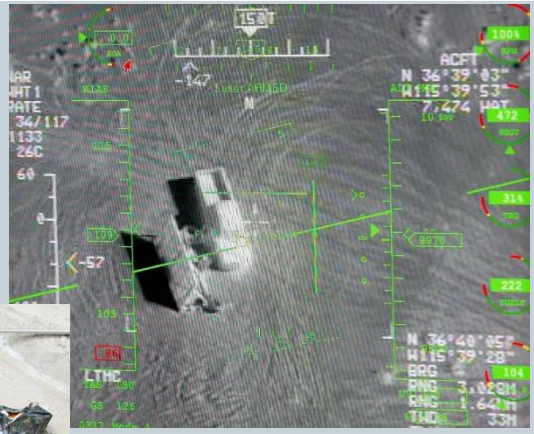
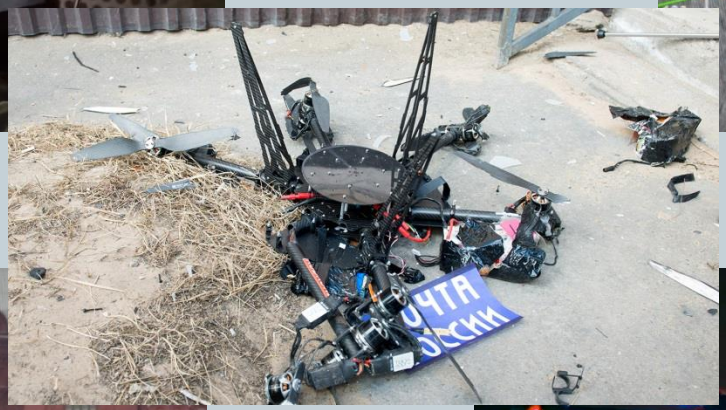
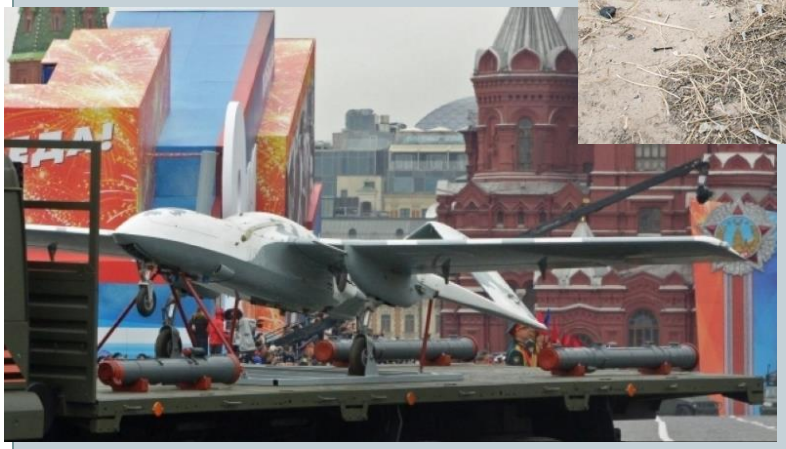
Проектирование надежного программного обеспечения дрона на основе партиционированной операционной системы реального времени

СТАРОЛЕТОВ СЕРГЕЙ
АЛТГТУ, КАФЕДРА ПРИКЛАДНОЙ МАТЕМАТИКИ

Летающие дроны сегодня







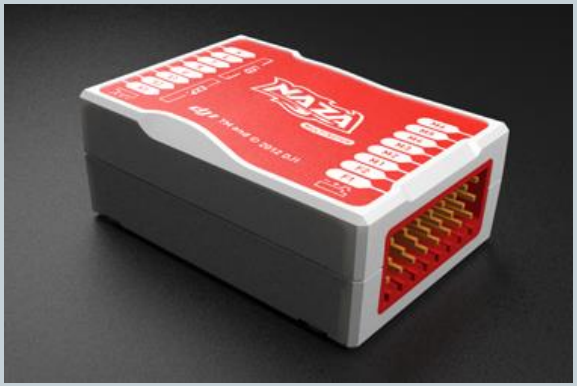
Цель исследования

- Сборка летающего дрона с целью получения прототипа аппаратной части (из комплектующих, массово представленных на рынке)
- Разработка программного решения полётного контроллера с увеличенными требованиями надежности, который будет в какой-то мере отвечать стандартам программного обеспечения авионики ARINC 653
- Получение модели для преподавания курсов проектирования компонентов операционных систем и верификации программного обеспечения.

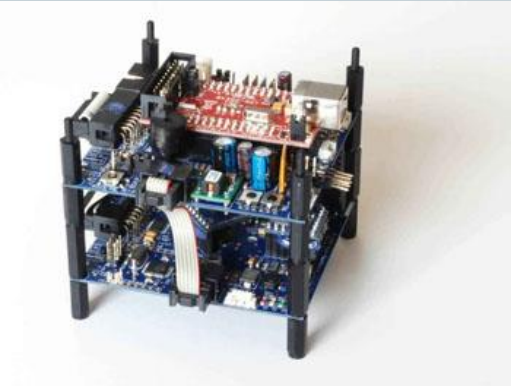
Полётные контроллеры

- Микропроцессорное устройство для управления дроном, причем как для удаленного, так и автономного

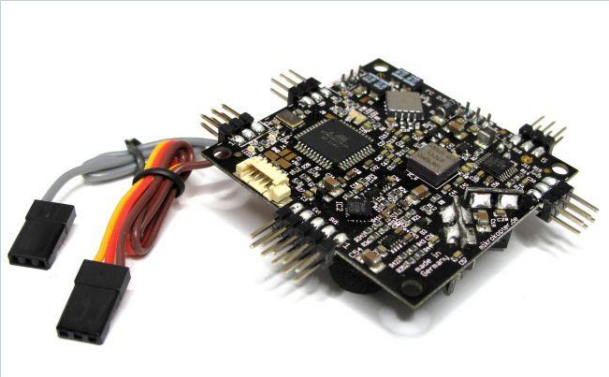
DJI Naza (закрытый)



UAVP-NG



FlightCtrl MicroCopter



Ardupilot (ardupilot.org)

- OpenSource-проект для создания программного обеспечения полетных контроллеров
- Может использоваться как в OpenSource-контроллерах (PixHawk, APM), так и в ClosedSource-решениях (Intel Aero, SnapDragon)

(Отправка из RU) JMT apm2.8 **APM** 2.8 MultiCopter Игровые

1 736,65 - 2 122,71 руб. / шт.
Бесплатная доставка

Pixhawk PX4 автопилот PIX 2.4.8 Полет контроллер 32 бит ARM

3 692,70 руб. / Комплект
Бесплатная доставка

Новинка; Бесплатная доставка; CUAV Pixhack V3 Полет

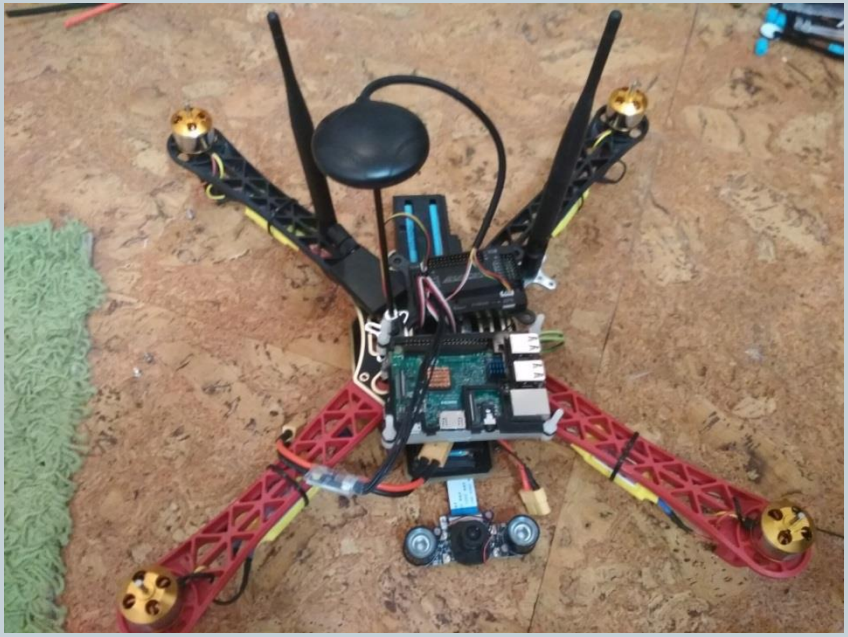
10 643,77 руб. / шт.
Бесплатная доставка

Решение для самостоятельной сборки

- APM 2.6/2.8
- Модуль питания
- Рама F450
- ESC 30A
- Двигатели 2212 1000kv
- 3dr радио телеметрия
- Компас и GPS модуль
- Батарея LiPo 3S 3000mA/h

- Цена: 8000-9000р. Грузоподъемность 1-2кг.
- Время полёта 10-20мин.

Сборка



Постановка задачи на исследование ОС для полётного контроллера

- Обзор исходного кода ardupilot https://github.com/ArduPilot/ardupilot
- Понимание принципа работы оборудования дрона и его контроллера
- Реализация своего решения на основе платы Raspberry Pi3B , исходного кода APM и решений Realtime OS для Raspberry
- Основной принцип – максимальное обеспечение надежности и будущей верифицируемости решения

Эволюция аппаратного обеспечения и кода ardupilot

- Переход от Arduino Mega к 32-битным (STM32F4)
- Branch до 3.1

```
void loop()
{
    uint32_t timer                  = micros();
    static bool run_50hz_loop = false;
    uint16_t num_samples;
    num_samples = imu.num_samples_available();
    if (num_samples >= NUM_IMU_SAMPLES_FOR_100HZ) {
        G_Dt                            = (float)(timer - fast_loopTimer) / 1000000.f;
        fast_loopTimer          = timer;
        mainLoop_count++;
        // Execute the fast loop
        fast_loop();////
        // run the 50hz loop 1/2 the time
        run_50hz_loop = !run_50hz_loop;
        if( run_50hz_loop ) {
            #if DEBUG_MED_LOOP == ENABLED
            Log_Write_Data(51, (int32_t)(timer - fiftyhz_loopTimer));
            #endif
            // store the micros for the 50 hz timer
            fiftyhz_loopTimer               = timer;
            // port manipulation for external timing of main loops
            // reads all of the necessary trig functions for cameras, throttle, etc.
            update_trig();
            // Rotate the Nav_lon and nav_lat vectors based on Yaw
            calc_loiter_pitch_roll();
            // check for new GPS messages
            update_GPS();
            // perform 10hz tasks
            medium_loop();
            // Stuff to run at full 50hz, but after the med loops
            fifty_hz_loop();
            counter_one_herz++;
            // trgger our 1 hz loop
            if(counter_one_herz >= 50) {
                super_slow_loop();
                counter_one_herz = 0;
```

Современный код

```
/*
  scheduler table for fast CPUs - all regular tasks apart from the fast_loop()
  should be listed here, along with how often they should be called (in hz)
  and the maximum time they are expected to take (in microseconds)
 */
const AP_Scheduler::Task Copter::scheduler_tasks[] = {
    SCHED_TASK(rc_loop,              100,    130),
    SCHED_TASK(throttle_loop,         50,     75),
    SCHED_TASK(update_GPS,            50,    200),
#if OPTFLOW == ENABLED
    SCHED_TASK(update_optical_flow,  200,    160),
#endif
    SCHED_TASK(update_batt_compass,   10,    120),
    SCHED_TASK(read_aux_switches,     10,     50),
    SCHED_TASK(arm_motors_check,      10,     50),
#if TOY_MODE_ENABLED == ENABLED
    SCHED_TASK_CLASS(ToyMode,              &copter.g2.toy_mode,         update
#endif
    SCHED_TASK(auto_disarm_check,     10,     50),
    SCHED_TASK(auto_trim,             10,     75),
#if RANGEFINDER_ENABLED == ENABLED
    SCHED_TASK(read_rangefinder,      20,    100),
#endif
#if PROXIMITY_ENABLED == ENABLED
    SCHED_TASK_CLASS(AP_Proximity,         &copter.g2.proximity,        update
#endif
```

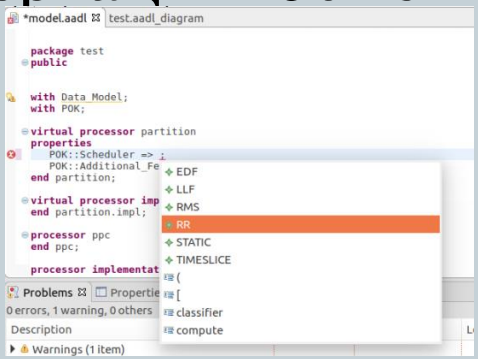
Партиционированные (partitioning) операционные системы реального времени

- Высоконадежная и верифицируемая архитектура требует изоляции ресурсов* (партицирования)
- Ресурсом может быть процессорное время и память, что инкапсулируется в партицию
- Партиция содержит процессы, которые делят ее адресное пространство, но оно недоступно из других процессов
- Планировщик партиций и процессов в партиции
- Порты и каналы как единственный способ контролируемой коммуникации между партициями
- Спецификация ARINC 653 определяет требования к партицированным ОС реального времени

*Mathematically Verified Software Kernels: Raising the Bar for High Assurance Implementations

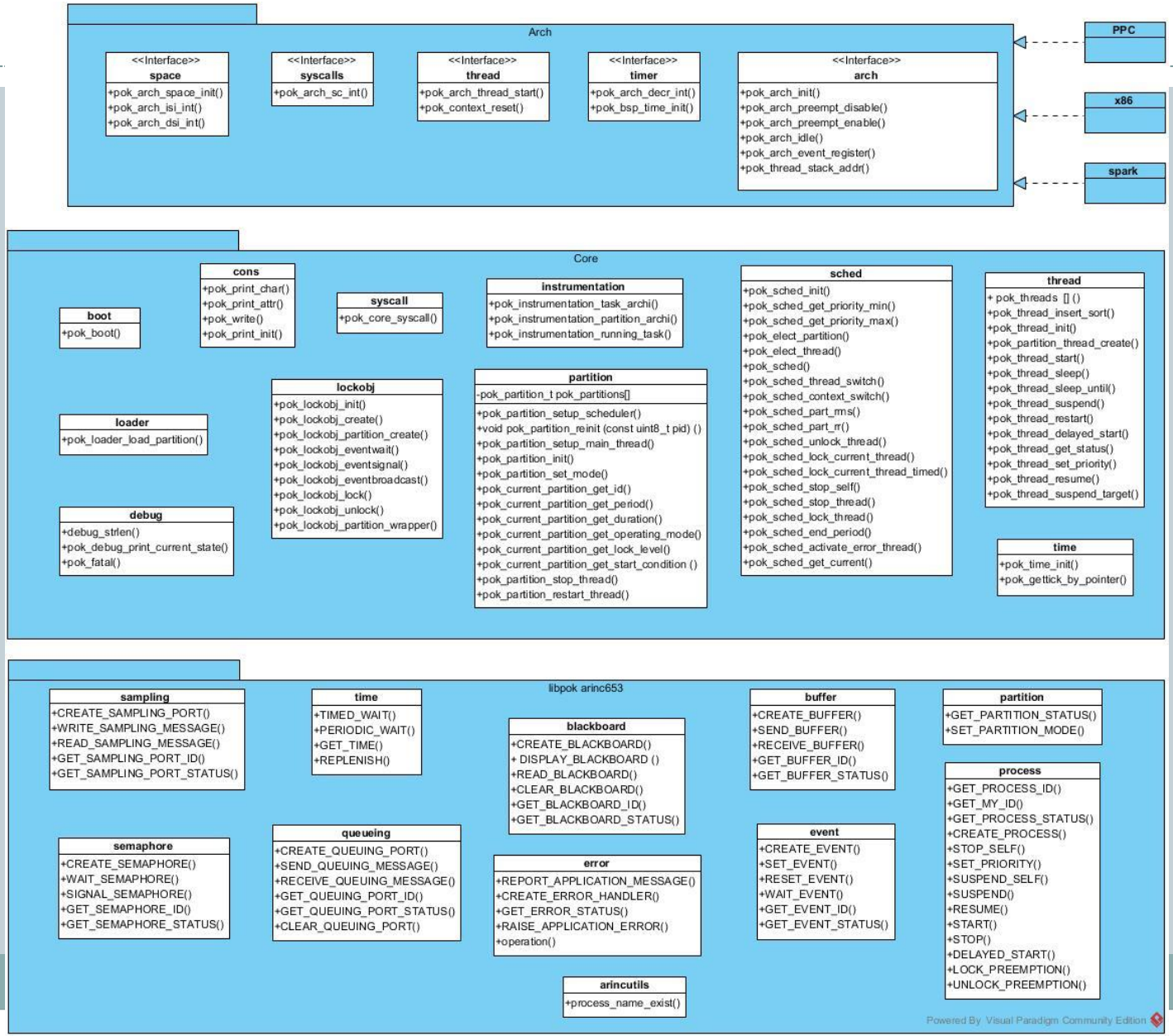
POK (Partitioned Operating Kernel)

- POK – это ядро и примеры его использования, созданные во Франции в качестве диссертации Julien Delange
- Частично совместим с ARINC 653*
- Исходники по BSD-2 доступны на https://github.com/pok-kernel/pok
- Выложена в общий доступ в феврале 2014г.
- Процесс конфигурирования может производиться по принципу MDD с описанием модели на AADL и генерации кода средством Ocarina
- Взят за основу при разработке отечественной ОС для авионики JetOS



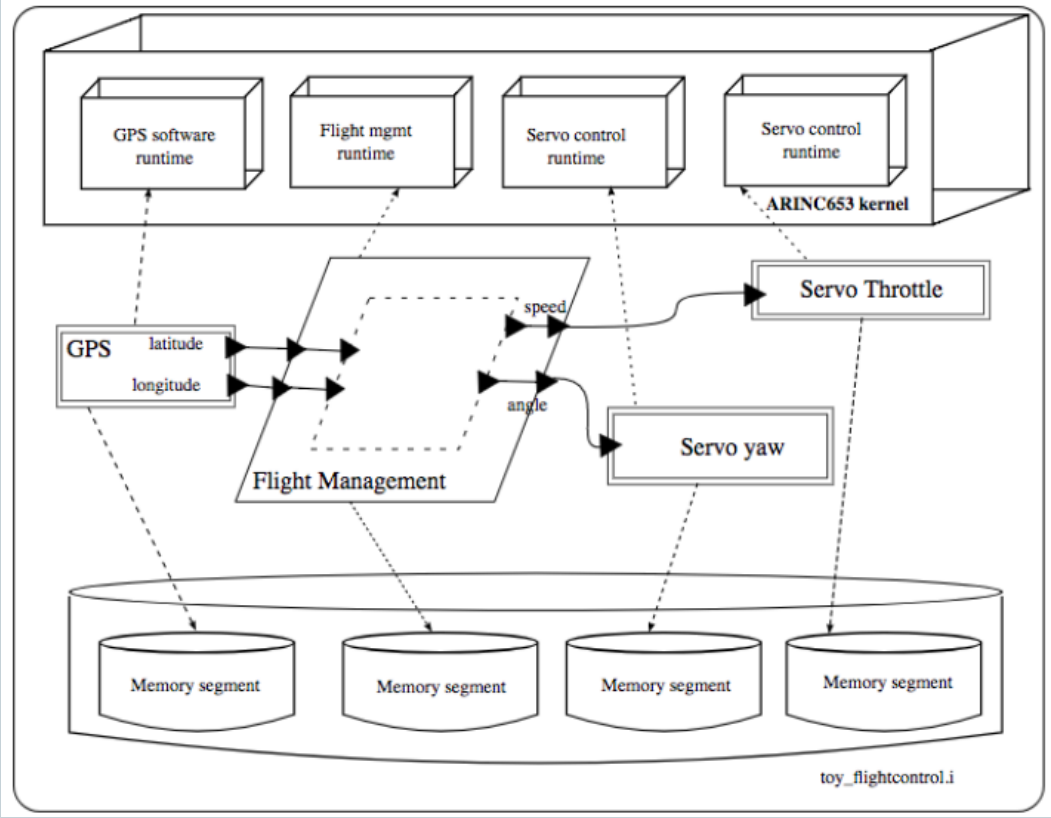
*Design and architecture of real-time operating system

POK (Partitioned Operating Kernel)

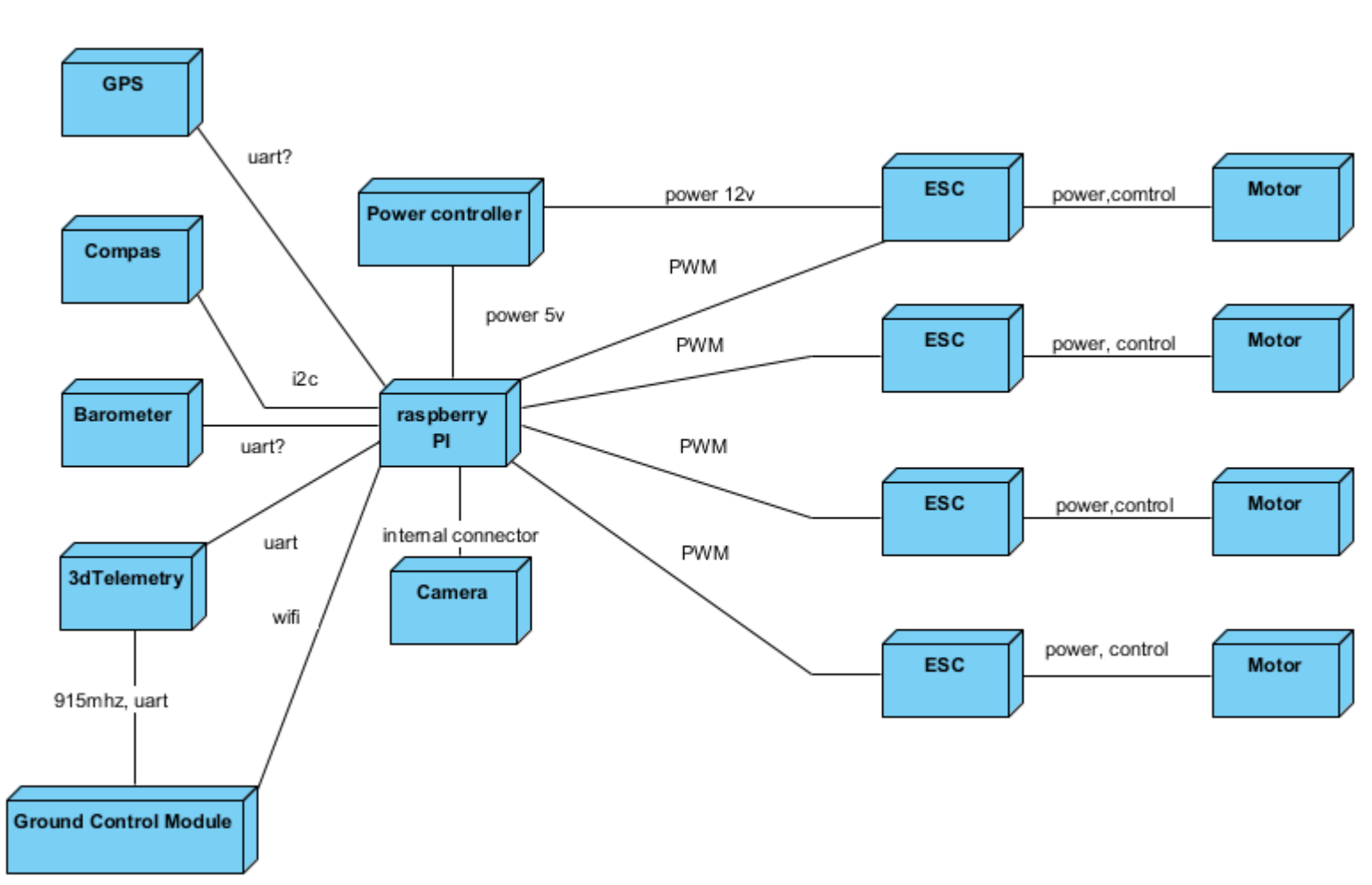


Приложение концепции партицированной операционной системы к Ardupilot

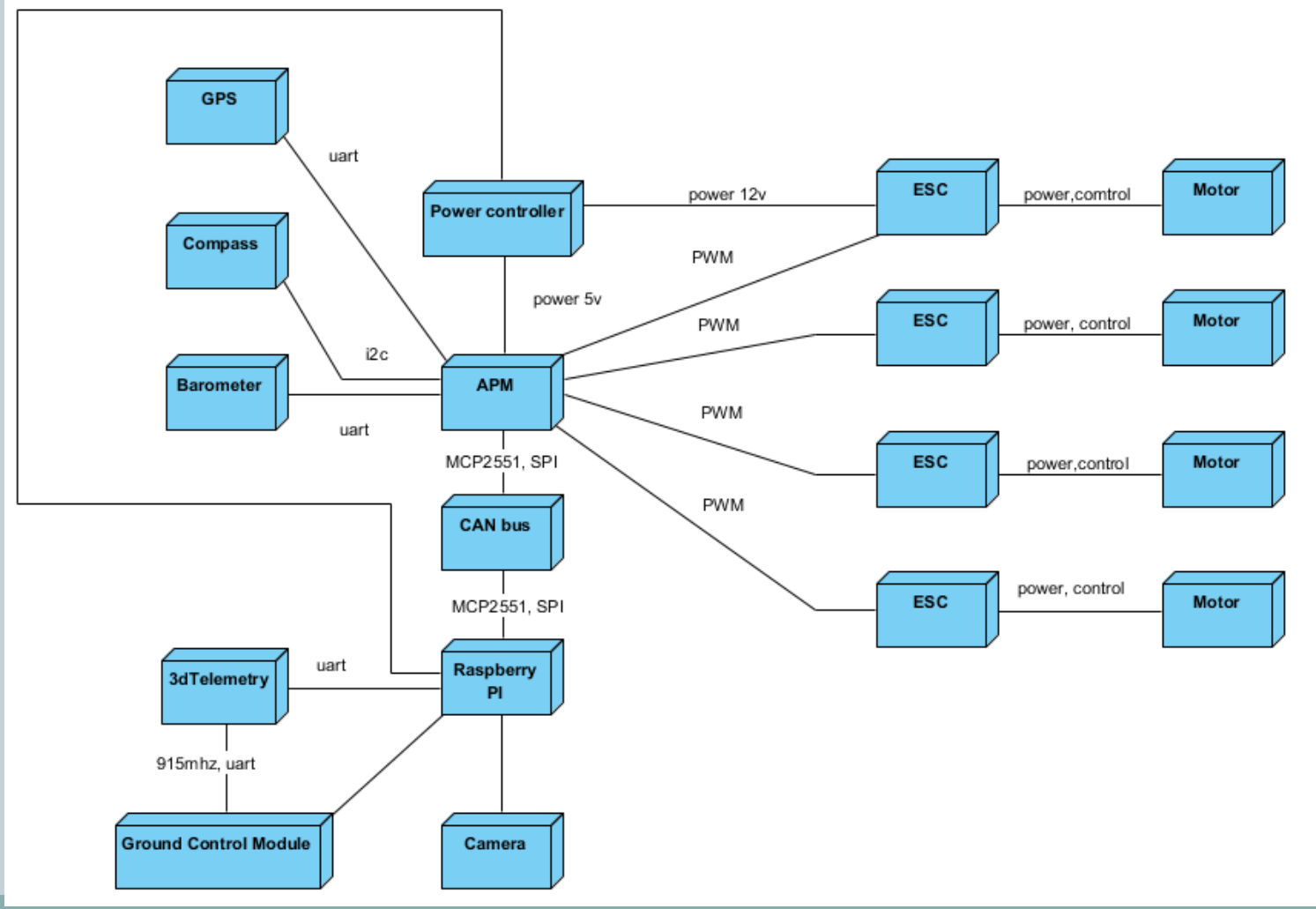
- POK Ardupilot Case study



Возможная архитектура решения



Возможная архитектура решения



Вопросы портирования РОК под ARM

- Raspberry PI3 имеет процессор ARM Cortex-A53, следовательно нужно портирование РОК под архитектуру ARMv8/ARMv7.
- План:

1. Поиск порта RealTime OS под Raspberry*
2. Использование официального firmware для загрузки платы
3. Компилируем с gcc-arm-none-eabi, заливаем и получаем из Raspberry микроконтроллер с main() и работающим диспетчером
4. Далее согласно коду РОК для других платформ, коду Realitme OS, документации на процессор делаем порт
5. Поддержка устройств реализуется согласно Datasheets от BroadCom

* Найдено https://github.com/Forty-Two/RaspberryPi-FreeRTOS

Вопросы надежности и верификации

- Само по себе использование партицированной ОС привносит надежность в используемые технологии
- Согласно заявлениям разработчиков*, 90% кода покрыто
- Дедуктивная верификация кода обработки данных от устройств достаточно тривиальна (циклы, код на состояниях) – ACSL аннотации**
- Код получения данных с регистров устройства под вопросом, так как нужнен поиск сепарационной логики

* https://pok-kernel.github.io/about/ ** ACSL By Example Towards a Verified C Standard Library

Вопросы надежности и верификации

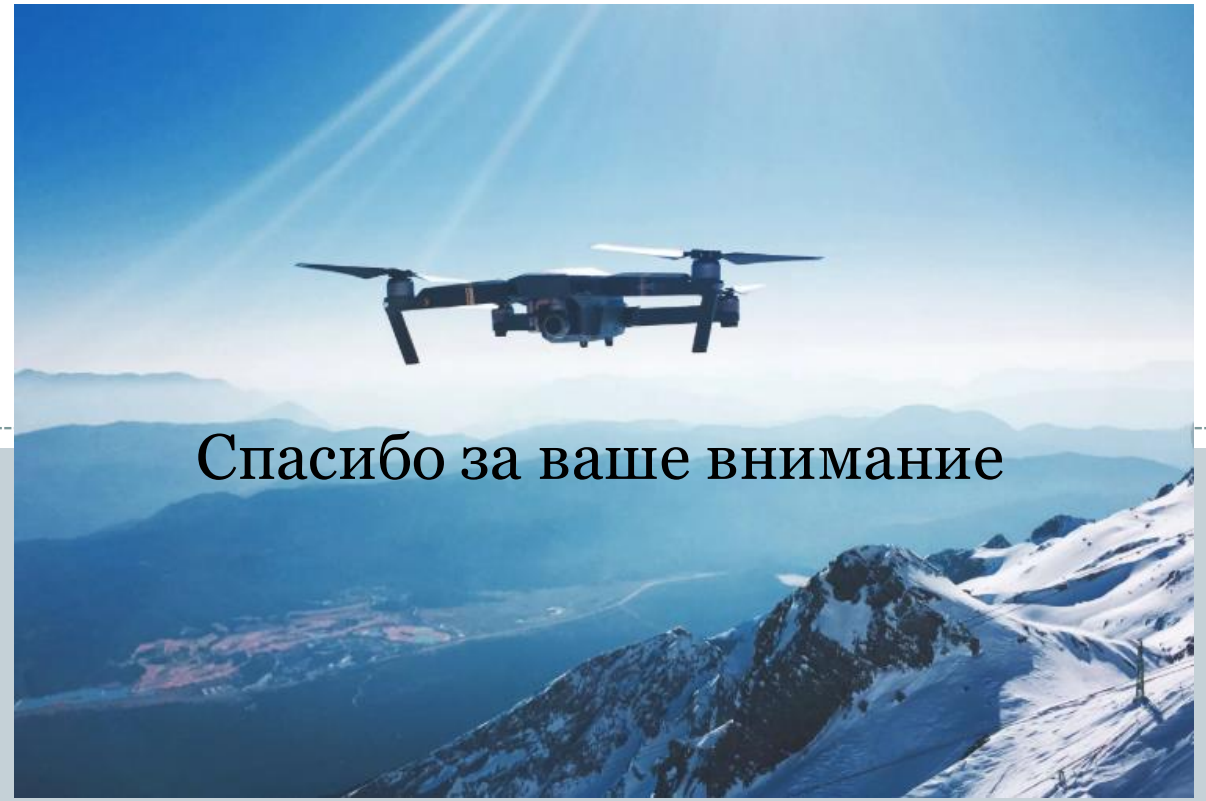
Верификация может быть достигнута:

- Уровень кода: аннотации на код и их дедуктивное доказательство (средство Frama-C)
- Уровень решения: использование расширенных временных взаимодействующих автоматов, средство UPPAAL (поскольку процессы в РОК слабосвязаны, то взаимодействие хорошо ложится на моделирование с акторной моделью)
- Уровень физических процессов: технология гибридной верификации, средство KeYmaera

Вопросы надежности и верификации

- Полная верификация должна включать в себя разработки требований к логике дрона с целью сохранения устройства при повреждении аппаратуры

- Необходимо создание интегрированного проверенного решения, с написанием своего кода как для логики дрона, так и для контроллеров двигателей и модуля питания, которые сейчас выполнены аппаратно



СТАРОЛЕТОВ СЕРГЕЙ
АЛТГТУ, КАФЕДРА ПРИКЛАДНОЙ МАТЕМАТИКИ