Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

“САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ”

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

«РАЗРАБОТКА СИСТЕМЫ УПРАВЛЕНИЯ МАЛОГАБАРИТНЫМ БЕСПИЛОТНЫМ ВЕРТОЛЕТОМ НА ОСНОВЕ АВТОМАТНОГО ПРОГРАММИРОВАНИЯ И МАШИННОГО ОБУЧЕНИЯ»

Автор: Русин Никита Сергеевич _____

Направление подготовки (специальность): 01.04.02 Прикладная математика и информатика

Квалификация: магистр

Руководитель: Шалыто А.А., д.т.н., проф. _____

К защите допустить

Зав. кафедрой Васильев В.Н., д.т.н.. проф. _____

“___”_____ 20 ____г.

Санкт-Петербург, 2016 г.

Студент Русин Н.С. **Группа** М4239 **Кафедра** компьютерных технологий
Факультет информационных технологий и программирования

Направленность (профиль), специализация Технологии проектирования и разработки программного обеспечения

Консультант(ы):
а) _____ _____

б) _____ _____

Магистерская диссертация выполнена с оценкой _____

Дата защиты “____”_____20 ____г.

Секретарь ГЭК _____

Листов хранения _____

Демонстрационных материалов/Чертежей хранения _____

Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

“САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ”

УТВЕРЖДАЮ

Зав. кафедрой компьютерных технологий

д.т.н., проф. Васильев В.Н. _____

«____» «_____» 20___ г.

З А Д А Н И Е
НА МАГИСТЕРСКУЮ ДИССЕРТАЦИЮ

Студенту Русин Н.С**. Группа** М4239 **Кафедра** Компьютерных технологий

Факультет: Информационных технологий и программирования

Руководитель: Шалыто Анатолий Абрамович, д.т.н., проф., зав. кафедрой технологий программирования Университета ИТМО.

1 Наименование темы: Разработка системы управления малогабаритным беспилотным вертолетом на основе автоматного программирования и машинного обучения.

Направление подготовки (специальность): 01.04.02 Прикладная математика и информатика

Направленность (профиль): Технологии проектирования и разработки программного обеспечения

Квалификация: Магистр

2 Срок сдачи студентом законченной работы «___» «_____» 20_____г.

3 Техническое задание и исходные данные к работе

- Необходимо разработать усовершенствованный алгоритм навигации для системы управления беспилотным вертолетом.
- Сравнить полученный алгоритм с имеющимся решением.
- Нужно провести испытания разработанной технологии на практике.

4 Содержание магистерской диссертации (перечень подлежащих разработке вопросов)

- Описание существующих подходов
- Описание исходного решения, полученного ранее
- Разработка усовершенствованной версии алгоритма

– Реализация алгоритма навигации на практике

5 Перечень графического материала (с указанием обязательного материала)

Не предусмотрено

6 Исходные материалы и пособия

Русин Н.С. Разработка системы слепой посадки малогабаритного беспилотного вертолета., 2014
Шалыто А. А. Поликарпова Н. И. Автоматное программирование, СПб.: Питер, 2010, с. 176.
Шалыто А. А. Switch-технология. Алгоритмизация и программирование задач логического управления., СПб., Наука, 1998, с. 628.
Samek M., Practical Statecharts in C/C++: Quantum Programming for Embedded Systems. NY, CMP Books, 2002.
Quantum Leaps, LLC. About QM. http://www.state-machine.com/qm/

КАЛЕНДАРНЫЙ ПЛАН

№№ п/п	Наименование этапов магистерской диссертации	Срок выполнения этапов работы	Отметка о выполнении, подпись руков.
1	Анализ исходного решения	01.10.2015	
2	Разработка технологии проектирования	01.12.2015	
2	Разработка усовершенствованного решения	01.01.2016	
3	Исследование полученного решения	01.02.2016	
4	Изучение возможности реализации на практике	01.03.2016	
5	Реализация аппаратно-программного комплекса	01.05.2016	
6	Написание пояснительной записки	01.06.2016	

8 Дата выдачи задания «____» «_____» 20____г.

Руководитель_____
(подпись)

Задание принял к исполнению_____ «____» «_____» 20____г.
(подпись)

Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

“САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ”

АННОТАЦИЯ

МАГИСТЕРСКОЙ ДИССЕРТАЦИИ

Студент: Русин Никита Сергеевич
Наименование темы ВКР: Разработка системы управления малогабаритным беспилотным вертолетом на основе автоматного программирования и машинного обучения.
Наименование организации, где выполнена ВКР: Университет ИТМО

ХАРАКТЕРИСТИКА ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

1 Цель исследования: разработка системы управления малогабаритным беспилотным вертолетом на основе автоматного программирования.
2 Задачи, решаемые в ВКР: реализация системы управления малогабаритным беспилотным вертолетом в физической модели
3 Число источников, использованных при составлении обзора: 5
4 Полное число источников, использованных в работе: 13
5 В том числе источников по годам

Отечественных			**Иностранных**		
Последние 5 лет	От 5 до 10 лет	Более 10 лет	Последние 5 лет	От 5 до 10 лет	Более 10 лет
	1	1	10	3	

6 Использование информационных ресурсов Internet: да, пять
7 Использование современных пакетов компьютерных программ и технологий
В разделе 3 использовался полный стек разработки и компиляции программ на основе GNU GCC
8 Краткая характеристика полученных результатов: разработана и реализована система навигации для малогабаритного беспилотного вертолета

9 Полученные гранты, при выполнении работы:

отсутствуют.

10 Наличие публикаций и выступлений на конференциях по теме выпускной работы: да

а) IV Всероссийский конгресс молодых ученых
Тема: Система слепой посадки малогабаритного беспилотного вертолета

б) LV научная и учебно-методическая конференция Университета ИТМО.
Тема: Технология разработки программного обеспечения систем управления ответственными объектами на основе методов машинного обучения и конечных автоматов.

Выпускник Русин Н.С. _____

Руководитель Шалыто А.А. _____

“_____”_____20___г.

ОГЛАВЛЕНИЕ

Введение 6

Глава 1. Описание исходного решения 7

1.1. Актуальность 7

1.2. Обзор аналогов 8

1.3. Принцип работы 10

1.4. Алгоритм навигации 12

1.5. Построение управляющих автоматов 15

1.6. Эмулятор комплекса 21

1.7. Достоинства и недостатки 25

1.8. Выводы по главе 1 25

Глава 2. Усовершенствованное решение 26

2.1. Постановка задачи 26

2.2. Построение автоматов 26

2.3. Аналитическое решение 28

2.4. Улучшенный эмулятор комплекса 32

2.5. Сравнение с исходным решением 33

2.6. Выводы по главе 2 34

Глава 3. Реализация программно-аппаратного комплекса 36

3.1. Устройство комплекса 36

3.2. Блок регистрации датчиков 40

3.3. Главный модуль управления 42

3.4. Тестирование, настройка и отладка 43

3.5. Возможные улучшения 44

3.6. Выводы по главе 3 45

Заключение 46

Приложения 47

Список использованных источников 53

ВВЕДЕНИЕ

Идея использовать беспилотные летательные аппараты появилась практически одновременно с созданием пилотируемых летательных аппаратов. С тех пор прошло много времени, и аппараты обоих типов стали намного совершеннее.

На сегодняшний день применение беспилотных летательных аппаратов различными компаниями, армией, пользователями издерживаются большим числом факторов: стоимостью, сложностью в эксплуатации, недостаточным качеством полезной нагрузки и т.д.

Современный беспилотный аппарат представляет из себя крайне сложный комплекс, включающий систему автоматического управления, систему навигации, различные специфические средства.

Круг применения такого типа аппаратов значительно шире, чем можно себе представить. Это и аэросъемка, и исследование, и спасательные операции, и многое другое.

В случае массового внедрения беспилотных летательных аппаратов обязательно встает вопрос об облегчении эксплуатации через автоматизацию процесса управления таким образом, чтобы человек в нем не выполнял никаких задач, кроме контроля.

Помимо этого, «узким местом» в современных беспилотных вертолетах является продолжительность автономного полета, то есть источник питания. Ситуация значительно улучшилась с изобретением литий-ионных, а в последствии и литий-полимерных аккумуляторов, но это не решило проблему полностью.

Данная работа базируется на бакалаврской работе автора и является ее логическим продолжением.

Цель работы заключается в построении улучшенного решения, которое могло бы быть внедрено в рабочую модель квадрокоптера.

ГЛАВА 1. ОПИСАНИЕ ИСХОДНОГО РЕШЕНИЯ

В данной главе описывается опыт построения системы слепой посадки малогабаритного беспилотного вертолета. В рамках предыдущей работы стояла задача разработка нового метода навигации и исследование его на предмет стабильности и устойчивости к помехам.

В главу вошел обзор уже имеющихся подобных систем и их краткое описание. Также описывается полученный алгоритм посадки с применением автоматного подхода, процесс эмуляции алгоритма навигации и перечисление его особенностей.

1.1. Актуальность

В последнее время ДПЛА (Дистанционно пилотируемые летательные аппараты) набирают все большую популярность. Существует очень много различных вариантов. Эти летательные аппараты могут выполнять очень широкий круг задач, начиная от видеосъемки и заканчивая автоматической доставкой грузов.

Для выполнения некоторых задач, таких как постоянный мониторинг местности, требуется выполнение замкнутого цикла автономной работы этого аппарата, как показано на рисунке 1

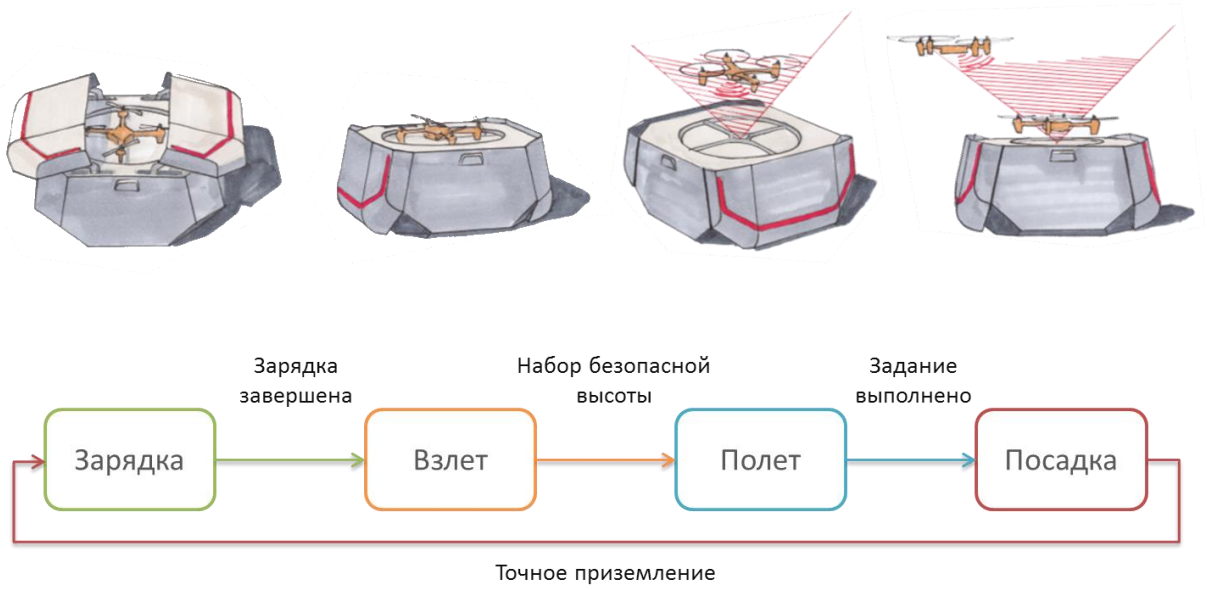


Рисунок 1 – Цикл автономной работы ДПЛА.

Сейчас уже решены задачи, касающиеся безопасной зарядки аккумулятора, благодаря изобретению нового типа батарей и управляющих схем. Взлет и полет по заданному маршруту реализуются управляющими контроллерами, которые используют бесплатформенные инерциальные навигационные системы и системы спутниковой навигации GPS/GLONASS.

Однако, для реализации всего цикла работы в автономном режиме необходимо наличие метода посадки, обладающего достаточной точностью, чтобы сесть на контактную площадку. Без этого ДПЛА не сможет приступить к зарядке аккумулятора устройства, и, как следствие, цикл не будет замкнут.

1.2. Обзор аналогов

В работе [1] рассматривается метод посадки на базе компьютерного зрения. Навигация осуществляется путем распознавания, красного маркера для определения направления движения.

Принцип посадки, описанный в работе [2], использует похожую идею, за исключением того, что используется инфракрасная камера и сетка светодиодов.

Работа этой системы показана на рисунке 2.

Рисунок 2 – Работа системы посадки, описанной в [2]

Похожим образом работает и система IRLock [3], она является совсем новой на момент написания этой работы. Ее работа показана на рисунке 3.

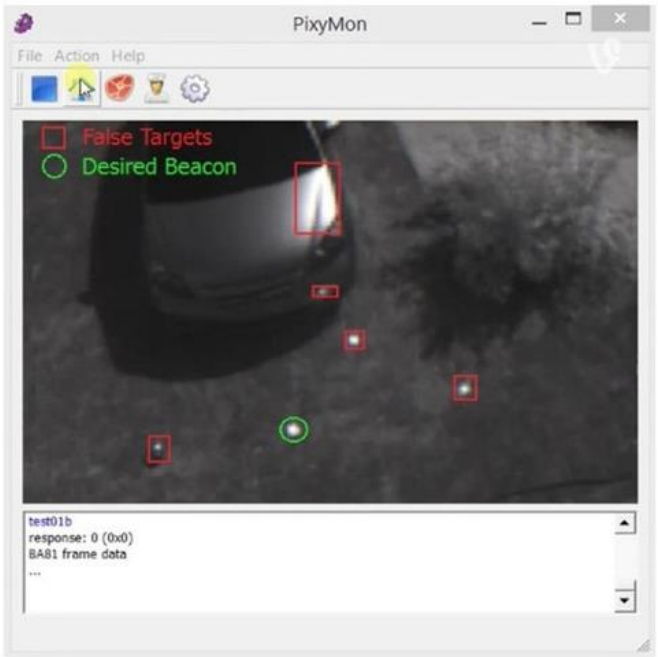


Рисунок 3 – Работа системы IRLock.

В работе [4] произведено полное описание того, как использование таких маркеров позволяет определять направление до вектора площадки.

Однако, принцип работы, на котором основаны эти методы, имеет существенные физические ограничения по дальности работы, главным из которых является разрешение камеры, поскольку она не имеет достаточную детализацию для определения маркера, например, с 10-15 метров. Более того необходима достаточная вычислительная мощность для определения образа маркера на изображении перед тем, как принимать решение, куда двигаться.

Более сложные системы, такие как [5], [6] основаны на радиолокации и использовании большого числа сложных дополнительных устройств, а также точного оборудования для осуществления навигации по радиосигналам. Они используются в основном в военной технике, где эти ограничения не существенны.

Существуют также системы, основанные на построение радиолокационной карты, системы, обеспечивающие лишь поддержку при посадке, но они, как правило, также очень громоздки, имеют высокую стоимость и сложны.

Различные факторы не позволяют использовать данные системы в тех случаях, когда необходима компактность и простота используемого оборудования. При автономном регулярном мониторинге местности с ДПЛА каждое усложнение будет дополнительным риском отказа системы, прерывания цикла, что потребует вмешательства человека для устранения неисправности.

1.3. Принцип работы

Система состоит из двух основных модулей: посадочной площадки с вращающейся лазерной плоскостью и управляющим модулем на борту ДПЛА (дистанционно пилотируемом летательном аппарате), снабженным набором датчиков. Схема расположения модулей и датчиков изображена на рисунке 4.

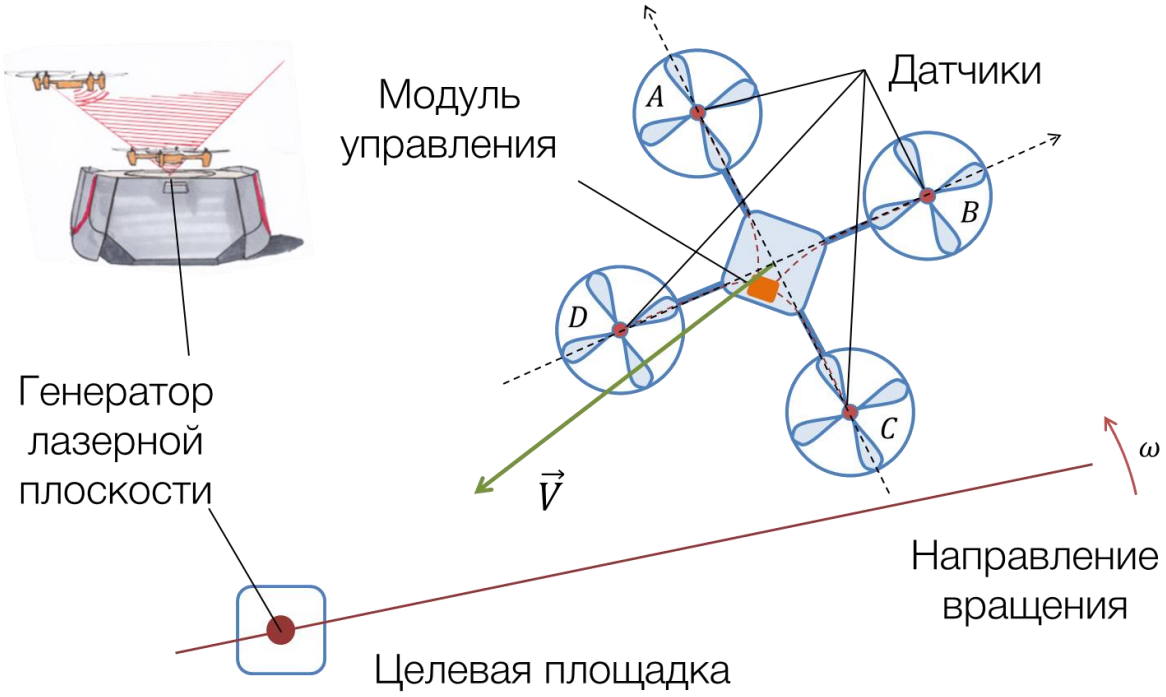


Рисунок 4 – Принцип работы. Вид сверху.

Стоит отметить, что число и расположение датчиков определяется выбранным алгоритмом и остается неизменным с момента установки на ДПЛА. При этом одному и тому же расположению датчиков может соответствовать несколько алгоритмов.

В модуль, отвечающий за площадку посадки входит контактная поверхность и вращающийся источник лазерного излучения с линзой, формирующей плоскость. При этом лазерное излучение дополнительно модулируется определенной частотой для возможности точного детектирования прохождения луча, а также для измерения приблизительного расстояния до площадки посадки.

Когда лазерная плоскость проходит через датчик, информация об этом фиксируется и ему сопоставляется символ и время регистрации. Далее эта пара параметров подается в управляющий автомат.

Управляющий автомат строится с учетом расположения датчиков, а также дополнительных опций детектирования. Определение направления движения осуществляется по поступившему символу и времени регистрации, и выглядит как

вектор направления движения к оси вращения плоскости. Примером такого вектора может служить вектор $\vec{V}$ на рисунке 4.

1.4. Алгоритм навигации

При прохождении вращающейся плоскости фиксируется срабатывание датчиков, которые можно объединить в последовательность фиксаций – сигналов. Будем считать, что в зону охвата плоскости попадают все датчики, поскольку этот случай является основным. Количество датчиков обозначим за n.

Сопоставим каждому датчику символ. Назовем один из датчиков синхронизирующим, пусть ему сопоставлен символ z. В реальности, так как происходит непрерывная подача этих символов, то это будет бесконечный поток F, среди которых нужно выявить последовательности, отвечающие за то, в каком секторе относительно ДПЛА находится площадка посадки.

Рассмотрим алгоритм построения контрольной последовательности для сектора:

а) Выбираем подпоследовательность P длиной n из F

б) Удваиваем P и получаем последовательность PP длины $2n$

в) В последовательности PP находим синхронизирующий символ z и, начиная с него, выделяем подпоследовательность $Z \in PP$, длиной n.

г) Полученная последовательность Z является контрольной для исходного сектора.

Используя этот алгоритм для каждого сектора – получаем контрольную последовательность.

Назовем классом секторов набор секторов с одинаковой контрольной последовательностью. Для каждого класса определим вектор движения, направленный из центра ДПЛА. Установим соответствие между каждой контрольной

последовательностью и соответствующей ей классом секторов. Назовем такое соответствие картой секторов. Пример такой карты изображен на рисунке 5.

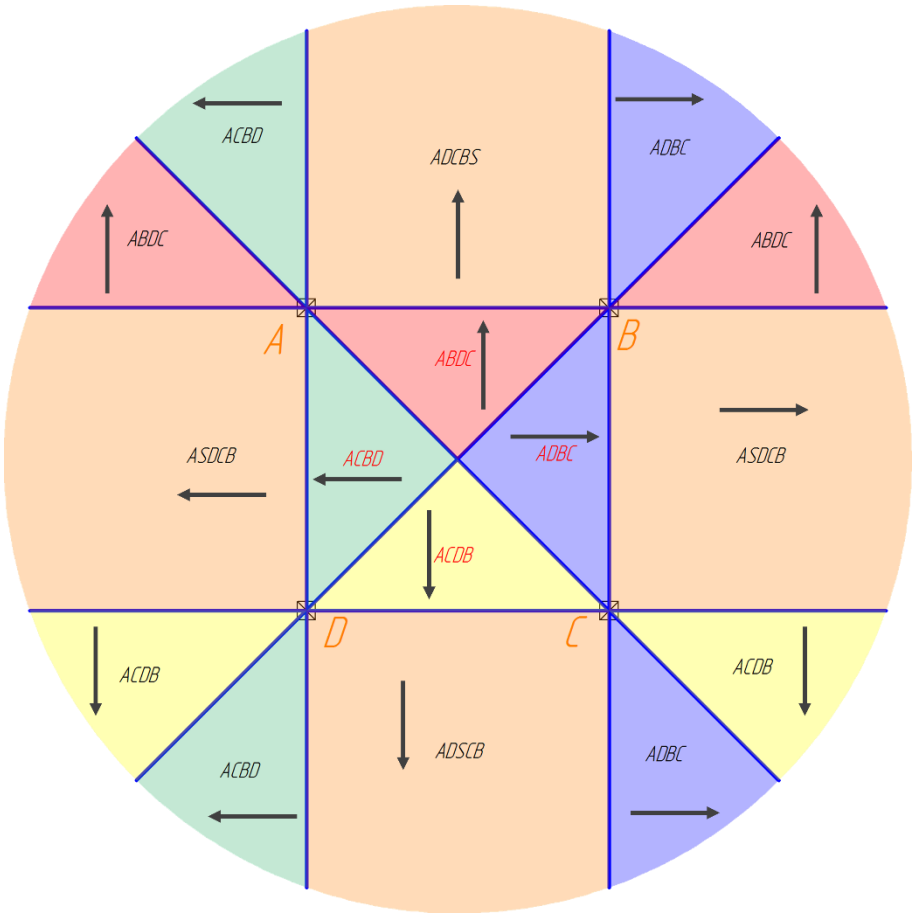


Рисунок 5 – Карта секторов для выбранного расположения датчиков.

Рассмотрим карту подробнее. Если пометить сектора с совпадающими контрольными последовательностями, то становится заметно, что каждому сектору соответствует особый класс, имеющий вектор направления.

Главным же критерием сходимости предлагаемого метода посадки является отсутствие противонаправленных векторов для соседних секторов, иначе при попадании на границу возникнет колебание БПЛА на месте без необходимого смещения к посадочной площадке.

В результате должно получиться так, чтобы последовательность переходов между секторами по соответствующим им векторам сходилась к центру карты – приводила квадрокоптер к центру посадочной площадки.

Однако существуют классы секторов с нулевым вектором направления движения. При данном расположении такой класс образуют «прямые» открытые сектора:

- $\{(CD)(AD)(AB)\}$;
- $\{(AD)(AB)(BC)\}$;
- $\{(AB)(BC)(CD)\}$;
- $\{(BC)(CD)(AD)\}$.

Эти сектора создают большую проблему, поскольку при появлении соответствующей им контрольной последовательности невозможно определить направление движения.

Эта проблема требует решения. Одним из возможных путей устранения этой проблемы стало добавление псевдодатчика, обозначим его S. Пусть такой датчик срабатывает, если на протяжении заданного времени отсутствовала активность с основных сенсоров. Будем называть новый датчик «пробелом», потому что в потоке символов срабатывание этого датчика разграничивает периоды прохождения лазерной плоскости.

Использование этого приема позволяет уточнять направление для конкретного сектора в проблемных классах, тем самым разбивая их на новые, состоящие из одного сектора, где можно будет точно определить вектор направления.

В дальнейшем карта секторов используется для построения автомата навигации. В него подается непрерывной поток символов, обозначающих сработавшие датчики на летательном аппарате. В случае, если определяется последовательность, отвечающая за один из классов секторов, то автомат выдает вектор движения, перемещение по которому приближает посадочную площадку.

1.5. Построение управляющих автоматов

Если рассматривать систему навигации в целом, система управления будет состоять из нескольких автоматов, взаимодействующих между собой. Их взаимодействие изображено на рисунке 6.

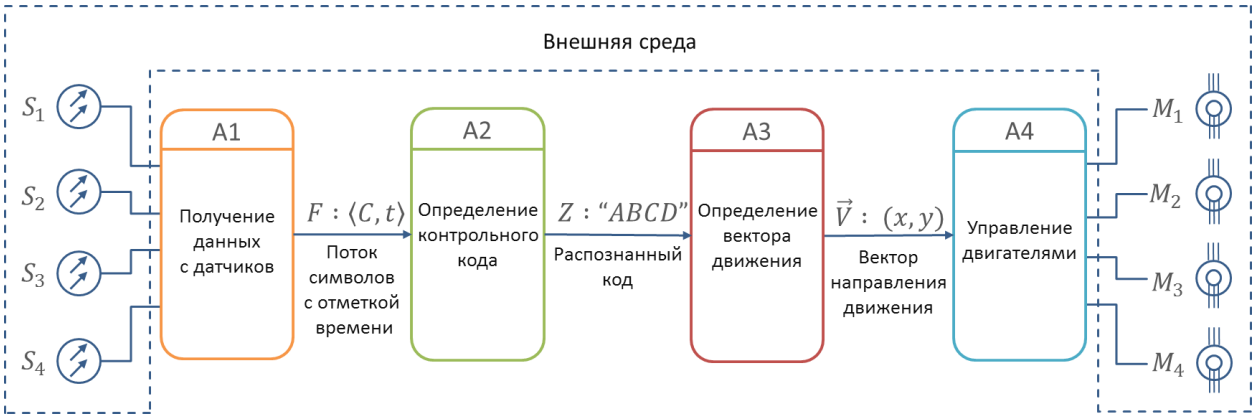


Рисунок 6 – Общая схема взаимодействия автоматов.

Автомат A1 отвечает за регистрацию показаний с датчиков. Он сопоставляет при срабатывании каждому из них символ и время регистрации. В задачу этого автомата также входит регистрация срабатываний псевдодатчиков.

Автомат A2 является автоматом распознавания кода. Этот автомат выдает на выход код, обозначающий класс секторов, в котором находится площадка.

Автомат A3 сопоставляет каждому коду вектор направления движения в координатах квадрокоптера, рассчитанный заранее.

Автомат A4 управляет двигателями квадрокоптера.

Наиболее интересным является построение автоматов распознавания A2 и выбора вектора A3. Основные особенности работы A2 можно описать так:

Если символ повторился, то вернуть предыдущий результат.

Код с большим число псевдодатчиков имеет преимущество над основными кодами в равных условиях.

Псевдодатчики не встраиваются в последовательности основных кодов.

Для построения автоматов в качестве исходных данных требуется карта контрольных последовательностей – отображение из класса контрольных последовательностей в класс направляющих векторов. Обозначим исходную карту как $codes$.

В рабочем состоянии автомату на вход подается символ c, по которому, в зависимости от предыстории, должно быть принято решение о выборе кода $code$ из набора $codes$. Затем выбранному $code$ сопоставляется вектор направления $vector$.

Перейдем к построению автомата A2.

В качестве состояний в этом автомате выступают символы, обозначающие датчики. Переходы и дополнительная информация генерируются следующим образом:

Так как в используемых схемах коды могут встречаться, начиная с любого символа, то для каждого кода $code$ перед обработкой выполняется приписывание справа первого символа.

Для каждого кода $code$ последовательно по каждой паре соседних символов ab выполняется создание между состояниями, соответствующими этим символам, перехода $A \to B$, если такого перехода не было до этого. Затем добавляется информация о том, что по этому переходу можно пройти, используя код $code$.

Также для того, чтобы исключить обязательность включения в коды псевдосенсоров, для каждого кода $code$, если он не включает символ этого самого псевдосенсора, выполняется добавление информации о возможности прохода по коду через все переходы в состояние, соответствующее псевдосенсору.

Таким образом, граф переходов представляет собой почти полный граф, в котором каждому ребру $A \to B$ соответствует информация, какие коды содержат пару ab.

Алгоритм получения реакции выглядит так:

а) Если на вход поступает символ c, соответствующий текущему состоянию, то он игнорируется и возвращается результат предыдущего реагирования. Это необходимо для исключения множественных ложных срабатываний сенсоров.

б) Из дополнительной памяти автомата берется набор возможных кодов $possible$ и счетчиков для них.

в) Для каждого кода из набора $current$ на переходе, который вызывает полученный символ c, увеличиваем счетчик, если такой уже имеется в наборе *possible* и добавляем в набор в ином случае.

г) В случае, если код встречается в наборе $possible$, но отсутствует в наборе $current$, удаляем его из набора возможных кодов $possible$. Исключение составляют коды, не содержащие символа псевдокода t в случае, если $t = c$.

д) Для тех кодов из набора *possible*, счетчик для которых достиг длины кода, производим добавление в набор *actions* и уменьшаем счетчик на единицу.

е) Из набора *actions* выбираем любой код $action$, содержащий максимальное число символов, соответствующих псевдодатчикам.

ж) Выбираем результирующий вектор $result$, соответствующий коду $action$ из набора $codes$, который используется при построении и инициализации автомата.

з) Выполняем переход к новому состоянию и обновляем информацию в указанной выше памяти, возвращая в качестве результата $result$.

Таким образом, получается автомат, в дополнительной памяти которого содержатся счетчики для возможных кодов, информация о предыдущем состоянии автомата и о предыдущем результате.

Общее устройство и примеры работы автомата изображены на рисунках 7–9.

1	ABDC	••••
2	ACDB	••
3	ACBD	•
4	ADBC	•
5	ADCBS	•
6	ADCSB	•
7	ADSCB	•
8	ASDCB	•
w	ABCSD	•
x	ABCDS	•••
y	ASBCD	•••
z	ABSCD	••

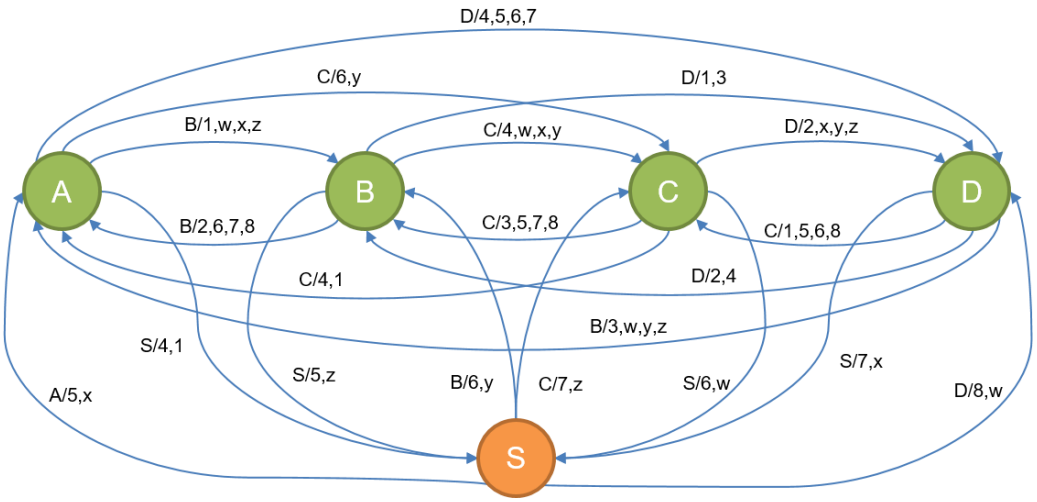


Рисунок 7 – Граф переходов автомата A2. Промежуточное состояние.

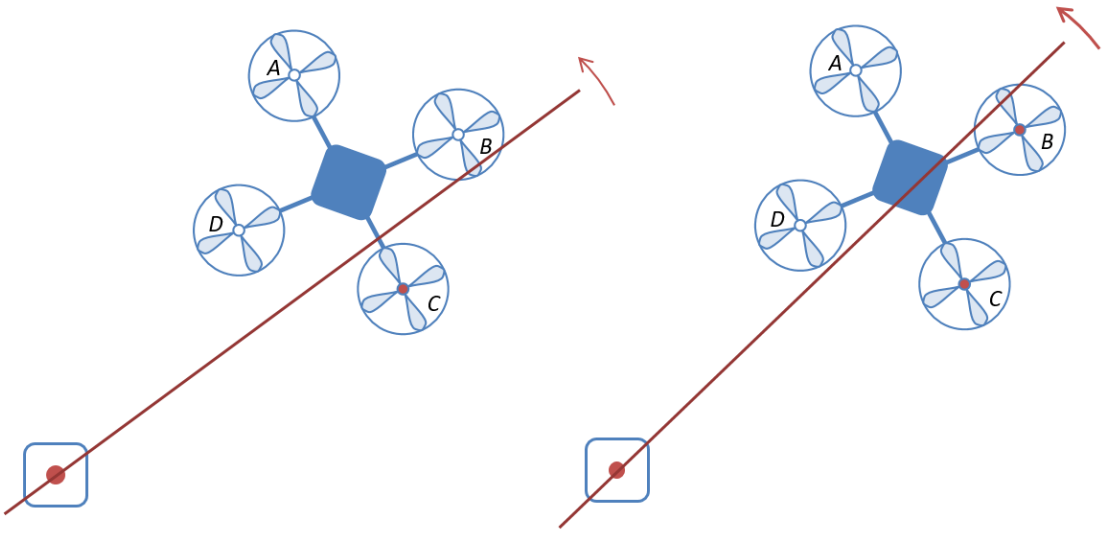


Рисунок 8 – Процесс получения последовательности.

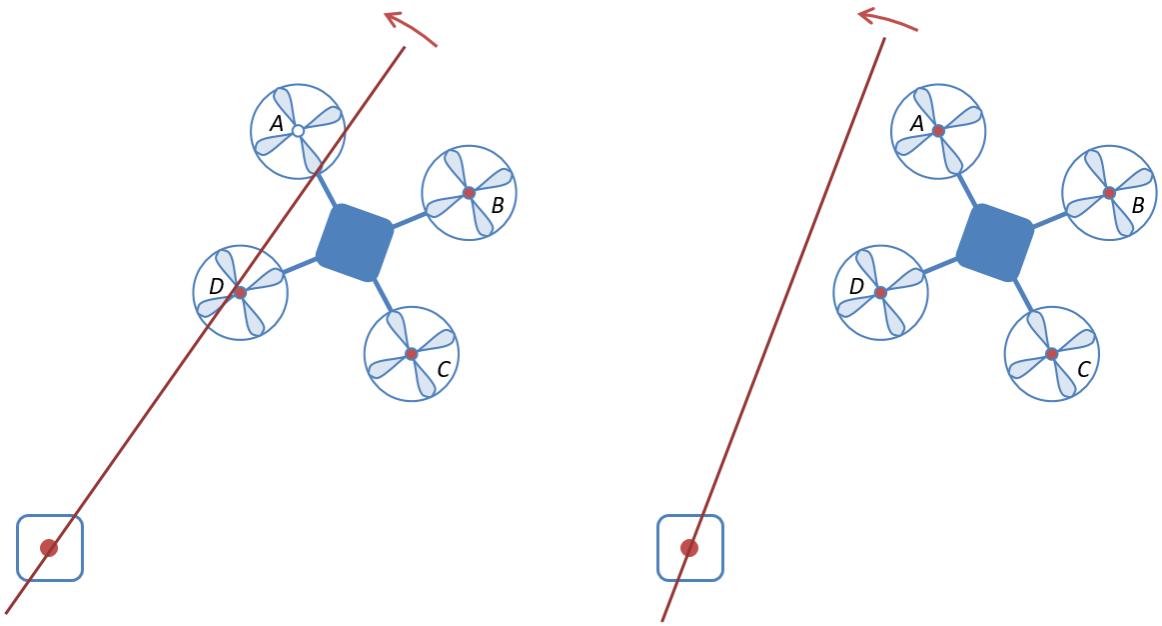


Рисунок 9 – Процесс получения последовательности.

При прохождении луча последовательно регистрируются показания с датчиков $CBDA$. Эта последовательность по одному символу и подается в автомат распознавания А3. При получении символа, автомат делает переход по символу в одноименное состояние и делает записи в ячейках, обозначенных на переходе. Полностью процесс заполнения ячеек можно наблюдать на рисунках 10–12.

1	ABDC
2	ACDB
3	ACBD ••
4	ADBC
5	ADCBS •
6	ADCSB •
7	ADSCB •
8	ASDCB •
w	ABCSD
x	ABCDS
y	ASBCD
z	ABSCD

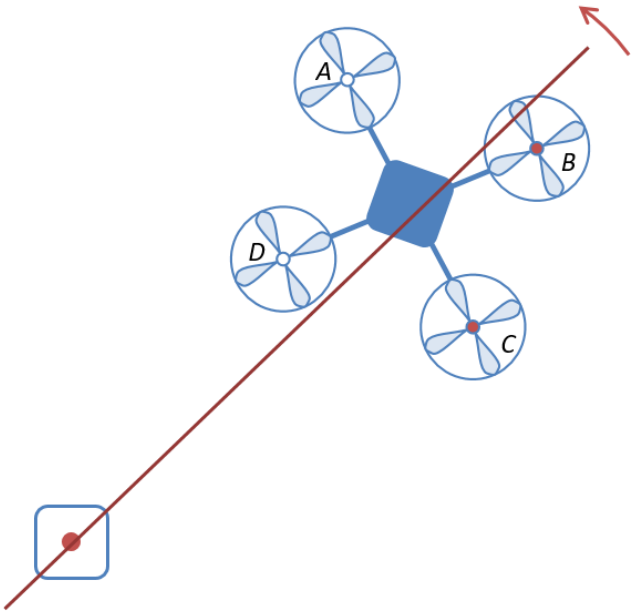


Рисунок 10 – Процесс обработки последовательности.

1	ABDC •
2	ACDB
3	ACBD •••
4	ADBC
5	ADCBS
6	ADCSB
7	ADSCB
8	ASDCB
w	ABCSD
x	ABCDS
y	ASBCD
z	ABSCD

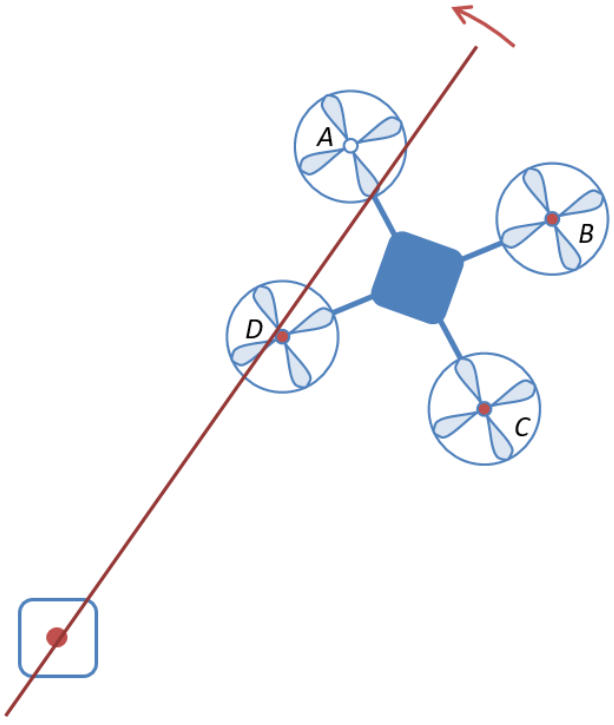


Рисунок 11 – Процесс обработки последовательности.

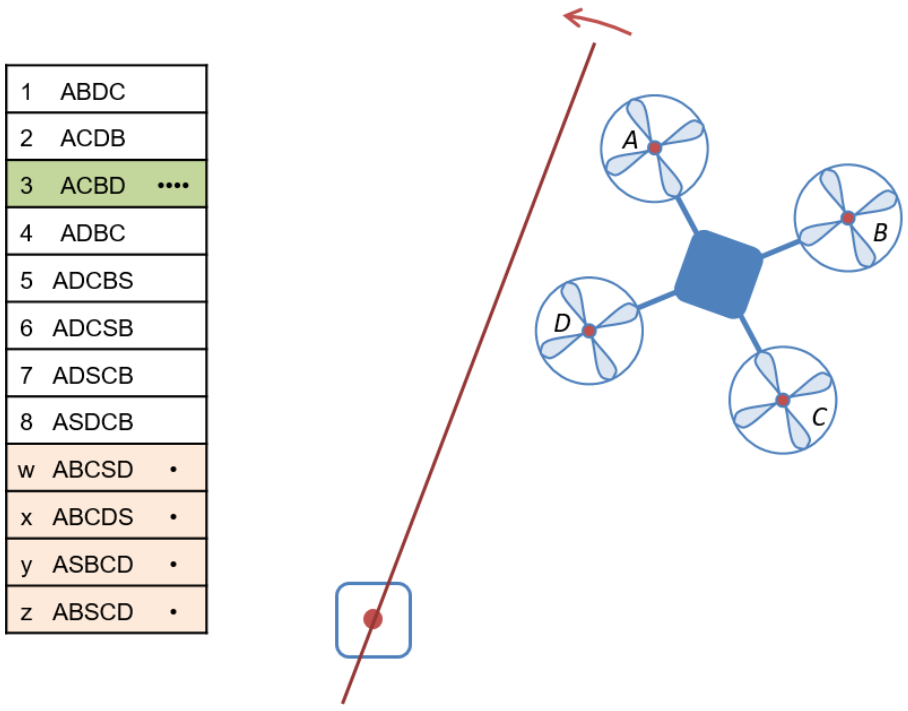


Рисунок 12 – Процесс обработки последовательности.

Бежевым на рисунках обозначены последовательности, имеющие потенциал для срабатывания, зеленым обозначена сработавшая последовательность. Она и выбирается для задания направления движения.

Описанный автомат распознавания реализован при моделировании автомата управления и является единственным источником кода для передачи автомату А3, при тестировании и получении результатов.

1.6. Эмулятор комплекса

Для тестирования и испытания решения в работе встал вопрос о создании эмулятора среды, который бы позволял:

а) удостовериться в работоспособности алгоритма;

б) проверить работу в условиях помех;

в) представить результаты работы в наглядном виде.

В качестве языка реализации был выбран язык Python 2.7. Выбор на него пал по нескольким причинам:

- понятный и простой синтаксис;
- высокая скорость разработки;
- наличие большого числа библиотек.

При построении эмулятора были учтены особенности тестируемого алгоритма, что позволило определить точность и глубину отражения действительности в эмуляторе.

Конечная реализация выглядит следующим образом:

а) эмуляция происходит пошагово, перемещение объекта дискретное;

б) число шагов для завершения заранее определено;

в) все действия и расчеты происходят в рамках одного шага;

г) вся история сохраняется;

д) результатом является график траекторий полета.

Для простоты реализации все воздействия выделили на уровне одного шага, что позволило сократить время разработки, но при этом обеспечивало достаточную точность.

На каждом этапе происходит получение обхода методом расчета углов до датчиков и выстраивание их по возрастанию угла. Далее происходит получение через алгоритм навигации вектора направления и смещение объекта на заданный вектор. При необходимости происходит добавление шума в виде:

- случайного смещения в рамках заданных границ;
- поворота на случайный угол.

Эмуляция происходит для каждого объекта в отдельности. Их число, как и начальные расположения могут конфигурироваться различными способами. Самым наглядным и при этом покрывающим все возможные точки старта является размещение n объектов на границе окружности радиусом R.

Результаты работы эмулятора, а также непосредственно результаты работы алгоритма навигации представлены на рисунках 13–15, на которых изображены траектории квадрокоптеров, начинающих свое движение из разных точек окружности. Точка посадки же находится в центре и является целью навигации.

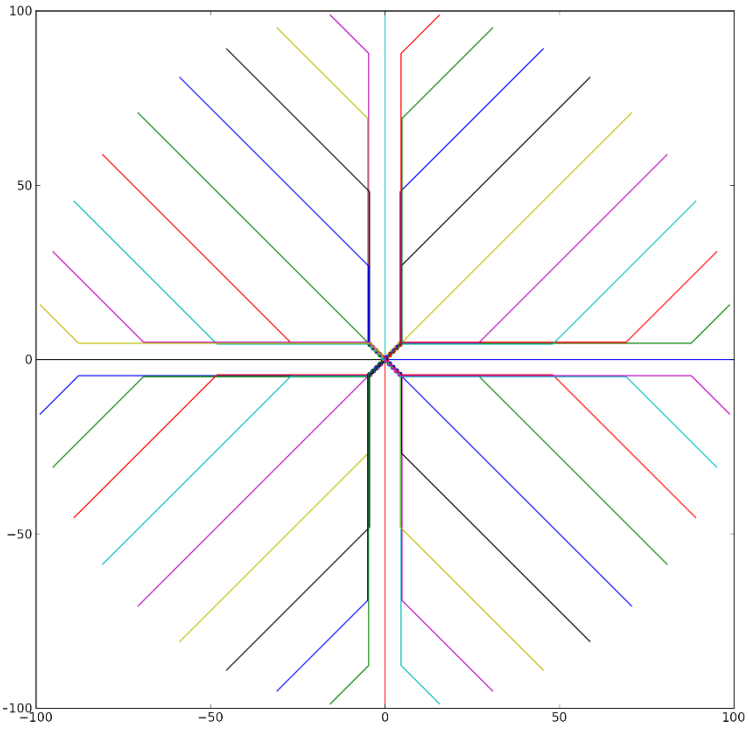


Рисунок 13 – Поведение алгоритма навигации в идеальных условиях.

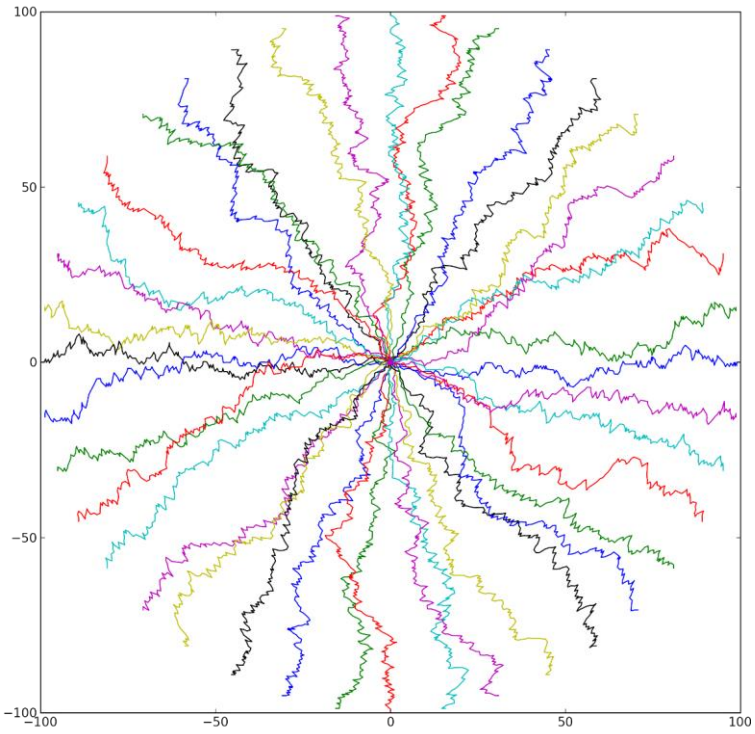


Рисунок 14 – Поведение алгоритма при случайном повороте.

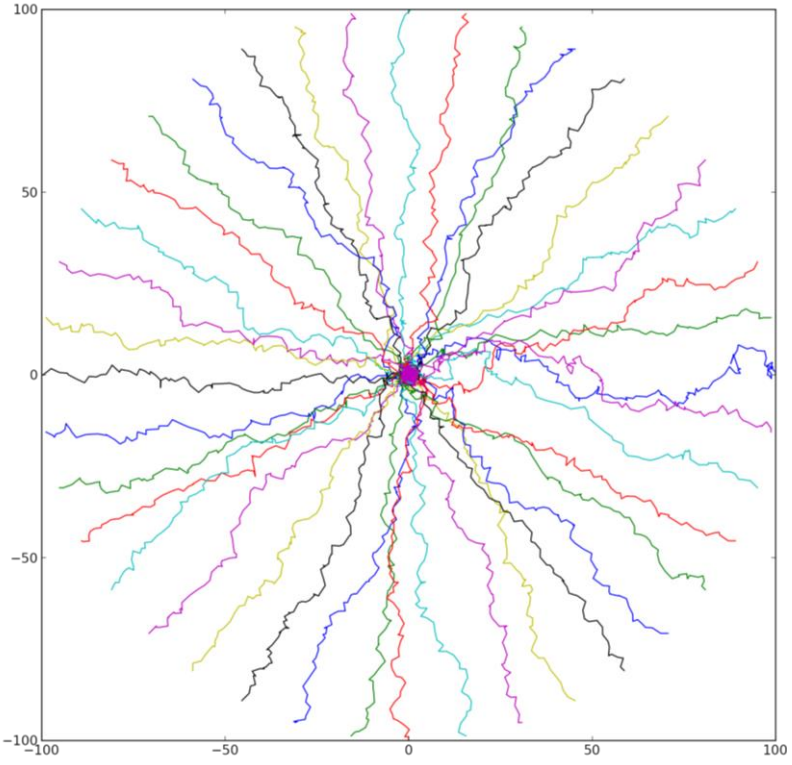


Рисунок 15 – Поведение алгоритма при случайном повороте и смещении.

В обычных условиях алгоритм ведет себя стабильно и сходится к точке посадки, что можно сказать и про случай случайного вращения.**Ошибка! Источник ссылки не найден.** Однако при случайном смещении объект при достижении площадки посадки начинает кидать, но это обусловлено лишь внешним воздействием и считается нормальным поведением

1.7. Достоинства и недостатки

Представленный алгоритм, несмотря на свою стабильность и устойчивость к помехам имеет несколько недостатков.

В первую очередь, он напрямую не выдает точного вектора направления до цели посадки, что затрудняет определение скорости движения и создает одинаковые колебания как на удалении, так и непосредственно над площадкой посадки.

Помимо этого, алгоритм имеет достаточно медленную сходимость к точке, учитывая, что его основное средство – это колебания между секторами, что достаточно медленно приближает его к цели.

Несмотря на перечисленные недостатки, приведенный алгоритм прост в реализации и не требователен к аппаратной платформе, что делает его крайне пригодным для использования в целевых условиях.

1.8. Выводы по главе 1

В данной главе рассмотрен исходный алгоритм. Также приведено описание его работы, моделирование, а также перечислены достоинства и недостатки.

Использование данного алгоритма не было испытано на реальной модели, а лишь была проведена эмуляция среды.

ГЛАВА 2. УСОВЕРШЕНСТВОВАННОЕ РЕШЕНИЕ

Учитывая перечисленные факты в конце предыдущей главы, встал вопрос о необходимости нового, улучшенного решения данной задачи. В данной главе описываются пути поиска этого решения и возникшие трудности, преодоление которых также описаны в этом разделе.

2.1. Постановка задачи

Исходя из принципа работы системы в общем случае мы имеем непрерывный поток из кортежей $\{S, t\}$, где S – символ, соответствующий датчику, t – время регистрации показаний с датчика.

Имеющееся решение использует только половину данных, а именно последовательность из символов S. Предполагая, что, взяв во внимание другую часть исходных данных, можно ускорить сходимость, было принято решение о разработке нового алгоритма, использующего весь набор данных.

Таким образом, задачей является разработка нового алгоритма, отличающегося от имеющегося более быстрой сходимостью, и, как следствие более точным вектором направления до площадки посадки для каждой точки пространства.

2.2. Построение автоматов

Используя опыт построения автоматов из предыдущей работы, была выдвинута гипотеза о том, что можно получить управляющий автомат, который бы получал последовательность кортежей вида $\{S, t\}$, а выдавал бы вектор (v_x, v_y) до площадки посадки.

В процессе исследования было испробовано несколько алгоритмов и методик по получению данных автоматов. Важно отметить, что все исследования проводились на новом эмуляторе, особенности которого описаны ниже в данной главе.

Основным исследованием стало изучение возможности построения такого автомата эволюционными методами.

В качестве исходных данных во время различных экспериментов брались:

- готовые траектории, полученные исходным решением, включающие в себя последовательности из символов с отметками времени;
- набор точек, взятый с различных окружностей, экспоненциально удаляющихся от точки посадки, включающих в себя обход и рекомендуемый вектор направления.

Был выбран обычный восходящий поиск, в котором варьировались как мутации, так и методики перемешивания поколений. В итоге каждая особь состояла из некой структуры-графа, над которой уже и проводились операции изменения.

Отдельную сложность представлял fitness-функция по ряду причин:

- необходимо назначать штраф за изменение траектории;
- готовое решение не должно «пролетать» площадку посадки;
- расстояние до площадки посадки не должно увеличиваться в определенных границах;
- автомат должен быть минимальной структуры.

В результате получалось решение, которое имеет очень сложную fitness-функцию, основанную на выборе направления по восьми равно распределенным векторам. Однако, «вырастить» готовое решение не получилось по причине того, что все равно эта функция оказалась неидеальной. Также поиск в поле решений стал больше быть похожем на угадывание, нежели на сведение к оптимальному решению ввиду слишком «плоской» функции.

Можно также утверждать, что, возможно, применив другие методы построения автоматов, удалось так получить бы требуемое решение. Поиск такого решения занимает большое количество времени, а также становится слишком сложной разработка целевой fitness-функции.

Кроме того, в случае нахождения решения, похожего на идеальное, требуется проверка его на сходимость и стабильность. Что представляется отдельной задачей, представляющей чисто академический интерес и не имеющей ничего общего с практикой.

В результате исследования по этому направлению решено было прекратить по причине нерентабельности, а также параллельно было найдено аналитическое решение, речь о котором пойдет ниже.

2.3. Аналитическое решение

Параллельно с попытками автоматического построения автомата управления производился поиск аналитического решения, поскольку казалось, что данных для получения точного вектора до площадки посадки достаточно. В итоге такое решение было найдено.

В целом задача сводится к нахождению второй точки пересечения двух окружностей. Для наглядности объяснения рассмотрим рисунок 16.

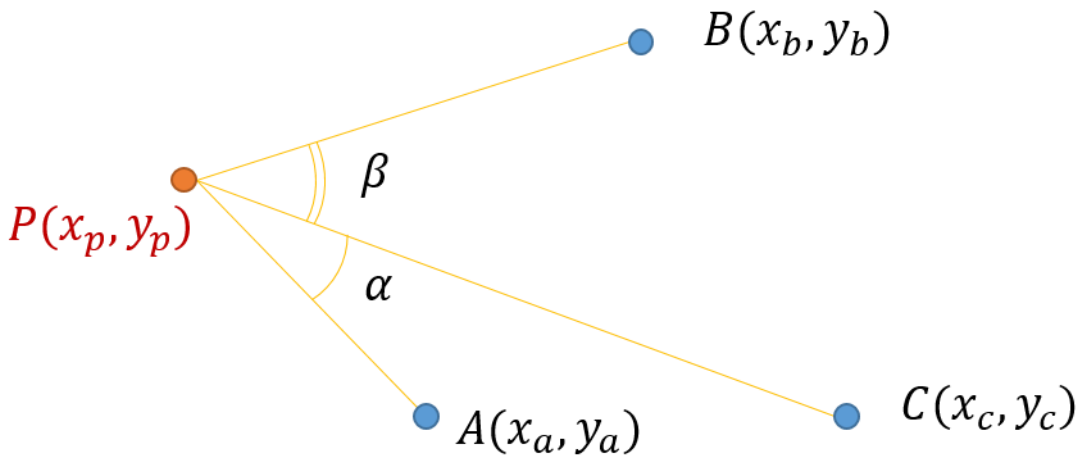


Рисунок 16 – Постановка задачи.

Итак, имеем в качестве исходных данных:

– Углы α и β. Они вычисляются из времен регистрации датчиков и их нахождение описано ниже.

– Координаты точек A, B, C . По сути это координаты установленных датчиков.

Основной целью решения же является нахождение координат точки P – (x_p, y_p).

Но все по порядку. Если координаты точек A, B, C заранее известны и жестко заданы, то углы α и β предстоит найти.

Нахождение угла α происходит с использованием разницы времен регистрации датчиков C и A.

$$t_\alpha = (t_C - t_A)\ mod\ (T/2)$$

$$\gamma = t_\alpha \frac{2\pi}{T}$$

Здесь T – период вращения плоскости, который известен заранее, а mod – оператор получения остатка от деления. Деление на половинный период обусловлено тем, что в конструкции системы посадки используется вращающаяся плоскость, а не луч.

При этом полученный угол γ – это угол между положениями прямой в моменты регистрации датчиков. В зависимости от момента времени мы можем получить несколько вариантов соотношения с углом α.

$$\gamma = \pi - \alpha \qquad\qquad \gamma = \alpha$$

Теперь рассмотрим рисунок 17, на котором указаны соотношения между вписанными и центральным углами. Заметим, что центр вписанной окружности находится в той же полуплоскости, что и острый вписанный угол.

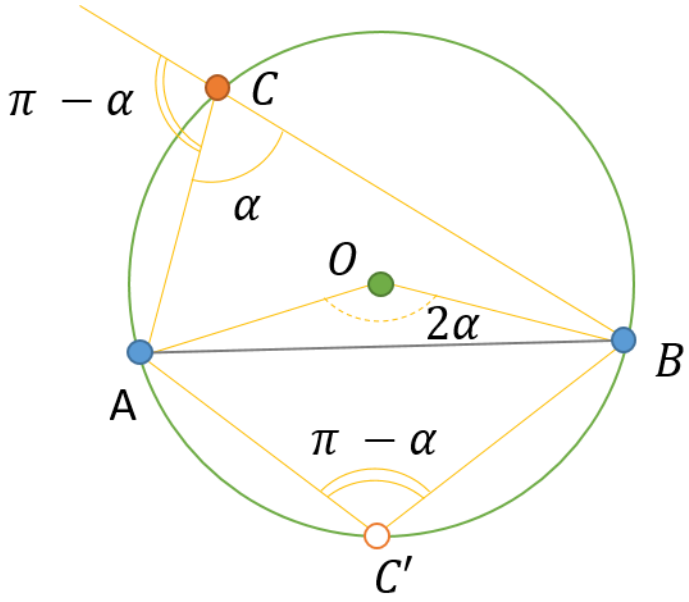


Рисунок 17 – Соотношения углов, опирающихся хорду окружности.

Опираясь на это свойство, а также зная направление вращения плоскости, а, следовательно, и порядок регистрации датчиков. Будем определять полуплоскость через знак выражения $(t_C - t_A)$.

Теперь обратим внимание на треугольник AOB. Так как он равнобедренный, нам известна сторона, противоположный угол и полуплоскость, то мы можем вычислить координаты точки O через координаты точек A и B. Это делается при помощи стандартных формул вычислительной геометрии. Код вычисления описанной окружности приведен в приложении.

Вернемся к начальной задаче. Рассмотрим рисунок 18.

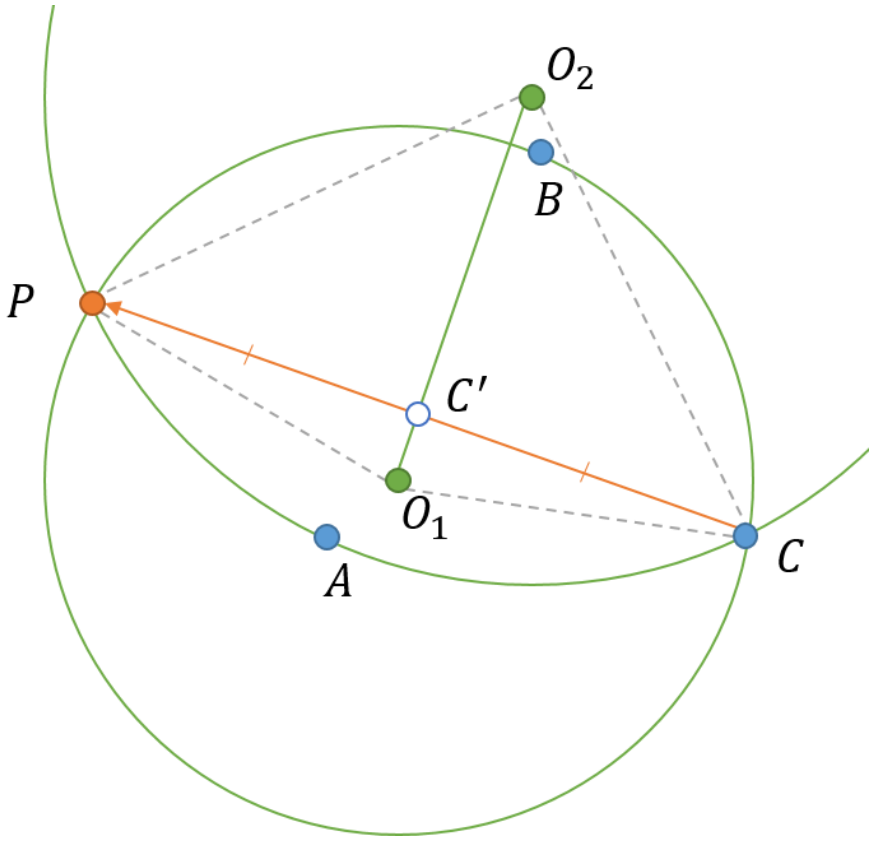


Рисунок 18 – Пересечение двух описанных окружностей.

Вычисление точки P происходит следующим образом:

а) Используя алгоритм, описанный выше, строим точки O_1 и O_2, являющиеся центрами описанных окружностей вокруг треугольников PBC и PCA соответственно.

б) Проецируем точку C на прямую, соединяющую точки O_1 и O_2. Важно отметить, что проекция C' не обязательно будет находится внутри отрезка O_1O_2, что не мешает дальнейшему вычислению.

в) Удваивая полученный вектор CC' и откладывая его от начальной точки C, получаем готовые координаты точки P.

Весь код алгоритма, описанный выше, также приведен в приложении.

Таким образом, получаем алгоритм, который в любой момент времени при наличии как минимум трех актуальных измерений. В качестве исходных данных при

этом используются только времена регистрации и заранее заданные координаты датчиков. Описанный алгоритм был реализован в программно-аппаратном комплексе, описанном в главе 3.

2.4. Улучшенный эмулятор комплекса

Учитывая то, что новое аналитическое решение использует разницу времен между регистрациями датчиков, то стало понятно, что старый эмулятор, показывающий обход датчиков из точки посадки не годится для работы с новым алгоритмом.

Новый симулятор должен был отвечать следующим требованиям:

- выдавать помимо символа сработавшего датчика еще и время регистрации;
- поддерживать непрерывное движение объекта;
- работать относительно быстро.

Для построения такого симулятора необходимо было решить достаточно нетривиальную математическую задачу.

Пусть у нас имеется вращающаяся прямая в точке $(0, 0)$. Также у нас имеется объект, представляющий собой набор точек, которые могут двигаться с какой-либо скоростью.

Требуется уметь рассчитывать для любого момента времени следующий момент пересечения прямой с любым из датчиков, при том, что датчики могут иметь константную скорость.

Это потребовало на каждом шаге численного решения двух нелинейных систем уравнений, что, конечно, сказалось на времени расчета каждого шага.

Теперь эмулятор выглядел следующим образом:

а) имеем начальный момент времени (0 при старте системы);

б) численно находим следующий момент регистрации датчика;

в) применяем изменения вектора движения, используя любой из алгоритмов;

г) переходим к шагу А.

Стоит отметить, что, конечно, в отличии от предыдущего эмулятора, новый не поддерживает симуляцию вращения, однако, это частично симулируется запуском набора объектов все так же окружности вокруг площадки посадки.

В противном случае пришлось бы строить систему уравнений с учетом этого самого вращения вокруг точки, являющейся центром квадрокоптера, что очень сильно бы усложнило расчет и, как следствие, усложнило систему.

В новом же эмуляторе есть ряд крайне важных достоинств. Во-первых, это наиболее точное отражение действительности. Теперь можно действительно применять изменения вектора движения после регистрации каждого датчика. Во-вторых, время в симуляторе теперь является непрерывной величиной и, помимо его непосредственного использования, появилась возможность ограничивать работу алгоритма для симуляции с его использованием.

Эта реализация симулятора использовалась как при проверке результатов аналитического решения, так и при построении автоматов алгоритма навигации.

2.5. Сравнение с исходным решением

Новый алгоритм навигации по сравнению с предыдущим имеет ряд неоспоримых преимуществ.

В первую очередь стоит отметить точность выдаваемого результата аналитическим решением по сравнению с автоматом вычисления вектора. Это позволяет более качественно определять как расстояние до цели, чего не было ранее возможно в принципе, так и непосредственно направление движения. Имея расстояние до цели можно рассчитать скорость сближения, что позволяет экономить время и, как следствие, запас аккумулятора.

Кроме этого, если раньше для получения решения по направлению требовалось как минимум четыре сработавших датчика, при этом обновление показаний обязательно было на каждом из них, то в новом алгоритме достаточно всего трех, и обновление показаний по лишь одному.

Также важно отметить поведение объекта навигации вблизи цели. Если раньше сходимость происходила за счет колебания из одного внутреннего сектора в другой, то сейчас колебания будут происходить вокруг оси вращения, и они будут менее выражены, учитывая, что сейчас имеется информации о расстоянии до цели.

Однако, аналитическое решение проигрывает в одном вопросе исходному решению – это зависимость от времени. Если раньше не требовалось точного измерения времени регистрации срабатывания датчиков, то сейчас оно критически важно и влияет на точность.

Таким образом, аналитическое решение, не смотря на зависимость от времени, сильно выигрывает у исходного решения за счет принципиально другого подхода.

2.6. Выводы по главе 2

В результате работы, описанной в данной главе, было показано, что построение управляющего автомата методами машинного обучения и эволюционными алгоритмами не всегда приводит к успеху.

Иногда этому мешает как особенность задачи, так и условия, наложенные извне на готовое решение. Зачастую такого вида задачи представляют чисто академический интерес и мало отражают потребность такого сложного построения решения для реальных задач.

Более того, в случае, если требуются гарантии на выполнение каких-либо условий, то необходимо также верифицировать полученный автомат, что представляет собой, как часто бывает, намного более сложную задачу и достойно отдельной работы.

В результате было получено аналитическое решение данной задачи, приводящее к абсолютно точному результату (в идеальных условиях). Решение было получено математически.

Полученное точное решение решает задачу, поставленную в данной главе и является опорным результатом для использования при реализации на практике.

ГЛАВА 3. РЕАЛИЗАЦИЯ ПРОГРАММНО-АППАРАТНОГО КОМПЛЕКСА

В данной главе описан процесс реализации усовершенствованного алгоритма навигации в частности, а также всей системы посадки в целом. Описаны трудности, возникшие при реализации комплекса, а также некоторые конструктивные и программные решения.

3.1. Устройство комплекса

Для реализации поставленных задач необходимо было собрать настоящий рабочий образец квадрокоптера. На сегодняшний день существует огромное число различных полетных компьютеров. Все они похожи между собой тем, что содержат большое число датчиков, имеют возможность подключения внешних, а также имеют открытый исходный код.

Для реализации этого проекта выбор пал на полетный компьютер Pixhawk, обладающий рядом достаточно серьезных преимуществ:

- 32-х битный процессор с FPU (ускорением работы с числами с плавающей точкой);
- наличие невероятно большого числа всевозможных типов подключений;
- поддержка нескольких проектов автопилота без изменения внутреннего устройства;
- открытый исходный код;
- большое количество различной документации.

В результате кропотливой сборки, настройки и отладки получился квадрокоптер, внешний вид которого представлен на рисунке 19.

Рисунок 19 – Общий вид собранного квадрокоптера.

Более подробный и детальный вид имеется в приложении. К отличительным особенностям данного аппарата можно отнести:

– наличие GPS-приемника;
– наличие ультразвукового сонара-альтиметра;
– наличие установленной собственной новой системы точной посадки.

Отдельного внимания заслуживают датчики. Каждый из них является достаточно сложной аналоговой схемой с выставлением уровня. Датчики отличаются особой чувствительностью и должны быть защищены от внешнего электромагнитного излучения. Готовые датчики в сборе и установленные и подключенные на квадрокоптер можно наблюдать на рисунке 20.

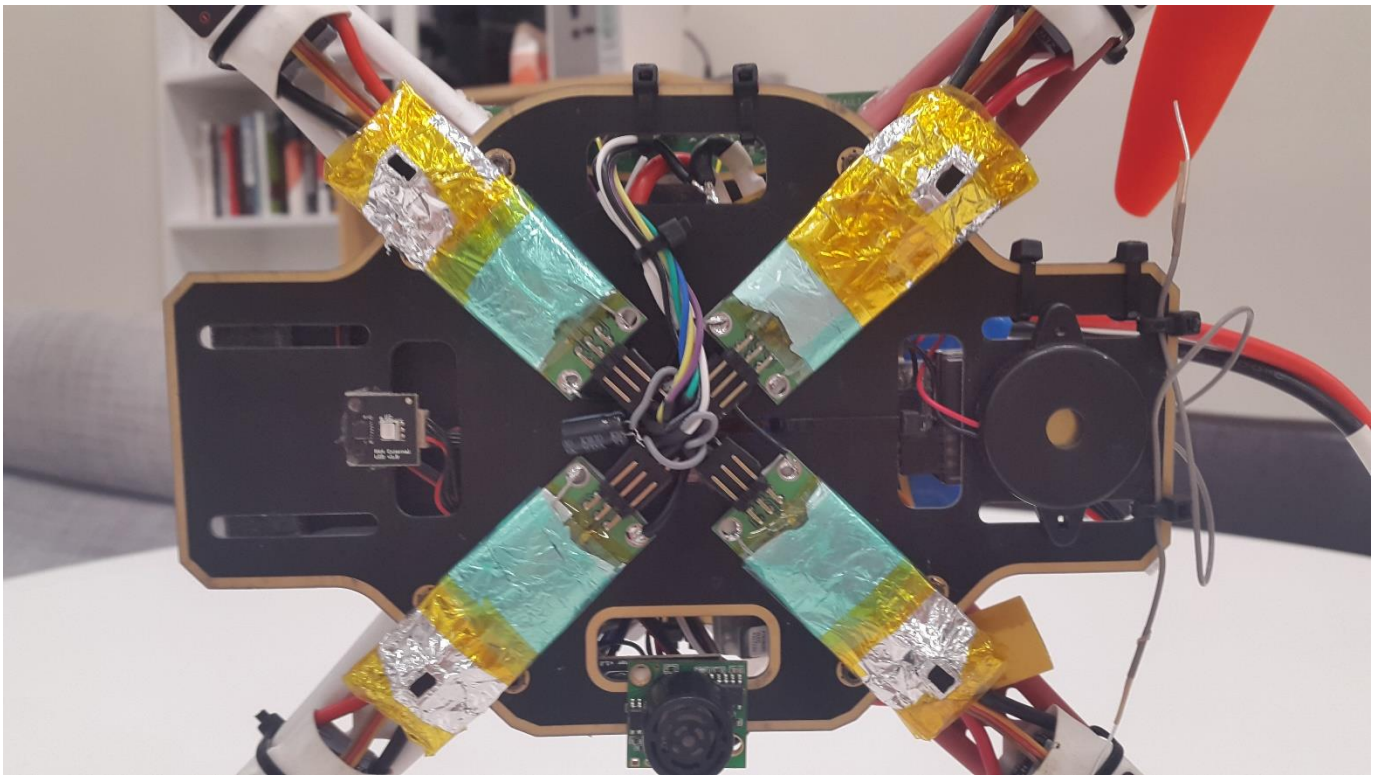


Рисунок 20 – Установленные датчики посадки.

Ответной частью к датчикам, исходя из конструкции системы посадки, является вращающаяся лазерная плоскость. Общий вид готовой конструкции изображен на рисунке 21.

Рисунок 21 – Готовая конструкция генератора вращающейся лазерной плоскости.

Она была сконструирована из инфракрасного лазера, имеющего уже в комплекте линзу, дающую плоскость, разобрана и затем собрана в новом корпусе. Корпус же сделан вручную: часть распечатана на 3D-принтере, часть – собрана из других материалов.

Сложность конструкции заключается в том, что помимо нужной частоты вращения, необходимо соблюсти соосность лазера и линзы. В противном случае система не будет давать конус луча на выходе, и ее область работы будет сильно ограничена.

Конструкция вращающейся оси была сделана таким образом, чтобы она могла работать от внешнего аккумулятора и подключаться через разъем USB.

Теперь что касается подключения системы посадки к главному бортовому компьютеру. Схема подключения представлена на рисунке 22.

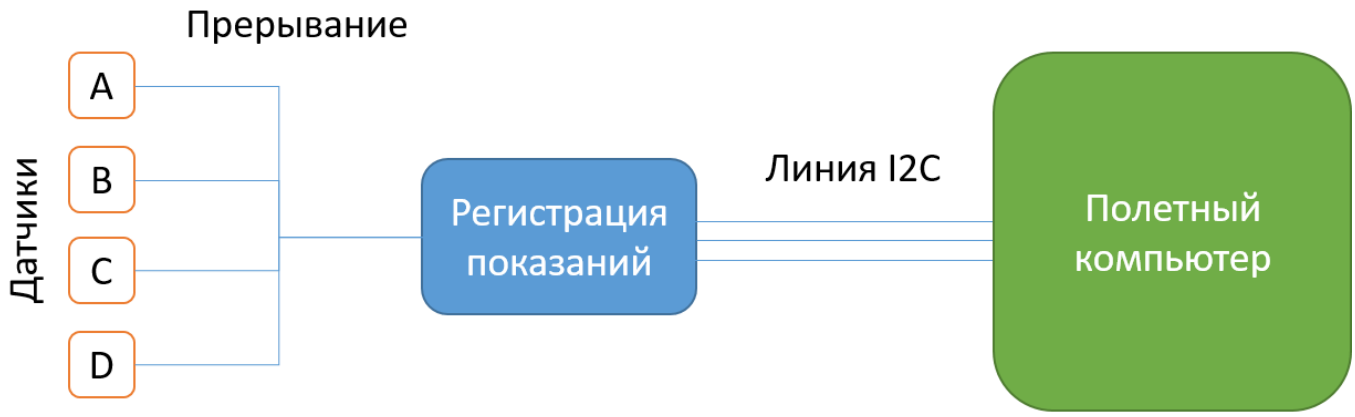


Рисунок 22 – Схема подключения системы посадки.

В итоге получившаяся система состоит из:

- самих датчиков;
- промежуточного модуля, регистрирующего время срабатывания сенсоров;
- главного модуля, периодически опрашивающего блок регистрации датчиков и производящего все вычисления.

Каждая из крупных подсистем будет описана далее.

3.2. Блок регистрации датчиков

Изначально система задумывалась таким образом, чтобы не приходилось разрабатывать сложный модуль вычисления вектора посадки в полетном компьютере. Однако, для ускорения расчета, все же пришлось реализовывать логику вычисления на главном компьютере, а остальное, за неимением внешних интерфейсов для прерываний, было отдано отдельному устройству – блоку регистрации датчиков.

Таким образом, для этого блока были выдвинуты следующие требования, а именно выполнение одновременно всех условий:

- наличие четырех векторов внешних прерываний для датчиков;
- наличие трех внутренних таймеров достаточного разрешения для фиксации времени регистрации срабатывания;

– наличие шины данных I2C(для подключения к главному модулю).

Отдельного абзаца достоин выбор платы для реализации задуманной системы. Оказалось, что наиболее популярная готовая платы для разработки Arduino не подходят под все требования одновременно, за исключением самой мощной версии. В наличии ее не оказалось, поэтому выбор пал на отладочную плату STM8L.

Была использована отладочная плата STM8L-DISCOVERTY на базе процессора STM8L152C6. Внешний вид смонтированной платы на квадрокоптере изображен на рисунке 23.

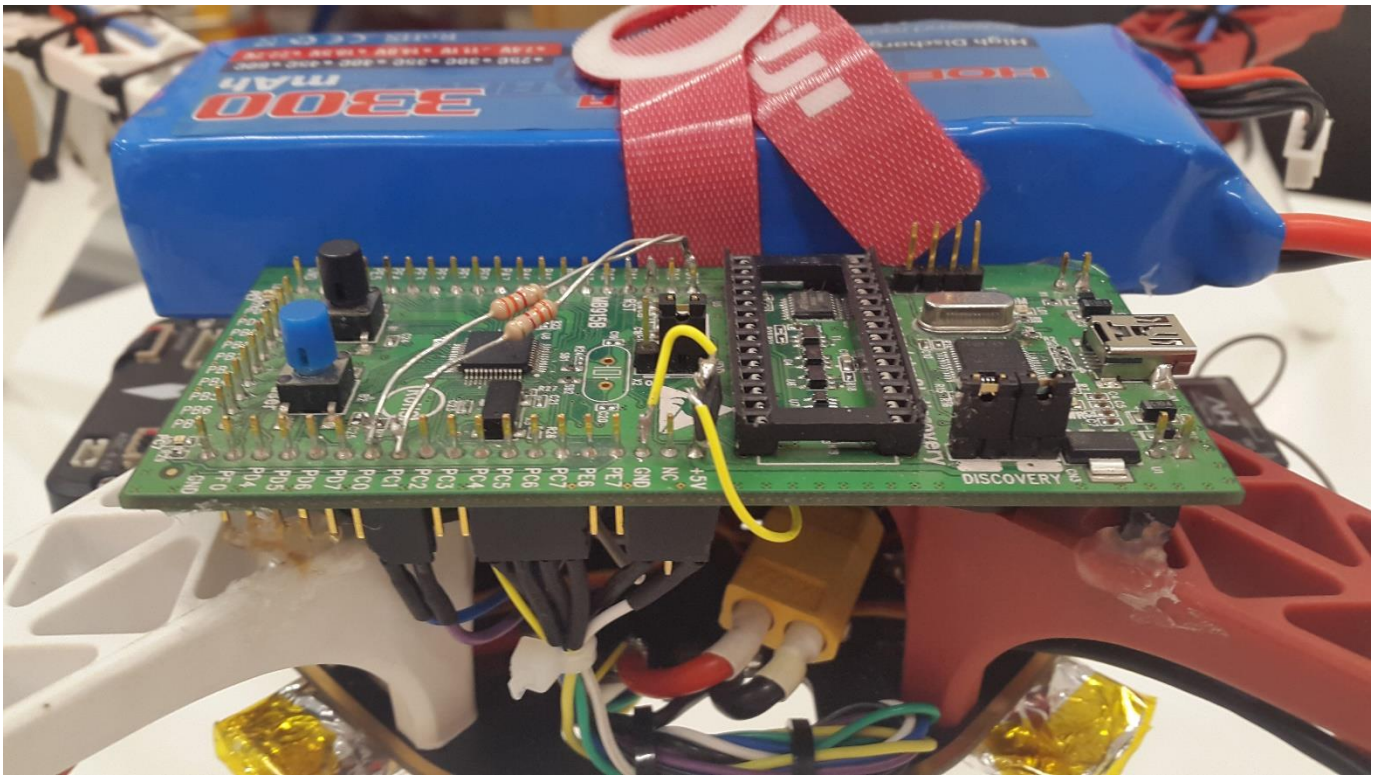


Рисунок 23 – Блок регистрации показаний.

Для реализации поставленной задачи была задействована следующая периферия процессора:

– внешние прерывания, по одному на каждый датчик;
– аппаратные таймера TIM1, TIM2, TIM3, для регистрации соответственно секунд, миллисекунд, микросекунд времени срабатывания датчика;

– модуль I2C для общения с главным полетным компьютером.

Протокол общения при этом выглядит достаточно просто. На каждый запрос данных устройство возвращает пять чисел типа uint32_t (без знака 32х-битное целое):

а) четыре времени регистрации по каждому датчику.

б) разница между предыдущим и текущим показаниями для последнего датчика (по сути это и есть $T/2$ из раздела Аналитическое решение2.3).

В итоге один пакет данных занимает 20 байт.

3.3. Главный модуль управления

Главным модулем управления является полетный компьютер Pixhawk. К нему напрямую подключается вся остальная периферия, в том числе и блок регистрации показаний через общую шину I2C. Общий вид установленного полетного компьютера на квадрокоптере представлен на рисунке 24.

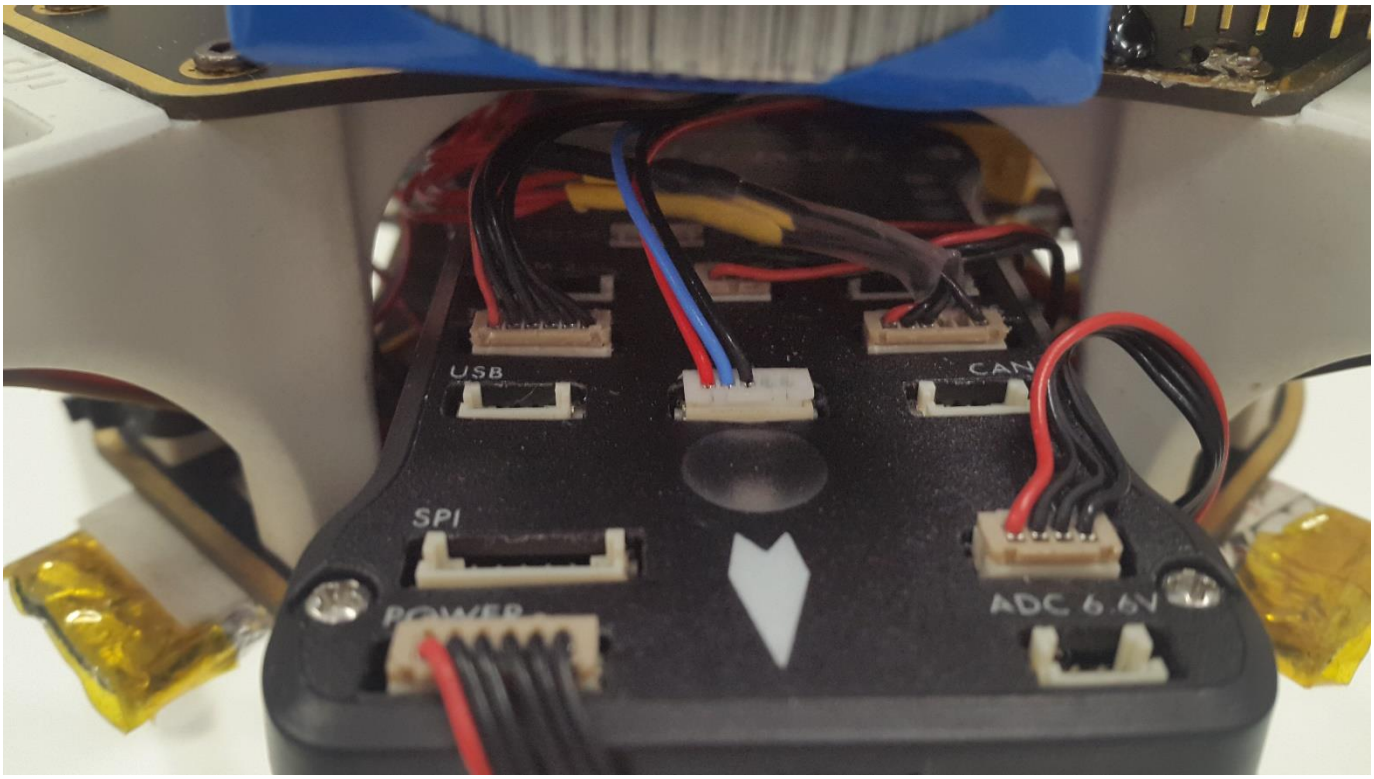


Рисунок 24 – Главный полетный компьютер.

Вся основная логика работы реализована в виде модуля к программе управления посадкой. Компьютер периодически опрашивает блок регистрации датчиков и пересчитывает рекомендуемый вектор направления.

Расчет производится полностью по новому алгоритму. Заметим, что для работы алгоритма достаточно всего трех датчиков, однако в нашем распоряжении имеется целых четыре. Это используется для получения более точного вектора направления путем усреднения результатов по всем сочетаниям из трех датчиков.

Также расчет результирующего вектора до оси вращения лазерной плоскости производится с учетом того, что точность прихода показаний с блока регистрации может «плавать». Для этого были реализованы достаточно простые, но эффективные фильтары.

В ходе отладки выяснилось, что необходимо знать смещение относительно координатной оси NE (север-восток) в сантиметрах, что и было реализовано.

Теперь при наличии лазерной плоскости в «области видимости» и включенном режиме «Land» квадрокоптер смещается к точке посадки.

Результаты работы алгоритма, показания телеметрии, а также скриншоты с видео посадки находятся в приложении.

3.4. Тестирование, настройка и отладка

Тестирование и отладка такой системы сильно отличается от обычных программ и представляет достаточно сложную задачу. Во-первых, был установлен дополнительный модуль Bluetooth, что обеспечило возможность беспроводного снятия показаний с датчиков и отладку через сообщения по протоколу Mavlink.

Во-вторых, все показания во время испытаний записывались на карту памяти, что давало возможность анализа после работы системы в спокойной обстановке.

Для отладки переменных был доработан способ приема-отправки сообщений, что обеспечило возможность отладки реализации самого алгоритма. В ходе отладки

выяснилось, например, что для работы с числами с плавающей точкой требуются отдельные функции, а порядок байт различается с блоком регистрации показаний.

Тестирование проводилось на начальном этапе без винтов и полета в принципе. Проверка работоспособности достигалась путем ручного смещения относительно площадки посадки.

Также для удобства тестирования был модифицирован режим полета «Brake», так, чтобы в этом режиме квадрокоптер центрировался по оси.

Во время тестирования и отладки выяснились некоторые тонкости, связанные с тестированием подобного рода устройств, а именно:

- в любой ситуации нужно быть готовым к переключению управления в ручной режим;
- необходимо озаботиться функцией экстренного отключения двигателей, либо питания.

Оказалось, что мощности установленного лазера недостаточно для работы системы на больших дистанциях.

Итоговая точность системы посадки составила менее пяти сантиметров (< 5 см).

3.5. Возможные улучшения

Для улучшения работы системы посадки, в первую очередь следует увеличить мощность лазера, создающего лазерную плоскость. Это позволило бы регистрировать показания датчиков на больших дистанциях.

Можно заменить контроллер блока регистрации на контроллер, имеющий большую частоту работы ядра и разрядности таймеров. Это бы позволило увеличить точность регистрации датчиков.

Также в дальнейшем было бы правильно усовершенствовать датчики, улучшив их «отзывчивость» на регистрацию лазера.

3.6. Выводы по главе 3

В данной главе был описан процесс построения программно-аппаратного комплекса для апробации системы посадки. По результатам тестирования системы можно понять, что модель расчета верна, дальнейшее достижение точности достигается путем улучшения аппаратной составляющей.

ЗАКЛЮЧЕНИЕ

Данная работа является логическим продолжением бакалаврской работы автора.

В работе описано исходное решение, его достоинства и недостатки, которые были устранены в ходе построения улучшенного решения.

Установлено, что автоматическое построение автомата управления для текущей задачи не является целесообразным, поскольку было разработано аналитическое решение.

В целом создана работоспособная модель квадрокоптера с установленной разработанной системой посадки.

По результатам испытания показана работоспособность представленного аналитического решения, в приложении приведены показания телеметрии и скриншоты видеозаписи полета.

К достоинствам получившийся системы можно отнести простую конструкцию, отсутствие требований в большим вычислительным мощностям и, как следствие, малый вес оборудования.

Использование представленного алгоритма посадки позволит осуществить бесперебойный цикл БПЛА, замкнув цикл недостающим элементом в виде зарядки аккумулятора. Это обусловлено возможностью теперь производить посадку точно на контактные площадки зарядного устройства, при реализации такового.

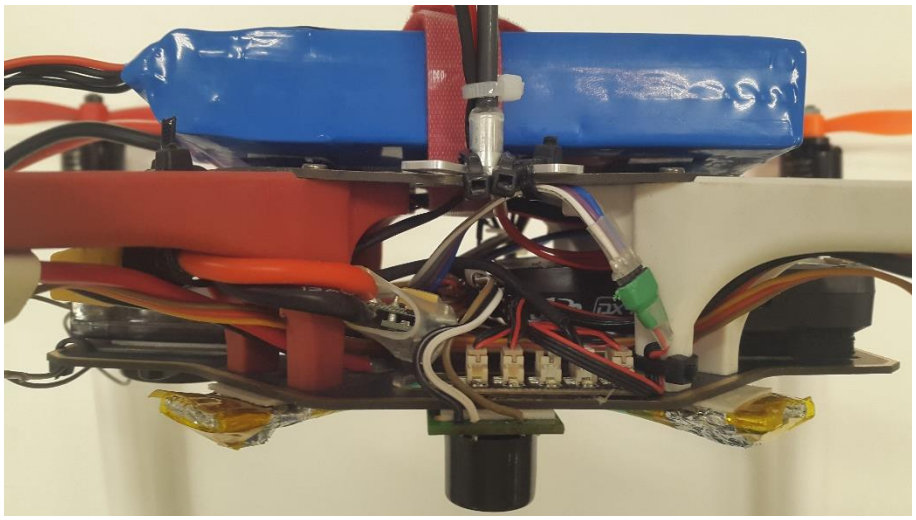
ПРИЛОЖЕНИЯ

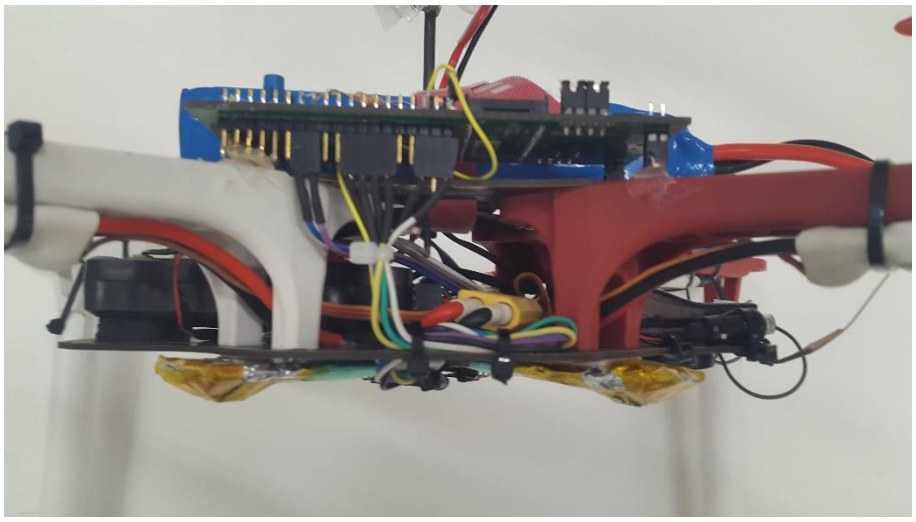
Приложение 1 – Листинг кода расчета вектора направления.

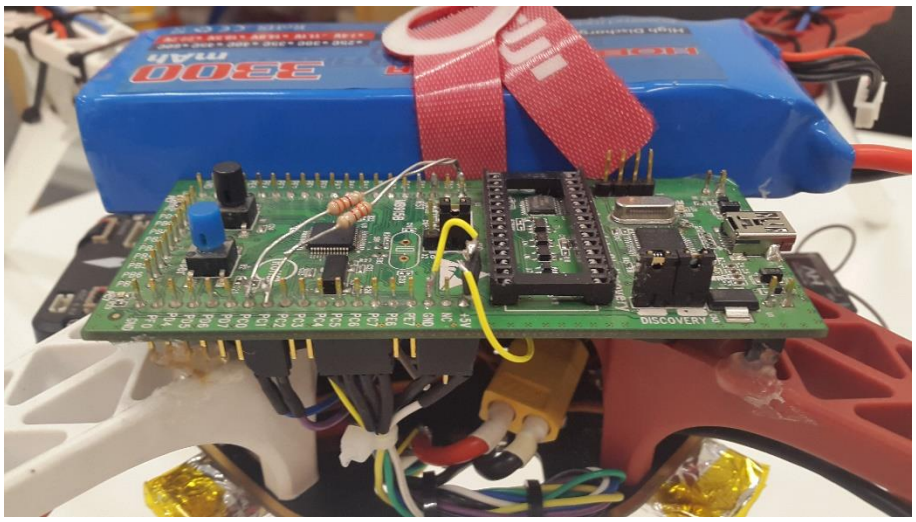
```
float AC_PrecLand_IRFlat::_angle(int32_t t1, int32_t t2)
{
    float t;
    t = fmod(float(t2 - t1), _half_period);
    t *= M_PI / _half_period;

    if (abs(t) >= M_PI_2) {
        if (t > 0) {
            t -= M_PI;
        } else {
            t += M_PI;
        }
    }

    return t;
}

Vector2f AC_PrecLand_IRFlat::_center(const Vector2f& a, const Vector2f& b, const
float angle)
{
    Vector2f ab_v = b - a;
    float r = ab_v.length() / (2 * sinf(absf(angle)));


    Vector2f h_p = (a + b) / 2;
    float ho = sqrtf(r * r - (ab_v.length() / 2) * (ab_v.length() / 2));

    float rot = copysign(1, angle);
    Vector2f ho_v = Vector2f(-ab_v.y * rot, ab_v.x * rot).normalized() * ho;

    Vector2f result = ho_v + h_p;
    return result;
}

Vector2f AC_PrecLand_IRFlat::_intersect(const Vector2f& a, const Vector2f& b, const
Vector2f& c, const float alpha, const float beta)
{
    Vector2f o1_p = _center(a, c, alpha);
    Vector2f o2_p = _center(b, c, beta);

    Vector2f o1o2_v = (o2_p - o1_p).normalized();
    Vector2f o1c_v = (c - o1_p);

    Vector2f vp_v = o1o2_v * (o1c_v * o1o2_v);
    Vector2f cp_p = vp_v + o1_p;
```

```
    Vector2f result = ((cp_p - c) * 2) + c;
    return result;
}

void AC_PrecLand_IRFlat::_calculate() {
    Vector2f res = Vector2f();

    if (_sensor_ok) {

        const uint8_t triples[6] = {0, 1, 2, 3, 0, 1};

        uint8_t a, b, c;
        for (uint8_t i = 0; i < 4; i++) {
            a = triples[i];
            b = triples[i + 1];
            c = triples[i + 2];

            float alpha = _angle(_sensor_times[a], _sensor_times[c]);
            float beta = _angle(_sensor_times[b], _sensor_times[c]);

            Vector2f tmp = _intersect(
                _sensor_positions[a],
                _sensor_positions[b],
                _sensor_positions[c],
                alpha,
                beta
            );

            _result_splitted[i] = _result_splitted[i] * (1.0f -
PRECLAND_IRFLAT_FILTER) + tmp * PRECLAND_IRFLAT_FILTER ;

            float summ = absf(alpha) + absf(beta);
            res += _result_splitted[i];
        }

        res /= 4.0f;
        res *= 5.0f;

        _last_calced_time_ms  = AP_HAL::millis();
    }

    _result = _result * (1.0f - PRECLAND_IRFLAT_FILTER) + res *
PRECLAND_IRFLAT_FILTER;
}
```

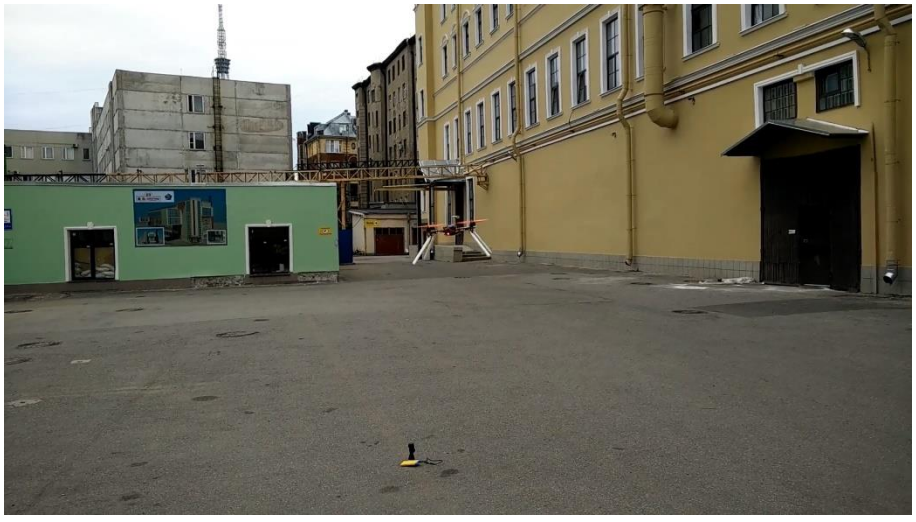
Приложение 2 – Фотографии устройства программно-аппаратного комплекса.






3800
mAh
DISCOVERY

Приложение 3 – Испытания.
Скриншоты взяты из видео, доступного по ссылке: *https://youtu.be/2-haxLOuRCY*.



СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Meerman I., Mulder T., Peters J. Autonomous Landing of a Quadcopter on a Predefined Marker, http://staff.science.uva.nl/~arnoud/activities/IMAV/QuadcopterLanding.pdf.

2 Ginkel R., Meerman I, Mulder T, Peters J. Autonomous Landing of a Quadcopter on a Predefined Marker. http://staff.science.uva.nl/~arnoud/activities/IMAV/QuadcopterLanding.pdf.

3 Infrared Tracking System for Drones and Robot automation. http://irlock.com/

4 Wenzel K., Masselli A, Zell A. Automatic Take Off, Tracking and Landing of a Miniature UAV on a Moving Carrier Vehicle, http://www.cogsys.cs.uni-tuebingen.de/publikationen/2010/Wenzel2010uav.pdf.

5 High-Precision Helicopter Landing System, http://www.gnss-technology.com/projects_helicopter.htm..

6 Samek M., Practical Statecharts in C/C++: Quantum Programming for Embedded Systems. NY, CMP Books, 2002.

7 Quantum Leaps, LLC. About QM. http://www.state-machine.com/qm/.

8 Quantum Leaps, LLC. About QP/C++. http://www.state-machine.com/qpcpp/.

9 Basic Workflow for Building a Stateflow Chart. http://www.mathworks.com/help/toolbox/stateflow/gs/bqnmvk8.html.

10 Шалыто А. А. Поликарпова Н. И. Автоматное программирование, СПб.: Питер, 2010, с. 176.

11 Шалыто А. А. Switch-технология. Алгоритмизация и программирование задач логического управления., СПб., Наука, 1998, с. 628.

12 PX4 Development Guide. http://dev.px4.io/

13 ArduPilot Development Site. http://ardupilot.org/dev/index.html