Разработка WEB API для оптико-электронной системы межсамолетной навигации беспилотного летательного аппарата

В.Г. Бондарев, E-mail: dimkaao@yandex.ru
Д.В. Лопаткин, О.Н. Роговенко, Д.С. Монгуш, Д.Е. Титов, М.А. Чернышов

Военный учебно-научный центр Военно-воздушных сил «Военно-воздушная академия имени профессора Н.Е. Жуковского и Ю.А. Гагарина»

Аннотация. *В работе рассмотрена разработка программного обеспечения для оптико-электронной системы межсамолетной навигации, разработка графического интерфейса пользователя в формате single page aplication с использованием современных web-технологий на языке программирования javascript и протоколами обмена информации между клиентом и сервером websocket с использованием языка программирования C++.*

Ключевые слова: *Система технического зрения, application programming interface, single page application, оптико-электронная система межсамолетной навигации, протокол обмена данными, websocket, javascript , C++*

Введение

Неоспоримый факт, что на сегодняшний день одно из наиболее актуальных направлений в разработке информационных систем разработка веб-приложений. Эта отрасль стремительно развивается и вместе с тем находит все более широкое применение в решении широкого круга задач в различных сферах деятельности. Особенно высокую востребованность имеет разработка веб-приложений типа single page application (SPA), с использованием средства интеграции приложений application pragramming interface (API) с наличием протоколов обмена информации в режиме реального времени.

В данной работе будет рассмотрена разработка программного обеспечения для калибровки параметров полета группы беспилотных летательных аппаратов, оснащенных оптико-электронной системой межсамолетной навигации с применением современных средств и методов разработки веб приложений. [1]

1. Разработка веб-приложения

Первоочередным в разработке ПО является построение его архитектуры. Проект можно разделить на следующие основные составляющие: модуль с вычислительной частью, написанный на языке C++, модуль с графическим интерфейсом и реализованный с помощью стандартизированного языка гипертекстовой разметки HTML, каскадных таблиц стилей CSS, мультипарадигменного языка программирования JavaScript и инфраструктуры адаптивных компонентов пользовательских интерфейсов bootstrap. Кроме того, реализован вспомогательный модуль сборки, который с помощью кроссплатформенной утилиты автоматизации сборки CMake генерирует файлы сборки и позволяет собрать воедино все подключаемые к проекту библиотеки и файлы.

Вычислительный модуль включает в себя исполнительные и заголовочные файлы форматов «.cpp» и «.h», соответственно реализующие расчеты координат, обработку изображения и передачу данных на сервер. Рассмотрим его подробнее. Работа всего модуля в целом описана в исполнительном файле main.cpp. В первую очередь инициализируются переменные для параметров трэкбаров камеры и детектора.[2]

Структура директории с графическим интерфейсом имеет следующий вид:

–директория подключенных модулей NodeJs;

–директория Public.

В директории Public описан сам интерфейс. Эта директория включает в себя папки css, img, js и файл index.html. В папке css хранятся каскадные таблицы стилей проекта в целом и подключенные файлы библиотеки bootstrap. В папке js находятся файлы, которые обеспечивают функциональную часть интерфейса, конкретно, отправка запросов для получения данных с сервера и отправка данных обратно на сервер, а также анимация отдельных элементов интерфейса, например, загрузки при переходе на страницу интерфейса в браузере.[3]

В папке img хранятся изображения, используемые в интерфейсе. При этом все используемые изображения выполнены в векторной графике. Использование векторной графики позволяет изображениям адаптироваться практически под любое расширение монитора.[4]

Интерфейс (рис. 1) имеет пять вкладок. При их переключении изменяется контент окна «параметры». Окна «изображение с СТЗ», «телеметрия» и «лог работы» остаются неизменными при переключении вкладок.

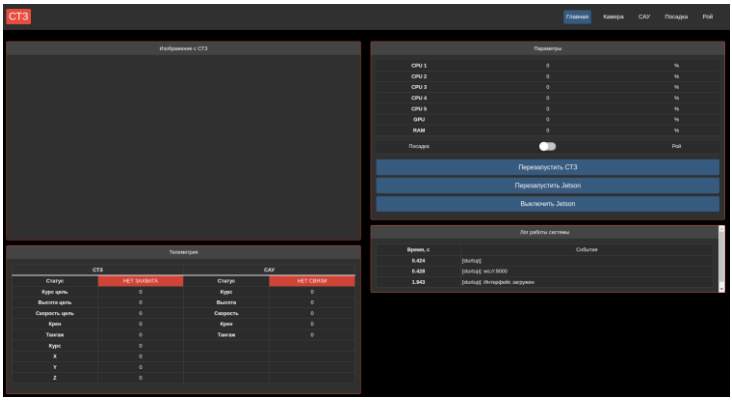

Рис. 1. Графический интерфейс при выключенной СТЗ

Как можно заметить, в верхнем левом углу присутствует красный блок с надписью: «СТЗ». Он является индикатором работы системы технического зрения. При её исправном функционировании он имеет зеленый фон. Также индикаторами работы системы технического зрения и связи с системой автоматического управления служат два поля статуса в окне «телеметрия»

При исправной работе системы технического зрения эти поля аналогично имеют зеленый фон и выдают уведомление о том, что маяки захвачены и связь с автопилотом присутствует. Помимо этого, в окне «телеметрия» выводятся все значения, рассчитываемые в программе написанной на языке C++, а также данные выдаваемые автопилотом. Во вкладке изображения с СТЗ поступает изображение с камеры.

В окне «лог работы системы» в реальном времени выдается статус работы программы, статус подключения к серверу и изменения параметров с указанием времени этого события как показано на рис. 2.

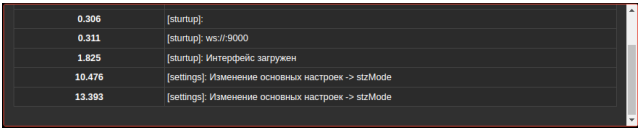

Рис. 2. Работа окна «лог системы» при изменении режима полета

Во вкладке “камера” (рис. 3) задаются значения фокуса камеры, устанавливается режим изображения: цветной или черно-белый и

задаются все диапазоны и значения трэкбаров параметров обработки изображения, получаемого СТЗ.

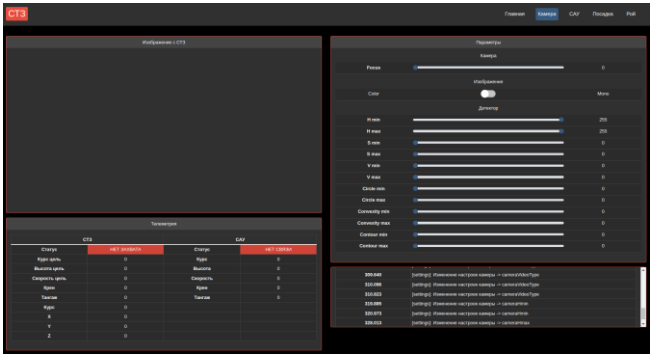

Рис. 3. Вкладка камера

Во вкладке ”САУ” (рис. 4) устанавливается режим управления и задаются его параметры.

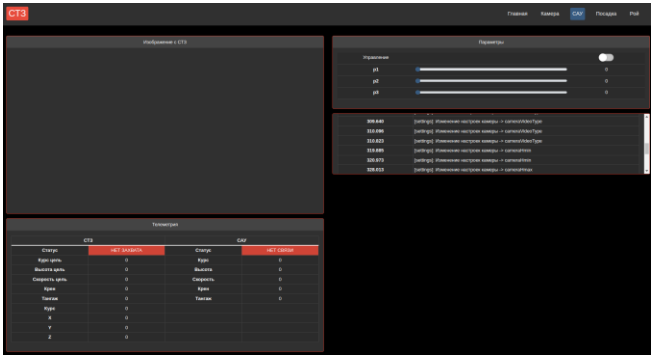

Рис. 4. Вкладка САУ

Во вкладке “посадка” (рис. 5) задается конфигурация маяков для осуществления посадки. Для наглядности присутствует изображение с их расположением и обозначением изменяемых параметров.

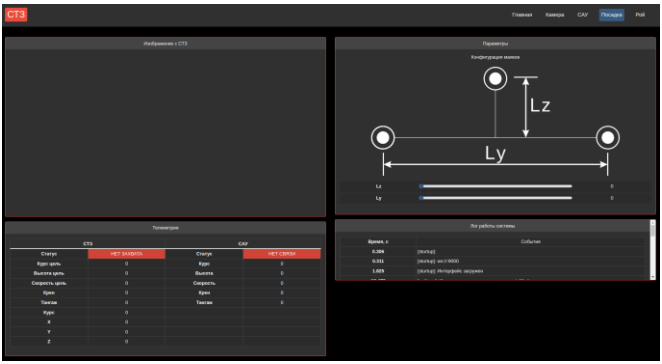

Рис. 5. Вкладка посадка

Во вкладке “рой” (рис. 6) также устанавливается конфигурация маяков, но уже для режима полета. Помимо этого задается конфигурация построения, также с рисунком, на котором обозначены изменяемые параметры.

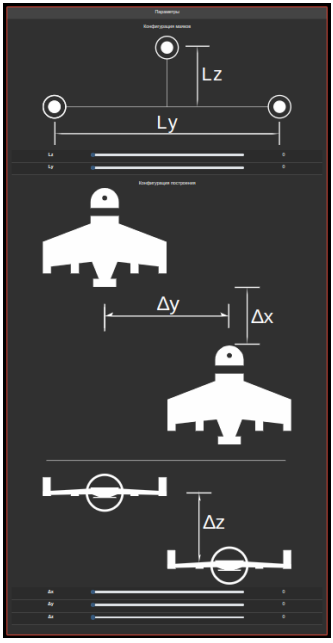

Рис. 6. Вкладка рой

Связь между интерфейсом и системой технического зрения осуществляется посредством запуска локального сервера и установления websocket-соединения. Данный подход позволяет производить обмен данными между сервером и API, в режиме реального времени, что не могло быть осуществимо используя другой протокол обмена, например, http. При таком подходе, для получения актуальных данных с сервера приходилось бы каждый раз обновлять страницу с интерфейсом, что не подходит для реализации поставленной задачи, так как для корректной работы системы технического зрения и наиболее эффективного управления данные должны поступать непрерывно и без задержек.

Принцип работы взаимодействия интерфейса и системы технического зрения следующий. При запуске системы автоматически запускается локальный сервер. Для решения данной задачи был использован Express.js.

В первую очередь пожключается Express. Затем создается класс, который инициализирует веб-приложение и переменную, в которой указывается порт, на котором будет находится содержимое отправляемое на сервер. После чего прописывается метод отправляющий необходимую html форму на сервер и метод будет “слушающий” события поступающие по заданному порту.[5]

Далее взаимодействие осуществляется с помощью прописанных классов в программе клиента и сервера. Задаются условия передачи данных, их парсинг и события, которые отслеживают получение данных и реализуют отправку данных на сервер. Алгоритм обмена данными сводится к тому, что при изменении значений слайдеров в интерфейсе или нажатии на одну из кнопок формируется массив данных, который преобразуется в json-формат и затем создается событие, которое отправляет эти данные на сервер. Параллельно работает метод, который отслеживает эти события и при их обнаружении захватывает данные с сервера, получает их в программе C++ и преобразует из json-формата в необходимый для работы формат. После чего осуществляется двусторонний обмен данными аналогичным способом. В программе C++ данные необходимые для передачи в интерфейс формируют массив формата json и передают их на сервер. Методы, прописанные в программе клиента, отслеживают событие и принимают данные с сервера, после чего также преобразуют из json-формата в необходимый и отправляют в заданные формы html, где они отображаются.

Заключение

В результате разработано кроссплатформенное программное обеспечение с адаптивным графическим интерфейсом, поддерживаемое большинством браузеров, повышающее эффективность калибровки оптико-электронной системы межсамолетной навигации за счет интуитивно понятного расположения элементов управления и возможности изменять параметры полета в режиме реального времени.

Список литературы

1. Верба, В.С. Комплексы с беспилотными летательными аппаратами. Кн. 1. Принципы построения и особенности применения комплексов с БЛА / В.С. Верба. – М.: Радиотехника, 2016. – 512 с.
2. Пат. 2626017 Российская Федерация, МПК51 G 01S 13/46. Способ навигации подвижного объекта / Кудаев А.Н., Косенко А.А., Бондарев В.Г., Ипполитов С.В., Озеров Е.В., Лопаткин Д.В. (РФ); заявители и патентообладатели ВУНЦ ВВС «ВВА». – № 2016130484; заявл. 25.07.16. – 7 с.
3. Дронов, В.А. JavaScript в Web-дизайне / В.А. Дронов. – СПб.: БХВ-Петербург, 2017. – 880 с.
4. Дронов, В.А. Laravel. Быстрая разработка современных динамических Web-сайтов на PHP, MySQL, HTML и CSS. / В.А. Дронов. – СПб.: БХВ-Петербург, 2017. – 768 с.
5. Книга веб-программиста. Секреты профессиональной разработки веб-сайтов / Б. Хоган [и др] . – Москва: Мир, 2013. – 288 с.