi\_z;
7. if (servdiff\_y>LIMITDIFF) servdiff\_y=LIMITDIFF;
8. if (servdiff\_y<-LIMITDIFF) servdiff\_y=-LIMITDIFF;
9. if (servdiff\_x>LIMITDIFF) servdiff\_x=LIMITDIFF;
10. if (servdiff\_x<-LIMITDIFF) servdiff\_x=-LIMITDIFF;
11. if (servcomp\_z>LIMITDIFF) servcomp\_z=LIMITDIFF;
12. if (servcomp\_z<-LIMITDIFF) servcomp\_z=-LIMITDIFF;
13. servsumm2=intdist\*K\_idist+edist\*K\_pdist;
14. if (servsumm2>LIMITSERV+STARTMINUS) servsumm2=LIMITSERV+STARTMINUS ;
15. servsumm=servsumm2-(intaz)\*Ki\_az - (az-startaz)/36000 \*0;
16. if (ultrasonic.duration <8000) servsumm=servsumm-(8000-ultrasonic.duration);
17. serv[0] = 10000\*1.14+(servsumm+servdiff\_y-STARTMINUS+servcomp\_z)\*Kc;
18. serv[1] = 10000 +(servsumm-servdiff\_y-STARTMINUS+servcomp\_z) ;
19. serv[2] = 10000 +(servsumm-servdiff\_x-STARTMINUS-servcomp\_z) ;
20. serv[3] = 10000 +(servsumm+servdiff\_x-STARTMINUS-servcomp\_z) ;
21. servN=0;
22. SH.newTask(impulse);
23. if (flag==1) SH.newTask(task1,10);
24. }

4-6 Вычисление управления.

7-12 Защита от ошибок, ограничивающая возможности системы управления.

13-14 вычисление сигнала управления тягой.

16 – защита, отключающая двигатели рядом с землей.

17-20 Блок суммирования сигналов управления.

21 – 22 вызов системы выдачи импульсов.

23 – задержка на повторный вызов.

**Выдача телеметрии (информация поступает на главный контроллер и оттуда на экран ПК):**

1. void taskPrint()
2. {
3. Serial.print(intgx\*Ki\_x); Serial.write("\t");
4. Serial.print(intgy\*Ki\_y); Serial.write("\t");
5. Serial.print(intgz\*Ki\_z); Serial.write("\t\t");
6. Serial.print(intphi\_x\*Kintphi\_x); Serial.write("\t");
7. Serial.print(intphi\_y\*Kintphi\_y); Serial.write("\t");
8. Serial.print(intphi\_z\*Kintphi\_z); Serial.write("\t\t");
9. Serial.print(intaz\*Ki\_az); Serial.write("\t\t");
10. Serial.print(servsumm); Serial.write("\t\n ");
11. SH.newTask(taskPrint,20);
12. }

11 – задержка на повторный вызов.

**8. Итоги работы и выводы, направление дальнейшей работы.**

В работе исследована линейная САР, требования по порядку передаточных функций корректирующих звеньев.

Проведена адаптация найденных функций к реальной системе.

Проработан вопрос связи Макета с ПК и управления им по этому каналу.

В процессе работы над макетом не выявлено никаких принципиальных сложностей в плане построения СУ. Однако, анализ системы затруднен в силу отсутствия информации о параметрах исполнительных органов системы и коэффициентов сопротивления воздуха. Без знания этих коэффициентов проводить сколь-нибудь точное моделирование затруднительно.

Существующая на данный момент модель Simulink отражает процесс только качественно.

Планируется уточнение коэффициентов модели, по результатам испытания макета в режиме реального полета. Для чего необходимо доработать систему дистанционного управления.

Список приложений:

- Схема контроллера KKboard.

- Документация датчика MPU-6050

- Тексты программ и библиотек.

- Модель Simulink канала крена.

Список использованной литературы:

1. В.Н. Бранец, И.П. Шмыглевский. «Введение в теорию бесплатформенных инерциальных навигационных систем». «Наука». Физматлит. 1992.

2. В.А. Бесекерский, Е.П. Попов. «Теория систем автоматического регулирования». «Наука» Физматлит. 1975.