РАЗРАБОТКА СИСТЕМ УПРАВЛЕНИЯ КАМЕРОЙ ДЛЯ СПУТНИКА ФОРМАТА CUBESAT

Оразов А.В.

г. Орел, МБОУ «СОШ №50», 9 «А» класс

Руководитель: Демушкина О.В., г. Орел, МБОУ «СОШ №50», учитель информатики высшей квалификационной категории

Во время смены «Большие вызовы», организованного ОЦ «Сириус», я занимался апгрейдом наноспутника SiriusSat. Моей целью было установить на спутнике камеру Raspberry Pi Camera Module v2.1 на платформе Raspberry pi. Об этом процессе я и хотел бы рассказать.

Цель эксперимента: установка камеры на спутник и осуществление автономного управления ею.

Этапы работы:

1. Знакомство со спутником.
2. Цели установки камеры.
3. Процесс подключения.
4. Налаживание связи.
5. Интеграция с бортовым компьютером спутника.
6. Разработка ПО.
7. Передача данных по CAN шине.
8. Анализ работы и выводы.

1. Пару слов о спутнике

Разрабатываемый нашей командой спутник BumbleBee (приложение 1, рисунок 1) должен был стать усовершенствованным аналогом спутников SiriusSat-1 и SiriusSat-2 (приложение 1, рисунок 2), разработанных во время смены «Большие вызовы» в 2017 году. Основными отличиями нашей миссии являлось то, что мы хотели запускать его на полярную орбиту (в отличие от SiriusSat'ов, которые сейчас находятся на орбите МКС), а также провести эксперимент эксплуатации камеры Raspberry Pi Camera module v 2.1 (приложение 2, рисунок 1) в открытом космосе.

2. Зачем устанавливать камеру?

Установка камеры позволяет осуществить:

а) восстановление ориентации спутника в космосе по снимкам с камеры,

б) получение изображения Земли из космоса.

Конечно, есть некоторый процент вероятности, что камера не сможет получить внятного изображения, на это есть ряд причин, например, высокоэнергетические частицы в атмосфере земли, способные засветить матрицу. Но этот риск вполне оправдан: за сравнительно небольшую цену в размере менее 1% от общей стоимости спутника, мы имеем шанс получить дешёвый способ фотографировать космос!

3. Процесс подключения

Налаживание связи

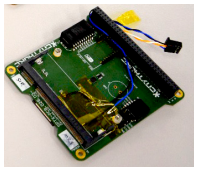
Прежде всего, возникла необходимость «подружить» камеру и процессор Raspberry Pi 3. Для этого к плате была подключена камера, монитор, клавиатура, а далее произведена настройка опций операционной системы «Raspbian» для поддержки работы камеры. Данная операционная система обладает встроенной библиотекой для работы с камерой на языке программирования Python. Самые первые снимки были получены с помощью консольной команды и тривиального кода на этом языке, использующего общую библиотеку.

Интеграция с бортовым компьютером спутника

С подключением камеры к спутнику было не всё так просто. В отличие от используемой для отладки платы, внутри космического аппарата был установлен Raspberry Pi Compute Module 3, для программирования которого требовалась специальная микросхема, которой у нас, к сожалению, не оказалось. Модуль был «намертво» припаян к плате спутника, в связи, с чем нам пришлось вручную выводить и паять требуе-

мые для камеры порты. В результате плата приняла следующий облик (прил. 3, рис. 1).

Разработка ПО



Основными требованиями по функционалу камеры и процессора были:

• возможность по запросу пользователя производить снимок с заданным разрешением и цветным или черно-белым режимами;

• скачивать из внутренней памяти спутника выбранный по имени файл;

• осуществлять повторные попытки скачивания выбранных файлов;

• выводить на экран пользователя список всех доступных снимков во внутренней памяти спутника;

• удалять выбранный по имени файл из внутренней памяти спутника, для освобождения памяти;

• форматировать хранилище снимков спутника.

Очень важно было сделать так, чтобы спутник фотографировал «по щелчку», а потом отправлял фотографию на землю. Операционная система позволяла свободно программировать на языке Python, но я решил попробовать написать код на языках «C++» и «C».Для этого потребовалось установить компилятор. Что заняло около часа по времени, т.к. процессор не обладал достаточной вычислительной мощностью. Кроме того, эту процедуру требовалось повторить несколько раз на разных Compute Module'ях.

В Интернете я нашёл библиотеку для C++, позволяющую управлять камерой и производить снимки. Она обладала всем функционалом, который нам был необходим – регулируемым разрешением картинки и изменяемыми цветовыми режимами. Весь основной функционал писался на C++, но впоследствии пришлось перейти на C. Си++ стал использоваться только для управления камерой. Что послужило причиной такого перехода я объясню позже.

Ещё одной проблемой стало то, что найденная мною библиотека сохраняла снимки в несжатом формате bmp. Они весили порядка 100 кб при маленьком разрешении, что было совершенно недопустимо. Находящиеся в нашем распоряжении антенны могли передавать информацию со скоростью 7000–8000 бод. При больших размерах, была велика вероятность того, что снимок попросту мог не догрузиться и один непринятый пакет (из нескольких сотен пакетов) мог испортить всю картину. Для решения этой проблемы я использовал скрипт на Python, конвертирующий несжатые снимки в jpg, а также провёл опыты с черно-белым снимком. Серая jpg-картинка при наименьшем разрешении весила в ~25 раз меньше, чем цветная bmp-картинка. Затем я добавил в алгоритм захвата снимков автоматическую конвертацию из bmp в jpg и переименования снимка в «1.jpg», «2.jpg» и т.д. Так закончилась работа над первым пунктом.

Вывод списка файлов на экран также был реализован с помощью Python скрипта, вызываемого из основного C++ файла. Некоторые простые задачи я старался решать просто – использовать коды на Python.

Прописать процессы удаления файлов и полного форматирования хранилища также не составило особого труда. Для удобства я разделял типы операций на административные (вывод списка и удаления файлов, форматирование) и рабочие (фотографирование и скачивание снимков).

Продумать все возможности скачивания снимков на подключенный ноутбук оказалось сложнее – на орбите не будет ноутбука и сети типа Wi-Fi.

Передача данных по CAN шине

В космическом пространстве общение между спутником и Землёй производится с помощью антенн. Но, кроме общения с Землёй, спутнику также необходимо «общаться» с его внутри бортовыми системами (рисунок 2, приложение 3). Для этого он снабжён CAN шиной, на сегодняшний день широко используемой в автомобильной промышленности и устройствах «умный дом».

Передача информации по протоколу CAN отличается от привычных для нас способов, например протокола I2C. В CAN шинах типы сигналов делятся на 4 вида: кадр данных, удаленный кадр, кадр ошибки и кадр перегрузки. Я не буду останавливаться на каждом из них, так как мы пользуемся в основном кадром данных для передачи изображений. Кадр данных не может передать большое количество информации, сравнимое с размерами полученных нами фотографий. Для этого информацию необходимо раздробить на множество пакетов, которые затем по очереди отправляются посредством антенны.

Автоматическое фрагментирование информации было включено в специальную библиотеку для работы с информацией в протоколе CAN, которую нам предоставила компания «СПУТНИКС». Но и тут

обнаружились свои «подводные камни» – библиотека была написана для работы с С и не хотела «дружить» с С++. Именно это и послужило причиной переноса всех административных функций на язык С. Мне оказалось проще реализовать простые функции наряду со сложным процессом передачи файлов. И, как уже было упомянуто ранее, С++ использовался только для написания кода манипуляций с камерой.

Для того чтобы спутник начал передавать нам данные, его необходимо было об этом «попросить». Поэтому приём сигналов осуществлялся антеннами, которые по CAN шине передавали их в процессор компьютера. Чтобы исключить возможность перепутывания «команд» системами спутника, каждой из них был присвоен отдельный уникальный номер (ID) «команды», указываемый в самом начале CAN сигнала. Для процессора Raspberry мы выделили отдельные ID «команд». Библиотека позволила не только посылать сигналы, но и принимать их по протоколу CAN. По умолчанию ПО постоянно «говорит» процессору «слушать» какие команды в CAN шине, и если ID сигнала совпадает с ожидаемым ID, то программа считывает данные команды и в соответствии с ними осуществляет нужное действие.

Итогом проделанной работы стал алгоритм работы ПО, с которым вы можете познакомиться в приложении 3(рис. 3).

Заключение

Итак, за месяц, проведённый в ОЦ «Сириус», нашей команде удалось реализовать эксперимент по разработке систем управления камерой для спутника формата CubeSat на 99,5 %. Мы встретили множество препятствий, но ни одно из них не помешало нам реализовать поставленные задачи и прийти к заветной цели.

Прежде я не занимался написанием программ на С++ и Python. Мне были известны лишь основные принципы алгоритмизации и частичное строение кода на С. Полученный за время экспериментальной работы опыт, стал для меня очень ценным.

Итогом работы – были тесты работы функций ПО. Программа помогала стабильно отлавливать сигналы из CAN шины и передавать требуемые команды. Полученные снимки были не высокого качества, но очертания предметов на них вполне различим. При запросе вывода списка фотографий по CAN шине благополучно передавались данные, а после команды удаления при повторном запросе списка можно было заметить уменьшение количества файлов. Скачивание снимка сопровождалось большим потоком данных в терминале CAN шины, тоже происходило при запросе повторной загрузки картинки. А если после тестирования накопилось множество не нужных фотографий, то команда форматирования успешно стирала их. Таким образом, весь необходимый функционал работал отлажено в штатном режиме.

Список литературы

1. Библиотека для работы с камерой на языке С++. – https://www.uco.es/investiga/grupos/ava/node/40.

2. Руководство по настройке камеры – https://projects.raspberrypi.org/en/projects/getting-started-with-picamera.

3. Краткое пособие по CAN шине.. – http://www.micromax.ru/solution/theory-practice/articles/2160/.

Приложение 1

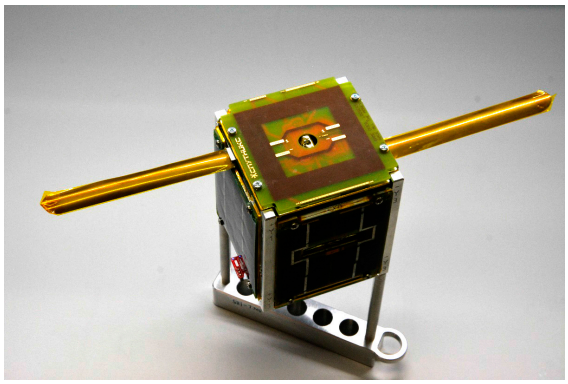
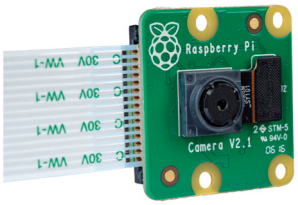


Рис. 1. Наноспутник BumbleBee формата Cubesat

Рис. 2. Наноспутники SirusSat-1 и SiriusSat-2

Приложение 2



Raspberry Pi Camera Module v 2.1

Приложение 3

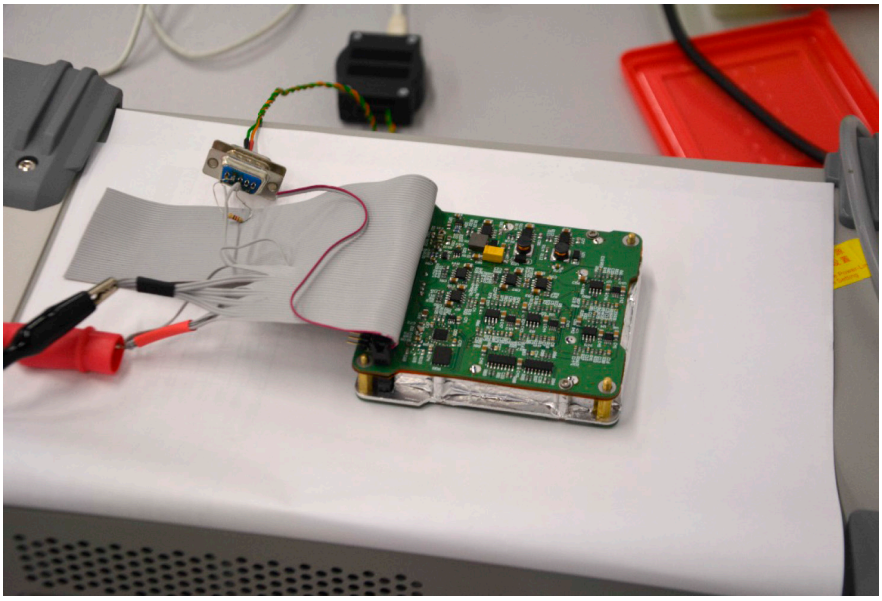


Рис. 1. Перепаянная плата

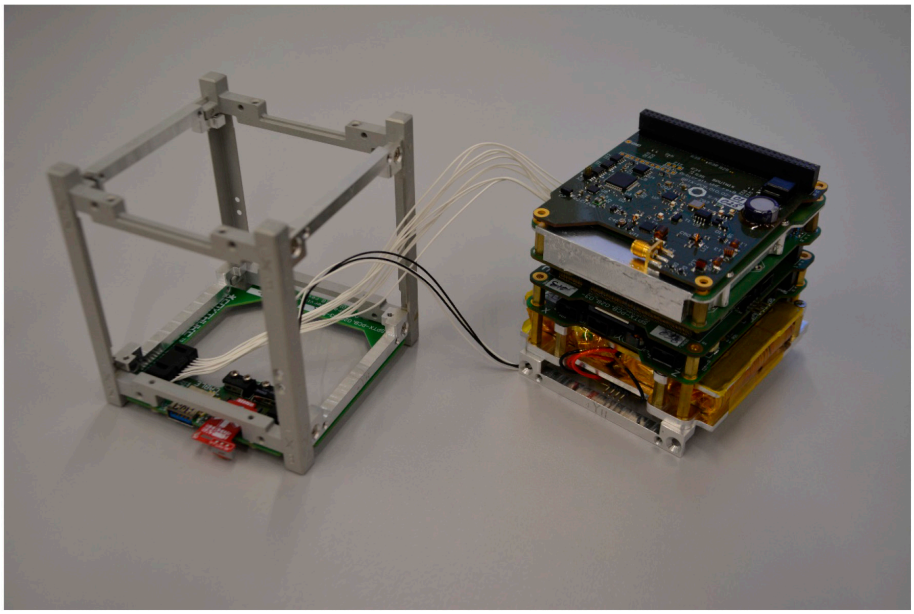


Рис. 2. Спутник в собранном виде

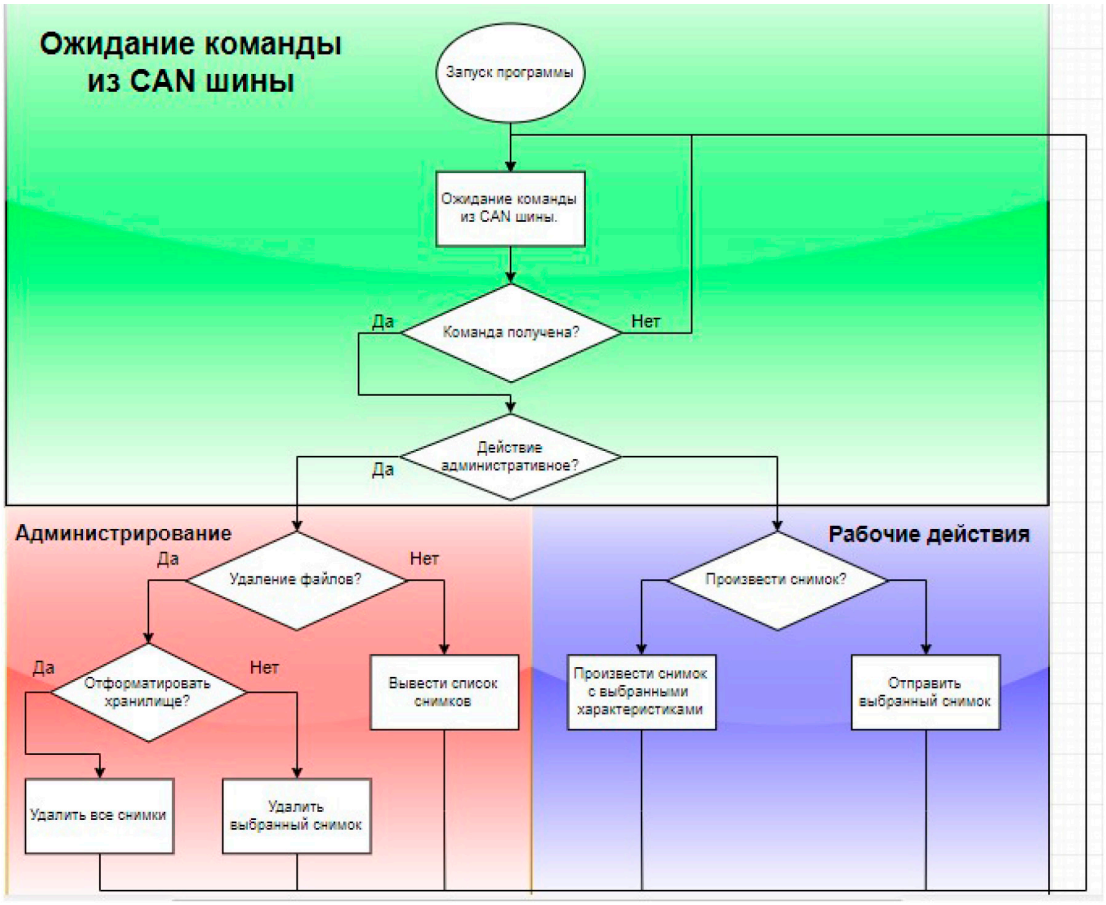


Рис. 3. Алгоритм работы ПО