МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»

Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

РАБОТА ПРОВЕРЕНА
Рецензент
_____
«___»_____ 2019 г.

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой ЭВМ
_____ Г.И. Радченко
«___»_____ 2019 г.

Разработка мобильного робототехнического комплекса для создания 3D карт замкнутых пространств

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

Руководитель работы,
к.т.н., доцент каф. ЭВМ
_____ П.О. Шабуров
«___»_____2019 г.

Автор работы,
студент группы КЭ-452
_____ О.И. Морозов
«___»_____2019 г.

Нормоконтролёр,
ст. преп. каф. ЭВМ
_____ В.В. Лурье
«___»_____2019 г.

Челябинск-2019

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

УТВЕРЖДАЮ
Заведующий кафедрой ЭВМ
_____ Г.И. Радченко
«___»_____2019 г.

ЗАДАНИЕ

на выпускную квалификационную работу бакалавра

студенту группы КЭ-452
Морозов Олег Иванович
обучающемуся по направлению
09.03.01 «Информатика и вычислительная техника»

1. **Тема работы: «**Разработка мобильного робототехнического комплекса для создания 3D карт замкнутых пространств» утверждена приказом по университету от 25 января 2019г. №899

2. **Срок сдачи студентом законченной работы:** 6 июня 2019 г.

3. **Исходные данные к работе:**

Спроектировать робототехнический комплекс с возможностью:

1. Удалённый осмотр местности.
2. Лазерное сканирование пространства.
3. Управление манипулятором.
4. Навигация под землёй.
5. Высокая дальность управления.
6. Передача результатов анализа среды.

7. Автоматическое управление при потере сигнала.

4. **Перечень подлежащих разработке вопросов:**
    - анализ существующих аналогов по тематике работы;
    - детализация набора требований к приложению;
    - выбор средств реализации;
    - создание аппаратной и программной частей комплекса;
    - тестирование разработанного программного обеспечения.

5. **Дата выдачи задания:** 1 декабря 2018 г.

Руководитель работы _____/ *П.О. Шабуров* /

Студент _____/ *О.И. Морозов* /

КАЛЕНДАРНЫЙ ПЛАН

Этап	Срок сдачи	Подпись руководителя
Введение и обзор литературы	01.03.2019	
Разработка модели, проектирование	01.04.2019	
Реализация системы	01.05.2019	
Тестирование, отладка, эксперименты	15.05.2019	
Компоновка текста работы и сдача на нормоконтроль	24.05.2019	
Приемка приложения при комиссии высшего учебного заведения	6.06.2019	

Руководитель работы _____/*П.О. Шабуров*/

Студент _____/*О.И. Морозов*/

Аннотация

О.И. Морозов. Разработка мобильного робототехнического комплекса для создания 3D карт замкнутых пространств. – Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ВШЭКН; 2019, 87 с., 35 ил., библиогр. список – 20 наим.

В рамках выпускной квалификационной работы производится детальный анализ современных исследовательских робототехнических комплексов. Организуется разработка мобильного робототехнического комплекса для создания 3D карт замкнутых пространств. Производится выборка аппаратных решений и средств разработки программного обеспечения. Созданы прототип аппаратного комплекса и программное обеспечение, Производится анализ результатов работы системы, в домене специально разработанных задач.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Роботы значительно упростили жизнь в современном мире. Роботов в основном применяют в медицине (роботизированная хирургия, бионические протезы), космосе, системах безопасности, производстве (автоматизированные промышленные роботы), быту (робот-пылесос, робот-газонокосильщик, умный дом), развлечениях (детские игрушки), образование[1].

Роботы выполняют интеллектуальную, ловкую, силовую работу. Три цели создания робота:

- упрощение или замена труда, выполняемого человеком;
- решение задач, не посильных человеку;
- охрана здоровья человека.

Часто геологам, археологам, шахтёрам и спасательным службам необходимо исследовать место, не доступное для прямого осмотра или анализа по различным причинам:

- узость пространства;
- вероятность обвала;
- заражённость места.

Такого рода роботы должны выполнять роль исследователя недоступных мест, а значит, что он должен обладать хорошей проходимостью. То есть иметь:

- малые габариты;
- удобный способ передвижения;
- удароустойчивость;
- влагозащищённость;
- устойчисость к переворотам;
- продолжительную автономность;
- обратную связь из данных робота.

Для определения состояния ситуации на растоянии используются показания датчиков и видеосвязь. Также для необходимых действый с небольшими объектами рнужно иметь манипуятор. Он позволит роботу извлечь предмет из опасной среды или доставить предметы первой необходимости.

Есть необходимость построение трёхмерных моделей местности для определиения любых геометрических параметров рельефа – растояний, высот, объёмов и т.п. Датчики для измерения расстояния могут использовать различные принципы измерений: индуктивный, ультразвуковой или оптический. Датчики с рассеянным отражением и аналоговым выходом могут измерять расстояния в широких пределах. Оптические датчики радарного типа, преимущественно лазерные, могут измерять большие расстояния[2].

Лидар (лазерный радар) – это оптический датчик, приняемый ряде систем активной безопасности транспортных средств[3]. Датчики предоставляют компьютеру трёхмерное облако точек, обозначающее окружающее пространство.

Целью выпускной квалификационной работы является разработка мобильной системы, управляемой дистанционно и передающей видео и данные для построения 3D модели окружающей местности.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. ОБЗОР АНАЛОГОВ

На сегодняшний день уже существуют роботы исследователи [4], имеющие данное решение. Но не все из них удовлетворяют по разному ряду причин.

1.1.1. Наземный робототехнический комплекс “TurtleRover”[5].

Достоинства:

- возможность просмотра местности;
- возможность сканирования местности;
- не высокая цена, около $1 400.

Недостатки:

- имеющаяся система геопозиционирования не позволяет определять местоположения под землёй;
- дистанция связи через препятствия очень низкая;
- низкая проходимость по неровной местности.

1.1.2. Наземный робототехнический комплекс “Micro Tactical Grounds Robot (MTGR)”[6].

Достоинства:

- возможность просмотра местности;
- средняя дистанция управления;
- возможность сканирования местности;
- возможность работать манипулятором;
- высокая проходимость за счёт четырёх гусеничного шасси.

Недостатки:

- имеющаяся система геопозиционирования не позволяет определить местоположение под землёй;
- высокая цена, более $5 000;
- возможность работать манипулятором.

1.1.3. Летательный аппарат “Autonomous underground drone”[7].

Достоинства:

- навигация в замкнутых пространствах;
- возможность сканирования местности.

Недостатки:

- большие размеры;
- высокая вероятность отказа работы при столкновении с препятствием;
- отсутствие манипулятора;
- высокая цена, более $7 000.

Российские роботы МЧС [8-10] и военные роботы [11, 12] имеют преимущества как скорость обработки информации, мобильность и высокую проходимость, но имеют недостатки – малая скорость передвижения и большие габариты. Данные недостатки критичны для подземных условий. Для прохода на опасную территорию нужен легковесный, малогабаритный робот для разведки. Российское роботостроение только сейчас выходит на данный уровень, поэтому роботы МЧС при малых размерах имеют только обзор местности и действия манипулятором.

Сведём результаты в таблицу.

Таблица 1.1 – Преимущества и недостатки аналогов:

Критерий	Turtle Rover	MTGR	Autonomous underground drone	Российские аналоги
Навигация под землёй	–	–	+	–
ДУ под землёй	–	+	–	+
Обзор местности	+	+	–	+
Зарисовка местности	–	–	+	–
Работа манипулятором	+	+	–	+

Продолжение таблицы 1.1

Критерий	Turtle Rover	MTGR	Autonomous underground drone	Российские аналоги
Стоимость	$1 349	$5 000	$ 7 000	–

1.2. АНАЛИЗ ОСНОВНЫХ ТЕХНОЛОГИЧЕСКИХ РЕШЕНИЙ

Необходимо решить проблему дальности управления. Частота радиоканала прямо пропорциональна скорости передачи данных и обратно пропорциональна дистанции передачи информации. Из анализа передаваемого трафика, нам необходима частота передачи не менее 2.4 ГГц.

В настоящее время наиболее распространенной технологией беспроводного доступа, которая повсеместно применяется для передачи большого количества трафика различного вида, является стандарт беспроводных локальных сетей IEEE 802.11. Одним из самых перспективных направлений развития технологии Wi-Fi стали mesh-сети [13]. Решением проблемы дальности передачи данных будет создание mesh-сети на аккумуляторном питании в которой будет один или несколько Wi-Fi ретрансляторов. На данный момент популярным и дешёвым решением будет сборка Wi-Fi ретранслятора на базе модуля ESP8266 (рисунок 1.2.1).

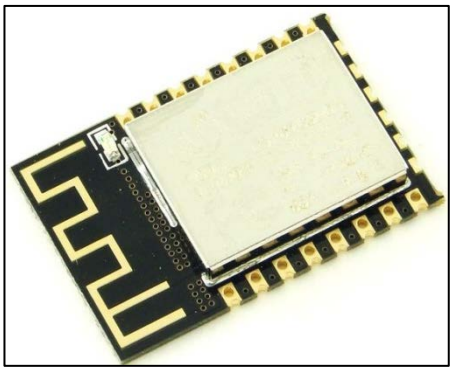


Рисунок 1.2.1 – Микроконтроллер “ESP8266”

Особенности:

- Поддержка беспроводного стандарта 802.11 b/g/n;
- Поддержка 2 режима работы Wi-Fi Direct (P2P), soft-AP;
- Интегрирован стек протокол TCP/IP;
- Выходная мощность в режиме 802.11b: +19.5dBm;
- Поддержка подключения нескольких TCP Client;

Проблема проходимости робота решается установкой четырёхступенчатой гусеницы (рисунок 1.2.2). Две внутренних гусеницы

остаются неподвижными, в то время как две внешних могут вращаться, давая вашему роботу возможность подниматься по низким выступам.

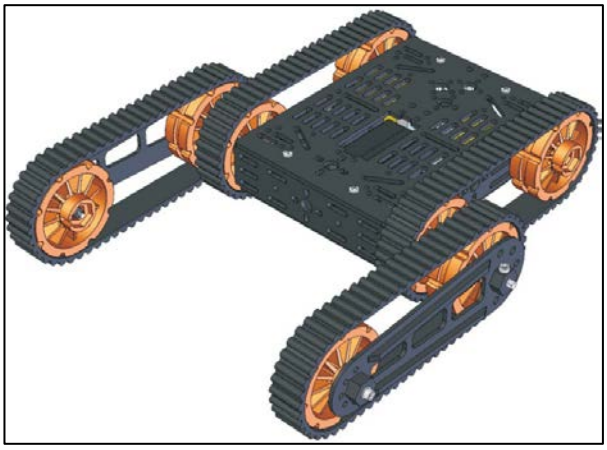


Рисунок 1.2.2 – Четырёхступенчатое гусеничное шасси

Для навигации по заданной траектории, обхода препятствий, что вероятно в бытовых условиях, предлагается выделить три вида навигационных систем: глобальная система, локальная и автономная системы. Задача глобальной системы — определение абсолютных координат, т. е. широты и долготы. Это такие системы как GPS, RTK-GPS, Глонасс, которые используют спутники для позиционирования. Точность таких систем зависит от множества факторов, но в условиях, близких к идеальным наиболее развитая из данных систем, GPS, способна обеспечить точность с ошибкой в пределах 60–90 см [14]. Применение систем глобального позиционирования осложняется их зависимостью от условий использования. Затруднительно или невозможно использовать данные системы внутри зданий, подземных сооружениях и т. д. Таким образом, использование глобальных систем позиционирования имеет смысл при следовании по достаточно длинным маршрутам. В рамках автономной системы навигации применяются гироскопы, цифровые компасы. Это вносит определенные ограничения на их использование. Автономные системы навигации находят применение в условиях, когда передача или прием сигналов извне затруднен или невозможен [15]. Локальные системы используют для позиционирования некоторую точку, обычно стартовую. Данные системы могут применяться на

относительно больших локациях, например, для тактических беспилотных самолетов, работающих в рамках известной территории. В условиях замкнутого пространства целесообразно применение локальной системы позиционирования. В настоящее время наиболее часто применяются системы, использующие дальномеры: лазерные, инфракрасные, ультразвуковые и т. д.

Для позиционирования под землёй подойдёт локальная система навигации можно использованием инерциальной навигационной системы. Simultaneous localization and mapping (SLAM) – метод построения карты в неизвестном пространстве с одновременным контролем текущего местоположения и пройденного пути. Для точного позиционирования в трёхмерном пространстве нужен многоканальный лидар. Однако такие типы лидаров из-за технологии производства имеют высокую стоимость, выше $4000. Альтернативой дорогому датчику послужат 2 одноканальных лидара, зафиксированных в двух разных плоскостях: горизонтальной и вертикальной.

Необходимо подобрать программное обеспечение (ПО) для робота, пульта управления и шлема виртуальной реальности, делающий замеры препятствий в нескольких плоскостях.

Для создания мобильного приложения в рамках выпускной квалификационной работы был выбран смартфон на операционной системе Android. Выбор сделан в связи с наличием на рынке недорогих смартфонов с данной операционной системой. Наиболее популярной средой разработки Android приложений является Android Studio – свободно распространяемая интегрированная среда разработки.

В данном проекте должен производиться замер окружения, предобработка данных, а затем построение точек препятствий в графической среде. Сам робототехнический комплекс может быть исполнен на одноплатном компьютере Raspberry PI 3 B+ (Рисунок 1.2.3). При проведении анализа существующих аналогов было установлено, что эта платформа имеет

более высокие вычислительные возможности, а также хорошо приспособлена к перегреву.

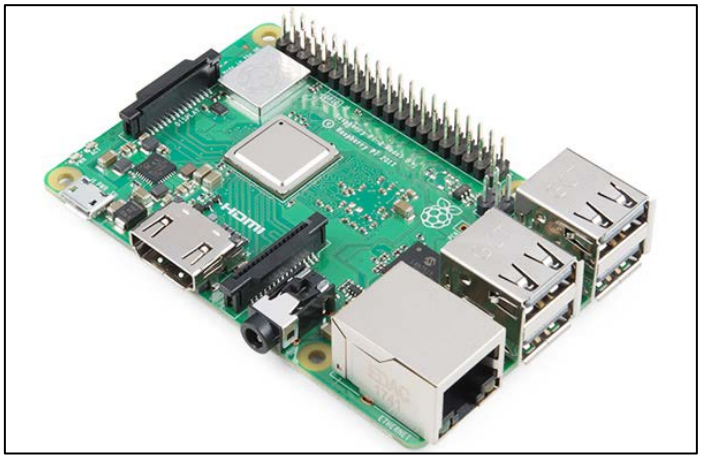


Рисунок 1.2.3 – Компьютер “Raspberry Pi 3 B+”

Программировать поведение GPIO можно на большом количестве различных языков — Python, Ruby, Perl, Java, C, C++ и т.д. Python из-за популярности программирования на Raspberri Pi имеет множество готовых решений для работы с периферийными устройствами через порты ввода вывода общего назначения. Его и возьмём за язык программирования одноплатного компьютера.

Для обеспечения обмена данными между процессами в рамках выпускной квалификационной работы используем интерфейс Socket. Интерфейс обеспечивает взаимодействия между машинами с помощью стека протоколов TCP/IP. Серверным сокетом послужит робот, а клиентским сокетом – ноутбук и VR-шлем.

На ноутбук операционной системой выбран Linux дистрибутив Ubuntu. Данный дистрибутив, в отличии от операционной системы Windows, бесплатный. Популярные языки программирования на Linux дистрибутивах:

- Си;
- C++;
- Java;
- Python;

- JavaScript;
- Shell.

Для большей совместимости с роботом по интерфейсу Socket выбираем язык Python.

Когда вы пишете приложение с помощью Python, вам для этого понадобится использовать GUI (graphical user interface). Существует много вариантов Python GUI. В рамках выпускной квалификационной работы используем для Python инструмент Tkinter. Tkinter – это кроссплатформенная библиотека для разработки графического интерфейса на языке Python. Tkinter является свободным программным обеспечением.

Средство графического отображения карты на ноутбуке выбрано VTK. Это кроссплатформенная библиотека с открытым исходным кодом, предоставляющая разработчикам обширный набор программных инструментов для трехмерной компьютерной графики, обработки изображений и визуализации. VTK поддерживает широкий спектр алгоритмов визуализации, а также передовые методы моделирования. Инструментарий поддерживает параллельную обработку и интегрируется с различными базами данных в графических инструментах GUI, таких как Qt и Tk.

1.3. ВЫВОД

Исследование рынка роботов показало, что ни один из них не является полнофункциональным. Следовательно, создание робототехнического комплекса, свободного от выявленных недостатков, является актуальной задачей. Для ее решения выбраны следующие технологические решения: для усиления сигнала – Wi-Fi ретрансляторы на ESP8266, для улучшения проходимости – установка четырёх гусеничного шасси. Из рассмотренных методов для условий замкнутого пространства подходит использование локальных систем позиционирования. Основным преимуществом таких

методов является высокая точность, а также работа в условиях высокой зашумленности окружающей среды или осутствие спутниковой навигации. Проблема дороговизны световых радаров – замена дорогого лидата на 2 дешёвых с установкой в разнрых плоскостях. Для разработки ПО выбраны языки программирования Pyton и Java. Подобраны бесплатные стредства разработки Android Studio и Geany.

2. ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ

2.1. Составные части аппаратного комплекса

Аппаратный комплекс должен состоять из следующих частей:

– робот;

– пульт управления роботом;

– шлем виртуальной реальности.

2.2. Функциональные требования системы

На рисунке 2.2.1 показана схема взаимодействия объектов системы. Движениями робота управляет компьютер, движением камеры управляет телефон. Робот отправляет координаты препятствий компьютеру, данные камеры – VR-шлему, дополнительные показания датчиков – всем управляющим устройствам.

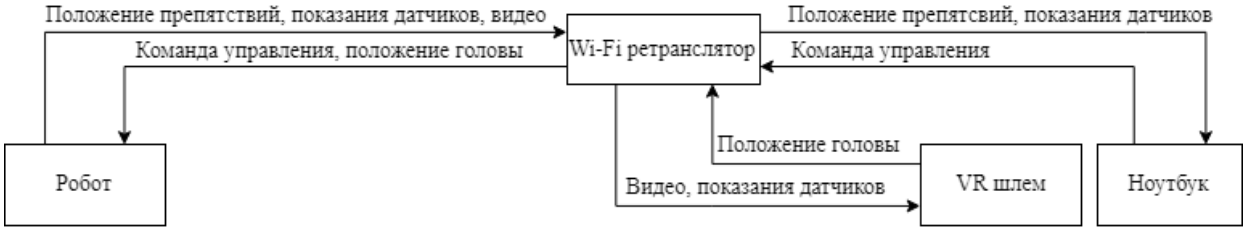


Рисунок 2.2.1 – Схема взаимодействия объектов системы

2.3. Требования к роботу в целом

- размеры устройства: не более 300(Ш)x400(Д)x250(В) мм;
- вес устройства: не более 12Кг;
- максимальная преодолимая высота препятствия: 200 мм;
- камера: не менее 8 Мп;
- защита от воды, огня, кинетических ударов по нижней части, а также пыли, грязи;
- автономность: 4 часов;

- дальность управления под землёй: 200 м;
- WI-FI: 2.4 ГГц
- наличие ёмкости для хранения WI-FI ретрансляторов.
- наличие манипулятора грузоподъёмностью не менее 500 грамм.

2.4. Требования к конструкции робота

В состав робота должны входить такие датчика, как 9-осевой сенсор для стабилизации камеры и инерциальной навигационной системы, датчик движения гусениц, ультразвуковые датчики (Рисунок 2.4.1, позиция 11) для реакции робота на резкие обрывы, датчик заряда аккумуляторной батареи. Для сканирования замкнутых помещений на роботе стоит два датчика Lidar (Рисунок 2.4.1, позиция 10), расположенных горизонтально и вертикально для сканирования соответствующих плоскостей.

Робот будет оборудован стереокамерой (Рисунок 2.4.1, позиция 1), установленной на механический стабилизатор, состоящий из пяти редукторных моторов. Для обеспечения дальности связи робота с точкой управления в ходе движения будут устанавливаться на землю ретрансляторы сигнала (Рисунок 2.4.1, позиция 3), заранее размещённые на роботе. Для лучшей проходимости по неровной местности типом перемещения выбраны гусеницы, состоящие из таких частей:

- основная гусеница (Рисунок 2.4.1, позиция 7);
- вспомогательная гусеница (Рисунок 2.4.1, позиция 8);
- шаговый двигатель в качестве регулятора наклона дополнительной гусеницы (Рисунок 2.4.1, позиция 13);
- редукторный мотор (Рисунок 2.4.1, позиция 12).

Благодаря двум парам гусениц, робот сможет двигаться по очень сложным поверхностям, преодолевать препятствия, которые непреодолимы для обычных спасательных средств.

Также робот снабжён аккумуляторной батарей (Рисунок 2.4.1, позиция 14).

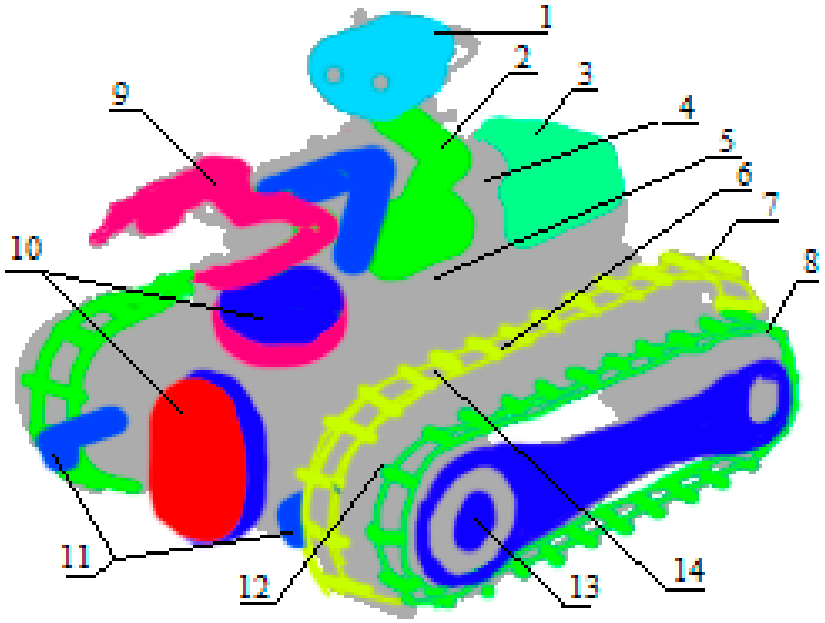


Рисунок 2.4.1 – Предполагаемая конструкция робота

2.5. Требования к пульту управления и VR-шлему

Пульт управления роботом представляет собой переносной персональный компьютер (ноутбук, нетбук). Пульт связывается с роботом, подавая ему команды и данные шлема виртуальной реальности (далее VR-шлем).

Минимальные технические характелистики:

- процессор:
  - архитектура: x86, x64;
  - частота: от 1 Гц;
- ОЗУ: от 4 Гб;
- ПЗУ: от 20 Гб;
- ОС: ОС семейства Linux.

В качестве недорогого устройства за аналог VR-шлема выступит портативный компьютер (смартфон). Смартфон будет соединён с роботом для получения видео со стереокамеры робота и показаний выбранных

датчиков. VR-шлем должен отправлять координаты поворота головы роботу для выбора напрвления осмотра.

Минимальные технические характелистики:

- ОС: не ниже Android 4.4;
- ОЗУ: не ниже 2 Гб;
- диагональ: 5.5 Дюйма;
- разрешение: 1920x1080.

2.6. Требования к программной части

2.6.1. Требования к управлению роботом:

- возможность двигаться вперёд/назад;
- возможность поворачиваться влево/вправо на месте/в движении;
- возможность поворачивать камеру не менее, чем на 200 градусов по горизонтали, 120 градусов по вертикали;
- возможность двигать манипулятором, хватать предметы;
- возможность управлять дополнительной гусеницей.

2.6.2. Требование к программному обеспечению ноутбука

- проверка исправности камеры;
- возможность выдачи команд управления;
- наличие органов управления динамическими характеристиками робота;
- проверка исправности ходовой части;
- проверка сигнала сети;
- возможность задания тайм-аута для проверки наличия сети;
- проверка наличия сети с заданным тайм-аутом;
- автоматическая выдача манипулятору команда на установку ретранслятор при обнаружении отсутствия сети;
- возможность принудительной проверки наличия сети вне заданного таймаута;

- возможность отключения автоматической установки ретранслятора.

2.6.3. Требование к программному обеспечению смартфона

- отображение видео с камер робта;
- возможность сохранения видео;
- возможность захвата кадра;
- вожможность настраивать параметры видео;
- возможность настраивать сетевое подключение.

2.7. Требования к помещению

- высота: не менее 300 мм;
- ширина: не менее 300 мм;
- угол плоскости движения: 45 градусов;
- максимальная высота водных препятствий: 100 мм;
- отсутствие высокого радиоактивного фона.

3. ПРОЕКТИРОВАНИЕ

3.1. АРХИТЕКТУРА ПРЕДЛАГАЕМОГО РЕШЕНИЯ

Raspberry Pi 3 имеет на борту 40-пиновую рейку портов ввода/вывода общего назначения (Рисунок 3.1.1). Но 12 портов из них представляют из себя пины питания 3.3 В, 5 В и общие пины GND. Также порты GPIO0 и GPIO1 используются для конфигурации EEPROM Малинки для работы с устройствами поверхностного монтажа и использование этих пинов крайне не рекомендуется. Фактически получается, что GPIO-пинов не 40, а 28.

Составим схему подтлючения датчиков и периферийных устройств робота. Необходимо подключить две камеры к одноплатному компьютеру через один интерфейс видеокамеры MIPI CSI camera port. Потребуется плата мультиплексор камер Raspberry Pi. Такой модуль подключается и управляется по интерфейсу I2C. Число серво моторов превышает имеющееся количество портов с возможностью ШИМ на шине GPIO. Для решения проблемы подойдет внешний многоканальный ШИМ-контроллер. Управление сервоприводами вёдётся через интерфейс I2C. Для управления шаговым двигателем и двумя редукторными моторами необходим внешний контроллер коммутации двигателей. Подойдёт плата расширения на контроллере L293D. Модуль занимает порты GPIO17, GPIO22, GPIO23, GPIO24, GPIO25, GPIO27. Датчики анализа среды и состояния робота имеют интерфейс I2C и 1-Wire.

Два лидара будут подключены к компьютеру Raspberry Pi по интерфейсу USB через модуль вкючения/отключенрия питания USB.

Спроектируем схему подключения модулей робота (Приложение Г).

Два лидара подключаются к одноплатному компьютеру по интерфейсу USB.

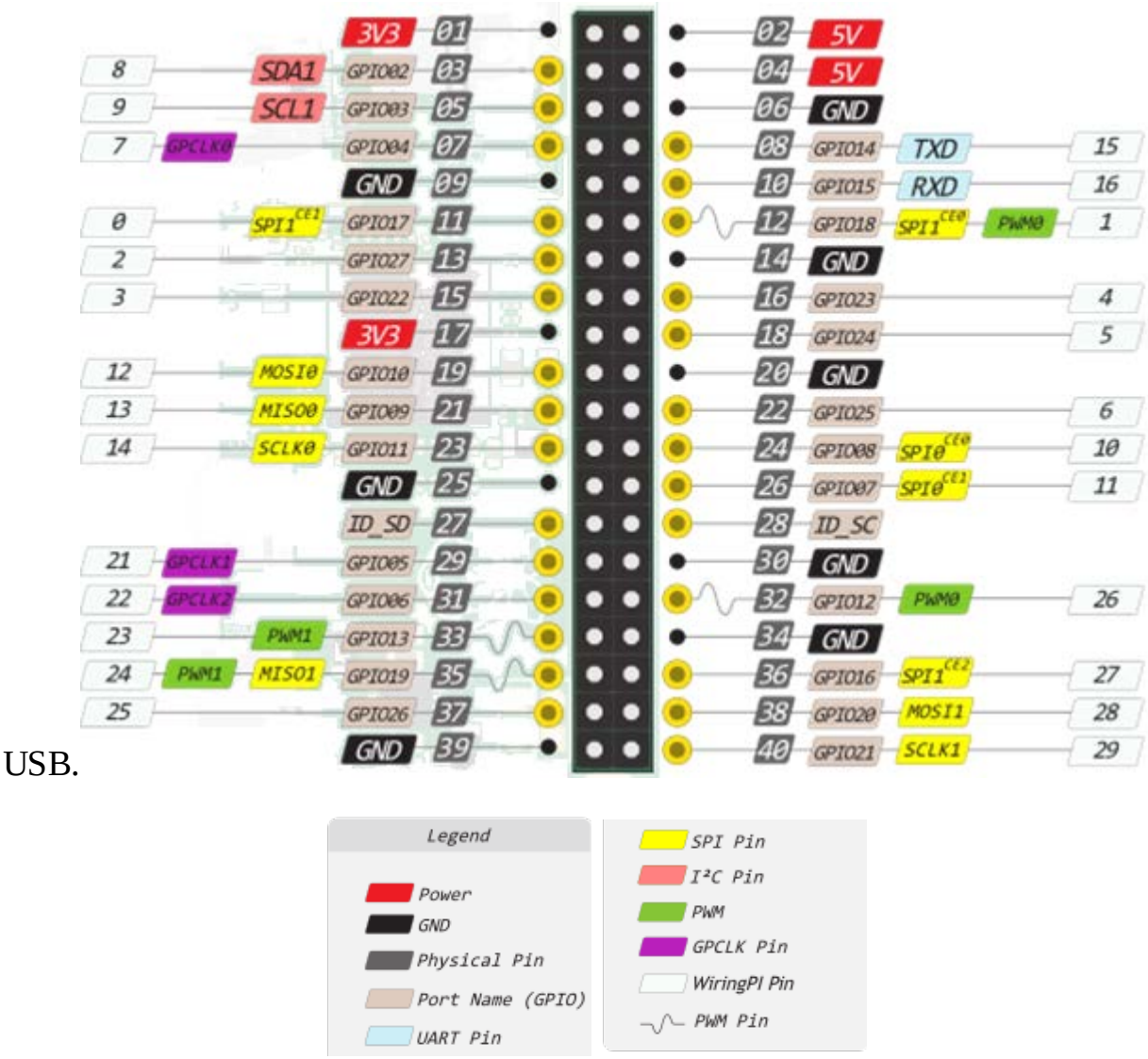


Рисунок 3.1.1 – Схема портов GPIO компьютера “Raspberry Pi 3 B+”

Спроектируем схему Wi-Fi ретранслятора. Питание ретранслятора возьмёт литий-ионный аккумулятор формата 18650. Для удопства включения усилителя питание к микроконтроллеру ESP8266 проведено через герметизированный контакт (Геркон). Геркон имеет особенность изменять состояние подключённой электрической цепи при воздействии магнитного поля. Применим геркон с нормально замкнутым контактом, чтобы при отсутствии магнитного поля контакт был замкнут. При установке датчиков с отсеки с макгитным дном сить питания ретрансляторов размыкаются и ретрасляторы остаются

отключеными до тех пор,пока робор манипулятором не извлечёт выбранный усилитель из отсека и бросит на землю.

Электрическая схема сборки Wi-Fi ретранслятора указана на рисунке 3.1.2.

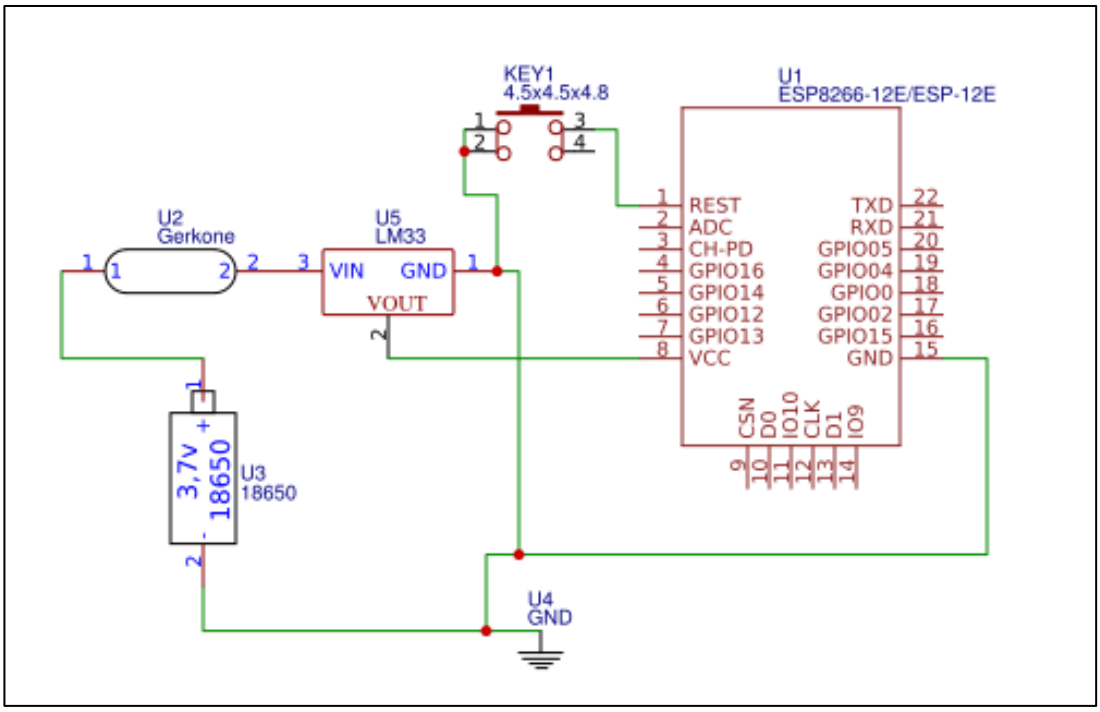


Рисунок 3.1.2 – Схема сборки Wi-Fi ретранслятора

3.2. АЛГОРИТМЫ РЕШЕНИЯ ЗАДАЧИ

На рисунке 3.2.1 представлен укрупнённый алгоритм работы программной части робота.

Перед подготовкой привязываемого сокета к принятию входящих соединений (так называемое «прослушивание»), робот запускает службу ведения логов и инциализирует подлюченое оборудование – перефирийные устройства и датчики. После обнаружения подключения клиента происходит аутентификация и только в случае подтверждения атентификации серверный сокет начинает принимать сообщения от клиента и по надобности отправлять клиенту сообщения в ответ. После получения сообщения с просьбой

завершить подключенрие сервернрый сокед закрывает соединение с клиентским сокетом.

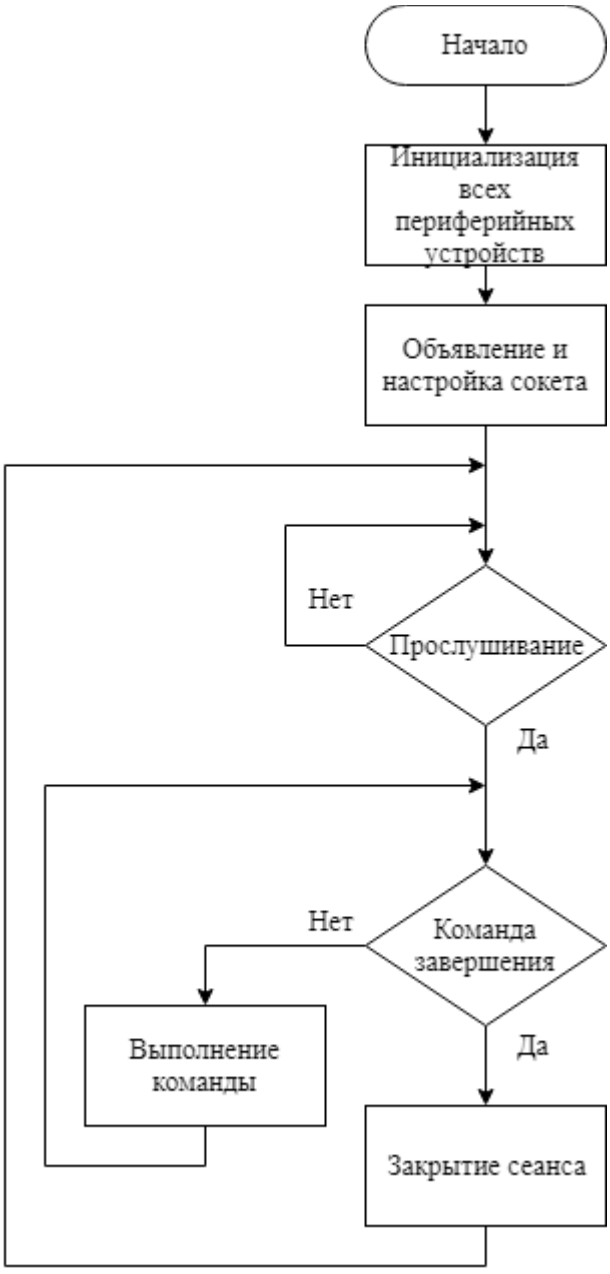


Рисунок 3.2.1 - Укрупненный алгоритм работы робота

Общий принцип работы SLAM показан на рисунке 3.2.1. Сначала происходит сбор информации с двух лидаров. Следующим шагом является использование фильтра частиц для предсказания наилучшей позиции и направления объекта с учетом полученных данных. Иначе говоря, относительно ориентации и координат, которые на этот момент считается

наиболее подходящими для управляемого объекта, происходит, разброс теоретических частиц, каждая из которых имеет направление и позицию, с нормальным распределением. Далее, с помощью собранных данных и карты препятствий, которая содержит предыдущие измерения лидара, проводится весовая оценка каждой частицы. Определив частицу с большим весом и соответствующие ей координаты и направление, система переходит к обновлению карты препятствий. На этом этапе данные с лидара наносятся на карту с учётом найденной позиции и ориентации в фильтре частиц[16].

Рисунок 3.2.1 – Алгоритм SLAM

3.3. ОПИСАНИЕ ДАННЫХ

Форматы файлов VTK (Visualization Toolkit) предоставляет ряд исходных и пишущих объектов для чтения и записи популярных форматов файлов данных. Visualization Toolkit поддерживает пять различных форматов набора данных: структурированные точки, структурированная сетка, прямолинейная сетка, неструктурированная сетка и полигональные данные. Формат файла со структурированными точками поддерживает наборы структурированных точечных данных 1D, 2D и 3D. На рисунке показан пример VTK файлас описанием его внутреннего устроя.

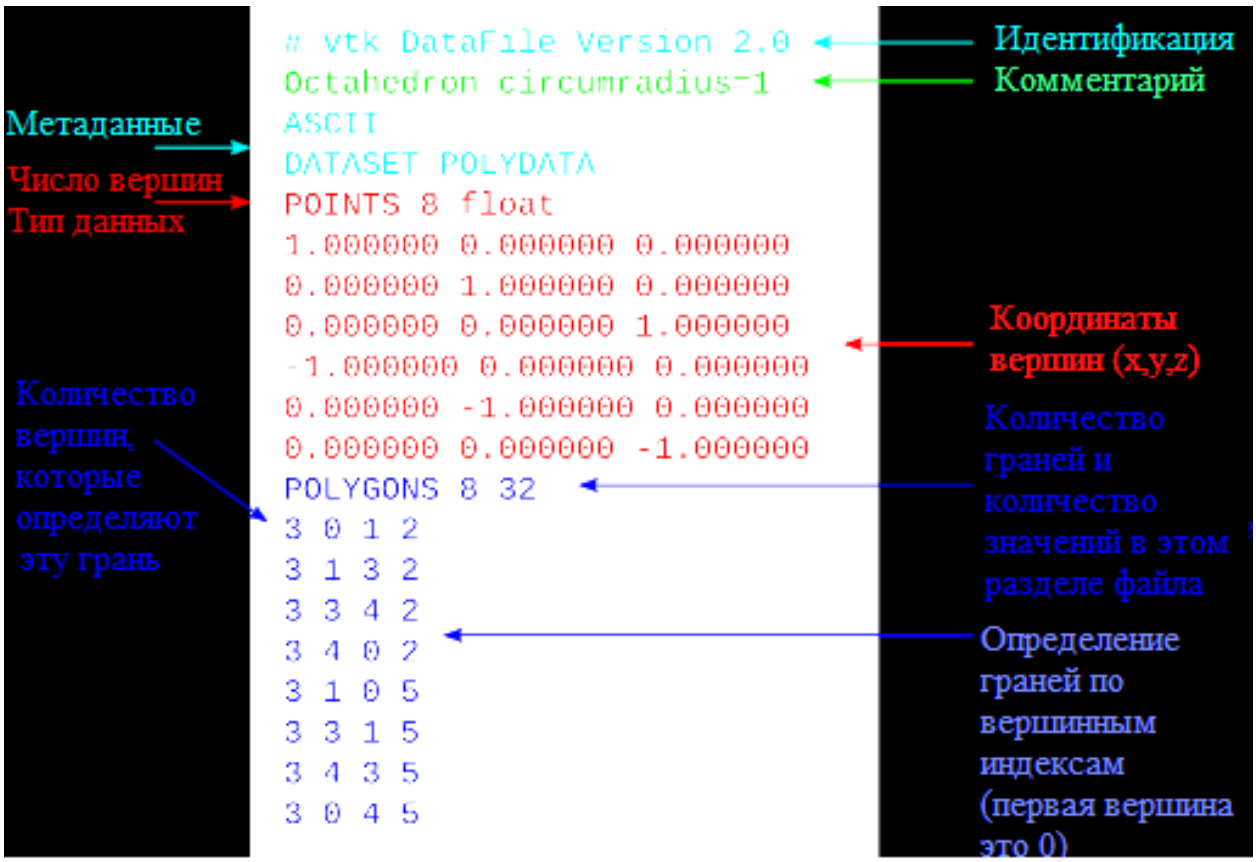


Рисунок 3.3.1 – Пример VTK файла

4. РЕАЛИЗАЦИЯ

4.1. Сборка прототипа

Чтобы проверить в действии алгоритмы работы системы, необходимо собрать макет с участием упомянутых выше плат. Заказ оборудования производился из российского магазина «ЧИП и ДИП»[17]. Не найденое на территории Росии оборудование било заказано с иностранных онлайн-магазинов «Amazon»[18] и «AliExpress»[19].

Одноплатный компьютер “Raspberry Pi 3 B+”

Рисунок 4.1.1– Одноплатный компьютер “Raspberry Pi3 B+”

Таблица 4.1.1– Характеристики компьютера “Raspberry Pi3 B+”:

Габариты	85x56x17 мм
Процессор	64-битный 4-ядерный ARM Cortex-A53 с тактовой частотой 1,4 ГГц на однокристальном чипе Broadcom BCM2837
Оперативная память	1ГБ LPDDR2 SDRAM
USB порты	USB 2.0×4
Беспроводная сеть	WiFi 2,4/5 ГГц, 802.11n
Разъем видеокамеры	MIPI Camera Serial Interface (CSI-2)
ПРорты ввода-вывода	40
Стоимость	3 850 руб.
Источник	https://www.chipdip.ru/product/raspberry-

	pi-3-model-b-2

Модуль камеры “Camera Module V2 8MP Sensor 160 Degree”

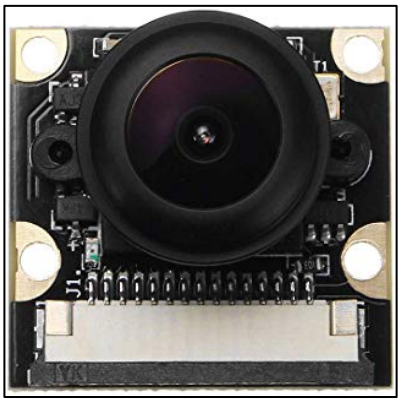


Рисунок 4.1.2 – Модуль камеры “Camera Module V2 8MP Sensor 160 Degree”

Таблица 4.1.2– Характеристики камеры “ Camera Module V2 8MP ”:

- ы	¼ дюйма
Диафрагма	2.35 F
Поле зрения	диагональный угол 160 градусов, горизонтальный угол 120 градусов
Сенсором	Sony IMX219 (8 Mpx)
Стоимость	2 140 руб.
Источник	https://www.chipdip.ru/product/rpi-camera-g-waveshare

Плата “Arducam Multi Camera Adapter Module V2.1”

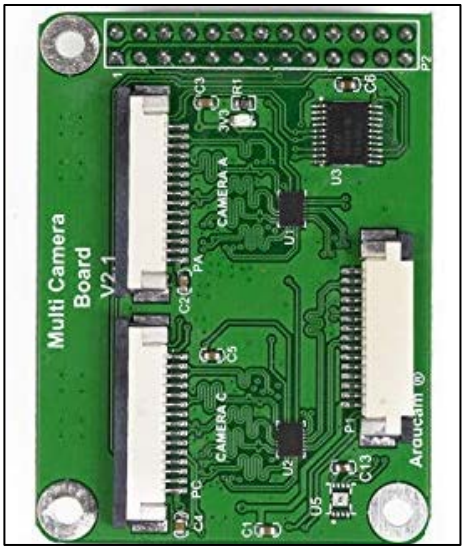


Рисунок 4.1.3 – Плата “Arducam Multi Camera Adapter Module V2.1”

Таблица 4.1.3– Характеристики платы “Arducam Multi Camera Adapter Module V2.1 ”:

Число возможных размещений камер	до 4 шт.
Поддержываемые модули	5-мегапиксельная и 8-мегапиксельная камеры Raspberry Pi
Моддержка моделей Raspberry Pi	Поддержка на моделей Raspberry Pi Model A/B/B+, Pi 2 and Raspberry Pi 3,3b+
Стоимость	4 577 руб.
Источник	https://www.amazon.com/Arducam-Camera-Adapter-Raspberry-Cameras/dp/B07JHP1T9K

Лидар “RPLiDAR A1”

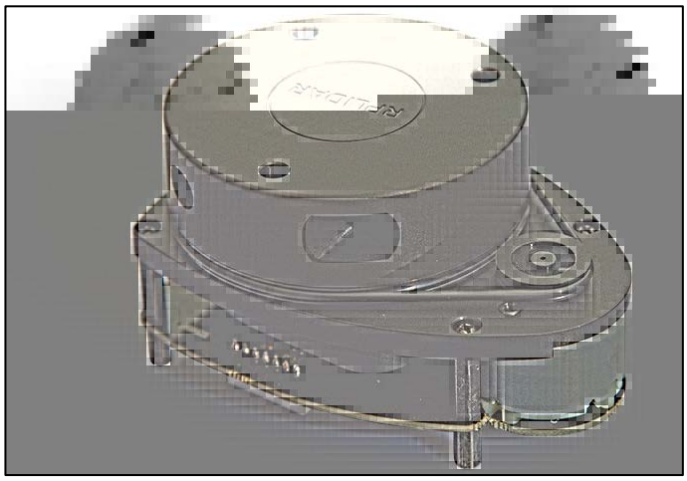


Рисунок 4.1.4 – Лидар “RPLiDAR A1”

Таблица 4.1.4– Характеристики лидара “RPLiDAR A1”:

Дальность	12 м.
Угловой диапазон	360 градусов
Скорость замеров	8000 раз/сек

Угловое разрешение	менее 1 градуса

Продолжение таблицы 4.1.4

Питание	5 Вольт
Стоимость	6 659 руб.
Источник	https://is.gd/JLGjkJ

Блоки инерциальных датчиков “MPU9250”

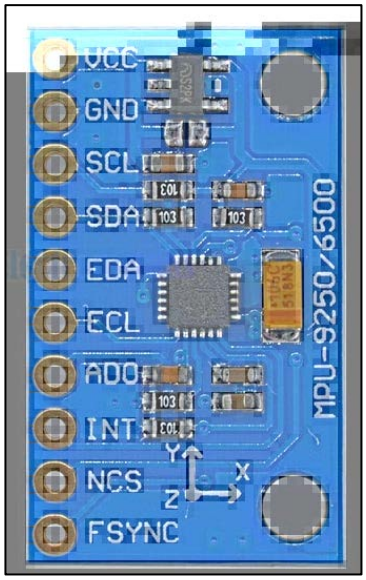


Рисунок 4.1.5 – Датчик “MPU9250”

Таблица 4.1.5– Характеристики датчика “MPU9250”:

Габариты	25 x 16 x 3 мм
Диапазон измерения акселерометра	+/- 2G, +/- 4G, +/- 8G, +/- 16G
Диапазон измерения гироскопа	+/- 250, +/- 500, +/- 1000, +/- 2000°/c
Чувствительность гироскопа	131, 65,5, 32,8, 16,4 LSB/°/c
Диапазон измерения компаса магнитометра	+/- 4800 мкТл
Интерфейс	I2C (400кГц) / SPI (1 МГц)
Питание	3 - 5 Вольт
Стоимость	420 руб.

Источник	https://www.chipdip.ru/product0/9000407709

Много гусеничное шасси “Multi chassis Tank”

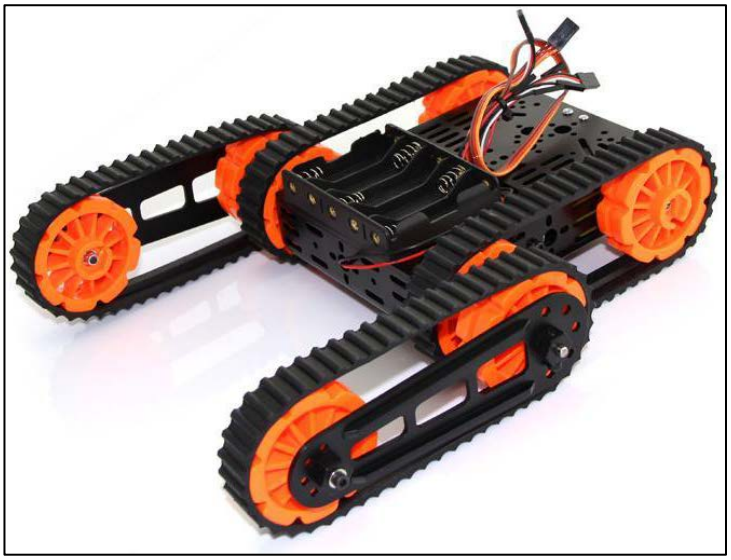


Рисунок 4.1.6 – Много гусеничное шасси “Multi chassis Tank”

Таблица 4.1.6– Характеристики шасси “ Multi chassis Tank ”:

Размеры	262 x 222 x 60 мм
Питание	6–9 Вольт
Максимальная загрузка	На ходу: 7,5 кг (на земле, тест с 7,4 В 1100 мАч 10C 8,14 Втч. Литий-ионный аккумулятор) Не двигается: 50 кг (на устойчивой опоре)
Стоимость	13 515 руб.
Источник	https://ru.aliexpress.com/store/product/Multi-Chassis-Tank-Rescue-Version-Car-Plaform-Robot-Chassis/4308031_32911862330.html

Плата расширения ШИМ сигналов на PCA9685 16 каналов

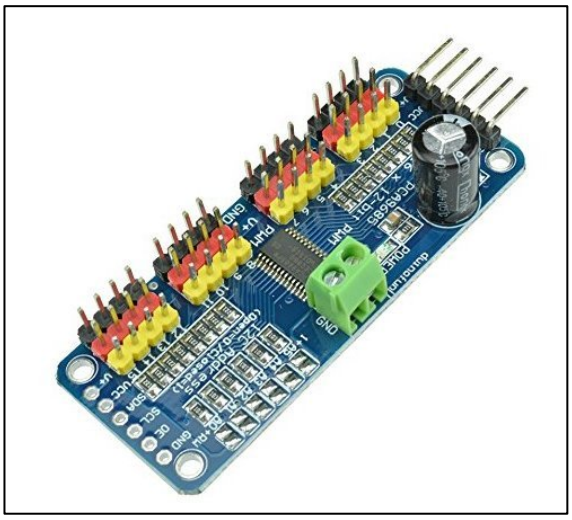


Рисунок 4.1.7 – Плата расширения ШИМ сигналов на PCA9685

Таблица 4.1.7– Характеристики платы ШИМ сигналов на PCA9685:

Габариты	66 x 25 мм
Количество подключаемых нагрузок	16 шт.
Интерфейс	I2C
Питание	2.3 – 5.5 Вольт
Стоимость	105 руб.
Источник	https://is.gd/kW7doZ

Микроконтроллер “Espressif ESP8266”

Рисунок 4.1.8 – Микроконтроллер “ESP8266”

Таблица 4.1.8– Характеристики микроконтроллера “ESP8266”:

Питание	2,2…3,6 Вольт
Процессор	Двухъядерный 32-бит Tensilica Xtensa® LX6 с FPU и MAC. 240 МГц (600 DMIPS)
Памсять	448 кБайт ПЗУ, 520 кБайт ОЗУ. Внешние ОЗУ/ПЗУ на SPI интерфейсе, до 4*16 МБайт. Внешняя память может быть криптографически защищена.
Стоимость	410 руб.
Источник	https://www.chipdip.ru/product/esp8266-wi-fi-module

Сервопривод SG90

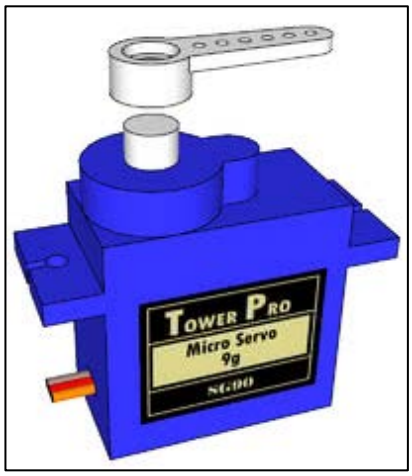


Рисунок 4.1.9 – Сервопривод “SG90”

Таблица 4.1.9– Характеристики сервопривода “SG90”:

Размеры	22мм x 11.5мм x 22.5мм
Питание	3 - 7.2 Вольт
Стоимость	99 руб.

Источник	https://is.gd/aOrvip

Ультразвуковой датчик расстояния HC-SR04

Рисунок 4.1.10 – Ультразвуковой датчик расстояния “HC-SR04”

Таблица 4.1.10– Характеристики датчика “HC-SR04”:

Питание	5 Вольт
Угол обзора	15 градусов
Измеряемое расстояние	от 0.6 до 5 м
Стоимость	260 руб.
Источник	https://www.chipdip.ru/product/hc-sr04

Дисплей OLED с контроллером SSD1306

Рисунок 4.1.11 – Дисплей OLED с контроллером “ SSD1306”

Таблица 4.1.11– Характеристики дисплея “SSD1306”:

Питание	2.8 - 5.5 Вольт
Разрешение дисплея	128 на 64 точек
Диагональ дисплея	0,96 дюйма
Интерфейс	I2C
Стоимость	182 руб.
Источник	https://is.gd/ygEb8Y

Датчик газа MQ-2

Рисунок 4.1.12 – Датчик газа“MQ-2”

Таблица 4.1.12– Характеристики датчика “MQ-2”:

Размеры	40 x 21 мм
Питание	2 - 5 Вольт
Виды газов	углеводородные газы, дым
Стоимость	550 руб.
Источник	https://www.chipdip.ru/product/mq-2-gas-sensor

Датчик температуры, давления и влажности BME280

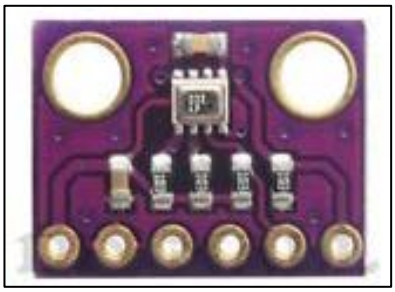


Рисунок 4.1.13 – Датчик “BME280”

Таблица 4.1.13– Характеристики датчика “BME280”:

Питание	3.3 Вольт
Диапазон измерений давления	300-1100hPa
Диапазон измерений температуры	-40 - +85 °C
Диапазон измерений влажности:	0 - 100 %
Интерфейс	SPI, I2C
Стоимость	194 руб.
Источник	https://is.gd/R7LG9b

Инфокрасный датчик TCRT5000

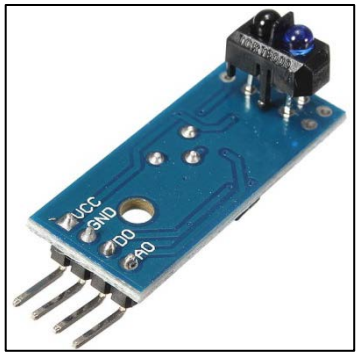


Рисунок 4.1.12 – Инфокрасный датчик “TCRT5000”

Таблица 4.1.12– Характеристики датчика “TCRT5000”:

Питание	5 Вольт
Дистанция реагированрия	15 мм
Стоимость	23 руб.
Источник	https://is.gd/05uEWi

Шаговый двигатель 28BYJ-48

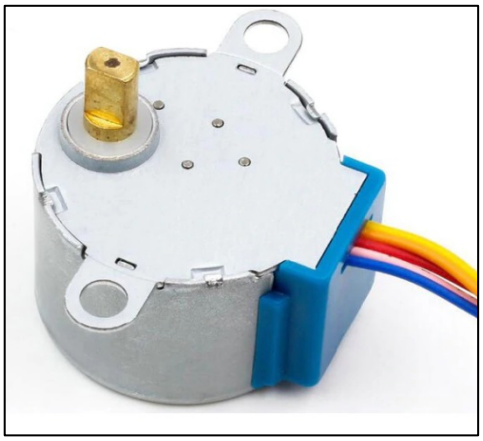


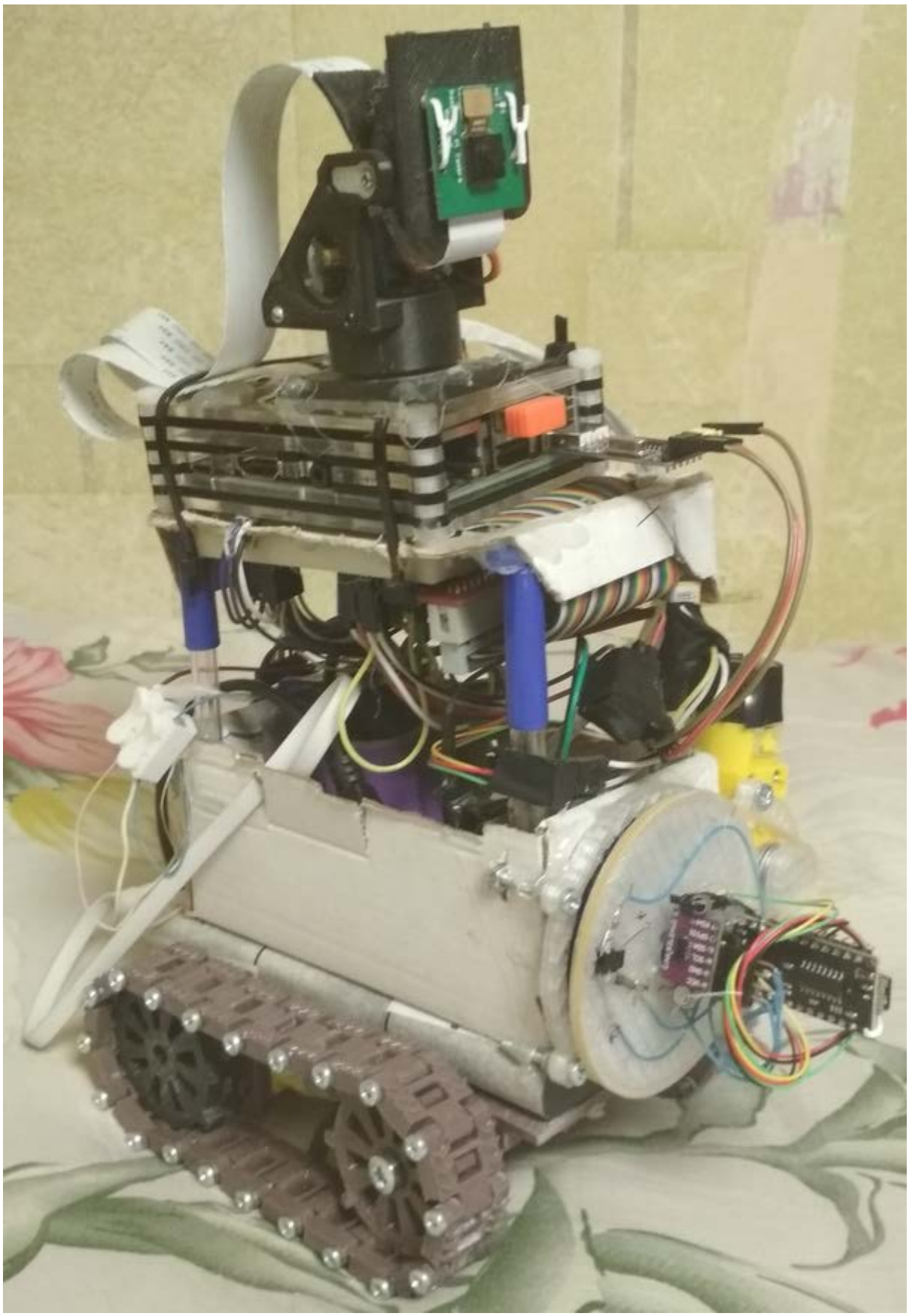
Рисунок 4.1.13 – Шаговый двигатель 28BYJ-48 “28BYJ-48”

Таблица 4.1.13– Характеристики двигателя “28BYJ-48”:

Питание	5-12 Вольт
Дистанция реагированрия	15 мм
Угол действия	5,625x1/64
Коэффициент снижения	1/64
Стоимость	51 руб.
Источник	https://is.gd/uMxOUv

Для проверки работоспособности системы и взаимодействия её ключевых компонентов реализован прототип робототехнического комплекса.

Также была подсчитана сумма расходов на разработку прототипа (Таблица 3.1.1). Данная цена не является конечной. При подсчёте не сделана оценка затрат материаля для сборки корпуса.



Русунок 4.1.14 – Прототип

Таблица 4.1.1 – Экономическая составляющая робота

Наименоване объекта	Стоимость, руб.	Количество, шт	Сумма, руб.
Raspberry Pi 3 B+	3850	1	3850
Camera Module V2 8MP Sensor 160 Degree	2140	2	4280
Arducam Multi Camera Adapter Module V2.1	4577	1	4577
RPLiDAR A1 A1M8	6659	2	13318
MPU9250	420	2	840
Multi chassis Tank	13 515	1	13515
ESP8266	410	5	2050
Сервопривод SG90	99	10	990
Ультразвуковой датчик расстояния HC-SR04	260	2	520
Дисплей OLED с контроллером SSD1306	182	1	182
Датчик газа MQ-2	550	1	550
Датчик температуры, давления и влажности BME	194	1	194
Инфокрасный датчик TCRT5000	23	2	46
Плата расширения ШИМ сигналов на PCA9685	105	1	105
шаговый двигатель	51	1	51
Акуумуляторы 18650	209	5	1045
Геркон	33	5	165
ИТОГО			46278

4.2. Реализация программного обеспечения

4.2.1. Приложение смартфона

На момент запуска приложение должно предлагать сканировать сеть на поиск серверного сокета и предложениея переподключиться к сервреному сокету по адрессу прошлой сессии. Также создадим кнопку сонфигураций для настройки приложения перед соединением с роботом.

На рисунке 4.2.1.1 показано основное окно приложения. Поле 1 отображает сообщение статуса подключения. Поле 2 позволяет перейти в настройки приложения. Кнопка 4 сканирует сеть для подключения к роботу. Кнопка 3 выполняет переподключение к роботу по прошлому найденному адресу.

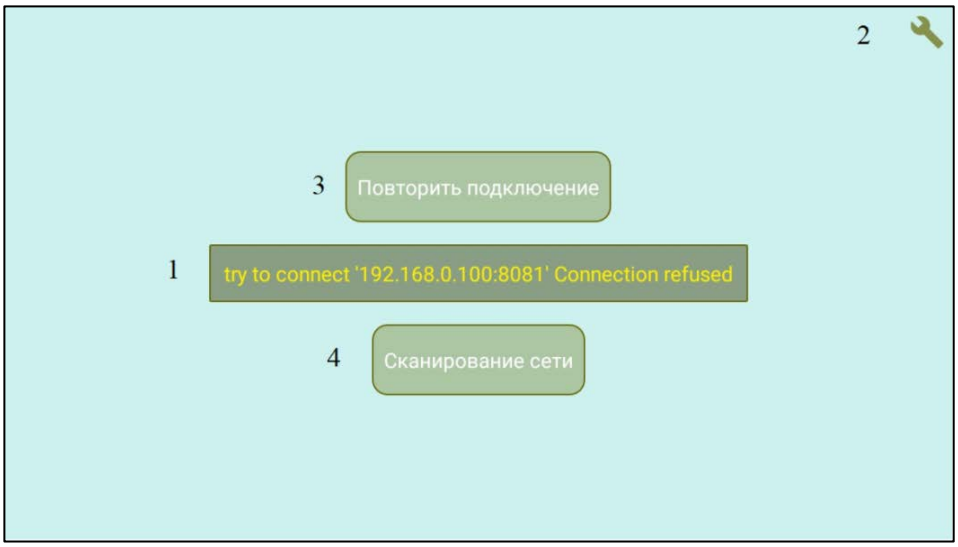


Рисунок 4.2.1.1 – Android приложение. Главное меню

При нражатии на кнопку конфигураций париложение перейдёт в окно настроек сети. Добавим возможность настоить подключение с роботом и параметры отображения камеры. На рисунке 4.2.1.2 показано окно настроек. В них входит настройки интернета (поле 1) и настройки видеокамеры (поле 2).

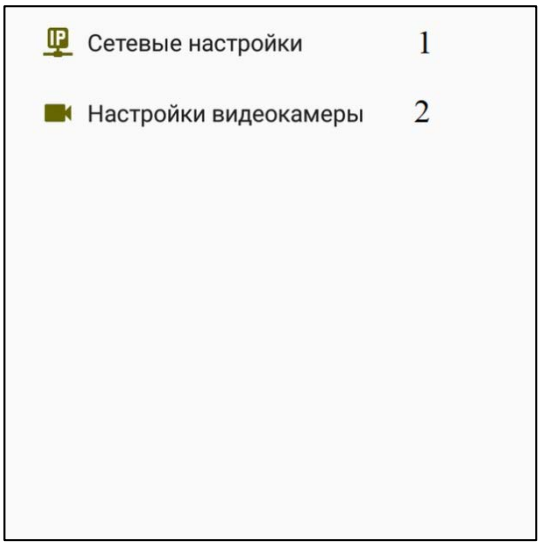


Рисунок 4.2.1.2 – Android приложение. Окно настроек

Сиздадим окно настроек сетевого подключения. Будут возможно изменение таких параметров, как номер порта TCP/IP хоста, тайм-аутр соединения с хостом, частный хост сети Wi-Fi и пароль аутентификации. На рисунке 4.2.1.4 изобораженно готовое окно приложения «Сетевые настройки».

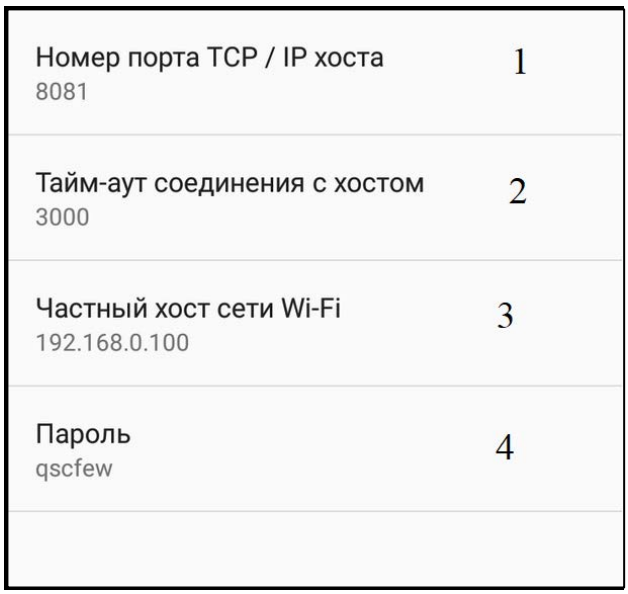


Рисунок 4.2.1.3 – Android приложение. Окно настроек сетевого соединения

Создадим окно настроек камеры. Будут изменяться такие конфигурации, как качество камеры, частота кадров каметы, светочувствительность камеры (ISO), битрейт потока камеры и разрешение камеры. зададим настройки камеры. Качество камеры выставляется в диапазоне от 0 до 40, где 0 - «приемлемое качество», 10 - очень высокое качество, 40 - очень низкое. На рисунке 4.2.1.4 изобораженно готовое окно приложения «Настройки видеокамеры».

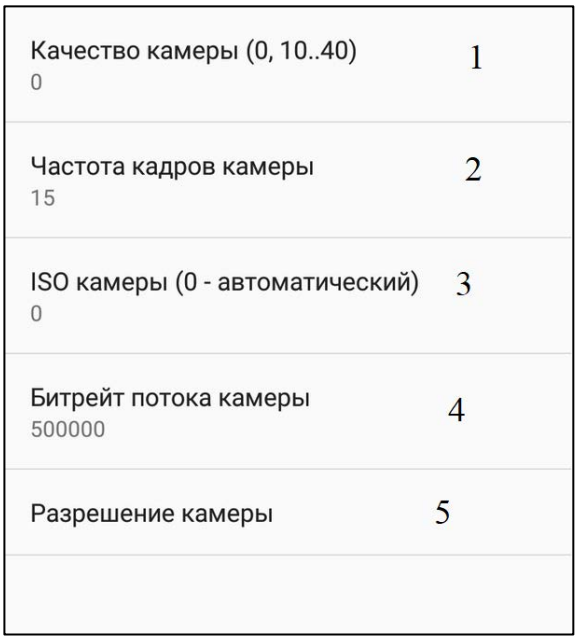


Рисунок 4.2.1.4 – Android приложение. Окно настроек камеры

Во время показа видео с камер робота (Рисунок 4.2.1.5, поле 1) кнопка настроек останется видимой (Рисунок 4.2.1.5, поле 2), чтобы не отключаясь от робота была возможность поменять конфигурации видеосвязи. Дополним интерфейс кнопками «завершить соединение» (Рисунок 4.2.1.5, кнопка 3), «включить запись» (Рисунок 4.2.1.5, кнопка 4) и «захват кадра» (Рисунок 4.2.1.5, кнопка 5). Добавим также поле для показа параметров подключения и показаний датчиков робота.

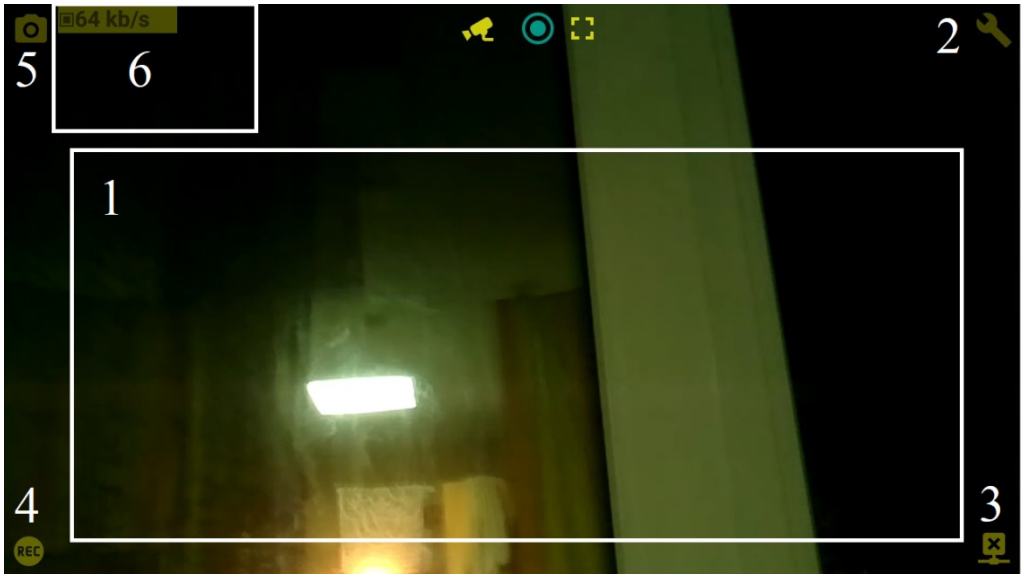


Рисунок 4.2.1.5 – Android приложение. Окно видеотрансляции

Код Android приложения прикреплён к приложению В. Полный код приложения прилогается по электронному ресурсу “github.com”[20].

4.2.2. Приложение ноутбука

На рисунке 4.2.2.1 указано строение приложения на PC. Окно инструментов расположено в поле 1. Поле 2 является основным полем и отображает карту, загруженную у робота. Код программного обеспечения указан в приложении Б.

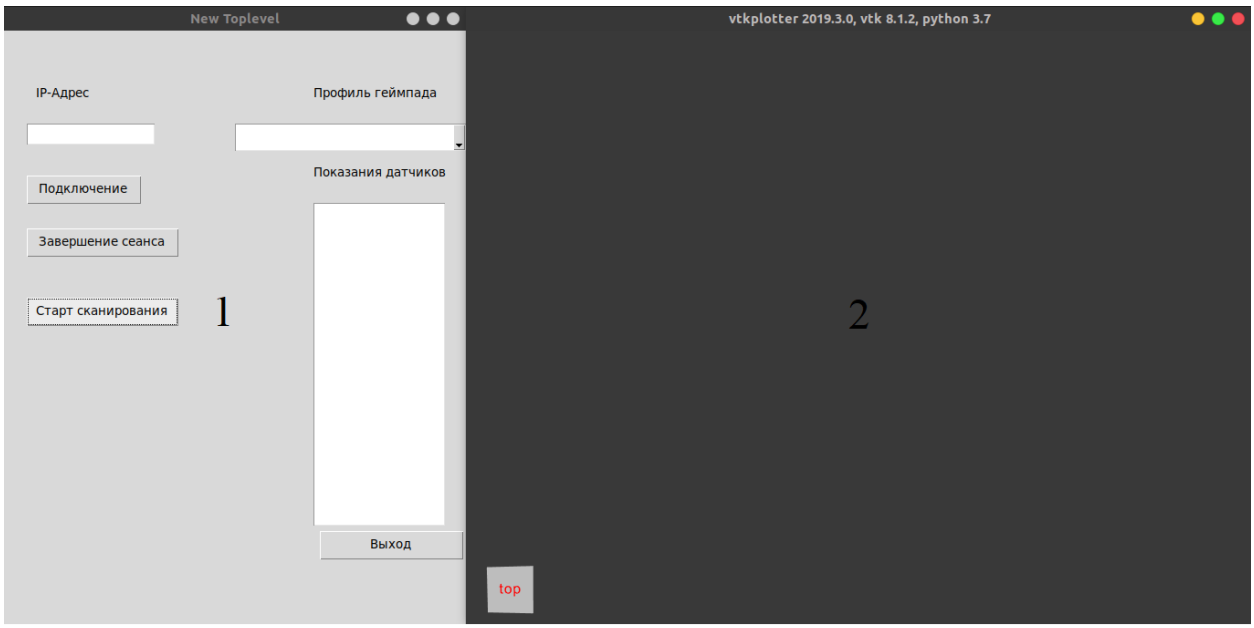


Рисунок 4.2.2.1 – Графический интерфес приложения на ноутбуке

4.2.3. Программное обеспечение Wi-Fi усилителя

Программное обеспечение Wi-Fi усилителя прилогается по электронному ресурсу “github.com”[20].

5. ТЕСТИРОВАНИЕ

5.1. МЕТОДОЛОГИЯ ТЕСТИРОВАНИЯ

Так как система представляет собой программно-аппаратный комплекс и в результате данной работы уделено большее внимание его аппаратной части и её взаимодействию с сервером, проводилось альфа-тестирование реализованного функционала. Альфа-тестирование – имитация реальной работы с системой разработчиком, либо реальная работа с системой потенциальными пользователями. Этот тип тестирования может проводиться как на ранней стадии разработки продукта, так и для законченного продукта в качестве внутреннего приёмочного тестирования. Иногда альфа-тестирование выполняется под отладчиком или с использованием окружения, которое помогает быстро выявлять найденные ошибки.

Обнаруженные ошибки могут быть переданы тестировщикам для дополнительного исследования в окружении, подобном тому, в котором будет использоваться программа.

5.2. ПРОВЕДЕНИЕ ПРОЦЕДУРЫ ТЕСТИРОВАНИЯ

Проверим задержку передачи видеопотока между смартфоном и роботом. Результаты показаны на рисунках 5.2.1 и 5.2.2.

Тест показал, что задержка составляет 350..600ms. Но не более 0.6 сек и стабильно в этом диапазоне. Стоит заметить, что это в очень «RF зашумленной» среде многоквартирных домов, с множеством WiFi сетей в округе.

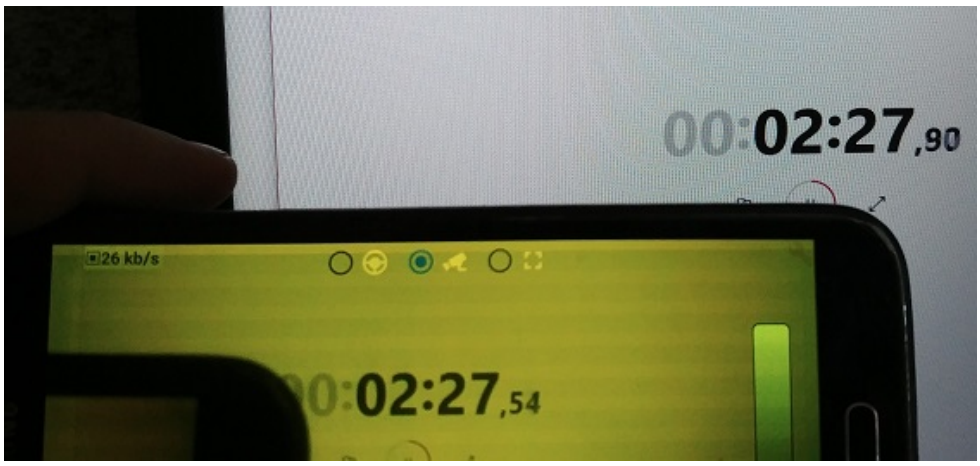


Рисунок 5.2.1 – Процедура тестирования задержки №1

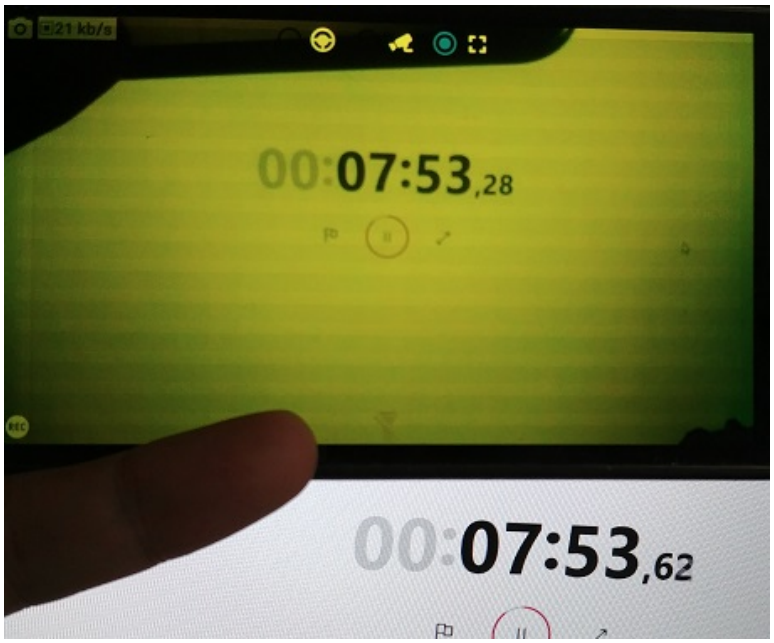


Рисунок 5.2.2 – Процедура тестирования задержки №2

Проведём испытание картопостроения. Робот запущен в коридор длинной более 50 метров, шириной 2 метра и высотой, 3,5 метра (Рисунок 5.2.3). Запустим сканирование местности. В результате рна основе результатов роботов была постноена модель помещения формой прямоугольный тоннель (Рисунок 5.2.4). Высоких отклонений не наблюдается. Значит испыпание сканирования пространства завершено успешно.

Рисунок 5.2.3 – Объект сканирования

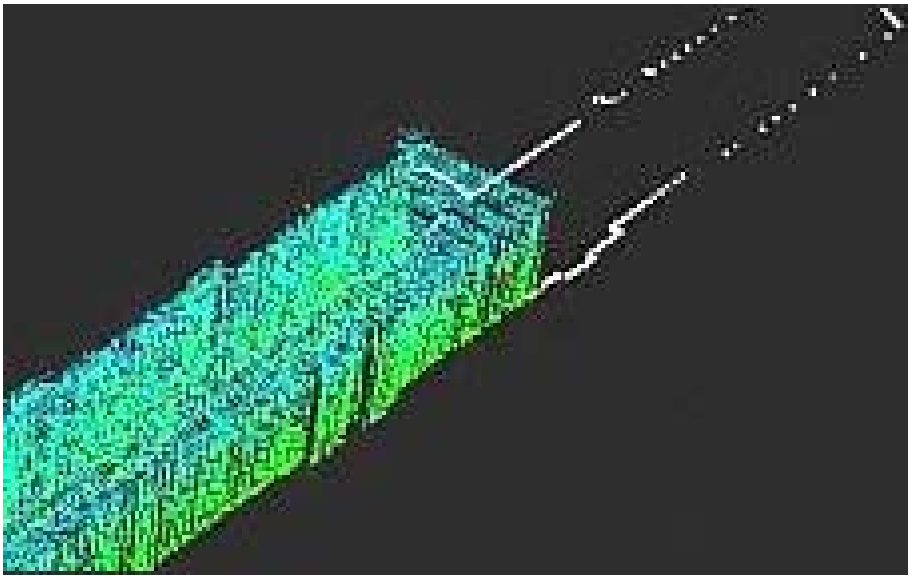


Рисунок 5.2.4 – Показание ноутбука. Карта коридора

6. ЗАКЛЮЧЕНИЕ

В представленной работе были разобраны все основные темы, связанные с проектированием, разработкой и вводом в эксплуатацию мобольного робототехнического комплекса. Подобраны аппаратные решения, подходящие под требования системы. На стадии разработки, внедрения и эксплуатации использовалось свободное ПО: Python, Geany и Android Studio. Был сконструирован прототип системы и написано программное обеспечение под мобильного робота, ноутбука в качестве пульта управления и VR-шлема в виде Android устройства. В ходе проектирования было оценена стоимость прототипа и стоимость завершённого проекта. Система получила стоимость ниже своих аналогичных решений. Система может вести дистанциооное управление робота через Wi-Fi ретрансляторы. В процессе управления робот передал показания датчиков, данные лидара и видеосвязь.

Система прошла альфа-тестирование. Из-за несоответствия нехоторых характеристик протопита с будущей разрабатываемой системы, были проверены только следующие функции: время задержки видеосвязи между роботом и VR-шлемом и построение 3D модели помешения.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Веб-сайт «Robo-sapiens». [Электронный ресурс]. URL: https://robo-sapiens.ru/stati/primenenie-robotov-v-sovremennom-mire/. (Дата обращения: январь 2019 года).

2 Brian Duval, Журав А.А Аналоговые датчики расстояния. – http://www.gaw.ru/html.cgi/txt/publ/sensor/analog_sensor.htm.

3 Лидар [Электронный ресурс]: - Электрон.дан. - http://gufo.me/content_tehenc/lidar-7673.html (дата обращения: 15.05.17);

4 Певзнер Л.Д., Ким М.Л. Робототехника в горном деле. – https://cyberleninka.ru/article/v/robototehnika-v-gornom-dele.

5 Веб-сайт «Roboteam». [Электронный ресурс]. URL: http://www.robo-team.com/products/mtgr/#s-6. (Дата обращения: февраль 2019 года).

6 Веб-сайт «Leo Rover». [Электронный ресурс]. URL: https://www.leorover.tech/. (Дата обращения: февраль 2019 года).

7 Веб-сайт «Emesent». [Электронный ресурс]. URL: https://emesent.io/products/mining-program/. (Дата обращения: февраль 2019 года).

8 SteveGrehl, HelmutMischo, BernhardJung, Мобильные роботы на подземной добыче. –https://zolotodb.ru/article/11665.

9 Веб-сайт «is.gd». [Электронный ресурс]. URL: https://is.gd/. (Дата обращения: май 2019 года).

10 Олег Корякин, В россии создают миниатюрных роботов-разведчиков. –https://rg.ru/2014/11/07/robot-site-anons.html.

11 Веб-сайт «set-1.ru». [Электронный ресурс]. URL: https://www.set-1.ru/products/detail.php?ELEMENT_ID=140. (Дата обращения: декабрь 2018 года).

12 Веб-сайт «mchs.gov.ru». [Электронный ресурс]. URL: https://www.mchs.gov.ru/upload/site1/document_file/eCAZWmsaUt.doc. (Дата обращения: январь 2019 года).

13 Легков К.Е Беспроводные mesh сети специального назначения – https://cyberleninka.ru/article/n/besprovodnye-mesh-seti-spetsialnogo-naznacheniya.

14 GPS: принципы работы системы и точность определения координат — http://sfort-telecom.ru/tochnost/.

15 Васильев П. В Повышение точности корректируемой инерциальной навигационной системы. – https://cyberleninka.ru/article/v/povyshenie-tochnosti-korrektiruemoy-inertsialnoy-navigatsionnoy-sistemy.

16 Барамия Д.А., Дьяков М.С., Лаврентьев М.М. Разработка системы одновременной локализации и построения карты на основе данных с лидара и видеокамер. – Вести. Новосибю госю ун-та. Серия: Информационные технологии, 2015. – 15 с.

17 Веб-сайт «ЧИП и ДИП». [Электронный ресурс]. URL: https://www.chipdip.ru/. (Дата обращения: май 2019 года).

18 Веб-сайт «Amazon». [Электронный ресурс]. URL: https://www.amazon.com/. (Дата обращения: май 2019 года).

19 Веб-сайт «AliExpress». [Электронный ресурс]. URL: https://aliexpress.com/. (Дата обращения: май 2019 года).

20 Морозов О.И Исходный код ВКР «Разработка мобильного робототехнического комплекса для создания 3D карт замкнутых пространств» – https://github.com/olegon17/Mobile_Robot_FALLBAT.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ РОБОТА

Server.py

```
import socket
import threading
import time, calendar
from datetime import datetime
import sys, subprocess, struct
import logging
import argparse
import binascii
import RPi.GPIO as GPIO
import l293d.driver as l293d
print l293d.Config.set_pin_numbering('BCM')
print GPIO.getmode()
import picamera
import picamera.array
import numpy as np
import Adafruit_SSD1306
from PIL import Image
from PIL import ImageDraw
from PIL import ImageFont
import subprocess

Cam_X = 6
Cam_Y = 8
servoPINV = 4
servoPINH = 18
password = 'qscfew'
fileNameFormat = '%y-%m-%d_%H-%M-%S.%f'
snapshotExec = '/home/pi/motion_cp.sh'

#класс видеообработки
class VideoProcessing:
    motionTime = 0
    snapshotBodyJpg = None
    snapshotMotionValue = 0
    motionData = None
    lockSnapshot = threading.Lock()
    #инициализация
    def __init__(self):
        self._thread = None
        self._threadSnapshot = None
        self._stream_interrupt = threading.Event()
        self._stream_interrupt.set()
        self._snapshotEvent = threading.Event()
        self._snapshotEvent.clear()
        # значения по умолчанию
        self.stream_width = '1920'
        self.stream_height = '1080'
        self.stream_fps = '15'
        self.stream_bitrate = '100000'
        self.stream_iso = '0'
        self.stream_quality = '25'
    #поток снимков
    def _snapshot(self):
        logger.debug('Начать поток снимков')
```

```
        while True:
            self._snapshotEvent.wait()
            if self._stream_interrupt.isSet():
                break
            file_name = '/tmp/' + datetime.now().strftime(fileNameFormat) + '.jpg'
            logger.debug('Запуск. Делается снимок:' + file_name)
            camera.capture(file_name, use_video_port=True, format='jpeg', quality=80)
            with VideoProcessing.lockSnapshot:
                with open(file_name, mode='rb') as file:
                    VideoProcessing.snapshotBodyJpg = file.read()
                VideoProcessing.motionTime = calendar.timegm(time.gmtime())
                VideoProcessing.snapshotMotionValue = DetectMotion.motionValue
                VideoProcessing.motionData = np.copy(DetectMotion.motionData)
            logger.debug('Запуск. отправка снимка. mv number:' +
str(VideoProcessing.snapshotMotionValue))
            if snapshotExec is not None:
                p = subprocess.Popen('exec ' + snapshotExec + ' ' + file_name,
stdout=subprocess.PIPE,
                                     stderr=subprocess.PIPE, shell=True)
                while True:
                    s = p.stdout.readline()
                    if not s:
                        break
                    logging.debug("Отправить снимок stdout << " + s)
                while True:
                    s = p.stderr.readline()
                    if not s:
                        break
                    logging.debug("Отправить снимок stderr << " + s)
                p.wait()
            logger.debug('Конец. Снимок')
            time.sleep(DetectMotion.cfg_motionSnapshotDt)
            if self._stream_interrupt.isSet():
                break
            self._snapshotEvent.clear()
        logger.debug('Завершение потока снимков')
    #трансляция
    def _stream_thread(self, out_stream):
        camera.resolution = (int(self.stream_width), int(self.stream_height))
        camera.framerate = int(self.stream_fps)
        camera.iso = int(self.stream_iso)
        try:
            logger.debug('трансляция камеры.начало показа')
            camera.start_preview()
            time.sleep(1)
            camera.start_recording(out_stream, format='h264',
quality=int(self.stream_quality),
                                   bitrate=int(self.stream_bitrate),
                                   inline_headers=True)
            logger.debug('трансляция ожидает прерывая')
            self._stream_interrupt.wait()
        except socket.error:
            logger.debug('stream thread: socket.error')
            pass
        finally:
            camera.stop_recording()
            logger.debug('трансляция камеры.остановка записи')
    def _detect_motion_thread(self):
        try:
            camera.resolution = (1296, 730)
```

```
            camera.framerate = 30
            logger.debug('motion camera.start_recording')
            stream = picamera.PiCameraCircularIO(camera, seconds=10, bitrate=1000000)
            camera.start_recording(stream, format='h264',
motion_output=DetectMotion(camera, self._snapshotEvent))
            self._stream_interrupt.wait()
        finally:
            camera.stop_recording()
            logger.debug('motion camera.stop_recording()')
    #стоп
    def stop(self):
        logger.debug('start interrupting VideoProcessing thread')
        self._stream_interrupt.set()
        if self._thread is not None:
            logger.debug(' start interrupting the camera thread.')
            self._thread.join()
            logger.debug(' end interrupting the camera thread')
            self._thread = None
        self._snapshotEvent.set()
        if self._threadSnapshot is not None:
            logger.debug(' start interrupting the snapshot thread.')
            self._threadSnapshot.join()
            logger.debug(' end interrupting the snapshot thread')
            self._threadSnapshot = None
        self._stream_interrupt.clear()
        self._snapshotEvent.clear()
        logger.debug('end interrupting VideoProcessing thread')
    #старт
    def start(self, a, out_stream):
        [self.stream_width, self.stream_height, self.stream_fps, self.stream_bitrate,
self.stream_quality,
         self.stream_iso] = a
        logger.debug('new stream w:%s h:%s fps:%s br:%s\n' % (
            self.stream_width, self.stream_height, self.stream_fps, self.stream_bitrate))
        self.stop()
        self._thread = threading.Thread(name='tcp stream', target=self._stream_thread,
args=[(out_stream)])
        self._thread.start()

#вход в программу
if __name__ == '__main__':
    parser = argparse.ArgumentParser(description='video streamer and motion detector')
    parser.add_argument("-d", "--debug_level", dest="logLevel",
                        choices=['DEBUG', 'INFO', 'WARNING', 'ERROR', 'CRITICAL'],
help='Set the logging level',
                        default='DEBUG')
    parser.add_argument("-f", "--log_file", dest="logFile", required=False, help='log
file')
    args = parser.parse_args()
    if args.logFile:
        hdlr = logging.FileHandler(args.logFile, 'a')
    else:
        hdlr = logging.StreamHandler()
    hdlr.setFormatter(logging.Formatter(u'[LINE:%(lineno)d]# %(levelname)-8s
[%(asctime)s]  %(message)s'))
    logging.getLogger().addHandler(hdlr)
    logging.getLogger().setLevel(getattr(logging, args.logLevel))
    logger = logging.getLogger('vh')
    logger.info('----------- start vhost -----------')
    GPIO.setwarnings(False)
```

```
RST = None
disp = Adafruit_SSD1306.SSD1306_128_64(rst=RST)
disp.begin()
disp.clear()
disp.display()
width = disp.width
height = disp.height
image = Image.new('1', (width, height))
draw = ImageDraw.Draw(image)
draw.rectangle((0,0,width,height), outline=0, fill=0)
padding = -2
top = padding
bottom = height-padding
x = 0
font = ImageFont.load_default()
draw.rectangle((0,0,width,height), outline=0, fill=0)
cmd = "hostname -I | cut -d\' \' -f1"
IP = subprocess.check_output(cmd, shell = True )
cmd = "top -bn1 | grep load | awk '{printf \"CPU Load: %.2f\", $(NF-2)}'"

draw.text((x, top),       "IP: " + str(IP),  font=font, fill=255)
disp.image(image)
disp.display()
time.sleep(.1)
draw.text((x, top),       "IP: " + str(IP),  font=font, fill=255)

motorL = l293d.DC(25,24,23)
motorR = l293d.DC(22,27,17)
camera = picamera.PiCamera()
server_socket = socket.socket()
vs = VideoProcessing()
GPIO.setup(servoPINV, GPIO.OUT)
GPIO.setup(servoPINH, GPIO.OUT)
v = GPIO.PWM(servoPINV, 50)
v.start(2.5)
v.ChangeDutyCycle(Cam_Y)
h = GPIO.PWM(servoPINH, 50)
h.start(2.5)
h.ChangeDutyCycle(Cam_X)
try:
    server_socket.bind(('0.0.0.0', 8081))
    server_socket.listen(0)
    while True:
        try:
            conn, address = server_socket.accept()
            logging.debug('accept:' + str(address))
            inp = conn.makefile("rt")
            isAuthenticated = False
            while True:
                data = inp.readline().strip()
                if not data:
                    logger.debug('no commands')
                    break
                l = data.split(',')
                args = l[1:]
                cmd = l[0]
                logger.debug('rcv cmd: "%s" args:%s' % (cmd, str(args)))
                if cmd == 'AUTH' and args[0] == password:
                    isAuthenticated = True
                    continue
```

```
                if not isAuthenticated:
                    logger.error('Запрос не аутентифицирован')
                    break
                if cmd == 'START_STREAM':
                    vs.start(args, conn.makefile("wb"))
                elif cmd == 'ROTATE':
                    if args[0]=='X':
                        h.ChangeDutyCycle(arg[1])
                    if args[0]=='Y':
                        v.ChangeDutyCycle(arg[1])
                elif cmd == 'REBOOT':
                    sys.exit(0)
                elif cmd == 'Caterpillar':
                    if args[0]=='L':
                        if args[1]=='UP':
                            motorL.clockwise()
                        elif args[1]=='DOWN':
                            motorL.anticlockwise()
                        elif args[1]=='STOP':
                                                    motorL.stop()
                    elif args[0]=='R':
                        if args[1]=='UP':
                            motorR.clockwise()
                        elif args[1]=='DOWN':
                            motorR.anticlockwise()
                        elif args[1]=='STOP':
                                                    motorR.stop()
                elif cmd == 'FINISH':
                    vs.stop()
                    break
                else:
                    logger.error('Неопределенные данные:"' + data + '"')
            except socket.error as e:
                logger.debug(e, exc_info=True)
            finally:
                conn.close()
    finally:
        vs.stop()
        v.stop()
        h.stop()
        server_socket.close()
        camera.close()
```

ПРИЛОЖЕНИЕ Б

ИСХОДНЫЙ КОД ПРОГРАММЫ НОУТБУКА

Client.py

```
from tkinter import *
from tkinter import ttk
import tkinter as tk
import os, sys
from vtkplotter import *
import numpy as np
from threading import Thread
import socket
from inputs import get_gamepad
RTrigger=''
LTrigger=''
RShoulder=''
LShoulder=''
Rood_X=''
Rood_Y=''
Rood_X_State = 0
Rood_Y_State = 0
socket_status = 0

global input_path
input_path = "/dev/input/by-id/"
gamepad_path=''
dirs = os.listdir(input_path)
ip_address = '192.168.0.100'
mapdir='/home/morozovn/GIT/Mobile_Robot_FALLBAT/PC_FIRMWARE/Client/'
password = 'qscfew'

def select_profile(num):
	global RTrigger,LTrigger,RShoulder,LShoulder,Rood_X,Rood_Y
	if num=='1':
		RTrigger='BTN_BASE2'
		LTrigger='BTN_BASE'
		RShoulder='BTN_PINKIE'
		LShoulder='BTN_TOP2'
	if num=='2':
		RTrigger='BTN_TR'
		LTrigger='BTN_TL'
		RShoulder='BTN_Z'
		LShoulder='BTN_WEST'
		Rood_X='ABS_HAT0X'
		Rood_Y='ABS_HAT0Y'
	print('Выбран профиль ',num)

def start_visualisation():
	print('Запуск окна карты')
	act =
load('/home/morozovn/GIT/Mobile_Robot_FALLBAT/PC_FIRMWARE/Client/'+"bunny.obj").normalize
().subdivide()
	act.color("k").alpha(0.05).wire(True)
	pts = act.coordinates(copy=True)
	vp = Plotter(N=1, axes=5)
	a = Points(pts)
	vp.show(a, at=0,interactive=1)
```

```
def read_gamepad():
	global
RTrigger_Enabled,LTrigger_Enabled,RShoulder_Enabled,LShoulder_Enabled,Rood_X_State,Rood_Y
_State
	RTrigger_Enabled = 0
	LTrigger_Enabled=0
	RShoulder_Enabled=0
	LShoulder_Enabled=0
	Rood_X_State =0
	Rood_X_State=0
	try:
		while True:
			if socket_status==0:
				thread2.join()
			events = get_gamepad()
			for event in events:
				if event.ev_type=='Key':
					if event.code==RTrigger:
						RTrigger_Enabled=event.state
					if event.code==LTrigger:
						LTrigger_Enabled=event.state
					if event.code==RShoulder:
						RShoulder_Enabled=event.state
					if event.code==LShoulder:
						LShoulder_Enabled=event.state
				if event.ev_type=='Absolute':
					if event.code==Rood_X:
						Rood_X_State=event.state
					if event.code==Rood_Y:
						Rood_Y_State=event.state

	finally:
		thread2.join()

def grand_thread():
	flag_CR_down = 0
	flag_CL_down = 0
	flag_CR_up = 0
	flag_CL_up = 0
	flag_CL_ustop = 1
	flag_CR_ustop = 1
	flag_CL_dstop = 1
	flag_CR_dstop = 1
	Y_Max = 12
	Y_Min = 5
	X_Max = 13
	X_Min = 1
	DeltaX = 1
	DeltaY = 1
	Cam_X = 6
	Cam_Y = 9

	try:
		global sock
		sock = socket.socket()
		print ('Соединение по адресу ',ip_address)
		sock.connect((ip_address, 8081))
		print ("Подключено")
		sock.send('AUTH,qscfew\n'.encode())
		global socket_status
```

```
            socket_status = 1
            thread2 = Thread(target=read_gamepad)
            thread2.start()
            while True:
                  if RTrigger_Enabled==1 and flag_CR_up ==0:
                        sock.send('Caterpillar,R,UP\n'.encode())
                        flag_CR_ustop = 0
                        flag_CR_up = 1
                  if RTrigger_Enabled==0 and flag_CR_ustop == 0:
                        sock.send('Caterpillar,R,STOP\n'.encode())
                        flag_CR_ustop = 1
                        flag_CR_up = 0

                  if LTrigger_Enabled==1 and flag_CL_up ==0:
                        sock.send('Caterpillar,L,UP\n'.encode())
                        flag_CL_ustop = 0
                        flag_CL_up = 1
                  if LTrigger_Enabled==0 and flag_CL_ustop == 0:
                        sock.send('Caterpillar,L,STOP\n'.encode())
                        flag_CL_ustop = 1
                        flag_CL_up = 0

                  if RShoulder_Enabled==1 and flag_CR_down ==0:
                        sock.send('Caterpillar,R,DOWN\n'.encode())
                        flag_CR_dstop = 0
                        flag_CR_down = 1
                  if RShoulder_Enabled==0 and flag_CR_dstop == 0:
                        sock.send('Caterpillar,R,STOP\n'.encode())
                        flag_CR_dstop = 1
                        flag_CR_down = 0

                  if LShoulder_Enabled==1 and flag_CL_down ==0:
                        sock.send('Caterpillar,L,DOWN\n'.encode())
                        flag_CL_dstop = 0
                        flag_CL_down = 1
                  if LShoulder_Enabled==0 and flag_CL_dstop == 0:
                        sock.send('Caterpillar,L,STOP\n'.encode())
                        flag_CL_dstop = 1
                        flag_CL_down = 0
                  if Rood_X_State==1 and Cam_X<=X_Max:
                        Cam_X=Cam_X+DeltaX
                        str_send='ROTATE,X,',Cam_X,'\n'
                        sock.send(str(str_send).encode())
                  if Rood_X_State==-1  and Cam_X>=X_Min:
                        Cam_X=Cam_X-DeltaX
                        str_send='ROTATE,X,',Cam_X,'\n'
                        sock.send(str(str_send).encode())
                  if Rood_Y_State==-1 and Cam_Y<=Y_Max:
                        Cam_Y=Cam_Y+DeltaY
                        str_send='ROTATE,Y,',Cam_Y,'\n'
                        sock.send(str(str_send).encode())
                  if Rood_Y_State==1 and Cam_Y>=Y_Min:
                        str_send='ROTATE,Y,',Cam_Y,'\n'
                        sock.send(str(str_send).encode())
      finally:
            thread1.join()

class New_Toplevel_1:
    def __init__(self, top=None):
        _bgcolor = '#d9d9d9'  # X11 color: 'gray85'
```

```
_fgcolor = '#000000'  # X11 color: 'black'
_compcolor = '#d9d9d9' # X11 color: 'gray85'
_ana1color = '#d9d9d9' # X11 color: 'gray85'
_ana2color = '#ececec' # Closest X11 color: 'gray92'
self.style = ttk.Style()
if sys.platform == "win32":
    self.style.theme_use('winnative')
self.style.configure('.',background=_bgcolor)
self.style.configure('.',foreground=_fgcolor)
self.style.configure('.',font="TkDefaultFont")
self.style.map('.',background=
    [('selected', _compcolor), ('active',_ana2color)])

top.geometry("600x450+253+264")
top.title("New Toplevel")

self.Entry1 = ttk.Entry(top)
self.Entry1.place(relx=0.05, rely=0.156,height=23, relwidth=0.277)
self.Entry1.configure(background="white")
self.Entry1.configure(font="TkFixedFont")
self.Entry1.bind('<Leave>',func=self.get_ip)

self.Button1 = ttk.Button(top)
self.Button1.place(relx=0.05, rely=0.244, height=31, width=125)
self.Button1.configure(text='''Подключение''')
self.Button1.bind('<Button-1>',func=self.connect_socket)

self.Button2 = ttk.Button(top)
self.Button2.place(relx=0.05, rely=0.333, height=31, width=166)
self.Button2.configure(text='''Завершение сеанса''')
self.Button2.bind('<Button-1>',func=self.finish_session)

self.Button3 = ttk.Button(top)
self.Button3.place(relx=0.05, rely=0.45, height=31, width=166)
self.Button3.configure(text='''Старт сканирования''')
self.Button3.bind('<Button-1>',func=self.start_lidar)

self.TCombobox1 = ttk.Combobox(top)

self.TCombobox1.place(relx=0.5, rely=0.156, relheight=0.047
        , relwidth=0.5)
self.TCombobox1['values']=['Профиль 1','Профиль 2']#dirs
self.TCombobox1.configure(textvariable=tk.StringVar())
self.TCombobox1.configure(background="#ffff80")
self.TCombobox1.configure(takefocus="")
self.TCombobox1.bind('<<ComboboxSelected>>',func=self.select_combobox)

self.Label1 = ttk.Label(top)
self.Label1.place(relx=0.067, rely=0.089, height=21, width=62)
self.Label1.configure(text='''IP-Адрес''')

self.Label2 = ttk.Label(top)
self.Label2.place(relx=0.667, rely=0.089, height=21, width=171)
self.Label2.configure(text='''Профиль геймпада''')

self.Listbox1 = tk.Listbox(top)
self.Listbox1.place(relx=0.667, rely=0.289, relheight=0.547
        , relwidth=0.29)
self.Listbox1.configure(background="white")
self.Listbox1.configure(font="TkFixedFont")
```

```
        self.Listbox1.configure(width=174)

        self.Label3 = ttk.Label(top)
        self.Label3.place(relx=0.667, rely=0.222, height=21, width=150)
        self.Label3.configure(text='''Показания датчиков''')

        self.Button3 = ttk.Button(top)
        self.Button3.place(relx=0.683, rely=0.844, height=31, width=157)
        self.Button3.configure(text='''Выход''')
        self.Button3.configure(width=157)

    def select_combobox(self,top=None):
        profile_name=self.TCombobox1.get()
        select_profile(profile_name[8])

    def connect_socket(self,top=None):
        try:
            thread1 = Thread(target=grand_thread)
            thread1.start()
        finally:
            sock.close()
    def get_ip(self,top=None):
        global ip_address
        ip_address=self.Entry1.get()
    def finish_session(self,top=None):
        global socket_status
        socket_status = 0
        sock.send('FINISH,70,150,300,Y,1.0,1.0,3\n'.encode())

    def start_lidar(self,top=None):
        thread3 = Thread(target=start_visualisation)
        thread3.start()
def main():
             root = Tk()
             top = New_Toplevel_1(root)
             root.mainloop()


if __name__ == "__main__":
    main()
```

ПРИЛОЖЕНИЕ В

ИСХОДНЫЙ КОД ПРОГРАММЫ ПРИЛОЖЕНИЯ ANDROID

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#ccf1ee"
    tools:context=".MainActivity">

    <TextureView
        android:id="@+id/cameraView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@+id/toggleButtonTorch"
        android:layout_marginBottom="0dp" />

    <TextView
        android:id="@+id/textViewProcessMsg"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:layout_margin="10dp"
        android:background="@drawable/msg_background"
        android:padding="10dp"
        android:textColor="#ffee00"
        android:visibility="invisible" />

    <Button
        android:id="@+id/buttonNetworkScanner"
        style="@style/MyButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/textViewProcessMsg"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="5dp"
        android:soundEffectsEnabled="true"
        android:text="@string/scanning_network"
        android:visibility="invisible" />

    <Button
        android:id="@+id/buttonReConnect"
        style="@style/MyButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/textViewProcessMsg"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="5dp"
        android:soundEffectsEnabled="true"
        android:text="@string/try_to_reconnecting"
        android:visibility="invisible" />

    <ProgressBar
        android:id="@+id/connectProgressBar"
        style="?android:attr/progressBarStyle"
        android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
    android:layout_centerInParent="true" />

<ImageButton
    android:id="@+id/imageButtonSnapshot"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentStart="true"
    android:layout_alignParentTop="true"
    android:adjustViewBounds="true"
    android:background="@null"
    android:padding="5dp"
    android:contentDescription="@string/snapshot"
    android:scaleType="centerInside"
    android:soundEffectsEnabled="true"
    android:src="@drawable/ic_camera_snapshot" />

<TextView
    android:id="@+id/textViewTraffic"
    android:layout_width="75dp"
    android:layout_height="wrap_content"
    android:layout_toRightOf="@+id/imageButtonSnapshot"
    android:background="#81929904"
    android:text=""
    android:textStyle="bold" />

<ImageButton
    android:id="@+id/imageButtonConfigure"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentEnd="true"
    android:layout_alignParentTop="true"
    android:background="@null"
    android:padding="5dp"
    android:contentDescription="@string/configure"
    android:src="@drawable/ic_configure" />

<RadioGroup
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:orientation="horizontal">

    <RadioButton
        android:id="@+id/radioButtonMoveCameraMode"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="10dp"
        android:checked="false"
        android:drawableEnd="@drawable/ic_move_camera_mode" />

    <RadioButton
        android:id="@+id/radioButtonEmptyScreenMode"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="10dp"
        android:checked="true"
        android:drawableEnd="@drawable/ic_empty_screen_mode" />
</RadioGroup>
```

```
    <View
        android:id="@+id/viewCamMoveHorizontal"
        android:layout_width="274dp"
        android:layout_height="32dp"
        android:layout_below="@+id/buttonNetworkScanner"
        android:layout_toEndOf="@+id/imageButtonSnapshot"
        android:background="@drawable/shape_horizontal" />

    <View
        android:id="@+id/viewCamMoveVertical"
        android:layout_width="32dp"
        android:layout_height="233dp"
        android:layout_alignParentTop="true"
        android:layout_marginTop="70dp"
        android:layout_toStartOf="@+id/imageButtonConfigure"
        android:background="@drawable/shape_vertical" />

    <ToggleButton
        android:id="@+id/toggleButtonVideoRecord"
        android:layout_width="30dp"
        android:layout_height="30dp"
        android:layout_alignParentBottom="true"
        android:layout_alignParentStart="true"
        android:background="@drawable/selector_video_recorder_button"
        android:checked="false"
        android:padding="5dp"
        android:text=""
        android:textOff=""
        android:textOn="" />

    <ImageButton
        android:id="@+id/imageButtonCloseNetwork"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentEnd="true"
        android:background="@null"
        android:padding="5dp"
        android:contentDescription="@string/close_network"
        android:src="@drawable/ic_close_network" />

</RelativeLayout>
```

MainActivity.java

```
package home.mm.vcontroller;

import android.Manifest;
import android.annotation.SuppressLint;
import android.app.Activity;
import android.app.FragmentManager;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.pm.PackageManager;
import android.graphics.Bitmap;
import android.graphics.SurfaceTexture;
import android.media.MediaCodec;
import android.media.MediaFormat;
import android.os.Build;
import android.os.Bundle;
```

```
import android.os.Environment;
import android.os.Handler;
import android.os.Looper;
import android.os.StrictMode;
import android.preference.PreferenceManager;
import android.support.v4.content.ContextCompat;
import android.util.Log;
import android.view.MotionEvent;
import android.view.Surface;
import android.view.TextureView;
import android.view.View;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.Button;
import android.widget.CompoundButton;
import android.widget.ImageButton;
import android.widget.ProgressBar;
import android.widget.RadioButton;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.ToggleButton;

import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.net.Socket;
import java.nio.ByteBuffer;
import java.util.Arrays;
import java.util.Date;

import home.mm.vcontroller.utils.ErrorInfoDialog;
import home.mm.vcontroller.utils.NetworkConnect;
import home.mm.vcontroller.utils.Settings;
import home.mm.vcontroller.utils.ViewControllerPWM;
import home.mm.vcontroller.utils.cmdSpiInterface;



//библиотеки на сенсор
import java.util.List;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import java.util.Timer;
import java.util.TimerTask;



public class MainActivity extends Activity implements TextureView.SurfaceTextureListener,
cmdSpiInterface {


    SensorManager sensorManager;
    Sensor gyroscopeSensor;
    SensorEventListener gyroscopeSensorListener;

        public enum CMD {
```

```
        START_STREAM,
        ROTATE,
        FINISH,
        AUTH,
        SPI
    }

    private final static String LOG_TAG = "Main";
    private static final Object socketlock = new Object();
    private static final Object recordVideoLock = new Object();
    private Handler mAppHandler;
    private Socket socket = null;

    private TextView mTextViewProcessMsg, mTextViewTraffic;
    private Button mBtnReconnect, mBtnNetworkScanner;
    private ImageButton mBtnSnapshot, mBtnCloseNetwork;
    private ProgressBar mConnectProgressBar;
    private RadioButton mRadioButtonMoveCameraMode, mRadioButtonEmptyScreen;
    private ToggleButton mToggleButtonVideoRecording, mToggleButtonTorch;
    private View mViewCamMoveVertical, mViewCamMoveHorizontal;

    private TextureView mCameraView;
    private MediaCodec mMediaCodec = null;
    private volatile boolean isMediaCodecStarted = false;
    private Surface mSurface = null;
    private int surfaceTextureWidth = 1920, surfaceTextureHeight = 1080;
    private DecoderInputThread decoderInputThread;
    private DecoderOutputThread decoderOutputThread;
    private BufferedOutputStream mOutputStreamVideoRecord;

    private ViewControllerPWM steeringLeft = new ViewControllerPWM((byte) 1, this);
    private ViewControllerPWM steeringRight = new ViewControllerPWM((byte) 0, this);

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Log.v(LOG_TAG, "-------- onCreate()");

        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            // Intent(Intent.ACTION_VIEW); dirty way
            StrictMode.VmPolicy.Builder builder = new StrictMode.VmPolicy.Builder();
            StrictMode.setVmPolicy(builder.build());
        }

        if                          (ContextCompat.checkSelfPermission(MainActivity.this,
Manifest.permission.WRITE_EXTERNAL_STORAGE) != PackageManager.PERMISSION_GRANTED) {
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
                requestPermissions(new
String[]{Manifest.permission.WRITE_EXTERNAL_STORAGE}, 1);
            }
        }

        final     Animation     btnAnimation     =     AnimationUtils.loadAnimation(this,
R.anim.anumation4button);
        setContentView(R.layout.activity_main);
        mAppHandler = new Handler(Looper.getMainLooper());
        mCameraView = findViewById(R.id.cameraView);
        mCameraView.setSurfaceTextureListener(this);
        mTextViewProcessMsg = findViewById(R.id.textViewProcessMsg);
        mTextViewTraffic = findViewById(R.id.textViewTraffic);
```

```
        mConnectProgressBar = findViewById(R.id.connectProgressBar);
        mBtnReconnect = findViewById(R.id.buttonReConnect);
        mBtnReconnect.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                v.startAnimation(btnAnimation);
                setTryConnectionState();
            }
        });
        mBtnNetworkScanner = findViewById(R.id.buttonNetworkScanner);
        mBtnNetworkScanner.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                v.startAnimation(btnAnimation);
                setNetworkScannerState();
            }
        });
        mBtnSnapshot = findViewById(R.id.imageButtonSnapshot);
        mBtnSnapshot.setOnClickListener(new View.OnClickListener() {//скриншот
            @SuppressLint("NewApi")
            @Override
            public void onClick(View v) {
                String fname = "VMM-" + android.text.format.DateFormat.format("yyyy-MM-
dd_hh-mm-ss", new Date()) + ".png";
                try {
                    File                            dirPictures                            =
Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES);
                    File dir = new File(dirPictures, getString(R.string.app_name));
                    if (!dir.exists()) //noinspection ResultOfMethodCallIgnored
                        dir.mkdirs();

                    Bitmap bitmap = mCameraView.getBitmap();
                    File f = new File(dir, fname);
                    OutputStream os = new FileOutputStream(f);
                    fname = f.getAbsolutePath();
                    bitmap.compress(Bitmap.CompressFormat.PNG, 100, os);
                    os.close();
                    Toast.makeText(MainActivity.this, fname, Toast.LENGTH_SHORT).show();
                } catch (Exception e) {
                    Toast.makeText(MainActivity.this,  String.format("Save  file  error.
[%s]: %s", fname,
                            e.getMessage()), Toast.LENGTH_LONG).show();
                }
            }
        });
        ImageButton mBtnConfigure = findViewById(R.id.imageButtonConfigure);
        mBtnConfigure.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(MainActivity.this, SettingsActivity.class);
                intent.putExtra("width", surfaceTextureWidth);
                intent.putExtra("height", surfaceTextureHeight);
                startActivity(intent);
            }
        });

        mRadioButtonMoveCameraMode = findViewById(R.id.radioButtonMoveCameraMode);
        mRadioButtonMoveCameraMode.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
```

```
            @Override
            public void onCheckedChanged(CompoundButton compoundButton, boolean b) {
                int st = b ? View.VISIBLE : View.INVISIBLE;
                mViewCamMoveVertical.setVisibility(st);
                mViewCamMoveHorizontal.setVisibility(st);
            }
        });
        mViewCamMoveVertical = findViewById(R.id.viewCamMoveVertical);
        mViewCamMoveVertical.setOnTouchListener(new View.OnTouchListener() {
            @SuppressLint("ClickableViewAccessibility")
            @Override
            public boolean onTouch(View view, MotionEvent motionEvent) {
                if (motionEvent.getAction() == MotionEvent.ACTION_DOWN) {
                    float p = 1 - (motionEvent.getY() / view.getHeight()) * 2;
                    sendCmd(CMD.ROTATE + ",V," + p + "\n");
                }
                return true;
            }
        });

        mViewCamMoveHorizontal = findViewById(R.id.viewCamMoveHorizontal);
        mViewCamMoveHorizontal.setOnTouchListener(new View.OnTouchListener() {
            @SuppressLint("ClickableViewAccessibility")
            @Override
            public boolean onTouch(View view, MotionEvent motionEvent) {
                if (motionEvent.getAction() == MotionEvent.ACTION_DOWN) {
                    float p = (motionEvent.getX() / view.getWidth()) * 2 - 1;
                    sendCmd(CMD.ROTATE + ",H," + p + "\n");

                }
                return true;
            }
        });

        mRadioButtonEmptyScreen = findViewById(R.id.radioButtonEmptyScreenMode);

        final    Animation    btnRecordAnimation    =    AnimationUtils.loadAnimation(this,
R.anim.anumation_video_recording);
        mToggleButtonVideoRecording = findViewById(R.id.toggleButtonVideoRecord);
        mToggleButtonVideoRecording.setChecked(false);
        mToggleButtonVideoRecording.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(CompoundButton compoundButton, boolean state) {
                if (state) {
                    compoundButton.startAnimation(btnRecordAnimation);
                    closeVideoRecordingStream();
                    String fname = "VMM-" + android.text.format.DateFormat.format("yyyy-
MM-dd_hh-mm-ss", new Date()) + ".mov";
                    try {
                        File                          dirPictures                          =
Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES);
                        File dir = new File(dirPictures, getString(R.string.app_name));
                        if (!dir.exists())
                            dir.mkdirs();
                        File f = new File(dir, fname);
                        fname = f.getAbsolutePath();
                        mOutputStreamVideoRecord    =    new    BufferedOutputStream(new
FileOutputStream(f));
                    } catch (Exception e) {
```

```
                        Toast.makeText(MainActivity.this, String.format("Save file error.
[%s]: %s", fname,
                                e.getMessage()), Toast.LENGTH_LONG).show();
                    }
                } else {
                    closeVideoRecordingStream();
                    compoundButton.clearAnimation();
                }
            }
        });

//=============================================================================================

        mBtnCloseNetwork = findViewById(R.id.imageButtonCloseNetwork);
        mBtnCloseNetwork.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                closeVideoRecordingStream();
                socketClose();
            }
        });

        //==========================датчики===============
        super.onCreate(savedInstanceState);

        sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
        gyroscopeSensor =
                sensorManager.getDefaultSensor(Sensor.TYPE_GYROSCOPE);
        gyroscopeSensorListener = new SensorEventListener() {

            @Override
            public void onSensorChanged(SensorEvent sensorEvent) {
                // More code goes here
                if (sensorEvent.values[0]> 0.2f){
                    float p = -sensorEvent.values[0];
                    sendCmd(CMD.ROTATE + ",H," + p + "\n");// отправляем сообщение об
поворотe камеры по горизонтали
                } else if (sensorEvent.values[0] < -0.2){
                    float p = -sensorEvent.values[0];
                    sendCmd(CMD.ROTATE + ",H," + p + "\n");// отправляем сообщение об
поворотe камеры по горизонтали
                }
            }

            @Override
            public void onAccuracyChanged(Sensor sensor, int i) {
            }

        };
    }

    @Override
    protected void onPause() {
        isMediaCodecStarted = false;
        Log.v(LOG_TAG, "-------- onPause()");
        if (decoderInputThread != null) {
            decoderInputThread.interrupt();
            decoderInputThread = null;
        }
        if (decoderOutputThread != null) {
```

```
            decoderOutputThread.interrupt();
            decoderOutputThread = null;
        }
        if (mMediaCodec != null) {
            try {
                mMediaCodec.stop();
                mMediaCodec.release();
                mMediaCodec = null;
            } catch (Exception ex) {
                Log.e(LOG_TAG, "", ex);
            }
        }
        closeVideoRecordingStream();
        socketClose();

        super.onPause();

        sensorManager.unregisterListener(gyroscopeSensorListener);
    }

    @Override
    public void onStop() {
        super.onStop();
        Log.v(LOG_TAG, "-------- onStop()");
    }

    @Override
    public void onResume() {
        super.onResume();
        Log.v(LOG_TAG, "-------- onResume()");
        try {
            SharedPreferences                              settings                              =
PreferenceManager.getDefaultSharedPreferences(this);
            Settings.loadConfig(settings);
            Settings.saveConfig(settings); // сохранить значения по умолчанию
            mMediaCodec = MediaCodec.createDecoderByType("video/avc");
        } catch (final Exception e) {
            Log.e(LOG_TAG, "", e);
            mAppHandler.post(new Runnable() {
                @Override
                public void run() {
                    FragmentManager fm = getFragmentManager();
                    new ErrorInfoDialog().showInfo(fm, e.toString());
                }
            });
        }
        setTryConnectionState();
        (decoderInputThread = new DecoderInputThread()).start();
        (decoderOutputThread = new DecoderOutputThread()).start();


sensorManager.registerListener(gyroscopeSensorListener,gyroscopeSensor,SensorManager.SENS
OR_DELAY_FASTEST);
    }


    private NetworkConnect.Callback networkCallback = new NetworkConnect.Callback() {

        @Override
        public void result(Socket s, String err) {
```

```
            Log.e(LOG_TAG, "NetworkConnect.Callback ");
            if (err == null) {
                setConnectionReadyState(s);
                try {

Settings.saveConfig(PreferenceManager.getDefaultSharedPreferences(MainActivity.this));
                } catch (Exception e) {
                    Log.e(LOG_TAG, "load config.", e);
                    final String msg = e.toString();
                    mAppHandler.post(new Runnable() {
                        @Override
                        public void run() {
                            FragmentManager fm = getFragmentManager();
                            new ErrorInfoDialog().showInfo(fm, msg);
                        }
                    });
                }
            } else setNoConnectionState(err);
        }

        @Override
        public void showProgress(String msg) {
            mTextViewProcessMsg.setText(msg);
        }
    };

    private void setScreenState(boolean isVideoStreamMode) {
        int st = isVideoStreamMode ? View.VISIBLE : View.INVISIBLE;
        mBtnSnapshot.setVisibility(st);
        mBtnCloseNetwork.setVisibility(st);
        mToggleButtonVideoRecording.setVisibility(st);
        mTextViewTraffic.setVisibility(st);
        mRadioButtonMoveCameraMode.setVisibility(st);
        mRadioButtonEmptyScreen.setVisibility(st);



        if (isVideoStreamMode)
            st = mRadioButtonMoveCameraMode.isChecked() ? View.VISIBLE : View.INVISIBLE;
        mViewCamMoveVertical.setVisibility(st);//оборажение слайдера поворота камеры
        mViewCamMoveHorizontal.setVisibility(st);

        st = isVideoStreamMode ? View.INVISIBLE : View.VISIBLE;
        mTextViewProcessMsg.setVisibility(st);
        mBtnReconnect.setVisibility(st);
        mBtnNetworkScanner.setVisibility(st);
        mConnectProgressBar.setVisibility(st);
    }

    private void setNetworkScannerState() {
        socketClose();
        setScreenState(false);
        mBtnReconnect.setVisibility(View.INVISIBLE);
        mBtnNetworkScanner.setVisibility(View.INVISIBLE);
        new NetworkConnect(true, networkCallback).execute();
    }

    private void setNoConnectionState(String msg) {
        Log.v(LOG_TAG, "setNoConnectionState:" + msg);
        setScreenState(false);
```

```
        mTextViewProcessMsg.setText(msg);
        mConnectProgressBar.setVisibility(View.INVISIBLE);
        if                                              (!Settings.isInternalNetwork)
mBtnNetworkScanner.setVisibility(View.INVISIBLE);
    }

    @SuppressLint("DefaultLocale")
    private void setConnectionReadyState(Socket s) {
        socket = s;
        sendCmd(String.format("%s,%s\n", CMD.AUTH.name(), Settings.password));
        sendCmd(String.format("%s,%d,%d,%d,%d,%d,%d\n", CMD.START_STREAM.name(),
                (int) (surfaceTextureWidth / Settings.resolutionDiv),
                (int) (surfaceTextureHeight / Settings.resolutionDiv),
                Settings.vFps,
                Settings.vBitrate,
                Settings.vQuality,
                Settings.vISO
        ));
        setScreenState(true);
        Toast.makeText(MainActivity.this,
String.format(getString(R.string.toast_connect_info),
                Settings.isInternalNetwork        ?        Settings.internalHost        :
Settings.externalHost, Settings.tcpIpPort), Toast.LENGTH_LONG).show();
    }

    private void setTryConnectionState() {
        socketClose();
        setScreenState(false);
        mBtnReconnect.setVisibility(View.INVISIBLE);
        mBtnNetworkScanner.setVisibility(View.INVISIBLE);
        mTextViewProcessMsg.setVisibility(View.INVISIBLE);
        new NetworkConnect(false, networkCallback).execute();
    }

    @Override
    public void sendCmdSPI(final byte[] cmd) {
        if (socket != null)
            new Thread(new Runnable() {
                @Override
                public void run() {
                    try {
                        StringBuilder                s                =                new
StringBuilder(CMD.SPI.name()).append(",n,").append(Settings.cmdSpiMagicBytes);
                        String hh = "0123456789ABCDEF";
                        for (byte b : cmd)
                            s.append(hh.charAt((((int)      b)      &      0x0F0)      >>
4)).append(hh.charAt(((int) b) & 0x0F));
                        s.append('\n');
                        synchronized (socketlock) {
                            socket.getOutputStream().write(s.toString().getBytes());
                            socket.getOutputStream().flush();
                        }
                    } catch (IOException e) {
                        Log.e(LOG_TAG, "sendCmdSPI", e);
                    }
                }
            }).start();
    }

    private void sendCmd(final String cmd) {
```

```
    if (socket != null)
        new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    synchronized (socketlock) {
                        socket.getOutputStream().write(cmd.getBytes());
                        socket.getOutputStream().flush();
                    }
                } catch (final IOException e) {
                    Log.e(LOG_TAG, "socket write cmd");
                    socketClose();
                    mAppHandler.post(new Runnable() {
                        @Override
                        public void run() {
                            setNoConnectionState(e.getMessage());
                        }
                    });
                }

            }
        }).start();
}

private void socketClose() {
    if (socket != null) {
        new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    synchronized (socketlock) {
                        socket.getOutputStream().write((CMD.FINISH.name() +
                                "\n").getBytes());
                    }
                    try {
                        Thread.sleep(500);
                    } catch (InterruptedException ignore) {
                    }
                    socket.close();
                } catch (Exception ignored) {
                }
                socket = null;
            }
        }).start();
    }
}

private void closeVideoRecordingStream() {
    if (mOutputStreamVideoRecord != null) {
        try {
            synchronized (recordVideoLock) {
                mOutputStreamVideoRecord.flush();
                mOutputStreamVideoRecord.close();
                mOutputStreamVideoRecord = null;
            }
        } catch (Exception ex) {
            Log.e(LOG_TAG, "", ex);
        }
    }
}
```

```
    public boolean withTorch() {
        return mToggleButtonTorch.isChecked();
    }

    @Override
    public void onSurfaceTextureAvailable(SurfaceTexture surfaceTexture, int width, int
height) {
        this.mSurface = new Surface(surfaceTexture);
        surfaceTextureWidth = width;
        surfaceTextureHeight = height;
        Log.d(LOG_TAG, "onSurfaceTextureAvailable width:" + width + " height:" + height);
    }

    @Override
    public void onSurfaceTextureSizeChanged(SurfaceTexture surfaceTexture, int width, int
height) {
    }

    @Override
    public boolean onSurfaceTextureDestroyed(SurfaceTexture surfaceTexture) {
        return false;
    }

    @Override
    public void onSurfaceTextureUpdated(SurfaceTexture surfaceTexture) {
    }

    private class DecoderOutputThread extends Thread {
        private MediaCodec.BufferInfo mBufferInfo = new MediaCodec.BufferInfo();

        @Override
        public void run() {
            while (!isInterrupted()) {
                if (isMediaCodecStarted) {
                    try {
                        int status = mMediaCodec.dequeueOutputBuffer(mBufferInfo, 0);
                        if (status >= 0) {
                            long       dt       =       System.currentTimeMillis()       -
mBufferInfo.presentationTimeUs;
                            if (dt > 150) Log.v(LOG_TAG, "decoder delay:" + dt);
                            mMediaCodec.releaseOutputBuffer(status, true);
                        }
                    } catch (Exception ex) {
                        Log.e(LOG_TAG, "", ex);
                    }
                }
            }
        }
    }

    private class DecoderInputThread extends Thread {
        private final static int NAL_SIZE_INC = 4096;
        private byte[] nalBuff = new byte[4096 * 2];
        private byte[] inpBuff = new byte[4096];
        int nalSz = 0;
        int numZeroes = 0;
        private boolean trafficMsgBlink = true;

        @SuppressLint("SetTextI18n")
        @Override
```

```
        public void run() {
            try {
                long bytesTraffic = 0;
                long timeTraffic = System.currentTimeMillis();
                while (!isInterrupted()) {
                    if (socket == null || mSurface == null) {
                        try {
                            Thread.sleep(50);
                        } catch (InterruptedException ignore) {
                        }
                        continue;
                    }

                    int sz;
                    try {
                        sz = socket.getInputStream().read(inpBuff);
                        if (sz < 0) throw new IOException("Сокет закрыт");
                    } catch (final IOException e) {
                        Log.e(LOG_TAG, "read from InputStream");
                        socketClose();
                        mAppHandler.post(new Runnable() {
                            @Override
                            public void run() {
                                setNoConnectionState(e.getMessage());
                            }
                        });
                        continue;
                    }

                    bytesTraffic += sz;
                    long t = System.currentTimeMillis();
                    if (t - timeTraffic > 1000) {
                        long tmp = (bytesTraffic + 500) * 1000L;
                        trafficMsgBlink = !trafficMsgBlink;
                        final String msg = (trafficMsgBlink ? '□' : '▣') +
Long.toString((tmp / (t - timeTraffic)) / 1000L) + " kb/s";
                        mAppHandler.post(new Runnable() {
                            @Override
                            public void run() {
                                mTextViewTraffic.setText(msg);
                            }
                        });

                        bytesTraffic = 0;
                        timeTraffic = System.currentTimeMillis();
                    }

                    for (int i = 0; i < sz && !isInterrupted(); i++) {
                        if (nalSz == nalBuff.length)
                            nalBuff = Arrays.copyOf(nalBuff, nalBuff.length +
NAL_SIZE_INC);
                        nalBuff[nalSz++] = inpBuff[i];
                        if (inpBuff[i] == 0) numZeroes++;
                        else {
                            if (inpBuff[i] == 1 && numZeroes == 3) {
                                if (nalSz > 4) {
                                    if (!decodeNAL()) {
                                        nalBuff[0] = nalBuff[1] = nalBuff[2] = 0;
                                        nalBuff[3] = 1;
                                    }
```

```
                            }
                            nalSz = 4;
                        }
                        numZeroes = 0;
                    }
                }
            }
        } catch (Exception ex) {
            Log.e(LOG_TAG, "decoder", ex);
        }
    }


    private boolean decodeNAL() {
        if (nalSz < 8 || nalBuff[0] != 0 || nalBuff[1] != 0 && nalBuff[2] != 0 &&
nalBuff[3] != 1)
            return false;
        int type = nalBuff[4] & 0x1F;
        if (type == 7) {
            if (!isMediaCodecStarted && mMediaCodec != null) {
                MediaFormat mediaFormat = MediaFormat.createVideoFormat("video/avc",
surfaceTextureWidth, surfaceTextureHeight);
                Log.v(LOG_TAG, "MediaFormat:" + mediaFormat);
                mMediaCodec.configure(mediaFormat, mSurface, null, 0);
                mMediaCodec.start();
                isMediaCodecStarted = true;
            }
        }

        if (type > 0 && isMediaCodecStarted && mMediaCodec != null) {
            int i = mMediaCodec.dequeueInputBuffer(0);
            if (i >= 0) {
                ByteBuffer inputBuffer = mMediaCodec.getInputBuffer(i);
                if (inputBuffer != null) {
                    inputBuffer.put(nalBuff, 0, nalSz - 4);
                    long t = System.currentTimeMillis();
                    try {
                        if (mOutputStreamVideoRecord != null) {
                            synchronized (recordVideoLock) {
                                mOutputStreamVideoRecord.write(nalBuff, 0, nalSz -
4);
                            }
                        }
                    } catch (IOException e) {
                        Log.e(LOG_TAG, "", e);
                    }
                    mMediaCodec.queueInputBuffer(i, 0, nalSz - 4, t, 0);
                }
            }
        }
        return true;
    }
}
```

SettingsActivity.java

```
package home.mm.vcontroller;
```

```
import android.annotation.TargetApi;
import android.content.Intent;
import android.content.res.Configuration;
import android.os.Build;
import android.os.Bundle;
import android.preference.ListPreference;
import android.preference.Preference;
import android.preference.PreferenceActivity;
import android.app.ActionBar;
import android.preference.PreferenceFragment;
import android.preference.PreferenceManager;
import android.preference.PreferenceScreen;
import android.util.Log;
import android.view.MenuItem;
import android.support.v4.app.NavUtils;

import java.util.ArrayList;
import java.util.List;

public class SettingsActivity extends PreferenceActivity {
    private static int surfaceTextureWidth = 1920, surfaceTextureHeight = 1080;

    private                  static                  Preference.OnPreferenceChangeListener
sBindPreferenceSummaryToValueListener = new Preference.OnPreferenceChangeListener() {
        @Override
        public boolean onPreferenceChange(Preference preference, Object value) {
            String stringValue = value.toString();

            if (preference instanceof ListPreference) {
                Log.v("-----------------", "ListPreference stringValue:" + stringValue);
                ListPreference listPreference = (ListPreference) preference;
                int index = listPreference.findIndexOfValue(stringValue);

                preference.setSummary(
                        index >= 0
                                ? listPreference.getEntries()[index]
                                : null);

            } else {
                preference.setSummary(stringValue);
```

```
        }
        return true;
    }
};

private static void bindPreferenceSummaryToValue(Preference preference) {
    preference.setOnPreferenceChangeListener(sBindPreferenceSummaryToValueListener);
    sBindPreferenceSummaryToValueListener.onPreferenceChange(preference,
            PreferenceManager
                    .getDefaultSharedPreferences(preference.getContext())
                    .getString(preference.getKey(), ""));
}

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    ActionBar actionBar = getActionBar();
    if (actionBar != null)
        actionBar.setDisplayHomeAsUpEnabled(true);
    Intent intent = getIntent();
    int v = intent.getIntExtra("width", -1);
    surfaceTextureWidth = v < 0 ? surfaceTextureWidth : v;
    v = intent.getIntExtra("height", -1);
    surfaceTextureHeight = v < 0 ? surfaceTextureHeight : v;
}


@Override
public boolean onMenuItemSelected(int featureId, MenuItem item) {
    int id = item.getItemId();
    if (id == android.R.id.home) {
        if (!super.onMenuItemSelected(featureId, item)) {
            NavUtils.navigateUpFromSameTask(this);
        }
        return true;
    }
    return super.onMenuItemSelected(featureId, item);
}

@Override
```

```
    public boolean onIsMultiPane() {
        return (this.getResources().getConfiguration().screenLayout
                &                 Configuration.SCREENLAYOUT_SIZE_MASK)                 >=
Configuration.SCREENLAYOUT_SIZE_XLARGE;
    }

    @Override
    @TargetApi(Build.VERSION_CODES.HONEYCOMB)
    public void onBuildHeaders(List<Header> target) {
        loadHeadersFromResource(R.xml.pref_headers, target);
    }

    protected boolean isValidFragment(String fragmentName) {
        return PreferenceFragment.class.getName().equals(fragmentName)
                || NetworkPreferenceFragment.class.getName().equals(fragmentName)
                || CameraPreferenceFragment.class.getName().equals(fragmentName)
                || MotionPreferenceFragment.class.getName().equals(fragmentName)
                || ExternalSpiPreferenceFragment.class.getName().equals(fragmentName);
    }

    @TargetApi(Build.VERSION_CODES.HONEYCOMB)
    public static class NetworkPreferenceFragment extends PreferenceFragment {
        @Override
        public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            addPreferencesFromResource(R.xml.pref_network);
            setHasOptionsMenu(true);

bindPreferenceSummaryToValue(findPreference(getString(R.string.cfg_connectTimeout)));

bindPreferenceSummaryToValue(findPreference(getString(R.string.cfg_tcpIpPort)));

bindPreferenceSummaryToValue(findPreference(getString(R.string.cfg_internalHost)));

bindPreferenceSummaryToValue(findPreference(getString(R.string.cfg_externalHost)));

bindPreferenceSummaryToValue(findPreference(getString(R.string.cfg_password)));
        }

        @Override
```

```
        public boolean onOptionsItemSelected(MenuItem item) {
            int id = item.getItemId();
            if (id == android.R.id.home) {
                startActivity(new Intent(getActivity(), SettingsActivity.class));
                return true;
            }
            return super.onOptionsItemSelected(item);
        }
    }

    @TargetApi(Build.VERSION_CODES.HONEYCOMB)
    public static class CameraPreferenceFragment extends PreferenceFragment {
        @Override
        public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            addPreferencesFromResource(R.xml.pref_camera);
            setHasOptionsMenu(true);

bindPreferenceSummaryToValue(findPreference(getString(R.string.cfg_cam_fps)));

bindPreferenceSummaryToValue(findPreference(getString(R.string.cfg_cam_quality)));

bindPreferenceSummaryToValue(findPreference(getString(R.string.cfg_cam_bitrate)));
            bindPreferenceSummaryToValue(findPreference(getString(R.string.cfg_vISO)));
            //--------------
            ListPreference            preference            =            (ListPreference)
findPreference(getString(R.string.cfg_cam_vDivider));
            List<String> values = new ArrayList<>();
            List<String> dividers = new ArrayList<>();
            for (double i = 5; i >= 0; i -= 0.5) {
                int h = (int) (surfaceTextureHeight / i);
                if (h >= 480 && h <= 1080) {
                    dividers.add(Double.toString(i));
                    values.add(Integer.toString((int) (surfaceTextureWidth / i)) + "x" +
Integer.toString(h));
                }
            }
            preference.setEntries(values.toArray(new String[values.size()]));
            preference.setEntryValues(dividers.toArray(new String[dividers.size()]));
            bindPreferenceSummaryToValue(preference);
```

```
        }
        @Override
        public boolean onOptionsItemSelected(MenuItem item) {
            int id = item.getItemId();
            if (id == android.R.id.home) {
                startActivity(new Intent(getActivity(), SettingsActivity.class));
                return true;
            }
            return super.onOptionsItemSelected(item);
        }
    }

    @TargetApi(Build.VERSION_CODES.HONEYCOMB)
    public static class MotionPreferenceFragment extends PreferenceFragment {
        @Override
        public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);

            setHasOptionsMenu(true);

bindPreferenceSummaryToValue(findPreference(getString(R.string.cfg_motionSnapshotFileMask
)));

bindPreferenceSummaryToValue(findPreference(getString(R.string.cfg_motionCheckPollingDt))
);

bindPreferenceSummaryToValue(findPreference(getString(R.string.cfg_motionThresholdValue))
);

bindPreferenceSummaryToValue(findPreference(getString(R.string.cfg_motionThresholdCnt)));

bindPreferenceSummaryToValue(findPreference(getString(R.string.cfg_motionThresholdSAD)));

bindPreferenceSummaryToValue(findPreference(getString(R.string.cfg_modeSnapshot)));


bindPreferenceSummaryToValue(findPreference(getString(R.string.cfg_motionSnapshotDt)));

bindPreferenceSummaryToValue(findPreference(getString(R.string.cfg_motionDetectTimeWindow
)));
```

```
bindPreferenceSummaryToValue(findPreference(getString(R.string.cfg_motionNumberInTimeWind
ow)));

            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
                PreferenceScreen screen = getPreferenceScreen();

screen.removePreference(findPreference(getString(R.string.cfg_hasNotificationsSound)));

screen.removePreference(findPreference(getString(R.string.cfg_notifications_ringtone)));

screen.removePreference(findPreference(getString(R.string.cfg_vibrateMotionNotification))
);
            }
        }

        @Override
        public boolean onOptionsItemSelected(MenuItem item) {
            int id = item.getItemId();
            if (id == android.R.id.home) {
                startActivity(new Intent(getActivity(), SettingsActivity.class));
                return true;
            }
            return super.onOptionsItemSelected(item);
        }
    }

    @TargetApi(Build.VERSION_CODES.HONEYCOMB)
    public static class ExternalSpiPreferenceFragment extends PreferenceFragment {
        @Override
        public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            addPreferencesFromResource(R.xml.pref_external_spi);
            setHasOptionsMenu(true);

bindPreferenceSummaryToValue(findPreference(getString(R.string.cfg_externalControllerBatt
eryVoltageAlarmLevel)));
        }
        @Override
        public boolean onOptionsItemSelected(MenuItem item) {
            int id = item.getItemId();
```

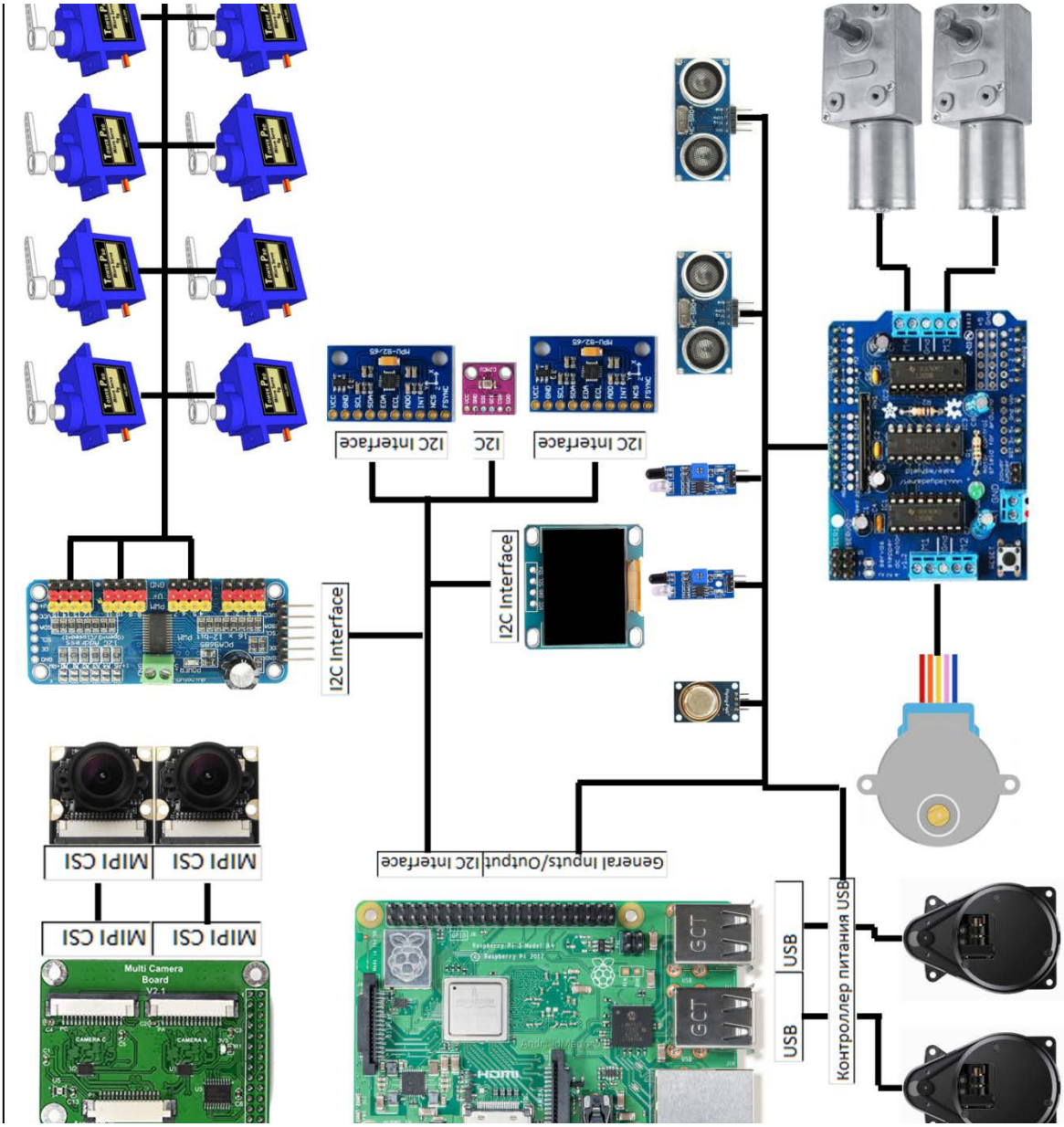
```
            if (id == android.R.id.home) {
                startActivity(new Intent(getActivity(), SettingsActivity.class));
                return true;
            }
            return super.onOptionsItemSelected(item);
        }
    }
}
```

Settings.java

```
package home.mm.vcontroller.utils;

import android.content.SharedPreferences;
import android.os.Environment;

import java.lang.reflect.Field;
import java.lang.reflect.Modifier;

public class Settings {
    public final static int MODE_SNAPSHOT_NO_MOTION_VECTORS = 0;
    public final static int MODE_SNAPSHOT_MOTION_VECTORS_DRAW_VECTORS = 1;

    public static boolean isInternalNetwork = true;
    public static int vFps = 15, vBitrate = 500000, vQuality = 0, vISO = 0;
    public static double resolutionDiv = 2;

    public static String internalHost = "192.168.0.0.77";
    public static String externalHost = "ans42.ru";
    public static int tcpIpPort = 8081;
    public static int connectTimeout = 3000;
    public static String password = "qscfew";

    public static boolean startCheckMotionService = true;
    public static boolean vibrateMotionNotification = false;
    public static boolean loadMotionSnapshot = true;
    public static String motionSnapshotFileMask = "dd-MM-yy_hh-mm-ss";
    public            static            String            notificationsRingtone           =
"content://settings/system/notification_sound";
    public static boolean hasNotificationsSound = false;
    public static int motionCheckPollingDt = 10;
    public static int motionThresholdCnt = 150;
    public static int motionThresholdValue = 70;
    public static int motionThresholdSAD = 300;
    public static boolean motionNoiseFilter = true;
    public static float motionSnapshotDt = 1.0f;
    public static float motionDetectTimeWindow = 1.0f;
    public static int motionNumberInTimeWindow = 3;

    public static boolean snapshotColorless = false;
    public static int modeSnapshot = MODE_SNAPSHOT_NO_MOTION_VECTORS;
    public static String cmdSpiMagicBytes = "666F";
    public static boolean hasExternalControllerSPI = true;
    public static boolean checkExternalControllerBatteryVoltage = true;
    public static float externalControllerBatteryVoltageAlarmLevel = 3.5f * 4;
```

```
    public static void loadConfig(SharedPreferences settings) throws Exception {
        Class aClass = Settings.class;
        Field[] fields = aClass.getFields();
        for (Field field : fields) {
            if ((field.getModifiers() & Modifier.FINAL) != 0) continue;
            String name = field.getName();
            String type = field.getType().getCanonicalName();
            if (type.equals("int")) {
                field.set(null,                 Integer.parseInt(settings.getString(name,
Integer.toString(field.getInt(aClass)))));
            } else if (type.equals("double")) {
                field.set(null,               Double.parseDouble(settings.getString(name,
Double.toString(field.getDouble(aClass)))));
            } else if (type.equals("float")) {
                field.set(null,                 Float.parseFloat(settings.getString(name,
Float.toString(field.getFloat(aClass)))));
            } else if (type.contains("java.lang.String")) {
                String s = settings.getString(name, field.get(aClass).toString());
                if (s != null && !s.isEmpty())
                    field.set(null, s);
            } else if (type.equals("boolean")) {
                field.set(null, settings.getBoolean(name, field.getBoolean(aClass)));
            }
        }
    }

    public static void saveConfig(SharedPreferences settings) throws Exception {
        SharedPreferences.Editor editor = settings.edit();
        Class aClass = Settings.class;
        Field[] fields = Settings.class.getFields();
        for (Field field : fields) {
            String name = field.getName();
            String type = field.getType().getCanonicalName();
            if (type.equals("int")) {
                editor.putString(name, Integer.toString(field.getInt(aClass)));
            } else if (type.equals("double")) {
                editor.putString(name, Double.toString(field.getDouble(aClass)));
            } else if (type.equals("float")) {
                editor.putString(name, Float.toString(field.getFloat(aClass)));
            } else if (type.contains("java.lang.String")) {
                editor.putString(name, field.get(aClass).toString());
            } else if (type.equals("boolean")) {
                editor.putBoolean(name, field.getBoolean(aClass));
            }
        }
        editor.apply();
    }
}
```

ПРИЛОЖЕНИЕ Г



ПРИЛОЖЕНИЕ Д