Санкт-Петербургский государственный университет

АХРЕМЧИК Ян Валерьевич

Выпускная квалификационная работа

Программно-аппаратная платформа для решения и апробации подходов к визуальной одометрии БПЛА

Уровень образования: магистратура

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

Основная образовательная программа *ВМ.5006.2019 «Математическое обеспечение и администрирование информационных систем»*

Научный руководитель:
д. ф.-м. н., проф Терехов А. Н.

Рецензент:
ведущий разработчик ООО «Системы компьютерного зрения» к.т.н. Федоренко С. И.

Санкт-Петербург
2022

Saint Petersburg State University

Software and Administration of Information Systems Software Engineering

Group

Akhremchik Yan Valerevich

Hardware and software platform for solving and testing approaches to UAV visual odometry

Internship report

Scientific supervisor:
professor Terekhov Andrey Nikolaevich

Saint Petersburg
2022

Оглавление

Введение

Автономные летательные средства давно будоражат умы различного рода фантастов и исследователей. В свете современных веяний в науке и технике, эта задача близка к решению как никогда. Способствует этому всё более набирающая популярность частная авиация в виде всевозможного обилия “коптеров”, используемых для различных целей.

Одной из главных задач для автономного аппарата локализация и навигация в пространстве. Важность этих задач сложно переоценить. Возможность ориентироваться в пространстве необходима для сохранения положения, следования за объектом и так далее. В современных системах проблема локальной и глобальной локализации решается с помощью GNSS (Global Navigation Satellite System) [9]. Такой подход имеет свои ограничения:

- Снижение качества позиционирования при плохих погодных условиях
- Невозможность ориентации в закрытых помещениях
- Возможность потенциального перехвата управления устройством с помощью подмены сигнала со спутника [18]

Решением этих проблем является подход для определения собственного положения без учёта внешних устройств типа GNSS. В дальнейшем понятие “автономной локализации” будет применимо именно с условием данного подхода. Автономная локализация может быть вычислена с помощью различных подходов. Один из наиболее применимых подходов к решению задачи автономной навигации и локализации на БПЛА (Беспилотный Летательный Аппарат) называется “Визуальная одометрия”.

Визуальная одометрия - метод оценки положения устройства с помощью анализа изображений, снятых, установленной на нём одной или несколькими камерами.

Для автономной навигации и локализации с помощью визуальной одометрии существует множество решений. Однако решения редко до-

стигают этапа апробации в реальном физическом мире и в лучшем случае реализуются в симуляторе, а то и вовсе остаются только на бумаге. На рынке присутствуют готовые решения, позволяющие решать задачу визуальной одометрии, например [2] [22], но они либо являются специализированными под конкретную аппаратную платформу и используют её SDK, либо являются и вовсе закрытыми решениями.

В существующих реалиях наличие открытой платформы, позволяющей исследователям испытать свои подходы по автономной навигации и локализации как в симуляции так и в реальном мире, несомненно вывело бы исследования на новый уровень и подогрело бы интерес новых исследователей.

1. Постановка задачи

1.1. Цель

Целью данной работы является проектирование и реализация программно-аппаратной платформы, позволяющей осуществлять решение и апробацию подходов к визуальной одометрии БПЛА.

1.2. Задачи

- Провести анализ предметной области
- Провести анализ каждого из составных компонентов
- Сформулировать подход для реализации программно-аппаратной платформы
- Разработать архитектуру программно-аппаратной платформы для решения и апробации подходов к визуальной одометрии БПЛА
- Реализовать платформу
- Провести апробацию платформы

2. Обзор предметной области

Для проектирования платформы необходимо разбить её на составляющие и исследовать каждый её отдельный компонент чтобы затем иметь возможность спроектировать платформу из необходимых компонентов. Помимо выбора компонентов работы самой платформы необходимо исследовать решения, применяемые для задачи автономной локализации и продемонстрировать работу данных решений на реализованной платформе.

Решения, позволяющие осуществлять автономную локализацию, зависят от двух главных составляющих: аппаратных компонент, позволяющих осуществлять сбор внешней информации и программных компонентов, позволяющих эту информацию обрабатывать.

К аппаратным компонентам можно отнести: лидары, сонары, различного типа камеры и прочие датчики, (однако в рамках данной работы мы остановимся исключительно на камерах). К программным компонентам относится любое программное решение или комплекс таких решений, позволяющих выполнять задачи обработки информации, а также принятия решения на её основе.

Однако аппаратные и программные компоненты необходимо рассматривать в совокупности, так как от аппаратных компонент напрямую зависит используемый подход, а следовательно и программные компоненты.

2.1. Программные компоненты: база

Базовый набор компонентов для платформы подобного типа состоит и связки полётного контроллера и компьютера-компаньона, который полётному контроллеру команды и отдаёт. Для каждого из этих компонентов необходимо выбрать программное обеспечение, поддерживающее как работу на реальном устройстве, так и в симуляции.

2.1.1. Программное обеспечение компьютера-компаньона

ROS Распространённым стандартом для создания платформ такого типа является Robot Operating System (ROS) [17], которая по сути является открытой экосистемой для программирования роботов. Первая версия была выпущена в 2007 году и с тех пор поддержка и разработка продолжается по сей день. Одним из ключевых особенностей можно выделить открытый исходный код, выпускаемый с условиями лицензии BSD. К другим же особенностям можно отнести:

- Одновременная поддержка нескольких языков программирования
- Особый архитектурный подход, который включает в себя следующие понятия:
  - Понятие узлов
  - Понятие сообщений
  - Понятие пакетов
  - Понятие топиков
  - Понятие сервисов
  - Понятие действий
- Нативная поддержка распределённой работы
- Широкая поддержка сообщества

Набор данных особенностей, а так же поддержка со стороны разработчиков программного обеспечения для полётных контроллеров, позволяет обосновать выбор данного решения в качестве основного для разработки платформы. Так же ROS поддерживает работу с популярным физическим симулятором Gazebo [8].

На рынке на данный момент не существует настолько же массивного и модульного решения, которое вдобавок было бы открытым. Однако внутри самого ROS на данный момент существует 2 вариации, которые различны между собой.

ROS и ROS2 Первая альфа версия ROS2 появилась в 2015 году как ответ на требования безопасности, работы в реальном времени и улучшения процесса распределённой работы. На данный момент и продолжается работа по этим направлениям. Соответственно эти направления и стоит вынести в преимущества версии ROS2 над ROS. Однако ввиду относительной новизны версии и радикально изменившемуся подходу к сборке отдельных пакетов и проектов данная версия слабо поддерживается разработчиками и обладает более скудной документацией относительно ROS. Так же ROS2 поддерживает работу "старых" решений через ROS-bridge, что характеризует ROS как более универсальное решение.

2.1.2. Программное обеспечение полётного контроллера

ArduPilot [12] представляет из себя программный пакет с открытым исходным кодом по лицензии GPL[7]. Данный пакет является одним из нескольких, осуществляющих подобную функциональность. Данное ПО применяется как для автономных устройств так и для управляемых устройств с FPV. К особенностям Ardupilot стоит отнести:

- Открытость ПО
- Длительное время разработки и поддержки
- Стабильность ПО
- Широкую поддержку сообщества и разработчиков

Также Ardupilot достаточно хорошо оптимизирован, что позволяет запускать его на достаточно маломощных устройствах. Помимо всего прочего Ardupilot работу в режиме Software in The Loop (SITL), что позволяет его использовать в качестве виртуального устройства при симуляции.

PX4 [13] является другим популярным программным пакетом для полётных контроллеров. Код данного пакета распространяется по лицен-

зии BSD что даёт преимущества при коммерческой разработке. Перед Ardupilot у данного пакета следующие преимущества:

- Более активная разработка проекта
- Большее количество поддерживаемых моделей
- BSD лицензия против GPL

PX4 так же позволяет работать в режиме SITL, однако у него отсутствует режим GUIDED, который в множестве программных компонентов используется для перевода в состояние следования заданной программе.

Вывод Помимо переведённых преимуществ программные продукты являются схожими и схожим образом работают, отличаясь лишь архитектурно. Так как платформа планируется быть открытой, а перед количеством поддерживаемых моделей в приоритете ставится надёжность, было принято решение воспользоваться программным обеспечением Ardupilot.

2.2. Программные компоненты: визуальная одометрия

Так как задачей ставится именно разработка платформы, то для проверки работы платформы будут использоваться уже существующие подходы к визуальной одометрии.

2.2.1. Визуальная одометрия с помощью опорных меток

В качестве решения задачи собственного позиционирования и вычисления относительного перемещения могут применяться опорные метки. Тип меток может варьироваться в различных пределах, от простого квадрата, нарисованного на бумаге до QR кода, несущего в себе байты информации. Важно правильно поставить задачу и подобрать необходимый инструментарий.

Имеет смысл сосредоточиться на трёх основных типах меток, широко используемых в робототехнике: ARTag[6], CALTag[1], AprilTag[23]. К меткам предъявлены требования, которые вытекают из поставленной задачи и позволят обосновать свой выбор.

Требования:

- Метка должна быть робастна к повороту по трём осям
- Метка должна определяться на как можно более дальней дистанции
- Метка должна иметь уникальный номер, по которому её можно однозначно идентифицировать

Исследование Было изготовлено два типа меток:

ARTag: 15.2 x 15.2 cm

AprilTag: 17 x 17 cm

Для каждого из них был проведён набор тестов для выявления наиболее подходящего кандидата. В дополнение к этому были взяты материалы из статьи [1]. В качестве камеры использовалась "sony playstation eye" (640x480@60HZ, fov=75°).

ARTag Для детекции меток типа ARTag использовался модуль "aruco" из библиотеки opencv. Метки данного типа поддерживают уникальную идентификацию путём сопоставления метке её ID. Однако эти метки не очень хорошо справляются с поворотами. При повороте более чем на 60 градусов относительно каждой из осей метка перестаёт распознаваться. Это является одним из ключевых пунктов, так что этот тип метки не применим в данном случае.

CALTag Данные об этой метке были взяты целиком из статьи [1]. В данной статье метка показала высокую робастность к поворотам, относительно трёх осей. Более того, метка проявила устойчивость к разным типам перекрытий, что несомненно является плюсом, однако в нашем случае слабо применимо. Однако метка не имеет собственного

ID, который бы позволил её идентифицировать, что ограничивает её применение. И не подходит в нашем случае. Более того метка плохо детектируется на больших расстояниях.

AprilTag Для данных меток использовалось ПО, доступное на сайте разработчиков AprilTag и имеющее открытый API. Данный тип меток показал робастность к поворотам а также различным типам освещения. В дополнение к этому, метки данного типа детектировались на самом дальнем расстоянии из вышеперечисленных. К тому же алгоритм, для поиска, взятый с сайта разработчиков показал высокую скорость работы, что согласуется с идеей системы реального времени.

	ARTag	AprilTag	CALTag
Робастность к повороту	-	+	+
Максимальная дистанция детекции	~1-2м	~7м	~2-3м
Уникальность детекции	есть	есть	нет

Рис. 1: Сравнение реализаций опорных меток

На основе приведённых данных было принято решение использовать тип метки AprilTag для вычисления собственного положения наблюдателя.

2.2.2. SLAM

Одним из распространённых методов для решения задачи визуальной одометрии является SLAM [4] (simultaneous localization and mapping) - синхронное определение местоположения и составление карты. Метод SLAM состоит из следующих частей:

- Построение карты окружения

- Одновременная локализация (определение собственного положения) на этой карте

Каждая из этих частей может быть реализована различными способами, что приводит к различным вариациям подходов при реализации SLAM. Формально метод SLAM записывается так:

Дано:

- Набор наблюдений $o_{1:t} = \{o_1, o_2, o_3, ..., o_t\}$

Требуется найти:

- Карту окружения m
- Текущее положение наблюдателя x_t

$$P(m_t, x_t | o_{1:t}) \tag{1}$$

Формула (1) описывает апостериорную вероятность существования такой карты m и такого положения x в дискретный момент времени t при данном наборе наблюдений $o_{1:t}$, которое и требуется найти.

$$P(x_t | o_{1:t}, m_t) = \sum_{m_{t-1}} P(a_t | x_t, m_t) \sum_{x_{t-1}} P(x_t | x_{t-1}) P(x_{t-1} | m_t, o_{1:t-1}) / Z \tag{2}$$

$$P(m_t | x_t, o_{1:t}) = \sum_{x_t} \sum_{m_t} P(m_t | x_t, m_{t-1}, o_t) P(m_{t-1}, x_t | o_{1:t-1}, m_{t-1}) \tag{3}$$

Формулы (2) (3) описывают вероятностные распределения при обновлении апостериорного местоположения и апостериорной карты окружения. Сложности в решении такой задачи заключается в том, что формулы (2) (3) должны вычисляться одновременно.

Алогритмы SLAM разделяют по типам информации, которая подаётся на вход:

- Данные с одной камеры (monocular)

- Данные со стереопары (binocular)
- Данные с камеры с датчиком глубины (RGBD)

Однако все они разделяют одну схему работы (рис. 2)

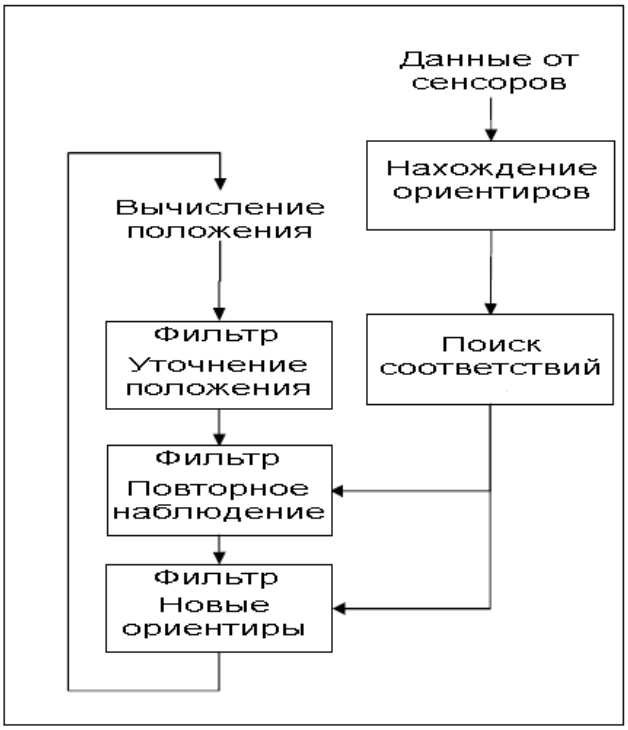


Рис. 2: Общий принцип работы SLAM

ORB-SLAM Одним из самых перспективных подходов на сегодняшний день по реализации SLAM, является ORB-SLAM [14] [15]. Особенностью его яаляется метод нахождения ориентиров ORB (Oriented Fast and Rotated BRIEF). Данный метод позволяет получить ORB-дескрипторы ориентиров с достаточно высокой степенью инвариантности к освещённости, углу зрения и повороту камеры. Также его особенностью является скорость, которая позволяет достигать вычислений дескрипторов в реальном времени без использования графических ускорителей.

В своей статье авторы сравнивают ORB-SLAM с другой популярной реализацией LSD-SLAM [5]. Они рассматривают датасет KITTI [21] на котором вводят две метрики: среднеквадратичную ошибку абсолютного сдвига (the absolute translation RMSE) t_{abs} и ошибки среднего относительного сдвига (the relative translation) t_{rel} и поворота (the relative rotation) r_{rel}.

	ORB-SLAM2 (stereo)			Stereo LSD-SLAM		
Sequence	t_{rel} (%)	r_{rel} (deg/100m)	t_{abs} (m)	t_{rel} (%)	r_{rel} (deg/100m)	t_{abs} (m)
00	0.70	**0.25**	1.3	**0.63**	0.26	**1.0**
01	**1.39**	**0.21**	10.4	2.36	0.36	**9.0**
02	**0.76**	0.23	5.7	0.79	0.23	**2.6**
03	**0.71**	**0.18**	**0.6**	1.01	0.28	1.2
04	0.48	**0.13**	0.2	**0.38**	0.31	0.2
05	**0.40**	**0.16**	**0.8**	0.64	0.18	1.5
06	**0.51**	**0.15**	**0.8**	0.71	0.18	1.3
07	**0.50**	**0.28**	0.5	0.56	0.29	0.5
08	**1.05**	0.32	**3.6**	1.11	**0.31**	3.9
09	**0.87**	0.27	**3.2**	1.14	**0.25**	5.6
10	**0.60**	**0.27**	**1.0**	0.72	0.33	1.5

Рис. 3: Сравнение реализаций SLAM

Как видно на таблице (рис. 3), авторы сравнивают бинокулярные реализации алгоритмов, что играет важную роль в контексте рассматриваемого БПЛА, так как на нём как раз планируется использовать стереопару. Также из таблицы видно преимущество алгоритма ORB-SLAM перед LSD-SLAM.

V-SLAM Отталкиваясь от решений, принятых в аппаратной части, стоит упомянуть про модуль intel realsense t265. Разрабатывая это решение, компания intel на аппаратном уровне реализовала ещё одну реализацию алгоритма SLAM, которую они назвали Visual SLAM. Данная реализация является запатентованной и использует данные с камер вместе с данными с imu. Таким образом они решают задачу не визуальной одометрии, но визуально-инерциальную её вариацию.

	Visual Odometry	IMU Odometry
Update Frequency	Low	High
Stability over Time	High	Low
Scene Dependent	Yes	No
Re-localization	Yes	No
CPU Utilization	High	Low

Рис. 4: Разбиение задачи одометрии в понятии intel

Как видно из таблицы (рис. 4), производитель пытается компенсировать недостатки одного подхода, преимуществами другого.

К сожалению, intel не предоставляет никаких значений метрик, по которым Visual SLAM можно было бы сравнить с другими реализациями. Однако в статье [3] авторы проводят сравнение как раз этих двух подходов. В статье авторы приходят к выводу что оба метода сравнимы с друг другом по качеству и не очень отличаются, однако решене от intel позволяет получать собственную позу в 10 раз быстрее решения ORB-SLAM2.

2.2.3. Визуально-инерциальная одометрия

Это модификация простой визуальной одометрии с добавлением к ней инерциального измерительного устройства (imu). За счет более высокой скорости опроса данного устройства, а так же большей точности

(если отбросить шум от измерений, зная его распределение), тандем из подходов позволяет достичь большего качества результатов.

Kimera-VIO Отдельный модуль библиотеки Kimera [10]. Использует Shi-Tomasi [19] ключевые точки, а затем отслеживает их с помощью трекера Лукаса-Канаде [26] для получения информации о сдвиге между изображениями. Данных подход примечателен тем, что обладает хорошим быстродействием. Более того, при использовании вместе с подходом Kimera-RPGO из той же библиотеки, позволяет с помощью алгоритма замыкания петли определять свою глобальную позицию в пространстве, что делает его эквивалентом SLAM подхода при решении задачи визуальной одометрии.

2.3. Аппаратные компоненты

2.3.1. Датчики сбора информации

Для небольших устройств типа БПЛА существует значительное ограничение на вес переносимых приборов, что соответственно сказывается на размере датчиков их весе, а так же энергопотреблении.

Рассматривая небольшой размер дрона, можно рассчитывать приблизительно на 600-1000 грамм полезной нагрузки. Основываясь на этом числе хочется 10-15% отвести на часть, ответственную за сенсоры. Такой запас обуславливается желанием расширяемости дрона а также необходимостью экономить заряд аккумулятора. Габариты в данном случае не настолько важны, так как решения такого веса обычно имеют соответствующий размер. Итого, имеем ограничения для набора датчиков - 60 - 150 грамм.

Одним из самых важных типов информации об окружающем мире, которую можно получить, является визуальная информация. Для получения такого типа информации используются камеры, либо их системы камер в паре с другими датчиками, для получения более полной информации, например с помощью датчика глубины (TOF sensor), можно получить карту глубины на изображении, которая позволит определять

расстояние от наблюдателя до произвольного объекта.

Одним из поставщиков компактных модулей для компьютерного зрения является компания intel с линейкой устройств Realsense [25]. В рамках данной линейки производится несколько типов устройств.

Устройства категории T (tracking) представляют из себя платформу, в которую входит:

- 2 камеры с углами обзора 170 градусов
- Инерциальный измерительный модуль (imu)
- Интегральная схема специального назначения

Данное устройство позволяет получать видеосигнал частотой дискретизации 30 гц (30 кадров в секунду), данные с гироскопа с частотой 200 гц, данные с акселерометра с частотой 62 гц. Устройство позиционируется как решение для роботизированных систем и беспилотных устройств. Благодаря интегральной схеме, оно позволяет производить вычисление SLAM прямо на устройстве в режиме реального времени.

Категория устройств D (depth) представляет платформу, которая включает в себя:

- Инфракрасный излучатель с углом обзора 67x41x75 градусов
- 2 тепловизора с углами обзора 65x40x72 градусов
- 1 RGB камеру с углами обзора 69x42x77 градусов
- Инерциальный модуль (в одной из модификаций)

Преимуществом данной платформы является возможность получения качественной карты глубины, которую в дальнейшем можно использовать для задачи избегания препятствий. Также есть модификация данной платформы, включающая в себя инерциальный измерительный модуль, что позволяет добиться большей точности, при решении задачи визуальной одометрии (Visual Odometry). Однако на устройствах данного типа отсутствует интегральная схема специального назначения, что означает, что все вычисления должны будут проводиться

на главном вычислительном устройстве. Устройство позиционируется как решение для точных роботизированных задач. Модификация с imu рассматривается как решение для беспилотных устройств, позволяющее решать задачу навигации в условиях малого освещения.

Одним из самых важных аспектов при получении изображения с БПЛА является угол обзора, так как в зависимости от размера области, которую можно увидеть и проанализировать, зависит и количество решений, которое можно принять на основе такого анализа. Также более широкий угол обзора позволяет уменьшить количество устройств при потребности получения панорамного изображения вокруг БПЛА. Не менее важной составляющей является возможность построения карты глубины, ибо от этого напрямую зависит задача детекции препятствий. Наличие imu, встроенного в платформу, приводит к более робастному решении задачи визуальной одометрии, так как позволяет определять направление движения даже при отсутствии или малом количестве особых точек на изображении. Стоит отметить, что все решения из обеих линеек обладают весом < 100 гр, а также энергопотреблением, которое обеспечивает 1 usb порт.

Ввиду всех перечисленных особенностей было выбрано устройство линейки t, а именно intel realsense t265, как обладающее минимальным энергопотреблением из всех, обеспечивающее большой угол обзора, а также обладающее двумя идентичными камерами. Бинокулярное решение позволяет строить карту глубины с большей точностью, нежели чем с использованием монокулярного решения, однако с меньшей точностью, нежели чем с использованием TOF сенсора. Наличие же интегральной платы, упростит решение задачи SLAM, хоть и не обязательно к использованию.

2.3.2. Центральный модуль обработки информации

Для обработки информации и последующего принятия решений необходимо устройство, способное к декодированию видеопотока, а желательно нескольких сразу, в режиме реального времени, и способное к обработке этих данных.

Решения такого типа относятся к категории single board computer (SBC). Рынок SBC устройств очень обширен, начиная от нескольких десятков за устройство и заканчивая платформами за несколько тысяч долларов. Однако в рассматриваемом случае накладываются те же самые ограничения, что и на модули для сбора сенсорной информации, а именно размер, вес, энергопотребление. Кроме того, ввиду сложности операций, для проведения вычислений необходимо GPGPU решение.

Одним из самых распространённых реализаций GPGPU является CUDA от компании Nvidia. К тому же, при желании применять различные алгоритмы машинного обучения, CUDA является единственным выбором для большинства фреймворков, предоставляющих такую функциональность.

Сбалансированным решением по соотношению цена, производительности и энергопотреблению, является линейка jetson от той же компании Nvidia [16]. В рамках этой линейки представлено несколько устройств:

1. Nvidia jetson Nano
    - Module Power: 5-10W
    - GPU: 128-core Maxwell
    - CPU: ARM Cortex A57 4-cores
    - HW video decoding: (8x) 1080p@30
2. Nvidia jetson TX2
    - Module Power: 7.5-15W
    - GPU: 256-core Pascal
    - CPU: ARM A57 2-cores
    - HW video decoding: (14x) 1080p@30
3. Nvidia jetson Xavier NX
    - Module Power: 10-15W
    - GPU: 384-core Volta GPU

- CPU: ARM Nvidia Carmel 6-cores
- HW video decoding (16x) 1080p@30

4. Nvidia jetson Xavier AGX 10-30W

    - Module Power: 10-30W
    - GPU: 512-core Volta GPU
    - CPU: ARM Nvidia Carmel 8-cores
    - HW video decoding: (12x) 4k@30

Устройства предоставляют различный уровень производительности по возрастанию номера в списке. В рассматриваемом случае необходимо смотреть на такие параметры как: тип видеокарты, энергопотребление, возможность аппаратного декодирования видео (желательно не нагружать лишний раз цпу, если это возможно), а также на тип цпу.

Как видно из данных, приведенных выше, самым “простым” решением является jetson Nano, который также обладает минимальным энергопотреблением, что сказывается на времени автономности БПЛА и на необходимости активного охлаждения. Тем не менее данное устройство обладает достаточным запасом вычислительной мощности для выполнения необходимых задач.

Ввиду вышеперечисленных аргументов, в качестве модуля для обработки информации был выбран Jetson Nano.

3. Апробация

3.1. Требования

Платформа должна обладать возможностью работать как на физическом устройстве так и в симуляции для проверки гипотез алгоритмов и ради безопасности. В таком случае общее представление архитектуры можно представить следующим образом:

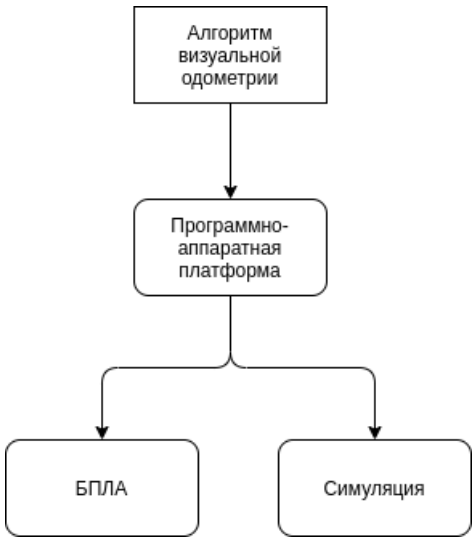


Рис. 5: Общее представление

Таким образом вне зависимости от от типа алгоритма, платформа останется неизменной.

Так же платформа должна обладать свойством модульности, что позволяло бы использовать несколько алгоритмов вместе или по-отдельности не затрагивая опять же саму платформу.

Таким образом получаем более детальное представление платформы:

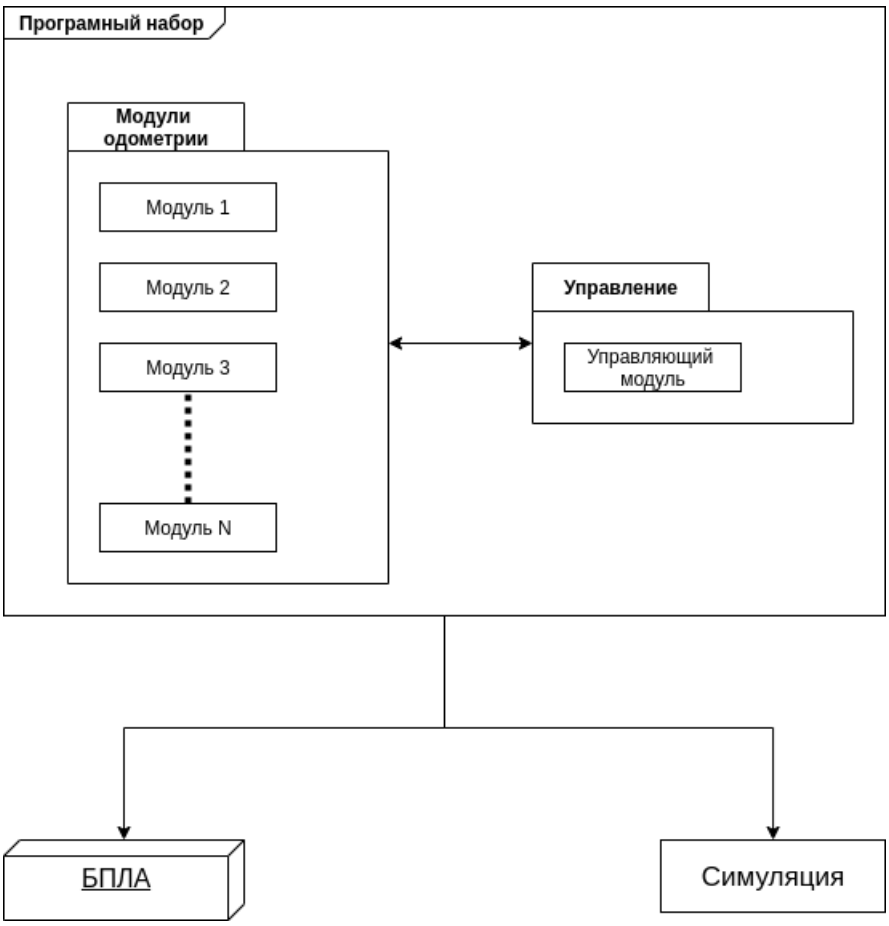


Рис. 6: Детальное представление

Как можно видеть из изображения 6 программная часть платформы состоит из двух компонентов. Один компонент представляет собой набор модулей для визуальной одометрии. Другой компонент осуществляет взаимодействие с модулями одометрии и генерацию управляющего сигнала, который затем передаётся в симуляцию, либо на физическое устройство.

3.2. Автономная локализация

3.2.1. Фильтр Калмана

Для локализации в пространстве необходимо использовать все возможные данные, которые можно получить. Данное решение кажется избыточным только на первый взгляд. Это обосновано тем, что у каждого из методов локализации, будь то локальной или глобальной, существу-

ет погрешность. В случае с General Navigation Satelite System (GNSS), имеет смысл говорить об абсолютной локализации и её погрешности, которая зависит от качества принимаемого сигнала и количества антенн.

В случае с локальной локализацией мы говорим об локализации, относительно окружающего пространства. В данном случае это зависит от неидеальности сенсоров, с которых получаются данные, а также алгоритмов, которые эту локализацию высчитывают. Более того в контексте локальной локализации существует проблема накопления ошибки и релокализации(определение возвращения в исходную точку).

Одним из решений этих проблем является фильтр Калмана [24]. Данный фильтр позволяет дооценить текущий вектор состояния и тем самым уточнить собственное положение. Данный процесс происходит с помощью получения и анализа данных с нескольких источников. Чем больше источников для получения информации, тем точнее оценка.

В качестве источников информации выступают сенсоры, получающие информацию о передвижении БПЛА, а так же алгоритмы, позволяющие получать эту информацию с помощью камер. В качестве сенсоров используются следующие:

- GNSS
- imu, встроенный в pixhawk
- imu, встроенный в realsense камеру

В качестве алгоритмов для оценки используются следующие:

- Intel VSLAM
- ORB-SLAM2
- Apriltag estimator

Набор данных объединяется и после применения фильтра Калмана результатом будет точная оценка абсолютного местоположения. Подход проиллюстрирован на рис. 7

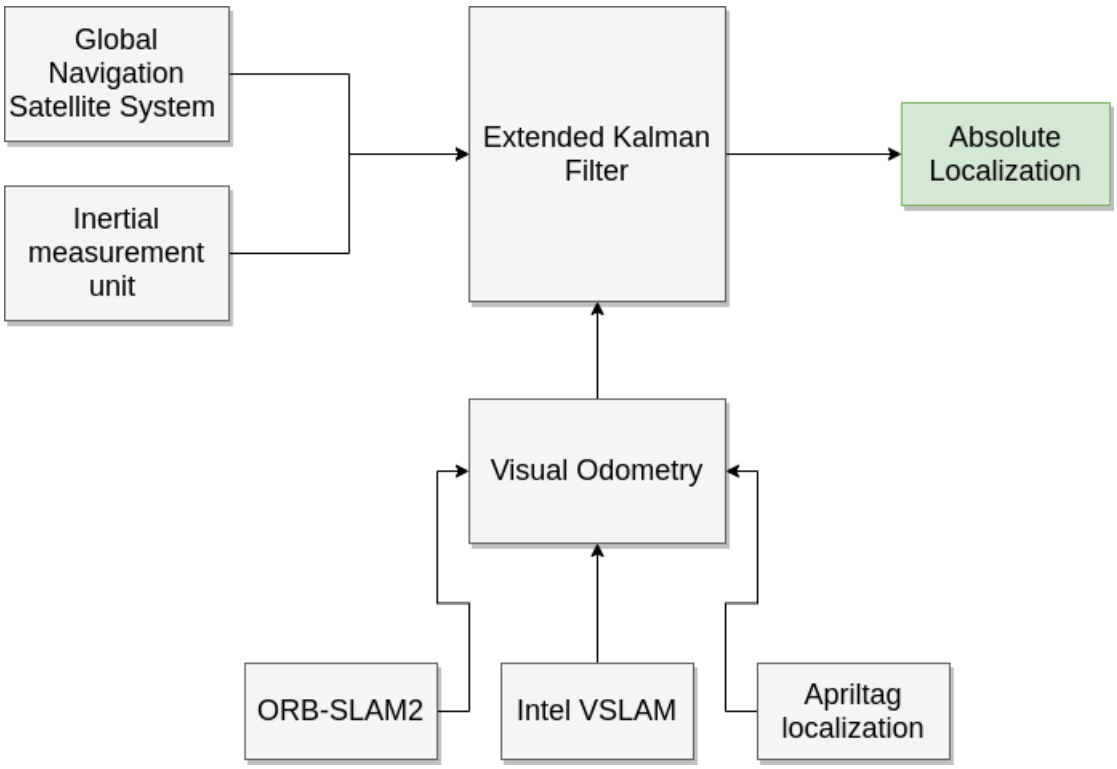


Рис. 7: Подход к определению абсолютного местоположения

3.2.2. Дерево преобразований

Для вычисления собственного положения в пространстве необходимо понимать несколько вещей: где устройство находится относительно своего предыдущего местоположения и где устройство находится глобально относительно карты, которую оно построило. Так же необходимо понимать взаимосвязь между этими передвижениями. В платформе ROS граф таких взаимодействий называется деревом преобразований (transformation tree). В случае платформы с описанными устройствами это дерево будет выглядеть следующим образом:

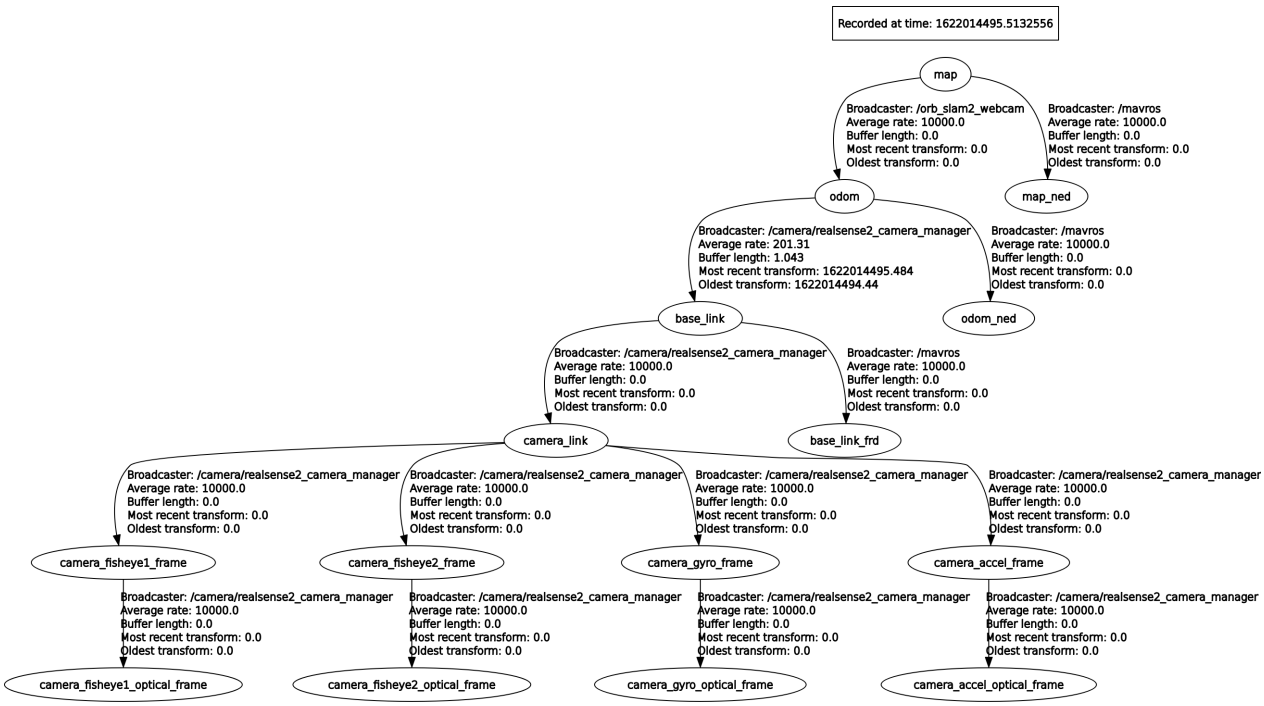


Рис. 8: Дерево преобразований

Узел **map** на данной карте представляет собой глобальную карту в пространстве которой находится рассматриваемое устройство (дрон).

Отношение узлов **map->odom** является представлением локализации устройства на карте. Логично что при такой локализации могут быть “разрывы” и “прыжки”, что как раз является допустимым от кадра к кадру. Такое отношение высчитывается с помощью различных SLAM алгоритмов и алгоритмов детекции закрытия цикла.

Отношение узлов **odom->base_link** являет собой представление отношения положения устройства (его позу) относительно него же в предыдущий момент измерения. В данном отношении не допускается “прыжков” и “разрывов” и измерения должны быть как можно более частыми. Данное отношение можно получит с помощью инерциальных измерительных устройств либо с помощью того же алгоритма SLAM.

Соответственно получение отношения **map->odom->base_link** является целевым при решении задачи визуальной одометрии так как помимо положения на карте необходимо так же знать позу устройства на этой карте.

3.3. Программная составляющая платформы

В данной секции будет рассмотрена архитектура программной части платформы для реализации и тестирования алгоритмов для БПЛА.

Для аппаратной части платформы важны следующие аспекты:

- Расширяемость
- Переносимость
- Модульность

Было принято решение использовать в качестве основного подхода сочетание программного обеспечения для полётных контроллеров Ardupilot и операционной системы для роботов ROS с применением MAVLINK в качестве протокола коммуникации.

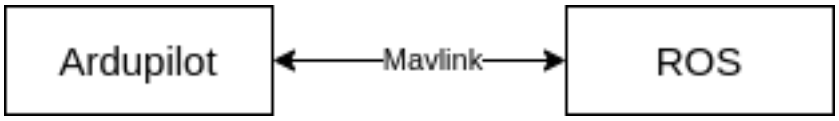


Рис. 9: Ardupilot и ROS

Таким образом в элементарном представлении существуют два устройства, одно из которых отвечает за управление дроном и осуществление связи с наземной системой отслеживания, (в диаграмме аппаратной реализации оно называется pixhawk) и работает с использованием программного пакета Ardupilot. Другое устройство отвечает за сбор и анализ данных, а так же принятие решений (в диаграмме аппаратной реализации оно называется “Offboard”) и работает с использованием ROS.

Преимущество данного подхода заключается в независимости программных компонент, также в возможности программной симуляции поведения аппаратных компонент.

Важным компонентом данной программной составляющей является работа устройства под управлением ROS. Так как это устройство работает на ROS, то основными её частями будут выступать узлы (Node),

которые работают независимо друг от друга и общаются путём отправки и получения сообщений из топиков (topic). Необходимо выделить следующие узлы:

1. Control Node
2. Telemetry Node
3. Localization Node
4. SLAM Node
5. Realsense Node
6. Apriltag Node

Узел **1** отвечает за получение обработанных данных, принятие решений и генерацию управляющих команд. Он получает данные от узла **2** и **3**.

Узел **2** отвечает за получение данных от pixhawk. Он отправляет данные о текущем положении и значении сенсоров imu в узел **3**, а оставшиеся данные, необходимые для принятия решений, отправляет в узел **1**.

Узел **3** отвечает за реализацию фильтра Калмана, который получает данные из узлов **4**, **5**, **6**.

Узел **4** отвечает за реализацию SLAM алгоритма для вычисления собственного положения. Данный узел получает данные из узла **5**, которые представляют из себя поток изображений.

Узел **5** отвечает за реализацию получения данных с realsense камеры. Эти данные представляют из себя поток данных с камер, поток данных с imu сенсора, поток данных с asic'a, который является решением задачи SLAM. Эти данные передаются в узлы **6**, **4**, **3**.

Узел **6** отвечает за реализацию алгоритма алгоритма вычисления собственного положения относительно меток Apriltag. (На данный момент не реализован)

Данные о положении передаются в узел **3**. Архитектура узлов и их взаимодействия представлена на рис.10

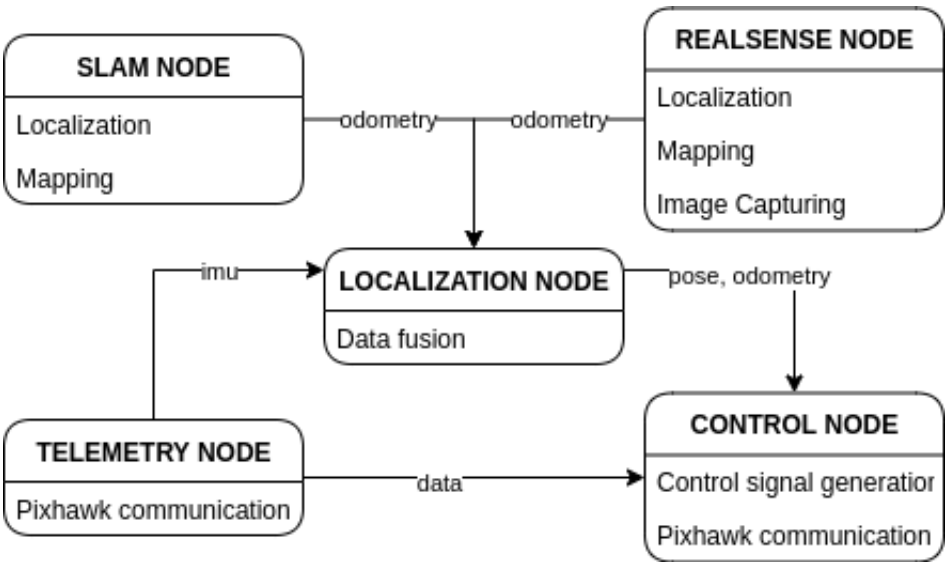


Рис. 10: Архитектура решения на базе ROS

3.3.1. Симуляция

Испытание программно-аппаратной платформы (а именно самого собранного БПЛА) - действие расточительное и не всегда обоснованное. В связи с чем было принято решение перенести как можно больше компонент в симуляцию, для того чтобы проводить испытания не боясь сломать физический прототип.

Для этих целей хорошо подходит симулятор gazebo [11]. К его преимуществам можно отнести простоту настройки, плотную интеграцию с ROS и Ardupilot, а также обильную поддержку сообщества разработчиков. Для тестирования в виртуальном окружении нужны следующие компоненты:

1. Механизм для симуляции поведения полётного контроллера
2. ПО для отправки для отправки команд и получения данных с полётного контроллера
3. Система “контроля с земли”
4. Физически корректный симулятор

Данные компоненты описываются на следующей схеме 11.

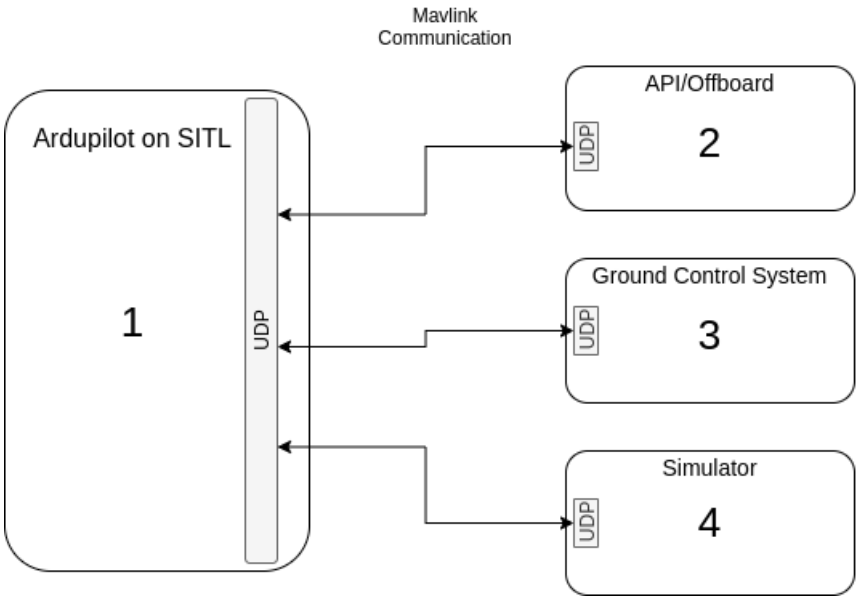


Рис. 11: Архитектура компонент симуляции

Как видно из схемы выше, внутри системы компоненты симуляции взаимодействуют посредством протокола Mavlink поверх транспортного протокола UDP. Данный подход удобен тем, что протокол Mavlink так же используется при передачи данных через интерфейс UART [20] при взаимодействии с БПЛА напрямую.

В рамках симуляции необходимо было максимально приблизиться к физической реализации БПЛА. Данная цель была достигнута, в частности использовалась идентичная модель камеры “intel realsense t265” и близкая реализация модели дрона “gazebo iris”. Фото и видео с симуляции предоставлено в приложении 2.

3.3.2. Итог

В рамках работы над программным компонентом было реализовано решение, позволяющее проводить тестирование и апробацию подходов к автономной навигации как в симуляции, так и на реальном устройстве. Ссылка на репозиторий в приложении 3. Данное решение поддерживает работу со следующими компонентами:

- Камера intel realsense t265
- Intel V-SLAM

- ORB-SLAM2
- Kimera-VIO

А так же является полностью совместимым с ROS.

3.4. Аппаратная платформа для испытаний

В качестве платформы для испытаний были выбраны следующие компоненты:

Платформа	F450
Полётный контроллер	pixhawk 2.4.8
Моторы	Emax MT-2216 KV810
Батарея	4S 5000mAh
Нагрузка	jetson nano, intel realsense t265

Данные компоненты позволили достичь необходимых лётных характеристик, сохраняя относительную дешевизну и прочность конструкции.

3.4.1. Архитектура

В ходе исследований была спроектирована и реализована следующая оптимальная архитектура для платформы.

Одним из наиболее важных компонентов платформы является полётный контроллер pixhawk 2.4.8, который позволяет выполнять все возложенные на него функции при относительной дешевизне. В функции данного контроллера входят:

- Получение управляющего сигнала по радио каналу и/или USB
- Отправка телеметрических данных по радио каналу и/или USB
- Получение данных от GPS модуля
- Отправка аналогового сигнала на контроллеры каждого из моторов

- Получение информации о состоянии батареи

Pixhawk позволяет поддерживать непрерывное общение одновременно с несколькими устройствами по разным протоколам передачи данных. Таким образом можно одновременно осуществлять управление по USB, а так же отслеживать его состояние на базовой станции контроля и переопределять управляемый сигнал “на лету”, при возникновении проблем.

Модулем, отвечающим за обработку информации и принятие решений является Nvidia jetson Nano. Он выполняет следующие функции:

- Получение данных с сенсоров, которые не подключены к pixhawk
- Получение данных с pixhawk
- Анализ полученных данных
- Принятие решения о поведении дрона
- Отправка управляющего сигнала на pixhawk

Jetson Nano позволяет в режиме реального времени анализировать ситуацию и принимать соответствующие решения. Данный факт крайне важен в таких задачах, как автономная навигация БПЛА.

Архитектура подхода и взаимодействие компонентов проиллюстрированы на рис. 12

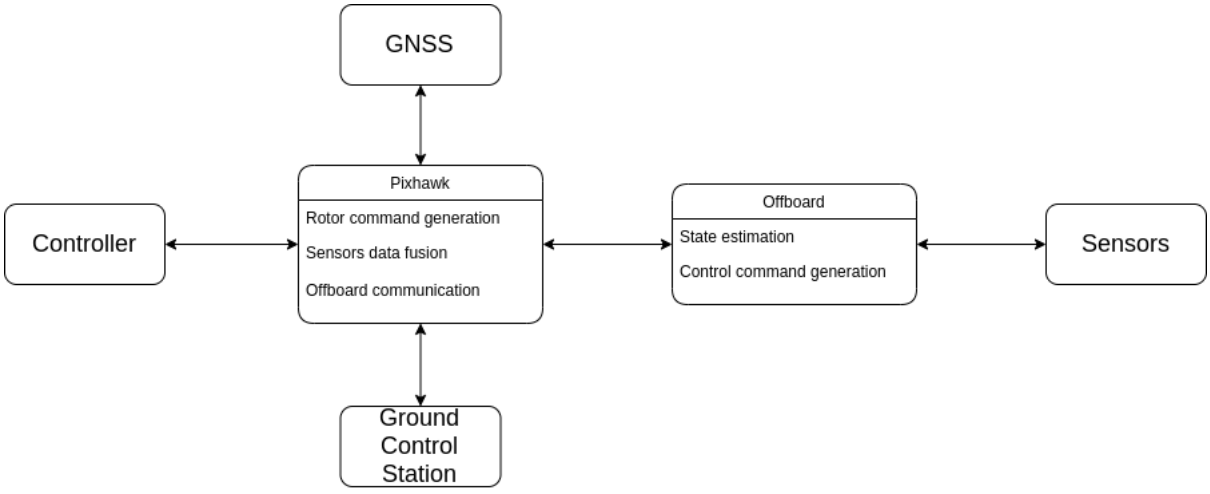


Рис. 12: Архитектура взаимодействия аппаратных модулей

3.4.2. Реализация

Платформа была собрана и успешно протестирована. Пример сборки показан на рис. 13 Большее количество фотографий представлено в

Рис. 13: Аппаратная платформа

приложении 4.

3.4.3. Итог

Данная платформа позволяет переносить до 900 гр. полезной нагрузки. Также ввиду использования продвинутого компьютера-компаньона, находящегося на борту, обеспечивается минимальная задержка между командами, а так же возможность исполнения произвольного ресурсоёмкого кода.

Заключение

В ходе данной работы были достигнуты результаты:

- Проведён анализ предметной области
- Проведён анализ составных компонентов
- Сформулирован подход для реализации программно-аппаратной платформы
- Разработана архитектура платформы
- Реализована платформа
- Проведена апробация платформы

Приложения

Приложение 1

Был протестирован модуль intel realsense t265 вместе с вычислительной платформой Jetson Nano.

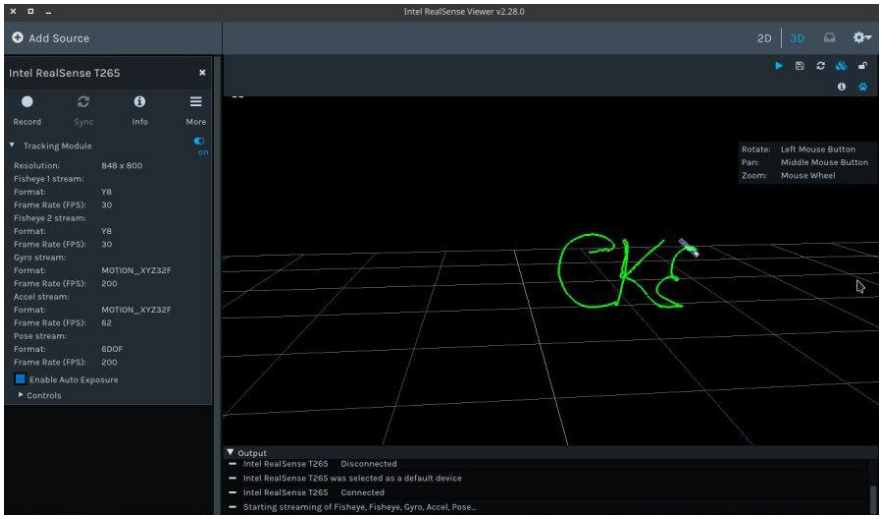


Рис. 14: Эксперимент с модулем intel realsense t265

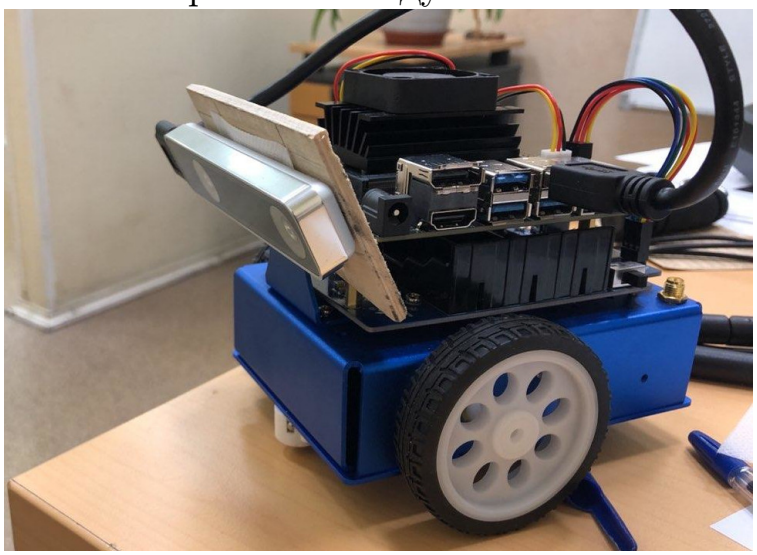


Рис. 15: Вычислительная платформа Jetson Nano с модулем intel realsense t265 на базе автономного робота JetBot

В ходе экспериментов было установлено, что модуль t265 обеспечивает уверенное качество локализации при хороших условиях освещённости и при относительно небольших скоростях передвижения. На (рис. 14) можно увидеть путь, который совершило устройство (бук-

ва “з” не в ту сторону, да). Это позволяет судить о так называемой “offline” реализации метода SLAM, отличие которой заключается в том, что помимо определения текущего положения на карте, устройство также хранит весь пройденный путь. Однако реализация SLAM на данном модуле показывают большую погрешность при возвращении в точку, из которой было начато движение. Так же было выяснено, что SLAM работает даже при полностью заблокированной камере стереопары, что наталкивает на мысль о том, что был реализован монокулярный подход, либо же какая-то его модификация.

В ходе экспериментов с платформой jetson nano, было выяснено, что она позволяет без труда обрабатывать данные, поступающие с сенсорного модуля. Это вселяет уверенность в правильном выборе платформы.

Приложение 2: Программная реализация: Симуляция

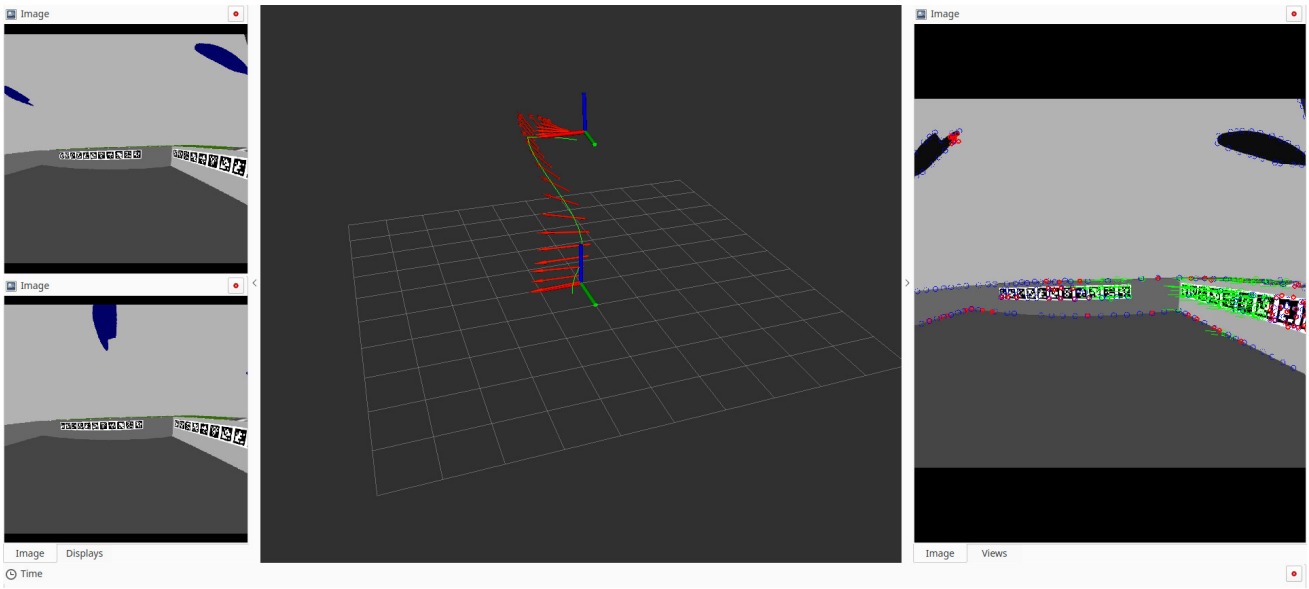


Рис. 16: Симуляция положения БПЛА в пространстве

В ходе реализации решения была получена траектория движения летательного аппарата в симулируемой среде без использования GNSS. Так

же в каждой оцениваемой точке данной траектории есть возможность оценить ориентацию БПЛА в пространстве по 6 степеням свободы. На изображении, представленном справа выделены ключевые точки, используемые для реализации алгоритма SLAM. На центральном изображении показан путь, проделанный БПЛА с указанием начальной и текущей позиции и ориентации, а так же промежуточных состояний. На левых изображениях представлены видеопотоки с двух сонаправленных камер симулированного сенсора intel realsense t265.

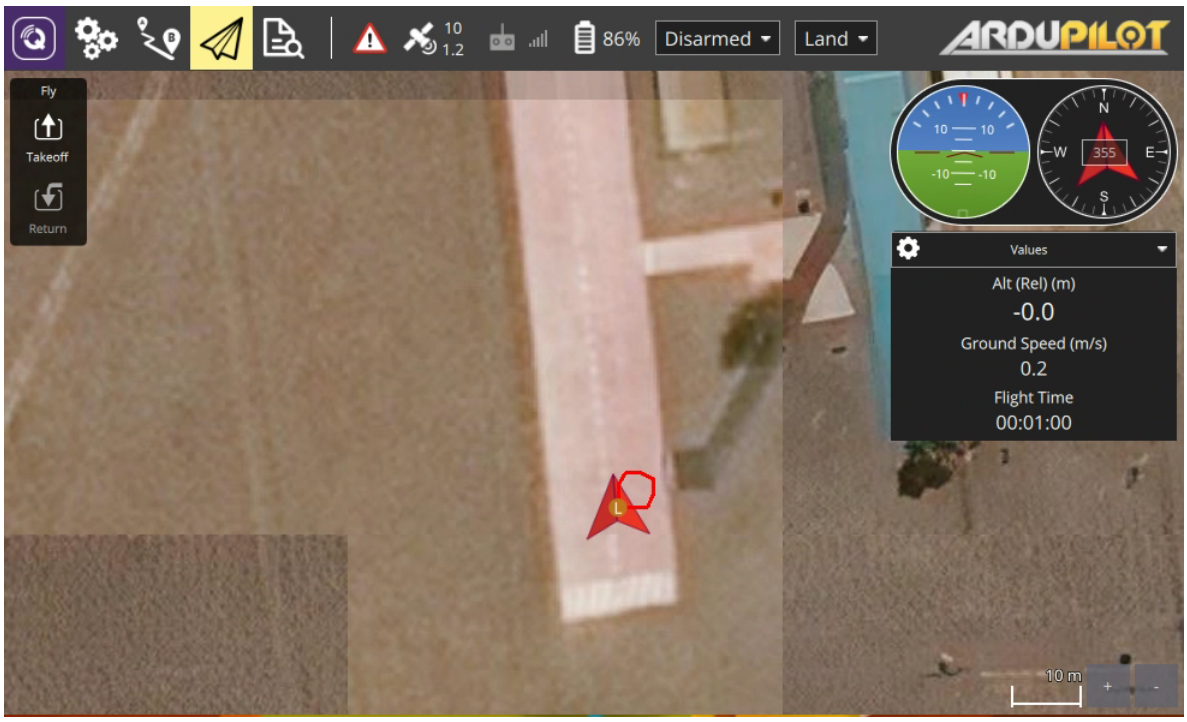


Рис. 17: Отчёт о задаче БПЛА в QGroundControl

Видео: БПЛА, Симуляция

Приложение 3: Программная реализация, репозиторий

Ссылка на репозиторий

Приложение 4: Аппаратная реализация

Рис. 18: Сборка БПЛА

Рис. 19: Тестирование БПЛА в помещении

Рис. 20: Тестирование БПЛА на улице

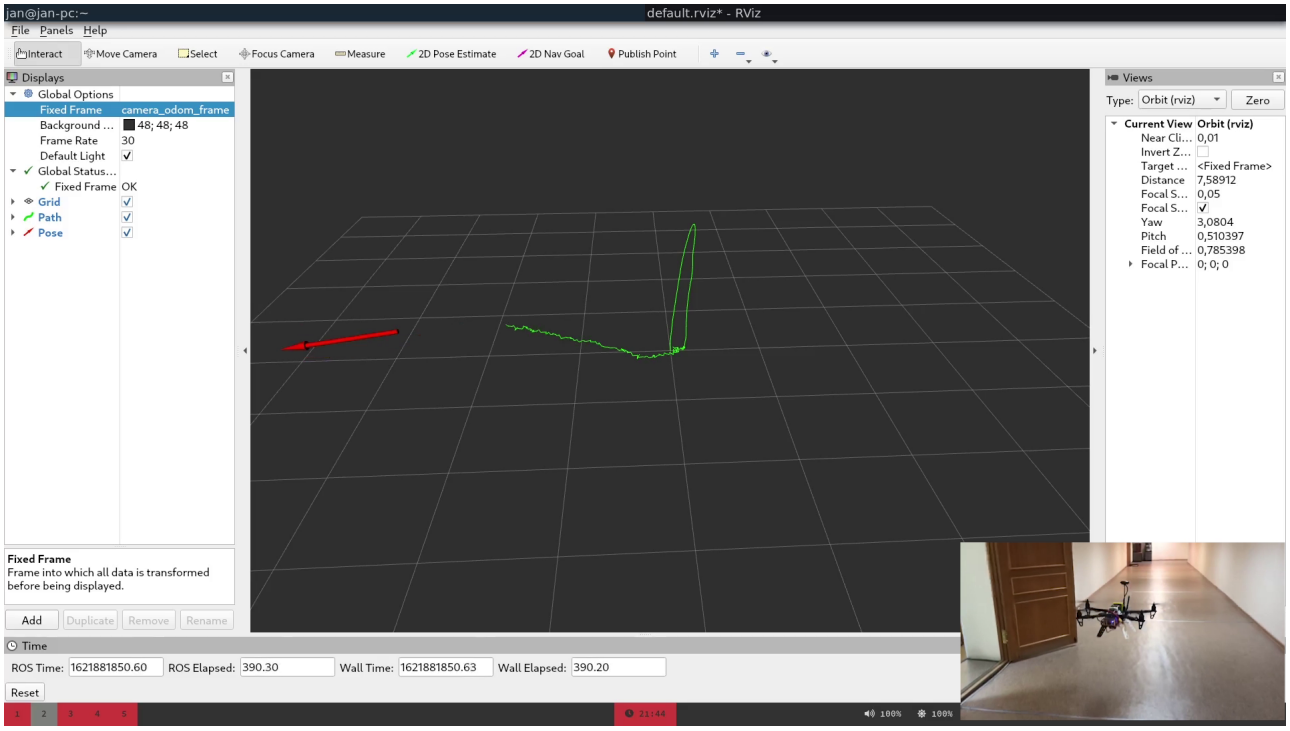
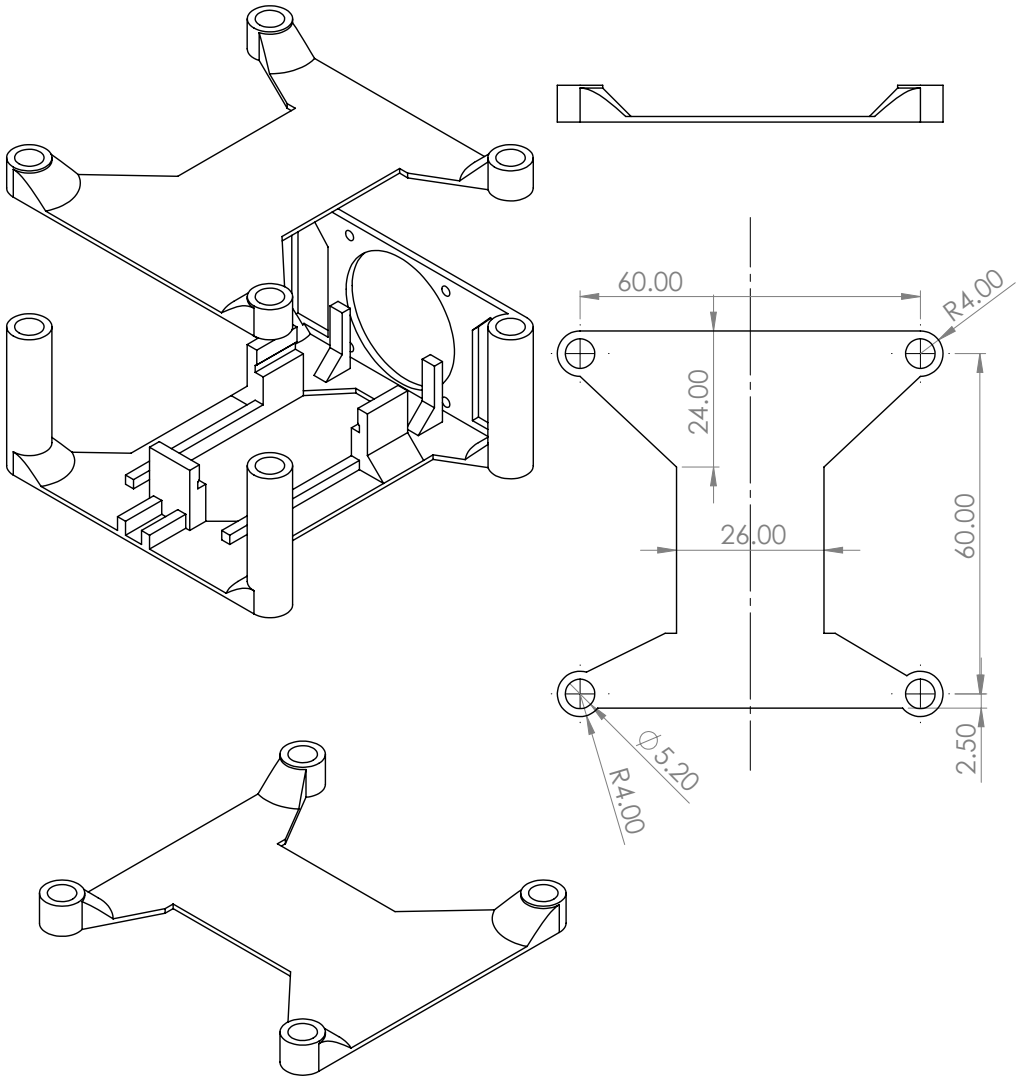


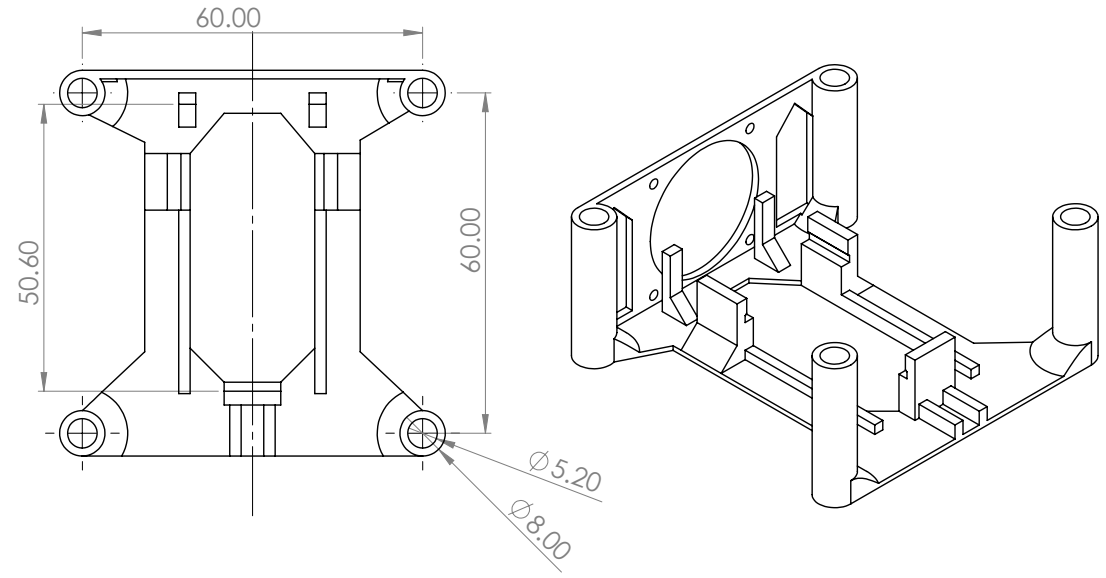
Рис. 21: Тестирование автономного пролёта БПЛА в помещении

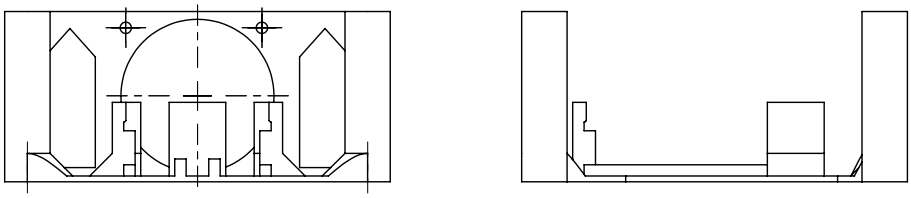
Видео: БПЛА, полёт в помещении

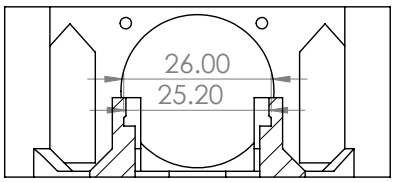
Приложение 5: Аппаратная реализация: чертежи

Чертёж: крепление для модуля стабилизатора питания









SECTION A-A

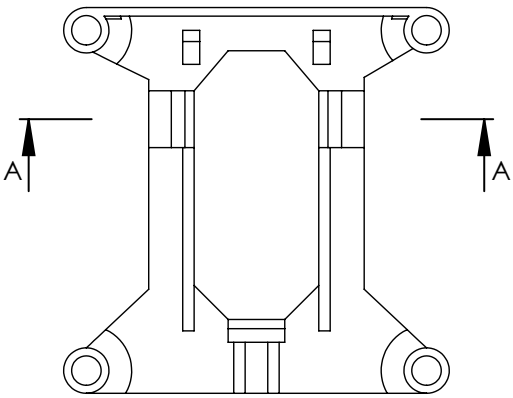


Чертёж: стойки для рамы платформы

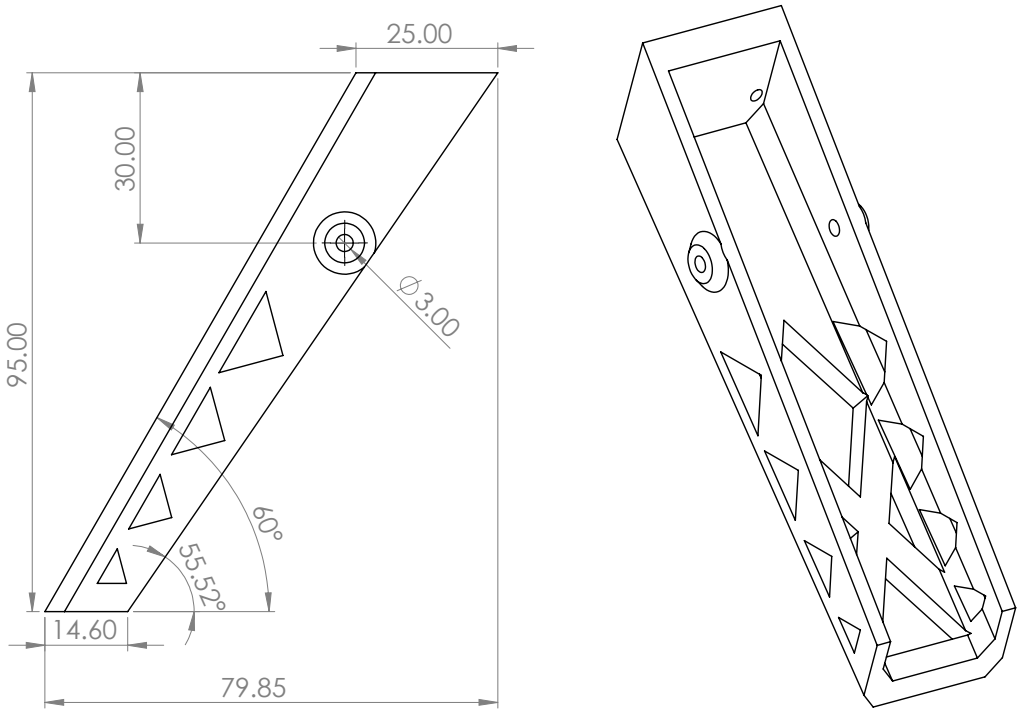
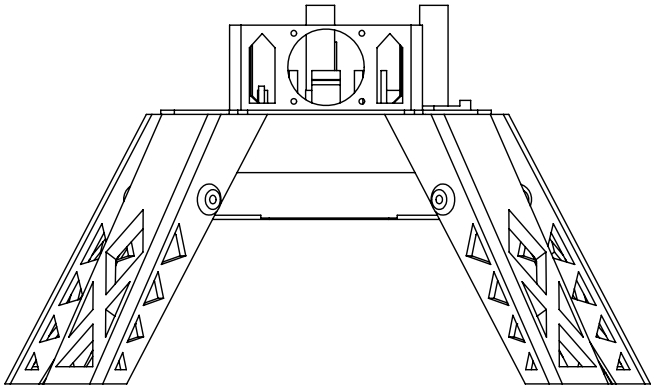
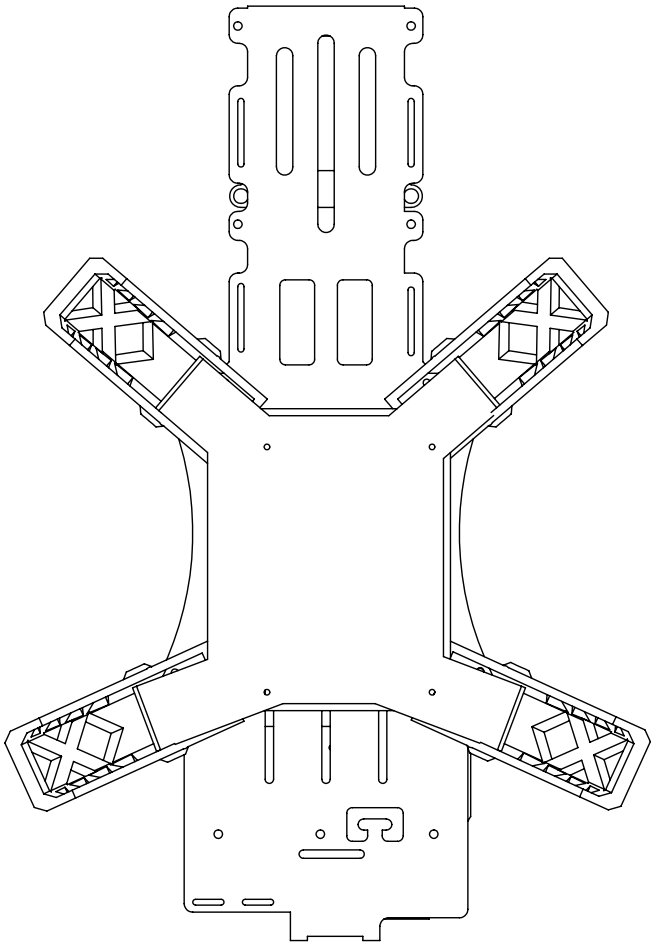
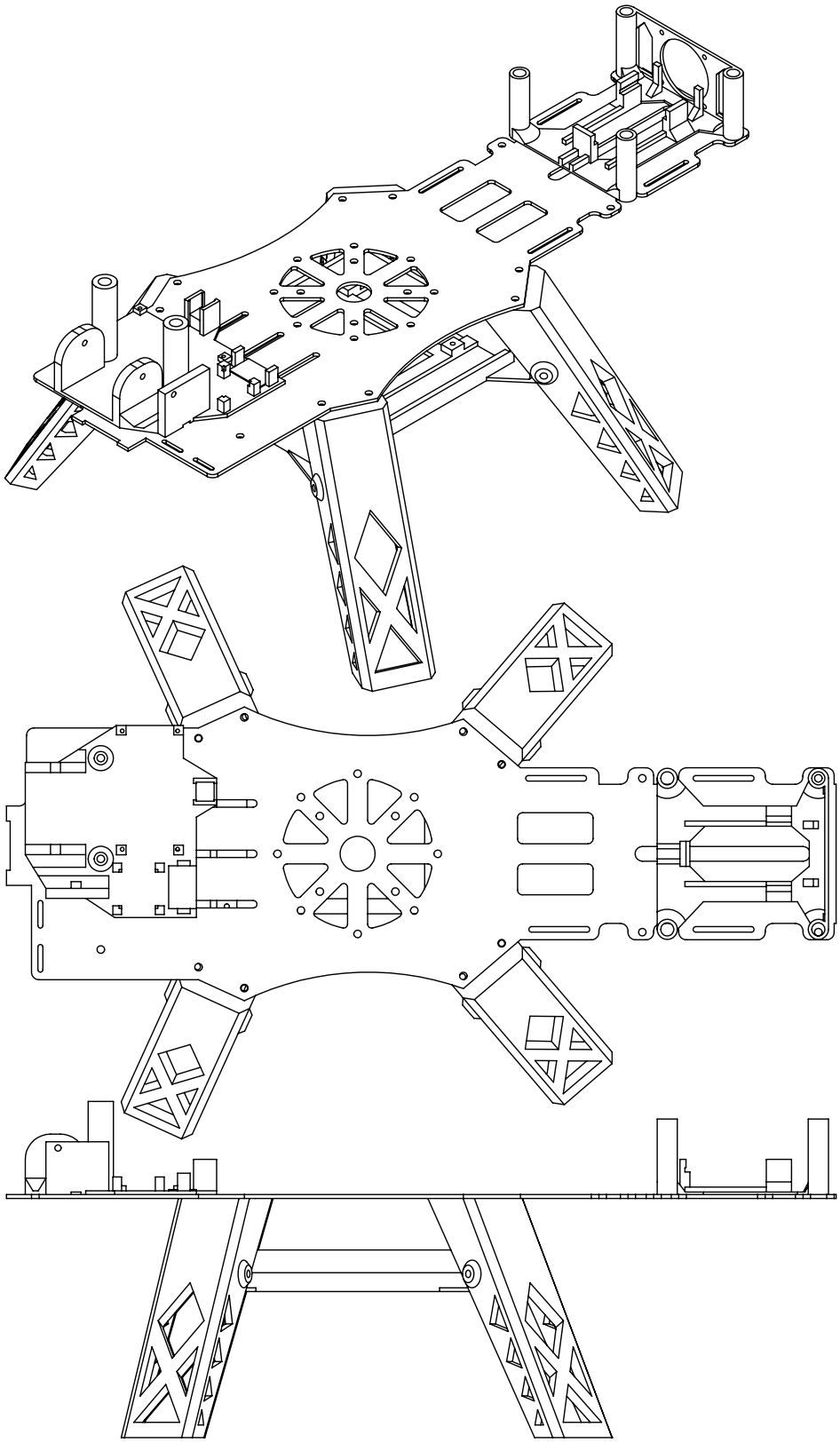


Чертёж: рама со стойками и и креплениями







Список литературы

[1] Atcheson Bradley, Heide Felix, Heidrich Wolfgang. CALTag: High Precision Fiducial Markers for Camera Calibration. — 2010. — 01. — P. 41–48.

[2] Build Your Own Visual-Inertial Drone: A Cost-Effective and Open-Source Autonomous Drone / Inkyu Sa, Mina Kamel, Michael Burri et al. // IEEE Robotics Automation Magazine. — 2018. — Mar. — Vol. 25, no. 1. — P. 89–103. — URL: http://dx.doi.org/10.1109/MRA.2017.2771326.

[3] Comparative Study of a Commercial Tracking Camera and ORB-SLAM2 for Person Localization / Safa Ouerghi, Nicolas Ragot, Rémi Boutteau, Xavier Savatier. — 2020. — 01. — P. 357–364.

[4] Durrant-Whyte H., Bailey T. Simultaneous localization and mapping: part I // IEEE Robotics Automation Magazine. — 2006. — Vol. 13, no. 2. — P. 99–110.

[5] Engel Jakob, Schöps Thomas, Cremers Daniel. LSD-SLAM: Large-Scale Direct Monocular SLAM // Computer Vision – ECCV 2014 / Ed. by David Fleet, Tomas Pajdla, Bernt Schiele, Tinne Tuytelaars. — Cham : Springer International Publishing, 2014. — P. 834–849.

[6] Fiala Mark. ARTag, An Improved Marker System Based on AR-Toolkit. — 2004. — 01.

[7] GNU General Public License, version 3. — http://www.gnu.org/licenses/gpl.html. — 2007. — June. — Last retrieved 2020-01-01.

[8] Inside the Virtual Robotics Challenge: Simulating Real-Time Robotic Disaster Response / C.E. Aguero, N. Koenig, I. Chen et al. // Automation Science and Engineering, IEEE Transactions on. — 2015. — April. — Vol. 12, no. 2. — P. 494–506.

[9] Johnston Gary, Riddell Anna, Hausler Grant. The international GNSS service // Springer handbook of global navigation satellite systems. — Springer, 2017. — P. 967–982.

[10] Kimera: an Open-Source Library for Real-Time Metric-Semantic Localization and Mapping / Antoni Rosinol, Marcus Abate, Yun Chang, Luca Carlone // IEEE Intl. Conf. on Robotics and Automation (ICRA). — 2020. — URL: https://github.com/MIT-SPARK/Kimera.

[11] Koenig Nathan, Howard Andrew. Design and Use Paradigms for Gazebo, An Open-Source Multi-Robot Simulator // IEEE/RSJ International Conference on Intelligent Robots and Systems. — Sendai, Japan, 2004. — Sep. — P. 2149–2154.

[12] Mathias H. David. An Autonomous Drone Platform for Student Research Projects // J. Comput. Sci. Coll. — 2016. — . — Vol. 31, no. 5. — P. 12–20.

[13] Meier Lorenz, Honegger Dominik, Pollefeys Marc. PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms // 2015 IEEE International Conference on Robotics and Automation (ICRA). — 2015. — P. 6235–6240.

[14] Mur-Artal Raúl Montiel J. M. M., Tardós Juan D. ORB-SLAM: a Versatile and Accurate Monocular SLAM System // IEEE Transactions on Robotics. — 2015. — Vol. 31, no. 5. — P. 1147–1163.

[15] Mur-Artal Raúl, Tardós Juan D. ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras // IEEE Transactions on Robotics. — 2017. — Vol. 33, no. 5. — P. 1255–1262.

[16] Nvidia. Nvidia autonomous machines // Nvidia autonomous machines. — 2020. — URL: https://www.nvidia.com/ru-ru/autonomous-machines/jetson-store/.

[17] ROS: an open-source Robot Operating System / Morgan Quigley, Ken Conley, Brian P. Gerkey et al. // ICRA Workshop on Open Source Software. — 2009.

[18] Shepard D.P., Bhatti Jahshan, Humphreys T.E. Drone Hack: Spoofing Attack Demonstration on a Civilian Unmanned Aerial Vehicle // GPS World. — 2012. — 08. — Vol. 23. — P. 30–33.

[19] Shi Jianbo, Tomasi Carlo. Good Features to Track. — 1994. — P. 593–600.

[20] Uart Project // VHDL Coding Styles and Methodologies. — Boston, MA : Springer US, 1999. — P. 353–381. — ISBN: 978-0-306-47681-5. — URL: https://doi.org/10.1007/0-306-47681-9_12.

[21] Vision meets Robotics: The KITTI Dataset / Andreas Geiger, Philip Lenz, Christoph Stiller, Raquel Urtasun // International Journal of Robotics Research (IJRR). — 2013.

[22] Visual odometry for autonomous outdoor flight of a quadrotor UAV / Hugo Romero, Sergio Salazar, Omar Santos Sánchez, R. Lozano. — 2013. — 05. — P. 678–684.

[23] Wang John, Olson Edwin. AprilTag 2: Efficient and robust fiducial detection // Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). — 2016. — October.

[24] An Introduction to the Kalman Filter : Rep. ; Executor: Greg Welch, Gary Bishop. — USA : 1995.

[25] intel. Intel Realsense Technology // Intel Realsense Technology. — 2020. — URL: https://www.intel.ru/content/www/ru/ru/architecture-and-technology/realsense-overview.html.

[26] yves Bouguet Jean. Pyramidal implementation of the Lucas Kanade feature tracker // Intel Corporation, Microprocessor Research Labs. — 2000.