ОЛИМПИАДА ШКОЛЬНИКОВ «ШАГ В БУДУЩЕЕ»
ПО ПРОФИЛЮ «ИНЖЕНЕРНОЕ ДЕЛО»

регистрационный номер

Секция: Робототехнические системы

ПРИМЕНЕНИЕ МАШИННОГО ЗРЕНИЯ ДЛЯ АВТОМАТИЗИРОВАННОГО УПРАВЛЕНИЯ БПЛА

Автор: Ибрагимов Артем Дмитриевич

ГБОУ «Инженерная Школа №1581», 11 класс

Научный руководитель: Николаева Ольга Юрьевна

ГБОУ «Инженерная Школа №1581», преподаватель

подпись научного руководителя

Москва – 2024

РЕФЕРАТ

Расчетно-пояснительная записка 17 с, 9 иллюстраций, 2 таблицы, 1 приложение, 3 главы, 7 использованных источников.

БЕСПИЛОТНЫЙ ЛЕТАТЕЛЬНЫЙ АППАРАТ, МАШИННОЕ ЗРЕНИЕ, ПЛАТА ESP-32 CAM, ГИРОСКОП-АКСЕЛЕРОМЕТР MPU-6050, БИБЛИОТЕКА OPENCV, БИБЛИОТЕКА YOLO, МОДУЛЬ MULTIWII.

Объектом исследования является автоматизация принятия решений при управлении беспилотным летательным аппаратом.

Цель работы – исследование возможных вариантов применения машинного зрения для автоматизированного принятия решений при управлении беспилотным летательным аппаратом и разработка такого устройства.

Результатом работы является разработанный и реализованный на практике беспилотный летательный аппарат, способный получать команды управления как в ручном режиме, так и с использованием модуля машинного зрения.

В результате исследования, на практике доказана возможность использования машинного зрения для автоматизированного принятия решений при управлении беспилотным летательным аппаратом. Дальнейшим развитием проекта будет разработка алгоритмов поведения БПЛА в зависимости от решаемых с его помощью задач.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время, беспилотные летательные аппараты (БПЛА) являются крайне востребованным средством для решения различных задач. Немаловажной ступенью в развитии БПЛА является их автоматизация, которую можно реализовать, используя инструмент машинного зрения. Использование такого метода позволит упростить решение ряда задач во множестве сфер жизни.

Целью настоящей работы является исследование возможности применения машинного зрения для автоматизированного принятия решений при управлении беспилотным летательным аппаратом и разработка такого устройства. Эта цель достигнута за счет использования в данной работе свободно распространяемого программного обеспечения и открытых протоколов передачи информации.

Для достижения поставленной цели использовались имеющиеся в открытом доступе источники информации, проводился анализ документации на программное обеспечение, информационных материалов, публикуемых в тематических сообществах, публикаций, посвященных проектированию БПЛА и применению машинного зрения.

1. ПРОЕКТИРОВАНИЕ УСТРОЙСТВА, СООТВЕТСВУЮЩЕГО ЗАЯВЛЕННЫМ ЦЕЛЯМ

1.1. Конструкция и расположение элементов

Для достижения цели проекта были рассмотрены различные конструктивные варианты БПЛА. Ввиду наибольшей универсальности, из всех типов был выбран мультикоптер. Так же было необходимо определиться с количеством винтов. После анализа разных вариантов, был сделан выбор в пользу четырех винтов (квадрокоптер). Такая конструкция обеспечивает достаточную стабильность и легкость в управлении (в сравнении с трехвинтовым коптером), при этом имея простую конструкцию и низкую стоимость (относительно большего количества двигателей). Кроме того, из всех типов мультироторных БПЛА, именно про квадрокоптер имеется наибольшее количество информации и исследований. В качестве формы корпуса выбрана модель Hybrid X [1]. Такая модель позволяет разместить большое количество элементов в одной плоскости и обеспечивает высокую стабильность, что особенно важно, поскольку видеосъемка должна производится в хорошем качестве для грамотной работы системы распознавания образов.

Первоначальная модель корпуса (Рисунок 1) была смоделирована с упором на легкость и малые размеры конструкции.

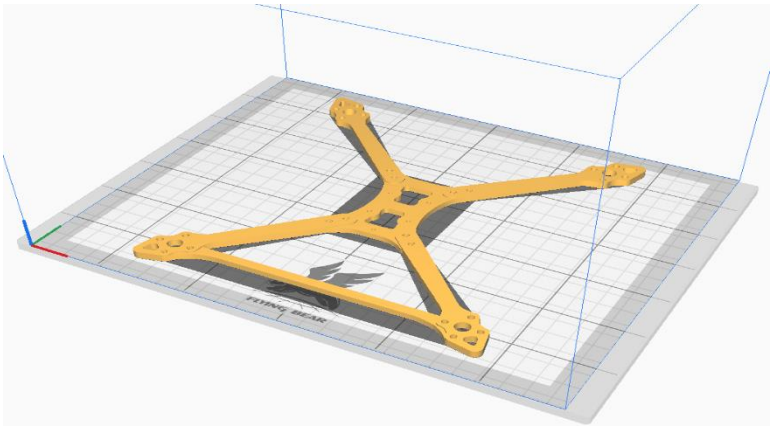


Рисунок 1. Первоначальная модель корпуса

Однако при испытаниях на прочность выяснилось, что модель обладает слишком большой гибкостью и малой прочностью. Такие характеристики не

позволили бы квадрокоптеру стабильно летать. Для решения возникшей проблемы была смоделирована новая модель (Рисунок 2), в которой были исправлены данные недостатки.

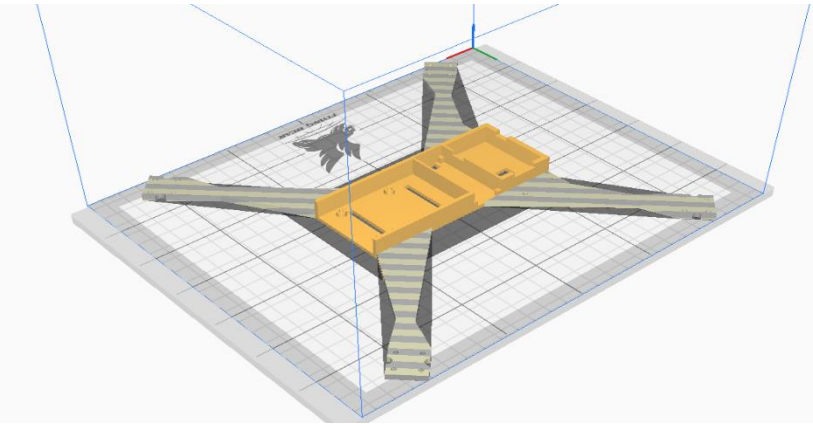


Рисунок 2. Окончательная модель корпуса

Благодаря геометрии новой модели, устройство больше не деформировалось при полете. Общая масса при этом увеличилась на 20 грамм.

1.2. Аппаратная часть полетного контроллера

Для создания оптимальной схемы полетного контроллера необходимо использовать набор элементов, достаточный для демонстрации возможностей машинного зрения как средства управления. Минимальный набор элементов, удовлетворяющий поставленным выше требованиям [2] следующий:

- гироскоп;
- акселерометр;
- вычислительное устройство;
- передатчик для связи с управляющим компьютером;
- камера.

Первоначально было решено использовать ESP-32 CAM в качестве основного вычислительного элемента (Рисунок 3).

Рисунок 3. Первоначальная структурная схема

Однако, недостаточное количество портов не позволило применить данный вариант. Тогда, в качестве бортового вычислительного устройства была выбрана плата Arduino Nano (Рисунок 4).

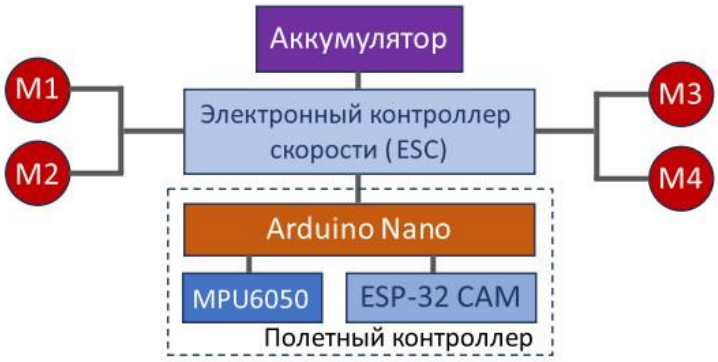


Рисунок 4. Итоговая структурная схема

В качестве камеры и передатчика используется вышеупомянутая плата ESP32-CAM, а в качестве гироскопа и акселерометра — датчик MPU-6050.

1.3.Система управления

Создание системы управления происходило в несколько этапов. Для начала, было решено реализовать систему ручного управления, для тестирования полетных характеристик готового устройства. Дистанционное управление осуществлялось по сети WI-FI, при помощи ESP-32 CAM [3]. После включения устройства, плата ESP создает точку доступа, при подключении к которой с управляющего компьютера, можно вручную управлять устройством. Команды с компьютера передаются на ESP-32 CAM, где при помощи протокола PPM их

принимает Arduino Nano (Рисунок 5). Для управления моторами, на Arduino установлена прошивка MultiWii [4]. Исходя из данных гироскопа-акселерометра и поступивших от передатчика команд, на устройстве, при помощи ранее упомянутой прошивки, вычисляется необходимая для каждого двигателя скорость вращения винтов. После этого, полетный контроллер направляет ШИМ-сигнал на ESC, с которого осуществляется непосредственная регуляция скорости вращения каждого мотора.

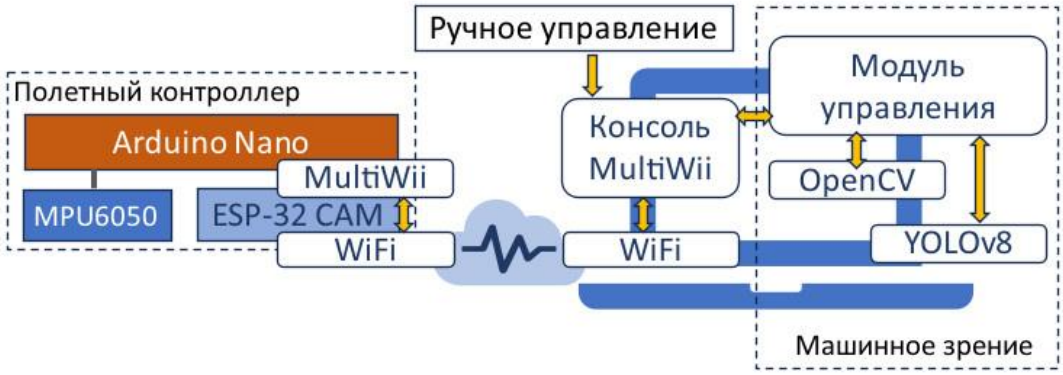
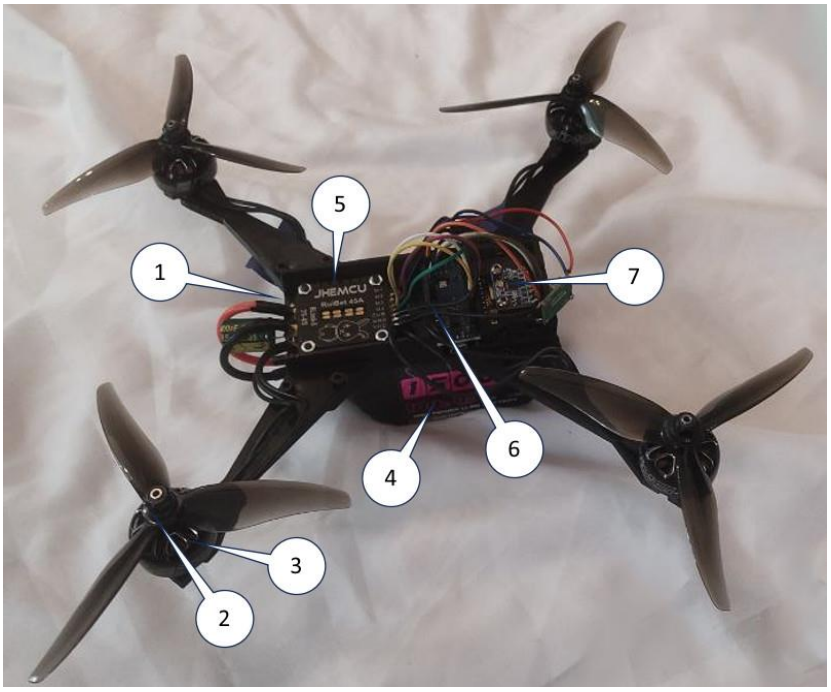


Рисунок 5. Схема управления

Вторым этапом создания системы управления стало внедрение машинного зрения. Для этого использовался язык программирования Python и библиотеки: OpenCV [5] и YOLOv8 [6]. OpenCV используется для захвата видеотрансляции с устройства, а YOLOv8 непосредственно как инструмент распознавания образов. Для этого, после обнаружения объекта с помощью YOLO, он начинает отслеживаться при помощи OpenCV. В зависимости от его координат, модулем управления на консоль MultiWii посылаются команды, цель которых — такое перемещение устройства, при котором объект будет находиться в центре кадра. От консоли MultiWii, управляющие воздействия поступают на устройство, где выполняются при помощи ранее описанной схемы передачи.

2. РЕАЛИЗАЦИЯ УСТРОЙСТВА

Моделью проекта является БПЛА винтового типа с четырьмя винтами(квадрокоптер). Корпус выполнен из пластмассы в формате Hybrid X. Полетный контроллер собран самостоятельно. На равном расстоянии от центра масс размещены моторы. Питание обеспечивается при помощи аккумулятора и платы понижения напряжения. Алгоритм управления аппаратом реализован на базе ESP-32 CAM. Видео передается на управляющий компьютер, далее на нем обнаруживаются образы, находятся их координаты и исходя из полученных данных принимается решение о дальнейшем полете квадрокоптера. Вся система принятия решений реализована на управляющем компьютере, задача же полетного контроллера состоит в регуляции скорости вращения моторов, для выполнения задач, отправленных с управляющего компьютера. Связь между этими устройствами осуществляется посредством технологии WI-FI, по радиоканалу в диапазоне 2,4 Ггц.



1. Корпус
2. Пропеллер GEMFAN 51466
3. Мотор Xing-e pro 2306 1700kv
4. Аккумулятор CNHL 1500 mAh
5. Регулятор оборотов JHEMCU RuiBet 45A
6. Плата Arduino Nano
7. Плата ESP-32 CAM и датчик MPU-6050

Рисунок 6. Устройство БПЛА

2.1.Корпус

Корпус смоделирован с учетом потребностей проекта, готовая модель была создана в программе Компас-3D и распечатана на 3D-принтере (Рисунок 7). В качестве материала для корпуса был выбран пластик PETG, так как он обладает

высокой износостойкостью, хорошим качеством печати и малой хрупкостью, в следствии чего является лучшим вариантом среди используемых для печати пластмасс. Смоделирован корпус был с учетом расположения элементов и центра масс, а высокий процент заполнения при печати обеспечил хорошую прочность. Лучи рамы напечатаны отдельно и прикреплены к раме при помощи винтов.

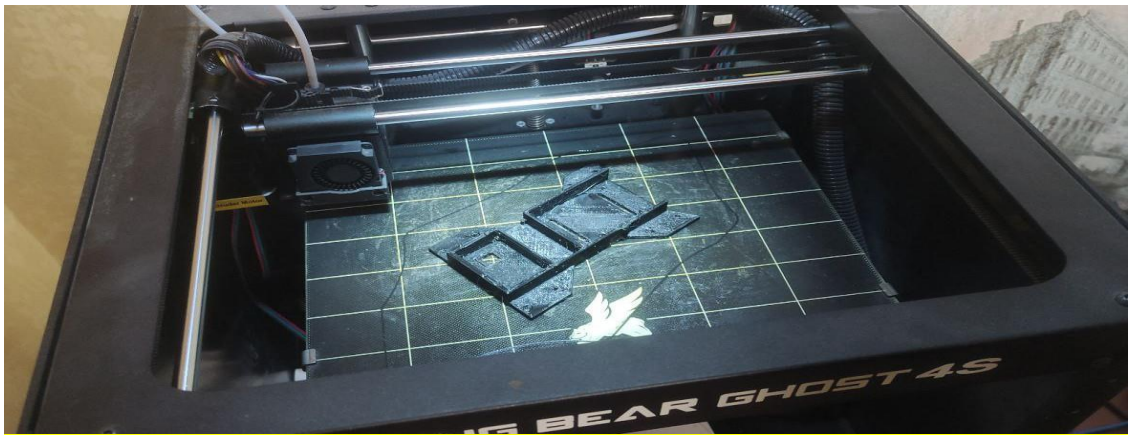


Рисунок 7. Изготовление корпуса

2.2.Полетный контроллер

Полетный контроллер построен, ввиду имеющихся навыков, на основе платы Arduino Nano (Рисунок 6, пункт 6). Она обеспечивает вычислительные мощности достаточные для выполнения поставленных задач. Так же, полетный контроллер оснащен трехосевым гироскопом-акселерометром - MPU-6050 (Рисунок 6, пункт 7). Этот датчик необходим для того, чтобы квадрокоптер мог сохранять горизонтальное и относительно устойчивое положение при зависании в воздухе.

2.3.Моторы и регулятор оборотов

В качестве регулятора оборотов (ESC) был выбран JHEMCU RuiBet 45A (Рисунок 6, пункт 5). Это ESC формата 4in1, который был выбран для упрощения конструкции устройства [7]. В качестве моторов были выбраны Xing-e pro 2306 1700kv (Рисунок 6, пункт 3). Эти моторы были выбраны в связи с хорошим качеством, относительно низкой ценой, балансом между мощностью и энергопотреблением, необходимыми для реализации проекта. Пропеллерами

были выбраны GEMFAN 51466 (Рисунок 6, пункт 2). Эти пропеллеры обеспечивают хорошую тягу и обладают высоким качеством. С этими винтами, моторы способны обеспечивать подъемную массу в 2 килограмма при 50% мощности, а при 100% - 5,8 килограмм. При массе квадрокоптера в 450 грамм, коптер способен поддерживать состояние зависания при работе моторов на 14% мощности, что обеспечивает среднее энергопотребление моторов в сумме 177,6 ватт.

2.4.Электропитание

В качестве аккумулятора был выбран CNHL на 1,5 А/ч и рабочим напряжением в 14,8 вольт (Рисунок 6, пункт 4). При таких характеристиках он способен обеспечить квадрокоптер длительностью полета в 10 минут. Этого достаточно, так как упор при выборе аккумулятора был сделан в пользу высокой мощности, а не долгой работоспособности. Сам аккумулятор подключается к упомянутому в части 2.3. ESC, откуда питание распределяется между моторами и платой понижения напряжения, от которой питание поступает на полетный контроллер.

2.5.Камера и передача данных

Передача видео и данных с датчиков на управляющий компьютер, реализована на базе ESP-32 CAM (Рисунок 6, пункт 7). Эта плата способна передавать видео в разрешении 720p, а также остальную необходимую информацию между полетным контроллером и управляющим компьютером, с минимальной задержкой. При включении квадрокоптера, ESP-32 CAM создает точку доступа WI-FI, после подключения к которой, при помощи компьютера можно осуществлять управление БПЛА. Схема передачи информации изображена на Рисунке 5.

2.6.Программное обеспечение

Программное обеспечение проекта состоит из двух частей: программы управления, которая находится на управляющем компьютере (Рисунок 8), и бортовой программы, которая расположена непосредственно на летательном аппарате. Машинное зрение, используемое в проекте, базируется на библиотеках OpenCV и YOLO (Приложение А). Программа бортового компьютера загружается в Arduino Nano. Информационное взаимодействие между программными модулями изображены на Рисунке 5.

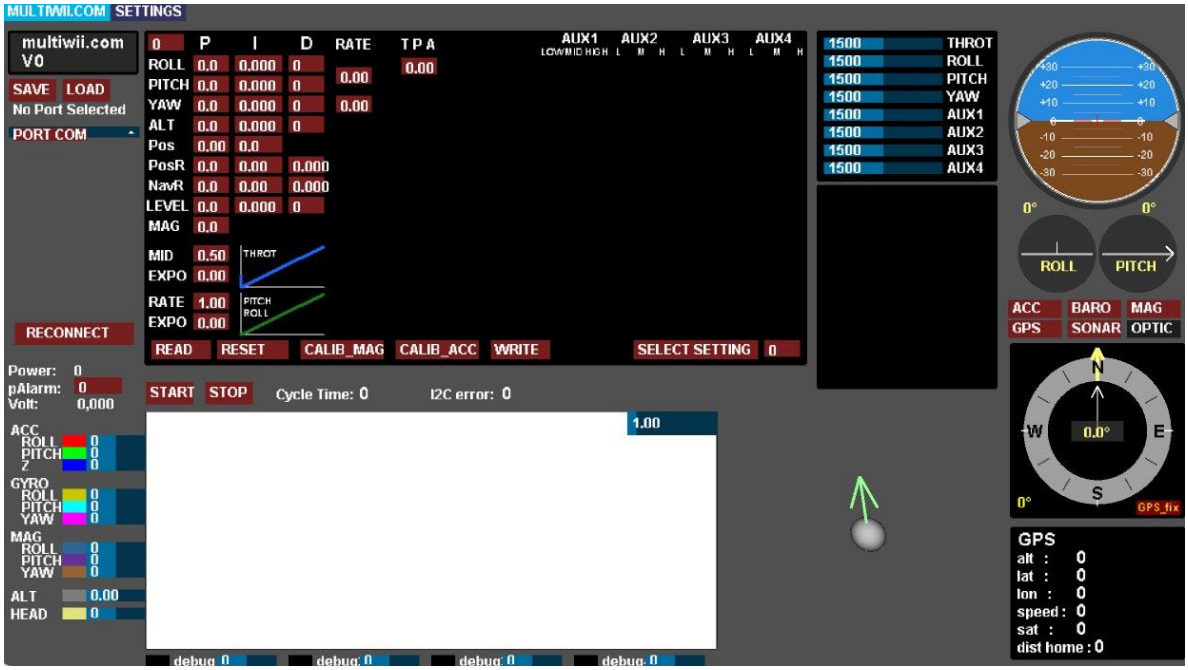


Рисунок 8. Интерфейс MultiWii

3. ИССЛЕДОВАНИЕ ПРИМЕНЕНИЯ МАШИННОГО ЗРЕНИЯ

В ходе исследований проведены автономные испытания элементов устройства, а также передачи сигналов между ними.

Аппаратная часть.

Все комплектующие корпуса крепятся надежно, без люфта. При нагрузке лучи рамы не деформируются. Моторы при подаче управляющего сигнала работают ровно, не вибрируют, не выкручиваются и соответствуют заявленным производителем характеристикам.

Получение данных с камеры, гироскопа и акселерометра.

Датчики откалиброваны при помощи программного обеспечения производителя. Показатели углов наклона по осям в состоянии зависания приведены на Рисунке 8.

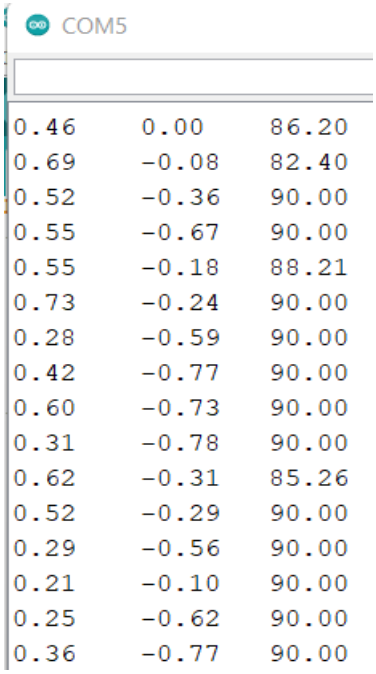


Рисунок 8. Углы наклона устройства по осям в состоянии зависания

Передача данных через модуль ESP-32 CAM на управляющий компьютер осуществляется стабильно. Видео может транслировать на управляющий компьютер с качеством 720p и минимальной частотой 40 кадров в секунду, на расстоянии до 80 метров. Таких показателей достаточно для работы устройства. Кроме того, модуль стабильно принимает данные от управляющего компьютера на том же расстоянии в 80 метров.

Обработка информации на компьютере:

OpenCV захватывает трансляцию без ощутимых потерь. Нейросеть YOLOv8s качественно распознает объекты с частотой 15 раз в секунду, чего достаточно для демонстрации работы устройства (Рисунок 9).

Рисунок 9. Распознавание объектов

Передача управляющего сигнала через Arduino Nano на регуляторе оборотов:

Диапазон сигналов широтно-импульсной модуляции на регуляторе оборотов совпадает с диапазоном на Arduino Nano, что позволяет выставлять на моторах все допустимые скорости. Зависимость мощности моторов от передаваемого показателя скважности приведена в Таблице 1.

Таблица 1 - Зависимость мощности от показателя скважности

Скважность	1000	**1140**	1500	1750	2000
Мощность двигателей	0%	**14%**[1]	50%	75%	100%

[1] Мощность состояния зависания

ЗАКЛЮЧЕНИЕ

В результате работы над проектом, были созданы БПЛА и система поддержки управления на основе машинного зрения и методе распознавания образов, построенных на базе искусственного интеллекта. Полученные в ходе испытаний результаты, демонстрируют возможность применения машинного зрения для управления БПЛА. Характеристики устройства и модуля управления приведены в Таблице 2.

Таблица 2 - Характеристики устройства и модуля управления

1. Устройство		
1.1	Масса, кг	0,45
1.2	Габариты, ДхШхВ, м	0,31x0,31x0,1
1.3	Энергопотребление в состоянии зависания, Вт	177,6
1.4	Время полета, с	600
2. Канал связи		
2.1	Дальность передачи информации, м	80
2.2	Параметры видеосигнала	720p, 40 кадр/с
2.3	Передаваемая информация датчиков	Ускорение и угловая скорость по осям XYZ.
3. Модуль управления		
3.1	Распознавание объектов, раз в секунду	15
3.2	Время формирования управляющего сигнала, с	<0,1

Дальнейшим развитием проекта будет разработка алгоритмов поведения БПЛА в зависимости от решаемых с его помощью задач.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Алексей Бойко «Основные конструкции беспилотников» // https://robotrends.ru/robopedia/osnovye-konstrukcii-bespilotnikov

2. Николай Радыш «Устройство беспилотных летательных аппаратов» // https://skvot.2035.university/ustrojstvo-bpla

3. «Руководство по использованию платы камеры ESP32-CAM» // https://docs.ai-thinker.com/en/esp32-cam

4. Виктор Денисенко «ПИД-регуляторы: вопросы реализации» // https://www.cta.ru/cms/f/374303.pdf

5. Комплекс документации «Библиотека компьютерного зрения с открытым исходным кодом OpenCV // https://docs.opencv.org/4.x/

6. Комплекс документации «Модель обнаружения объектов и сегментации изображений в реальном времени YOLOv8» // https://docs.ultralytics.com/ru

7. «Регулятор оборотов JHEMCU RuiBet 45A. Руководство по эксплуатации» // https://www.jhemcu.com/e_productshow/?58-JHEMCU-RuiBet-45A-3-6S-Dshot600-BLHELI_-S-four-in-one-electric-regulator-FPV-58.html

Приложение А

Программный код модуля управления.

```
import cv2import numpy as np
import urllib.request
url = 'http://ip/cam-hi.jpg'
cap = cv2.VideoCapture(url)whT=320
confThreshold = 0.5nmsThreshold = 0.3
classesfile='coco.names'classNames=[]
with open(classesfile,'rt') as f:    classNames=f.read().rstrip('\n').split('\n')

modelConfig = 'yolov8.cfg'modelWeights= 'yolov8.weights'
net =
cv2.dnn.readNetFromDarknet(modelConfig,modelWeights)net.setPreferableBackend(cv2.dn
n.DNN_BACKEND_OPENCV)
net.setPreferableTarget(cv2.dnn.DNN_TARGET_CPU)def findObject(outputs,im):
   hT,wT,cT = im.shape    bbox = []
   classIds = []    confs = []
   found_person = False    found_bird = False
   for output in outputs:        for det in output:
         scores = det[5:]           classId = np.argmax(scores)
         confidence = scores[classId]           if confidence > confThreshold:
            w,h = int(det[2]*wT), int(det[3]*hT)              x,y = int((det[0]*wT)-w/2),
int((det[1]*hT)-h/2)
            bbox.append([x,y,w,h])              classIds.append(classId)
            confs.append(float(confidence))
   indices = cv2.dnn.NMSBoxes(bbox,confs,confThreshold,nmsThreshold)    print(indices)
     for i in indices:
      i = i[0]       box = bbox[i]
      x,y,w,h = box[0],box[1],box[2],box[3]
      if classNames[classIds[i]] == 'person':          found_person = True
      elif classNames[classIds[i]] == 'bird':          found_bird = True
            cv2.rectangle(im,(x,y),(x+w,y+h),(255,0,255),2)
      cv2.putText(im, f'{classNames[classIds[i]].upper()} {int(confs[i]*100)}%', (x,y-10),
cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255,0,255), 2)

while True:    img_resp=urllib.request.urlopen(url)
   imgnp=np.array(bytearray(img_resp.read()),dtype=np.uint8)    im =
cv2.imdecode(imgnp,-1)
   sucess, img= cap.read()
blob=cv2.dnn.blobFromImage(im,1/255,(whT,whT),[0,0,0],1,crop=False)
   net.setInput(blob)    layernames=net.getLayerNames()
   outputNames = [layernames[i[0]-1] for i in net.getUnconnectedOutLayers()]
   outputs = net.forward(outputNames)
   findObject(outputs,im)
   cv2.imshow('IMage',im)
   cv2.waitKey(1)
```