РАЗРАБОТКА ПО ЛАБОРАТОРИИ УДАЛЕННОГО ДОСТУПА К РЕСУРСАМ ИМИТАТОРА СМКА

Д. Н. Гринь

Белорусский государственный университет, г. Минск;
dngrin@outlook.com; науч. рук. – И. А. Шалатонин

В данной научной работе предлагается архитектура системы удаленного доступа к ресурсам имитатора сверхмалого космического аппарата (СМКА), призванной решить проблему удаленного обучения студентов аэрокосмических специальностей. Также рассматривается реализация компонента системы под названием Пакетный сервер, предоставляющего сетевой интерфейс к имитатору СМКА. Пакетный сервер написан на языке программирования Python с использованием библиотеки Celery и веб-фреймворка Flask. Его архитектура является модульной, что позволяет добавлять и тестировать новые варианты сетевых интерфейсов и протоколов взаимодействия. Параметры объекта наблюдения, такие как виды датчиков и наборы команд, вынесены в отдельные конфигурационные файлы в формате JSON, что позволяет обеспечить единообразную настройку всех компонентов системы удаленного доступа. Сложные и специфичные телеметрические данные предлагается приводить к единому формату в виде таблицы с помощью специализированных декодеров на JavaScript. Подобная конфигурация объекта наблюдения затем используется в клиентах Пакетного сервера для единообразного отображения данных и представления доступного набора команд.

Ключевые слова: имитатор космического аппарата; система удаленного доступа; python web programming; flask web framework; celery task queue.

ИМИТАТОРЫ КОСМИЧЕСКИХ АППАРАТОВ

С целью обучения специалистов аэрокосмических специальностей и отработки оборудования космических аппаратов широко используются имитаторы и, в частности, имитаторы сверхмалых космических аппаратов (СМКА). Примером такого имитатора является EyasSAT [1], в котором имитируются все основные подсистемы реального космического аппарата: связи, ориентации и стабилизации, питания и пр. Однако данный имитатор обладает некоторыми недостатками: закрытость внутреннего ПО, что сильно ограничивает расширяемость, отсутствие дублирования, которое является нормой на полноценных космических аппаратах, а также довольно высокая стоимость имитатора.

Для решения перечисленных недостатков на факультете радиофизики и компьютерных технологий был разработан собственный имитатор СМКА [2, 3], уже применяющийся в учебном процессе. Имитатор работает на ОС Debian GNU/Linux, внутреннее ПО написано на языках Go, Python и C, используются открытые и известные протоколы передачи

данных и управления пакетными контроллерами (AX.25, KISS), открытые форматы для телеметрии и команд (MsgPack). В системе имеется полноценное дублирование всех подсистем космического аппарата.

К сожалению, существующее ПО наземной станции обеспечивает возможность работы с имитатором только лишь в пределах учебной лаборатории. Для того чтобы открыть доступ к имитатору из любой точки мира посредством Интернета было принято решение разработать новую систему удаленного доступа.

АРХИТЕКТУРА СИСТЕМЫ УДАЛЕННОГО ДОСТУПА

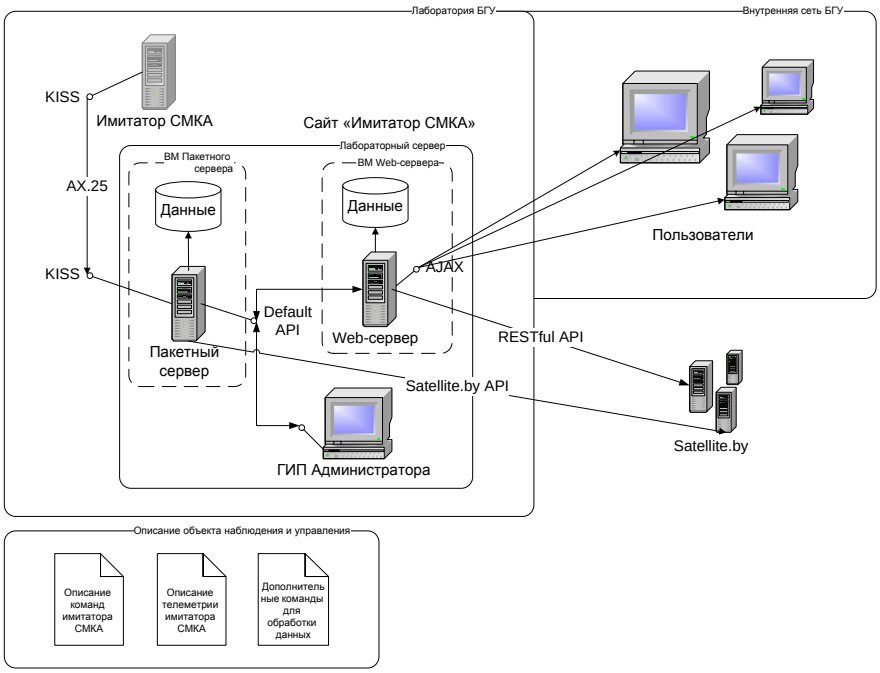

Рис. 1. Архитектура системы

На рис. 1 представлена архитектура системы и возможный вариант подключения её компонентов с использованием одного единственного компьютера и нескольких виртуальных машин. Список основных компонентов: **Имитатор СМКА**, **Описание объекта наблюдения и управления** (включает наблюдаемые параметры, возможные команды и декодеры дополнительной или нестандартной информации, используется для обеспечения целостной конфигурации всей системы), **Пакетный сервер** (предоставляет удаленный доступ клиентам через различные API с опциональной гарантированной доставкой), **Web-сервер** (является клиентом пакетного сервера, предоставляет доступ к информации и управлению пользователям через сеть Интернет, использует сторонний сервис satellite.by для аутентификации и авторизации пользователей), **Satel-**

lite.by (сервис для хранения и предоставления различной информации по космическим аппаратам в сети Интернет, также является клиентом пакетного сервера, но использует свой API) и **ГИП Администратора** (клиент пакетного сервера, обычное приложение, предоставляющее полный доступ к объекту).

РЕАЛИЗАЦИЯ ПАКЕТНОГО СЕРВЕРА

Архитектура Пакетного сервера представлена на рис. 2. Он был создан с использованием следующего стека технологий: язык программирования Python 3, очередь задач Celery (в связке с брокером сообщений RabbitMQ), микро-вебфреймворк Flask, СУБД PostgreSQL с ORM библиотекой SQLAlchemy, библиотека KISS для работы с пакетным контроллером, библиотека requests для выполнения HTTP запросов.

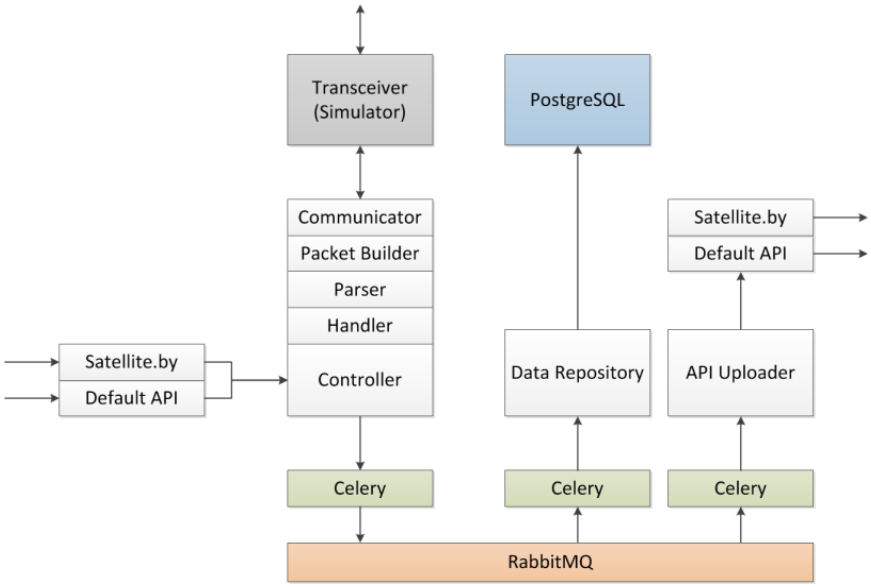

Рис. 2. Архитектура Пакетного сервера

Пакетный сервер состоит из двух частей – Flask-приложение и процесс-исполнитель (worker) Celery. Задача Flask-приложения – взаимодействовать с приемопередатчиком и выполнять команды, поступающие в виде HTTP запросов. Процесс-исполнитель Celery выполняет фоновые задачи, запускаемые Flask-приложением (запись данных в БД, проброс данных от объекта клиентам и т.д.).

Основные компоненты, обеспечивающие создание и разбор пакетов имитатора СМКА (см. рисунок 2): **Communicator** (самый низкий уровень, принимает и отправляет пакеты для объекта), **PacketBuilder** (используя определенный протокол, выделяет/добавляет информацию о пункте назначения и источнике к полезным данным), **Parser** (переводит полезные данные в формате наблюдаемого объекта в ассоциативный

список), **Handler** (переводит данные во внутренний формат Пакетного сервера, определяет, является ли пакет ответом на команду или телеметрией) и **Controller** (осуществляет непосредственную обработку данных по уровням и предоставление внешнего интерфейса, имеет callback на принятие пакета и метод для отправки команд).

ВЫВОДЫ

Многоуровневая архитектура системы позволяет адаптировать её к широкому классу задач наблюдения и управления всевозможными объектами. В частности, в качестве демонстрации этого был реализован виртуальный Имитатор СМКА, который можно использовать для тестирования и обучения. Модульность API позволяет создавать новые интерфейсы взаимодействия с наблюдаемым объектом и расширять клиентскую базу системы. Описание объекта, например Имитатора СМКА, хранится в конфигурационных файлах, которые также задают конфигурацию пользовательского интерфейса. Любое изменение конфигурации или добавление нового декодера отразится во всех компонентах системы путем обычного обновления дерева исходных кодов. Такая организация обеспечивает гибкую систему для взаимодействия с объектом различных сторон. Благодаря гибкости и модульности, данное ПО позволяет работать как со старым имитатором СМКА, так и с новым, который находится в разработке.

Библиографические ссылки

1. EyasSat Classroom Satellite [Электронный ресурс]. URL: https://www.cubesatshop.com/product/eyassat-classroom-satellite (дата обращения: 24.05.2019).
2. Учебно-методический комплекс на базе программно-аппаратных средств для лабораторной отработки нано и пикоспутников / И. А. Шалатонин [и др.] // Комплексные проблемы техносферной безопасности : материалы Междунар. науч.-практ. конф. Воронеж, 2017. Ч. IV. С. 54–58.
3. Использование программно-аппаратных средств для лабораторной отработки нано и пикоспутников для тестирования оборудования / А. А. Спиридонов [и др.] // Комплексные проблемы техносферной безопасности : материалы Междунар. науч.-практ. конф. Воронеж, 2017. Ч. IV. С. 58–63.
4. Проект первого белорусского университетского наноспутника [Электронный ресурс]. URL: http://satellite.by (дата обращения: 24.05.2019).