ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

Қ.И. Сәтбаев атындағы Қазақ ұлттық техникалық зерттеу университеті

Ақпараттық және телекоммуникациялық технологиялар институты

Электроника, телекоммуникациялар және ғарыштық технологиялар кафедрасы

Абдраманов Б.Ж.

Arduino микроконтроллері негізінде квадрокоптерді құру

ДИПЛОМДЫҚ ЖҰМЫС

5В071900 – «Радиотехника, электроника және телекоммуникация» мамандығы

Алматы 2019

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

Қ.И. Сәтбаев атындағы Қазақ ұлттық техникалық зерттеу университеті

Ақпараттық және телекоммуникациялық технологиялар институты

Электроника, телекоммуникациялар және ғарыштық технологиялар кафедрасы

ҚОРҒАУҒА ЖІБЕРІЛДІ
Кафедра меңгерушісі
техн. ғыл. канд-ы
_____ Е. Т. Таштай
«__15__» ____05____ 2019 ж.

ДИПЛОМДЫҚ ЖҰМЫС

Тақырыбы: «Arduino микроконтроллері негізінде квадрокоптерді құру»

5В071900 – «Радиотехника, электроника және телекоммуникация» мамандығы

Орындаған: Абдраманов Б.Ж.

Рецензия беруші
ҚазҰАУ, ЭҮжА каф.
доктор PhD.,
қауымдастырылған профессор
_____ Әлібек Н.Б.
«__06__» ____05____ 2019 ж.

Ғылыми жетекші
ЭТжҒТ кафедрасының
қауымдастырылған профессор
_____ Л. Б. Илипбаева
«__25__» ____04____ 2019 ж.

Алматы 2019

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ
Қ. И. Сәтбаев атындағы Қазақ ұлттық техникалық зерттеу университеті
Ақпараттық және телекоммуникациялық технологиялар институты
Электроника, телекоммуникация және ғарыштық технологиялар кафедрасы
5В071900 – Радиотехника, электроника және телекоммуникация

БЕКІТЕМІН
Кафедра меңгерушісі,
техн.ғыл.канд.
_____ Е. Таштай
«20» ___01___ 2019 ж.

Дипломдық жұмыс орындауға
ТАПСЫРМА

Білім алушы *Абдраманов Бекарыс Жангелдинулы*

Тақырыбы «Arduino микроконтроллеріне негізделген квадрокоптерді құру»

Университет ректорының *«16» қазан 2018 ж. № 1162-б* бұйрығымен бекітілген.

Дипломдық жұмыстың бастапқы берілістері:

а) Arduino AtMega628P микроконтроллерінің квадрокоптерді басқару жүйесіндегі атқаратын қызметі;

ә) Квадрокоптердің максималды ұшу уақытына қатысты аккумуляторлық батарея сыйымдылығының есебі;

б) 2.4 ГГц жиілікті Flysky FS-i6 қабылдағышы пен таратқышының максималды ақпарат алмасу қашықтығы.

Аяқталған жұмысты тапсыру мерзімі «25» сәуір 2019 ж.

Дипломдық жұмыста қарастырылатын мәселелер тізімі:

1) Қазіргі заманғы мультироторлы жүйелерге әдебиеттік шолу;

2) Квадрокоптерді басқару платасының негізгі блоктарын зерттеу;

3) Бағдарламалық қамтамасыз ету және практикалық іске асыру.

Сызбалық материалдар тізімі (міндетті сызбалар дәл көрсетілуі тиіс).

Квадрокоптердің бағдарламалық кодтары қосымшада көрсетілген.

Ұсынылатын негізгі әдебиет 24 атау.

Дипломдық жұмысты (жобаны) дайындау

КЕСТЕСІ

Бөлімдер атауы, қарастырылатын мәселелер тізімі	Ғылыми жетекшіге және кеңесшілерге көрсету мерзімі	Ескерту
Қазіргі заманғы мультироторлы жүйелерге әдебиеттік шолу	20.01.2019 - 01.03.2019	Әлок
Квадрокоптерді басқару платасының негізгі блоктарын зерттеу	02.03.2019-02.04.2019	Әлок
Бағдарламалық қамтамасыз ету және практикалық іске асыру	01.04.2019–15.04.2019	Әлок

Дипломдық жұмыс (жоба) бөлімдерінің кеңесшілері мен норма бақылаушының аяқталған жұмысқа(жобаға) қойған

қолтаңбалары

Бөлімдер атауы	Кеңесшілер (аты, әкесінің аты, тегі, ғылыми дәрежесі, атағы)	Қол қойылған күні	Қолы
Норма бақылау	PhD докторы, ЭТжҒТ каф.сениор-лекторы Тайсариева К.Н.	14.05.19	

Ғылыми жетекшісі _____ Л. Б. Илипбаева

(қолы)

Тапсырманы орындауға алған білім алушы _____ Б. Абдраманов

Күні "14" 05 2019 ж.

АҢДАТПА

Дипломдық жұмыста квадрокоптердің ұшу реттегіші ретінде қолданылатын Arduino микроконтроллері қарастырылады.

Бұл дипломдық жұмыста квадрокоптердің көлбеу бұрыштарын анықтау үшін 3 осьтік гироскоппен және акселерометр қолданылады және олармен жұмыс істеу үшін Arduino микроконтроллері пайдаланылады, сонымен қатар ол тұрақты токтың коллекторсыз қозғалтқышын басқару және квадрокоптердің орналасуын басқарады. Микроконтроллерді қашықтан басқару үшін 2.4 ГГц жиілікті таратқыш қолданылады. Сонымен қатар математикалық модельдеу, квадрокоптерлерге қолданылатын автоматты басқару теориясы, датчиктермен жұмыс қарастырылады. Төртвинтті мультикоптердің математикалық моделі және оның бұрыштарын басқару жүйесі әзірленеді.

АННОТАЦИЯ

В дипломной работе рассматривается микроконтроллер Arduino, используемый в качестве полетного контроллера квадрокоптера.

Для определения углов наклона квадрокоптов в этой работе используется 3-осевой гироскоп и акселерометр, и для работы с ними используется микроконтроллер Arduino, который также управляет расположением квадрокоптера и управление бесколлекторным двигателем постоянного тока. Для дистанционного управления микроконтроллером используется передатчик частоты 2.4 ГГц. Также рассматриваются математическое моделирование, теория автоматического управления, используемая для квадрокоптеров, работа с датчиками. Разрабатывается математическая модель четырехвинтовых мультикоптеров и система управления ее углами.

ANNOTATION

The research paper examines an Arduino microcontroller, used as the flight controller of the quadcopter.

To determine the angles of inclination quadrocopters in this work uses a 3-axis gyroscope and accelerometer, and to work with them uses the Arduino microcontroller, and controls the location of quadrocopters and management of ENDLESS-brushed DC motor. The 2.4 GHz frequency transmitter is used for remote control of the microcontroller. Mathematical modeling, theory of automatic control used for quadrocopters, work with sensors are also considered. The mathematical model of four-screw multicop and the control system of its angles is developed.

МАЗМҰНЫ

КІРІСПЕ

Қазіргі заманғы электрониканы қолдану, ең алдымен, электронды құрылғыларды әзірлеу процесін жеңілдетеді. Құрылғы және оның орындайтын міндеттері неғұрлым күрделі болған сайын, оны заманауи микроконтроллердің көмегімен жобалау оңайырақ болады. Ал қазіргі таңдағы ең өзекті компоненттердің бірі Arduino микроконтроллері болып табылады.

Дипломдық жұмыстың мақсаты Arduino микроконтроллері негізінде жұмыс істейтін квадрокоптердің моделін құрып іске асыру болып табылады.

Arduino микроконтроллерінің қолдану мүмкіндіктерін зерттеу мақсатында Arduino Uno микроконтроллері негізіндегі квадрокоптерді әзірлеу ойластырылады. Квадрокоптерді құрмастан бұрын қазіргі заманғы мультироторлы жүйелерге аналитикалық шолу жасалынып, олардың заманауи шешімдері қарастырылады.

Квадрокоптерді басқару платасының негізгі блоктарын зерттеу барысында компоненттердің техникалық сипаттамалары мен квадрокоптерді әзірлеу приниптері анықталады және ұшу уақыты бойынша батареяның қажетті сыйымдылығы есептеледі.

Сонымен қатар, MATLAB ортасында квадрокоптердің басқару жүйесінің моделін құрып, қажетті көрсеткіштер алынады. Бұл жүйе таратқыштың ауқымындағы кез келген орынның ішінде отырып, қашықтағы жүйемен немесе таратқышпен басқарылады. Бұл тұжырымдама қадағалау қызметін жеңілдетеді.

1 Қазіргі заманғы мультироторлы жүйелерге әдебиеттік шолу

Мультикоптер – бұл қарама-қарсы бағытта диагональды айналатын салмақ түсетін бұрандалардың еркін саны бар ұшу аппараты. Еркін салмақ винттері бар ұшу аппараттары мультикоптер деп аталады. Мультикоптерлер тікұшақ жасау кезінде дами бастады, бірақ осы бағытты одан әрі дамытуға бір қозғалтқыштың барлық винттерге айналу конструкциясының күрделілігі және классикалық тікұшақтарға арналған қисық аппаратының өнертабысы кедергі жасады. Мультикоптерлердің қарқынды дамуы 21-ғасырда ұшқышсыз ұшу аппараттары ретінде басталды. Дизайн қарапайымдылығының арқасында мультикоптер әуесқойлық модельдеуде кеңінен қолданылады. Оларды камералармен және GPS қабылдағыштармен жабдықтайды, бұл биіктіктен жергілікті жерді сапалы түсіруді жүргізуге мүмкіндік береді, GPS болуы мультикоптердің автономды қозғалысының бағытын жүргізуге мүмкіндік береді[1].

1.1 Мультироторлы жүйелердің заманауи шешімдері

Мультикоптерлер шағын өлшемге ие, жеңіл, өте маневрлік, салыстырмалы түрде арзан және оңай. Бұл қасиеттер осы ұшу аппараттарын әскерде, құтқару қызметтерінде, зерттеу жұмыстарында, ұсақ габаритті жүктерді жеткізу және т.б. үшін алғышарттар болып табылады. Қазір әуесқой мультикоптерлер миномет атыстарын түзету және жерді барлау үшін кейбір армиялармен пайдаланылады. Бұл растауды интернетте түрлі ресурстардан көптеген бейнероликтерден табуға болады. Бірақ мультикоптарда бір үлкен минус бар – ұшу уақыты аз. Мультикоптерлердің көп бөлігі әрбір бұранда бір-бірден қуатты шоғырландырылмаған электрроторлармен жабдықталған. Аккумуляторлардың аз сыйымдылығы-бұл мультикоптерлерді жаппай пайдалану жолында тұрған маңызды проблемалардың бірі. Бір батарея зарядында мультикоптердің орташа ұшу уақыты 20 минутты құрайды, бұл шағын көрсеткіш болып табылады.

Мультикоптер келесі негізгі компонеттерден тұрады:

- ұшу контроллері;
- рама;
- моторлар;
- жылдамдық реттегіштері;
- аккумуляторлар;
- бұрандалар;
- радио басқару пульті;
- радиоқабылдағыш.

Мультикоптердің басты бөлігі оның ұшу контроллері (ми) болып табылады, ол радиобасқару командаларын қозғалтқыштардың командаларына

аударады. Тұрақты ұшуды қамтамасыз ету үшін мультикоптерлер көлбеу және бұрылу бұрышын бекітетін үш гироскоптармен міндетті түрде жабдықталады. Бұл үш бұрыш крен, тангаж және айналу деп аталады. Крен – аппараттың ұзына бойлық осі (мұрыннан құйрыққа дейін өтетін ось) айналасында бұрылуы. Тангаж – бұл оның көлденең осінің айналасындағы бұрылыс (мұрын тұмсығы, құйрығы). Айналу – тік осьтің айналасында бұрылыс. Қосалқы құрал ретінде, кейде акселерометр қолданылады, оның деректері процессорға абсолюттік көлденең жағдайды орнатуға мүмкіндік береді, сондай-ақ аппаратты қажетті биіктікте бекітуге мүмкіндік беретін барометр датчигі. Бұдан басқа, сонар немесе УД-қашықтықты дәл ұстап тұру және автономды қону үшін, сондай-ақ кедергілерді ұшып өту үшін пайдаланылатын қашықтықтан өлшеуіш қолданылады. Ұшу бағытын алдын ала, компьютерден және айналымға жазуға мүмкіндік беретін GPS-қабылдағыш мультикоптерге берілген нүктелер бойынша ұшуды автоматты түрде жасауға, сондай-ақ басқару радиосигналын жоғалтқан жағдайда аппаратты ұшу орнына автоматты түрде қайтаруға мүмкіндік береді[2].

Аппарат орталығына қатысты қозғалтқыштардың саны мен орналасуымен ерекшеленетін мультикоптерлердің көптеген түрлері бар.

Негізгі түрлері:

- үшроторлы (трикоптер)
- төртроторлы (квадрокоптер)
- алтыроторлы (гексакоптер)
- сегізроторлы (октокоптер)

Әр сорттардың өз артықшылықтары мен кемшіліктері бар.

Трикоптер – үшроторлы мультикоптер түрі (1.1-сурет).

Ерекшеліктері: екі алдыңғы мотор қарама-қарсы бағытта, ал үшіншісі кез келген жаққа айналады. Моторлардың бірі оське, қозғалмалы платформада орналасқан, бұрылу бұрышы бұрылысымен өзгеретін - сол сияқты өз осінің айналасында аппаратты бұрып бұруды жүзеге асырады.

Артықшылықтары: төмен салмақ, ықшамдық, бұл ең арзан мультикоптер, өйткені оларды салу үшін тек 3 мотор және 3 жылдамдық реттегіші қажет.

Мұндай аппараттардың кемшіліктері күрделі конструкция, шағын жүк көтергіштігі және бір қозғалтқыш істен шыққан кезде аппарат сөзсіз құлайды.

Сурет 1.1 – Трикоптердің сыртқы көрінісі

Квадрокоптер – төртроторлы мультикоптер түрі (1.2-сурет).
Артықшылықтары: бұрылыс механизмінсіз қарапайым дизайн.
Кемшіліктері: бір қозғалтқыш істен шыққан кезде сөзсіз құлдырау керек.

Сурет 1.2 - Квадрокоптердің сыртқы бейнесі

Гексакоптер алтыроторлы мультикоптер түрі (1.3-сурет).
Олар квадрокоптердің барлық оң қасиеттері бар.
Артықшылықтары: бір мотордың істен шығуы кезінде аппарат құлайды, үлкен жүк көтергіштігі, тұрақтылығы, желге аз сезімталдығы.
Кемшіліктері: қысқа ұшу уақыты, үлкен габариттер және жоғары баға.

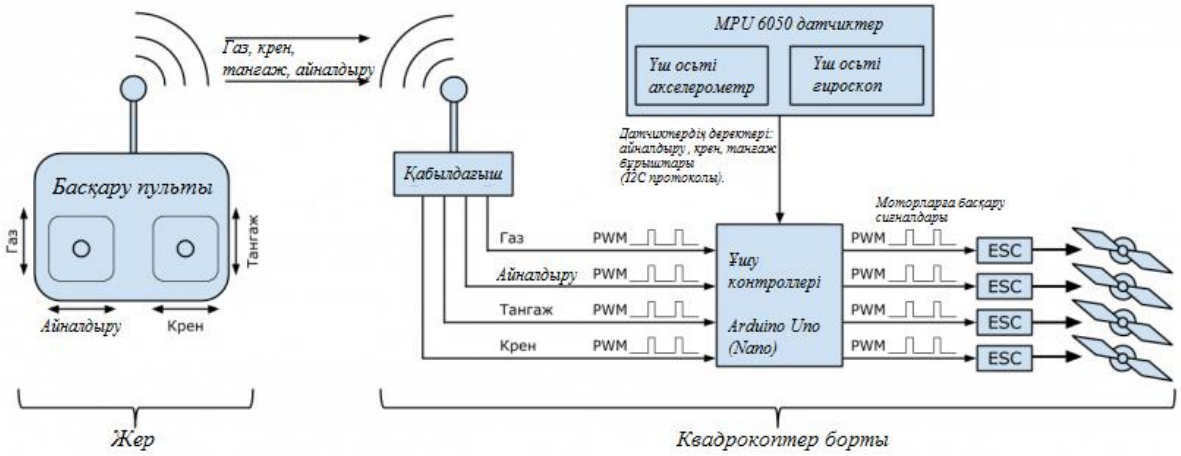
Сурет 1.3 - Гексакоптердің үлгісі

Октокоптер – сегізроторлы мультикоптер түрі (1.4-сурет).
Артықшылықтары: өте тұрақты және желге сезімтал емес, бір немесе екі мотор істен шыққан кезде – аппарат құлайды! Өте үлкен жүк көтергіштігі (16-17 кг дейін). Тек осы аппараттарға кәсіби, ауыр және қымбат фото-бейне камераларды ауаға көтеруді сенуге болады.
Бұл аппараттардың кемшіліктері: үлкен энергия тұтыну және жоғары баға[3].

Сурет 1.4 - Октокоптердің сыртқы көрінісі

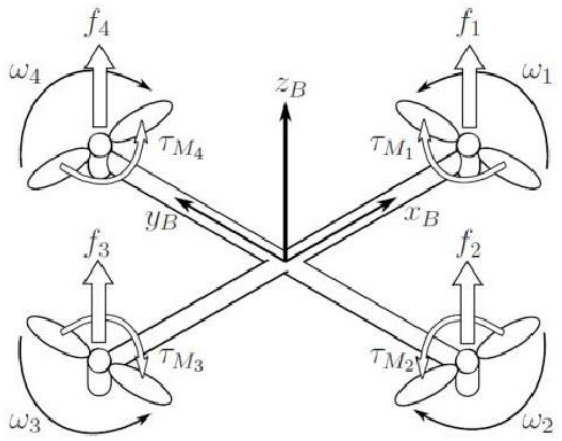
1.2 Квадрокоптер жұмысының жалпы принциптері

Квадрокоптердің ұшу жүйесі жоғары қысымды ауа көтеру құбылыстарының принципі бойынша жұмыс істейді. Пропеллер жоғары қысыммен ішкі ауаны мәжбүр етеді, оған көтеру күш құрылған және нәтижесінде бүкіл жүйеде қолданбалы әрекет реакция заңы. Бұл күш Жер гравитация күші үстем болғанда, бүкіл жүйе ауада ұша бастайды. Бірақ пропеллерді айналдыру мәселесі бар. Егер бұрамаларды сағат тілі бағытында айналдырсақ, осы айналымның арқасында айналдыру сәті бүкіл жүйеде бір бағытта қолданылатын болады. Дәл сол сияқты, егер бұрамаларды сағат тіліне қарсы бағытта айналдырсақ, айналмалы сәт бүкіл жүйе бойынша жасалады және барлық жүйе сағат тіліне қарсы айнала бастайды. Бұл мәселені шешу үшін екі бұранданы сағат тіліне, ал қалған екі бұранданы сағат тіліне қарсы айналдырамыз. Бұл құбылыстар қарама-қарсы бағытта айналмалы момент жасайды және олар теңдестірілген және жүйе тұрақтандырылған әзірге ұшады. Екі негізгі құбылыстар квадрокоптерді тарту және айналдыру үшін қолданылады. Квадрокоптер тарту жасайтын және квадрокоптерді жоғары көтерілуге көмектесетін қозғалтқышқа бекітілген төрт бұранданы пайдаланады[4].



Сурет 1.5 – Квадрокоптердің функционалдық схемасы

Квадрокоптер қозғалысы оған берілген кіріс мәндері (x, y, z, θ, ϕ, ψ) негізінде айқындалады. Бұрандалармен біріктірілген төрт қозғалтқыштың екі қозғалтқышы сағат тілі (CW) бойынша, ал екіншісі - сағат тіліне (CCW) қарсы айналады. Осылайша, квадрокоптердің қозғалысы негізінен үш қозғалыспен бақыланады. Бұл орын ауыстырулар ретінде жіктеледі.



Сурет 1.6 - Квадрокоптердің координаттық үлгісі

1 және 3 қозғалтқыштары сағат тіліне қарсы айналуды жасайды, ал 2 және 4 қозғалтқыштары сағат тіліне айналады, осылайша коптердің айналмалы сәттерінің теңгерімі жүреді. Төртвинтті коптерде алты еркіндік дәрежесі бар (массалар орталығының үш қозғалыс осі және үш айналу осі). Барлық алты еркіндік дәрежесі коптердің бұрамаларының айналу жылдамдығы арқылы басқарылуы мүмкін.

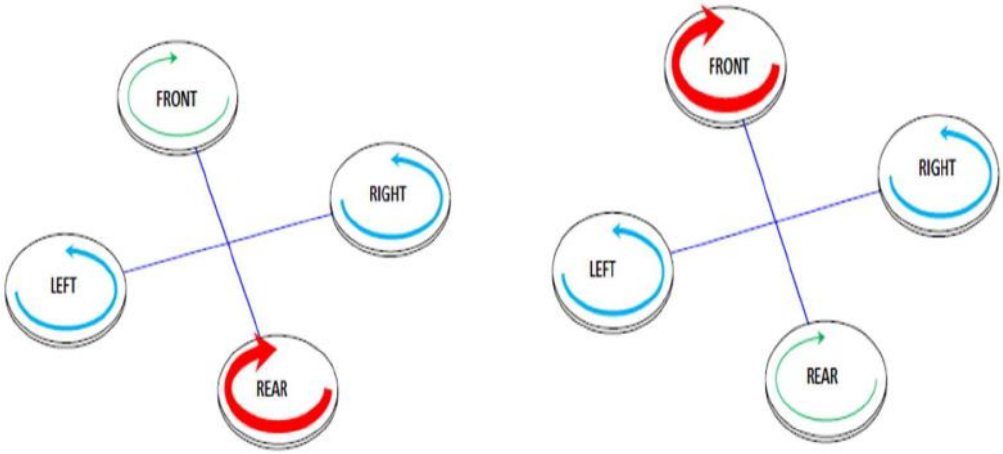
Жоғары және төмен қозғалыс барлық қозғалтқыштардың айналу жылдамдығын арттыру және төмендету есебінен жүзеге асырылады. Бұрандалардың жиынтық күші дронның салмағын теңестіргенде, ол тұрақты биіктікті қолдайды. Барлық төрт бұрандалардың жылдамдығы арасындағы арақатынасты өзгертіп, коптер қозғалысының әр түрлі нұсқаларын қамтамасыз етуге болады (1.6-суретті қараңыз).

1.3 Ұшу барысында орындалатын негізгі операциялар

Крен бұрышы бойынша басқару – мұндай қозғалыс түрін алу үшін алдыңғы бұранданың жылдамдығы артады, ал артқы бұранданың жылдамдығы тиісті түрде азаяды. Бұрандалардың жылдамдықтарын осындай өзгерісті жасай отырып, өзгермейтін тартым мен момент қалады. Бірақ

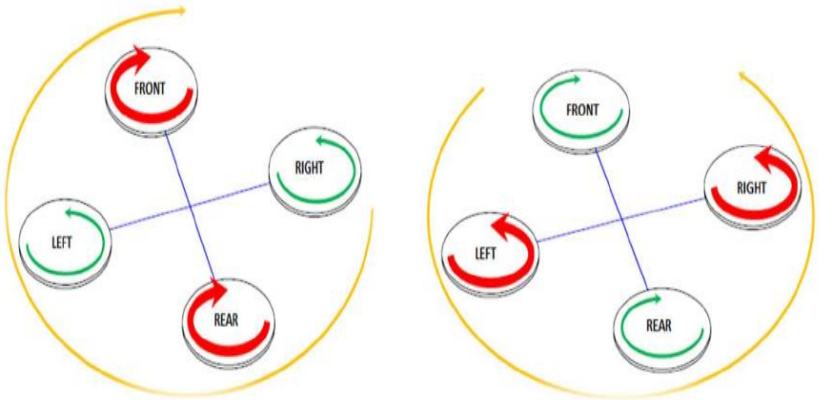
тартым өзгермейтініне қарамастан, коптердің көлбеу бұрышы өзгереді, онда күш тепе-теңдігі бұзылады және квадрокоптер төмендей бастайды.

Тангаж бұрышы бойынша басқару – крен бұрышы бойынша басқару сияқты қағидат бойынша жүзеге асырылады, бүйірлік бұрандалардың бірінің айналу жылдамдығы азаяды және сонымен бірге сол шамаға екінші бүйірлік бұранданың жылдамдығы артады (1.7-суретті қараңыз). Алдыңғы жағдайдағыдай, күш тепе-теңдігінің бұзылуына байланысты квадрокоптер төмендейді[5].



Сурет 1.7 - Коптердің тангаж бұрышы бойынша алға және артқа қозғалу үлгісі

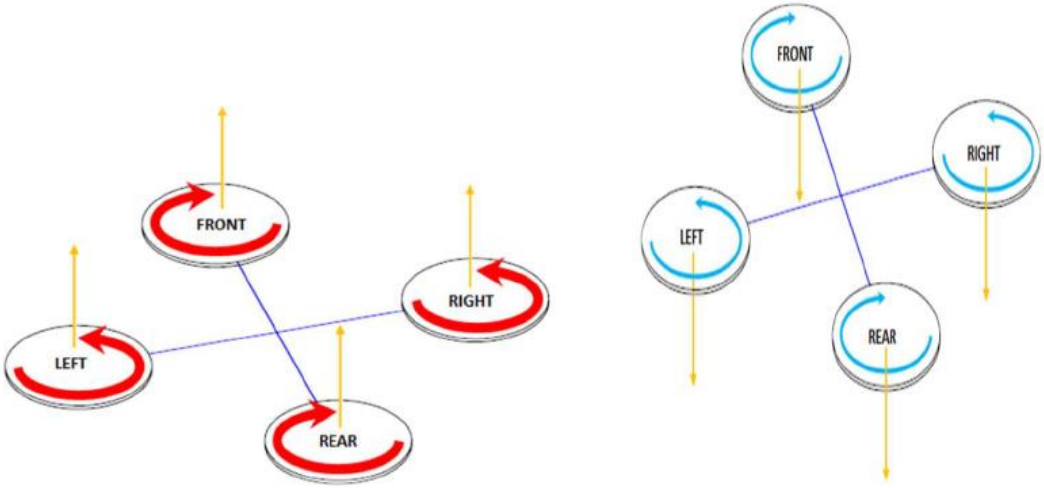
Айналу бұрышы бойынша басқару – квадрокоптердің солға немесе оңға қозғалысы ретінде анықталады және ол таратқыштың дроссельді таяқшасымен басқарылады. Айналу бұрышы квадрокоптердің бағытын шешеді (1.8 –суретті қараңыз).



Сурет 1.8 - Квадрокоптердің оңға және солға айналу қозғалысы

Айналу бұрышы бойынша басқару айналмалы моменттердің тепе-теңдігін бұзу есебінен жүреді. Мысалы, айналу үшін оңға айналуды сағат тілі

бойынша жасайтын бұрандалардың айналу жылдамдығы азаяды және сол мәнге айналуды сағат тіліне қарсы жасайтын бұрандалардың айналу жылдамдығы артады. Тартым бірдей қалады, бірақ айналмалы моменттер тепе-теңдігінің бұзылуы коптердің бұрылуына әкеледі.



Сурет 1.9 – Квадрокоптердің ұшу және қону принципі

Квадрокоптер негізгі контроллер ретінде Arduino Atmega328 микроконтроллерін пайдаланады. Бұл жүйе контроллер платасына бекітілген бір қабылдағышты және таратқышты пайдаланады, ол квадрокоптердің қозғалысын бақылайды. Аппараттық қамтамасыз ету Atmega328 микроконтроллері бар Arduino қарапайым платасынан, пропеллерлер, электронды жылдамдық контроллерінен, таратқыш пен қабылдағыштан және теңдестірілген ұшуға арналған гироскоптан тұрады.

Квадрокоптерді басқару жүйесі, негізінен, қашықтан басқару құралы мен автоматты толықтыру функциясы бар адамдарды басқарады[6].

Қашықтықтан басқару қабылдағышымен ұшақтар оператордың қашықтағы жұмысын аяқтау үшін команда қабылдайды, сонымен қатар ұшуды қабылдау және автоматты реттеу функциясы бар. Барлық басқару жүйесі өзіне қуат беру модулі, сымсыз байланыс модулі, датчик модулі, қозғалтқыш жетегінің модулі, басқару модулі кіреді.

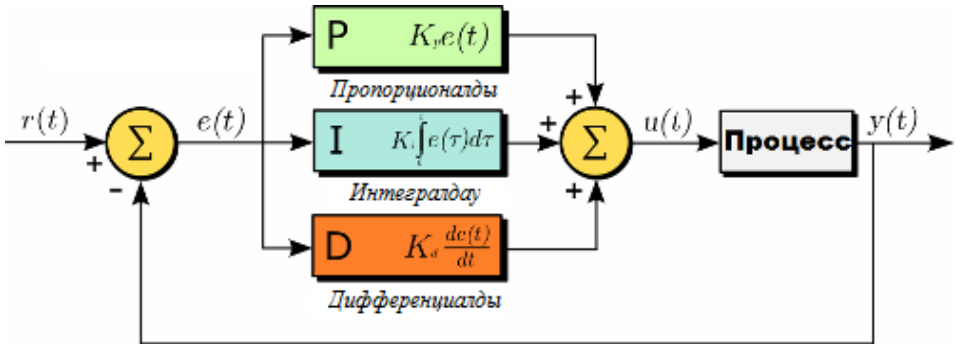
Қабылдағыш таратқыштың сымсыз модулі қашықтан басқару пультінен келіп түсетін басқарушы сигналды қабылдайды, онда басқару контроллер модуліне беріледі. Жылдамдату датчигінің үш осьтік модулі және гироскоптар нақты уақыт режимінде коптердің ұшу жағдайын қадағалайды және деректер контроллерге беріледі.

Контроллер модулі сенсорлық модуль мен сымсыз байланыс модулін қабылдағаннан кейін, деректерге, мақсаттық коэффициентке және деректердің нақты коэффициентіне негізделген, бірқатар күрделі алгоритмдерді, арақатынасы мен позициясын атқарады, квадрокоптерден алынған ақпарат бақылау мәнін есептейді, драйвер тізбегі арқылы сәйкес PWM сигналына айналады. Ұшу аппаратын тұрақты ұшудың және телеметрияның төрт

роторын ұстап тұрып, сымсыз байланыс модулі арқылы жер үсті бақылау станциясына жіберіледі[7].

ПИД-реттеуші (ағылш. P-proportional, I-integral, D — derivative) – кері байланыс буынымен жабдықталған басқару контурында қолданылатын құрылғы деп аталады. Бұл реттеушілер автоматты жүйелердегі басқару сигналын қалыптастыру үшін пайдаланылады, онда өтпелі процестердің сапасы мен дәлдігіне жоғары талаптарға қол жеткізу қажет[7].

ПИД реттеушінің басқару сигналы үш компоненттің қосылуы нәтижесінде алынған: бірінші - қателік сигналының шамасына пропорционал, екінші сигнал сигналының интегралына, ал үшіншіден оның туындысына пропорционалды (1.13-суретті қараңыз). Егер осы үш компоненттің қайсыбірі қосу процесіне қосылмаған болса, онда реттеуші - ПИД емес, жай ғана пропорционал, тепе-тең-тең немесе интегралдаушыларға пропорционал болады[8].



Сурет 1.13 – ПИД реттеушінің құрылымдық сұлбасы

2 Квадрокоптерді басқару платасының негізгі блоктары

2.1 Компоненттердің техникалық сипаттамалары

HJ-450 рамасы – бұл шыны талшықты квадрокоптер рамасы, ол өте қарапайым және шеңберді құру үшін оңай-жалын дөңгелегі бірнеше маңызды себептер үшін ең танымал рамалардың бірі:

- Бағасы салыстырмалы арзан
- Материалы мықты
- Орталық пластина қуаттың үлестіру ақысы ретінде екі еселенеді
- Дизайны өте жақсы ойластырылған

Қабылдағыш үшін орын көп, басқару платасы, ЭЖК және батарея, монтаждау нұсқаларымен және GoPro немесе басқа камераны орнату үшін орын бар (2.1-сурет).

Нарықтағы ең танымал квадрокоптердің бірі ретінде, шасси, қалта аспа және т. б сияқты таңдау қосалқы бөлшектер мен аксессуарлардың кең таңдауы бар.

Сурет 2.1 – HJ-450 рамасының бейнесі

Әрбір квадрокоптер немесе басқа да көп моторлы ұшақ барлық басқа компоненттерді орналастыру үшін рамаға мұқтаж. Рамаларды, сондай-ақ, алюминий немесе балшық табағын пайдалана отырып, үйде салуға болады. Бірақ нәтижелер жасалған кадрлардан эстетикалық жағынан да, ұшу атрибуттары тұрғысынан да ерекшеленетін болады[9].

Электрондық жылдамдық контроллері (ЭЖК) немесе The electronic speed control (ESC) – бұл кез-келген уақытта тез айналатын қозғалтқыштар дейді(2.2-сурет). Квадрокоптер үшін төрт ЭЖК қажет, біреуі әрбір қозғалтқышқа қосылған. Содан кейін ЭЖК тікелей аккумуляторға сымдардың жгуты немесе тарату ақысы арқылы қосылады. Көптеген ЭЖК батареяның элиминаторы орнатылған схемасымен бірге жеткізіледі, ол сізге ұшуды басқару ақысы және радиоқабылдағыш сияқты заттарды тікелей батареяға

қоспай қосуға мүмкіндік береді. Квадрокоптердегі қозғалтқыштар нақты ұшуға жету үшін нақты жылдамдықпен айналуы тиіс болғандықтан, ЭЖК өте маңызды. Бұл ЭЖК жаңарту жиілігін өзгертеді, сондықтан қозғалтқыштар ЭЖК секундына әлдеқайда көп нұсқауларды алады, осылайша, квадрокоптердің мінез-құлқына үлкен бақылау бар.

Техникалық сипаттамалары:

- Ток: тұрақты 30A / 35A жарылады
- Кернеу ауқымы: 2-4s Li-Po
- Батарея элиминаторы: 5V, 3A сызықтық
- Салмағы: 35 г

Сурет 2.2 – ESC-30A электронды жылдамдық контроллерінің сыртқы бейнесі

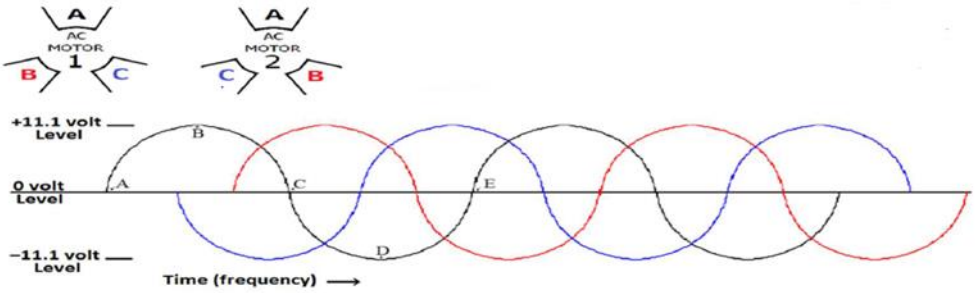
ЭЖК әдетте ең жоғары токқа есептелген. Әдетте, рейтинг жоғары болған сайын, ЭЖК соншалықты көп және ауыр, әдетте, ұшақтардағы салмақ пен балансты есептеу кезінде фактор болып табылады. Көптеген қазіргі заманғы ЭЖК никель-металлгидридті, литий-ионды полимерлі және литий-железофосфатты батареяларды қолдайды. Батарея түрі және қосылған ұяшықтар саны контроллерге немесе автономды блок ретінде кіріктірілген болсын, батарея элиминаторы сұлбасын таңдауда маңызды мәселе болып табылады. Қосылған ұяшықтардың көп саны қуатты төмендетеді және, демек, егер ол кернеудің сызықтық реттегішін қолданса, біріктірілген батареяның элиминаторы қолдайтын сервоприводтардың аз санына әкеледі. Ауыстыру реттегішін пайдаланып жақсы жобаланған батареяның элиминаторы ұқсас шектеулерге ие болмауы керек.

Жұмыс істеу принципі:

Реттеуіш кернеу емес, жиілігі айнымалы ток қозғалтқышының жылдамдығын басқарады. Егер 11.1 вольт батареяны қуат жүйесіне қоссаңыз, сізде 11.1 вольт бар. Жобада пайдаланатын щеткасы жоқ қозғалтқыштар, шындық 3-фазалық айнымалы ток электр қозғалтқыштары. Қозғалтқыштар айнымалы токпен жұмыс істейді. Трапециевидті толқын генераторының реттеуіші. Ол 3 жеке толқынды шығарады (қозғалтқышқа әрбір сым үшін бір). Қозғалтқыштың жылдамдығы кернеумен немесе күшейткіштермен ештеңе жоқ, бірақ оның орнына оған берілетін ток уақыты. 3 фазада трапециевидті толқынның ұзындығын (жиілігін) ұлғайтып және азайта отырып, ЭЖК қозғалтқышты жылдам және баяу айналдырады. ЭЖК толқындар жасау үшін

фазалардың полярлығын ауыстырады. Бұл дегеніміз, кез келген осы орама арқылы кернеу бір бағытта, содан кейін екіншісінде "кезекпен" ағады. Бұл әрбір орамның магнит өрісінде пушпуль әсерін жасайды, мотор мөлшері мен салмағы үшін күшті етеді. Қозғалтқыш және оған салынатын жүктеме – ЭЖК және батареядан күшейткіштің тартымын анықтайтын нәрсе.

Төменде суретте 3 полюсті 2 қозғалтқыш бар. Олардың орамдары "А", "В" және "С" полюсі ретінде белгіленген. Кестеде (2 қозғалтқышының астында) ЭЖК қозғалтқыш жетегі үшін генерациялайтын 3 жеке толқындар көрсетілген (2.3-сурет). Кесте сигналдар уақытының кернеуге қатынасын көрсетеді. Кестедегі қара толқын – бұл "а"орамына жіберілетін сигнал. Қызыл сигнал "в" орамасына барады, ал көк сигнал "с"орамасына барады. Егер сіз “1” айнымалы тоқтың қозғалтқышын және “2” айнымалы тоқтың қозғалтқышын қараңыз болса, және орамдарға жіберілетін кестеде көрсетілген сигналдар; кез-келген екі қозғалтқыш қосылымдарын ауыстырған кезде, толқындар орамаға соққы тәртібін өзгертеді және бұл қозғалтқыш бағытын өзгертеді[10].

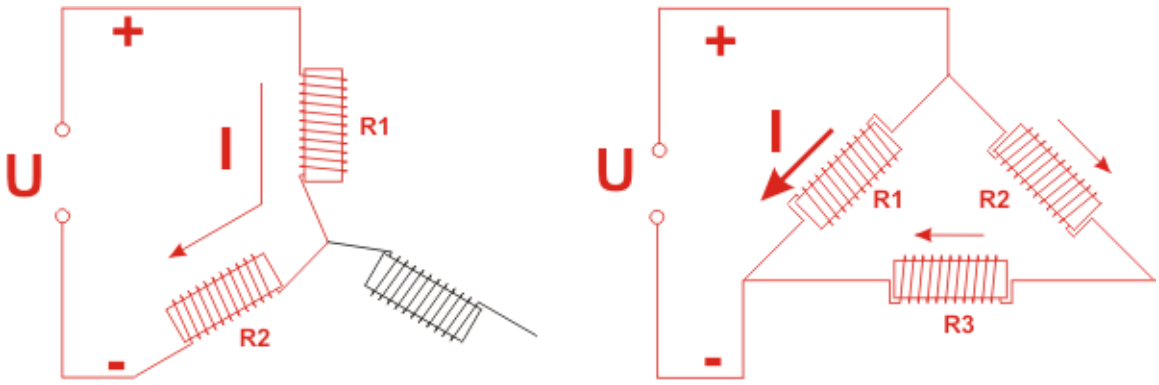


Сурет 2.3 – Вариациялық қозғалтқыштардағы жиіліктің өзгеруі

Щеткасыз тұрақты ток қозғалтқышы – тұрақты магниттері бар ротордан және орамдары бар статордан тұратын мотор.

Моторларда негізгі параметр — kV бар. Бұл мотор жасайтын минутына айналу саны, берілген вольт кернеу көрсеткіші. Бұл мотордың қуаты емес, бұл "беріліс саны". KV аз болса, соғұрлым айналым аз, бірақ моменті жоғары болады. Сол қуатта kV көп болса, соғұрлым көп айналым және төмен момент болады. Моторды таңдау кезінде ол штаттық режимде ең жоғарғы 50% қуаттылығымен жұмыс істейтініне бағдарланады. KV неғұрлым көп болса, соғұрлым жақсы деп ойлауға болмайды, типтік 3S-батареясы бар коптерлер үшін ұсынылатын Сан 700-ден 1000 kV-ге дейінгі диапазонда болады.

Коллекторсыз қозғалтқыш орамдарын жұлдыз немесе үшбұрыш (дельта) схема бойынша жалғайды (2.4-сурет).



Сурет 2.4 – Щеткасыз тұрақты ток қозғалтқыштарының орамдарының схемасы

Жұлдыз схемасымен қосылған кезде ток екі орамнан кейін өтеді. Қорытынды кедергі екі орамның қарсыласу жиынтығына тең R=R1+R2. Сәйкесінше I=U/(R1+R2) орама арқылы өтетін максималды мүмкін ток. Тұтынылатын қуат P=U*I. Егер кернеу 10 В, ал орамасының кедергісі 1 Ом болса, I=10/(1+1)=5А. Қуат шығыны P=10*5=50 Вт.

Үшбұрыш схемасымен қосылған кезде ток барлық орамдар арқылы өтеді. R=(R1*(R2+R3))/(R1+R2+R3) орама кедергісі R = (R1 * (R2 + R3) / (R1 + R2 + R3). Сәйкесінше, орамдар арқылы өтетін максималды мүмкін ток I=U/((((R1*(R2+R3)))/(R1+R2+R3).

Сол кернеу мен орамдардың кедергісі кезінде ток I=10/((1*(1+1))/(1+1+1))=15А. Қуат шығыны P=10*15=150 Вт.

Үшбұрыш схемасымен қосылған кезде қозғалтқыш айналымы да өседі. Үшбұрыш қосылған қозғалтқыш орамдары жұлдызбен қосылғаннан артық қызады.

Ораманы жұлдыздан үшбұрышқа ауыстыру арқылы мүлдем басқа сипаттамалары бар қозғалтқышты алуға болатыны анық.

Қосудың ұзақ режимі бар жоғары моментті қозғалтқыштарда жұлдызды қолданған жөн. Қысқа мерзімді режимде жұмыс істейтін, неғұрлым жоғары айналымды талап ететін қозғалтқыштарда үшбұрышты қолданған жөн.

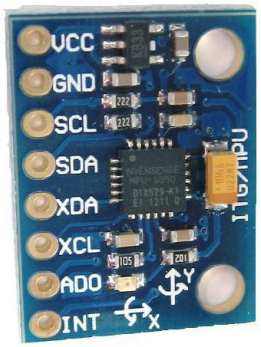
Кейде электр көлігінде старт пен екпін орамдарды жұлдызбен қосқан кезде орындалады (себебі бұл қосу біліктің жоғары моментін, бірақ аз айналымдарды қамтамасыз етеді), екпіннен кейін үшбұрышқа ауыстырып қосу орындалады (айналым жоғары, сәт аз). Бұл бастапқы сипаттамаларды сақтай отырып, қозғалтқыштың айналым диапазонын арттыруға мүмкіндік береді[11].

Техникалық сипаттамалары:

- Айналу моменті (КВ) : 1300КВ / 1000КВ / 850КВ / 750КВ
- Күші: 930г / 890г / 875г / 866г
- Мотор өлшемі (мм): Ф28 х 30
- Батарея: 2-4s Li-Po

Сурет 2.5 – 1000 КВ тұрақты ток щеткасыз қозғалтқышының бейнесі

MPU-6050 гироскоп және акселерометр – сенімді және дәл деректерді қамтамасыз ететін өлшеу құралы (2.6-сурет). Ол шағын, жұқа, өте төмен қуатты, 3 осьтік акселерометр және гироскоп. Құрылғы өте дәл, өйткені әрбір арна үшін аппараттық құралдардың 16-бит аналогты-сандық түрлендіргіші бар[8]. Ол гравитациядан туындаған статикалық үдеуді, ауытқуды анықтау міндеттерінде, сондай-ақ қозғалыс немесе соққы нәтижесінде динамикалық үдеуді өлшейді. Датчиктің "сандық қозғалыс процессоры" бар, ол микробағдарламалық қамтамасыздандырумен бағдарламалай болады және датчиктің мәндерімен күрделі есептеулерді жасай алады.



Сурет 2.6 – IMU/MPU 6050 гироскоп-акселерометрдің сыртқы бейнесі

FlySky FS-i6 пульты мен қабылдағышы – квадрокоптерді басқаруға мүмкіндік береді (2.7-сурет). Көптеген қолайлы модельдер бар, бірақ базалық квадрокоптер үшін кем дегенде төрт арна қажет. Электроникада және телекоммуникацияларда радиотаратушы - бұл антеннаның көмегімен радиотолқындар өндіретін электрондық құрылғы.

Таратқыштың өзі антеннаға берілетін радиожиілік айнымалы токты генерациялайды. Бұл айнымалы токпен қозғалғанда антенна радиотолқындарды шығарады. Таратқыш термині әдетте байланыс мақсаттары үшін радиотолқындарды генерациялайтын жабдықпен шектеледі; немесе радиолокациялық және навигациялық таратқыштар сияқты радиолокациялар. Таратқыш басқа Электрондық құрылғының ішіндегі жеке электрондық

жабдық немесе электр тізбегі болуы мүмкін. Таратқыш және қабылдағыш бір блокта трансивер деп аталады.

Көптеген таратқыштардың мақсаты – қашықтықтағы ақпараттың радиобайланысы болып табылады. Ақпарат таратушыға микрофоннан аудио (дыбыстық) сигнал, бейне (ТВ) теледидар камерасынан сигнал, немесе сымсыз желілік құрылғыларда компьютерден сандық сигнал сияқты электрондық сигнал түрінде беріледі. Таратқыш жиі тасушы туындататын радиотолқындарды шығаратын радиожиілік сигналымен алып тастау қажет ақпарат сигналын біріктіреді. Бұл процесс модуляция деп аталады[12].

Радиотаратушы – бұл электр энергиясын батареядан немесе электр желісінен радио жиілікке түрлендіретін электрондық схема, ол бағытын секундына миллион рет өзгертетін айнымалы ток. Мұндай тез айналатын токтағы энергия өткізгіштен (антеннадан) электромагниттік толқындар (радиотолқындар) түрінде сәулеленуі мүмкін.

Радиоқабылдағыш – бұл антеннадан кіріс алатын электрондық схема, талап етілетін радиосигналды осы антеннадан қабылданатын барлық басқа сигналдардан бөлу үшін электрондық сүзгілерді пайдаланады, оны одан әрі өңдеуге сәйкес келетін деңгейге дейін күшейтеді және, ақырында, демодуляция және декодтау арқылы сигналды дыбыс, сурет, сандық деректер, өлшеу мәндері, навигациялық жағдайлар және т.б. сияқты тұтынушының пайдалануы үшін жарамды қалыпқа түрлендіреді. Ол алдымен оларды кодтаған жіберушіден декодталған хабарламалар мен ақпаратты алады. Кейде қабылдағыш декодер қосу үшін модельденеді. Радиоқабылдағыштар сияқты нақты әлемнің қабылдағыштары шулы арнаны кодтау туралы теоремамен болжанғандай, сонша ақпарат алады деп күтуге болмайды.

Радиобасқару пультының тұтқаларының қызметі:

- оң жақ тұтқа крен мен тангажды бақылайды. Басқаша айтқанда, ол квадрокоптеріңізді солға / оңға және артқа / алға жылжытады.
- сол жақ тұтқа айналдыру мен дроссельді бақылайды. Басқаша айтқанда, ол квадрокоптерді сағат тіліне немесе сағат тіліне қарсы айналдырады және сіз ұшатын биіктікті реттейді.

Сурет 2.7 – Flysky FS-i6 пульты мен қабылдағышының сыртқы көрінісі

Жерден квадрокоптерді көтеру үшін дроссельді алғаш рет басқанда, сіз автоматты түрде көлбеу және бір бағытта (немесе бірнеше) ұшады деп байқай аласыз. Бұл элементтер теңгерілмеген кезде орын алады. Оларды теңдестіру үшін, кейбір басқару элементтері кесілуі тиіс[13].

Техникалық сипаттамалары:

- Арналар саны: 6
- Басқару объекттері: тікұшақ, ұшақ, планер
- Модуляция: GFSK
- Код түрі: PCM
- Қуат көзі: ТТ 12В (1.5 AAA*8) салмағы: 680 г
- Антенна ұзындығы: 26 мм.

Пропеллер – тарту, айналдыру қозғалысын түрлендіру арқылы күш беретін желдеткіш түрі (2.8-сурет). Қысымның айырмашылығы фольганың алдыңғы және артқы ауа беттерінің арасында жасалған-нысанды жүздің және сұйықтық (ауа немесе су сияқты) жүздің артындағы жүрісті жылдамдатады. Винт динамикасы Бернулли принципімен де, Ньютон үшінші заңымен де модельдеуі мүмкін. Теңіз пропеллері кейде кең жерде винт қадамы ретінде белгілі. Жалпы, үлкейген тангаж және пропеллер ұзындығы көп ағысты тартады. Сондай-ақ, тангажды тіреудің бір айналу қашықтығы ретінде анықтауға болады. Екі сөзбен айтқанда, жоғары қадам баяу айналуды білдіреді, бірақ сіздің автокөлік жылдамдығын арттырады, ол сондай-ақ көп энергияны пайдаланады.

Квадрокоптердің ұшуы үшін пропеллердің екі түрін қажет. Сағат тілі бойынша (CW) және сағат тіліне қарсы (ACW) пропеллер жұбы қажет. Бұрандалардың мөлшерін соңғы пысықтау туралы ойлану керек.

Ұзындығы мен өрісі туралы шешім қабылдаған кезде, Сіз жақсы баланс табу керек. Жалпы, төмен тангаж нөмірлері көп айналдыру сәтін жасай алады. Моторларға күшті жұмыс істеу қажет емес, сондықтан олар осы түрмен аз ағысты тартады. Егер сіз акробатикамен айналысқыңыз келсе, сізге энергия жүйесіне аз қысым беретін бұранда қажет болады. Бұрандалардың төменгі қадамы тұрақтылықты жақсарту.

Пропеллердің жоғары қадамы турбуленттілікті тудыруы мүмкін ауаның көп мөлшерін жылжытады және ұшақ тоқтап тұру кезінде тербелуге мәжбүр етеді. Егер сіз оны квадрокоптеріңізбен байқасаңыз, пропеллердің төмен қадамын таңдауға тырысыңыз.

Ол ұзындығына келгенде, пропеллер тиімділігі ауамен тіректің байланыс ауданы тығыз байланысты, сондықтан тірек ұзындығын аз арттыру пропеллер тиімділігін арттырады[14].

Сурет 2.8 – Квадрокоптерде пайдаланылатын пропеллер үлгісі (10x45)

Литий-полимерлі (Li-Po) аккумулятор – литий-ионды технологияны және желе тәрізді полимерлі электролитті сұйық орнына қолданады (2.9-сурет). Олар өте жоғары меншікті энергия сыйымдылығын (масса бірлігіне) қамтамасыз етеді және сондықтан ұялы телефондарда, мультикоптерлерде және басқа да ұялы жабдықтарда кеңінен қолданылады.

Батарея таңдау кезінде оның үш параметріне назар аудару керек: миллиампер-сағаттарда өлшенетін сыйымдылық, батарея сыйымдылықтарындағы ең жоғары разряд тогы (C) және ұяшықтар саны (S). Алғашқы екі параметр бір-бірімен байланысты, және оларды ауыстырып жатқанда сіз осы батареяны ұзақ уақыт бере алатындай токтың қанша екенін білуге болады. Мысалы, моторлардың әрқайсысы 10 А және олардың төрт дана тұтынады, ал батареяның 2200 мА*сағ, 30/40C параметрлері бар, осылайша коптерге 4 * 10 А = 40 А талап етіледі, ал батарея 2,2 А * 30 = 66 А немесе 2,2 А * 40 = 88 А 5-10 секунд ішінде бере алады, бұл – аппаратты қоректендіру үшін жеткілікті. Сондай-ақ, бұл коэффициенттер батарея салмағына тікелей әсер етеді.

Ұяшықтар саны (S) батареядағы LiPo-элементтердің санын көрсетеді, әрбір элемент 3,7 В береді, мысалы, 3S-аккумулятор шамамен 11,1 В береді.

Сурет 2.9 – Li-Po 2200 мА*сағ 11,1 В аккумуляторының бейнесі

2.2 Arduino микроконтроллерінің пайдалану мүмкіндіктері

Arduino Uno – Atmega328P чип негізіндегі микроконтроллер платасы (2.10-сурет). Ол 14 сандық кіріс-шығыс тесіктері (6 PWM шығу ретінде

пайдалануға болады), 6 аналогты енгізу, 16 МГц кварц кристалы, USB қосылымы, Jack қуат қосқышы, ICSP коллекторы және қайтару батырмасы бар. Ол микроконтроллерді қолдау үшін қажетті барлық нәрселерді қамтиды; онымен жұмысты бастау үшін USB кабелі немесе AC-DC адаптері немесе батареямен компьютерге қосу керек.

Модульдер мен датчиктерден барлық ақпарат орталық микроконтроллерге беріледі, ол оны консольге шығарады және орнатылған скриптке сәйкес өңдейді.

Arduino микроконтроллерінің басқа платалардан артықшылықтары:

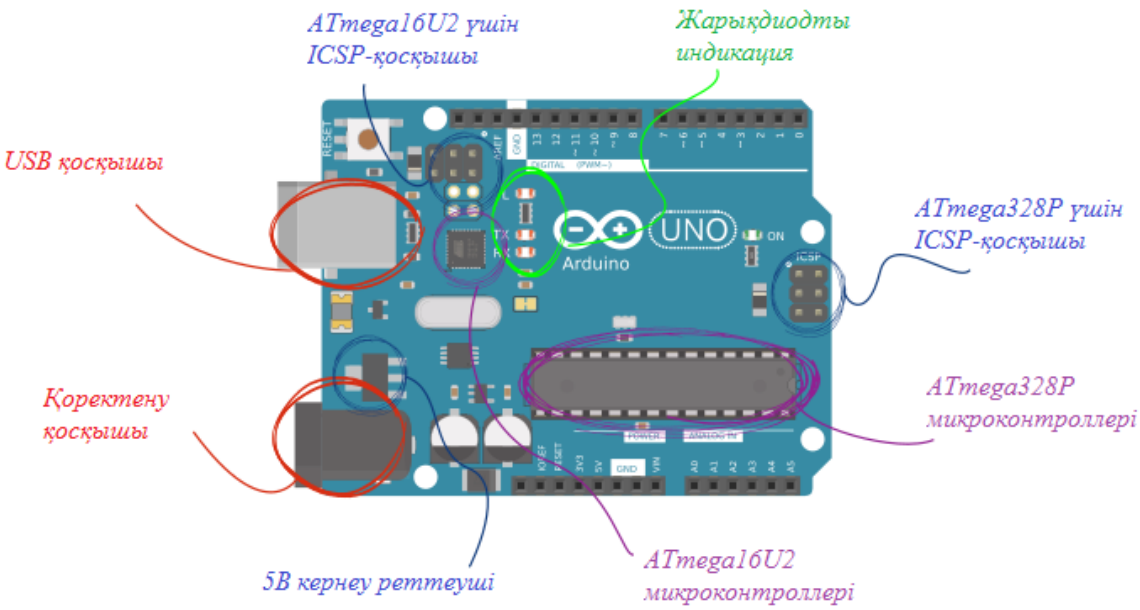
- Тек плата авторларымен ғана емес, сонымен қатар қоғамдастықпен құрылатын кітапханалар. Осының арқасында кез келген тапсырма бойынша қажет құралдарды табуға болады..
- Шағын өлшемді. Бұл соңғы бұйымның корпусында үлкен кеңістік жоқ кәсіби төлем жасауға мүмкіндік береді. Ал габариттері ақылды үйден өз жылыжайын құруға дейін барлық салаларда өте маңызды.
- Көптеген модульдер.

Arduino микроконтроллеріне кез келген қажетті модульді табуға болады. Түтін немесе жарық сенсоры, тіпті шағын динамик болсын. Сонымен қатар, периферияны қауымдастықтың өзі құрады, соның арқасында қосымша микроконтроллерлерді арзан сатып алып үнемдеуге болады[15].

Arduino – бұл жүйенің барлық басқа элементтері қосылатын шағын құрылғы. Arduino тиісті электр сигналдарын бере отырып, онда жазылған скрипттердің көмегімен олардың жұмысын үйлестіруі тиіс. Стандартты МК үшін Arduino сигналы 5 вольт болып табылады – бұл бірлік, ал сигналдың болмауы – нөлдік.

Дәл осы принцип бойынша екілік кодпен бағдарламалау салынған. Бірақ мұндай жүйеден қазір қолданыстан шықты, сондықтан құрылғыға айнымалы ток трансформаторлары мен қосымша резисторларды қосуға болады, өйткені кейбір модульдерге 3.2-4.7 В кернеуі қажет.

Arduino – қоршаған ортамен тығыз байланыста болатын жады және процессоры бар плата. Платасында көптеген желілер бер. Олар арқылы батырмаларға, светодиодтарға, микрофондар мен динамикаларға, электроқозғалтқыштар мен дисплеиге, радиоесептегіштерге (RFID және NFC), ультородыбыстық және лазерлік дальнометрлерге, bluetooth, WiFi және Ethernet модульдерге жалғап, байланыс орнатуға болады. Оған қоса оншақты датчиктермен де жұмыс жасайды.



Сурет 2.10 - Arduino Uno микроконтроллерінің бейнесі

Демек, Arduino – бұл бүкіл жүйенің жұмысын оған салынған кітапханалардың көмегімен үйлестіруге мүмкіндік беретін құрал. Ал, кітапханалардың өздері микроконтроллердің жұмысын толық бақылауды қамтамасыз ететін төмен деңгейлі C++ платформасында жазылған.

Шындығында, Arduino микроконтроллері тек электрлік сигналдарды жібере алады және оларды қосылған модульдерден алады. Алайда, егер микроконтроллерді басқа тұрғыдан қарастырсақ, ол бәріне дерлік қабілетті, оған сапалы кодты қою және қажетті сенсорларды қосу жеткілікті.

Arduino микроконтроллерінің қолдану аялары:

- Ақылды үй. Ақылды үйдің әрбір элементін өз қолымен жасауға болады. Автоматты перделер мен есіктерден сигнализацияға және реттелетін жарықтандыруға дейін.

- Кодтық құлыптар. Жоба қарапайым және жаңадан келгендерге қолайлы. Кез келген сенсорды пайдалану және белгілі бір дыбысталу ырғағына немесе смартфонның жақындауына жауап беретін құлыптарды жасау жеткілікті.

- Автоматтандырылған жылыжайлар.

- Ақпараттарды қашықтыққа жеткізу.

Жобалар саны мыңдаған есе көп, сіз өз қиялыңызды қосу ғана қалады, ал құрал ретінде Arduino қызмет етеді[16].

2.3 Ұшу уақыты бойынша батареяның қажетті сыйымдылығын есептеу

Теңдеумен анықталатын жылдамдықпен қозғалатын ауа ағынында салмағы M ұшу аппаратын жылжыту бойынша A жұмыс (2.1):

$$A = FS = MgV_{dn}t = Mghnt \ [\text{I}] \quad (2.1)$$

Барлық қозғалтқыштардың жиынтық қуаты:

$$N = Mghn \ [\text{Вт}] \quad (2.2)$$

Батарея сыйымдылығы [Ah] бар екенін ескере отырып, I [A] тогы және оның жұмыс уақыты t [c] келесі қатынасымен байланысты болуы мүмкін:

$$I = \frac{3600\ C}{t} \ [A] \quad (2.3)$$

Электр тогының қуатын есептеу үшін N = IU формуласын пайдалана отырып, мұнда U-батареядағы кернеу [В]:

$$C = \frac{Mght}{3600U\eta_{en}} \ [\text{А} * \text{сағ}] \quad (2.4)$$

Мұнда η_{en} – қозғалтқыш қондырғысының пайдалы әсер коэффициенті.

Егер аппараттың іліну уақыты 11 мин (660 с) болса, қозғалтқыш қондырғысының пайдалы әсер коэффициенті η_{en} = 0,6, ал батареядағы кернеу U = 10 В, бұл үшбанкалы Li-Po батареяның жұмыс режимі үшін (2.4) теңдеу бойынша аламыз. Бұл жағдай үшін қажетті батарея сыйымдылығы 2,2 А*сағ құрайды. Аппараттың ұшу уақытын есептеу (2.4) теңдеуден батарея сыйымдылығы және аппараттың басқа да сипаттамалары бойынша ұшу уақытын есептеу арқылы формуланы алу оңай:

$$t = \frac{CU\eta_{en}}{Mghn} 3600 \ [\text{с}] \quad (2.5)$$

Осы аппарат үшін басқа ұқсас жағдайларда, M = 0,94 кг, h = 0,1 м, N =93 с^{-1}, η_{en} = 0,6, U = 10 В, C = 2,9 А*сағ кезінде, ұшуда болу уақыты 13 мин. теңдеу бойынша есептік мән 731,4 с немесе 12,2 мин құрайды. (2.5) формулада M ұшу массасы M_{kv} = 0,7 кг ұшу аппаратының массасын және M_{BAT} батареясының массасын қамтиды, ол өз кезегінде батарея сыйымдылығына байланысты. Бұдан басқа, аппараттың ілінуінің статикалық режиміндегі N қозғалтқыштардың айналым саны оның толық салмағына байланысты болады.

Көрсетілген тәуелділіктер үшін ең кіші квадраттар әдісімен келесі аппроксимациялық формулалар алынды.

Батарея массасы үшін

$$M_{bat} = \frac{aC + b}{1000} = \frac{69C + 16.37}{1000} \ [kg] \quad (2.6)$$

Мұндағы C – батарея сыйымдылығы [A*сағ],
a = 69
b = 16,37 – 30 C дейінгі максималды разрядтау тогы бар әртүрлі Li-Po 3s 48 жұп деректер (массасы және сыйымдылығы) аппроксимация жолымен алынған өлшемдік коэффициенттер.

Бұл коэффициенттердің мәні технологиялардың осы деңгейіне ғана тән екенін атап өткен жөн. Болашақта технологиялардың жетілуіне қарай олардың мәндері өзгеруі мүмкін[17].

(2.6) формулаға сәйкес аппараттың ұшу массасынан әуе винттерінің айналу жиілігіне арналған теңдеу 0,5 дәрежелі көрсеткішпен далалық тәуелділік түрі болуы тиіс. Жобадағы аппарат үшін деректерді аппроксимациялау келесі теңдеуді алуға мүмкіндік берді:

$$n = k(\frac{M_{kv} + M_{bat}}{M_{kv}})^{0.5} = 76(\frac{0.7 + M_{bat}}{M_{kv}})^{0.5} \ [\text{с}^{-1}] \qquad (2.7)$$

(2.6) және (2.7) теңдеулерді (2.5) формулаға қоя отырып, аппараттың іліну уақытының батарея сыйымдылығынан тәуелділігін алуға болады:

$$t(C) = \frac{CU\eta_{en}3600}{ghk\left[M_{kv} + \frac{aC + b}{1000}\right][1 + \frac{aC + b}{1000M_{kv}}]^{0.5}} \ [\text{с}] \qquad (2.8)$$

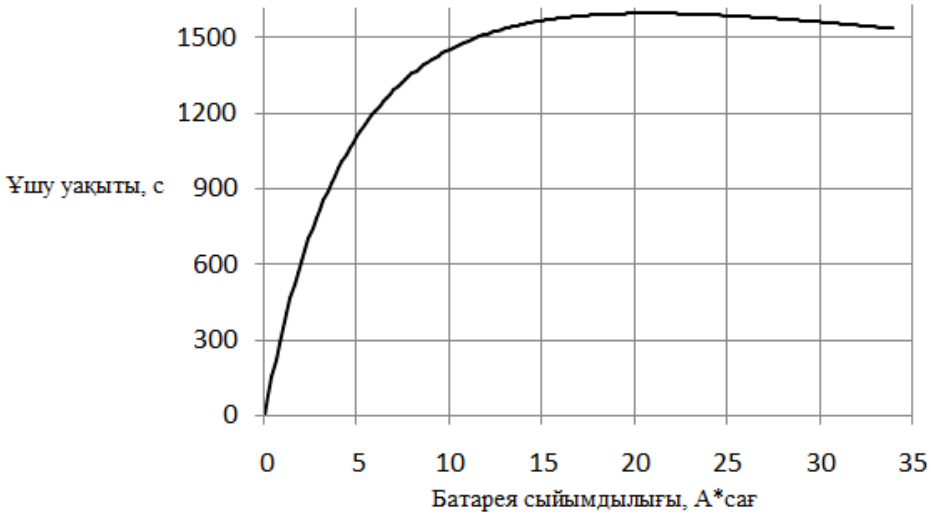
Бұл функция максимумға ие (2.11-сурет), яғни ұшу аппаратының берілген түрі үшін батареяның оңтайлы сыйымдылығы бар – C*, оның ұшудағы максималды уақытын қамтамасыз ететін – t_{max}. Бұл көрсеткіштерді анықтау үшін dt(C)/dC туындысын анықтаймыз және оны нөлге теңестіру керек.

$$\frac{dt(C)}{dC} = \frac{18000000\sqrt{10}\eta_{en}U(-aC + 2b + 2000M_{kv})}{ghkM_{kv}^2(\frac{aC + b + 1000M_{kv}}{M_{kv}})^{5/2}} = 0 \qquad (2.9)$$

$$C^* = \frac{2(b + 1000M_{kv})}{a} \ [A * \text{сағ}] \qquad (2.10)$$

C* өрнегін есептеу үшін өрнектерді (2.8) теңдеуге қойып, аппараттың максималды ұшу уақыты (2.11) t_{max} алуға болады:

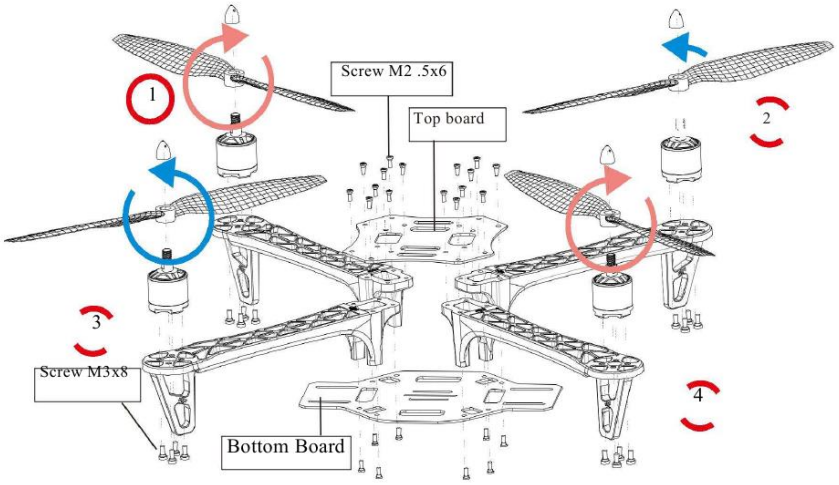
$$t_{max} = \frac{2400(b + 1000M_{kv})U\eta_{en}}{aghk(M_{kv} + \frac{b}{1000})\sqrt{3(1 + \frac{b}{1000M_{kv}})}} \text{ [c]} \qquad (2.11)$$



Сурет 2.11 - Мультироторлы ұшу аппаратының ұшудағы уақыттың батарея сыйымдылығына тәуелділігі

2.4 Квадрокоптерді құру және дәнекерлеу

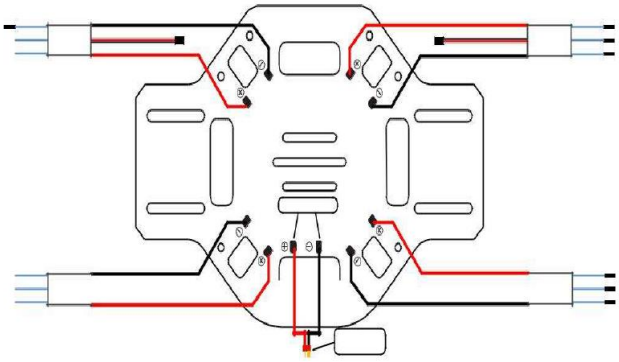
HJ-450 рамасымен орнатылған шасси ESC қосу үшін дәнекерленген болуы тиіс. Шасси электр қуатына арналған PCB басылған тақта ретінде жұмыс істейді (2.12-сурет).



Сурет 2.12 – HJ-450 рамасын құрастыру жолы

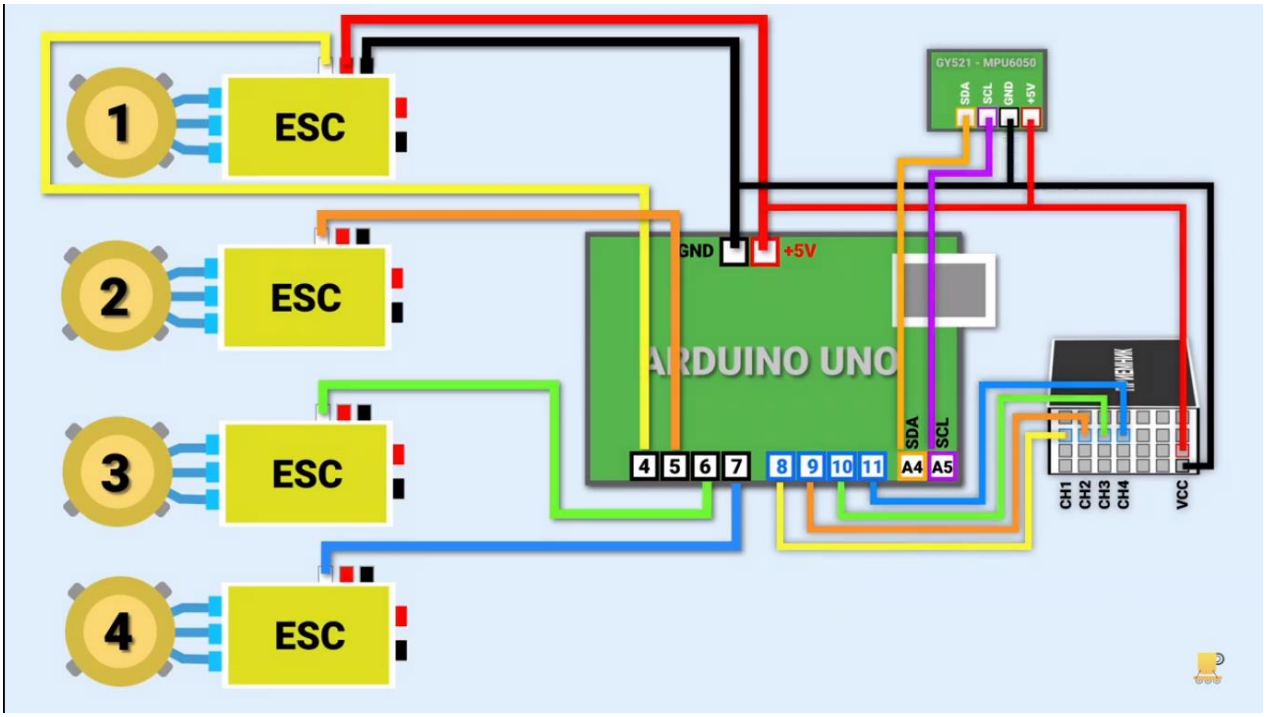
Дәнекерлеу үшін оқшаулау материалын пайдаланылды. Дәнекерлеу кезінде ашық немесе жабық тізбектің жоқ екеніне көз жеткізу керек (2.13-сурет).

4 ЭЖК (ESC) дәнекерлегеннен кейін HJ-450 рамасының шассиіне қосылуы тиіс. Қысқа тұйықталу алу үшін тиісті күтім жасау керек.



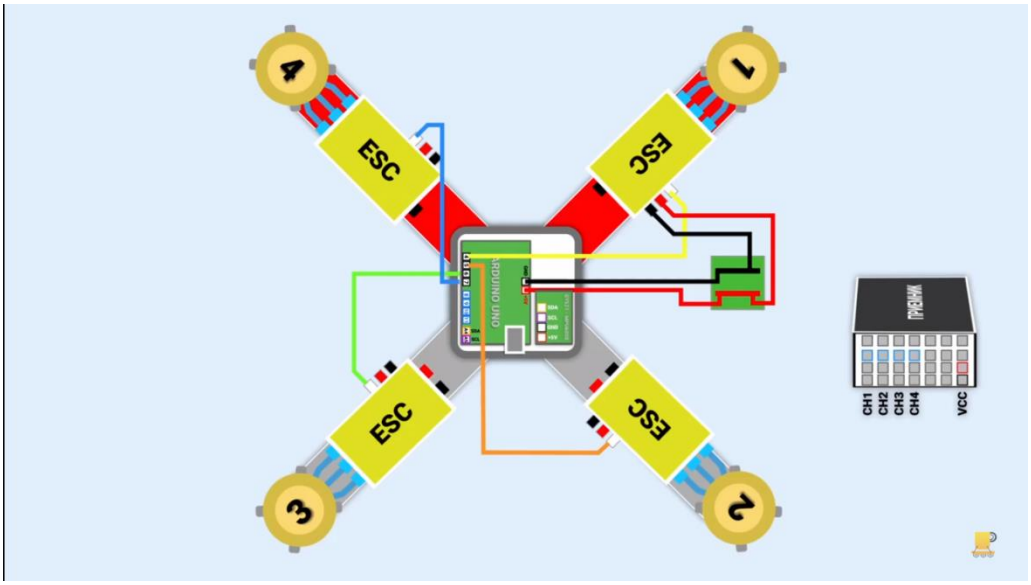
Сурет 2.13 – ЭЖК (ESC) сымдарын шассиге дәнекерлеу жолдары

2.14-суретте квадрокоптердің компоненттерінің бір-бірімен байланысу жолы көрсетілген.



Сурет 2.14 – Квадрокоптер компоненттерінің қосылу сұлбасы

Жоғарыда айтылып кеткендей, 2 қозғалтқыш сағат бағытына қарсы айналатын болса (2 жыне 4) , ал қалған 2 қозғалтқыш сағат тілі бағытымен айналуы тиіс[18]. ЭЖК (ESC) мен қозғалтқыштарды квадрокоптерге орналастыру сұлбасы 2.15-суретте көрсетілген.



Сурет 2.15 – ЭЖК (ESC) мен қозғалтқыштарды квадрокоптерге орналастыру тәртібі

3 Бағдарламалық қамтамасыз ету және практикалық іске асыру

3.1 MATLAB Simulink математикалық пакетінде квадрокоптер басқару жүйесінің моделін құру

Зерттелген теориялық материалды негізге ала отырып, практикалық іске асыруға көшу қажет.

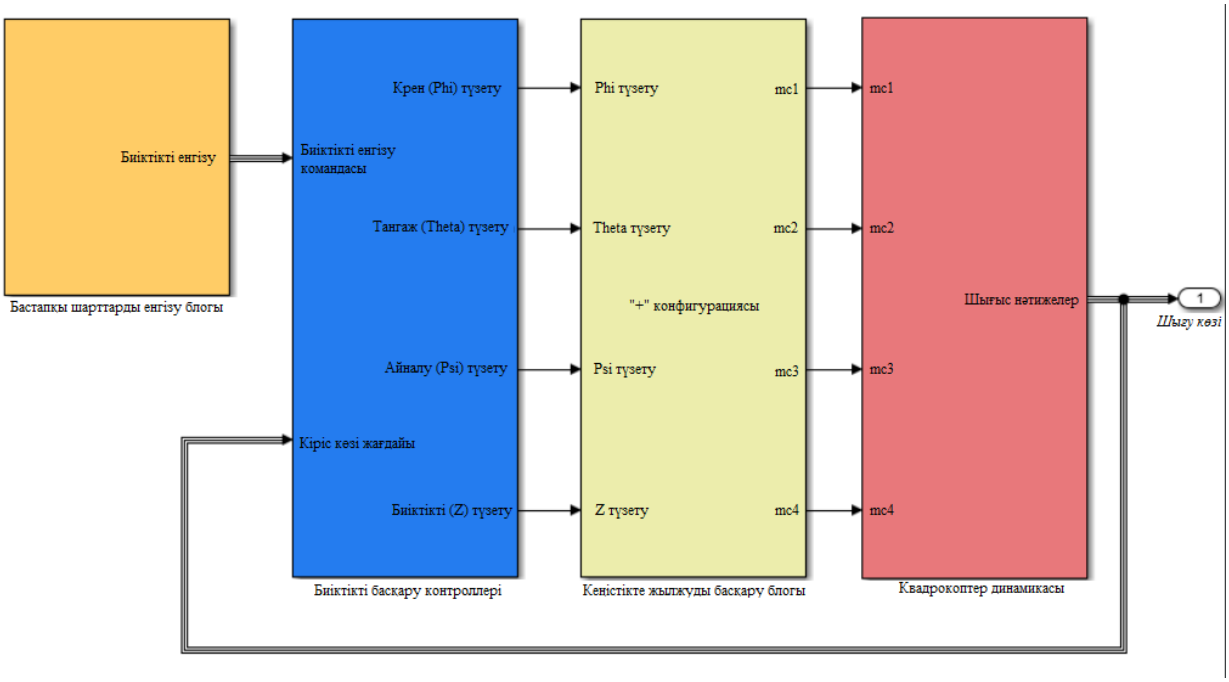
MATLAB – бұл жоғары деңгейлі тіл және бағдарламалау, сандық есептеулер және нәтижелерді визуализациялау үшін интерактивті орта. MATLAB көмегімен мәліметтерді талдауға, алгоритмдерді әзірлеуге, модельдер мен қосымшаларды құруға болады. Simulink-бұл бағытталған графтар түріндегі блок-диаграммалардың көмегімен дискретті,үздіксіз және гибридті, сызықты емес және үзілу жүйелерін қоса алғанда динамикалық модельдерді құруға мүмкіндік беретін имитациялық модельдеудің графикалық ортасы[19].

Квадрокоптерді басқару жүйесінің моделін құру барысында зерттеулерді одан әрі пайдалануды жеңілдету үшін жеке блоктарға бөлінді:

- Бастапқы шарттардың енгізу блогы
- Биіктікті басқару контроллері блогы
- Кеңістікте жылжуды басқару блогы
- Квадрокоптер динамикасы блогы.

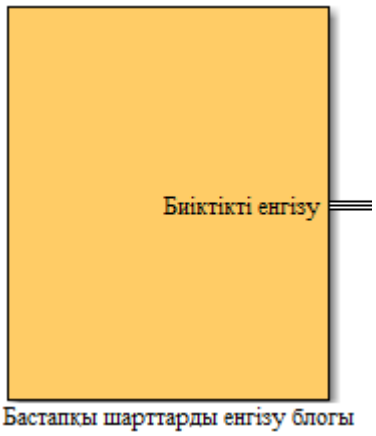
Квадрокоптерді басқару жүйесінің моделі 3.1-суретте көрсетілген.

Ұсынылған жүйе 4 модульден тұрады. Әрбір модуль-басқарудың жалпы жүйесінің бір бөлігі. Бұл модульдер белсенді элементтер болып табылады және оған екі рет басқанда, осы блоктың құрылымы ашылады. Басқару жүйесін құрудың мұндай әдісі жүйенің белгілі бір бөлігіне жылдам қол жеткізу және қажет болған жағдайда оны тез өзгерту арқылы онымен өзара іс-қимыл жасауға мүмкіндік береді. Сондай-ақ, схемада екі белсенді бағдарламаланған түймелер бар. Жүйе теріс кері байланысты қосу арқылы ұйымдастырылған ауытқу бойынша басқару принципі бар. Басқару әсері ауытқу бойынша басқару принципін пайдалану кезінде басқарылатын шаманың талап етілетін мәннен ауытқуын түрлендіру нәтижесінде шығарылады. Бұдан әрі әрбір блок бөлек бөлшектеледі.



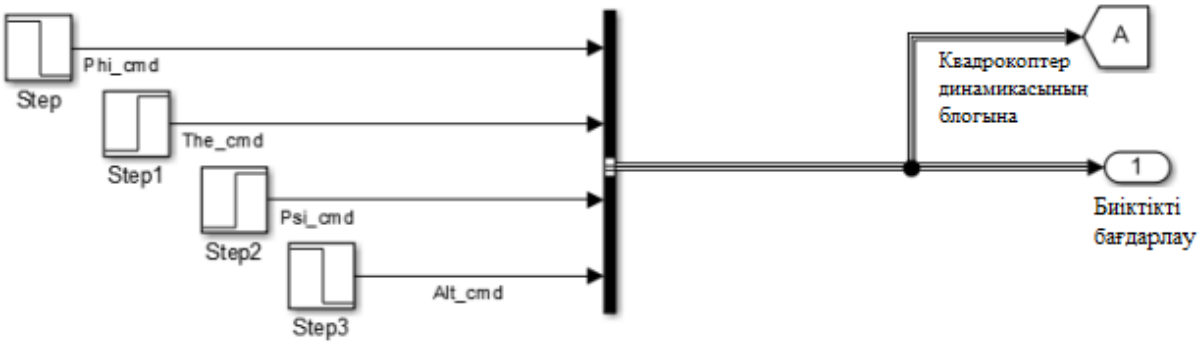
Сурет 3.1 – Квадрокоптер басқару жүйесінің моделі

Бастапқы шарттарды енгізу блогы (3.2-сурет) – ұшу аппаратының бастапқы жағдайы беріледі, сондай-ақ ұшу аппаратына тәуелді болатын биіктік қойылады.



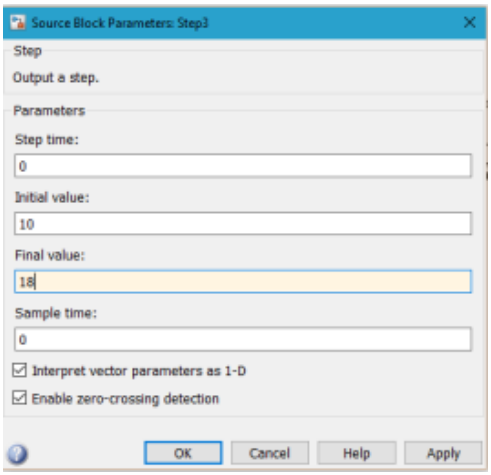
Сурет 3.2 – Бастапқы шарттарды енгізу блогы

Осы блоктың ашылуынан кейін оның құрылымдық схемасын көруге болады (3.3-сурет) . Мұнда стандартты Step блоктары қолданылған. Оператор кеңістіктегі квадрокоптердің бастапқы жағдайын қойып, бастапқы биіктікті және тұрақтандырудың биіктігін де көрсете алады[20].



Сурет 3.3 - Бастапқы шарттарды енгізу блогының құрылымдық сұлбасы

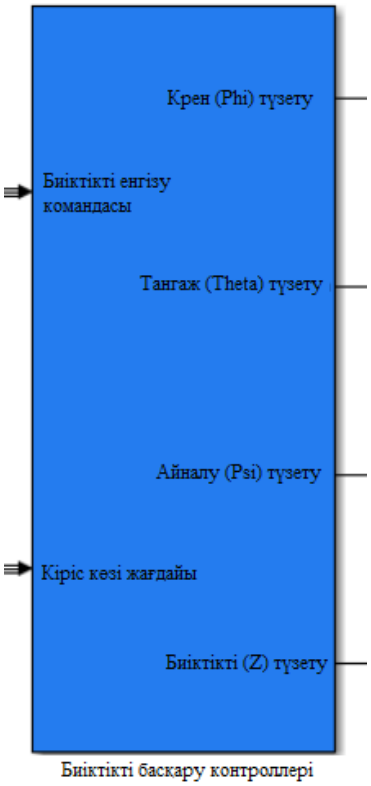
Берілген сигналдар биіктігі басқару контроллерінің блогына және квадрокөтер динамикасының блогына түседі. Тұрақтандыру үшін қажетті бастапқы биіктікті енгізу терезесін қарастырайық (3.4-сурет).



Сурет 3.4 - Биіктік параметрлерін енгізу терезесі

Осылайша кеңістіктегі ұшу аппаратын анықтайтын басқа да параметрлер үшін бастапқы шарттарды қойылады.

Биіктікті басқару контроллерінің блогы (3.5-сурет) – квадрокоптер қозғалысының түзету процесін іске асыруды қамтамасыз етеді.

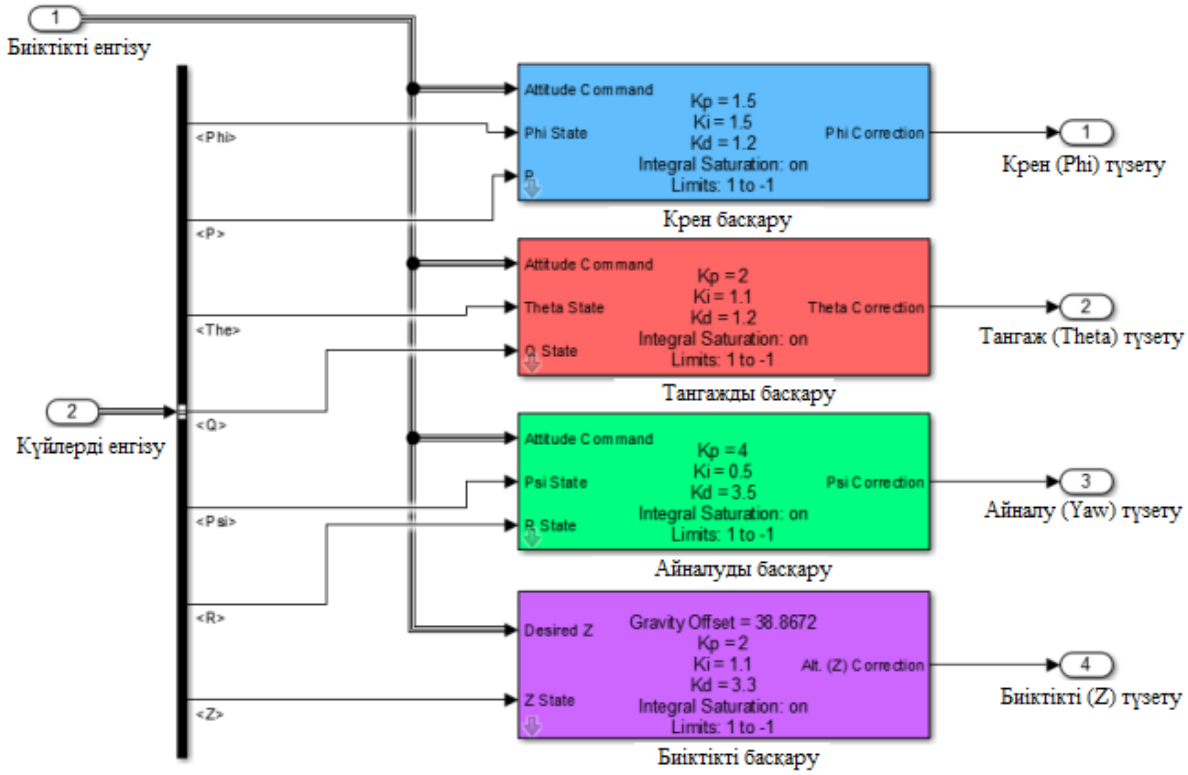


Сурет 3.5 – Биіктікті басқару контроллерінің блогы

Квадрокоптердің қозғалысын түзету ПИД-реттеуіштерді пайдалану арқылы жүргізіледі. Ушу кезінде ушу аппараты қозғалысының барлық осьтері бойынша тұрақтандыруды қамтамасыз ету қажет. Осылайша, төрт ПИД-реттегішті іске асыру қажет. Біріншісі кренді тұрақтандыратын болады, екіншісі тангажды тұрақтандыратын болады, үшіншісі рысканьені тұрақтандыратын болады, ал төртіншісі биіктігі бойынша квадрокоптерді тұрақтандыруға жауап береді. Бұл зерттеуде негізгі міндет берілген биіктікте мультироторлы пилотсыз ушу аппаратын тұрақтандыруды қарастыру болды, сондықтан реттегіштерді теңшеу кезінде негізгі тірек биіктікті ұстап тұруға жауап беретін реттегішті теңшеуге болды. 3.6-суретте басқару контроллерінің құрылымдық сұлбасы көрсетілген.

Реттеуіштің ПИД-реттеу үшін Зиглер-Никольс әдісі қолданылды. Бұл әдіс реттеуіштің ПИД-реттеуішін теңшеу кезінде ең көп таралған болып табылады. Реттеу процедурасы п-реттеуіштен және берілген реттеу объектісінен тұратын жүйені эксперименталды зерттеуден басталады. Бұл жағдайда жүйе тұрақты тербеліс амплитудасымен тербеліс орнағанға дейін, яғни жүйе орнықтылық шекарасында болмайды[21].

Реттеуішті беру коэффициентінің мәні арқылы белгіленеді және белгіленеді, бұл кезде жүйе тұрақтылық шекарасында болады. Тербелістер жүйесінде қалыптасқан кезең өлшенеді. Таңдалған типті реттеуіш параметрлерінің мәндері 3.1-кестеде келтірілген формулалар бойынша есептеледі.



Сурет 3.6 (а) – Биіктікті басқару контроллері блогының құрылымдық сұлбасы

3.1-кесте – Типтік реттеуіштердің параметрлері

	$К_п$	$К_и$	$К_д$
П-реттеуші	$0{,}50\ К_п^*$		
ПИ-реттеуші	$0{,}45\ К_п^*$	$0{,}54\ К_п^*/Т^*$	
ПИД-реттеуші	$0{,}60\ К_п^*$	$1{,}2\ К_п^*/Т^*$	$0{,}075\ К_п^*\ Т^*$

Реттеуіш параметрлерін формулалар бойынша есептеу көбінесе мінсіз нәтиже бере алмайды. Аналитикалық есептер объектінің оңайлатылған моделіне негізделгендіктен. Сондықтан талдау есептерінен кейін реттеуішті қосымша құру ұсынылады. Қосымша құрылыс ереже негізінде орындалады.

Бұл ережелер сандық параметрлерден, теориялық талдаудан және тәжірибелерден алынған. Және келесі түрі бар:

- пропорционалды коэффициентті арттыру жылдам әрекетті арттырады және тұрақтылық қорын төмендетеді;
- интегралды құрамдас бөлікті азайтумен уақыт өте келе реттеу қателігі жылдам азаяды;
- тұрақты интегралдауды азайту тұрақтылық қорын азайтады;
- дифференциалдық құрауыштың ұлғаюы орнықтылық пен жылдамдық қорын арттырады.

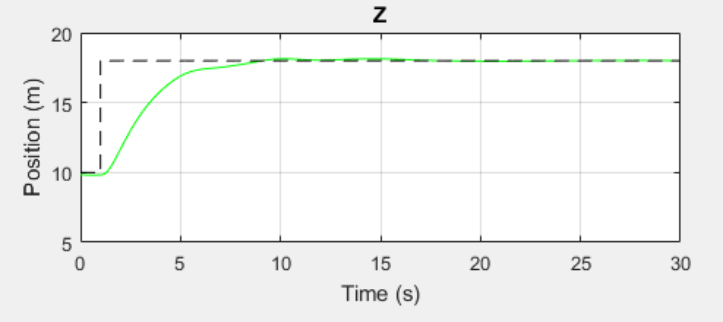
Осылайша, реттеуіш коэффициенттері анықталды және қосымша құрылыс орындалды:

Кп=1,9;

Кі=0.95;

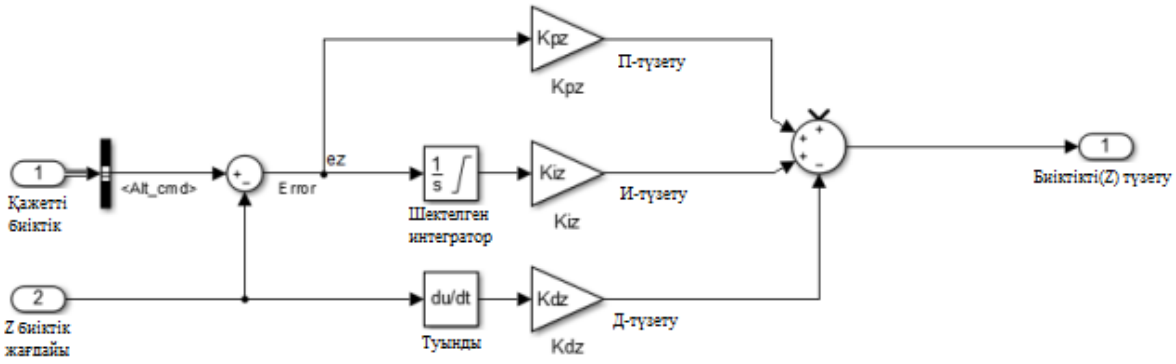
Kd=5.2.

Өтпелі процесс графигі 3.6(ә)-суретте көрсетілген, кестеден өтпелі процесс тұрақты болып табылады, қалыптасқан мән рұқсат етілген мәндер дәлізіне 5% кіреді. Мұндай өтпелі процесс бастапқы тапсырманы толық қанағаттандырады деп айтуға болады.



Сурет 3.6 (ә) – Биіктікті реттеудің өтпелі процесі

3.6(а)-суретте келтірілген құрылымдық схемада төрт ПИД-реттеуіш көрінеді, кез келген ПИД-реттеуішті блоктың сол жақ төменгі бұрышында бағыттауышты басу арқылы неғұрлым егжей-тегжейлі қарауға болады. Осылайша, реттеуіштің құрылымдық сұлбасы ашылады (3.7-сурет).



Сурет 3.7 - Биіктік ПИД-реттегішінің құрылымдық сұлбасы

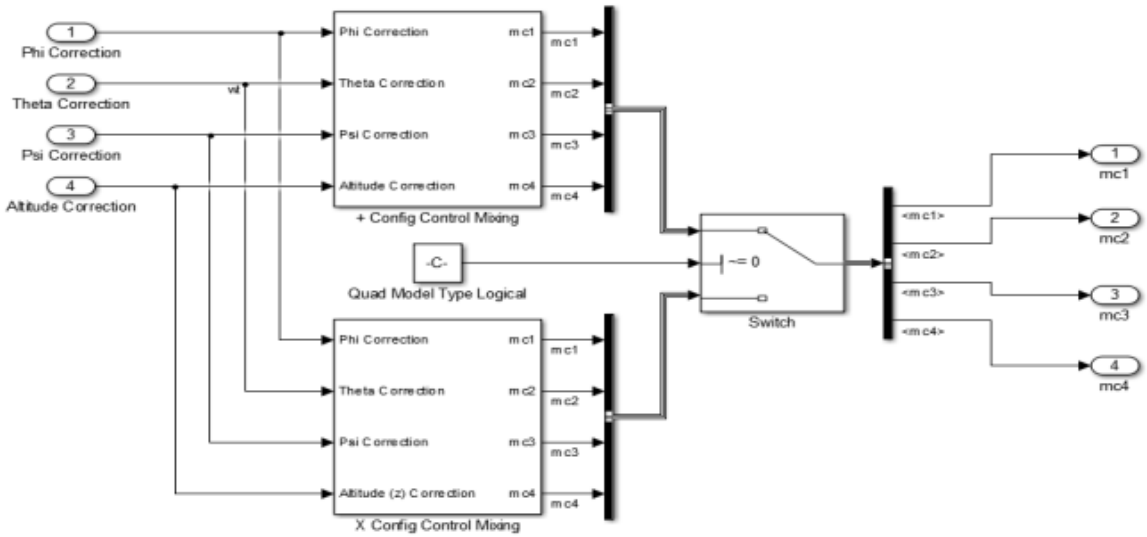
Басқару контроллерінің реттеу параметрлерін реттегеннен кейін биіктіктегі басқару контроллерінің блогы алынған нәтижелерді кеңістіктегі ығысуды басқару блогына береді.

Кеңістікте жылжуды басқару блогы – реттеуіштерден түзетілген параметрлерді қабылдайды және оларды қажетті роторға жіберу үшін біріктіреді. Квадрокоптердің екі негізгі конфигурациясы бар болғандықтан, оларды "+" конфигурация және "X" конфигурациясын атаймыз. "+"

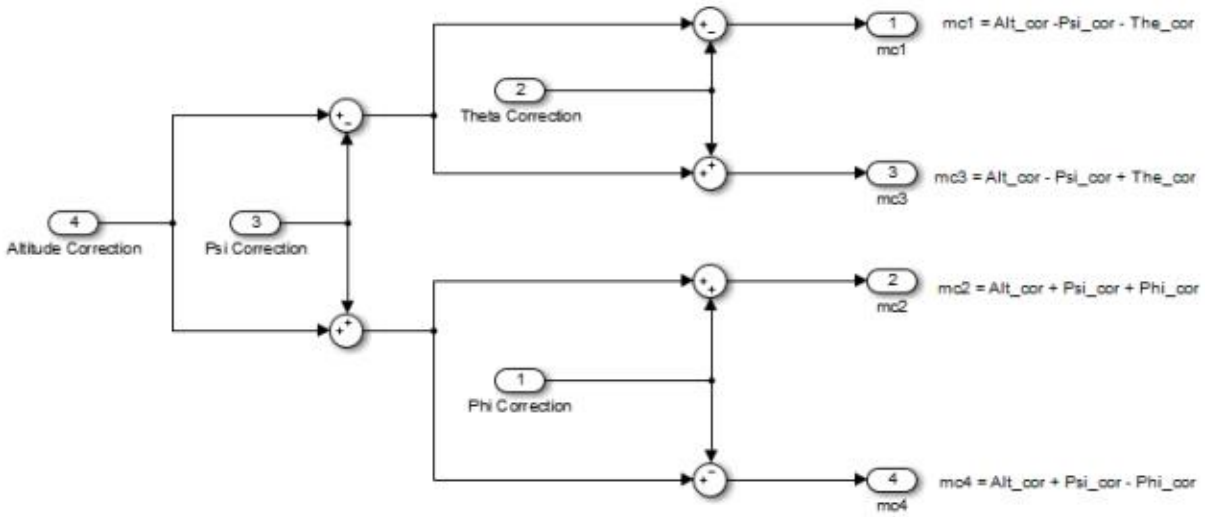
конфигурациясында X және Y осі ұшу аппаратының иығының бойында жатыр. X осі иық бойымен жатыр, онда ротор сағат тіліне бұрылады, ал Y осі сағат тіліне қарсы айнала отырып ротор орнатылған иық бойымен жатыр. "X" конфигурациясында X және Y осьтері ұшу аппаратының иықтары арасында орналасқан, яғни "+"конфигурациясына қатысты 45 градусқа бұрылады. Басқару блогы 3.8-суретте көрсетілген.

Сурет 3.8 – Кеңістікте жылжуды басқару блогы

Осы блокты ашқан кезде оның құрылымдық схемасын көруге болады (3.9-сурет, 3.10-сурет). Қосқышты пайдалану себебі басқару жүйесінің құрылымдық схемасына қосымша өзгерістер енгізбестен, квадрокоптер конфигурациясының екі нұсқаларын пайдалануға мүмкіндік береді.



Сурет 3.9 - Кеңістікте жылжуды басқару блогының құрылымдық сұлбасы

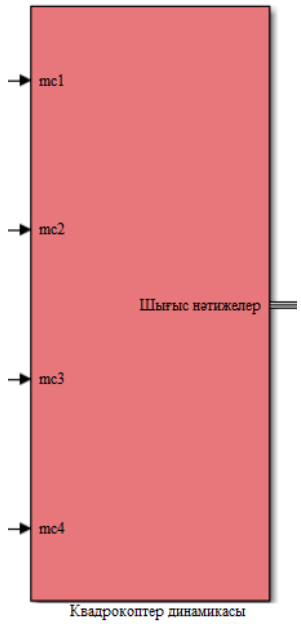


Сурет 3.10 - “+” конфигурациясына арналған түзетулердің құрылымдық сұлбасы

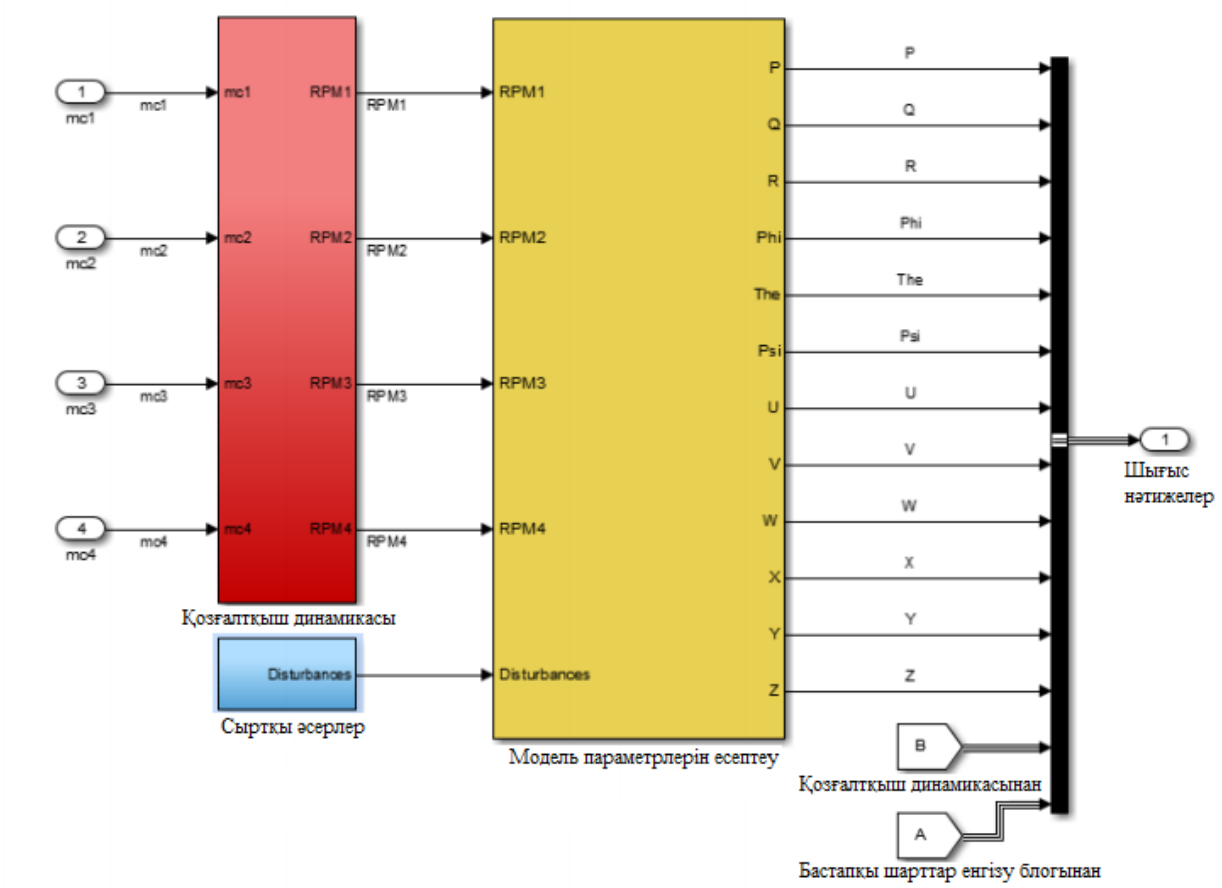
Роторларды басқаруға түзетулер енгізгеннен кейін осы блоктан деректер квадрокоптердің динамика блогына түседі.

Квадрокоптер динамикасының блогы (3.11-сурет) – атқарушы құрылғыларға, атап айтқанда ұшу аппаратының роторларын тікелей басқаруға жауап береді.

Бұл блоктың ішінде блоктар орналасқан: мотор динамикасы, сыртқы әсерлер, модель параметрлерін есептеу. 3.12-суретте квадрокоптер динамикасының блогының құрылымдық сұлбасы көрсетілген.



Сурет 3.11 – Квадрокоптер динамикасының блогы

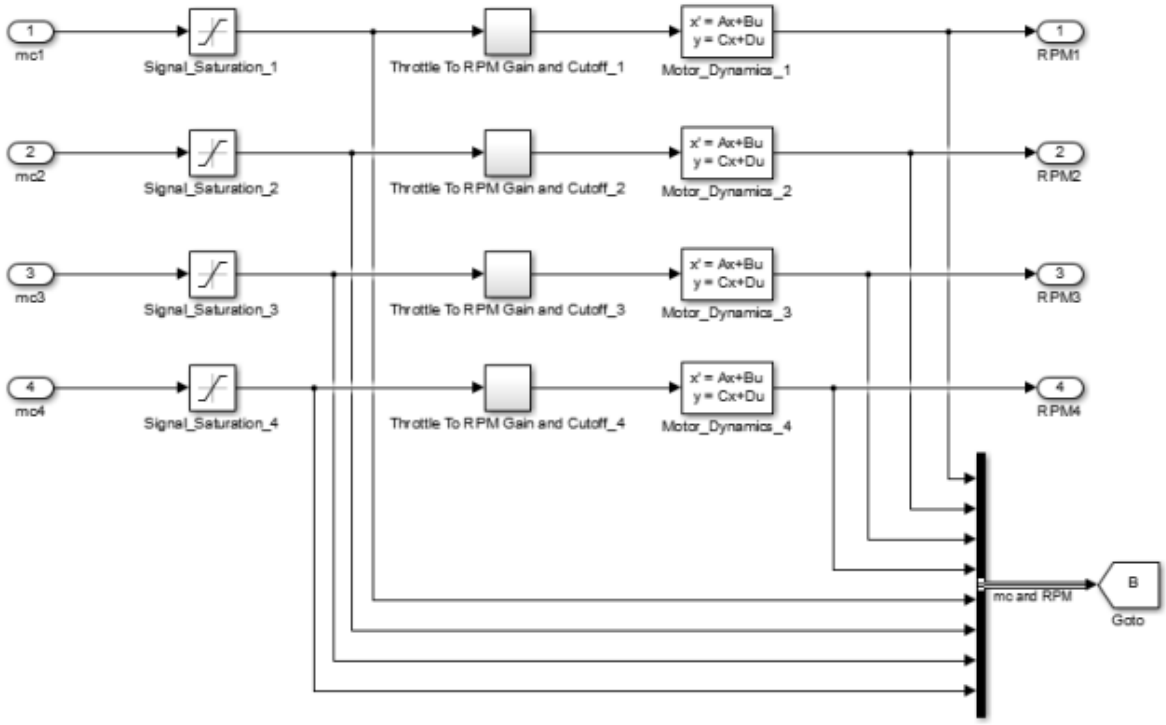


Сурет 3.12 – Квадрокоптер динамикасы блогының құрылымдық сұлбасы

Квадрокоптер динамикасының блогының құрамына кіретін блоктарды қарастырайық. Мотор динамикасының блогы тұрақты токтың коллекторсыз моторын іске асырудың құрылымдық схемасы болып табылады. Осы құрылымдық сұлба (3.13-сурет) Matlab Simulink әзірлеушілері ұсынған шешімдердің мысалдарынан алынған[22].

Signal_Saturation блоктары қозғалтқыштардың қуатын шектеу үшін қолданылады. Қозғалтқыштардың максималды қуаты 100% шектеледі.

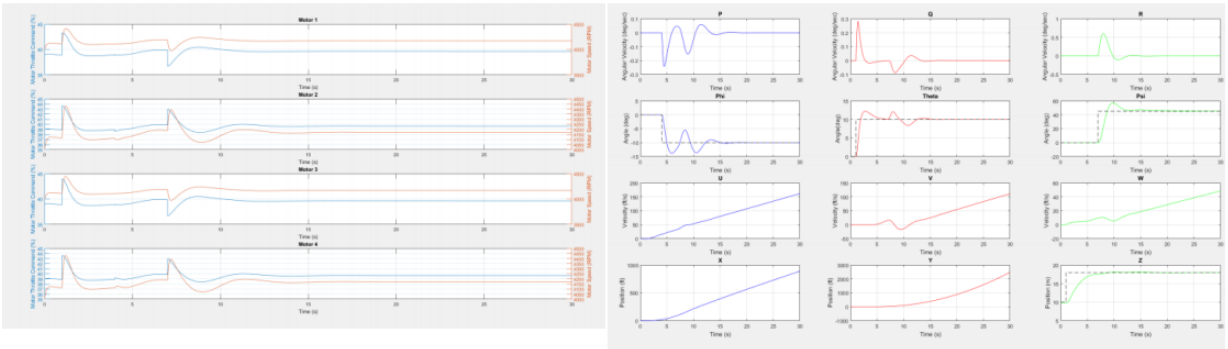
Сыртқы әсер ету блогы квадракоптердің осьтерінің бірінде желдің қосымша жылдамдығы сияқты сыртқы қоздырғыш сигналдарды қосу мүмкіндігін қамтиды.



Сурет 3.13 – Қозғалтқыш динамикасы блогының құрылымдық схемасы

Пайдалану ыңғайлы болу үшін жүйеде болып жатқан барлық процестердің графиктерін құру қосымша функциялар енгізілді (3.14-сурет).

Модель параметрлерін есептеу блогы теориялық формулалар бойынша есеп жүргізеді. Бұл блок бағдарламаланатын болып табылады, яғни ол MatLab бағдарламалау тілінде алдын ала жазылған бағдарламаны пайдаланады.



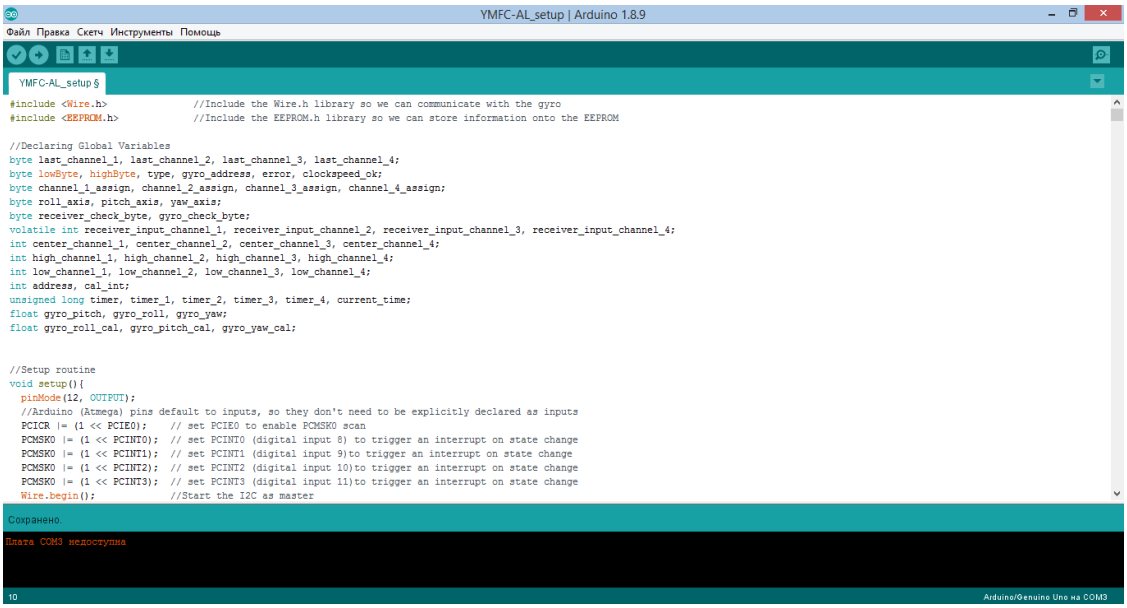
Сурет 3.14 – Квадрокоптер басқару жүйесінің барлық жағдайының графиктері

3.14-суретте көрсетілген графиктерде квадрокоптер ұшу барысында қозғалтқыштардың өзгеру динамикасын айқын бақылауға болады.

3.2 Arduino IDE қосымшасында бағдарламалық кодты талдау

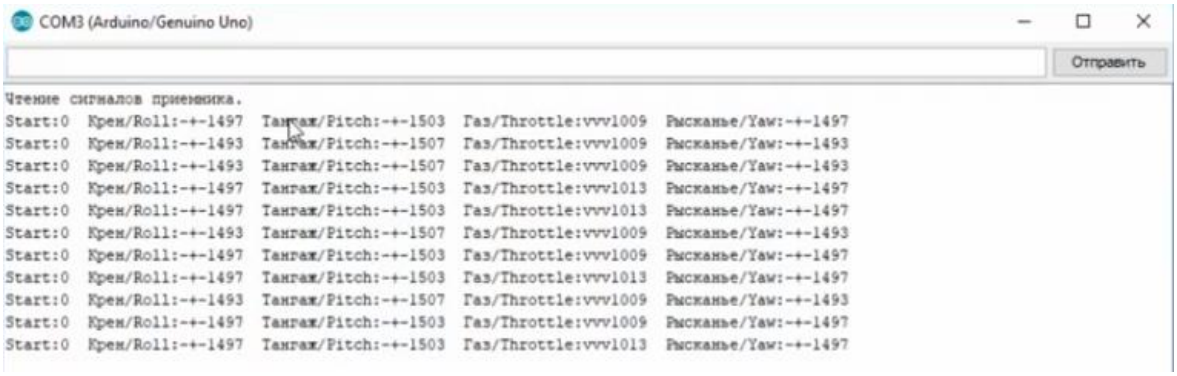
Arduino IDE – бұл ыңғайлы мәтіндік редакторда бағдарламалар құруға, оларды машиндық кодқа компирлей алатын және барлық нұсқаларды Arduino-ға жүктеуге мүмкіндік беретін қосымша.

3.15-суретте Arduino IDE бағдарламасының ашылу терезі көрсетілген.



Сурет-3.15 – Arduino IDE бағдарламасының бастапқы терезесі

Дербес компьютер мен Arduino микроконтроллері арасында ақпарат алмасу «Монитор порта» терезесі арқылы жүзеге асады (3.16-сурет).



Сурет 3.16 – ДК мен Arduino арасында ақпарат алмасу терезесі

Arduino микроконтролері квадрокоптердің барлық компоненттерін басқару үшін оған сапалы код жүктелу қажет. Жобадағы Arduino-ға жүктелетін басты бастапқы бағдарламалық код тізбегі 1-қосымшада қарастырылған.

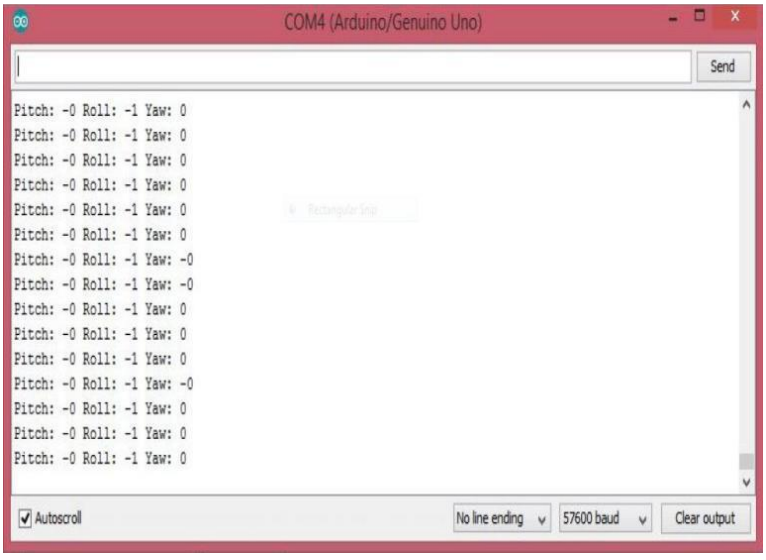
Arduino Uno микроконтроллері C тілін Arduino IDE бағдарламалық жасақтамасын қолдану арқылы бағдарламалайды. Бұл кодтау тіліне қосу және өңдеу үшін пайдаланушы интерфейсін пайдаланатын даму ортасы.

ЭЖК-ны бағдарламалық калибрлеу квадрокоптердің тұрақты ұшуына көмектеседі және міндетті шарт болып табылады. ЭЖК калибрлеуі әрбір ротордың және сәйкес ЭЖК модулінің көмегімен басымдық негізінде жасалады. Ол мынадай қадамдарды қамтиды:

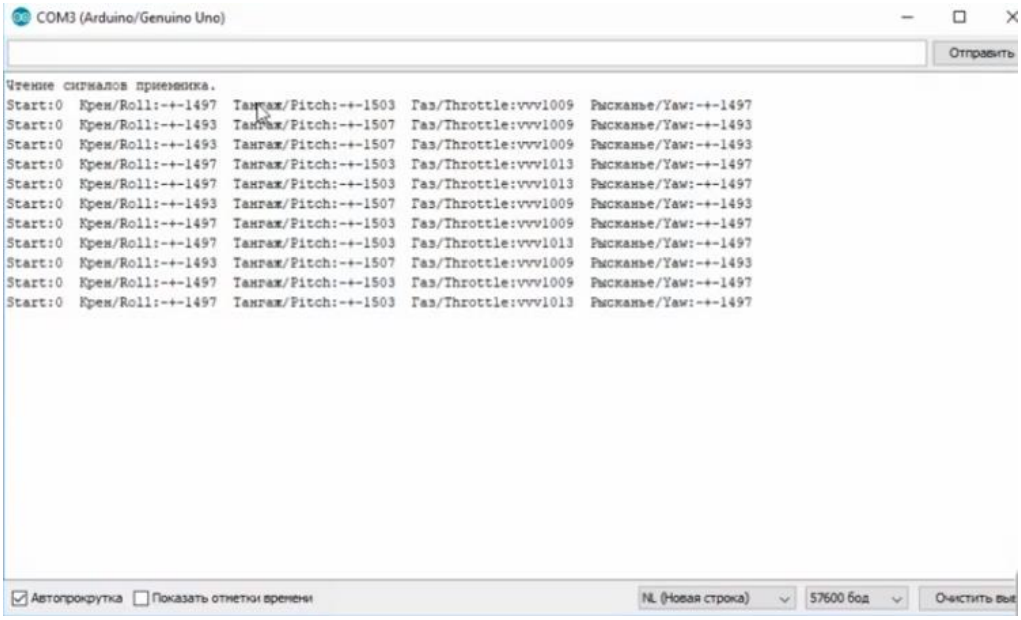
- Алдымен бағдарламаны контроллерлік тақтаға жүктеңіз, содан кейін таратқышты қосыңыз және дроссель таяқшасын максимумына қойыңыз.
- Енді батареяны қосыңыз. Автоматты ұшқыштың қызыл, көк және сары жарық диодты циклдік үлгілері жарықтанады, бұл ЭЖК-ның калибрлеу режиміне дайын екенін білдіреді.
- Таратқыштың дросселін жоғары ұстау арқылы батареяны ажыратып, қайта қосыңыз.
- Таратқышта жүйелі дыбыстық сигнал жоқ, сондықтан дыбыстар саны батарея ұяшығының санын білдіреді және қосымша екі дыбыс сигналы максималды дроссельдің басып алынғанын көрсетеді.
- Енді таратқыштың дросселын төменгі позициясына орнатыңыз.
- ЭЖК енді ең аз дроссельдің түсірілгенін және калибрлеу аяқталғанын білдіретін ұзын тонды дыбыс шығаруы тиіс.

3.17-суретте IDE бағдарламалық жасақтамасындағы орамның, қадамның және жіптің мәндері көрсетілген. Теңгерімді ұшу үшін гироскоптың бұл параметрлері тиісінше -0, -0 және 0 болғаны жөн.

Arduino микроконтроллеріне жүктелетін ұшу контроллерінің бағдарламалық коды 2-қосымшада көрсетілген.



Сурет 3.17 – ESC калибрлеу кезінде Arduino IDE сериялық мониторының көрсеткіштері



Сурет 3.18 – Барлық стиктер үшін (орталы күйде) IDE сериялық монитор көрсеткіштері

Ұшуды калибрлеу режимі - өрісті сынау режимі. Төртквадратты ұшу режиміне қойыңыз да, квадрокоптерді өріске тексеріңіз. Сынақ стендінде гироскопиялық параметрлерді баптағаннан кейін, квадрокоптерді тегін ұшу кезінде сынай бастадық. Бұл сынақтар ашық алаңда ашық ауада жүргізілді, алайда квадрокоптерден қауіпсіз қашықтықты сақтау қажет.

Бірқи-осьтік сынақтардан кейін, квадрокоптердің крен және тангаж осьтерде өте тұрақты болатынына сенімді болдық, бірақ ол ығысу бағытында едәуір шетке ауысты. Тек қана пропорционалдық бақылаумен басқару пульті бойынша бақылауды тұрақтандырғаннан кейін, төртбұрыштар азаяды және әлдеқайда басқарылатын болды. Төртіншіден, төртбұрыштың бақылауды басқаратын деректерге дұрыс жауап бермейінше, PID-де пропорционалды бақылау мерзімін ұлғайттық. Төртбұрыштың орамдағы және қысқыштық осьтеріндегі шағын, жылдам тербелістерді тапқанын анықтадық. Мұны түзету үшін дірілдерді теңестіру арқылы төртбұрышты вибрацияны азайтып, діріл сіңіргіш тіреуішті қолданып, қалған дірілден сенсор тақтасын оқшаулау қажет. Содан кейін квадрокоптердің ұшу режимі тұрақты болды (3.19-сурет).

Қозғалтқышты тексеру – оның ЭЖК командалық сигналына қатынасын анықтау үшін қозғалтқыштардың бірі жүктеме ұяшығына орнатылуы. Жүктемелік камерада қозғалтқыштан күтілетін максимум кемінде 1 кг үшін кемінде 5 кг жүктеме болды. Әрбір қозғалтқышы мен 10X4.5 пропеллері үшін 2000 мкс импульсімен бірге жеткізілгенде 0,602 кг-ға жетуге мүмкіндік алдық. Осылайша, төрт коллекторсыз тұрақты ток қозғалтқышын пайдалану арқылы аккумулятор толығымен зарядталған жағдайда, квадрокоптер үшін 4X0.602 = 2.408 кг максималды салмақ көтеру мүмкіндігін жасауға болады[23].

Сурет 3.19 – Arduino микроконтроллері негізінде құрылған квадрокоптердің соңғы макеті

Болашақтағы мүмкін жаңартулар - квадрокоптер жиынтығын жобалау, тестілеу және құрудан бөлек, қосымша өзгерістермен толықтыру. Болашақта оның қолданылуына негізделген әртүрлі ықтимал өзгерістер бар, ол мыналарды қамтиды:

- Нақты биіктікті анықтау үшін контроллер тақтасына sonic sensor модулін қосу
- GPS-модулін қадағалауға және шпиондық бағдарламаларға арналған жиынтыққа енгізу
- Бұл дизайн жоғары рейтингі бар Motor Driver немесе релелік драйверді коммерциялық қосымшалар үшін пайдалану
- Камераны пайдалану арқылы жылжымайтын мүлікті фотосуретке түсіру үшін қолдануға болады. Басқа қолданбалы камерамен жабдықталған квадрокоптер көмегімен кең ауқымды тексеру және бақылау
- Пестицидтерді шашырату
- Кішігірім салмақты жүктерді тасымалдау.

Салмақ көтеру есептеулеріне сүйене отырып, салмақ көтеру критерийлерін қанағаттандыратын әртүрлі модульдерді көтеру үшін бірыңғай үнемді жобадағы квадрокоптерді пайдалануға болады.

ҚОРЫТЫНДЫ

Arduino микроконтроллерінің қолдану мүмкіндіктерін зерттеу мақсатында Arduino Uno микроконтроллері негізіндегі квадрокоптерді әзірлеу ойластырылды. Квадрокоптерді құрмастан бұрын қазіргі заманғы мультироторлы жүйелерге аналитикалық шолу жасалынып, олардың заманауи шешімдері қарастырылды.

Квадрокоптерді басқару платасының негізгі блоктарын қарастыру үшін компоненттердің техникалық сипаттамалары мен квадрокоптерді әзірлеу приниптері зерттелінді. Ұшу уақыты бойынша батареяның қажетті сыйымдылығын есептеу нәтижесінде 2200мА*сағ сыйымдылықты 3S батарея қуаты 12,2 минут уақытқа жететінін көрсетті.

MATLAB Simulink пакетінде квадрокоптер басқару жүйесінің моделі құрылды. Нәтижесінде квадрокоптердің басқару жүйесінің әрбір блогына талдау жасалу арқылы көрсеткіштер алынды. Зерттеу нәтижелері MATLAB ортасында құрылған бағдарламалық кешеннің көмегімен жобаланған, қолданылған әдістің жұмыс қабілеттілігі мен тиімділігі көрсетілді.

Квадрокоптерді бастан құрудың практикалық маңызы зор, өйткені квадрокоптерді қашықтықтан радиобасқару, электр тізбегінің теориясын зерттеу, Arduino микроконтроллерінің пайдалану мүмкіндіктерін анықтау және бағдарламалау принциптерін түсінуге мүмкіндік берді.

Бұл дипломдық жұмыс жүйелік үлгілеуге, квадрокоптердің басқару жүйесін талдауға, жобаларды басқаруға, микроконтроллерді бағдарламалау мен ауқымды жобаны жүргізуге келгенде құнды тәжірибе берді.

ПАЙДАЛАНЫЛҒАН ӘДЕБИЕТТЕР ТІЗІМІ

1. Teppo Luukkonen, “Modelling and control of quadcopter”. School of Science, Espoo, August 22, 2011.

2. Stafford, Jesse, "How a Quadcopter works Clay Allen". University of Alaska, Fairbanks. Retrieved 2015-01-20.

3. Sandeeep Khajure, Vaibhav Surwade, Vivek Badak, “Quadcopter Design and Fabrication,” International Advance Research Journal in Science, Engineering and Technology (IARJSET), Vol. 3, Issue 2, February 2016.

4. David Roberts, "Construction and Testing of a Quadcopter," California Polytechnic State University, San Luis Obispo, CA, 93407, June, 2013.

5. J. Engel, J. Sturm, D. Cremers, "Camera-based navigation of a low cost quadrocopter", IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2815-2821, Oct. 2012, ISBN 2153-0858.

6. J. Valvano, "Embedded Microcomputer Systems: Real Time Interfacing", Brooks-Cole, 2000, ISBN 0534366422.

7. S. Bouabdallah, P. Murrieri, R. Siegwart, "Design and Control of an Indoor Micro Quadrotor", Robotics and Automation 2004. Proceedings. ICRA ‘04. 2004 IEEE International Conference on, vol. 5, pp. 4393-4398, 2004.

8. Гребеников, А. Г. Общие виды и характеристики беспилотных летательных аппаратов : справ. пособие / А. Г. Гребеников, А. К. Мялица, В. В. Парфенюк. - Харьков : Изд-во Харьковского ин-та, 2008. -377 с.

9. Корнилов, В. А. Система управления мультикоптером / В. А. Корнилов, Д. С. Молодяков, Ю. А. Синявская. - М., 2012. - 512 с.

10. Антохин, А. И. Концепция системы стабилизации на базе МЭМС гироскопа / А. И. Антохин и др. // Наука и образование : электрон. науч.-техн. изд., 2011. -311 с.

11. Фетисов, В. С. Беспилотная авиация: терминология, классификация, современное состояние / В. С. Фетисов, Л. М. Неугодникова, В. В. Адамовский и Р. А. Красноперов - Уфа, 2014. - 346 с.

12. Quadcopter [Электронная энциклопедия] – Режим доступа: https://en.wikipedia.org/wiki/Quadcopter, свободный.

13. https://joystick-pro.ru/multirotor_basic_concepts.php

14. https://www.hackster.io/robocircuits/arduino-quadcopter-e618c6

15. Петин В. А. Проекты с использованием контроллера Arduino – 2-е изд., перераб. и доп. – Санкт-Петербург: БХВ-Петербург, 2015. – 462 с.

16. http://earchive.tpu.ru/handle/11683/29429

17. Зенкевич С. Л., Галустян Н. К. Разработка математической модели и синтез алгоритма угловой стабилизации движения квадрокоптера. // Мехатроника, Автоматизация, Управление», No 3, 2014.- С.27-32.

18. Arduino: A Quick-Start Guide/ Maik Schmidt – Dallas, Texas – 2015. P.311

19. Соммер У. Программирование микроконтроллерных плат Arduino/Freeduino. – СПб.: БХВ-Петербург, 2012. – 256 с.

20. MPU-6000 and MPU-6050 Register Map and Descriptions Revision 4.2 [Электронный ресурс]. – URL: https://www.invensense.com/wp-content/uploads/2015/02/MPU-6000-RegisterMap1.pdf (дата обращения 18.11.2018).

21. Юшкин Д. А., Евдокимов С. А. Разработка адаптивного нечеткого ПИД-регулятора системы автоматического управления и стабилизации мультироторного БПЛА типа квадрокоптер // Актуальные проблемы современной техники, науки и образования, 2015. Т. 2, № 1. С. 194-198.

22. https://matlab.ru/products/simulink

23. http://earchive.tpu.ru/bitstream/11683/52294/1/conference_tpu-2018-C24_V1_p258-263.pdf

24. Arduino микроконтроллерінің бағдарламалау кодтары (скетч) [Электронды ресурс]: https://cediy.ru/blog/download?id=12

Қосымша 1

Arduino микроконтроллеріне жүктелетін бастапқы бағдарламалық код тізбегі (скрипт)

```
#include <Wire.h>               //Include the Wire.h library so we can communicate with the
#include <EEPROM.h>             //Include the EEPROM.h library so we can store
byte last_channel_1, last_channel_2, last_channel_3, last_channel_4;
byte lowByte, highByte, type, gyro_address, error, clockspeed_ok;
byte channel_1_assign, channel_2_assign, channel_3_assign, channel_4_assign;
byte roll_axis, pitch_axis, yaw_axis;
byte receiver_check_byte, gyro_check_byte;
int center_channel_1, center_channel_2, center_channel_3, center_channel_4;
int high_channel_1, high_channel_2, high_channel_3, high_channel_4;
int low_channel_1, low_channel_2, low_channel_3, low_channel_4;
int address, cal_int;
unsigned long timer, timer_1, timer_2, timer_3, timer_4, current_time;
float gyro_pitch, gyro_roll, gyro_yaw;
float gyro_roll_cal, gyro_pitch_cal, gyro_yaw_cal;
void setup(){
Serial.println(F(""));
Serial.println(F("========================================="));
  Serial.println(F("System check"));
  Serial.println(F("========================================="));
  delay(1000);
  Serial.println(F("Checking I2C clock speed."));
  delay(1000);
 Serial.println(F("Lift the left side of the quadcopter to a 45 degree angle within 10 seconds"));
    check_gyro_axes(1);
    if(error == 0){
      Serial.println(F("OK!"));
      Serial.print(F("Angle detection = "));
      Serial.println(pitch_axis & 0b00000011);
      if(pitch_axis & 0b10000000)Serial.println(F("Axis inverted = yes"));
      else Serial.println(F("Axis inverted = no"));
      Serial.println(F("Put the quadcopter back in its original position"));
      Serial.println(F("Move stick 'nose up' and back to center to continue"));
      check_to_continue();
      Serial.println(F(""));
      Serial.println(F(""));
      Serial.println(F("Rotate the nose of the quadcopter 45 degree to the right within 10 seconds"));
      check_gyro_axes(3);    }
    if(error == 0){
      Serial.println(F("OK!"));
      Serial.print(F("Angle detection = "));
      Serial.println(yaw_axis & 0b00000011);
      if(yaw_axis & 0b10000000)Serial.println(F("Axis inverted = yes"));
```

Қосымша 1 (жалғасы)

```
  Serial.println(F("Put the quadcopter back in its original position"));
     Serial.println(F("Move stick 'nose up' and back to center to continue"));
     check_to_continue();    }  }
  if(error == 0){
    Serial.println(F(""));
    Serial.println(F("================================================="));
    Serial.println(F("LED test"));
    Serial.println(F("================================================="));
    digitalWrite(12, HIGH);
    Serial.println(F("The LED should now be lit"));
    Serial.println(F("Move stick 'nose up' and back to center to continue"));
    check_to_continue();
    digitalWrite(12, LOW);  }
  Serial.println(F(""));
  if(error == 0){
    Serial.println(F("================================================="));
    Serial.println(F("Final setup check"));
    Serial.println(F("================================================="));
    delay(1000);
    if(receiver_check_byte == 0b00001111){
      Serial.println(F("Receiver channels ok"));    }
    else{
      Serial.println(F("Receiver channel verification failed!!! (ERROR 6)"));
      error = 1;    }
    delay(1000);
```

Қосымша 1 (жалғасы)

```
  Serial.println(F("Gyro
axes ok"));    }
    else{
      Serial.println(F("Gyro exes verification failed!!! (ERROR 7)"));
      error = 1;    }  }
  if(error == 0){
    Serial.println(F(""));
    Serial.println(F("================================================="));
    Serial.println(F("Storing EEPROM information"));
    Serial.println(F("================================================="));
    Serial.println(F("Writing EEPROM"));
    delay(1000);
    Serial.println(F("Done!"));
    EEPROM.write(0, center_channel_1 & 0b11111111);
    EEPROM.write(1, center_channel_1 >> 8);
    EEPROM.write(2, center_channel_2 & 0b11111111);
    EEPROM.write(3, center_channel_2 >> 8);
    EEPROM.write(4, center_channel_3 & 0b11111111);
    EEPROM.write(5, center_channel_3 >> 8);
    EEPROM.write(6, center_channel_4 & 0b11111111);
```

```
EEPROM.write(8, high_channel_1 & 0b11111111);
   EEPROM.write(9, high_channel_1 >> 8);
   EEPROM.write(10, high_channel_2 & 0b11111111);
   EEPROM.write(11, high_channel_2 >> 8);
   EEPROM.write(12, high_channel_3 & 0b11111111);
   EEPROM.write(13, high_channel_3 >> 8);
   EEPROM.write(14, high_channel_4 & 0b11111111);
   EEPROM.write(15, high_channel_4 >> 8);
   EEPROM.write(16, low_channel_1 & 0b11111111);
   EEPROM.write(17, low_channel_1 >> 8);
   EEPROM.write(18, low_channel_2 & 0b11111111);
   EEPROM.write(19, low_channel_2 >> 8);
   EEPROM.write(20, low_channel_3 & 0b11111111);
   EEPROM.write(21, low_channel_3 >> 8);
   EEPROM.write(22, low_channel_4 & 0b11111111);
   EEPROM.write(23, low_channel_4 >> 8);
   EEPROM.write(24, channel_1_assign);
   EEPROM.write(25, channel_2_assign);
   EEPROM.write(26, channel_3_assign);
   EEPROM.write(27, channel_4_assign);
   EEPROM.write(28, roll_axis);
   EEPROM.write(29, pitch_axis);
   EEPROM.write(30, yaw_axis);
   EEPROM.write(31, type);
   EEPROM.write(32, gyro_address);
   EEPROM.write(33, 'J');
   EEPROM.write(34, 'M');
   EEPROM.write(35, 'B');
   Serial.println(F("Verify EEPROM data"));
   delay(1000);
   gyro_roll=Wire.read()<<8|Wire.read();                    //Read high and low part of the angular
data
   if(cal_int == 2000)gyro_roll -= gyro_roll_cal;           //Only compensate after the calibration
   gyro_pitch=Wire.read()<<8|Wire.read();                   //Read high and low part of the angular
data
   if(cal_int == 2000)gyro_pitch -= gyro_pitch_cal;         //Only compensate after the calibration
   gyro_yaw=Wire.read()<<8|Wire.read();                     //Read high and low part of the angular
data
   if(cal_int == 2000)gyro_yaw -= gyro_yaw_cal;    }}
void check_receiver_inputs(byte movement){
  byte trigger = 0;
  int pulse_length;
  while(timer > millis() && trigger == 0){
    delay(250);
    if(receiver_input_channel_1 > 1750 || receiver_input_channel_1 < 1250){
      trigger = 1;
      receiver_check_byte |= 0b00000001;
      pulse_length = receiver_input_channel_1;    }
    if(receiver_input_channel_2 > 1750 || receiver_input_channel_2 < 1250){
```

```
    pulse_length = receiver_input_channel_2;    }
    if(receiver_input_channel_3 > 1750 || receiver_input_channel_3 < 1250){
      trigger = 3;
      receiver_check_byte |= 0b00000100;
      pulse_length = receiver_input_channel_3;    }
    if(receiver_input_channel_4 > 1750 || receiver_input_channel_4 < 1250){
      trigger = 4;
      receiver_check_byte |= 0b00001000;
      pulse_length = receiver_input_channel_4;    }   }
  if(trigger == 0){
    error = 1;
    Serial.println(F("No stick movement detected in the last 30 seconds!!! (ERROR 2)"));  }
  else{
    if(movement == 1){
      channel_3_assign = trigger;
      if(pulse_length < 1250)channel_3_assign += 0b10000000;    }
    if(movement == 2){
      channel_1_assign = trigger;
      if(pulse_length < 1250)channel_1_assign += 0b10000000;    }
    if(movement == 3){
      channel_2_assign = trigger;
      if(pulse_length < 1250)channel_2_assign += 0b10000000;    }
    if(movement == 4){
      channel_4_assign = trigger;
      if(pulse_length < 1250)channel_4_assign += 0b10000000;    }  }}
  while(timer > millis() && gyro_angle_roll > -30 && gyro_angle_roll < 30 && gyro_angle_pitch
> -30 && gyro_angle_pitch < 30 && gyro_angle_yaw > -30 && gyro_angle_yaw < 30){
    gyro_signalen();
    if(type == 2 || type == 3){
   gyro_angle_roll += gyro_roll * 0.00007;             //0.00007 = 17.5 (md/s) / 250(Hz)
      gyro_angle_pitch += gyro_pitch * 0.00007;
      gyro_angle_yaw += gyro_yaw * 0.00007;    }
    if(type == 1){
      gyro_angle_roll += gyro_roll * 0.0000611;             // 0.0000611 = 1 / 65.5 (LSB degr/s) /
250(Hz)
      gyro_angle_pitch += gyro_pitch * 0.0000611;
      gyro_angle_yaw += gyro_yaw * 0.0000611;    }
  if((gyro_angle_roll < -30 || gyro_angle_roll > 30) && gyro_angle_pitch > -30 &&
gyro_angle_pitch < 30 && gyro_angle_yaw > -30 && gyro_angle_yaw < 30){
    gyro_check_byte |= 0b00000001;
    if(gyro_angle_roll < 0)trigger_axis = 0b10000001;
    else trigger_axis = 0b00000001;  }
  if((gyro_angle_pitch < -30 || gyro_angle_pitch > 30) && gyro_angle_roll > -30 &&
gyro_angle_roll < 30 && gyro_angle_yaw > -30 && gyro_angle_yaw < 30){
    gyro_check_byte |= 0b00000010;
    if(gyro_angle_pitch < 0)trigger_axis = 0b10000010;
    else trigger_axis = 0b00000010;  }
```

```
 if((gyro_angle_yaw < -30 || gyro_angle_yaw > 30) && gyro_angle_roll > -30 && gyro_angle_roll < 30 && gyro_angle_pitch > -30 && gyro_angle_pitch < 30){
    gyro_check_byte |= 0b00000100;
    if(gyro_angle_yaw < 0)trigger_axis = 0b10000011;
    else trigger_axis = 0b00000011;  }
    Serial.println(F("No angular motion is detected in the last 10 seconds!!! (ERROR 4)"));  }
  else
  if(movement == 1)roll_axis = trigger_axis;
  if(movement == 2)pitch_axis = trigger_axis;
  if(movement == 3)yaw_axis = trigger_axis; }
ISR(PCINT0_vect){
  current_time = micros();
  if(PINB & B00000001){                                    //Is input 8 high?
    if(last_channel_1 == 0){                               //Input 8 changed from 0 to 1
      last_channel_1 = 1;                                  //Remember current input state
      timer_1 = current_time;           } }
  else if(last_channel_1 == 1){                            //Input 8 is not high and changed from 1 to 0
    last_channel_1 = 0;                                    //Remember current input state
    receiver_input_channel_1 = current_time - timer_1;       }
  if(PINB & B00000010 ){                                   //Is input 9 high?
    if(last_channel_2 == 0){                               //Input 9 changed from 0 to 1
      last_channel_2 = 1;                                  //Remember current input state
      timer_2 = current_time;           }  }
  else if(last_channel_2 == 1){                            //Input 9 is not high and changed from 1 to 0
    last_channel_2 = 0;                                    //Remember current input state
    receiver_input_channel_2 = current_time - timer_2;       }
     if(PINB & B00000100 ){                                  //Is input 10 high?
    if(last_channel_3 == 0){                               //Input 10 changed from 0 to 1
      last_channel_3 = 1;                                  //Remember current input state
      timer_3 = current_time;         }  }                  //Set timer_3 to current_time
  else if(last_channel_3 == 1){                            //Input 10 is not high and changed from 1 to 0
    last_channel_3 = 0;                                    //Remember current input state
    receiver_input_channel_3 = current_time - timer_3;      }          //Channel 3 is current_time - timer_3
  if(PINB & B00001000 ){                                   //Is input 11 high?
    if(last_channel_4 == 0){                               //Input 11 changed from 0 to 1
      last_channel_4 = 1;                                  //Remember current input state
      timer_4 = current_time;           }  }                  //Set timer_4 to current_time
  else if(last_channel_4 == 1){                            //Input 11 is not high and changed from 1 to 0
    last_channel_4 = 0;                                    //Remember current input state
    receiver_input_channel_4 = current_time - timer_4;    }}//Channel 4 is current_time - timer_4
void intro(){
  Serial.println(F("================================================"));
  delay(1500);
  Serial.println(F(""));
  Serial.println(F("Your"));
  delay(500);
  Serial.println(F("  Multicopter"));
```

Қосымша 2

Arduino микроконтроллеріне жүктелетін ұшу контроллерінің бағдарламалық коды (скрипт)

```
#include <Wire.h>                    //Include the Wire.h library so we can communicate with the gyro.
#include <EEPROM.h>                  //Include the EEPROM.h library so we can store information onto the EEPROM
float pid_p_gain_roll = 1.3;         //Gain setting for the roll P-controller
float pid_i_gain_roll = 0.04;        //Gain setting for the roll I-controller
float pid_d_gain_roll = 18.0;        //Gain setting for the roll D-controller
int pid_max_roll = 400;              //Maximum output of the PID-controller (+/-)
float pid_p_gain_pitch = pid_p_gain_roll;  //Gain setting for the pitch P-controller.
float pid_i_gain_pitch = pid_i_gain_roll;  //Gain setting for the pitch I-controller.
float pid_d_gain_pitch = pid_d_gain_roll;  //Gain setting for the pitch D-controller.
int pid_max_pitch = pid_max_roll;    //Maximum output of the PID-controller (+/-)
float pid_p_gain_yaw = 4.0;          //Gain setting for the pitch P-controller. //4.0
float pid_i_gain_yaw = 0.02;         //Gain setting for the pitch I-controller. //0.02
float pid_d_gain_yaw = 0.0;          //Gain setting for the pitch D-controller.
int pid_max_yaw = 400;               //Maximum output of the PID-controller (+/-)
boolean auto_level = true;           //Auto level on (true) or off (false)
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;
```

Қосымша 2 (жалғасы)

```
boolean gyro_angles_set;
void setup(){
  for(start = 0; start <= 35; start++)eeprom_data[start] = EEPROM.read(start);
  start = 0;                                                  //Set start back to zero.
  gyro_address = eeprom_data[32];                             //Store the gyro address in the
variable.
  Wire.begin();                                               //Start the I2C as master.
  TWBR = 12;                                                  //Set the I2C clock speed to 400kHz.
  DDRD |= B11110000;                                          //Configure digital poort 4, 5, 6 and 7
as output.
  DDRB |= B00110000;                                          //Configure digital poort 12 and 13 as
output.
  digitalWrite(12,HIGH);                                      //Turn on the warning led.
  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();                                       //Set the specific gyro registers.
  for (cal_int = 0; cal_int < 1250 ; cal_int ++){             //Wait 5 seconds before continuing.
    PORTD |= B11110000;                                       //Set digital poort 4, 5, 6 and 7 high.
    delayMicroseconds(1000);                                  //Wait 1000us.
    PORTD &= B00001111;                                       //Set digital poort 4, 5, 6 and 7 low.
    delayMicroseconds(3000);                                  //Wait 3000us.
  }
  for (cal_int = 0; cal_int < 2000 ; cal_int ++){             //Take 2000 readings for
calibration.
    if(cal_int % 15 == 0)digitalWrite(12, !digitalRead(12));  //Change the led status to
indicate calibration.
    gyro_signalen();                                          //Read the gyro output.
    gyro_axis_cal[1] += gyro_axis[1];                         //Ad roll value to gyro_roll_cal.
    gyro_axis_cal[2] += gyro_axis[2];                         //Ad pitch value to gyro_pitch_cal.
    gyro_axis_cal[3] += gyro_axis[3];                         //Ad yaw value to gyro_yaw_cal.
    //We don't want the esc's to be beeping annoyingly. So let's give them a 1000us puls while
calibrating the gyro.
    PORTD |= B11110000;                                       //Set digital poort 4, 5, 6 and 7 high.
    delayMicroseconds(1000);                                  //Wait 1000us.
    PORTD &= B00001111;                                       //Set digital poort 4, 5, 6 and 7 low.
    delay(3);                                                 //Wait 3 milliseconds before the next loop.
  }
  gyro_axis_cal[1] /= 2000;                                   //Divide the roll total by 2000.
  gyro_axis_cal[2] /= 2000;                                   //Divide the pitch total by 2000.
  gyro_axis_cal[3] /= 2000;                                   //Divide the yaw total by 2000.
  PCICR |= (1 << PCIE0);                                      //Set PCIE0 to enable PCMSK0 scan.
  PCMSK0 |= (1 << PCINT0);                                    //Set PCINT0 (digital input 8) to
trigger an interrupt on state change.
  PCMSK0 |= (1 << PCINT1);                                    //Set PCINT1 (digital input 9)to
trigger an interrupt on state change.
  PCMSK0 |= (1 << PCINT2);                                    //Set PCINT2 (digital input 10)to
trigger an interrupt on state change.
  PCMSK0 |= (1 << PCINT3);                                    //Set PCINT3 (digital input 11)to
trigger an interrupt on state change.
```

```
  while(receiver_input_channel_3 < 990 || receiver_input_channel_3 > 1020 ||
receiver_input_channel_4 < 1400){
    receiver_input_channel_3 = convert_receiver_channel(3);              //Convert the actual
receiver signals for throttle to the standard 1000 - 2000us
    receiver_input_channel_4 = convert_receiver_channel(4);              //Convert the actual
receiver signals for yaw to the standard 1000 - 2000us
    start ++;                                                  //While waiting increment start whith every
loop.
    PORTD |= B11110000;                                                 //Set digital poort 4, 5, 6 and 7 high.
    delayMicroseconds(1000);                                            //Wait 1000us.
    PORTD &= B00001111;                                                  //Set digital poort 4, 5, 6 and 7 low.
    delay(3);                                                  //Wait 3 milliseconds before the next loop.
    if(start == 125){                                             //Every 125 loops (500ms).
      digitalWrite(12, !digitalRead(12));                               //Change the led status.
      start = 0;                                               //Start again at 0.
    }
  }
  start = 0;                                                   //Set start back to 0.
  battery_voltage = (analogRead(0) + 65) * 1.2317;
  loop_timer = micros();                                              //Set the timer for the next loop.
  digitalWrite(12,LOW);                                                //Turn off the warning led.
}
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;           //Correct the drift of the gyro pitch angle with the accelerometer
pitch angle.
  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; //Calculate the pulse
for esc 1 (front-right - CCW)
    esc_2 = throttle + pid_output_pitch + pid_output_roll + pid_output_yaw; //Calculate the pulse
for esc 2 (rear-right - CW)
    esc_3 = throttle + pid_output_pitch - pid_output_roll - pid_output_yaw; //Calculate the pulse
for esc 3 (rear-left - CCW)
    esc_4 = throttle - pid_output_pitch - pid_output_roll + pid_output_yaw; //Calculate the pulse
for esc 4 (front-left - CW)
    if (battery_voltage < 1240 && battery_voltage > 800){                   //Is the battery connected?
      esc_1 += esc_1 * ((1240 - battery_voltage)/(float)3500);              //Compensate the esc-1 pulse
for voltage drop.
      esc_2 += esc_2 * ((1240 - battery_voltage)/(float)3500);              //Compensate the esc-2 pulse
for voltage drop.
      esc_3 += esc_3 * ((1240 - battery_voltage)/(float)3500);              //Compensate the esc-3 pulse
for voltage drop.
      esc_4 += esc_4 * ((1240 - battery_voltage)/(float)3500);              //Compensate the esc-4 pulse
for voltage drop.
    }
    if (esc_1 < 1100) esc_1 = 1100;                                         //Keep the motors running.
```

```
 if (esc_2 < 1100) esc_2 = 1100;                                  //Keep the motors running.
   if (esc_3 < 1100) esc_3 = 1100;                                  //Keep the motors running.
   if (esc_4 < 1100) esc_4 = 1100;                                  //Keep the motors running.
   if(esc_1 > 2000)esc_1 = 2000;                                    //Limit the esc-1 pulse to 2000us.
   if(esc_2 > 2000)esc_2 = 2000;                                    //Limit the esc-2 pulse to 2000us.
   if(esc_3 > 2000)esc_3 = 2000;                                    //Limit the esc-3 pulse to 2000us.
   if(esc_4 > 2000)esc_4 = 2000;                                    //Limit the esc-4 pulse to 2000us.
  }
  else{
    esc_1 = 1000;                                                 //If start is not 2 keep a 1000us pulse for ess-
esc_2 = 1000;                                                    //If start is not 2 keep a 1000us pulse for ess-2.
    esc_3 = 1000;                                                 //If start is not 2 keep a 1000us pulse for ess-.
    esc_4 = 1000;            }
  if(micros() - loop_timer > 4050)digitalWrite(12, HIGH);
  while(micros() - loop_timer < 4000);
  loop_timer = micros();                                          //Set the timer for the next loop.
  PORTD |= B11110000;                                             //Set digital outputs 4,5,6 and 7 high.
  timer_channel_1 = esc_1 + loop_timer;                                 //Calculate the time of the faling
edge of the esc-1 pulse.
  timer_channel_2 = esc_2 + loop_timer;                                 //Calculate the time of the faling
edge of the esc-2 pulse.
  timer_channel_3 = esc_3 + loop_timer;                                 //Calculate the time of the faling
edge of the esc-3 pulse.
  timer_channel_4 = esc_4 + loop_timer;                                 //Calculate the time of the faling
edge of the esc-4 pulse.
  gyro_signalen();
  while(PORTD >= 16){                                             //Stay in this loop until output 4,5,6 and
7 are low.
    esc_loop_timer = micros();                                    //Read the current time.
    if(timer_channel_1 <= esc_loop_timer)PORTD &= B11101111;              //Set digital output 4
to low if the time is expired.
    if(timer_channel_2 <= esc_loop_timer)PORTD &= B11011111;              //Set digital output 5
to low if the time is expired.
    if(timer_channel_3 <= esc_loop_timer)PORTD &= B10111111;              //Set digital output 6
to low if the time is expired.
    if(timer_channel_4 <= esc_loop_timer)PORTD &= B01111111;              //Set digital output 7
to low if the time is expired.
  }
}
ISR(PCINT0_vect){
  current_time = micros();
  if(PINB & B00000001){                                           //Is input 8 high?
    if(last_channel_1 == 0){                                      //Input 8 changed from 0 to 1.
      last_channel_1 = 1;                                         //Remember current input state.
      timer_1 = current_time;                                     //Set timer_1 to current_time.
    }
  }
```

```
  else if(last_channel_1 == 1){
    last_channel_1 = 0;                                   //Remember current input state.
    receiver_input[1] = current_time - timer_1;                    }
  if(PINB & B00000010 ){                                    //Is input 9 high?
    if(last_channel_2 == 0){                                //Input 9 changed from 0 to 1.
      last_channel_2 = 1;                                   //Remember current input state.
      timer_2 = current_time;                               //Set timer_2 to current_time.
    }
  }
  else if(last_channel_2 == 1){
    last_channel_2 = 0;
    receiver_input[2] = current_time - timer_2;                  }
  if(PINB & B00000100 ){                                    //Is input 10 high?
    if(last_channel_3 == 0){                                //Input 10 changed from 0 to 1.
      last_channel_3 = 1;                                   //Remember current input state.
      timer_3 = current_time;         }  }
  else if(last_channel_3 == 1){
    last_channel_3 = 0;                                   //Remember current input state.
    receiver_input[3] = current_time - timer_3;        }
  if(PINB & B00001000 ){                                    //Is input 11 high?
    if(last_channel_4 == 0){                                //Input 11 changed from 0 to 1.
      last_channel_4 = 1;                                   //Remember current input state.
      timer_4 = current_time;        }  }
  else if(last_channel_4 == 1){
    last_channel_4 = 0;                                   //Remember current input state.
    receiver_input[4] = current_time - timer_4;        }}
void gyro_signalen(){
  if(eeprom_data[31] == 1){
    Wire.beginTransmission(gyro_address);
    Wire.write(0x3B);
    Wire.endTransmission();                                 //End the transmission.
    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 * -1;
  pid_last_yaw_d_error = pid_error_temp;
}
int convert_receiver_channel(byte function){
  byte channel, reverse;                                    //First we declare some local variables
  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];  //Store the low
value for the specific receiver input channel
  center = (eeprom_data[channel * 2 - 1] << 8) | eeprom_data[channel * 2 - 2]; //Store the center
value for the specific receiver input channel
  high = (eeprom_data[channel * 2 + 7] << 8) | eeprom_data[channel * 2 + 6];   //Store the high
value for the specific receiver input channel
  if(actual < center){
    if(actual < low)actual = low;
    difference = ((long)(center - actual) * (long)500) / (center - low);
    if(reverse == 1)return 1500 + difference;                          //If the channel is reversed
    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;                          //If the channel is reversed
    else return 1500 + difference;                                    //If the channel is not reversed
  }
  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(0x1B);
    Wire.endTransmission();                                           //End the transmission
    Wire.requestFrom(gyro_address, 1);                                //Request 1 bytes from the gyro
    while(Wire.available() < 1);                                      //Wait until the 6 bytes are received
    if(Wire.read() != 0x08){                                         //Check if the value is 0x08
      digitalWrite(12,HIGH);                                          //Turn on the warning led
      while(1)delay(10);                                             //Stay in this loop for ever
    }
    Wire.beginTransmission(gyro_address);
    Wire.write(0x1A);
    Wire.write(0x03);
    Wire.endTransmission();                                           //End the transmission with the gyro
  }
```

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ және ҒЫЛЫМ МИНИСТРЛІГІ
Қ.И.СӘТБАЕВ атындағы ҚАЗАҚ ҰЛТТЫҚ ТЕХНИКАЛЫҚ ЗЕРТТЕУ УНИВЕРСИТЕТІ

ҒЫЛЫМИ ЖЕТЕКШІНІҢ ПІКІРІ

дипломдық жоба

Абдраманов Бекарыс Жангелдинулы

5В071900- Радиотехника, электроника және телекоммуникация

Тақырыбына: **Arduino микроконтроллері негізіндегі квадрокоптерді құру**

Заманауи мультироторлы жүйелерге, олардың жұмысының жалпы принциптеріне шолу жасалған. Квадрокоптерді басқару жүйесінің математикалық моделі ұсынылып , практика жүзінде іске асырылған.

Квадрокоптердің басты ұшу контроллері ретінде Arduino микроконтроллері таңдалды. Бұл кез-келген микроэлектроникалық элементтерді басқару үшін пайдаланылатын жүйе. Arduino IDE бағдарламасын қолдану арқылы квадрокоптерді басқаруды жүзеге асыруға болады.

Бірінші бөлімде қазіргі заманғы мультироторлы жүйелерге әдебиеттік шолу жасалынды. Олардың заманауи шешімдері және квадрокоптер жұмысының жалпы принциптері жіктелді.

Екінші бөлімде квадрокоптерді басқару платасының негізгі блоктарын зерттеу қарастырылған. Компоненттердің техникалық сипаттамаларына тоқталып, Arduino микроконтроллерінің пайдалану мүмкіндіктері қарастырылды.

Үшінші бөлімде бағдарламалық қамтамасыз ету және практикалық іске асыру қарастырылды. MATLAB Simulink математикалық пакетінде квадрокоптер басқару жүйесінің моделін құрылды. Arduino IDE қосымшасында бағдарламалық кодқа талдау жасалынып, квадрокоптерді басқару жүйесінің моделін іске асыру және жұмыс үлгілері көрсетілді.

Студент дипломдық жобаны жасауда өздігінен жұмыс істеу қабілетін көрсете алды. Дипломант Абдраманов Бекарыс Жангелдинулы жұмыс істей алатынын көрсетті. Жалпы дипломдық жобаны "85/В/ жақсы", деп бағалап, ал студент Абдраманов Бекарыс Жангелдинулы 5В071900 - «Радиотехника, электроника және телекоммуникация» мамандығы бойынша техника және технологиялар бакалавры біліктілігіне сай.

Ғылыми жетекші
ЭТжҒТ кафедрасының техн.,
ғыл.канд
қауымдастырылған профессор
Л. Б. Илипбаева
25. 04. 2018

ҚАЗАҚСТАН РЕСПУБЛИКАСЫНЫҢ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ
СӘТБАЕВ УНИВЕРСИТЕТІ

СЫН – ПІКІР

дипломдық жоба

Абдраманов Бекарыс Жангелдинулы

5В071900- Радиотехника, электроника және телекоммуникация

Тақырыбына: **Arduino микроконтроллері негізіндегі квадрокоптерді құру**

Орындалды:
а) графикалық бөлімі 35 бет;
б) түсіндірме жазбасы 61 бет.

ЖҰМЫСҚА ЕСКЕРТУ ЖАСАУ

Дипломдық жобада Абдраманов Бекарыс Жангелдинулы Arduino микроконтроллері негізіндегі квадрокоптер құруды қарастырған. Дипломдық жұмыс келесі бөлімдерден тұрады:

Бірінші бөлімде қазіргі заманғы мультироторлық жүйелер қарастырылды. Олардың заманауи шешімдері мен квадрокоптер жұмысының жалпы принциптері жіктелді.

Екінші бөлімде квадрокоптерді басқару платасының негізгі блоктарына зерттеу жасалынған. Сонымен қатар, Arduino микроконтроллерінің пайдалану мүмкіндіктері анықталынып, ұшу уақыты бойынша батареяның қажетті сыйымдылығын есептеу жүргізілді.

Үшінші бөлімде бағдарламалық қамтамасыз ету және практикалық іске асыру қарастырылған. MATLAB Simulink математикалық пакетінде квадрокоптерді басқару жүйесінің моделін құру, сондай-ақ, квадрокоптерді басқару жүйесінің моделін іске асыру және жұмыс үлгілері көрсетілді.

Жұмыс бағасы

Жалпы, дипломдық жұмыс "85/ жақсы" деген бағаға, ал студент Абдраманов Бекарыс Жангелдинулы 5В071900 - РЭТ мамандығы бойынша техника және технологиялар «бакалавр» академиялық дәрежесіне ұсынылады.

Рецензия беруші
ҚазҰАУ, ЭҮжА каф.
доктор PhD.,
қауымдастырылған профессор
_____ Әлібек Н.Б.
«06» 05 2019 ж.