ҚАЗАҚСТАН РЕСПУБЛИКАСЫ ҒЫЛЫМ ЖӘНЕ ЖОҒАРҒЫ БІЛІМ МИНИСТРЛІГІ

«Қ.И. Сәтбаев атындағы Қазақ ұлттық техникалық зерттеу университеті» коммерциялық емес акционерлік қоғамы

SATBAYEV UNIVERSITY

Автоматика және ақпараттық технологиялар институты

«Робототехника және автоматиканың техникалық құралдары» кафедрасы

Тлепбергенов Жумагали Уразгалиулы

«ҰҰА ұшуды тұрақтандыру жүйесін әзірлеу»

Дипломдық жобаға
ТҮСІНДІРМЕ ЖАЗБА

6В07111 – Робототехника және мехатроника

Алматы 2023

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ ҒЫЛЫМ ЖӘНЕ ЖОҒАРҒЫ БІЛІМ МИНИСТРЛІГІ

«Қ.И. Сәтбаев атындағы Қазақ ұлттық техникалық зерттеу университеті» коммерциялық емес акционерлік қоғамы

Автоматика және ақпараттық технологиялар институты

«Робототехника және автоматиканың техникалық құралдары» кафедрасы

ҚОРҒАУҒА ЖІБЕРІЛДІ
Кафедра меңгерушісі РТжАТҚ
техника ғылымының кандидаты
_____ Қ.Ә. Өжікенов
«__» мамыр 2023 ж.

Дипломдық жобаға
ТҮСІНДІРМЕ ЖАЗБА

Тақырыбы: «ҰҰА ұшуды тұрақтандыру жүйесін әзірлеу»

6В07111 – Робототехника және мехатроника

Орындаған _____ Тлепбергенов Ж.У.

Рецензент
ААА Жалпы білім беру кафедрасының меңгерушісі, т.ғ.к., қауымдастырылған профессоры
_____ Сейдилдаева А.К.
қолы аты-жөні
«30» мамыр 2023 ж.

Ғылыми жетекшісі
Қауымдастырылған профессор
_____ Карымсакова Н.
«30» мамыр 2023 ж.

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ ҒЫЛЫМ ЖӘНЕ ЖОҒАРҒЫ БІЛІМ МИНИСТРЛІГІ

«Қ.И. Сәтбаев атындағы Қазақ ұлттық техникалық зерттеу университеті» коммерциялық емес акционерлік қоғамы

Автоматика және ақпараттық технологиялар институты

«Робототехника және автоматиканың техникалық құралдары» кафедрасы

6В07111 – Робототехника және мехатроника

БЕКІТЕМІН
Кафедра меңгерушісі РТжАТҚ
техника ғылымының кандидаты
_____ Қ.Ә. Өжікенов
«31» мамыр 2023 ж.

Дипломдық жобаны орындауға арналған
ТАПСЫРМА

Білім алушы Тлепбергенов Жумагали Уразгалиулы
Тақырыбы: ҰҰА ұшуды тұрақтандыру жүйесін әзірлеу
_____ 2022 ж. «15» 11 №418-Пб бұйрығымен бекітілген
Аяқталған жұмысты тапсыру мерзімі: «31» мамыр 2023 ж.
Дипломдық жұмыстың бастапқы деректері: Arduino UNO, MPU 6050.
Дипломдық жұмыста қарастырылатын мәселелер тізімі:
а)Квадрокоптердің және оның негізгі компоненттерінің жұмыс принциптерін зерттеу.
б)Квадрокоптер контекстіндегі акселерометр мен гироскоптың (MPU 6050) функцияларын талдау.
в)Квадрокоптерді тұрақтандыру жүйесіндегі Arduino Uno рөлінің сипаттамасы.
г)Тұрақтандыру жүйесінде PID реттегішінің қолданылуын зерттеу.
д)Квадрокоптер жақтауын жобалау және сәйкес қозғалтқыштарды, бұрандаларды және батареяларды таңдау.
е)Flysky i6 радио басқару жүйесін орнату және қосу.
ё)Arduino Uno және MPU6050 қосылымы мен конфигурациясын қоса алғанда, квадрокоптерді құрастыру және сынау.
Графикалық материалдың тізбегі (міндетті сызбаларды дәл көрсете отырып):
жұмыс презентациясы слайтарда 15 көрсетілген
Ұсынылатын негізгі әдебиеттер: 12 атаулардан

Дипломдық жұмысты (жобаны) дайындау

КЕСТЕСІ

Бөлімдер атауы, әзірленетін сұрақтар тізбесі	Ғылыми жетекшіге ұсыну мерзімдері	Ескертпелер
Теориялық бөлім	16.01-12.02.2023 ж	Орындалды
Бағдарламалық бөлім	12.02-20.03.2023 ж.	Орындалды
Зерттеу бөлімі	20.03-17.04.2023 ж.	Орындалды
Қорытынды бөлім	17.04-15.05.2023 ж.	Орындалды

Аяқталған дипломдық жұмыс (жоба) үшін, оған қатысты бөлімдердің жұмыстарын (жобасын) көрсетумен, кеңесшілері мен қалып бақылаушының

қолдары

Бөлімдердің атауы	**Кеңесшілер, тегі, аты, әкесінің аты, (ғылыми дәрежесі, атағы)**	**Қол қойылған күні**	**Қол**
Қалып бақылаушы	Бигалиева Ж.С, техника ғылымдарының магистрі, оқытушы	26.05.23	
Негізгі бөлім	Карымсакова Н, Қауымдастырылған профессор	30.05.2023	
Есептеу бөлім	Карымсакова Н, Қауымдастырылған профессор	30.05.2023	

Ғылыми жетекшісі _____ Карымсакова Н

Білім алушы тапсырманы орындауға алды _____ Тлепбергенов Ж.У.

Күні «31» Мамыр 2023 ж.

АҢДАТПА

Бұл жұмыс шағын ұшқышсыз ұшатын аппараттардың (ҰҰА), атап айтқанда квадрокоптерлердің ұшуды тұрақтандыру жүйесін дамытуды ұсынады. Басқару жүйесінің негізгі элементі MPU 6050 гироскопымен біріктірілген Arduino Uno микроконтроллері болып табылады.Бұл құрылғы ұшқышсыз ұшу аппаратының орны мен айналу жылдамдығы туралы мәліметтерді өңдеу арқылы квадрокоптердің ұшуын тұрақтандыруға мүмкіндік береді.

Негізгі басқару құралы ретінде PID контроллерінің алгоритмі пайдаланылды. Басқарудың бұл әдісі басқару процесін оңтайландыруға және әртүрлі жағдайларда ұшқышсыз ұшу аппаратының ұшуының тұрақтылығын сақтауға мүмкіндік береді.

Квадрокоптер жақтауы құрылымның жеңілдігі мен беріктігін қамтамасыз ететін ПВХ құбырларынан жасалған. Ұшқышсыз ұшу аппараттарының ұшуын басқару үшін тиісті қабылдағышы бар Flysky i6 радио пульті таңдалды. Бұл қашықтықтағы ұшқышсыз ұшуды тиімді және сенімді басқаруды қамтамасыз етеді.

Жұмыс барысында квадрокоптерді басқарудың негізгі принциптері зерттелді, сондай-ақ оның ұшуын тұрақтандыру үшін бағдарламалық Алгоритмдер әзірленді және түзетілді. Дипломдық жұмыстың нәтижелерін әртүрлі модификациялар мен қиындық деңгейлерінде ұшқышсыз ұшу аппараттарын жасау үшін пайдалануға болады.

АННОТАЦИЯ

В данной работе представлена разработка системы стабилизации полета для малых безпилотных летательных аппаратов (БПЛА), в частности, квадрокоптера. Основным элементом системы управления является микроконтроллер Arduino Uno, совмещенный с гироскопом MPU 6050. Это устройство позволяет обеспечивать стабилизацию полета квадрокоптера посредством обработки данных о положении и скорости вращения БПЛА.

В качестве основного инструмента управления использовался алгоритм ПИД-регулятора. Этот метод контроля позволяет оптимизировать процесс управления и поддерживать стабильность полета БПЛА в различных условиях.

Каркас квадрокоптера был изготовлен из ПВХ-труб, что обеспечивает легкость и прочность конструкции. Для управления полетом БПЛА был выбран радиопульт Flysky i6 с соответствующим приемником. Это обеспечивает эффективное и надежное управление БПЛА на дистанции.

В ходе работы были исследованы основные принципы управления квадрокоптером, а также разработаны и отладены программные алгоритмы для стабилизации его полета. Результаты дипломной работы могут быть использованы для создания БПЛА различных модификаций и уровней сложности.

ABSTRACT

This paper presents the development of a flight stabilization system for small unmanned aerial vehicles (UAVs), in particular, quadrocopters. The main element of the control system is the Arduino Uno microcontroller, combined with the MPU 6050 gyroscope. This device allows you to stabilize the flight of the quadrocopter by processing data on the position and speed of rotation of the UAV.

The PID controller algorithm was used as the main control tool. This control method allows you to optimize the control process and maintain the stability of the UAV flight in various conditions.

The frame of the quadrocopter was made of PVC pipes, which ensures lightness and strength of the structure. To control the flight of the UAV, the Flysky i6 radio control with the appropriate receiver was chosen. This ensures efficient and reliable control of the UAV at a distance.

In the course of the work, the basic principles of quadrocopter control were investigated, and software algorithms were developed and debugged to stabilize its flight. The results of the thesis can be used to create UAVs of various modifications and levels of complexity.

МАЗМҰНЫ

КІРІСПЕ

Соңғы жылдары ұшқышсыз ұшу аппараттары (ҰҰа), атап айтқанда квадрокоптерлер агрокультура мен геодезиядан бастап күзет пен ойын - сауыққа дейін әртүрлі салаларда белсенді қолданыла бастады. Дегенмен, олардың кең таралуына қарамастан, квадрокоптерлердің ұшуын басқару және тұрақтандыру жүйелерін әзірлеу және откалау күрделі және өзекті болып қала береді. Бұл дипломдық жұмыс осы тақырыпқа арналған.

Жұмыстың мақсаты – квадрокоптердің мысалын пайдалана отырып, шағын ұшқышсыз ұшу аппараттары үшін ұшуды тұрақтандыру жүйесін әзірлеу және отладкалау. Басқару жүйесінің негізі Arduino UNO микроконтроллері және MPU6050 гироскопы болып табылады, олар ұшқышсыз ұшу аппараттарының орналасуы мен жылдамдығы туралы деректерді өңдейді, бұл оның ауада тұрақтануын қамтамасыз етеді.

Басқару процесін оңтайландыру және квадрокоптердің ұшуының тұрақтылығын сақтау үшін PID контроллерінің алгоритмі қолданылады. Бұл әдіс динамикалық жүйелерді басқару саласында кең таралған әдістердің бірі болып табылады және басқарудың жоғары дәлдігіне қол жеткізуге мүмкіндік береді.

Жұмыс барысында квадрокоптерді басқаруға арналған бағдарламалық Алгоритмдер әзірленеді және түзетіледі, сондай-ақ алынған жүйені нақты жағдайларда тестілеу орындалады.

Осылайша, бұл жұмыс ұшқышсыз ұшу аппараттарын әзірлеу саласына маңызды үлес болып табылады және оның нәтижелерін әртүрлі модификациялар мен қиындық деңгейлеріндегі ұшқышсыз ұшу аппараттарын жасау үшін пайдалануға болады.

1.Теориялық шолу

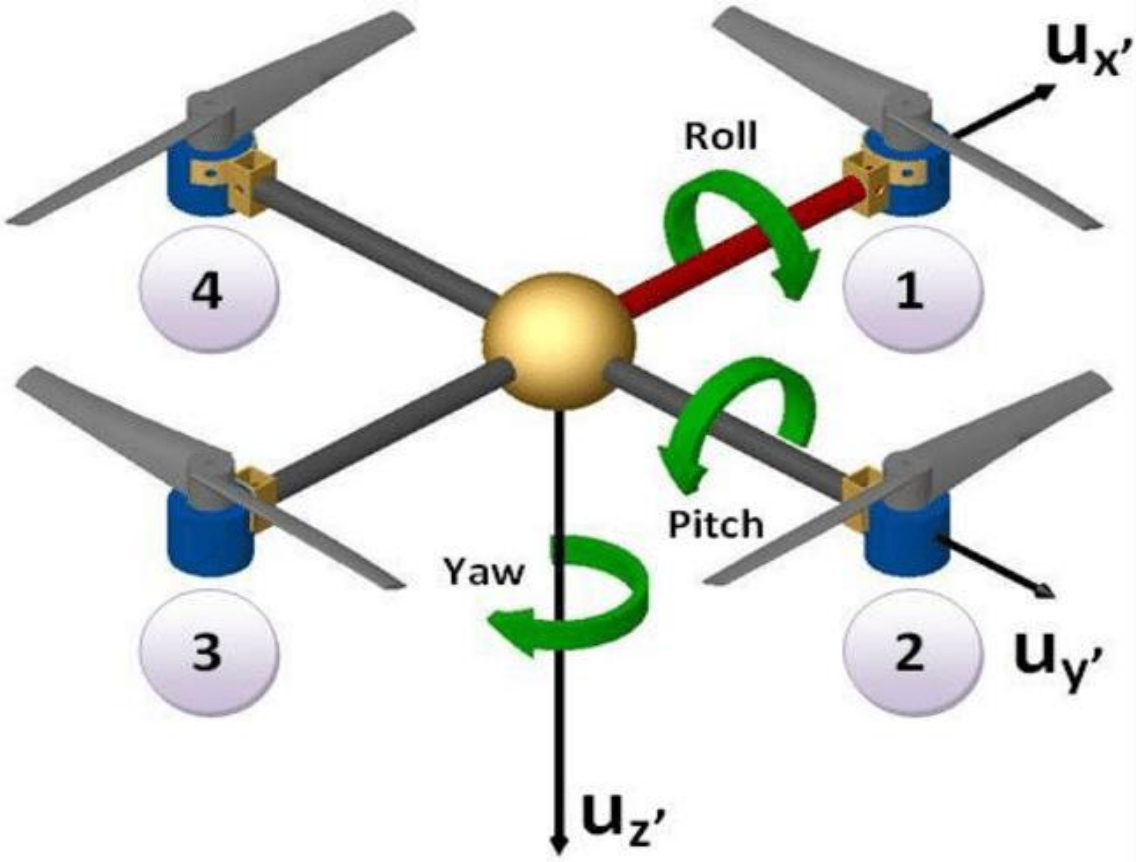
1.1 Квадрокоптердің жұмыс принципі

Квадрокоптер - көтеру және басқару қозғалысын жасау үшін төрт роторды пайдаланатын ұшқышсыз ұшатын аппараттың бір түрі. Қарама - қарсы екі лопсты сағат тілімен, ал қалған екеуі сағат тіліне қарсы айналады. Бұл айналу моментін өтеуге көмектеседі және квадрокоптердің тұрақтылығын сақтайды.

Ұшу үшін барлық төрт қозғалтқыш бірдей жылдамдықпен айналады, бұл бірдей көтеру күшін жасайды. Көтеру күші квадрокоптердің салмағынан асып түседі, бұл оның көтерілуіне мүмкіндік береді.

Солға немесе оңға бұрылу үшін (yaw деп аталатын құбылыс) бір жағындағы қозғалтқыштар екінші жағындағы қозғалтқыштарға қарағанда жылдамырақ айналады. Бұл квадрокоптерді айналдыратын айналу моментін жасайды.

Алға және артқа (pitch) немесе солға және оңға (roll) қозғалу үшін квадрокоптер қозғалтқыштарының айналу жылдамдығын өзгертеді, осылайша бір жағында көтеру күші екінші жағына қарағанда үлкен болады. Бұл квадрокоптерді еңкейтеді және квадрокоптерді дұрыс бағытта жылжытатын күштің көлденең компонентін жасайды.



1.1 - сурет – квадрокоптердің әртүрлі осьтерге көлбеуі.

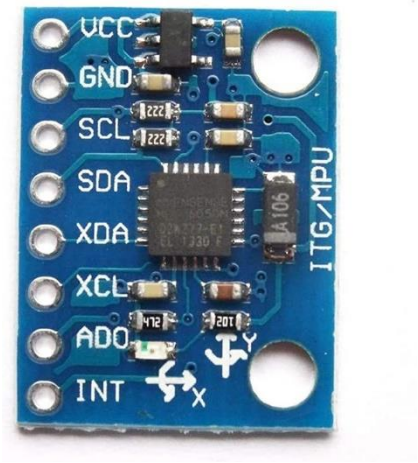
1.2 Акселерометр мен гироскоптың жұмыс принципі (MPU 6050)

MPU-6050 - бір блокта 3 осьті акселерометр мен 3 осьті гироскопты қамтитын құрама модуль.

Акселерометр үш ось бойынша (X, Y, Z) үдеуді өлшейді. Қозғалыс болмаған жағдайда акселерометр ауырлық векторын анықтай алады және квадрокоптердің жерге қатысты бұрышын анықтау үшін қолданылады.

Гироскоп үш осьтің айналасындағы бұрыштық айналу жылдамдығын өлшейді. Гироскоптар позицияның лезде өзгеруін анықтау үшін әсіресе пайдалы, мысалы, кенеттен қозғалыс немесе жел.

MPU-6050 таңдауы бірнеше себептерге байланысты. Ол акселерометр мен гироскопты біріктіреді, бұл оны қолдануға өте ыңғайлы етеді. I2C интерфейсін қолдана отырып, MPU-6050 Arduino Uno және басқа да көптеген микроконтроллерлерге оңай қосылады. Ақырында, ол қол жетімді бағамен жақсы дәлдік пен сенімділікке ие, бұл оны ұшқышсыз ұшу жобалары үшін жақсы таңдау етеді.



1.2 - сурет – MPU 6050

1.3 Arduino Uno сипаттамасы және оның тұрақтандыру жүйесіндегі рөлі

Arduino Uno-басқару жүйелерін әзірлеуге және прототиптеуге арналған ең танымал және қол жетімді платформалардың бірі. ATmega328P микроконтроллеріне негізделген бұл микроконтроллер тақтасы шағын ұшқышсыз ұшу аппараттарында (ұшқышсыз ұшу аппараттарында) ұшуды тұрақтандыру жүйелерін іске асырудың кең ауқымын ұсынады.

Arduino Uno-ның ұшуды тұрақтандыру жүйесіндегі рөлі акселерометр және гироскоп (бұл жағдайда MPU 6050) сияқты сенсорлардан алынған деректер

негізінде қозғалтқыштарды басқару және квадрокоптерді тұрақтандыру болып табылады. Arduino Uno қозғалтқыштардың айналу жылдамдығын басқару үшін датчиктерден алынған деректерді өңдейтін, басқару алгоритмдерін қолданатын және ESC-ге (Электрондық жылдамдықты басқару құралы) сәйкес сигналдарды беретін ұшу диспетчері ретінде әрекет етеді.

1.3 - сурет – Arduino UNO

Arduino Uno ұшуды тұрақтандыру жүйесінде қолдануға жарамды ететін бірнеше маңызды сипаттамаларға ие:

1. Atmega328p микроконтроллері: Arduino Uno Сенсорлардан деректерді өңдеу және қажетті басқару алгоритмдерін орындау үшін жеткілікті өңдеу қуаты мен жады бар atmega328p микроконтроллерімен жабдықталған.

2. Енгізу-шығару порттарының кең жиынтығы: Arduino Uno-да сенсорлар, қозғалтқыштар және басқа перифериялық құрылғылар сияқты әртүрлі компоненттерді қосу және басқару үшін пайдалануға болатын әртүрлі сандық және аналогтық енгізу-шығару порттары бар.

3. Кітапханалар және дамыған қоғамдастық: Arduino Uno - ны көптеген кітапханалар қолдайды және кең әзірлеушілер қауымдастығы бар. Бұл ұшуды тұрақтандыру жүйесін бағдарламалауды және дамытуды жеңілдетеді, өйткені дайын кітапханаларды пайдалануға және тәжірибелі әзірлеушілерден қолдау алуға болады.

4. Икемділік және теңшеу: Arduino Uno әзірлеушілерге бағдарламалық жасақтама мен басқару алгоритмдерін ұшуды тұрақтандыру жүйесінің талаптарына сәйкес реттеуге мүмкіндік береді. Бұл икемділік пен жүйені нақты шарттар мен талаптарға оңтайландыру мүмкіндігін қамтамасыз етеді.

Ұшуды тұрақтандыру жүйесінде Arduino Uno акселерометр мен гироскоптан деректерді алу үшін MPU 6050-мен байланысады, содан кейін бұл деректерді қозғалтқыштардың айналу жылдамдығын реттейтін және ұшу кезінде квадрокоптерді тұрақтандыратын басқару сигналдарын есептеу үшін пайдаланады.

Arduino Uno ұшуды тұрақтандыру жүйесінде маңызды рөл атқарады, ол шағын ұшқышсыз ұшу аппараттары үшін сенімді басқару мен ұшу тұрақтылығын қамтамасыз етеді. Оның икемділігі, функционалдығы және қолжетімділігі оны ұшуды басқару жүйелерін дамыту үшін тартымды таңдау жасайды.

1.4 PID реттегішінің жұмыс принципі және оны тұрақтандыру жүйесінде қолдану

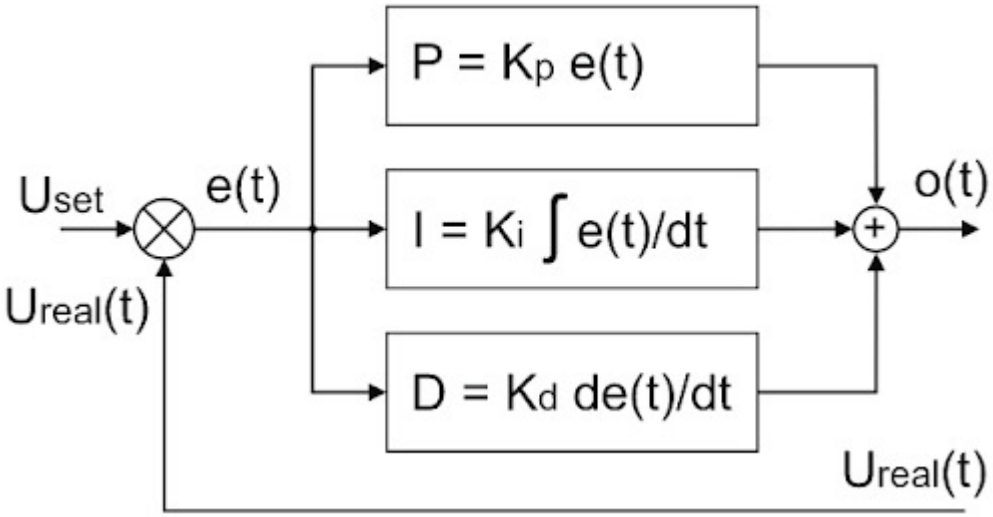
PID реттегіші (пропорционалды интегралды дифференциалды реттегіш) жүйенің тұрақтануы мен дәлдігін қамтамасыз ететін кеңінен қолданылатын басқару алгоритмі болып табылады. Шағын ұшқышсыз ұшу аппараттарының ұшуды тұрақтандыру жүйесінде PID контроллері тұрақтылықты сақтауда және ұшу қателіктерін азайтуда маңызды рөл атқарады.

PID контроллерінің жұмыс принципі қажетті мәнді (орнатылған бұрыш немесе жылдамдық) және ағымдағы өлшенген мәнді салыстыруға, содан кейін қатені түзету үшін басқару сигналын есептеуге негізделген. PID контроллері үш компоненттен тұрады:

1. Пропорционалды компонент (P): реттегіштің пропорционалды компоненті қажетті және өлшенген мән арасындағы ағымдағы қатеге пропорционалды. Қате неғұрлым көп болса, түзету соғұрлым күшті болады. Пропорционалды компонент қатеге жылдам жауап береді, бірақ жүйеде шамадан тыс реттеу мен тербелістерді тудыруы мүмкін.

2. Интегралдық компонент (I): Интегралдық компонент уақыт бойынша жинақталған қателіктерді ескереді. Ол қатеден интегралды есептейді және оны басқару сигналына қосады. Интегралды компонент жүйелік қателіктердің орнын толтыруға мүмкіндік береді және ұзақ мерзімді перспективада дәлдікті қамтамасыз етеді.

3. Дифференциалды компонент (D): дифференциалды компонент қатенің өзгеру жылдамдығын ескереді. Ол қатенің туындысын есептейді және оны басқару сигналын түзету үшін пайдаланады. Дифференциалды компонент тербелістердің алдын алуға көмектеседі және жүйенің тұрақтылығын арттырады.



1.4 - сурет – PID реттегіші

Шағын ұшқышсыз ұшу аппараттарының ұшуды тұрақтандыру жүйесінде квадрокоптердің қалаған бұрышын немесе тұрақтандыру деңгейін ұстап тұру үшін PID контроллері қолданылады. Ол квадрокоптердің ағымдағы бұрышын өлшеу үшін акселерометр мен гироскоптан (мысалы, MPU 6050) деректерді алады.

Шағын ұшқышсыз ұшу аппараттарының ұшуды тұрақтандыру жүйесінде квадрокоптердің қалаған бұрышын немесе тұрақтандыру деңгейін ұстап тұру үшін PID контроллері қолданылады. Ол квадрокоптердің ағымдағы бұрышын өлшеу үшін акселерометр мен гироскоптан (мысалы, MPU 6050) мәліметтерді алады және оларды қажетті мәндермен салыстырады. Осы салыстырудың негізінде PID контроллері қозғалтқыштардың айналу жылдамдығын өзгертетін және квадрокоптердің орнын түзететін басқару сигналдарын жасайды.

PID контроллері қатені азайту және квадрокоптерді тепе-теңдікте ұстау арқылы ұшу тұрақтылығын қамтамасыз етеді. Ол тұрақтандыру жүйесіне сыртқы кедергілерге тез жауап беруге және жағдайлар өзгерсе де ұшу дәлдігін сақтауға мүмкіндік береді.

Ұшуды тұрақтандыру жүйесінде PID контроллерін оңтайлы баптау маңызды рөл атқарады. Бұл квадрокоптердің нақты өнімділігі мен ұшу жағдайларына сәйкес келетін пропорционалды, интегралды және дифференциалды күшейтуді реттеуді қамтуы мүмкін.

2.Ұшуды тұрақтандыру жүйесін әзірлеу және іске асыру

2.1 ПВХ құбырынан квадрокоптердің рамасын жасау

Материалды таңдау: квадрокоптердің жақтауына материал таңдау кезінде беріктік, салмақ және құны сияқты факторлар ескерілді. ПВХ құбырлары олардың жеңілдігі, беріктігі және қол жетімділігі арқасында таңдалды.

Өндіріс процесі: ПВХ құбырының бір метрлік бөлігі негізге алынды. Ол әрқайсысы 23 см болатын төрт бірдей бөлікке бөлінді. Бөлшектер ПВХ Х қосқышымен біріктіріліп, жақтауға «Х» пішінін берді. Бұл пішін массаның біркелкі таралуын қамтамасыз етуге көмектеседі, бұл квадрокоптердің ұшуының тұрақтылығы үшін маңызды.

2.1 - Сурет – Квадрапотер рамсы

Қозғалтқыштарды монтаждау: Қозғалтқыштарды рамаға орнату үшін жақсы беріктігі мен тозуға төзімділігімен ерекшеленетін шыны талшықтан жасалған арнайы кронштейндер жасалды. Шыны талшықтың салмағы да аз,бұл квадрокоптер үшін маңызды параметр болып табылады.

2.2 - сурет – Рамаға кронштейн

Сынақ және итерация: Өндірілгеннен кейін жақтау беріктігі мен тұрақтылығы үшін сыналған. Сынақ нәтижелеріне сүйене отырып, жақтаудың тұрақтылығы мен беріктігін қамтамасыз ету үшін дизайнға шағын түзетулер енгізілді.

2.2 Қозғалтқыштар мен пропеллерлердің таңдауы мен сипаттамалары

Қозғалтқыштарды таңдау: бұл жоба үшін a2212/13T 1000KV коллекторсыз қозғалтқыштар таңдалды. Қозғалтқыштың бұл түрі жоғары тиімділікті, ұзақ қызмет ету мерзімін және тұрақты өнімділікті қамтамасыз етеді, бұл квадрокоптерлерде қолдануға өте ыңғайлы.

Қозғалтқыштардың сипаттамалары: A2212/13T 1000KV қозғалтқыштары 12В дейін жұмыс істей алады, максималды ток 12а. олар жоғары тиімділікке ие, бұл жұмыс кезінде жылуды азайтады және олардың қызмет ету мерзімін ұзартады.

2.3 - сурет – мотор A2212/13t 1000kv

Пропеллерді таңдау: квадрокоптердің ұшу тиімділігі мен тұрақтылығын арттыру үшін 1045 өлшемді пропеллерлер таңдалды.1045 өлшемі пропеллердің диаметрі 10 дюйм және қадамы 4.5 дюйм екенін көрсетеді.

Бұрандалардың сипаттамалары: 1045 Пропеллерлері өңдеу мен энергия тиімділігі арасындағы жақсы тепе-теңдікті қамтамасыз етеді. Олардың мөлшері мен қадамы тұрақты ұшуды және батареяның ұзақ қызмет ету мерзімін қамтамасыз ету үшін оңтайландырылған.

2.4 - сурте – 1045 Пропеллерлері

2.3 Квадрокоптерлер үшін қол жетімді батарея түрлеріне шолу

1. NiMH (никель-металл гидридті) батареялар: бұл батареялар салыстырмалы түрде төмен қуат сыйымдылығымен және орташа өнімділігімен танымал. Олар батареялардың басқа түрлерімен салыстырғанда біршама ауыр және көлемді. Дегенмен, олар арзанырақ және жаңадан бастаған ұшқыштар үшін

немесе жоғары өнімділікті немесе ұзақ ұшу уақытын қажет етпейтін жағдайларда пайдалы болуы мүмкін.

2.5 - сурет – NiMH (никель-металл гидридті) батареялар

2.Li-Ion (литий-ионды) батареялар: бұл батареялар NiMH-мен салыстырғанда жоғары қуат сыйымдылығына ие және жеңілірек болуы мүмкін. Олар сондай-ақ разряд кезінде кернеудің тұрақтылығына ие. Алайда, олардың жоғары ток беру қабілеті төмен, бұл сіздің квадрокоптеріңіздің қуатын шектеуі мүмкін.

2.6 -сурет – Li-Ion (литий-ионды) батареялар

3. Li-Po (литий-полимерлі) батареялар: Li-Po батареялары жоғары қуат сыйымдылығына, төмен салмаққа және жоғары ток беру қабілетіне ие, бұл оларды квадрокоптерлер үшін тамаша таңдау жасайды. Олар салыстырмалы түрде жоғары кернеуге ие және батареялардың басқа түрлерімен салыстырғанда қуаттың салмаққа қатынасы жақсы. Сонымен қатар, Li-Po батареялары әдетте үлкен сыйымдылыққа ие және ұзақ ұшу уақытын қамтамасыз ете алады.

2.7 -сурет – Li-Po (литий-полимерлі) батареялар

Li-Po батареясын таңдау негіздемесі:

Квадрокоптер үшін Li-Po аккумуляторы оның жоғары қуат сыйымдылығына, төмен салмағына және қозғалтқыштарға жеткілікті ток беру қабілетіне байланысты таңдалды. Бұл факторлар квадрокоптердің максималды өнімділігі мен ұзақ ұшу уақытын қамтамасыз ету үшін маңызды. Сонымен қатар, Li-Po батареялары квадрокоптердің жеңіл және маневрлі болуына мүмкіндік беретін жақсы қуат пен салмақ қатынасына ие.

Таңдалған батареяның сипаттамалары мен параметрлері:

2.8 - сурет – Квадрокоптерде қолданылатын батарея

Таңдалған батареяның сыйымдылығы 2200 mAh және номиналды кернеуі 11.1 V. батареяның Li-Po жағдайында әрбір ұяшықтың кернеуі шамамен 3.7 V, ал "3S" батареяда 3 ұяшық бар екенін білдіреді. Осылайша, батареяның жалпы кернеуі 3.7 V * 3 = 11.1 V. бұл батарея квадрокоптердің ұшу операцияларын жақсы ұшу ұзақтығымен қолдау үшін жеткілікті қуат пен сыйымдылықты қамтамасыз етеді.

2.4 Flysky i6: Радио басқару элементтерін қосу және конфигурациялау

Flysky i6-радио басқарылатын ұшақтарды, дрондарды және тікұшақтарды қоса алғанда, радио басқарылатын модельдерге арналған танымал қашықтан басқару құралы. Бұл қол жетімділікті, сенімділікті және пайдаланудың қарапайымдылығын бағалайтын әуесқойлар мен кәсіпқойлар үшін тамаша таңдау.

2.9 - сурет – Flysky i6 пульт

Flysky i6 негізгі мүмкіндіктері:

Арналар саны: Flysky i6 6 арнадан тұрады, ол басқару опцияларының кең ауқымын қамтамасыз етеді.

Байланыс ауқымы: Байланыс ауқымы жағдайларға байланысты өзгереді, бірақ орташа есеппен шамамен 500-1000 метрге жетеді.

Дисплей: Flysky i6 артқы жарықтандырылған монохромды экранмен жабдықталған, ол тіпті аз жарықта да құрылғының барлық параметрлерін басқаруды жеңілдетеді.

Таймер: қашықтан басқару пульті ұшу уақытын бақылауға мүмкіндік беретін таймермен жабдықталған.

Батарея: Flysky i6 4 AA батареясынан қуат алады, бұл оны жеңіл және далада қолдануға ыңғайлы етеді.

Модельдеу: Flysky i6 жадтағы 20-ға дейін әртүрлі үлгілерді қолдайды, бұл әртүрлі құрылғылар арасында қайта конфигурациялаусыз ауысуды жеңілдетеді.

Бағдарламалану мүмкіндігі: Flysky i6-да дәл реттеуді басқару, экспонент, қос жылдамдық режимі және т.б. сияқты теңшелетін әртүрлі мүмкіндіктер бар.

2.10 - сурет – FLYSKY i6 қабылдағыш

Flysky i6 қосу және конфигурациялау:

Ресиверді қосу: Flysky i6 қашықтан басқару құралы ресиверге арнайы кабель арқылы қосылады. Ресивер құрылғыңызға дұрыс орнатылған болуы керек.

Жұптастыру: Қашықтан басқару құралы мен ресиверді бірге пайдаланбас бұрын, оларды жұптастыру керек. Бұл қашықтан басқару пульті мен ресивердің бір-бірімен байланысуын қамтамасыз ететін процесс.

Контроллерді теңшеу: Flysky i6 құрылғыңыздың әртүрлі басқару функциялары үшін 6 арнаның әрқайсысын теңшеуге мүмкіндік береді.

Үлгі параметрлері: Flysky i6 әртүрлі үлгілер үшін параметрлерді сақтауға және жүктеуге мүмкіндік береді, бұл олардың арасында ауысуды жеңілдетеді.

Flysky i 6-бұл әуесқойлар мен радио басқарылатын модель мамандары үшін кең мүмкіндіктер ұсынатын сенімді және жан-жақты басқару пульті.

2.5 Arduino Uno және MPU 6050 қосу және конфигурациялау

MPU 6050-ді Arduino Uno-ға қосу:

Алдымен қуатты қосыңыз. MPU6050-де VCC-ді Arduino-да 3.3 V-ге, ал Mpu6050-де GND-ді Arduino-да GND-ге қосыңыз.

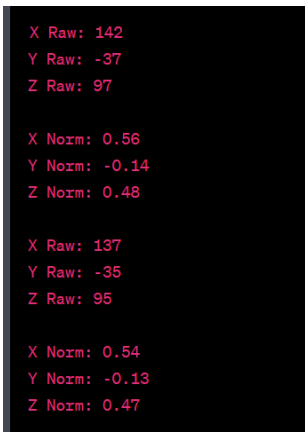
Содан кейін MPU6050-дегі SDA (сериялық деректер) Arduino-дағы A4-ке және Mpu6050-дегі SCL-ге (сериялық сағат) Arduino-дағы A5-ке қосыңыз.

Соңында, AD0-ді MPU6050-ге GND-ге қосып, оның 0x68 мекен-жайын орнатыңыз, бұл MPU6050 үшін стандартты мекен-жай.

Қосылымды орнату және тексеру:

```
// Подключение библиотеки MPU6050
#include <Wire.h>
#include <MPU6050.h>

// Создание объекта MPU6050
MPU6050 mpu;

void setup() {
  // Инициализация серийного порта
  Serial.begin(9600);
  // Проверка подключения MPU6050
  while (!mpu.begin(MPU6050_SCALE_2000DPS, MPU6050_RANGE_2G)) {
    Serial.println("Could not find a valid MPU6050 sensor, check wiring!");
    delay(500);
  }
}

void loop() {
  // Чтение данных с MPU6050
  Vector rawGyro = mpu.readRawGyro();
  Vector normGyro = mpu.readNormalizeGyro();

  // Вывод данных на серийный порт
  Serial.print("X Raw: "); Serial.println(rawGyro.XAxis);
  Serial.print("Y Raw: "); Serial.println(rawGyro.YAxis);
  Serial.print("Z Raw: "); Serial.println(rawGyro.ZAxis);

  Serial.print("X Norm: "); Serial.println(normGyro.XAxis);
  Serial.print("Y Norm: "); Serial.println(normGyro.YAxis);
  Serial.print("Z Norm: "); Serial.println(normGyro.ZAxis);

  delay(500);
}
```

2.11 сурет MPU6050 кітапханасының мысалдарынан тест скетч.



2.12 - Сурет – MPU6050 дұрыс жұмыс істейді.

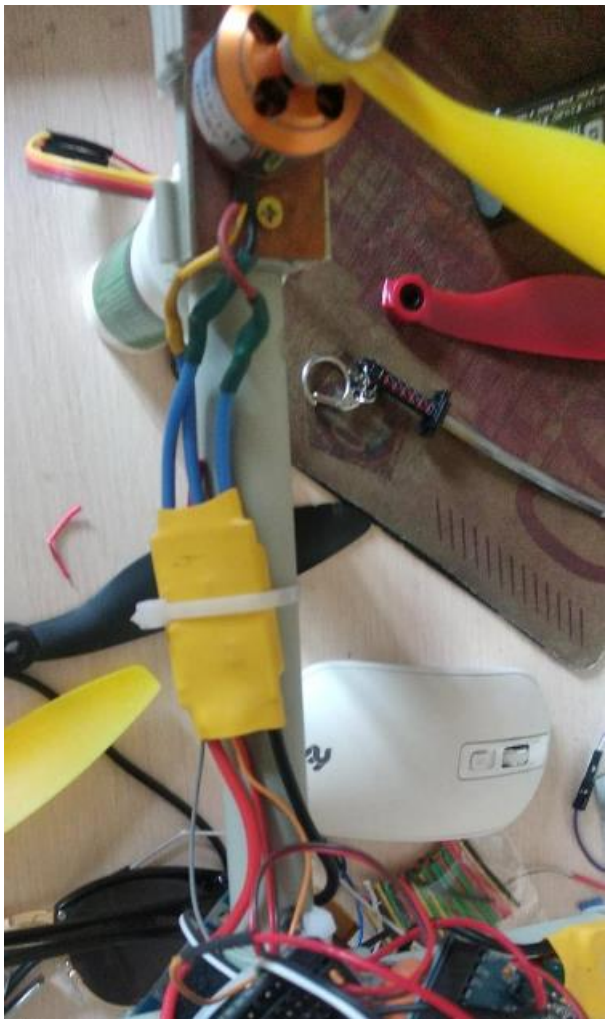
2.6 Квадрокоптерді толық құрастыру

Квадрокоптерді құрастырудың алғашқы қадамы қозғалтқыштарды рамаға орнату болып табылады. Әрбір қозғалтқыш жақтаудың ортасынан бірдей қашықтықта орнатылады, бұл жағдайда 22 см Қозғалтқыштарды рамаға орнату үшін арнайы әзірленген ұстағыштар пайдаланылды. Қозғалтқыштар ұстағыштарға M3 болттары арқылы бекітілді.

2.13 - сурет – Рамадағы мотор

Қозғалтқыштар жақтауға орнатылғаннан кейін келесі қадам-жылдамдық реттегіштерін орнату (ESC, Electronic Speed Controllers). ESC раманың әрбір "сәулесіне" орнатылады және қозғалтқыштарға қосылады.

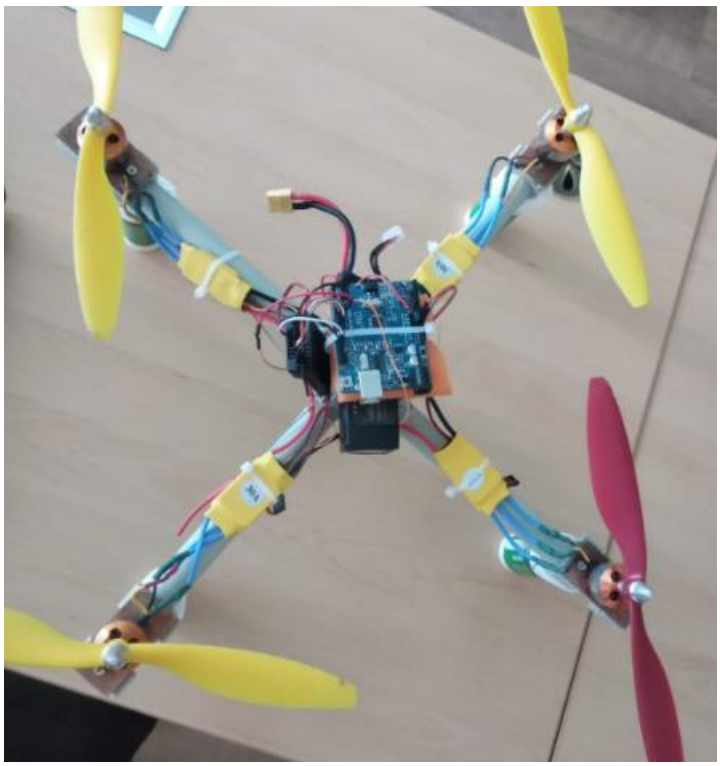
Келесі қадам - қуат тарату тақтасын орнату. Қуатты тарату тақтасы батарея қуатын ESC-тердің әрқайсысына және қуат қажет ететін квадрокоптердің басқа компоненттеріне таратады.



2.14 - сурет– Жақтауға ESC орнату

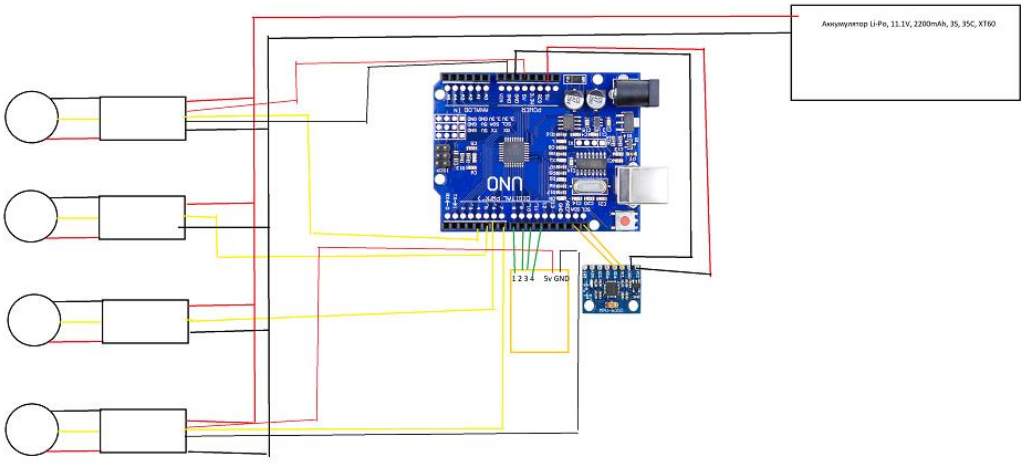
Жылдамдық реттегіштері мен қуат сымдарының тақтасын орнатқаннан кейін келесі қадам-Arduino және MPU6050 орнату. MPU6050-квадрокоптерге кеңістіктегі бағытын анықтауға мүмкіндік беретін гироскоп пен акселерометр модулі.

MPU6050 Arduino-ға екі жақты таспамен жабысады, осылайша екі компонент арасындағы физикалық байланысты қамтамасыз етеді. Содан кейін Arduino квадрокоптер жақтауының ортасына орнатылады. Бұл басқа компоненттерге қосылудың ыңғайлылығын қамтамасыз етеді және қозғалтқыштардың дірілінен болатын кедергілерді азайтады.



2.15 - сурет – квадрокоптер жинағының соңғы көрінісі

Барлық компоненттерді жақтауға орнатқаннан кейін, келесі қадам-электрониканы қосу. Бұл процеске Arduino Uno және MPU6050 қуат сымдарының тақтасына қосылу, сондай-ақ ESC қозғалтқыштары мен қуат сымдарының тақтасына қосылу кіреді.



2.16 - сурет – барлық компоненттердің электронды қосылуы схемасы

Барлық компоненттерді қосқаннан кейін біз скечті ҚОСЫМША А дан жүктейміз.

ҚОРЫТЫНДЫ

Осы дипломдық жұмыс аясында Arduino Uno және MPU 6050 негізіндегі квадрокоптерді, PID реттегіш алгоритмін және ПВХ құбырының жақтауын қолдана отырып, шағын ұшқышсыз ұшу аппараттарында ұшуды тұрақтандыру жүйесі жасалды. Жұмыс барысында келесі нәтижелер мен қорытындыларға қол жеткізілді:

Жұмыс барысында теориялық база жүргізіліп, квадрокоптер жұмысының негізгі принциптеріне талдау жүргізілді. Квадрокоптердің құрылымы, оның ішінде рамасы, Қозғалтқыштар мен пропеллерлер, акселерометр мен гироскоптың үдеулер мен бұрыштық жылдамдықтарды өлшеу принциптері зерттелді. PID реттегішінің алгоритмін зерттеуге және оны квадрокоптердің ұшуын тұрақтандыру үшін қолдануға баса назар аударылды.

Таңдалған компоненттер негізінде ұшуды тұрақтандыру жүйесі жасалды: Arduino Uno және MPU 6050. Arduino Uno қозғалтқыштарды басқару және акселерометр мен гироскоптан деректерді оқу үшін пайдаланылды. MPU 6050 үдеулер мен бұрыштық жылдамдықтарды дәл өлшеуге мүмкіндік берді.

Ұшуды тұрақтандыру үшін PID реттегіш алгоритмі енгізілді. Бұл алгоритм квадрокоптердің тұрақты ұшуын сақтау және сыртқы бұзылулардың орнын толтыру үшін басқару сигналдарын реттеуге мүмкіндік берді.

Тәжірибелер барысында тұрақтандыру жүйесі реттеліп, оның тиімділігі бағаланды.

PID контроллері алгоритмі бар Arduino Uno және MPU 6050 негізіндегі шағын ұшқышсыз ұшу аппараттары үшін әзірленген ұшуды тұрақтандыру жүйесі жақсы нәтиже көрсетті. Ол ұшудың тұрақтылығы мен бақылауын қамтамасыз етеді, сондай-ақ бұзылуларды өтеуге және көрсетілген ұшу режимдерінің дәл орындалуын қамтамасыз етуге мүмкіндік береді.

Әзірленген шағын ұшқышсыз ұшуды тұрақтандыру жүйесі жоғары тиімділікті көрсетеді және ұшқышсыз авиация саласында үлкен әлеуетке ие. Ол ұшуды сенімді және дәл басқаруды қамтамасыз етеді, бұл оны азаматтық және ғылыми мақсаттарды қоса алғанда, әртүрлі салаларда қолдануға мүмкіндік береді. Бұл жүйенің одан әрі дамуында оның функционалдығын жақсартуға, жұмыстың дәлдігі мен сенімділігін арттыруға, сондай-ақ оны нақты міндеттер мен талаптарға бейімдеуге болады. Бұл қосымша мүмкіндіктерді қосуды, басқару алгоритмдерін оңтайландыруды, басқа жүйелермен және құрылғылармен интеграцияны жақсартуды және миниатюризация мен энергия тиімділігін арттыруды қамтуы мүмкін. Мұндай жетілдірулер шағын ұшқыштарда ұшуды тұрақтандыру жүйесін қолдану аясын кеңейтіп, оның ұшқышсыз жүйелер нарығындағы бәсекеге қабілеттілігін арттыра алады.

Пайдаланылған дереккөздердің тізімі

1. Smith, J. (2018). Quadcopter Dynamics, Simulation, and Control. Springer International Publishing.

2. Arduino. (2021). Arduino Uno [Online]. Доступно по ссылке: https://www.arduino.cc/en/Guide/ArduinoUno

3. InvenSense. (2021). MPU-6050 Product Specification [Online]. Доступно по ссылке: https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf

4. Flysky. (2021). Flysky i6 User Manual [Online]. Доступно по ссылке: https://www.flysky-cn.com/uploadfile/2016/0919/20160919115607839.pdf

5. López-García, A. (2019). PID Control for Quadcopter Roll, Pitch, and Yaw Stabilization. International Journal of Aerospace Engineering, 2019.

6. Khan, A. H., et al. (2020). Quadcopter Design and Simulation: A Comprehensive Survey. International Journal of Advanced Robotic Systems, 17(2), 1-19.

7. Patel, M. (2017). Development of a Quadcopter Flight Controller Using Arduino. International Journal of Engineering and Advanced Technology, 6(6), 271-275.

8. http://www.brokking.net/ymfc-al_main.html

ҚОСЫМША А

```
#include <Wire.h>
#include <EEPROM.h>

float pid_p_gain_roll = 1.3;
float pid_i_gain_roll = 0.04;
float pid_d_gain_roll = 18.0;
int pid_max_roll = 400;
float pid_p_gain_pitch = pid_p_gain_roll;
float pid_i_gain_pitch = pid_i_gain_roll;
float pid_d_gain_pitch = pid_d_gain_roll;
int pid_max_pitch = pid_max_roll;
float pid_p_gain_yaw = 4.0;
float pid_i_gain_yaw = 0.02;
float pid_d_gain_yaw = 0.0;
int pid_max_yaw = 400;
boolean auto_level = true;
byte last_channel_1, last_channel_2, last_channel_3, last_channel_4;
byte eeprom_data[36];
byte highByte, lowByte;
volatile int receiver_input_channel_1, receiver_input_channel_2, receiver_input_channel_3, receiver_input_channel_4;
int counter_channel_1, counter_channel_2, counter_channel_3, counter_channel_4, loop_counter;
int esc_1, esc_2, esc_3, esc_4;
int throttle, battery_voltage;
int cal_int, start, gyro_address;
int receiver_input[5];
int temperature;
int acc_axis[4], gyro_axis[4];
float roll_level_adjust, pitch_level_adjust;
long acc_x, acc_y, acc_z, acc_total_vector;
unsigned long timer_channel_1, timer_channel_2, timer_channel_3, timer_channel_4, esc_timer, esc_loop_timer;
unsigned long timer_1, timer_2, timer_3, timer_4, current_time;
unsigned long loop_timer;
double gyro_pitch, gyro_roll, gyro_yaw;
double gyro_axis_cal[4];
float pid_error_temp;
```

```
float pid_i_mem_roll, pid_roll_setpoint, gyro_roll_input, pid_output_roll, pid_last_roll_d_error;
float pid_i_mem_pitch, pid_pitch_setpoint, gyro_pitch_input, pid_output_pitch, pid_last_pitch_d_error;
float pid_i_mem_yaw, pid_yaw_setpoint, gyro_yaw_input, pid_output_yaw, pid_last_yaw_d_error;
float angle_roll_acc, angle_pitch_acc, angle_pitch, angle_roll;
boolean gyro_angles_set;
void setup(){
  for(start = 0; start <= 35; start++)eeprom_data[start] = EEPROM.read(start);
  start = 0;
  gyro_address = eeprom_data[32];
  Wire.begin();
  TWBR = 12;
  DDRD |= B11110000;
  DDRB |= B00110000;
  digitalWrite(12,HIGH);
  while(eeprom_data[33] != 'J' || eeprom_data[34] != 'M' || eeprom_data[35] != 'B')delay(10);
 if(eeprom_data[31] == 2 || eeprom_data[31] == 3)delay(10);
  set_gyro_registers();
  for (cal_int = 0; cal_int < 1250 ; cal_int ++){
    PORTD |= B11110000;
    delayMicroseconds(1000);
    PORTD &= B00001111;
    delayMicroseconds(3000);
  }
  for (cal_int = 0; cal_int < 2000 ; cal_int ++){
    if(cal_int % 15 == 0)digitalWrite(12, !digitalRead(12));
    gyro_signalen();
    gyro_axis_cal[1] += gyro_axis[1];
    gyro_axis_cal[2] += gyro_axis[2];
    gyro_axis_cal[3] += gyro_axis[3];
    PORTD|= B11110000;
    delayMicroseconds(1000);
    PORTD &= B00001111;
    delay(3);
  }gyro_axis_cal[1] /= 2000;
gyro_axis_cal[2] /= 2000;
gyro_axis_cal[3] /= 2000;
PCICR |= (1 << PCIE0);
```

```
PCMSK0 |= (1 << PCINT0);
PCMSK0 |= (1 << PCINT1);
PCMSK0 |= (1 << PCINT2);
PCMSK0 |= (1 << PCINT3);
while(receiver_input_channel_3 < 990 || receiver_input_channel_3 > 1020 ||
receiver_input_channel_4 < 1400){
  receiver_input_channel_3 = convert_receiver_channel(3);
  receiver_input_channel_4 = convert_receiver_channel(4);
  start ++;
  PORTD |= B11110000;
  delayMicroseconds(1000);
  PORTD &= B00001111;
  delay(3);
  if(start == 125){
    digitalWrite(12, !digitalRead(12));
    start = 0;
  }
}
start = 0;
battery_voltage = (analogRead(0) + 65) * 1.2317;
loop_timer = micros();
digitalWrite(12,LOW);
}
void loop(){
  gyro_roll_input = (gyro_roll_input * 0.7) + ((gyro_roll / 65.5) * 0.3);
  gyro_pitch_input = (gyro_pitch_input * 0.7) + ((gyro_pitch / 65.5) * 0.3);
  gyro_yaw_input = (gyro_yaw_input * 0.7) + ((gyro_yaw / 65.5) * 0.3);
  angle_pitch += gyro_pitch * 0.0000611;
  angle_roll += gyro_roll * 0.0000611;
  angle_pitch -= angle_roll * sin(gyro_yaw * 0.000001066);
  angle_roll += angle_pitch * sin(gyro_yaw * 0.000001066);
  acc_total_vector = sqrt((acc_x*acc_x)+(acc_y*acc_y)+(acc_z*acc_z));
  if(abs(acc_y) < acc_total_vector){
    angle_pitch_acc = asin((float)acc_y/acc_total_vector)* 57.296;
  }
  if(abs(acc_x) < acc_total_vector){
    angle_roll_acc = asin((float)acc_x/acc_total_vector)* -57.296;
  }
  angle_pitch_acc -= 0.0;
  angle_roll_acc -= 0.0;
  angle_pitch = angle_pitch * 0.9996 + angle_pitch_acc * 0.0004;
```

```
  angle_roll = angle_roll * 0.9996 + angle_roll_acc * 0.0004;
  pitch_level_adjust = angle_pitch * 15;
  roll_level_adjust = angle_roll * 15;
  if(!auto_level){
    pitch_level_adjust = 0;
    roll_level_adjust = 0;
  }
  if(receiver_input_channel_3 < 1050 && receiver_input_channel_4 < 1050)start = 1;
  if(start == 1 && receiver_input_channel_3 < 1050 && receiver_input_channel_4 >
1450){
    start = 2;
    angle_pitch = angle_pitch_acc;
    angle_roll = angle_roll_acc;
    gyro_angles_set = true;
pid_i_mem_roll = 0;
pid_last_roll_d_error = 0;
pid_i_mem_pitch = 0;
pid_last_pitch_d_error = 0;
pid_i_mem_yaw = 0;
pid_last_yaw_d_error = 0;
}
if(start == 2 && receiver_input_channel_3 < 1050 && receiver_input_channel_4 >
1950)start = 0;
pid_roll_setpoint = 0;
if(receiver_input_channel_1 > 1508)pid_roll_setpoint = receiver_input_channel_1 -
1508;
else if(receiver_input_channel_1 < 1492)pid_roll_setpoint =
receiver_input_channel_1 - 1492;
pid_roll_setpoint -= roll_level_adjust;
pid_roll_setpoint /= 3.0;
pid_pitch_setpoint = 0;
if(receiver_input_channel_2 > 1508)pid_pitch_setpoint = receiver_input_channel_2 -
1508;
else if(receiver_input_channel_2 < 1492)pid_pitch_setpoint =
receiver_input_channel_2 - 1492;
pid_pitch_setpoint -= pitch_level_adjust;
pid_pitch_setpoint /= 3.0;
pid_yaw_setpoint = 0;
if(receiver_input_channel_3 > 1050){
  if(receiver_input_channel_4 > 1508)pid_yaw_setpoint = (receiver_input_channel_4
- 1508)/3.0;
```

```
  else if(receiver_input_channel_4 < 1492)pid_yaw_setpoint = (receiver_input_channel_4 - 1492)/3.0;
}
calculate_pid();
battery_voltage = battery_voltage * 0.92 + (analogRead(0) + 65) * 0.09853;
if(battery_voltage < 1000 && battery_voltage > 600)digitalWrite(12, HIGH);
throttle = receiver_input_channel_3;
if (start == 2){
  if (throttle > 1800) throttle = 1800;
  esc_1 = throttle - pid_output_pitch + pid_output_roll - pid_output_yaw;
  esc_2 = throttle + pid_output_pitch + pid_output_roll + pid_output_yaw;
  esc_3 = throttle + pid_output_pitch - pid_output_roll - pid_output_yaw;
  esc_4 = throttle - pid_output_pitch - pid_output_roll + pid_output_yaw;
  if (battery_voltage < 1240 && battery_voltage > 800){
    esc_1 += esc_1 * ((1240 - battery_voltage)/(float)3500);
    esc_2 += esc_2 * ((1240 - battery_voltage)/(float)3500);
    esc_3 += esc_3 * ((1240 - battery_voltage)/(float)3500);
    esc_4 += esc_4 * ((1240 - battery_voltage)/(float)3500);
  }
  if (esc_1 < 1100) esc_1 = 1100;
  if (esc_2 < 1100) esc_2 = 1100;
  if (esc_3 < 1100) esc_3 = 1100;
  if (esc_4 < 1100) esc_4 = 1100;
  if(esc_1 > 2000)esc_1 = 2000;
  if(esc_2 > 2000)esc_2 = 2000;
  if(esc_3 > 2000)esc_3 = 2000;
  if(esc_4 > 2000)esc_4 = 2000;
}
else{
  esc_1 = 1000;
  esc_2 = 1000;
  esc_3 = 1000;
  esc_4 = 1000;
}
if(micros() - loop_timer > 4050)digitalWrite(12, HIGH);
while(micros() - loop_timer < 4000);
loop_timer = micros();
PORTD |= B11110000;
timer_channel_1 = esc_1 + loop_timer;
timer_channel_2 = esc_2 + loop_timer;
timer_channel_3 = esc_3 + loop_timer;
```

```
timer_channel_4 = esc_4 + loop_timer;
gyro_signalen();
while(PORTD >= 16){
  esc_loop_timer = micros();
  if(timer_channel_1 <= esc_loop_timer)PORTD &= B11101111;
  if(timer_channel_2 <= esc_loop_timer)PORTD &= B11011111;
  if(timer_channel_3 <= esc_loop_timer)PORTD &= B10111111;
  if(timer_channel_4 <= esc_loop_timer)PORTD &= B01111111;
}
}
ISR(PCINT0_vect){
  current_time = micros();
  if(PINB & B00000001){
    if(last_channel_1 == 0){
      last_channel_1 = 1;
      timer_1 = current_time;
    }
  }
  else if(last_channel_1 == 1){
    last_channel_1 = 0;
    receiver_input[1] = current_time - timer_1;
  }
  if(PINB & B00000010){
    if(last_channel_2 == 0){
      last_channel_2 = 1;
      timer_2 = current_time;
    }
  }
  else if(last_channel_2 == 1){
    last_channel_2 = 0;
    receiver_input[2] = current_time - timer_2;
  }
  if(PINB & B00000100){
    if(last_channel_3 == 0){
      last_channel_3 = 1;
      timer_3 = current_time;
    }
  }else if(last_channel_3 == 1){
    last_channel_3 = 0;
    receiver_input[3] = current_time - timer_3;
  }
```

```
  if(PINB & B00001000 ){
    if(last_channel_4 == 0){
      last_channel_4 = 1;
      timer_4 = current_time;
    }
  }
  else if(last_channel_4 == 1){
    last_channel_4 = 0;
    receiver_input[4] = current_time - timer_4;
  }
}
void gyro_signalen(){
  if(eeprom_data[31] == 1){
    Wire.beginTransmission(gyro_address);
    Wire.write(0x3B);
    Wire.endTransmission();
    Wire.requestFrom(gyro_address,14);
    receiver_input_channel_1 = convert_receiver_channel(1);
    receiver_input_channel_2 = convert_receiver_channel(2);
    receiver_input_channel_3 = convert_receiver_channel(3);
    receiver_input_channel_4 = convert_receiver_channel(4);
    while(Wire.available() < 14);
    acc_axis[1] = Wire.read()<<8|Wire.read();
    acc_axis[2] = Wire.read()<<8|Wire.read();
    acc_axis[3] = Wire.read()<<8|Wire.read();
    temperature = Wire.read()<<8|Wire.read();
    gyro_axis[1] = Wire.read()<<8|Wire.read();
    gyro_axis[2] = Wire.read()<<8|Wire.read();
    gyro_axis[3] = Wire.read()<<8|Wire.read();
  }
  if(cal_int == 2000){
    gyro_axis[1] -= gyro_axis_cal[1];
    gyro_axis[2] -= gyro_axis_cal[2];
    gyro_axis[3] -= gyro_axis_cal[3];
  }
  gyro_roll = gyro_axis[eeprom_data[28] & 0b00000011];
  if(eeprom_data[28] & 0b10000000)gyro_roll *= -1;
  gyro_pitch = gyro_axis[eeprom_data[29] & 0b00000011];
  if(eeprom_data[29] & 0b10000000)gyro_pitch *= -1;
  gyro_yaw = gyro_axis[eeprom_data[30] & 0b00000011];
```

```
  if(eeprom_data[30] & 0b10000000)gyro_yaw *= -1;
  acc_x = acc_axis[eeprom_data[29] & 0b00000011];
  if(eeprom_data[29] & 0b10000000)acc_x *= -1;
  acc_y = acc_axis[eeprom_data[28] & 0b00000011];
  if(eeprom_data[28] & 0b10000000)acc_y *= -1;
  acc_z = acc_axis[eeprom_data[30] & 0b00000011];
  if(eeprom_data[30] & 0b10000000)acc_z *= -1;
}
void calculate_pid(){
  pid_error_temp = gyro_roll_input - pid_roll_setpoint;
  pid_i_mem_roll += pid_i_gain_roll * pid_error_temp;
  if(pid_i_mem_roll > pid_max_roll)pid_i_mem_roll = pid_max_roll;
  else if(pid_i_mem_roll < pid_max_roll * -1)pid_i_mem_roll = pid_max_roll * -1;
  pid_output_roll = pid_p_gain_roll * pid_error_temp + pid_i_mem_roll +
pid_d_gain_roll * (pid_error_temp - pid_last_roll_d_error);
  if(pid_output_roll > pid_max_roll)pid_output_roll = pid_max_roll;
  else if(pid_output_roll < pid_max_roll * -1)pid_output_roll = pid_max_roll * -1;
  pid_last_roll_d_error = pid_error_temp;
  pid_error_temp = gyro_pitch_input - pid_pitch_setpoint;
  pid_i_mem_pitch += pid_i_gain_pitch * pid_error_temp;
  if(pid_i_mem_pitch > pid_max_pitch)pid_i_mem_pitch = pid_max_pitch;
  else if(pid_i_mem_pitch < pid_max_pitch * -1)pid_i_mem_pitch = pid_max_pitch *
-1;
  pid_output_pitch = pid_p_gain_pitch * pid_error_temp + pid_i_mem_pitch +
pid_d_gain_pitch * (pid_error_temp - pid_last_pitch_d_error);
  if(pid_output_pitch > pid_max_pitch)pid_output_pitch = pid_max_pitch;
  else if(pid_output_pitch < pid_max_pitch * -1)pid_output_pitch = pid_max_pitch *
-1;
  pid_last_pitch_d_error = pid_error_temp;
  pid_error_temp = gyro_yaw_input - pid_yaw_setpoint;
  pid_i_mem_yaw += pid_i_gain_yaw * pid_error_temp;
  if(pid_i_mem_yaw > pid_max_yaw)pid_i_mem_yaw = pid_max_yaw;
  else if(pid_i_mem_yaw < pid_max_yaw * -1)pid_i_mem_yaw = pid_max_yaw * -
1;
  pid_output_yaw = pid_p_gain_yaw * pid_error_temp + pid_i_mem_yaw +
pid_d_gain_yaw * (pid_error_temp - pid_last_yaw_d_error);
  if(pid_output_yaw > pid_max_yaw)pid_output_yaw = pid_max_yaw;
  else if(pid_output_yaw < pid_max_yaw * -1)pid_output_yaw = pid_max_yaw;
  pid_last_yaw_d_error = pid_error_temp;
}
int convert_receiver_channel(byte function){
```

```
  byte channel, reverse;
  int low, center, high, actual;
  int difference;
  channel = eeprom_data[function + 23] & 0b00000111;
  if(eeprom_data[function + 23] & 0b10000000)reverse = 1;
  else reverse = 0;
  actual = receiver_input[channel];
  low = (eeprom_data[channel * 2 + 15] << 8) | eeprom_data[channel * 2 + 14];
  center = (eeprom_data[channel * 2 - 1] << 8) | eeprom_data[channel * 2 - 2];
  high = (eeprom_data[channel * 2 + 7] << 8) | eeprom_data[channel * 2 + 6];
  if(actual < center){
    if(actual < low)actual = low;
    difference = ((long)(center - actual) * (long)500) / (center - low);
    if(reverse == 1)return 1500 + difference;
    else return 1500 - difference;
  }
  else if(actual > center){
    if(actual > high)actual = high;
    difference = ((long)(actual - center) * (long)500) / (high - center);
    if(reverse == 1)return 1500 - difference;
    else return 1500 + difference;
  }
  else return 1500;
}
void set_gyro_registers(){
  if(eeprom_data[31] == 1){
    Wire.beginTransmission(gyro_address);
    Wire.write(0x6B);
    Wire.write(0x00);
    Wire.endTransmission();
    Wire.beginTransmission(gyro_address);
    Wire.write(0x1B);
    Wire.write(0x08);
    Wire.endTransmission();
    Wire.beginTransmission(gyro_address);
    Wire.write(0x1C);
    Wire.write(0x10);
    Wire.endTransmission();
    Wire.beginTransmission(gyro_address);
    Wire.write(0x1A);
    Wire.write(0x03);
```

```
  Wire.endTransmission();
} }
```