Санкт-Петербургский государственный университет

Макеев Владислав Дмитриевич

Выпускная квалификационная работа

Веб-приложение для удаленного контроля БПЛА

Уровень образования: бакалавриат

Направление *02.03.03 «Математическое обеспечение и администрирование информационных систем»*

Основная образовательная программа *СВ.5006.2017 «Математическое обеспечение и администрирование информационных систем»*

Профиль *Системное программмирование*

Научный руководитель:
доцент кафедры СП, к.т.н. Ю. В. Литвинов

Консультант:
ген. директор ООО “СКЗ” А. А. Пименов

Рецензент:
главный специалист ООО “СКЗ” Д. Н. Степанов

Санкт-Петербург
2021

Saint Petersburg State University

Makeev Vladislav Dmitrievich

Bachelor's Thesis

Web application for remote control of unmanned aerial vehicle

Education level: bachelor

Speciality *02.03.03 "Software and Administration of Information Systems"*

Programme *CB.5006.2017 "Software and Administration of Information Systems"*

Profile: *Software Engineering*

Scientific supervisor:
Docent of Software Engineering, C.Sc. Y. V. Litvinov

Consultant:
CEO of Computer Vision Systems LLC A. A. Pimenov

Reviewer:
Senior Specialist at Computer Vision Systems LLC D. N. Stepanov

Saint Petersburg
2021

Оглавление

Введение

Связанные с беспилотными летательными аппаратами исследования с каждым днём приобретают всё большую актуальность. Крупные компании вовсю экспериментируют с гражданскими применениями дронов, используя их в широком спектре индустрий, и количество исследований и разработок в этой области лишь увеличивается. Беспилотные летательные аппараты уже проникли во множество сфер деятельности — от фотожурналистики до курьерской доставки — и в каждой из них уже успели показать высокую степень своей пригодности для решения поставленных задач.

Помимо широко известных проблем — таких, как необходимость улучшения аэродинамических свойств, полётных алгоритмов и понижения себестоимости поступающих на рынок моделей — существует ряд других, более мелких и труднопредсказуемых на этапе проектирования задач. В сфере разработки программного обеспечения к одной из таких задач относится упрощение процесса взаимодействия между беспилотным летательным аппаратом и оператором.

Рынок заполнен различными приложениями, позволяющими контролировать беспилотный летательный аппарат и получать с него данные, будь то показания барометра или изображение с установленной на него камеры, однако подавляющее большинство таких программных продуктов требуют установки на устройство пользователя, что несёт с собой целый ряд проблем. Основными из них являются платформозависимость, занимаемое приложением место и отсутствие единообразия интерфейса из-за изначальной разработки под конкретное устройство. Ещё одной проблемой является сравнительно небольшой процент приложений для мобильных устройств, что также является довольно большой преградой на пути упрощения взаимодействия программного обеспечения беспилотного летательного аппарата и его оператора.

Использование тонкого клиента является логичным шагом в решении проблем взаимодействия пользователя с беспилотным летательным аппаратом. Выступая в качестве платформы для развертывания веб-

сервера, беспилотный летательный аппарат может позволять оператору подключаться к находящемуся на нём программному обеспечению, используя в качестве клиента обычный веб-браузер, уже предустановленный на многих устройствах, которые пользователь может захотеть использовать для управления беспилотным летательным аппаратом.

На математико-механическом факультете СПбГУ разрабатывается библиотека компьютерного зрения CoreCVS[1]. Именно в неё и планируется интеграция веб-приложения для упрощения взаимодействия беспилотного летательного аппарата с пользователем. В качестве же аппаратных средств должны выступать платформы, часто используемые на беспилотных летательных аппаратах, такие как Raspberry Pi, Orange Pi или NVidia Jetson Nano. Являясь платформами для Embedded Linux, они предполагают определённые ограничения, накладываемые на разрабатываемый программный продукт. Такие ограничения включают в себя ограничение на объём занимаемой памяти, ограничение на объём используемой оперативной памяти и некоторые другие.

Кроме аппаратной и программной платформы, необходимо также обсудить протокол общения между оператором и беспилотным летательным аппаратом. MavLink (Micro Air Vehicle Link) является наиболее популярным протоколом, получившим широкое распространение из-за своей бинарной сериализации. Это свойство позволяет обеспечить небольшой размер сообщений и надёжную передачу через различные средства беспроводной коммуникации.

Целью данной работы является разработка веб-приложения для Embedded Linux, интегрированного с библиотекой CoreCVS, которое бы предоставляло пользователям функциональность наземной станции управления через веб-интерфейс.

[1]https://github.com/PimenovAlexander/corecvs — Github репозиторий CoreCVS (дата обращения: 18.12.2020).

1. Постановка задачи

Целью данной работы является разработка интегрированного с CoreCVS веб-приложения для Embedded Linux, способного обеспечить удалённый контроль над беспилотным летательным аппаратом.

Для достижения цели были поставлены следующие задачи:

1. Изучение аналогов для выявления функциональных требований.
2. Изучение протокола MavLink.
3. Анализ веб-фреймворков и выбор наиболее подходящего.
4. Разработка прототипа веб-приложения.
5. Проведение апробации получившегося приложения.

2. Обзор

В данном разделе рассматриваются различные определения, предлагаемые для наземных станций управления, а также определяется их общая функциональность. Затем проводится знакомство с протоколом Mavlink и анализируется его использование поверх протокола HTTP. После этого описывается понятие каркаса веб-приложения, приводятся наиболее популярные решения и выбирается наиболее подходящее.

2.1. Наземные станции управления

Наземные станции управления являются одним из ключевых компонентов систем беспилотных летательных аппаратов. Согласно [16], НСУ (наземная станция управления) представляет из себя многокомпонентную систему, которая может включать в себя:

- станцию оператора (человеко-машинный интерфейс);
- станцию управления дополнительным оборудованием;
- станцию управления миссией;
- станцию обработки изображений;
- станцию управления радаром;
- станцию отображения данных.

В случае программной НСУ эти станции следует трактовать как отдельные модули приложения, отвечающие за различные части функциональности ПО.

В общем случае НСУ позволяют [22]:

1. Производить планирование миссии — получив информацию об окружении БПЛА и деталях миссии, НСУ производит соответствующие вычисления для определения траектории курса и временных рамок маневров.

2. Обеспечить навигацию и позиционирование — НСУ реагирует на изменения в окружении в случае появления непредсказуемых ситуаций во время полета БПЛА по запланированной траектории.

3. Наладить коммуникацию и сбор данных — во время проведения миссии данные с БПЛА передаются на НСУ для их анализа и предоставления сводок оператору.

Стандарт STANGAG 4586 [14] определяет следующие требования к программному обеспечению НСУ:

1. Загрузка плана миссии.

2. Получение данных полёта.

3. Передача команд управления полётному контроллеру.

В CoreCVS уже существует кодовая база, позволяющая получать данные с БПЛА через веб-интерфейс. Таким образом, для разработки ПО, отвечающего минимальным требованиям к НСУ, требуется написать лишь ту функциональность, которая отвечает за передачу команд управления и загрузку плана миссии.

Согласно [21], структура взаимодействия типичного программного обеспечения НСУ выглядит так, как указано на рис. 1.

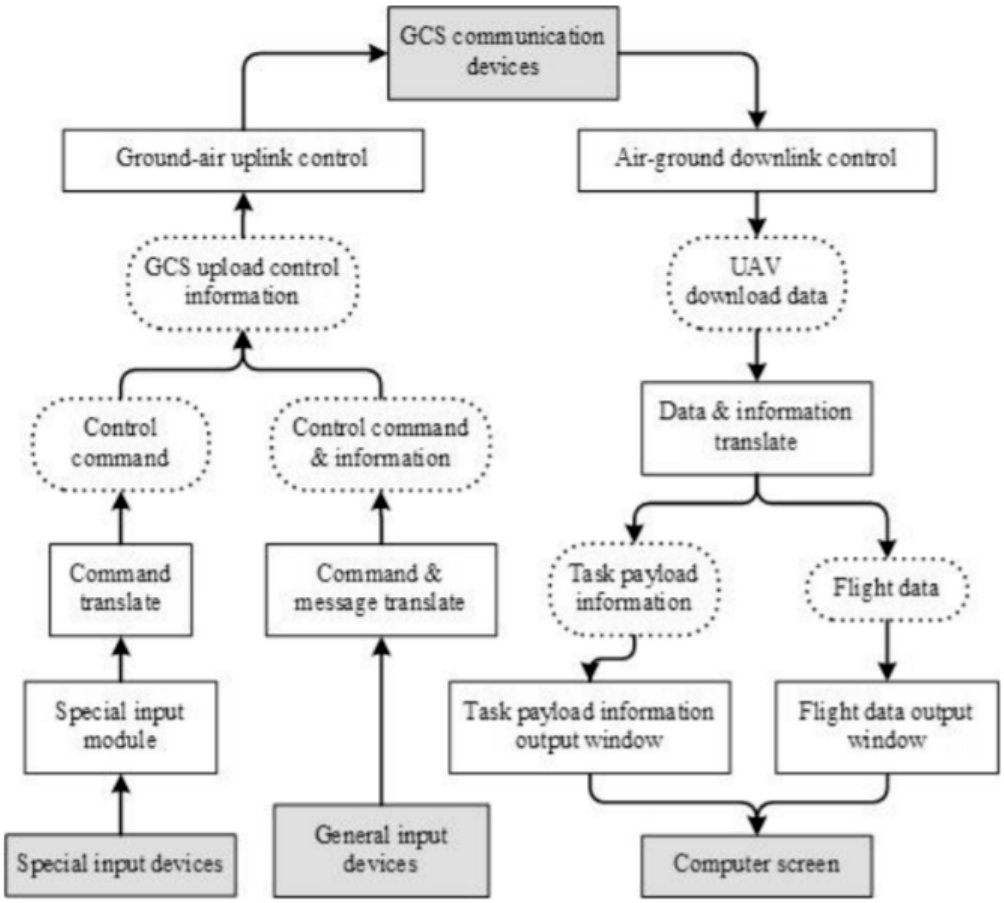

Рис. 1: Структура взаимодействия ПО НСУ [21]

Так как рассматриваемый в рамках этой работы проект предполагает унифицированный ввод данных, ввод со специальных устройств не входит в предполагаемую структуру ПО. В совокупности с отсутствием дополнительного оборудования это позволяет определить следующую структуру взаимодействия ПО НСУ (рис. 2).

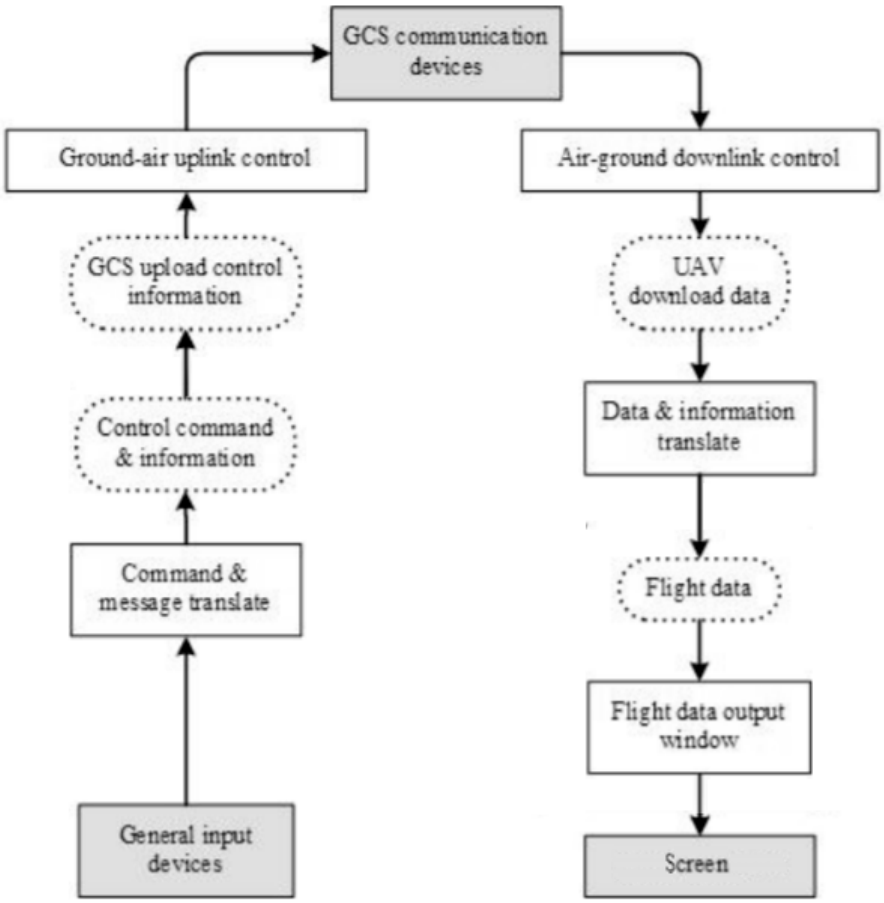

Рис. 2: Структура взаимодействия ПО НСУ для CoreCVS

2.2. Планирование миссии

Согласно [9], миссией БПЛА является набор из не менее чем двух точек в трёхмерном пространстве, где первая точка является стартом миссии (точкой взлёта), последняя — завершением миссии (точкой посадки), а все остальные — промежуточными точками, через которые БПЛА обязан пройти при прохождении миссии. Исходя из этого определения, для задания миссии — или же параметров миссии — необходимо задать цепочку точек в трёхмерном пространстве.

Многие популярные программные НСУ — такие как “Mission Planner”[2], “APM Planner 2.0”[3] и “QGroundControl”[4] — позволяют производить задание точек с помощью указания их на 2D карте, а также

[2]https://firmware.ardupilot.org/Tools/MissionPlanner — ссылка на Mission Planner (дата обращения: 16.12.2020).

[3]https://ardupilot.org/planner2 — ссылка на APM Planner 2 (дата обращения: 16.12.2020).

[4]http://qgroundcontrol.com/ — ссылка на QGroundControl (дата обращения: 16.12.2020).

позволяют добавлять или модифицировать точки в листе контрольных точек миссии. Именно такой подход из-за его популярности предполагается использовать в данной работе.

2.3. Протокол MavLink

MavLink является основным протоколом общения между НСУ и БПЛА. Он позволяет осуществлять коммуникацию через различные протоколы транспортного и физического уровня, такие как UDP и TCP, WiFi и Ethernet. MavLink также является протоколом с бинарной сериализацией, что означает предварительное превращение данных в последовательность байтов с последующим преобразованием их обратно в данные на стороне получателя. Это качество делает средний размер сообщений данного протокола очень низким [1] за счёт большого сжатия исходных данных. Каждое сообщение в MavLink v2 состоит из заголовка и тела сообщения, обладая весом от 25 до 270 байт.

Сами сообщения подразделяются на два основных типа [15]:

1. Сообщения состояния — посылаются от БПЛА к НСУ и содержат информацию о позиции БПЛА, данные с датчиков и дополнительную информацию, необходимую НСУ.

2. Сообщения управления — посылаются от НСУ к БПЛА и содержат запросы на проведение некоторых действий.

Попытки использования web-proxy для передачи MavLink сообщений через HTTP уже предпринимались. В [2] описывается подобный опыт на примере пакета для NodeJS под названием MavNode[5]. Тип сообщений “application/octet-stream” позволяет протоколу передавать бинарные данные без дальнейшей спецификации тела сообщения.

Возможной проблемой при использовании HTTP для трансляции MavLink сообщений может стать дополнительный размер, на который увеличивается сообщение из-за необходимости передачи HTTP-

[5]https://github.com/Jack-Rogers/MavNode — Github репозиторий MavNode (дата обращения: 18.12.2020)

заголовков. Так как при переходе на HTTP2 размер заголовка изменился незначительно [6] можно воспользоваться исследованием [5] для определения среднего размера заголовочной части сообщения, который в зависимости от запроса составляет от 200 байт до 2 килобайт. Таким образом полезная нагрузка HTTP-запроса с MavLink сообщением составляет от 1.2 % до 57 % со средним значением в 11.7 %.

Уточнение среднего размера интересующих заголовков можно получить при анализе данных из исследования [11]. Согласно этой работе, при отслеживании мобильного трафика запросы типа “application/octet-stream” составили около 1 % от общего числа запросов. При отправке 30 000 пакетов общий трафик заголовочных файлов пакетов “application/octet-stream” составил 101 811 байт для исходящих и 175 870 байт для входящих сообщений, что при учёте частоты запросов такого типа позволяет определить средний размер заголовочных файлов в 925 байт.

Таким образом, предполагаемая полезная нагрузка MavLink пакетов при передаче их по HTTP варьируется от 2.5 % до 22.5 % со средним значением в 13.7 %. Для сравнения возьмём такое популярное решение, как APM Plane. Там поток телеметрии ограничен значениями в 1450, 1790 или 1920 байт в секунду в зависимости от версии и конфигурации. В таком случае в соответствии с проведёнными ранее вычислениями поток данных при отправке каждого сообщения отдельным HTTP-пакетом в среднем составит около 12 килобайт в секунду.

Выявление минимального набора необходимых заголовков [10], а также группировка отправляемых пакетов может позволить уменьшить размер отправляемых сообщений. При граничных оценках (1.2 % полезной нагрузки при отправке 1920 байт в секунду) объём отправляемых пакетов для телеметрии равняется 160 килобайтам в секунду, что позволяет говорить о пригодности HTTP для передачи MavLink сообщений.

2.4. Каркас веб-приложения

В процессе развития индустрии веб-разработки было создано множество инструментов, призванных упростить написание программных продуктов и уменьшить количество возникающих в процессе разработки ошибок. Эти инструменты варьируются от небольших библиотек до web-framework'ов (каркасов веб-приложения) и даже надстроек над JavaScript (к примеру, таких, как TypeScript). Использование этих средств значительно упрощает разработку и зачастую не несёт за собой падения производительности [19].

В связи с этим в ходе этой работы было решено использовать один из наиболее популярных каркасов веб-приложений. Для этого были выбраны следующие кандидаты:

1. Angular[6]
2. React[7]
3. Vue[8]

К ним был применен метод взвешенных критериев [23], позволяющий определить наиболее подходящий базовый инструмент разработки. Этот метод состоит из двух частей:

1. Производится оценка кандидатов по каждому из критериев.
2. Сравнивается сумма полученных оценок для каждого из кандидатов.

Подобный метод выбора подвержен субъективности восприятия оценивающего на первом этапе, и для минимизации этой погрешности значения для каждого из критериев определялись на основании анализа соответствующего исследования [18]. Оценки являются положительно

[6]https://angular.io/ — Сайт Angular (дата обращения: 22.03.2019).
[7]https://reactjs.org/ — Сайт React (дата обращения: 22.03.2019).
[8]https://vuejs.org — Сайт Vue (дата обращения: 22.03.2019).

ориентированными, то есть сумму оценок следует максимизировать. Полученные оценки указаны в таблице 1.

Сравнение веб-фреймворков			
критерий	Angular	React	Vue
простота изучения	1	3	4
размер приложений	1	4	3
производительность	2	3	5
сумма	4	10	12

Таблица 1: Сравнение веб-фреймворков

Значения оценок в первом пункте объясняются необходимостью изучения TypeScript для работы с Angular. Этот веб-фреймворк также имеет более сложную структуру проекта, имея модули в дополнение к компонентам. Преимущество Vue над React в этом пункте обусловленно синтаксической простотой VueJS, включающего лишь Html, CSS и JS без каких-либо надстроек.

При наличии опции кэширования, достаточного места на БПЛА и наличии достаточного тестового покрытия остаётся лишь критерий производительности. При проведении замеров в исследовании [18] было выявлено, что в среднем на всех тестах Vue обгоняет React в два раза, а Angular — в пять раз.

Преимущество TypeScript в виде гораздо более высокого качества кода из-за наличия строгой типизации [8] в проекте привело к решению об использование TypeScript вместо JavaScript, что соответствующим образом повлияло на финальную оценку веб-фреймворков (Таблица 2).

Сравнение веб-фреймворков			
критерий	Angular	React	Vue
простота изучения	**2**	3	4
размер приложений	1	4	3
производительность	2	3	5
сумма	**5**	10	12

Таблица 2: Оценка веб-фреймворков

Исходя из вышеперечисленных критериев в качестве каркаса веб-приложения был выбран VueJS.

3. Реализация

В этом разделе описывается архитектура разработанного приложения, некоторые используемые библиотеки, детали реализации серверной части и пользовательский интерфейс.

3.1. Основные библиотеки

Добавление различной функциональности зачастую требует подключения сторонних библиотек. Выбранный каркас веб-приложения подразумевает наличие стандартного набора библиотек, которые принято использовать в сочетании с этим каркасом. Для VueJS такими библиотеками являются Webpack[9], Vue Loader[10] и некоторые другие библиотеки, входящие в стандартный набор инструментов Vue CLI[11]. В данном разделе описываются две используемые в проекте библиотеки, не входящие в стандартный набор, но являющиеся ключевыми для разрабатываемого ПО.

3.1.1. Mavlink

Использование MavLink требует предварительного выбора одного из диалектов, являющихся подмножествами команд этого протокола. После выбора диалекта и целевого языка производится сборка соответствующей библиотеки с использованием различных средств кодогенерации. Для данного проекта был выбран наиболее распространённый диалект “Common”[12], классовые файлы для которого были собраны с помощью кодогенератора Mavgenerate[13].

Несмотря на то, что официальная документация Mavlink описывает генерацию библиотек именно с помощью Mavgenerate [4], вер-

[9]https://webpack.js.org/ — сайт Webpack (дата обращения: 30.05.2021).

[10]https://github.com/vuejs/vue-loader — репозиторий Vue Loader (дата обращения: 30.05.2021).

[11]https://cli.vuejs.org/ — сайт Vue CLI (дата обращения: 30.05.2021).

[12]https://mavlink.io/en/messages/common.html — ссылка на документацию диалекта “Common” (дата обращения — 27.05.2021).

[13]https://github.com/mavlink/mavlink/blob/master/mavgenerate.py — репозиторий MavGenerate (дата обращения — 27.05.2021).

сия этого кодогенератора на 30.05.2021 не позволяет сгенерировать работоспособную библиотеку[14]. Для исправления ошибок требуется заменить название импортируемой библиотеки с "node-mavlink" на "@ifrunistuttgart/node-mavlink" в каждом классе сообщения.

3.1.2. Leaflet

Одной из задач НСУ является предоставление пользователю визуальной информации о текущем местоположении БПЛА и расположении контрольных точек миссии. Для этого требуется отображать карту, что подразумевает использование одной из существующих библиотек для интерактивных карт. Наиболее популярными решениями является Leaflet[15], Mapbox GL[16] и OpenLayers[17] [13].

Размер минифицированной версии MapboxGL пятикратно превышает размер Leaflet [17], из-за чего было решено не использовать MapboxGL. При изучении исследования [12], сравнивающего Leaflet и OpenLayers, выяснилось, что обе библиотеки подходят для разрабатываемого в рамках этой работы проекта, так как предоставляют всю необходимую функциональность, имеют обширную документацию и являются open-source проектами с лицензией BSD[18]. Библиотека Leaflet позволяет большую оптимизацию загрузки страниц из-за своей модульной структуры, что является преимуществом по сравнению с OpenLayers, имеющей больший размер библиотеки. Из-за этого в проекте было решено использовать Leaflet.

[14]https://github.com/ArduPilot/pymavlink/issues/432 — обсуждение на Github (дата обращения: 30.05.2021).

[15]https://leafletjs.com/ — ссылка на сайт библиотеки Leaflet (дата обращения: 21.05.2021).

[16]https://docs.mapbox.com/mapbox-gl-js/api/ — ссылка на сайт библиотеки Mapbox GL (дата обращения: 30.05.2021).

[17]https://openlayers.org/ — ссылка на сайт библиотеки OpenLayers (дата обращения: 30.05.2021).

[18]https://opensource.org/licenses/BSD-3-Clause — сайт Open Source Initiative, раздел с описанием лицензии BSD (дата обращения: 27.05.2021).

3.2. Архитектура

Набор инструментов Vue CLI[19] предоставляет возможность создавать как SPA (Single Page Application), так и MPA (Multiple Page Application). Одним из основных преимуществ MPA является его поисковая оптимизация. Для SPA же таким преимуществом является скорость работы страницы [20]. Из-за специфики работы преимущество MPA неприменимо, вследствие чего было решено разрабатывать проект с применением SPA подхода.

Веб-приложение состоит из Vue-компонентов (рис. 3).

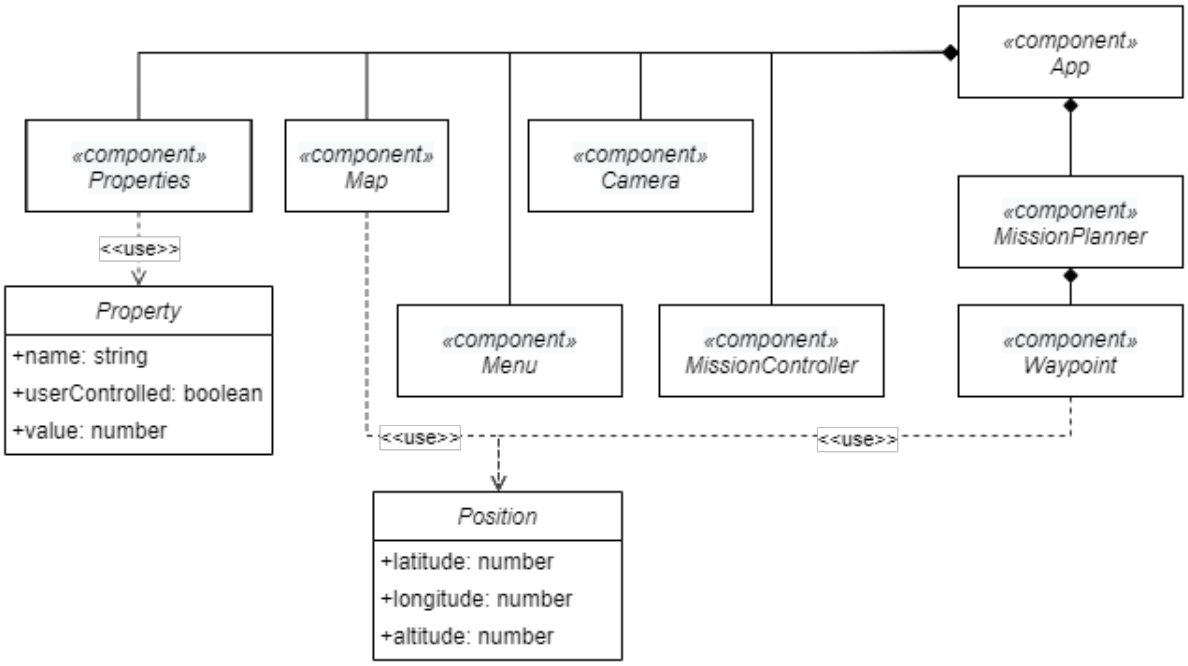

Рис. 3: Архитектура веб-приложения

Основным компонентом является App, управляющий глобальным состоянием приложения. Этот компонент содержит информацию об отображаемом компоненте для визуализации (Камера или карта), выбранном меню настроек (Компонент с данными или планировщик миссии), а также содержит ссылки на текущую миссию и карту. Этот компонент также содержит внутри себя все остальные компоненты приложения и обеспечивает их взаимодействие. После инициализации ком-

[19]https://cli.vuejs.org/ — ссылка на сайт Vue CLI (дата обращения: 26.05.2021).

понент App начинает посылать Heartbeat сообщения[20].

Компонент MissionPlanner отвечает за добавление, изменение и удаление контрольных точек миссии (рис. 4). Он также позволяет изменять порядок прохождения контрольных точек. Кроме добавления самой контрольной точки компонент также добавляет соответствующую ей метку и посылает MavLink сообщения для уведомления БПЛА о новых параметрах миссии.

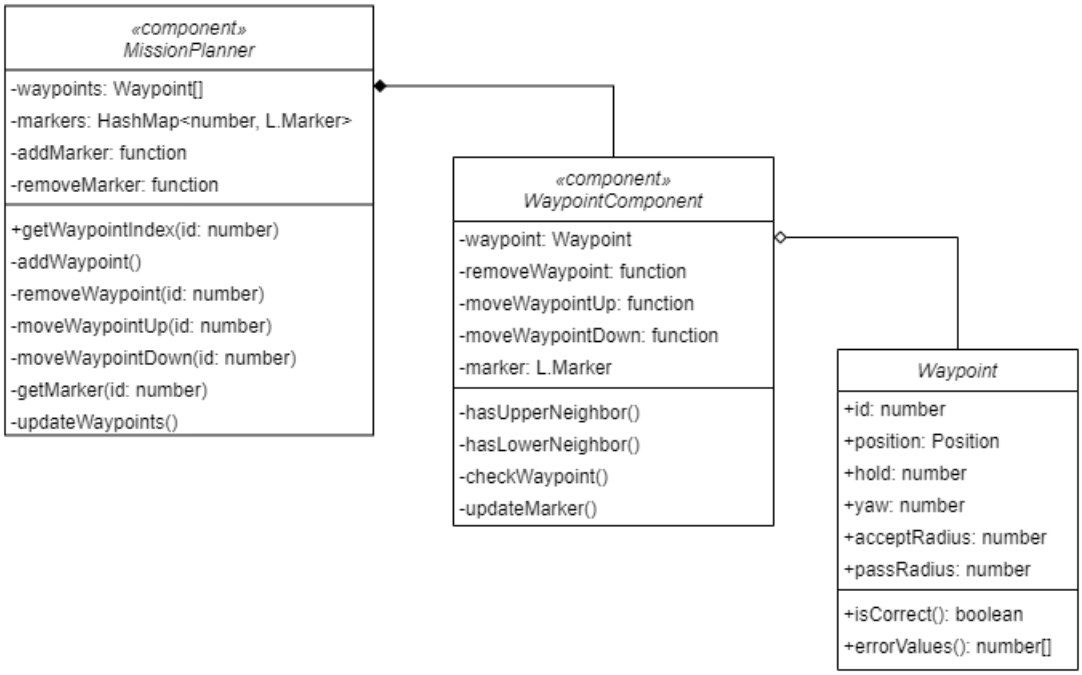

Рис. 4: Компонент MissionPlanner

Компонент Menu отвечает за переключение между компонентами Camera и Map. Компонент Camera получает изображение, а Map конфигурирует и отображает объект L.Map из библиотеки Leaflet. Компонент Map также предоставляет пользователю возможность центрировать камеру по введённым координатам и изменять приближение камеры с помощью текстового поля со значением текущего приближения.

[20]https://mavlink.io/en/services/heartbeat.html — ссылка на документацию Heartbeat сообщения (дата обращения: 21.05.2021).

3.3. Серверная часть

Для работы веб-приложения в CoreCVS должно быть запущено соответствующее серверное приложение (рис. 5) Оно представляет из себя встроенный модульный веб-сервер, сконфигурированный таким образом, что при обращении к URL "/" запрос обрабатывает модуль с ресурсами, к которым относится директория с собранным Vue-приложением. Пользовательские запросы — получение данных с приборов, изображения с камеры и отправка команд управления — обрабатываются соответственно модулями GraphModule, ImageListModule и MavLinkModule.

Для расшифровки полученных mavlink сообщений на сервере подключена собранная с помощью MavGenerate библиотека MavLink с диалектом "Common" для C++.

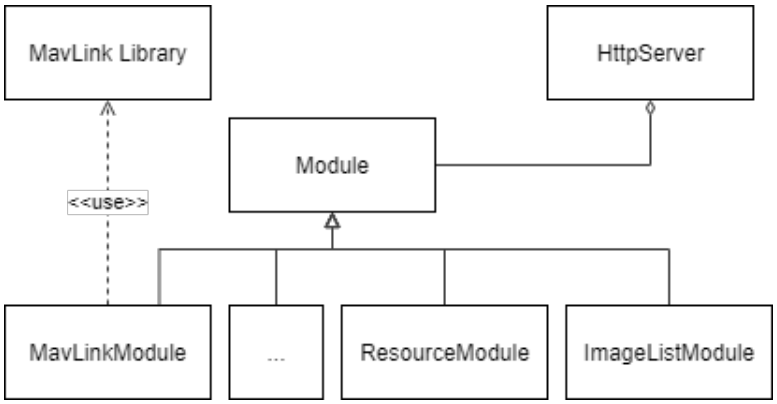

Рис. 5: Архитектура серверной части

Приложение собирается в несколько этапов. Сначала исходные файлы Vue-приложения собираются с помощью vue-cli-service[21]. При сборке CoreCVS полученные файлы записываются в C файл в виде массивов unsigned char с помощью CMake[22] скрипта. При сборке серверной части веб-приложения производится линковка файла с ресурсами, и тем самым Vue-приложение упаковывается вместе с исходным файлом серверного приложения.

[21]https://cli.vuejs.org/ — ссылка на Vue CLI (дата обращения: 30.05.2021).
[22]https://cmake.org/ — ссылка на CMake (дата обращения: 30.05.2021).

3.4. Пользовательский интерфейс

Пользовательский интерфейс оформлен в виде расположенного сбоку меню, раздела с графическими данными и панели ввода данных, переключаемой между списком контрольных точек миссии и списком параметров. Переключение между камерой и картой происходят независимо от переключения между разделом с миссией и разделом с параметрами. Такое разделение позволяет пользователю выводить на экран именно ту комбинацию окон, которая нужна ему в каждый момент управления БПЛА.

Перемещение по карте производится с помощью мыши или настройки центра камеры в соответствующем разделе. Для увеличения отзывчивости интерфейса кнопки выделяются при наведении на них курсора, а текстовые поля ввода подсвечиваются при введении неправильных данных.

Один из видов пользовательского интерфейса представлен на рис. 6.

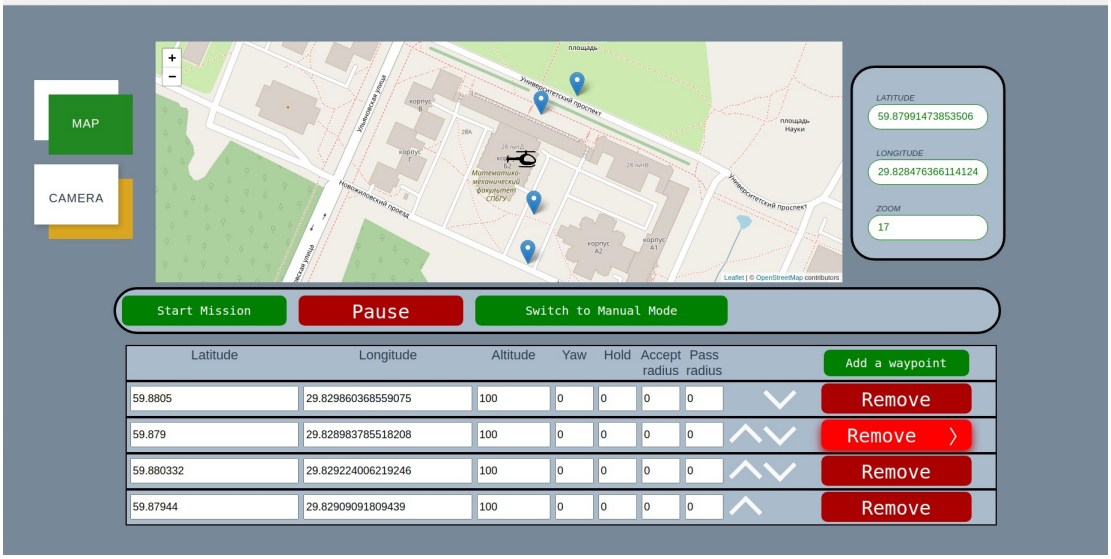

Рис. 6: Пользовательский интерфейс

4. Апробация

На момент написания работы приложение для управления полетом БПЛА всё ещё не разработано, что не позволяет проверить работу веб-приложения на реальной системе. В качестве платформы использовался персональный компьютер с операционной системой Ubuntu 18.04, 16 GB ОЗУ и процессором AMD Ryzen 7 3750H with Radeon Vega Mobile Gfx 2.30 GHz. В качестве клиента выступал другой персональный компьютер, находящийся в той же WiFi сети.

Замеры производительности представлены в таблице 3. Замеры производились с помощью монитора процессов “htop”. Указанные в таблице проценты CPU считаются от одного ядра. Так как основной задачей процессора является управление полетом, на коммуникацию следует отвести не более 5 %-10 % мощности бортового компьютера. Многие целевые платформы — такие, как Jetson Nano или Raspberry Pi 4 — имеют четырехядерный процессор с 1.5 GHz, поэтому для приблизительной оценки удовлетворения такого критерия процент CPU умножался на 2.2 (примерная разница в мощности ядер[23]), делился на 4 (количество ядер), а затем делился на 0.1 для получение процента от максимальной нагрузки в 10 %.

Из-за отсутствия данных с устройства веб-приложение постоянно генерирует различные имитации данных, будь то показания со счётчиков или изображение. Чтобы учесть влияние этих генераторов, они дополнительно запускались без веб-сервера, замерялись показания CPU, а затем показатели используемого процента CPU веб-приложения корректировались соответствующим образом.

[23]www.cpubenchmark.net — сайт с оценкой производительности процессоров (дата обращения: 30.05.2021).

Замеры производительности			
Тест	htop CPU %	Целевой CPU %	С учётом генерации данных
1 клиент, изображение 100x100	1.3 %	7.15 %	2.8 %
1 клиент, изображение 1280x720	8.5 %	46.75 %	23.6 %
10 клиентов, изображение 100x100	2.2 %	12.1 %	7.75 %
10 клиентов, изображение 1280x720	12.9 %	70.95 %	47.8 %

Таблица 3: Замеры производительности

Согласно таблице 3, используемый процент CPU не превышает отметки в 5 % от общего CPU целевой платформы, достигая 2.3 % для одного клиента, что позволяет говорить о достаточно низкой нагрузке бортового компьютера.

Размер build-версии передаваемой клиенту части веб-приложения составил 3.4 Мб. При замере с помощью curl оказалось, что загрузка страницы со всеми сопутствующими вложениями в среднем составляет 0.159898 секунд. С учётом рендеринга это не превышает рекомендуемые [7] значения для задержки загрузки страницы.

Среди запросов, отправляемых приложением, наибольшую задержку отклика имеет загрузка изображения с камеры. При генерации изображения размером 1280x720 пикселей средний отклик составляет 0.04628 секунды, что позволяет говорить о высокой отзывчивости приложения [3].

Заключение

В ходе работы были выполнены следующие задачи:

1. Изучены аналоги и выявлены функциональные требования.
2. Изучен протокол MavLink.
3. Проанализированы веб-фреймворки и выбран VueJS как наиболее подходящий для данного проекта.
4. Разработан прототип веб-приложения.
5. Проведена апробация получившегося приложения.

Исходный код проекта опубликован на GitHub[24].

[24]https://github.com/Darderion/corecvs/tree/apimenov_http — копия Github репозитория CoreCVS (дата обращения: 30.05.2021).

Список литературы

[1] A. Koubâa, A. Allouch, M. Alajlan, Y. Javed, A. Belghith and M. Khalgui, "Micro Air Vehicle Link (MAVlink) in a Nutshell: A Survey," in IEEE Access, vol. 7, pp. 87658-87680, 2019, doi: 10.1109/ACCESS.2019.2924410.

[2] Coombes M. et al. Development of a generic network enabled autonomous vehicle system //2014 UKACC International Conference on Control (CONTROL). – IEEE, 2014. – С. 621-627.

[3] Egger, S., Reichl, P., Hoßfeld, T. and Schatz, R., 2012, June. "Time is bandwidth"? Narrowing the gap between subjective time perception and Quality of Experience. In 2012 IEEE international conference on communications (ICC) (pp. 1325-1330). IEEE.

[4] Generating MAVLink Libraries. Дата обращения: 27.05.2021. Метод доступа: https://mavlink.io/en/getting_started/generate_libraries.html.

[5] Google's SPDY research project whitepaper — SPDY: An experimental protocol for a faster web — Дата обращения: 16.12.2020. — Метод доступа: http://dev.chromium.org/spdy/spdy-whitepaper.

[6] H. de Saxcé, I. Oprescu and Y. Chen, "Is HTTP/2 really faster than HTTP/1.1?," 2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Hong Kong, 2015, pp. 293-299, doi: 10.1109/INFCOMW.2015.7179400.

[7] International Telecommunication Union, "End-user multimedia QoS categories," ITU-T Recommendation G.1010, November 2001.

[8] Jansen, Remo H., Vilic Vane, and Ivo Gabe De Wolff. TypeScript: Modern JavaScript Development. Packt Publishing Ltd, 2016.

[9] K. Tulum, U. Durak and S. K. Yder, "Situation aware UAV mission

route planning," 2009 IEEE Aerospace conference, Big Sky, MT, 2009, pp. 1-12, doi: 10.1109/AERO.2009.4839602.

[10] Kazymyr V. Minimal HTTP header for traffic critical applications / V. Kazymyr, A. Mokrohuz, M. Moshel // Технічні науки та технології. — 2017. — № 2. — С. 123-128. — Режим доступа: http://nbuv.gov.ua/UJRN/tnt_2017_2_16.

[11] Kazymyr, V. and Mokrohuz, A., 2016. In HTTP (S) potential traffic overhead for mobile devices. International Journal "Information Theories and Applications, 23(4), pp.383-393.

[12] Khitrin M.O. Comparison of JavaScript libraries for web-cartography // Вестник ЮУрГУ. Серия: Компьютерные технологии, управление, радиоэлектроника. 2017. №3. URL: https://cyberleninka.ru/article/n/comparison-of-javascript-libraries-for-web-cartography (дата обращения: 27.05.2021).

[13] Map libraries comparison: Leaflet vs Mapbox GL vs OpenLayers - trends and statistics. Дата обращения: 27.05.2021. Метод доступа: https://www.geoapify.com/map-libraries-comparison-leaflet-vs-mapbox-gl-vs-openlayers-trends-and-statistics.

[14] Marques M. M. STANAG 4586—Standard interfaces of UAV control system (UCS) for NATO UAV interoperability //NATO Standardization Agency: Afeite, Portugal. – 2012. – С. 14.

[15] MAVLink Common Message Set Specifications. Дата обращения: 16.12.2020. Метод доступа: https://mavlink.io/en/messages/common.html.

[16] Natarajan, G.: Ground Control Stations for Unmanned Air Vehicles. J. Aeronautical Development Establishment, Bangalore 560075, 5–6 (2001).

[17] NPM trends — leaflet vs mapbox-gl vs mapbox.js vs openlayers. Дата обращения: 30.05.2021. Метод доступа:

https://www.npmtrends.com/leaflet-vs-mapbox-gl-vs-mapbox.js-vs-openlayers.

[18] Saks, E., 2019. JavaScript Frameworks: Angular vs React vs Vue. Дата обращения: 16.12.2020. Метод доступа: https://www.theseus.fi/bitstream/handle/10024/261970/Thesis-Elar-Saks.pdf?sequence=2.

[19] Samra, J., 2015. Comparing performance of plain PHP and four of its popular frameworks. Дата обращения: 16.12.2020. Метод доступа: https://www.diva-portal.org/smash/get/diva2:846121/FULLTEXT01.pdf.

[20] Single Page Application (SPA) vs Multi Page Application (MPA) – Two Development Approaches. Дата обращения: 26.05.2021. Метод доступа: https://asperbrothers.com/blog/spa-vs-mpa.

[21] Y. Kang and M. Yuan, "Software design for mini-type ground control station of UAV," 2009 9th International Conference on Electronic Measurement & Instruments, Beijing, 2009, pp. 4-737-4-740, doi: 10.1109/ICEMI.2009.5274691.

[22] Zeng J. (2011) System Framework and Standards of Ground Control Station of Unmanned Aircraft System. In: Zhu M. (eds) Electrical Engineering and Control. Lecture Notes in Electrical Engineering, vol 98. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-21765-4_41. Дата обращения: 16.12.2020.

[23] Подиновский В. В., Потапов М. А. Метод взвешенной суммы критериев в анализе многокритериальных решений: pro et contra //Бизнес-информатика. – 2013. – №. 3 (25).