МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ»
Кафедра «IT – инжиниринг»

ДОПУЩЕН К ЗАЩИТЕ
Заведующий кафедрой
PhD, доцент
_____ Т.С. Картбаев
«____» _____ 2019 г.

ДИПЛОМНЫЙ ПРОЕКТ

На тему: Разработка системы «умный чемодан»

Специальность 5В070400 – «Вычислительная техника и программное обеспечение»

Выполнил: Усербаев Асет Максатович Группа: ВТ-15-2
Научный руководитель: к.т.н., доцент Мусапирова Г. Д.

Консультанты:
по экономической части: к.э.н., профессор _____ Ж. Г. Аренбаева
«13» мая 2019 г.

по безопасности
жизнедеятельности: д.т.н., ст. преп. _____ Ш.Ш. Бекбасаров
«14» мая 2019 г.

по применению
вычислительной техники: ст. преп. _____ М.Н. Майкотов
«14» мая 2019 г.

Нормоконтролер: ст. преп. _____ А.А. Айткazина
«15» мая 2019 г.

Рецензент: зам. дир. ТОО «Сайман» _____ А.С. Алиев
«____» _____ 2019 г.

Алматы 2019

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ»

Институт систем управления и информационных технологий

Кафедра «IT инжиниринг»

Специальность 5В070400 – «Вычислительная техника и программное обеспечение»

ЗАДАНИЕ
на выполнение дипломного проекта

Студенту Усербаеву Асету Максатовичу

Тема проекта: Разработка системы «умный чемодан»

Утверждена приказом по университету № 124 от «26» октября 2018 г.

Срок сдачи законченного проекта «24» мая 2019 г.

Исходные данные к проекту (требуемые параметры результатов исследования (проектирования) и исходные данные объекта): информационные материалы о проектировании систем; международные стандарты ИСО-9001; данные преддипломной практики.

Перечень вопросов, подлежащих разработке в дипломном проекте, или краткое содержание дипломного проекта:

- аналитическая часть;
- проектирование системы;
- практическая часть;
- экономическая оценка системы;
- безопасность жизнедеятельности, расчет шума на заводе;
- приложение А. Техническое задание;
- приложение Б. Листинг программы;
- приложение В. Акт внедрения.

Перечень графического материала (с точным указанием обязательных чертежей): 46 иллюстраций, 15 таблиц.

Основная рекомендуемая литература:

1 Петин В.А., Биняковский А.А. Практическая энциклопедия Arduino. – М.: ДМК Пресс, 2017. – 152 с.

2 Орлов П.И. Основы конструирования: Справочник: В 2-х книгах. — М.: Машиностроение, 2011.

3 Блум Дж. Изучаем Arduino: инструменты и методы технического волшебства: Пер. с англ. – Спб.: БХВ-Петербург, 2015. – 366 с.

4 Белов А.В. Создаем устройства на микроконтроллерах. – Спб.: Наука и техника, 2007. – 304 с.

Консультации по проекту с указанием относящихся к ним разделов проекта:

Раздел	Консультант	Сроки	Подпись
Экономическая часть	Аренбаева Ж.Г.	04.03-13.05.19	
Безопасность жизнедеятельности	Бекбасаров Ш.Ш.	12.02.2019 14.05.2019	
Программное обеспечение	Майкотов М.Н.	02.05 - 14.05.2019	
Нормоконтролер	Айтказина А.А.	02.04-15.05.2019	

ГРАФИК
подготовки дипломного проекта

Наименование разделов, перечень разрабатываемых вопросов	Сроки представления научному руководителю	Примечание
Аналитическая часть	05.11.2018-21.12.2018	
Проектирование системы	07.01.2019-31.01.2018	
Разработка системы	04.02.2018-12.04.2018	

Дата выдачи задания «29» октября 2018г.

Заведующий кафедрой _____ Т.С. Картбаев

Научный руководитель дипломного проекта _____ Г.Д. Мусапирова

Задание принял к исполнению студент _____ А.М. Усербаев

Андатпа

Осы дипломдық жұмыстың мақсаты – «ақылды шабадан» жүйесін әзірлеу. Мұндай шабодан оның иесіне жеке заттарының қауіпсіздігі мен сақталуын, қарапайым әрі ыңғайлы қолдануды, шабодан жоғалған жағдайда оның орналасқан орнын бақылауға мүмкіндік береді. «Ақылды шабаданның» мұндай қызметтерін қамтамасыз ету үшін, ұялы қосымшаны әзірлеу қажет. Бұл жағдайда қосымша Android операциялық жүйесіне негізделеді. Ұялы қосымшаны жүзеге асыру үшін Java тілін қолдайтын Android Studio бағдарламасы пайдаланылады. «Ақылды шабадан» жүйесін әзірлеуден басқа, дипломдық жұмыста осы жүйенің табыстылығын растайтын экономикалық аспектілер де сипатталады. Сонымен қатар, аталған өнім әзірленген өндірістік жағдайлар қарастырылады.

Аннотация

Целью данной дипломной работы является разработка системы «умный чемодан». Такой чемодан обеспечит его владельцу безопасность и сохранность личных вещей, удобство и простоту использования, позволит отследить его местонахождение в случае кражи. Для того, чтобы обеспечить такую функциональность «умного чемодана», необходима разработка мобильного приложения. В данном случае приложение разрабатывается под операционную систему Android. Для реализации мобильного приложения используется программа Android Studio, которая поддерживает язык Java. Помимо самой разработки системы «умный чемодан», в дипломной работе описываются экономические аспекты реализации этой системы, которые подтверждают ее выгодность. Также рассматривается производственная обстановка, при которой была выполнена разработка данного продукта.

Annotation

The aim of the graduation thesis is to develop a "smart suitcase" system. Such suitcase will provide its owner with the followings: safety and security of personal belongings; convenient and easy use; tracing its location in case of theft. In order to provide such functions of the “smart suitcase”, we need to develop a mobile app. In this case, the app will be based on Android. To develop the mobile app, we use the Android Studio program, which supports Java. In addition to the development of the "smart suitcase" system, the thesis describes the economic aspects of its implementation, which confirm its profitability. The work also considers the production conditions of the product development.

Содержание

Введение

На сегодняшний день для путешествий из одного точки мира в другую человечество использует различные туристические сайты, а также специализированные мобильные приложения для построения наилучших маршрутов и покупки наиболее дешёвых билетов. Но, помимо выбора наиболее удачных условий пересечения границы города, человеку понадобится некоторый контейнер для удобства перемещения необходимых вещей. В мире современных технологий хотелось бы, чтобы такой контейнер мог использоваться также при помощи какого-либо инновационного подхода. Именно поэтому в данной дипломной работе рассматривается разработка «умного чемодана» - аксессуара, способного уместить в себя всё необходимое для перемещения в другую местность, а также обеспечить безопасность и сохранность вещей, помещенных в такой чемодан. Разрабатываемый чемодан является не обычным аксессуаром, а целой системой, оснащенной новыми технологиями и возможностью управления им при помощи мобильного устройства, на котором будет установлено специальное приложение.

Таким образом, для реализации «умного чемодана» необходимо создать мобильное приложение, которое обеспечит связь между смартфоном, работающим под операционной системой Android, и чемоданом. Такое приложение поможет владельцу «умного чемодана» не беспокоиться об утрате личных пренадлежностей и отслеживать нахождение своих вещей все время. С помощью мобильного устройства будет реализована возможность на расстоянии управлять замком чемодана, а именно блокировать его и разблокировать. Такая функция намного удобней и безопасней, чем обычный замок, имеющийся на современных чемоданах без технологичных возможностей. С помощью GPS, реализованным в таком приложении, «умный чемодан» будет легко найти в случае его пропажи. Особенность мобильного приложения, управляющего чемоданом, не только в его универсальной функциональности, но и в простом и понятном интерфейсе. Благодаря ему, каждый человек сможет интуитивно понять, как управлять чемоданом с мобильного устройства.

Операционная система Android, разработанная специально для смартфонов и планшетов, является открытой платформой, что дает возможность многим разработчикам расширять функции этой системы. Также, Android на сегодняшний день является наиболее распространенной системой, что гарантирует использование разработанного мобильного приложения многими пользователями. Именно поэтому приложение будет разработано именно для такой системы.

Для разработки приложения под операционную систему Android используется интегрированная среда Android Studio. В работе используется именно эта среда разработки, поскольку она имеет ряд преимуществ, таких как:

– возможность использования расширенного редактора макетов, а также функции предварительного просмотра макета на нескольких экранах;

– использование готовых шаблонов макетов и компонентов системы Android;

– компоновка приложений, основанная на Gradle – системе автоматической сборки;

– применение различных видов сборок и генерирование нескольких архивных исполняемых файлов-приложений, имеющих расширение .apk;

– возможность перепроектирования (переработки) кода;

– статический анализатор кода, который находит ошибки и недочеты в исходном коде написанной программы;

– возможность использования встроенной утилиты ProGuard для сокращения и оптимизации кода;

– достижимость разработки приложений для Android Wear – версии операционной системы Google Android, реализованной для переносных устройств, например, для «умных часов»;

– поддержка разработки приложений для Android TV – платформы, объединяющей телевизор и мобильные устройства;

– поддержка платформы Google Cloud Platform, которая обеспечивает интеграцию с сервисами Google Cloud Messaging и App Engine;

– возможность разработки приложения для новой программной платформы.

Разработка приложений в среде Android Studio реализуется с помощью таких языков программирования как Java и C++.

Таким образом, целью данного дипломного проекта является создание «умного чемодана», который управляется с помощью мобильного приложения, разработанного для операционной системы Android.

Из поставленной цели определены следующие задачи:

– собрать и проанализировать информацию по разработке мобильных приложений;

– обеспечить обычный чемодан необходимым функционалом;

– разработать мобильное приложение, способствующее взаимосвязи между мобильным устройством и чемоданом;

– рассмотреть экономические стороны реализованного устройства;

– проанализировать условия труда, при которых был создан «умный чемодан».

1 Аналитический обзор исследуемой проблемы

1.1 Оценка состояния изучаемого вопроса

На данный момент в мире существует большое количество различных видов чемоданов: которые можно переносить в руках или же способные катиться по полу с помощью колес; крупногабаритных размеров или же, наоборот, миниатюрные, предназначенные для коротких поездок; бросающиеся в глаза или же сохраняющие деловой стиль. [1, с. 71] Но, вне зависимости от разновидности, любой чемодан доставляет некоторый дискомфорт при эксплуатации и не способен обеспечить должный уровень сохранности вещей, находящихся внутри него. В связи с увеличением количества краж багажа и недобросовестной работой персонала аэропортов или вокзалов, путешествия не приносят должного удовольствия. Таким образом, обычные чемоданы уже не могут полностью удовлетворить все потребности современного человека. Поэтому, для решения подобного рода проблем на помощь приходят новые технологии, способные усовершенствовать предметы повседневной жизни.

Для улучшения стандартного чемодана необходимо обеспечить его различными устройствами, способные модернизировать его под системы. Снабжение чемодана Bluetooth-модулем позволит чемодану установить связь с мобильным приложением. Установка GPS-передатчика в чемодан поможет предотвратить его кражу или потерю. При этом владелец чемодана всегда сможет проследить его текущее положение на карте. Для того, чтобы чемодан мог передвигаться, его необходимо обеспечить моторной системой. Для поиска владельца чемодана и реализации возможности передвижения за ним следует использовать трехосевой цифровой компас. При помощи алгоритма вычисления угла поворота и дистанции между мобильным приложением и чемоданом, «умный чемодан» сможет определить, где находится хозяин и сможет следовать за ним. Так как в настоящее время многие люди пользуются различными гаджетами, в чемодан следует добавить powerbank-модуль. При помощи такого модуля гаджет можно будет зарядить. Для обеспечения всего устройства электроэнергией применяются литьевые аккумуляторы.

При использовании всех рассмотренных устройств обычный чемодан можно сделать более удобным и практичным. При этом, такой чемодан отлично подойдет для любых видов путешествий, а также послужит не плохим помощником в перемещении с большим количеством вещей. При создании автоматической системы следования за владельцем, можно перестать контролировать чемодан, а также положить на него сумки.

В настоящее время «умные чемоданы» редко используются из-за завышенной цены, поскольку разработчики используют дорогие устройства, не пытаясь заменить их на качественные, но более дешевые аналоги. Помимо этого, на чемоданы происходит завышение цены, поскольку разрабатывают их лишь в зарубежных странах, в следствии чего образуется наценка.

Таким образом, в данном дипломном проекте необходимо создать систему «умный чемодан», которая сможет облегчить передвижение людей при различных поездках и обезопасить имеющийся при них багаж от краж и потерь. Создав подобный чемодан на территории Республики Казахстан, его цена будет значительно ниже, при этом его функциональность не уступит зарубежным.

1.2 Обзор аналогов системы

Для того, чтобы определить, каким образом должна быть реализована система «умный чемодан» и каких ошибок следует избегать при ее разработке, необходимо произвести анализ подобных систем. Так как в Казахстане систем подобного рода еще не существует и для данной страны такая система является инновацией, необходимо изучить зарубежные аналоги и по полученным данным создать усовершенствованный «умный чемодан».

Самым известным «умным чемоданом» является Bluesmart, внешний вид которого представлен на рисунке 1.1. Данный чемодан содержит digital-замок, закрывающийся при помощи приложения на мобильном устройстве. При отдалении Bluesmart от владельца, чемодан закрывается автоматически. Такой чемодан обладает GPS-системой, которая обеспечивает владельцу постоянный контроль над его местоположением. Еще одним преимуществом чемодана Bluesmart является то, что он распознает степень своей наполненности, то есть точный вес багажа, и отображает его в приложении. Для того, чтобы можно было ускоренно пройти проверку багажа, в данном чемодане имеется отсек для ноутбука. Также предусмотрено встроенное зарядное устройство, в котором имеется возможность зарядить мобильный телефон до шести раз [2].

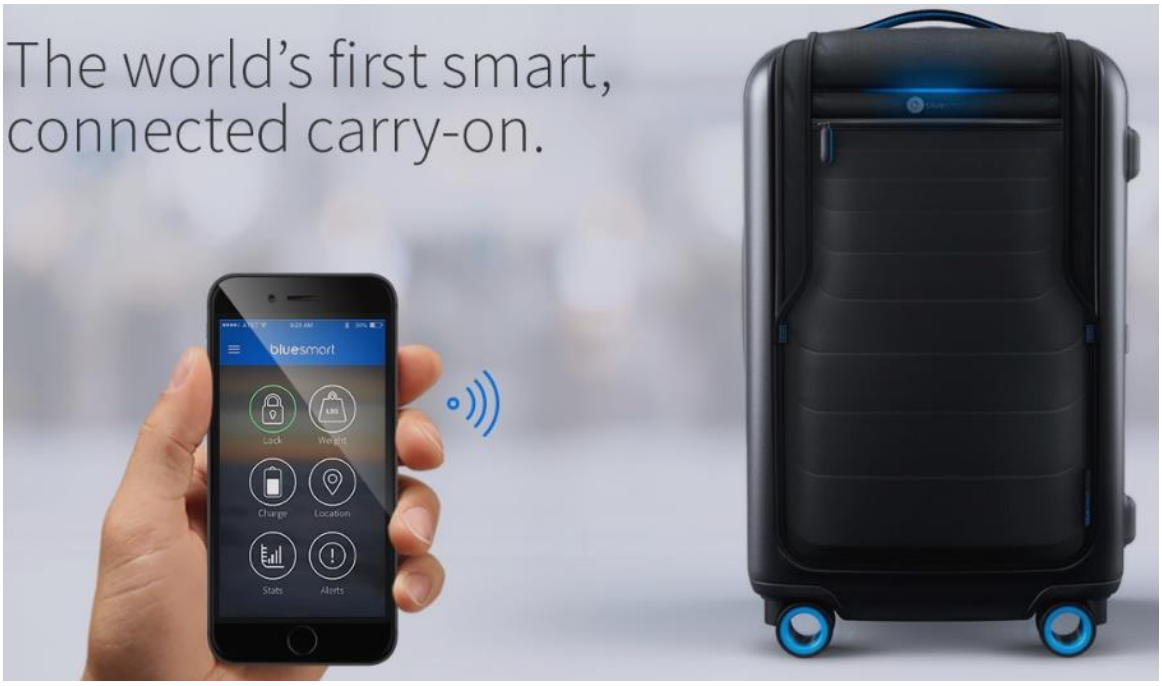


Рисунок 1.1 – «Умный чемодан» компании Bluesmart

Несмотря на то, что такие чемоданы понравились многим людям, через некоторое время данная компания была закрыта. А произошло это в связи с тем, что крупными американскими авиакомпаниями был наложен запрет на перевозку багажа в предметах с батареями, которые нельзя вытащить, поскольку дизайн чемодана не был продуман для таких целей. Решение о прекращении продаж «умного чемодана» было принято из-за взрывоопасности продуктов с литий-ионными батареями [3].

Следующей для изучения является спортивная смарт-сумка PAQSULE, показанная на рисунке 1.2. Хотя это и не чемодан, но данная сумка работает по такому же принципу – оснащение различными устройствами и управление при помощи мобильного телефона. Такая сумка умеет дизенфицировать вещи, которые находятся внутри нее, и имеет функцию зарядки мобильных телефонов. Такой функционал поддерживается в смарт-сумке при помощи встроенного специального электронного устройства, основанного на использовании технологии очищения вещей, применяемой в спортивных и медицинских помещениях, химчистках. Сумка PAQSULE обеззараживает не только вещи, но и оборудование, которое туда помещается. Для такой очистки не требуется большого количества времени. Активация очистки происходит с применением двух способов: при помощи мобильного приложения, разработанного под систему Android или IOS; нажатием кнопки, расположенной на самой сумке [4].

Рисунок 1.2 – Смарт-сумка PAQSULE

Главной особеностью обработки содержимого смарт-сумки является отстуствие необходимости вытаскивать из нее воду, продукты и т.д., поскольку ультрафиолетовое излучение и азон, при помощи которых происходит чистка, не оказывают влияние на вкус и качество пищи. Чтобы зарядить очистительное устройство используется micro USB, при этом встроенный аккумулятор при использовании одного заряда сможет функционировать от семнадцати до сорока тысяч раз. Помимо этого, аккумуляторные батареи используются в качестве зарядки для других USB-устройств [4].

Инновационным прорывом стал самодвижущийся чемодан компании Travelmate, способный ехать со скоростью до 11 км/ч за его владельцем самостоятельно без помощи рук (рисунок 1.3). Данный чемодан управляется при помощи смартфона, имитируя работу наземного дрона. Также можно установить режим следования за владельцем. При реализации такого чемодана были применены такие же технологии, что и в беспилотных автомобилях. Так, чемодан будет не просто передвигаться за владельцем, но и преодолевать препятствия, огибая их другим путем. Для предоставления Travelmate энергии применяются такие же батареи, как и в чемодане Bluesmart. Но, в отличии от него, в чемодане Travelmate предусмотрено вынимание и замена аккумуляторов. И, соответственно, в рассматриваемом чемодане также имеется возможность заряда мобильных устройств. В Travelmate содержится GPS-передатчик, при помощи которого можно отыскать чемодан в случае потери, или же просто проследить, где он находится в данный момент [5].

Рисунок 1.3 – самодвижущийся чемодан Travelmate

Несмотря на все достоинства чемодана Travelmate, имеется один, но очень весомый недостаток – высокая стоимость. Такой чемодан мало кто сможет приобрести. Некоторые функции Travelmate довольно бесполезны, но при этом повышают цену чемодана в несколько раз.

Последним для исследования является чемодан-скутер Modobag, изображенный на рисунке 1.4. Такой чемодан предназначается для людей, которые постоянно опаздывают на рейс или не хотят передвигаться пешком по большому аэропорту. Чемодан-скутер Modobag имеет колеса, педали в виде подставок для ног и руль. Он может разгоняться до 12 км/ч и выдерживать груз в размере до 118 килограмм. Для того, чтобы зарядить Modobag, применяются два USB-входа. Заряда батареи хватает на расстояние около 12 километров, после этого необходимо подзаряжать устройство, ожидая примерно два часа до окончания подзаряда. Размер чемодана Modobag невелик, вследствие чего позволяет взять его в салон самолета [6].

Рисунок 1.4 – чемодан-скутер Modobag

Основным недостатком чемодана-скутера Modobag является потеря места внутри чемодана по причине того, что встроенные элементы, отвечающие за передвижение, занимают около 20 процентов. Таким образом, при использовании чемодана в качестве скутера теряются его истинные функциональные назначения.

Выводы по проделанному анализу: многие производители стремятся максимально усовершенствовать чемодан, оснащая его слишком большим количеством дополнительных устройств, вследствие чего он теряет свою функциональную направленность. При этом, стоимость таким чемоданов становится неоправданной, поскольку многие функции попросту не нужны.

Таким образом, при создании системы «умный чемодан», необходимо продумать, какие детали будут необходимы, а без каких следует обойтись в пользу уменьшения стоимости и, тем самым, увеличения спроса на продукт.

1.3 Постановка задачи

В данном дипломном проекте необходимо разработать систему «умный чемодан», способную упростить переезды и перелеты, а также предоставить надежную защиту и сохранение личных вещей владельца чемодана. Помимо этого, следуют учитывать, что чемодан предназначается для переноса багажа, и не стоит увлекаться при добавлении слишком большого количества технологий, поскольку это повлияет на стоимость системы и ее назначение.

Чтобы сделать обычный чемодан максимально функциональным и удобным, необходимо добавить в него такие устройства, как: Bluetooth-модуль, GPS-передатчик, моторная система, трехосевой цифровой компас, зарядное устройство для смартфонов, литьевые аккумуляторы. Указанные устройства должны обеспечить чемодан следующими возможностями:

- установка соединения со смартфоном;
- организация безопасности личных вещей при помощи GPS;
- отслеживание текущего местоположения чемодана на карте;
- самостоятельное передвижение «умного чемодана»;
- поиск владельца и следование за ним при помощи смартфона;
- обеспечение чемодана зарядным устройством для гаджетов.

Для обеспечения чемодана работоспособностью и возможностью управляться при помощи смартфона, необходимо использовать плату микроконтроллера Arduino Uno, предварительно запрограммированную на языке C++ в среде разработки Arduino IDE. Помимо этого, необходимо разработать мобильное приложение на Android, при помощи которого владелец сможет управлять чемоданом со своего смартфона и подключаться к Bluetooth-модулю. Такое приложение должно иметь простой и многофункциональный интерфейс, для того, чтобы привлечь большее количество пользователей.

Разработка аппаратной части системы происходит методом сборки всех вышеуказанных устройств внутрь чемодана. Создание мобильного приложения происходит в среде разработки Android Studio на языке программирования Java.

1.4 Выбор средств разработки

Основной частью в чемодане является микроконтроллерная плата Arduino Uno, к которой подключаются все устройства. Arduino Uno контроллер построен на ATmega328 и показан на рисунке 1.5 и 1.6. Такая платформа содержит 6 аналоговых входов, 14 цифровых входов/выходов, USB

и силовой разъем, кварцевый генератор, кнопку перезагрузки и ICSP разъем. Для начала функционирования следует подключить платформу к вычислительной машине при помощи USB кабеля, также можно подать питание посредством адаптера AC/DC или батареи [7].

Arduino Uno потребляет энергию от внешнего источника питания или через USB подключение. Источник питания выбирается автоматически. Внешнее питание может поступать через преобразователь напряжения AC/DC или аккумуляторную батарею. Преобразователь напряжения подключается при помощи разъема 2.1 мм с центральным положительным полюсом. Провода от батареи подключаются к выводам Gnd и Vin разъема питания. Платформа может работать при внешнем питании от 6 В до 20 В. При напряжении питания ниже 7 В, вывод 5V может выдавать менее 5 В, при этом плата может работать нестабильно. При использовании напряжения выше 12 В регулятор напряжения может перегреться и повредить плату. Рекомендуемый диапазон от 7 В до 12 В [7].

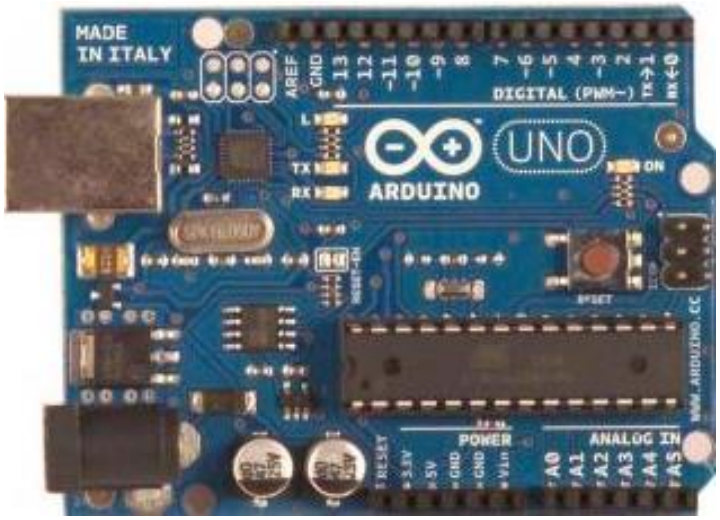


Рисунок 1.5 – Верхняя часть платы

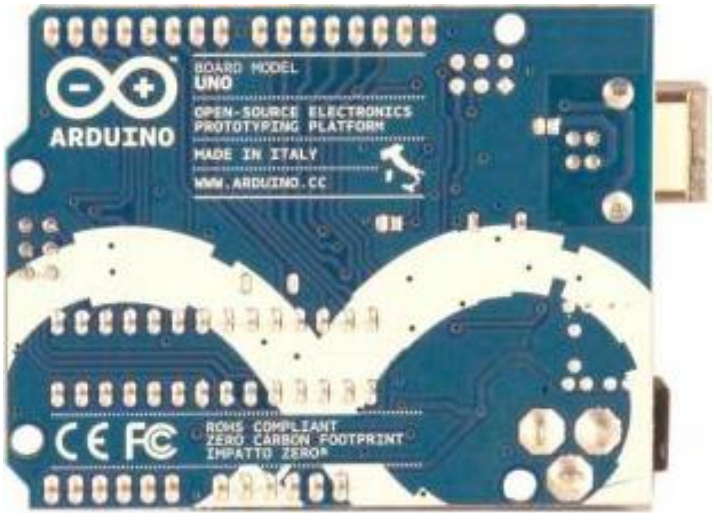


Рисунок 1.6 – Нижняя часть платы

На плате Arduino Uno имеется несколько устройств для связи с компьютером, другими устройствами Arduino или микроконтроллерами. ATmega328 поддерживают последовательный интерфейс UART TTL, осуществляемый выводами 0 и 1. Установленная на платформе микросхема ATmega8U2 направляет данный интерфейс через USB, программы на стороне компьютера взаимодействуют с платой через виртуальный COM порт. Прошивка ATmega8U2 использует стандартные драйвера USB COM, никаких сторонних драйверов не требуется, но на Windows для подключения потребуется файл ArduinoUno.inf. Мониторинг последовательной шины программы Arduino позволяет посылать и получать текстовые данные при подключении к платформе. Светодиоды RX и TX на платформе будут мигать при передаче данных через микросхему FTDI или USB подключение. Библиотекой SoftwareSerial возможно создать последовательную передачу данных через любой из цифровых выводов Uno. ATmega328 поддерживает интерфейсы I2C и SPI. В Arduino включена библиотека Wire для удобства использования шины I2C [7].

Для того, чтобы запрограммировать микроконтроллер Arduino Uno, используется интегрированная среда разработки Arduino IDE, написанная на языке программирования Java (рисунок 1.7). Средой Arduino используется принцип блокнота: стандартное место для хранения программ (скетчей). Скетчи из блокнота открываются через меню «File > Sketchbook» или кнопкой Open на панели инструментов. При первом запуске программы Arduino автоматически создается директория для блокнота. Расположение блокнота меняется через диалоговое окно Preferences.

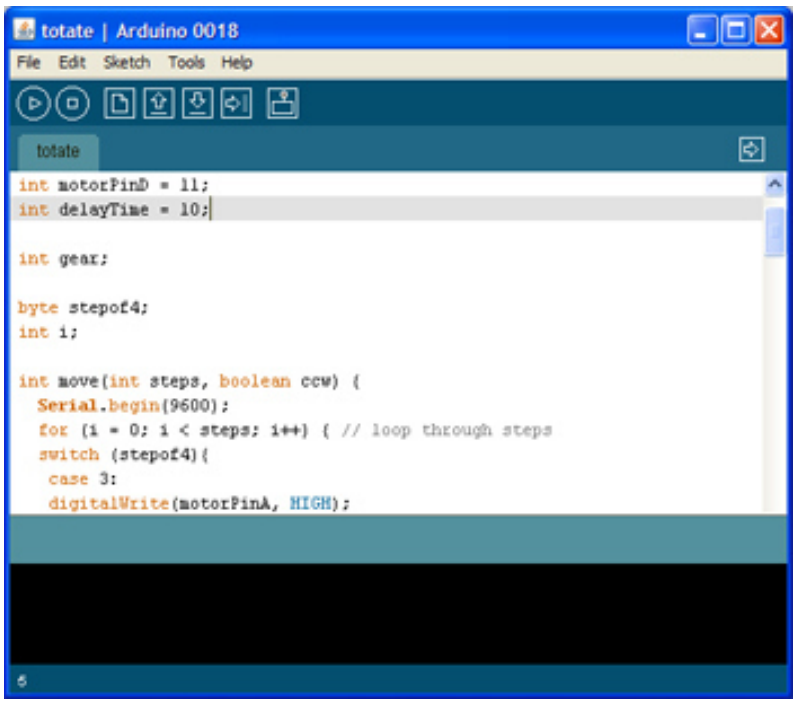


Рисунок 1.7 – Интегрированная среда разработки Arduino Uno IDE

Преимущества интегрированный среды разработки Arduino Uno IDE:

– Arduino IDE основан на AVRGCC. Изучение Arduino поможет изучить C++. Если не нравится конкретная высокоуровневая команда или библиотека для Arduino, почти всегда можно заменить её на аналогичную C++;

– можно программировать, питать и обмениваться сообщениями с Arduino при помощи одного USB кабеля;

– сделать простой проект можно за несколько минут, используя стандартные библиотеки, не вникая в них. Для считывания сигналов кнопок, вывода информации на семи сегментные или ЖК-дисплеи и управления двигателями для всего этого есть стандартные библиотеки, не требующие большого опыта в программировании;

– последовательные и SPI интерфейсы связи сделаны качественно.

К недостаткам такой среды можно отнести следующее:

– Arduino IDE. Интегрированная среда разработки Arduino — это кроссплатформенное приложение на Java, включающее в себя редактор кода, компилятор и модуль передачи прошивки в плату. Такой редактор является не самым лучшим, поскольку неудобен в использовании. Разработчик может перейти на более удобный сторонний редактор, но в любом случае IDE нужно для прошивки.

– загрузчик. Для того чтобы завершить проект с применением Arduino, необходимо вручную прошить загрузчик в каждый новый микроконтроллер ATmega. Он занимает 2Кб памяти;

– разнообразные варианты: в официальном модельном ряду есть варианты с памятью 30(32) Кб и 254(256)КБ. Если код занимает, допустим, 42 КБ, то единственным решением является использование полу-совместимого клона Sanguino и др.;

– отсутствие простого способа изменения тактовой частоты. Модель 3,3В/8МГц может работать на частоте 12МГц;

– digitalWrite() использует для выполнения 56 циклов. Без каких-либо затруднений можно узнать причину и перейти на прямой доступ к порту (вторая вещь которая заменяется после IDE). Arduino не очень удобна для время-зависимых приложений;

– невозможно легко отключить стандартную библиотеку для последовательной аппаратной части, для того чтобы брать прерывания с TX и RX, независимо от того, запущена она или нет. Строка в последовательный порт посылается при помощи конечного автомата с множеством пустых циклов ожидания флага опустошения буфера в основном теле программы – это расходование ресурсов – ведь есть прерывания;

– при переполнении ISR таймера прерывание происходит каждые 16K тактов в фоновом режиме. Это сделано для функций millis() и micros(), даже когда они не используются;

– пустой проект Arduino занимает 466 байт на Arduino Uno и 666 байт на Arduino Mega2560;

– также Arduino не отображает такие важные аспекты архитектуры микроконтроллеров как регистры, прерывания и таймеры.

Для создания мобильного приложения используется среда разработки Android Studio для смартфонов, функционирующих на базе операционной системы Android. Данная среда разработки показана на рисунках 1.8 и 1.9.

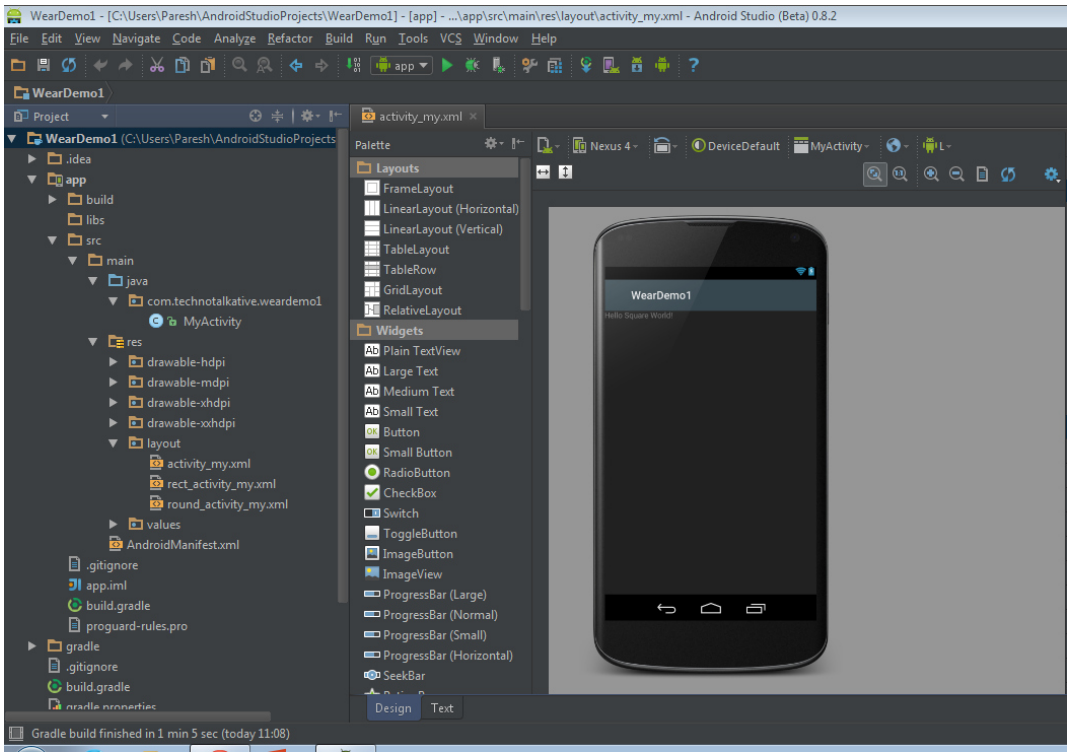


Рисунок 1.8 – Среда разработки Android Studio

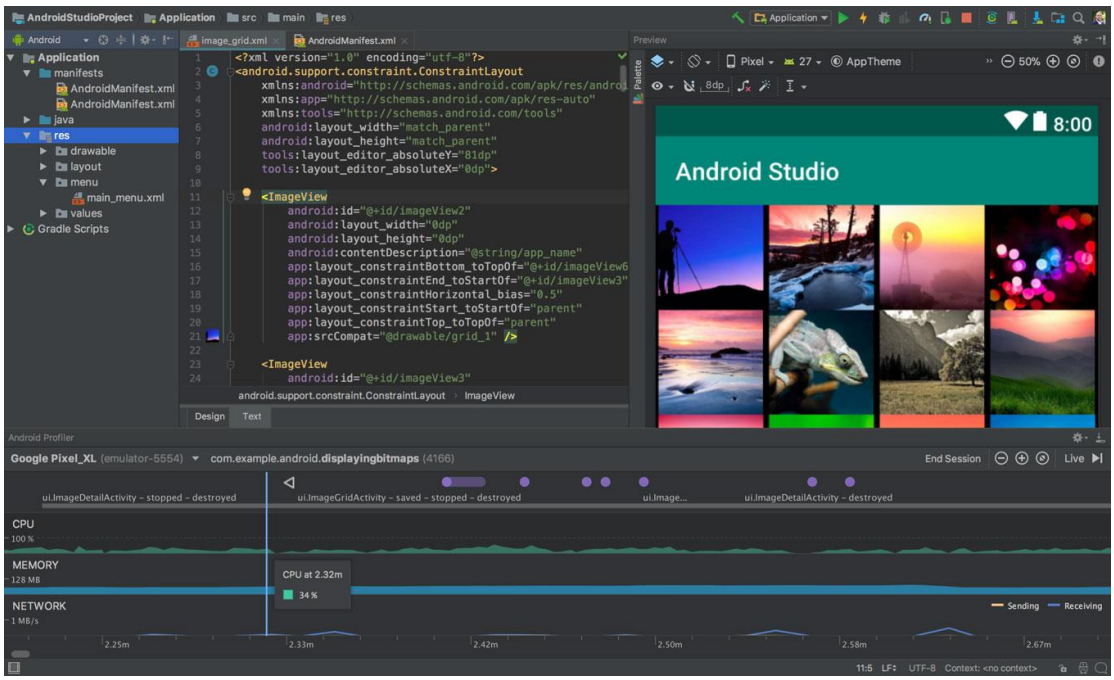


Рисунок 1.9 – Диагностика кода в Android Studio

Android Studio превосходит конкурентов по многим параметрам, к которым можно отнести: гибкость среды разработки; больший набор функций; процесс разработки, который подстраивается под разработчика. Во время создания приложений и утилит для операционной системы Android, пользователь программного обеспечения может наблюдать за изменениями в проекте, в режиме реального времени. Android Studio – универсальная среда разработки, так как позволяет оптимизировать работу будущих приложений для работы не только на смартфонах, но и на планшетах, портативных ПК, которые работают на основе рассматриваемой операционной системы. В программу встроен эмулятор, позволяющий проверить корректную работу приложения на устройствах с различными размерами экрана. Наиболее нужной эта функция стала после входа в тренды смартфонов, в которых установлены экраны с соотношением сторон 18:9. Особенность эмулятора – просмотр приблизительных показателей производительности при запуске приложения на самых популярных устройствах. Среда разработки для мобильных приложений Android Studio последней версии стала по настоящему удобной даже для начинающих разработчиков. В программе реализованы все современные средства для упаковки кода и его маркировки. Востребованная многими создателями ПО функция Drag-n-Drop, облегчающая перенос компонентов в среду разработки. Локализация приложений становится существенно проще с функцией SDK, которая также входит в перечень достоинств Android Studio. Связь с целевой аудиторией приложения после его релиза поможет реализовать инструмент от компании Google – Google Cloud Messaging [8].

Достоинства Android Studio: среда разработки поддерживает работу с несколькими языками программирования, к которым относятся самые популярные – C/C++, Java; позволяет разрабатывать приложения не только для смартфонов/планшетов, но и для портативных ПК, приставок для телевизоров Android TV, устройств Android Wear, новомодных мобильных устройств с необычным соотношением сторон экрана; тестирование корректности работы новых игр, утилит на той или иной системе происходит в эмуляторе; рефакторинг уже готового кода; большая библиотека с готовыми шаблонами и компонентами для разработки программного обеспечения; разработка приложения для Android N; предварительная проверка уже созданного приложения на предмет ошибок в нем; большой набор средств инструментов для тестирования каждого элемента приложения, игры; для начинающих разработчиков специально создано руководство по использованию Android Studio, размещенное на официальном сайте утилиты.

Также у данной утилиты есть небольшие недостатки. Несмотря на наличие встроенного Android-эмулятора в самой среде разработки, с тестированием новоразработанного приложения могут возникнуть трудности. Еще один недостаток – это невозможность создания серверных проектов на языке Java для ПК, Android устройств.

2 Проектирование системы

Проектирование – процесс определения архитектуры, компонентов, интерфейсов и других характеристик системы или её части. Результатом проектирования является проект – целостная совокупность моделей, свойств или характеристик, описанных в форме, пригодной для реализации системы [9]. Проектирование, наряду с анализом требований, является частью большой стадии жизненного цикла системы, называемой определением системы. Результаты этой стадии являются входной информацией для стадии реализации системы [10].

Таким образом, чтобы представить систему «умный чемодан» в полноценном виде, необходимо спроектировать структуру системы, а также мобильного приложения, благодаря которому владелец сможет управлять чемоданом на расстоянии.

2.1 Проектирование аппаратной части

Основой «умного чемодана» является аппаратная часть, состоящая из чемодана и различных устройств, отвечающих за функционирование системы. Для того, чтобы понять, как именно будет выглядеть и работать система, необходимо рассмотреть её функциональную структуру, представленную на рисунке 2.1.

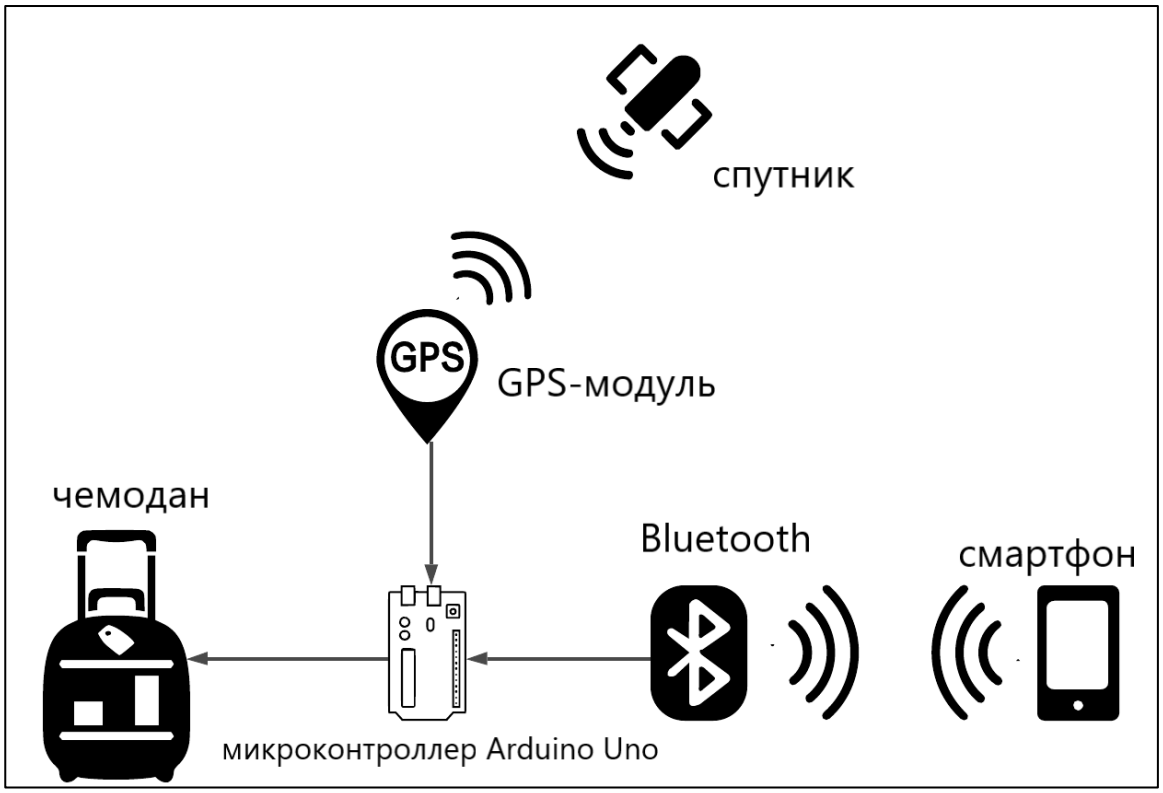


Рисунок 2.1 – Функциональная структура системы

На рисунке отображены основные составляющие системы, а именно: чемодан; микроконтроллер Arduino Uno; Bluetooth-модуль; мобильное устройство; GPS-модуль, принимающий связь со спутника.

В данном случае, чемодан представляет собой пространство для объединения всех компонентов системы. При разработке необходимо учесть, что, помимо элементов системы, чемодан должен вмещать в себя достаточное количество вещей. Поэтому следует поместить функциональную часть в отдельную область чемодана, так, чтобы она не занимала много места.

Так как микроконтроллер Arduino Uno не поддерживает беспроводную связь, необходимо использовать одно из множества внешних модулей для организации различных технологий беспроводной связи: модули Wi-Fi, GSM/GPRS, IR, Bluetooth, радиомодули для работы в различных частотных диапазонах. Технология Bluetooth используется для передачи данных между двумя устройствами, которые находятся в непосредственной близости друг с другом, причем необязательна прямая видимость. Bluetooth обеспечивает хорошую устойчивость к широкополосным помехам, что позволяет множеству устройств, находящихся в одном месте, одновременно общаться между собой, не мешая друг другу. Очень широко данная технология используется в телефонах, планшетах, ноутбуках.

В системе «умный чемодан» будет использован Bluetooth-модуль HC-05, который может работать как master (осуществлять поиск Bluetooth-устройств и инициировать установку связи), так и slave (ведомое устройство).

Bluetooth-модуль, изображенный на рисунке 2.2, служит для связи системы с мобильным приложением. С его помощью, при нажатии на определенный элемент интерфейса приложения, происходит запрограммированное действие.

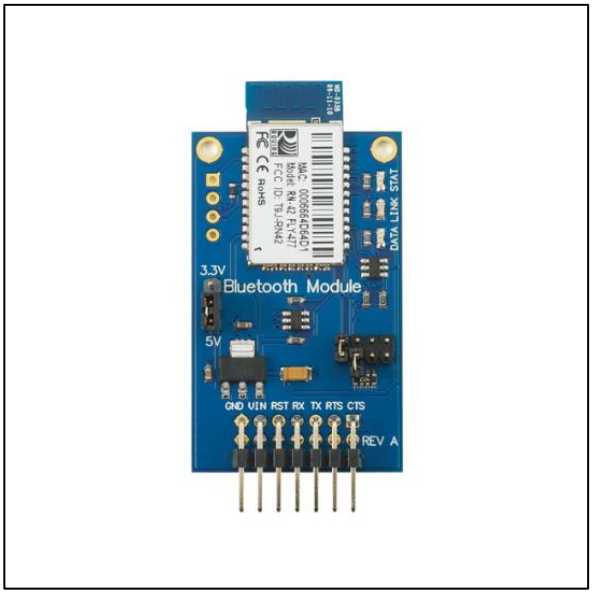


Рисунок 2.2 – Bluetooth-модуль

На рисунке 2.2 видно, что датчик имеет 6 выводов стандарта 2,54 мм:

– VCC (питание 3,6 – 6 В);
– GND (земля);
– TXD, RXD — UART интерфейс;
– STATE — индикатор состояния;
– KEY — контакт для входа в режим программирования.

Важной частью архитектуры Bluetooth, показанной на рисунке 2.3, является Host to Controller интерфейс (HCI), обеспечивающий взаимодействие софтовой подсистемы Host с железной подсистемой Controller. Взаимосвязь верхних уровней Bluetooth системы с ее аппаратной частью происходит через HCI-команды, инициируемые драйвером. Основные блоки архитектуры: RF (Radio), Baseband, Link Manager, L2CAP Layer [11].

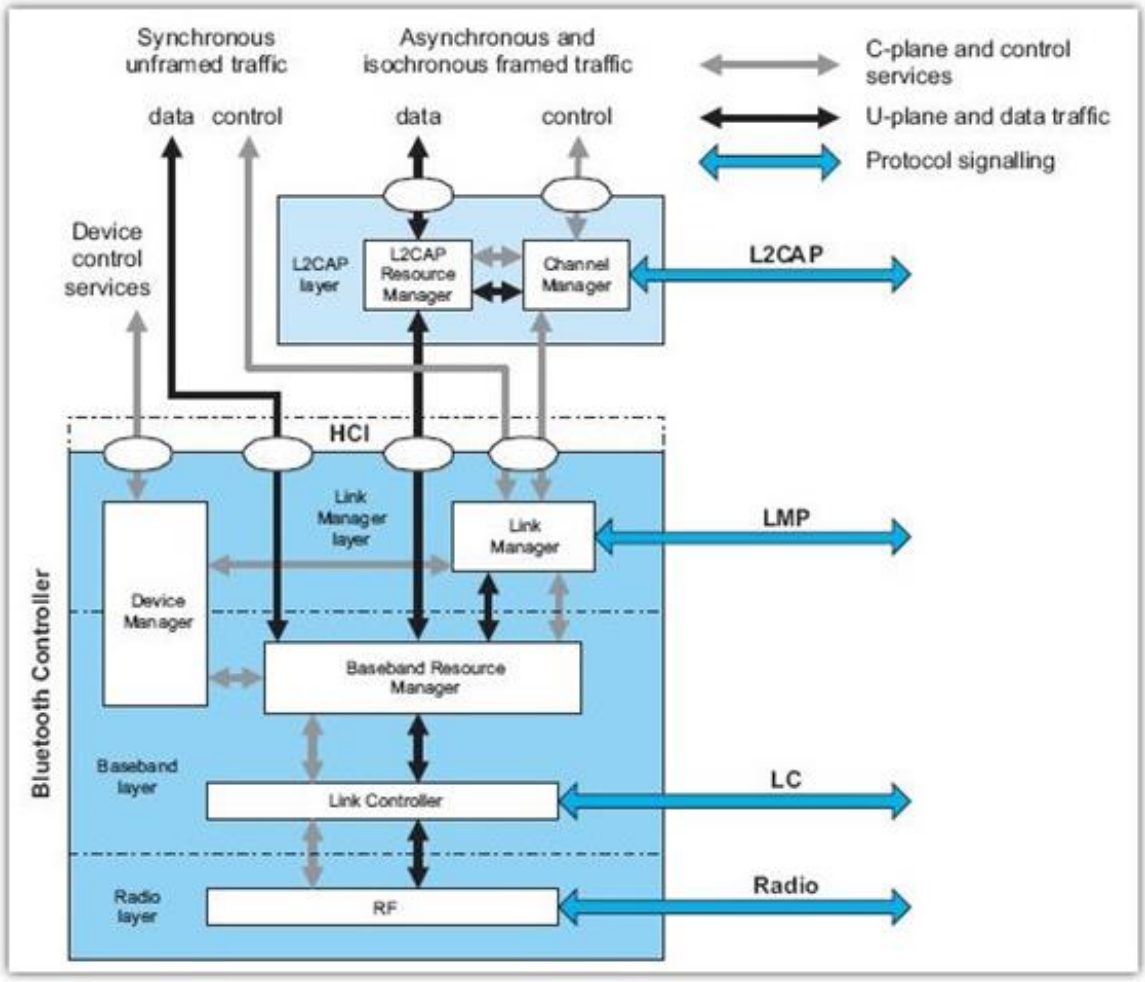


Рисунок 2.3 – Архитектура Bluetooth

Блок Radio занимается преобразованием битовой последовательности в радиосигналы. Вопросы модуляции, спектральных характеристик и физики процессов обеспечения битовой скорости — все это решается на нижнем уровне модели [11].

Уровень Baseband представлен в виде трех блоков, задача которых состоит в управлении физическими каналами, поверх которых устанавливаются физические соединения. Bluetooth-адресация, синхронизации генераторов устройств, управление кодами доступа к физическим каналам, поиск устройств и установление физического канала между ними — все это задачи Baseband-уровня [11].

После того, как два нижних уровня обеспечили пользователя физическим соединением между устройствами, необходимо организовать логические каналы, которые впоследствии станут базой для передачи трафика приложений. Link Manager отвечает за установление, изменение и освобождение логических соединений между устройствами, а также за обновление параметров физических соединений. Для этого Link Manager использует Link Management протокол (LMP) [11].

L2CAP Layer – высокоуровневый блок Bluetooth Host, оккупированный L2CAP уровнем. Logical Link Control and Adaptation Protocol (L2CAP) - протокол, работающий поверх созданных логических соединений, обеспечивающий сегментацию и восстановление пакетных данных от всех вышележащих приложений [11].

В целях определения местоположения чемодана при его утере или краже, необходимо использовать GPS-модуль. Основная задача GPS-модуля – установление координат принимающей антенны, в заданный промежуток времени, на основе информации радиосигнала. В данной системе будет использоваться GPS-модуль NEO-6M, отображенный на рисунке 2.4. Для обеспечения наилучшей возможной информации о местоположении, модуль NEO-6M использует новейшие технологии.

Рисунок 2.4 – GPS-модуль

Если расстояние от чемодана до приложения будет больше 50 метров, на смартфон придет уведомление и пользователь сможет посмотреть

местоположение «умного чемодана» на карте. Принцип работы GPS-модуля показан на рисунке 2.5.

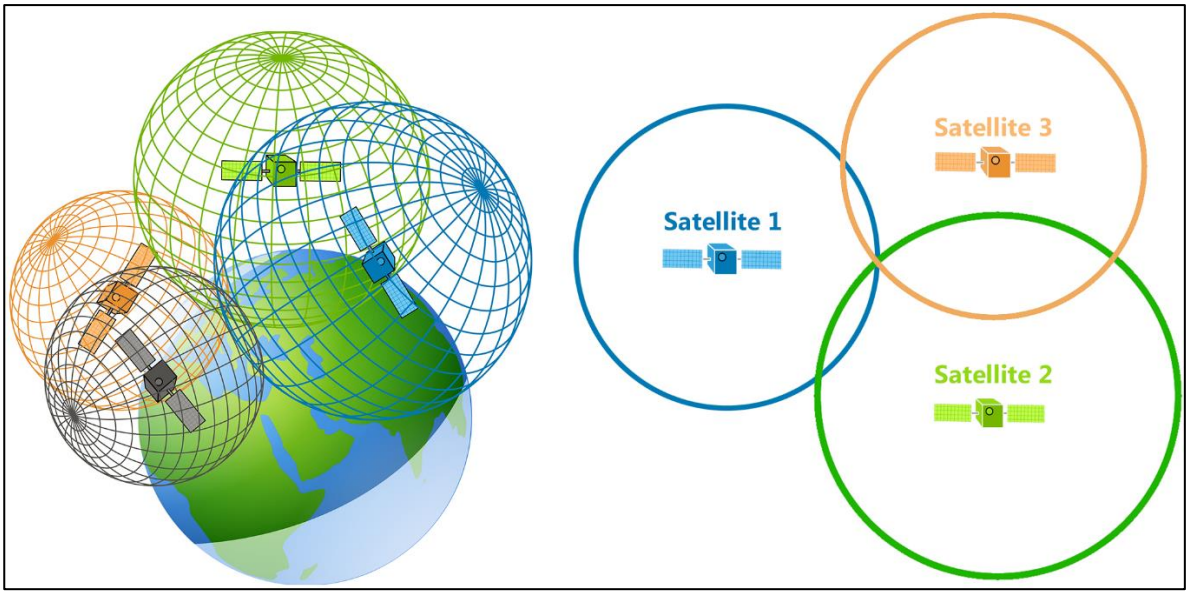


Рисунок 2.5 – Принцип работы GPS

Согласно рисунку 2.5, работу GPS системы можно представить так: несколько сфер, в центре которых находятся спутники, пересекаются, и в них находится пользователь. Радиус каждой из сфер, соответственно, равен расстоянию до этого видимого спутника. Сигналы от трех спутников дают возможность получить данные о широте и долготе, четвертый спутник дает информацию о высоте объекта над поверхностью. Полученные значения можно свести в систему уравнений, из которых можно найти координату пользователя. Таким образом, для получения точного местоположения необходимо провести 4 измерения дальностей до спутника [12].

Главной связующей системы является микроконтроллер Arduino Uno, который служит для получения и передачи сигналов от одного элемента к другому, а также выполнение различных вычислений на основе полученных данных. К нему подсоединяются два основных элемента: Bluetooth- и GPS-модули. Помимо этого, к микроконтроллеру подключаются такие устройства, как: powerbank-модуль, литьевые аккумуляторы, трехосевой цифровой компас, два моторных двигателя и драйвер для их управления.

2.2 Проектирование приложения

В целях управления всей системой используется мобильное устройство на базе Android, для которого было разработано приложение. Приложение играет ключевую роль в системе, так как пользователь в первую очередь обращает внимание на элемент, позволяющий взаимодействовать с «умным

чемоданом». Подробное описание мобильного приложения и самой системы представлены в техническом задании приложения А.

При проектировании нужно определить целевую аудиторию. Приложением будут пользоваться конкретные люди с конкретными задачами, поэтому для каждого из них нужно проработать самый удобный путь решения этих задач. Для прохождения этого этапа важно обладать полной информацией о целевой аудитории и знать об особенностях поведения представителей того или иного ее сегмента [13]. Мобильное приложение необходимо для тех людей, которые приобрели «умный чемодан». То есть это люди, которые часто путешествуют, имеющие достаточное количество денег, чтобы купить его. Примерный диапазон возрастов составляет от 22 до 63 лет. От сюда следует, что приложение должно быть стильным для молодежи, но при этом простым и понятным для людей постарше.

Пользователь должен понимать с каким элементом интерфейса ему нужно взаимодействовать для достижения своей конечной цели. Объектов интерфейса на экране должно быть не много, но при этом они должны полностью выполнять все необходимые функции системы. Кнопки и шрифты должны быть крупными для привлечения внимания и для слабовидящих.

Одним из самых сложных моментов в системе является подключение к GPS-модулю, поэтому следует разработать UML-диаграмму, при помощи которой можно объяснить последовательность процессов. Так, на рисунке 2.6 изображена диаграмма последовательности действий для получения местоположения чемодана на карте.

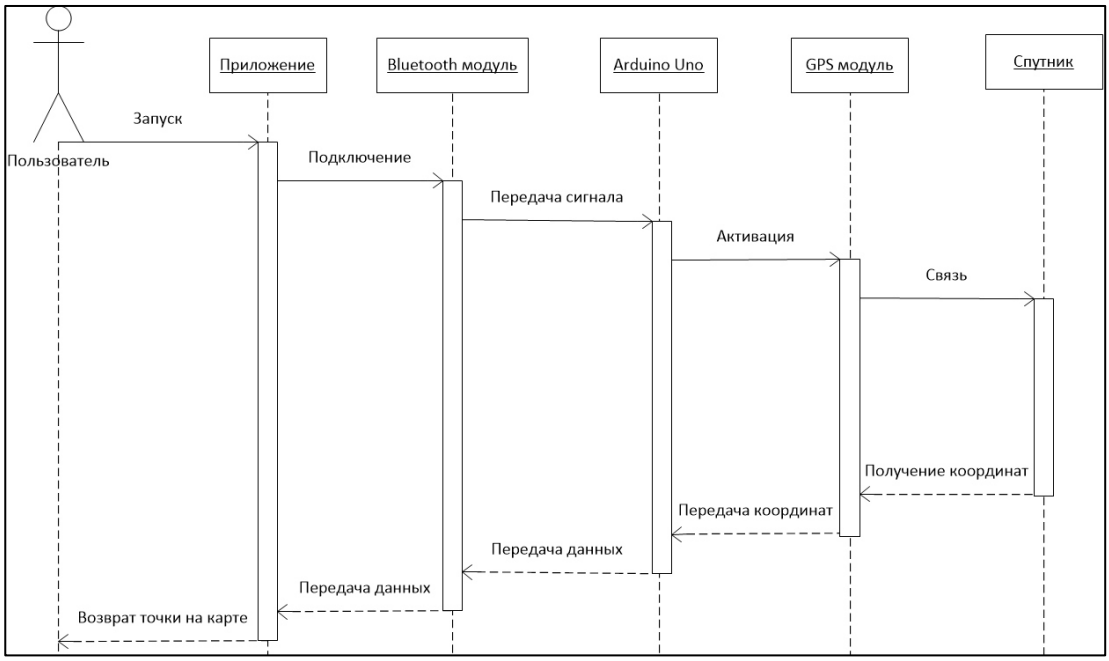


Рисунок 2.6 – Диаграмма последовательности действий

Данная диаграмма содержит подробный процесс взаимодействия пользователя с чемоданом. Так, в начале пользователь запускает приложение

на своем мобильном устройстве. Далее происходит подключение устройства к Bluetooth-модулю. При нажатии на карту в интерфейсе приложения, пользователь посылает сигнал на Bluetooth-модуль, который в свою очередь передает сигнал GPS-модулю. Затем сигнал проходит через антенну GPS-модуля к спутникам. Далее идет обратный процесс, в котором координаты переходят на экран мобильного устройства.

Для того, чтобы отобразить полный алгоритм работы мобильного приложения системы «умный чемодан», была разработана блок-схема, показанная на рисунке 2.7. В целях лучшего понимания каждого процесса блок-схемы, следует подробно ее описать.

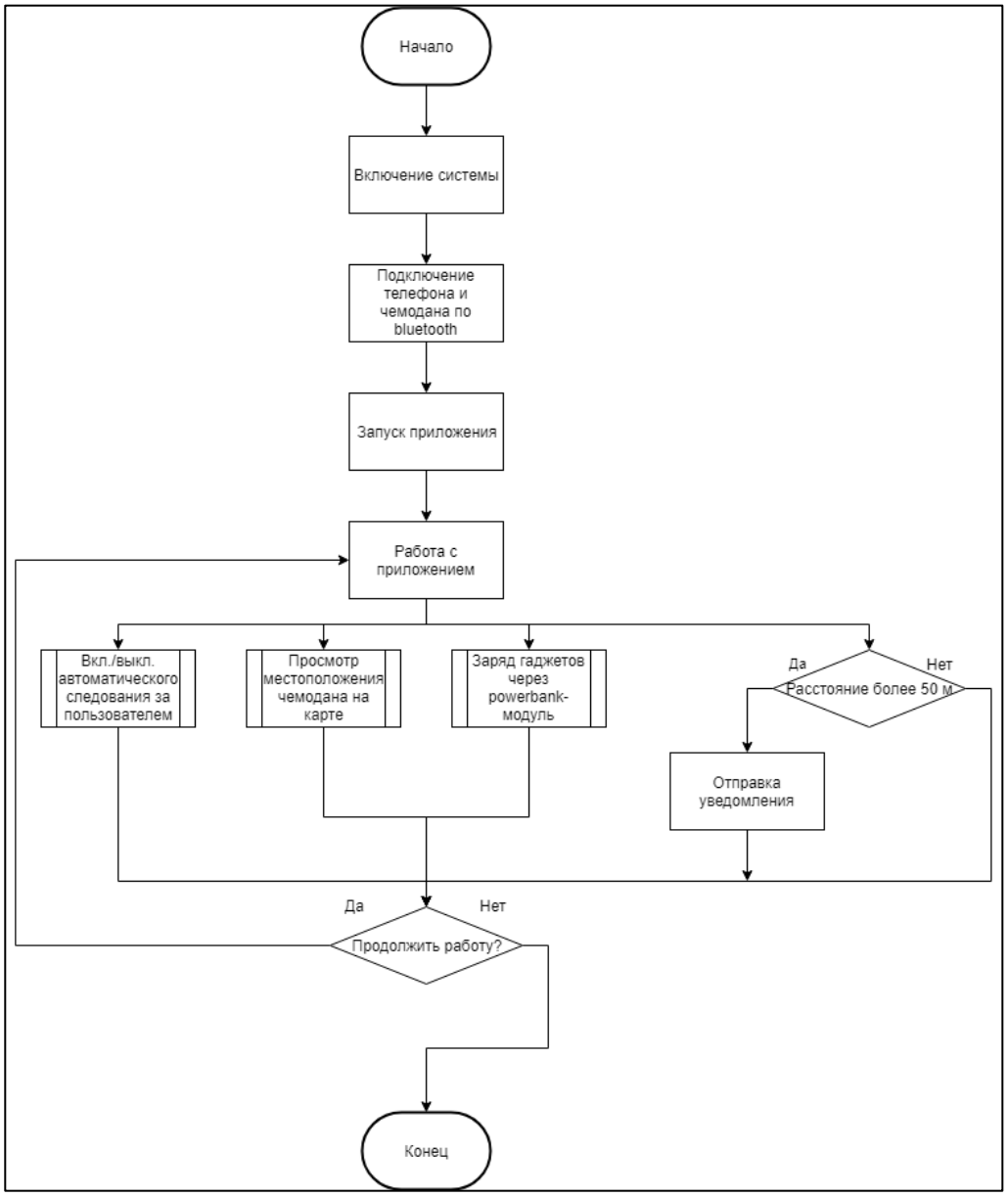


Рисунок 2.7 – Блок-схема алгоритма работы мобильное приложения

Как видно из рисунка 2.7, работа системы начинается с включения «умного чемодана» и мобильного приложения. На чемодане должна быть

предусмотрена кнопка включения/выключения для экономии заряда, поскольку при постоянной работе такой чемодан будет быстро разряжаться. После того, как система активирована, необходимо установить связь между чемоданом и мобильным устройством посредством Bluetooth-модуля. Так чемодан сможет получать указания при нажатии определенного элемента приложения.

После настройки всех устройств системы, начинается работа мобильного приложения. Приложение может предоставить чемодану выполнение следующих процессов системы:

– активация/отключение автоматического следования за пользователем;
– просмотр местоположения чемодана на карте;
– подача энергии для заряда гаджетов через powerbank-модуль.

Помимо этого, в системе следует предусмотреть случай, когда чемодан находится на расстоянии более 50 метров от владельца. Если расстояние больше 50 метров, то владельцу приходит уведомление об этом.

После выполнения определенных действий, мобильное приложение завершает работу.

После создания любого проекта на Android Studio, автоматически генерируется его дерево. Структура Android-проекта показана на рисунке 2.8.

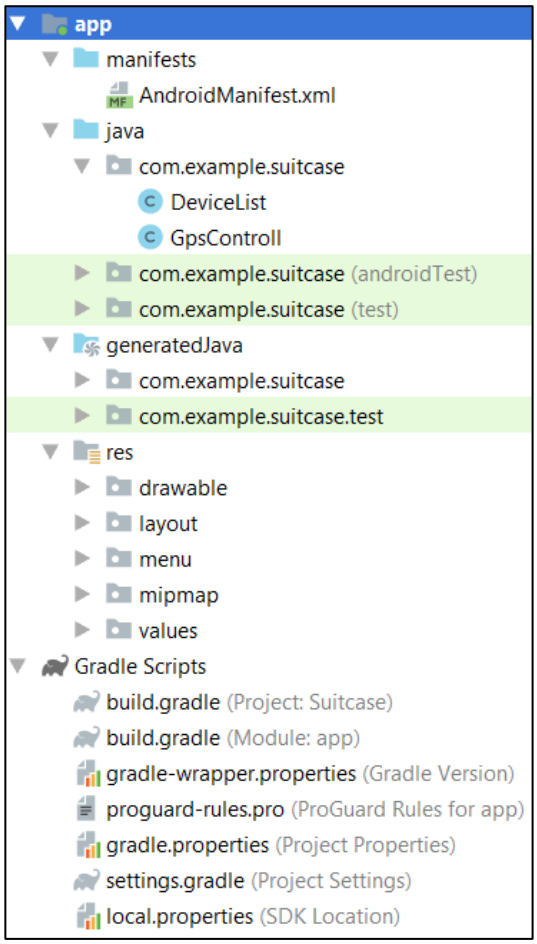


Рисунок 2.8 – Структура файлов приложения в Android Studio

В директории manifests находятся манифесты модулей. Манифест описывает, из каких частей будет состоять приложение. Он является XML-файлом, в котором описаны разрешения приложения, сервисы и метаданные.

Далее расположена директория java. В ней содержатся исходные коды мобильного приложения, которые делятся на две части: непосредственно основной код и код для тестирования.

Директория generatedJava создается автоматически во время разработки приложения. В ней хранится файл BuildConfig.java, который содержит в себе автоматические конфигурации, которые нельзя менять в ручную.

В директории res хранятся изображения, текст, аудиофайлы и другие медиа-файлы, которые используются в приложении.

Следующий элемент в дереве проектов – Gradle Scripts. Gradle – это система сборки проекта в Android Studio. В директории Gradle Scripts содержатся два файла build.gradle. Эти файлы описывают, каким образом нужно собрать проект. Первый файл отвечает за весь проект, а второй – за отдельный модуль. В мобильном приложении возможно создание нескольких модулей, и для каждого из них будет определен свой .gradle файл.

3 Разработка системы

В данном дипломном проекте была разработана система «умный чемодан». Разработка системы делится на три основные части:

– программирование микроконтроллера Arduino Uno;
– сборка аппаратной части;
– реализация мобильного приложения.

Каждый пункт разработки необходимо рассмотреть более подробно.

3.1 Программирование микроконтроллера

Основой системы является программный код для микроконтроллера Arduino Uno. С помощью него проходят все вычисления и алгоритмы работы системы. Программа создана в среде разработки Arduino IDE, с помощью которой, скетч, написанный на языке Arduino, проверяется, превращается в C++, компилируется, загружается в Arduino Uno.

В самом начале программного кода, при помощи директивы #include, подсоединяются необходимые библиотеки, рассмотренные на рисунке 3.1.

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_HMC5883_U.h>
#include <Servo.h>
#include <SoftwareSerial.h>
#include <TinyGPS++.h>
#include "./Definitions.h"
```

Рисунок 3.1 – Подключаемые библиотеки

Библиотека Wire позволяет Arduino взаимодействовать с различными устройствами по интерфейсу I2C / TWI. Начиная с версии языка Arduino 1.0, библиотека Wire наследует функции класса Stream, что позволяет ей быть совместимой с другими библиотеками, осуществляющими запись и чтение данных. Поэтому, методы send() и receive() были заменены методами read() и write().

Унифицированная библиотека датчиков Adafruit (Adafruit_Sensor) предоставляет общий интерфейс и тип данных для любого поддерживаемого датчика. Он определяет некоторую основную информацию о датчике. И возвращает стандартные единицы СИ определенного типа и масштаба для каждого поддерживаемого типа датчика. HMC5883L – это цифровой (I2C) компас (магнитометр). Магнитометр измеряет магнитную силу, которая полезна для обнаружения магнитного севера.

Библиотека Servo позволяет Arduino управлять работой серводвигателей. Сервопривод представляет собой двигатель со встроенным редуктором и выходным валом, положение которого можно точно контролировать. Стандартные серводвигатели позволяют задавать угол поворота вала в диапазоне от 0 до 180 градусов. В двигателях с непрерывным вращением вала можно задавать скорость его вращения.

Библиотека SoftwareSerial позволяет реализовать последовательный интерфейс на любых цифровых выводах Arduino с помощью программных средств, дублирующих функциональность UART (отсюда и название "SoftwareSerial"). Библиотека позволяет программно создавать несколько последовательных портов, работающих на скорости до 115200 бод. Для устройств, работающих с инвертированным сигналом, в библиотеке предусмотрен соответствующий параметр, включающий инвертирование.

TinyGPS++ - это новая библиотека Arduino для анализа потоков данных NMEA, предоставляемых модулями GPS. Как и ее предшественник, TinyGPS, эта библиотека предоставляет компактные и простые в использовании методы для извлечения местоположения, даты, времени, высоты, скорости и курса из пользовательских GPS-устройств. Тем не менее, интерфейс TinyGPS++ значительно проще в использовании, чем TinyGPS, и новая библиотека может извлекать произвольные данные из любого множества предложений NMEA.

В библиотеке Definitions содержатся директивы #define. Это удобная директива, которая позволяет дать имя константе перед тем, как программа будет скомпилирована. Определенные этой директивой константы не занимают программной памяти, поскольку компилятор заменяет все обращения к ним их значениями на этапе компиляции, соответственно они служат исключительно для удобства и улучшения читаемости текста программы. На рисунке 3.2 можно увидеть фрагмент кода файла Definitions.h.

```
// Pin variables
#define GPS_TX_PIN 6
#define GPS_RX_PIN 3

#define PIN_SPEED_RIGHT 9
#define PIN_SPEED_LEFT 5

#define BLUETOOTH_TX_PIN 10
#define BLUETOOTH_RX_PIN 11

// Bluetooth GPS input
#define SOP '<'
#define EOP '>'

// Motor stuffs
#define RC_NEUTRAL 1500
#define RC_MAX 1600
#define RC_MIN 1400
```

Рисунок 3.2 - Фрагмент кода файла Definitions.h

Далее в программе идет объявление глобальных переменных и объектов различных классов. Фрагмент кода отображен на рисунке 3.3. Для объявления глобальных переменных достаточно указать тип переменной и название. Также можно сразу определить значение переменной. В зависимости от типа данных меняется размер занимаемого места.

В этом фрагменте кода создаются объекты классов, таких как: TinyGPSPLUS, Servo, SoftwareSerial и Adafruit_HMC5883_Unified. В зависимости от конструктора класса используются два вида объявления: с параметрами и без них.

```
// GPS
TinyGPSPlus gps;
#define GPS_BAUD 4800

// Motors
Servo servoThrottleRight;
Servo servoThrottleLeft;

// Master Enable
bool enabled = true;

// Bluetooth input
bool started = false;
bool ended = false;
char inData[80]; // creates an 80 character array called "inData"
byte index; //creates a variable type=byte called "index"
double Long; //variable for longitude coordinate
double Lat; //variable for latitude coordinate

// Serial components
SoftwareSerial bluetoothSerial(BLUETOOTH_TX_PIN, BLUETOOTH_RX_PIN);
SoftwareSerial gpsSerial(GPS_RX_PIN, GPS_TX_PIN); // TXD to digital pin 6

/* Compass */
Adafruit_HMC5883_Unified mag = Adafruit_HMC5883_Unified(12345);
```

Рисунок 3.3 – Объявление глобальных переменных

В последующем фрагменте кода, показанном на рисунке 3.4, используются функции для работы с GPS-модулем.

Функция feedgps() служит проверкой на подачу информации с GPS-модуля, то есть, если навигационная система будет получать какие-либо данные, функция вернет значение true, в противном случае false.

Функция gpsdump() нужна для получения широты и долготы «умного чемодана».

Для объединения этих двух функций необходима функция checkGPS(). Через определенный интервал времени проверяется наличие каких-либо данных при помощи функции feedgps(), и если они имеются они возвращаются функцией gpsdump().

```
GeoLoc checkGPS() {
  Serial.println("Reading onboard GPS: ");
  unsigned long start = millis();
  while (millis() - start < GPS_UPDATE_INTERVAL) {
    // If we recieved new location then take the coordinates and pack them into a struct
    if (feedgps())
      return gpsdump();
  }

  GeoLoc robotLoc;
  robotLoc.lat = 0.0;
  robotLoc.lon = 0.0;

  return robotLoc;
}

// Get and process GPS data
GeoLoc gpsdump() {
  GeoLoc robotLoc;
  robotLoc.lat = gps.location.lat();
  robotLoc.lon = gps.location.lng();

  Serial.print(robotLoc.lat, 7); Serial.print(", "); Serial.println(robotLoc.lon, 7);

  return robotLoc;
}

// Feed data as it becomes available
bool feedgps() {
  while (gpsSerial.available() > 0) {
    if (gps.encode(gpsSerial.read()))
      return true;
  }
  return false;
}
```

Рисунок 3.4 – Функции для работы GPS

Чтобы тестировать программу, необходимо выводить определенные данные на экран. В таких целях используется функция displayCompassDetails, изображенная на рисунке 3.5.

```
void displayCompassDetails(void)
{
  sensor_t sensor;
  mag.getSensor(&sensor);
  Serial.println("------------------------------------");
  Serial.print  ("Sensor:       "); Serial.println(sensor.name);
  Serial.print  ("Driver Ver:   "); Serial.println(sensor.version);
  Serial.print  ("Unique ID:    "); Serial.println(sensor.sensor_id);
  Serial.print  ("Max Value:    "); Serial.print(sensor.max_value); Serial.println(" uT");
  Serial.print  ("Min Value:    "); Serial.print(sensor.min_value); Serial.println(" uT");
  Serial.print  ("Resolution:   "); Serial.print(sensor.resolution); Serial.println(" uT");
  Serial.println("------------------------------------");
  Serial.println("");
  delay(500);
}
```

Рисунок 3.5 – Функция отображения данных о компасе

Для того чтобы чемодан мог следовать за мобильным приложением нужно определить направление, расстояние и азимут. В этих целях используются специальные математические формулы. Для начала нужно определить константы для перевода градусов в радианы и наоборот. Константы изображены на рисунке 3.6.

```
#ifndef DEGTORAD
#define DEGTORAD 0.0174532925199432957f
#define RADTODEG 57.2957795130823208767f
#endif
```

Рисунок 3.6 – Константы

Далее используется формула для вычисления азимута «умного чемодана», рассмотренная на рисунке 3.7.

```
float geoBearing(struct GeoLoc &a, struct GeoLoc &b) {
  float y = sin(b.lon-a.lon) * cos(b.lat);
  float x = cos(a.lat)*sin(b.lat) - sin(a.lat)*cos(b.lat)*cos(b.lon-a.lon);
  return atan2(y, x) * RADTODEG;
}
```

Рисунок 3.7 – Определение азимут

Также необходимо определить расстояние между чемоданом и мобильным приложением. Дистанция вычисляется при помощи математической формулы. Функция изображена на рисунке 3.8.

```
float geoDistance(struct GeoLoc &a, struct GeoLoc &b) {
  const float R = 6371000; // radius of earth in metres
  float p1 = a.lat * DEGTORAD;
  float p2 = b.lat * DEGTORAD;
  float dp = (b.lat-a.lat) * DEGTORAD;
  float dl = (b.lon-a.lon) * DEGTORAD;

  float x = sin(dp/2) * sin(dp/2) + cos(p1) * cos(p2) * sin(dl/2) * sin(dl/2);
  float y = 2 * atan2(sqrt(x), sqrt(1-x));

  // returns distance in meters
  return R * y;
}
```

Рисунок 3.8 – Вычисление расстояния между двумя координатами

После определения дистанции, нужно найти направление чемодана. Алгоритм вычисления направления системы «умный чемодан» отображен на рисунке 3.9.

```
float geoHeading() {
  /* Get a new sensor event */
  sensors_event_t event;
  mag.getEvent(&event);

  // Hold the module so that Z is pointing 'up' and you can measure the heading with x&y
  // Calculate heading when the magnetometer is level, then correct for signs of axis.
  float heading = atan2(event.magnetic.y, event.magnetic.x);

  // Offset
  heading -= DECLINATION_ANGLE;
  heading -= COMPASS_OFFSET;

  // Correct for when signs are reversed.
  if(heading < 0)
    heading += 2*PI;

  // Check for wrap due to addition of declination.
  if(heading > 2*PI)
    heading -= 2*PI;

  // Convert radians to degrees for readability.
  float headingDegrees = heading * 180/M_PI;

  // Map to -180 - 180
  while (headingDegrees < -180) headingDegrees += 360;
  while (headingDegrees >  180) headingDegrees -= 360;

  return headingDegrees;
}
```

Рисунок 3.9 – Нахождение направления чемодана

Для того чтобы обеспечить передвижение чемодана используются функции запуска и остановки моторов, показанные на рисунке 3.10.

```
void setSpeedMotorA(int speed) {
  servoThrottleRight.writeMicroseconds(speed + MOTOR_A_OFFSET);
}

void setSpeedMotorB(int speed) {
  servoThrottleLeft.writeMicroseconds(speed + MOTOR_B_OFFSET);
}

void stop() {
  // stop the motors
  servoThrottleRight.writeMicroseconds(RC_NEUTRAL);
  servoThrottleLeft.writeMicroseconds(RC_NEUTRAL);
}
```

Рисунок 3.10 – Функции управления мотором

Следующая функция используется для задания скорости в зависимости от дистанции между чемоданом и приложением. Также в этой функции настраивается угол поворота чемодана в момент следования за хозяином.

```
void drive(int distance, float turn) {
  int fullSpeed = 230;
  int stopSpeed = 0;

  // drive to location
  int s = fullSpeed;

  if ( distance < 8 ) {
    int wouldBeSpeed = s - stopSpeed;
    wouldBeSpeed *= distance / 8.0f;
    s = stopSpeed + wouldBeSpeed;
  }

  int autoThrottle = constrain(s, stopSpeed, fullSpeed);
  autoThrottle = 230;

  float t = turn;
  while (t < -180) t += 360;
  while (t >  180) t -= 360;

  Serial.print("turn: ");
  Serial.println(t);
  Serial.print("original: ");
  Serial.println(turn);

  float t_modifier = (180.0 - abs(t)) / 180.0;
  float autoSteerA = 1;
  float autoSteerB = 1;

  if (t < 0) {
    autoSteerB = t_modifier;
  } else if (t > 0){
    autoSteerA = t_modifier;
  }

  Serial.print("steerA: "); Serial.println(autoSteerA);
  Serial.print("steerB: "); Serial.println(autoSteerB);

  int speedA = (int) (((float) map(autoThrottle, stopSpeed, fullSpeed, RC_NEUTRAL, RC_MIN)) * autoSteerA);
  int speedB = (int) (((float) map(autoThrottle, stopSpeed, fullSpeed, RC_NEUTRAL, RC_MAX)) * autoSteerB);

  setSpeedMotorA(speedA);
  setSpeedMotorB(speedB);
}
```

Рисунок 3.11 – Функция настройки скорости передвижения

Ключевая функция передвижения, которая объединяет в себе все необходимые функции, изображена на рисунке 3.12. Данная функция позволяет получать геопозицию чемодана, определять расстояние, направление и азимут, и после этого двигать чемодан в нужном направлении.

```
void driveTo(struct GeoLoc &loc, int timeout) {
  gpsSerial.listen();
  GeoLoc robotLoc = checkGPS();
  bluetoothSerial.listen();

  if (robotLoc.lat != 0 && robotLoc.lon != 0 && enabled) {
    float distance = 0;
    //Start move loop here
    do {
      gpsSerial.listen();
      robotLoc = checkGPS();
      bluetoothSerial.listen();

      distance = geoDistance(robotLoc, loc);
      float bearing = geoBearing(robotLoc, loc) - geoHeading();

      Serial.print("Distance: ");
      Serial.println(distance);

      Serial.print("Bearing: ");
      Serial.println(geoBearing(robotLoc, loc));

      Serial.print("Heading: ");
      Serial.println(geoHeading());

      drive(distance, bearing);
      timeout -= 1;
    } while (distance > 1.0 && enabled && timeout>0);

    stop();
  }
}
```

Рисунок 3.12 – Ключевая функция передвижения системы

Перед запуском системы необходимо настроить микроконтроллер. Функция setup() вызывается, когда стартует скетч. Используется для инициализации переменных, определения режимов работы выводов, запуска используемых библиотек и т.д. Функция setup запускает только один раз, после каждой подачи питания или сброса платы Arduino. Фрагмент кода отображен на рисунке 3.13.

```
void setupCompass() {
   /* Initialise the compass */
  if(!mag.begin())
  {
    /* There was a problem detecting the HMC5883 ... check your connections */
    Serial.println("Ooops, no HMC5883 detected ... Check your wiring!");
    while(1);
  }
  /* Display some basic information on this sensor */
  displayCompassDetails();
}

void setup()
{
  // Attaching motors
  servoThrottleRight.attach(PIN_SPEED_RIGHT);
  servoThrottleLeft.attach(PIN_SPEED_LEFT);

  //Debugging via serial
  Serial.begin(115200);

  // Compass
  setupCompass();

  //GPS
  gpsSerial.begin(GPS_BAUD);

  //Bluetooth
  bluetoothSerial.begin(9600);
}
```

Рисунок 3.13 – Настройка системы

После вызова функции setup(), которая инициализирует и устанавливает первоначальные значения, функция loop() делает точь-в-точь то, что означает её название, и крутится в цикле, позволяя программе совершать вычисления и реагировать на них. Эту функцию используют для активного управления платой Arduino. По своей сути, void loop – это главная функция, точка входа в программу. Arduino повторяет вызов этой функции миллионы раз в секунду. На рисунке 3.14 изображен программный код системной функции loop().

```
void loop()
{
  // Read all serial data available, as fast as possible
  while (bluetoothSerial.available() > 0)
  {
    char inChar = ((byte)bluetoothSerial.read());
    if (inChar == SOP)
    {
      index = 0;
      inData[index] = '\0';
      started = true;
      ended = false;
    }
    else if (inChar == EOP)
    {
      ended = true;
      break;
    }
    else
    {
      if (index < 79)
      {
        inData[index] = inChar;
        index++;
        inData[index] = '\0';
      }
    }
  }
  if (started && ended)
  {
    char *token = strtok(inData, ",");
    boolean first = true;
    while (token)
    {
      double val = atof(token);
      token = strtok(NULL, ",");
      if (first){
        Lat = val;
      }else{
        Long = val;
      }
      first = false;
    }

    // Reset for the next packet
    started = false;
    ended = false;
    index = 0;
    inData[index] = '\0';

    GeoLoc phoneLoc;
    phoneLoc.lat = Lat;
    phoneLoc.lon = Long;

    Serial.print(phoneLoc.lat, 7); Serial.print(", "); Serial.println(phoneLoc.lon, 7);
    driveTo(phoneLoc, GPS_WAYPOINT_TIMEOUT);
  }
}
```

Рисунок 3.14 – Функция loop()

3.2 Сборка аппаратной части

Помимо программы для микроконтроллера Arduino Uno, необходимо собрать всю систему, которая будет внедрена в обычный чемодан. В сборке аппаратной части используются такие элементы как:

- Arduino Uno;
- Breadboard;
- Bluetooth-модуль HC-05;
- GPS-модуль NEO-6M;
- трехосевой цифровой компас Adafruit HMC5833l;

– литьевые аккумуляторы;
– сервопривод;
– драйвер для управления моторами L298n;
– два мотора;
– два колеса.

В первую очередь нужно продумать местоположение функциональной части. Данная система должна быть компактной, чтобы не занимать большое пространство. Все детали и соединяющие элементы должны плотно сидеть в корпусе системы, так как чемоданы часто подвергаются тряске.

Для построения такой системы была создана схема, изображенная на рисунке 3.15.

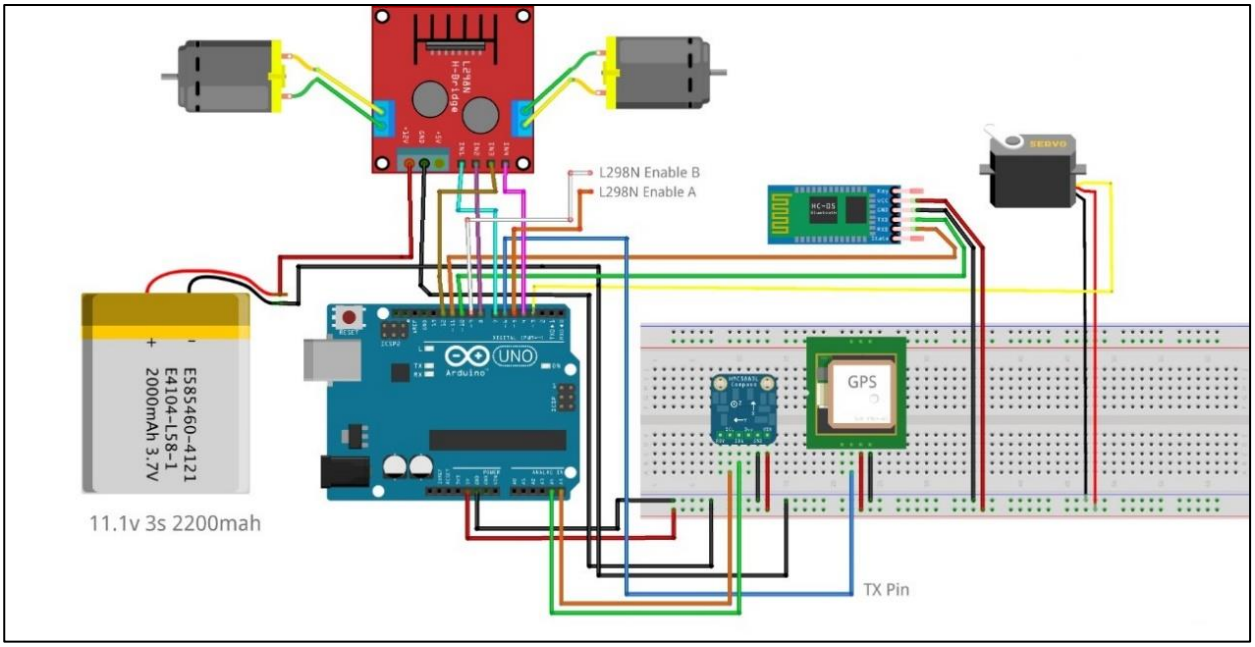


Рисунок 3.15 – Схема построения устройства

Bluetooth-модуль выведен наружу для лучшей связи с мобильным устройством. Остальные компоненты должны располагаться внутри системы. GPS-модуль и трехосевой цифровой компас подключаются через Breadboard. От микроконтроллера, при помощи проводов папа-папа, подключаются источник питания на 5 вольт и заземление.

Компас работает с I2C, поэтому необходимо подключить выводы SLC и SDA к A5 и A4 портам соответственно. Остальные контакты были подключены через цифровой ввод и вывод. К драйверу L298n подключаются два моторных двигателя. Также драйвер L298n потребляет заряд из литьевого аккумулятора. Сам микроконтроллер будет получать энергию от встроенной батареи. Порты Bluetooth-модуля TXD и RXD подключаются к 10 и 11 пинам.

При подключении устройств важно не перепутать последовательность действий, иначе может произойти сбой в системе.

3.3 Реализация мобильного приложения

В мобильном приложение реализован простой интерфейс, но при этом выполнены все функциональные назначения. При входе в приложение происходит поиск Bluetooth-устройств, показанный на рисунке 3.16.

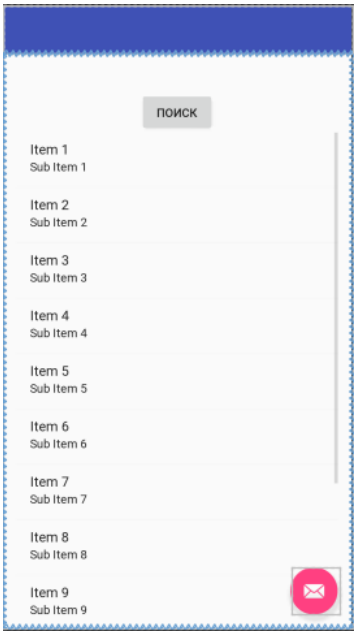


Рисунок 3.16 – Список Bluetooth-устройств

После того как пользователь выберет Bluetooth-модуль системы из этого списка, откроется новое окно, изображенное на рисунке 3.17.

Рисунок 3.17 – Основной раздел управления

Интерфейс для мобильного приложения пишется на языке XML. На рисунках 3.18 и 3.19 отображены программные коды для построения внешнего вида активностей.

```
<?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:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:context=".DeviceList"
    tools:showIn="@layout/activity_device_list">

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Поиск"
        android:id="@+id/button"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="28dp" />

    <ListView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/listView"
        android:layout_below="@+id/button"
        android:layout_centerHorizontal="true" />
</RelativeLayout>
```

Рисунок 3.18 – content_device_list.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:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:context=".GpsControll"
    tools:showIn="@layout/activity_gps_controll">

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Запуск"
        android:id="@+id/btnForward"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="42dp" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Отключиться"
        android:id="@+id/btn_disconnect"
        android:layout_below="@+id/btnForward"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="123dp" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Система готова к работе"
        android:id="@+id/textView"
        android:layout_below="@+id/btnForward"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="58dp" />

</RelativeLayout>
```

Рисунок 3.19 – content_gps_controll.xml

На рисунках 3.18 и 3.19 видно, что в начале кода объявляется версия и вид кодировки файла. Затем создается слой, на котором все будет отображено. Далее уже идут настройки слоя. И в конечном итоге описываются элементы интерфейса. Указываются ширина, высота, внешние и внутренние отступы.

Для того чтобы реализовать первое окно создается Activity под названием DeviceList, в котором содержится код, применяемый для вывода списка всех доступных Bluetooth-устройств. Фрагмент кода отображен на рисунке 3.20.

```
Button btnPaired;
ListView deviceList;
private BluetoothAdapter myBluetooth = null;
private Set<BluetoothDevice> pairedDevices;
public static String EXTRA_ADDRESS = "device_address";

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_device_list);
    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);
    deviceList = (ListView) findViewById(R.id.listView);

    btnPaired = (Button) findViewById(R.id.button);
    btnPaired.setOnClickListener((v) → { pairedDevicesList(); });

    myBluetooth = BluetoothAdapter.getDefaultAdapter();
    if (myBluetooth == null){

        // Show a message that the device has no bluetooth adapter
        Toast.makeText(getApplicationContext(), text: "Bluetooth Device Not Available", Toast.LENGTH_LONG).show();

        // Finish apk
        finish();
    }else if (!myBluetooth.isEnabled()){
        // Ask to the user turn the bluetooth on
        Intent turnBTon = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
        startActivityForResult(turnBTon, requestCode: 1);
    }
}
```

Рисунок 3.20 – Метод onCreate Activity DeviceList

В этом фрагменте кода создается класс DeviceList. Внутри класса объявляются и определяются объекты, такие как: Button, ListView и BluetoothAdapter. Метод onCreate() вызывается во время создания или перезапуске активности. Система может запускать и останавливать текущие окна в зависимости от происходящих событий. Внутри данного метода настраивается статический интерфейс активности. Также настраивается поведение системы при отсутствии на мобильном устройстве Bluetooth или его неактивности.

Далее в классе DeviceList создается метод pairedDevicesList, предназначенный поиска доступных Bluetooth-устройств. При нахождении хотя бы одного устройства, он выводит его название и адрес на экран смартфона. Иначе, если он не найдет Bluetooth-устройства, на экран выйдет сообщение. Также устанавливается обработчик событий. При нажатии на любой элемент списка будет установлена связь по Bluetooth. Фрагмент этого кода изображен на рисунке 3.21.

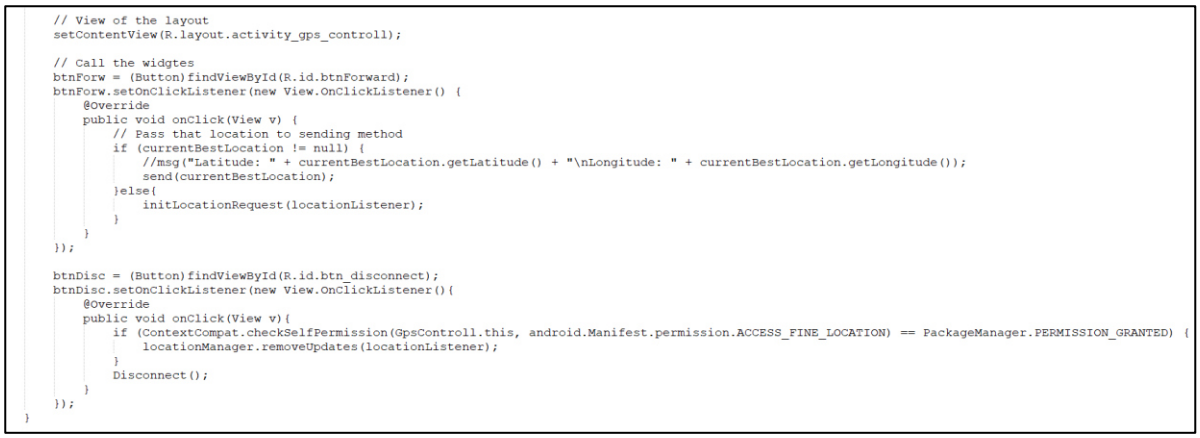
```
private void pairedDevicesList(){
    pairedDevices = myBluetooth.getBondedDevices();
    ArrayList list = new ArrayList();

    if (pairedDevices.size() > 0){
        for (BluetoothDevice bt : pairedDevices){
            list.add(bt.getName() + "\n" + bt.getAddress()); //Get the device's name and the address
        }
    }else{
        // We didn't find any paired bluetooth devices.
        Toast.makeText(getApplicationContext(), text: "No Paired Bluetooth Devices Found.", Toast.LENGTH_LONG).show();
    }

    final ArrayAdapter adapter = new ArrayAdapter( context: this,android.R.layout.simple_list_item_1, list);
    deviceList.setAdapter(adapter);
    deviceList.setOnItemClickListener(myListClickListener); //Method called when the device from the list is clicked
}

private AdapterView.OnItemClickListener myListClickListener = (av, v, arg2, arg3) → {
        // Get the device MAC address, the last 17 chars in the View
        String info = ((TextView) v).getText().toString();
        String address = info.substring(info.length() - 17);

        // Make an intent to start next activity.
        Intent i = new Intent( packageContext: DeviceList.this, GpsControll.class);

        //Change the activity.
        i.putExtra(EXTRA_ADDRESS, address); //this will be received at GpsControll (class) Activity
        startActivity(i);
};
```

Рисунок 3.21 – Методы работы со списком

После того как пользователь выбрал к Bluetooth-устройство, он переходит на следующую активность GpsControll. В этой активности происходит передача данных между приложением и «умным чемоданом». На рисунках 3.22 и 3.23 отображен программный код метода onCreate. В данном методе производятся первоначальные настройки активности: переписываются существующие методы и создаются новые. Также создаются новые обработчики событий для нажатия.

```
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_gps_controll);
        locationManager = (LocationManager) this.getSystemService(Context.LOCATION_SERVICE);

        locationListener = new LocationListener() {
            @Override
            public void onLocationChanged(Location location) {
                // New Location found by the network location provider
                if (isBetterLocation(location, currentBestLocation)){
                    currentBestLocation = location;
                    msg("New Location");
                }
            }

            @Override
            public void onStatusChanged(String provider, int status, Bundle extras) {}

            @Override
            public void onProviderEnabled(String provider) {}

            @Override
            public void onProviderDisabled(String provider) {}
        };

        // Receive the address of the bluetooth device
        Intent newint = getIntent();
        address = newint.getStringExtra(DeviceList.EXTRA_ADDRESS);
        new ConnectBT().execute();

        // Register the listener with location manager to start requesting location data
        if (ContextCompat.checkSelfPermission(this, android.Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions(this,
                    new String[]{android.Manifest.permission.ACCESS_FINE_LOCATION},
                    MY_PERMISSIONS_REQUEST_READ_CONTACTS);
        }else {
            initLocationRequest(locationListener);
        }
```

Рисунок 3.22 – Метод onCreate класса GpsControll



```
    // View of the layout
    setContentView(R.layout.activity_gps_controll);

    // Call the widgtes
    btnForw = (Button)findViewById(R.id.btnForward);
    btnForw.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            // Pass that location to sending method
            if (currentBestLocation != null) {
                //msg("Latitude: " + currentBestLocation.getLatitude() + "\nLongitude: " + currentBestLocation.getLongitude());
                send(currentBestLocation);
            }else{
                initLocationRequest(locationListener);
            }
        }
    });

    btnDisc = (Button)findViewById(R.id.btn_disconnect);
    btnDisc.setOnClickListener(new View.OnClickListener(){
        @Override
        public void onClick(View v){
            if (ContextCompat.checkSelfPermission(GpsControll.this, android.Manifest.permission.ACCESS_FINE_LOCATION) == PackageManager.PERMISSION_GRANTED) {
                locationManager.removeUpdates(locationListener);
            }
            Disconnect();
        }
    });
}
```

Рисунок 3.23 – Продолжение метода onCreate

В GpsControll идет передача данных о местоположении, а также передача сигнала между Bluetooth-модулем и мобильным приложением. На рисунке 3.24 отображена функция для передачи широты и долготы от GPS-модуля.

```
private void send(Location location)
{
    if (btSocket!=null)
    {
        try
        {
            msg(("<" + Double.toString(location.getLatitude()) + "," + Double.toString(location.getLongitude()) + ">"));
            btSocket.getOutputStream().write(("<" + Double.toString(location.getLatitude()) + "," + Double.toString(location.getLongitude()) + ">").getBytes());
        }
        catch (IOException e)
        {
            msg( s: "Error");
        }
    }
}
```

Рисунок 3.24 – Метод передачи координат

В корневой папке каждого приложения должен находиться файл AndroidManifest.xml. Файл манифеста содержит важную информацию о приложении, которая требуется системе Android. Только получив эту информацию, система может выполнить какой-либо код приложения. Среди прочего файл манифеста выполняет следующие действия:

– задает имя пакета Java для приложения. Это имя пакета служит уникальным идентификатором приложения;

– описывает компоненты приложения - операции, службы, приемники широковещательных сообщений и поставщиков контента, из которых состоит приложение. Он содержит имена классов, которые реализуют каждый компонент, и публикует их возможности. На основании этих деклараций система Android может определить, из каких компонентов состоит приложение и при каких условиях их можно запускать;

– определяет, в каких процессах будут размещаться компоненты приложения;

– объявляет, какие разрешения должны быть выданы приложению, чтобы оно могло получить доступ к защищенным частям API-интерфейса и взаимодействовать с другими приложениями;

– также объявляет разрешения, требуемые для взаимодействия с компонентами данного приложения;

– содержит список классов Instrumentation, которые при выполнении приложения предоставляют сведения о профиле и прочую информацию. Эти объявления присутствуют в файле манифеста только во время разработки и отладки приложения и удаляются перед его публикацией;

– объявляет минимальный уровень API-интерфейса Android, который требуется приложению;

– содержит список библиотек, с которыми должно быть связано приложение.

На рисунке 3.25 изображен код манифеста для мобильного приложения.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.suitcase">


    <uses-permission android:name="android.permission.BLUETOOTH" />
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />

    <uses-feature android:name="android.hardware.location.gps" />


    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity
            android:name=".DeviceList"
            android:label="@string/app_name"
            android:theme="@style/AppTheme.NoActionBar">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".GpsControll"
            android:label="@string/title_activity_gps_controll"
            android:theme="@style/AppTheme.NoActionBar"></activity>
    </application>

</manifest>
```

Рисунок 3.25 - AndroidManifest.xml

4 Оценка экономической эффективности системы

В данной дипломной работе описана разработка системы «умный чемодан», предназначенная для улучшения обычных чемоданов, т.е. добавление gps, bluetooth, powerbank модулей и обеспечение моторной системой для автоматического передвижения за его владельцем.

4.1 Трудоемкость разработки системы

Для определения трудоемкости разработки системы нужно составить список всех основных этапов и видов работ, которые должны быть выполнены в процессе реализации «умного чемодана». В таблице 4.1 показана трудоемкость системы.

Таблица 4.1 - Распределение работ и оценка их трудоемкости

Этап разработки	Вид работы на данном этапе	Трудоемкость разработки, чел.× ч.
Формулировка задания	Выбор цели	5
	Обоснование выбора средств разработки	4
Анализ	Исследование рынка	6
	Анализ существующих систем	4
Алгоритмизация	Описание алгоритмов и различных процессов	12
Создание аппаратной части	Проектирование системы	62
	Сборка аппаратной части	55
	Программирование микроконтроллера	33
Создание мобильного приложения	Проектирование пользовательского интерфейса	36
	Разработка приложения	50
Отладка и тестирование системы	Проверка работоспособности системы	20
	Исправление ошибок	24
Документация	Составление документации	11
ИТОГО трудоемкость выполнения системы		322

4.2 Расчет затрат на разработку системы

В данном разделе требуется рассчитать затраты на все материальные ресурсы, и затраты на расход электроэнергии, общие затраты на амортизационные отчисления, а также прочие расходы.

Необходимо определить общую сумму затрат на различные материальные ресурсы (Зм), по формуле, которая представлена ниже:

$$З_м = \sum_{i=1}^{n} Р_i \times Ц_i \quad (4.1)$$

где $Р_i$ - расход i-го вида материального ресурса, натуральные единицы;

$Ц_i$ - цена за единицу i-го вида материального ресурса, тг;

i - вид материального ресурса;

n - количество видов материальных ресурсов.

$$З_{общие} = 1\,500 + 2\,000 * 2 + 3\,000 * 2 + 5\,600 + 2\,300 + 1\,000 = 20\,400 \text{ тг}$$

Расчет затрат на материальные ресурсы представлены в таблице 4.2.

Таблица 4.2 - Затраты на материальные ресурсы

Наименование материального ресурса	Ед. измерения	Кол-во материала	Цена за единицу, тг	Сумма, тг
Бумага А4	шт	1	1 500	1 500
Картридж	шт	1	2 000	2 000
Микроконтроллер Arduino Uno	шт	1	3 000	3 000
GPS модуль	шт	1	5 600	5 600
Bluetooth модуль	шт	1	2 300	2 300
Литиевый аккумулятор	шт	1	3 000	3 000
Powerbank модуль	шт	1	1 000	1 000
Электромотор	шт	2	1 000	2 000
ИТОГО				20 400

Так как для разработки системы используется электричество, то необходимо рассчитать затраты на электроэнергию, которые представлены в таблице 4.3.

Общая сумма затрат на электроэнергию ($З_э$) рассчитывается по формуле:

$$З_э = \sum_{i=1}^{n} М_i \times К_i \times Т_i \times Ц, \quad (4.2)$$

где M_i - паспортная мощность i-го электрооборудования, кВт;

K_i - коэффициент использования мощности i-го электрооборудования (принимается K_i=0.7¸0.9);

T_i - время работы i-го оборудования за весь период разработки системы, ч;

Ц - цена электроэнергии, тг/кВт×ч;

i - вид электрооборудования;

n - количество электрооборудования.

$$З_{э\ ноутбук} = 0{,}315 \times 0{,}7 \times 322 \times 18{,}32 = 1\ 300{,}74 \text{ тенге}$$

$$З_{э\ принтер} = 0{,}205 \times 0{,}7 \times 7 \times 18{,}32 = 18{,}4 \text{ тенге}$$

$$З_{э\ общие} = 1\ 300{,}74+69{,}57+160{,}3+96{,}18+96{,}18+18{,}4 = 1\ 741{,}37 \text{ тенге}$$

Таблица 4.3 - Затраты на электроэнергию

Оборудование	Паспорт. мощность, кВт	Коэффициент использова-ния мощности	Время работы оборудования для разработки системы, ч	Цена эл-и, тг/кВт *ч	Сумма, тг
Ноутбук Asus	0,315	0,7	322	18,32	1 300,74
Смартфон Samsung Galaxy A5	0,155	0,7	35	18,32	69,57
Микроконт-роллер Arduino	0,05	0,7	250	18,32	160,3
Bluetooth модуль	0,03	0,7	250	18,32	96,18
GPS модуль	0,03	0,7	250	18,32	96,18
Принтер лазерный Samsung SCX	0,205	0,7	7	18,32	18,4
ИТОГО					1 741,37

В статье «Затраты на оплату труда» необходимо включить расходы по оплате труда всех работников, которые участвуют в разработке системы «Умный чемодан». Затраты на оплату труда отображены в таблице 4.4.

Таблица 4.4 - Затраты на оплату труда

Категория работника	Квалификация	Трудоемкость разработки системы, чел.×ч	Часовая ставка, тг/ч	Сумма, тг
Служащий	Программист	322	1 148,81	369 916,82
Служащий	Руководитель проекта	40	880,95	35 238
ИТОГО				405 154,82

Общая сумма затрат на оплату труда ($З_{ТР}$) определяется по формуле:

$$З_{тр} = \sum_{i=1}^{n} ЧС_i \times Т_i \quad (4.3)$$

где $ЧС_i$ - часовая ставка i-го работника, тг;
$Т_i$ - трудоемкость разработки ПП, чел.×ч;
i - категория работника;
n - количество работников, занятых разработкой ПП.

Часовая ставка работника может быть рассчитана по формуле:

$$ЧС_i = \frac{ЗП_i}{ФРВ_i} \quad (4.4)$$

$$ЧС_i = \frac{ЗП_i}{ФРВ_i} = \frac{193\,000}{168} = 1\,148{,}81$$

$$ЧС_{рук} = \frac{ЗП_{рук}}{ФРВ_{рук}} = \frac{148\,000}{168} = 880{,}95$$

где ФРВ – фонд рабочего времени сотрудника, который составляет примерно 168 часов в месяц.

В статью «Социальный налог» включается сумма, которая рассчитывается как 9,5% на социальный налог и 1,5% на медицинскую страховку от затрат на оплату труда всех работников ($З_{тр}$), занятых разработкой системы. В ходе расчета нужно учесть, что пенсионные отчисления (10% от $З_{тр}$) не облагаются социальным налогом (ставки указаны на 2019 год).

Расчет затрат на оплату труда для программиста:

$$З_{по} = З_{тр} \times 0,1 = 369\ 916,82 \times 0,1 = 36\ 991,682$$

$$З_{тр}\,(\text{с уч п. о. }) = З_{тр} - З_{по} = 369\ 916,82 - 36\ 991,682 = 332\ 925,138$$

$$Н_с = З_{тр}(\text{с уч п. о. }) \times 0,11 = 332\ 925,138 \times 0,11 = 36\ 621,7652$$

Расчет затрат на оплату труда для руководителя проекта:

$$З_{по} = З_{тр} \times 0,1 = 35\ 238 \times 0,1 = 3\ 523,8$$

$$З_{тр}\,(\text{с уч п. о. }) = З_{тр} - З_{по} = 35\ 238 - 3\ 523,8 = 31\ 714,2$$

$$Н_с = З_{тр}(\text{с уч п. о. }) \times 0,11 = 31\ 714,2 \times 0,11 = 3\ 488,562$$

Таким образом отчисления на социальные нужды составляют:

$$36\ 621,7652 + 3\ 488,562 = 40\ 110,3272 \text{ тенге}$$

В статью «Амортизация основных фондов» включается сумма амортизационных
Общая сумма амортизационных отчислений определяется по формуле:

$$З_{АМ} = \sum_{i=1}^{n} \frac{Ф_i \times Н_{Ai} \times Т_{НИРi}}{100 \times Т_{Эфi}} \qquad (4.5)$$

где $Ф_i$ - стоимость i-го ОФ, тг;

$Н_{Ai}$ - годовая норма амортизации i-го ОФ, %;

$Т_{НИРi}$ - время работы i-го ОФ за весь период разработки системы, ч;

$Т_{Эфi}$ - эффективный фонд времени работы i-го ОФ за год, ч/год;

i - вид ОФ;

n - количество ОФ.

Годовые нормы амортизации ОФ принимаются исходя из возможного срока полезного использования ОФ:

$$Н_{Ai} = \frac{100}{Т_{Ni}} \qquad (4.6)$$

$$Н_{А1} = Н_{А6} = \frac{100}{4} = 25\%$$

$$Н_{А2} = Н_{А3} = Н_{А4} = \frac{100}{6} = 16,67\%$$

$$Н_{А5} = \frac{100}{7} = 14,3\%$$

где T_{Ni} - возможный срок использования i-го ОФ, год.

Возможный срок полезного использования ОФ может быть принят от 3 до 10 лет (по согласованию с консультантом по экономической части).

$$З_{АМ1} = \frac{200\,000 \times 25 \times 322}{100 \times 1968} = 8\,180{,}9 \text{ тенге}$$

$$З_{АМ2} = \frac{3\,000 \times 16{,}67 \times 250}{100 \times 1968} = 63{,}53 \text{ тенге}$$

$$З_{АМ3} = \frac{5\,600 \times 16{,}67 \times 250}{100 \times 1968} = 118{,}59 \text{ тенге}$$

$$З_{АМ4} = \frac{2\,300 \times 16{,}67 \times 250}{100 \times 1968} = 48{,}7 \text{ тенге}$$

$$З_{АМ5} = \frac{50\,000 \times 14{,}3 \times 7}{100 \times 1968} = 25{,}43 \text{ тенге}$$

$$З_{АМобщ} = 8\,180{,}9 + 63{,}53 + 118{,}59 + 48{,}7 + 25{,}43 + 457{,}9 = 8\,895{,}05 \text{ тенге}$$

Таблица 4.5 - Амортизация основных фондов (ОФ)

Наименование оборудования	Цена оборудования, тг	Годовая норма амортизации, %	Эффект. фонд врни работы оборудования, ч/год	Время работы оборудования, ч	Сумма, тг
Ноутбук Asus	200 000	25	1968	322	8 180,9
Микроконтроллер Arduino	3 000	16,67	1968	250	63,53
GPS модуль	5 600	16,67	1968	250	118,59
Bluetooth модуль	2 300	16,67	1968	250	48,7
Принтер лазерный Samsung SCX	50 000	14,3	1968	7	25,43

Продолжение таблицы 4.5

Наименование оборудования	Цена оборудования, тг	Годовая норма амортизации, %	Эффект. фонд врни работы оборудова	Время работы оборудования, ч	Сумма, тг

			ния, ч/год		
Смартфон Samsung	103 000	25	1968	35	457,9
ИТОГО					8 895,05

В статью «Прочие затраты» включаются расходы на арендную плату, включая коммунальные платежи, затраты на лицензирование и сертификацию, расходы на рекламу, канцелярские и прочие хозяйственные расходы. Затраты на арендную плату указаны в таблице 4.6.1, на интернет в таблице 4.6.2.

Таблица 4.6.1 – Расходы на арендную плату

Площадь, $м^2$	Цена за 1 $м^2$, тг	Стоимость за 1 месяц, тг	Длительность аренды (в месяцах)	Сумма, тг
40	3000	120 000	2	240 000

Таблица 4.6.2 – Расходы за интернет-услуги

Цена за 1 месяц, тг	Количество месяцев	Сумма, тг
4 000	2	8 000

Затраты на прочие расходы составят 240 000 + 8 000 = 248 000 тенге.

На основании полученных данных по отдельным статьям составляется смета затрат на разработку системы по форме, приведенной в таблице 4.6.

Таблица 4.6 - Смета затрат на разработку системы

Статьи затрат	Сумма, тг	Проценты, %
1. Материальные затраты, в том числе: - материалы - электроэнергия	 20 400 1 741,37	 2,82 0,24
2. Затраты на оплату труда	405 154,82	55,94
3. Отчисления на социальные нужды	40 110,3272	5,54
4. Амортизация основных фондов	8 895,05	1,23
5. Прочие затраты	248 000	34,23
ИТОГО по смете	724 301,57	100

Структура себестоимости системы представлена на рисунке 4.1.

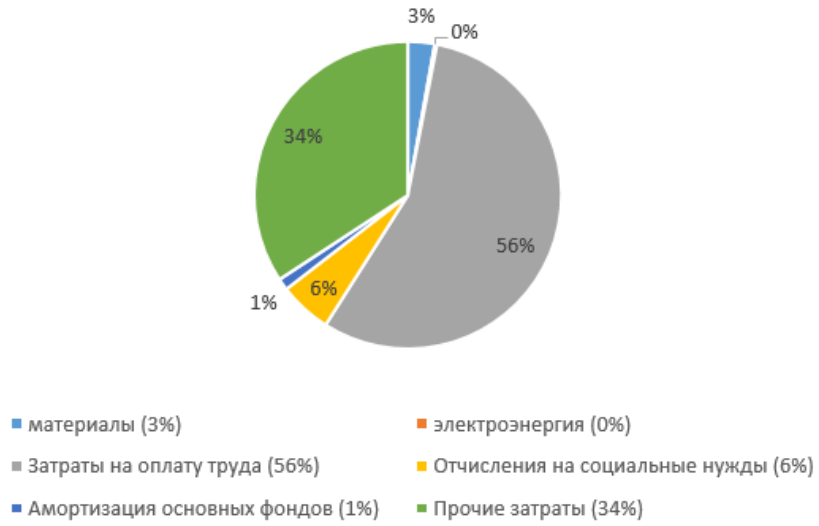


Рисунок 4.1. Смета затрат по разработке системы «Умный чемодан»

4.3 Определение договорной цены

Величина договорной цены системы должна быть определенна с учетом эффективности, качества и сроков ее выполнения на уровне, отвечающем экономическим интересам заказчика и исполнителя. Договорная цена ($Ц_Д$) рассчитывается по формуле:

$$Ц_д = З_{нир} \times \left(1 + \frac{Р}{100}\right) \quad (4.7)$$

где $З_{нир}$ - затраты на разработку системы (из таблицы 4.6), тг;
Р - средний уровень рентабельности, который составляет 25%.

$$Ц_д = З_{нир} \times \left(1 + \frac{Р}{100}\right) = 724\,301{,}57 \times (1 + 0{,}25) =$$

$$= 724\,301{,}57 + 181\,075{,}39 = 905\,376{,}96 \text{ тенге}$$

Далее определяется цена реализации с учетом налога на добавленную стоимость (НДС). На 2019 год ставка НДС установлена налоговым кодексом РК в размере 12%. Цена реализации с учетом НДС рассчитывается по формуле:

$$Ц_р = Ц_д + Ц_д \times НДС \quad (4.8)$$

$$Ц_р = 905\,376{,}96 + 905\,376{,}96 \times 0{,}12 = 1\,014\,022{,}19 \text{ тенге}$$

По итогам: себестоимость системы составляет - 724 301,57 тенге; прибыль составляет - 181 075,39 тенге; договорная цена - 1 014 022,19.

5 Безопасность жизнедеятельности

5.1 Условия труда

В данном дипломном проекте разрабатывается система «умный чемодан». Так как в дальнейшем такой чемодан будет разрабатываться на заводе, необходимо проанализировать условия труда в производственном помещении, в котором будет происходить сборка деталей для функционирования системы. Произвести анализ необходимо для того, чтобы обеспечить работникам завода комфортные условия труда и обезопасить их от травм и профессиональных заболеваний.

Основными показателями производственной безопасности сотрудников и условий труда на заводах являются освещение, микроклимат и шум. Необходимо контролировать каждый из этих показателей и, в случае несоответствия их с установленными нормами контроля, следует производить перерасчет до тех пор, пока показатели не станут удовлетворять установленным требованиям. Далее будет произведено исследование каждого аспекта, влияющего на условия труда работников завода.

Одним из рассматриваемых показателей является освещение. На заводе, который будет выпускать «умный чемодан», предусмотрены и правильно спроектированы и естественное, и искусственное освещения. Для естественного освещения используются высокие окна, хорошо пропускающие свет. Для искусственного освещения в производственном помещении используются светодиодные светильники, которые обеспечивают работников завода хорошим светом, не вызывающим отрицательных последствий и располагающим к продуктивному труду.

Помимо освещения, на заводе отслеживается микроклимат производственного помещения. Это выражается в постоянном контролировании воздуха и проветривании помещения при помощи оконных проемов. Помимо этого, на заводе используются подвесные вытяжки, а также напольно-потолочные мульти сплит системы, не мешающие рабочему процессу и обеспечивающие хорошую вентиляцию.

При использовании различных приборов для создания «умного чемодана» на производстве появляется шумная обстановка, которая может отрицательно сказаться на работниках завода. Основным источником шума, используемым на заводе, являются роботизированные сборщики. Поэтому, в данном дипломном проекте необходимо рассчитать, каков уровень шума подобного рода приборов и, в случае превышающих показателей, провести мероприятия по его снижению.

Исходные данные для расчета. Вид оборудования: роботизированный сборщик – 3 шт; расстояние от ИШ до РТ: r1 = 8,8; r2 =10,5; r3 = 9,5; отношение $B/S_{огр} = 1$, $l_{max} = 1,6$; объем помещения = 3200; параметры кабины

наблюдения: 15*10*5; площадь глухой стены: S1 = 75, S2 = 150; площадь двери: S3 = 4; площадь окна: S4 = 3.

5.2 Акустический расчет

Октавные уровни звукового давления L в дБ в расчетных точках помещений, в которых несколько источников шума в зоне прямого и отраженного звука, следует определять по формуле:

$$L_{общ} = 10 \times \lg(\sum_{i=1}^{m} \frac{\Delta i \times \chi \times \Phi_i}{S_i} + \frac{4 \times \psi}{B} \times \sum_{i=1}^{n} \Delta i) \quad (5.1)$$

где m – количество оборудования, которое участвует в акустическом расчете (т.е. источников, для которых $r_i < 3\ r_{min}$);

n – общее количество источников шума в помещении.

Минимальное расстояние от расчетной точки до акустического центра и ближайшего к ней источника r_{min} = 8,8 м, $3*r_{min}$= 26.4 м. Общее количество источников шума, принимаемых в расчет и расположенных вблизи расчетной точки (r <26.4 м), будет равно 3 (m = 3), т.е. учитываются все данные источники, расположенные на расстояниях r_1, r_2, r_3;

Δ_i – рассчитывается по формуле:

$$\Delta_i = 10^{0,1 \cdot Lpi} \quad (5.2)$$

где L_{pi} – октавный уровень звуковой мощности дБ, создаваемый i–тым источником шума. Например, для частоты 63 Гц Lp=103 дБ, то есть $\Delta_i = 1{,}995 \cdot 10^{10}$.

Таблица 5.1 – Уровни звукового давления (дБ)

Вел-на	Среднегеометрическая частота октавной полосы, Гц							
	63	125	250	500	1000	2000	4000	8000
Lpi	103	105	94	95	93	90	88	85
Δ_i	$1{,}995 \cdot 10$	$3{,}16 \cdot 10^{10}$	$0{,}25 \cdot 10^{10}$	$0{,}316 \cdot 10^{1}$	$0{,}1995 \cdot 10^{1}$	$0{,}1 \cdot 10^{10}$	$0{,}6*10^{9}$	$0{,}316 \cdot 10$

χ – коэффициент влияния ближайшего акустического поля и принимаемый в зависимости от отношения r_{min}/l_{max};

l_{max} – наибольший габаритный размер источников шума. Величина r_{min}/l_{max}= 8,8/1,6 = 5,5 т.к. $r_{min}/l_{max} > 1.7$ (по рисунку 5.1) принимаем χ =1;

Ф – фактор направленности источника шума, принимается равным 1;

S – площадь воображаемой поверхности правильной геометрической формы, окружающей источник и проходящий через расчетную точку. Так как для всех источников выполняется условие $2 \cdot l_{max} < r$; $2 \cdot 1.6$ м < 8,8 м, то можно принять $S_i = 2\pi r^2_i$:

$S_1 = 2 \times 3,14 \times 8,8^2 = 486,3232$ (м^2);
$S_2 = 2 \times 3,14 \times 10,5^2 = 692,37$ (м^2);

$S_3 = 2 \times 3,14 \times 9,5^2 = 566,77$ (м^2);

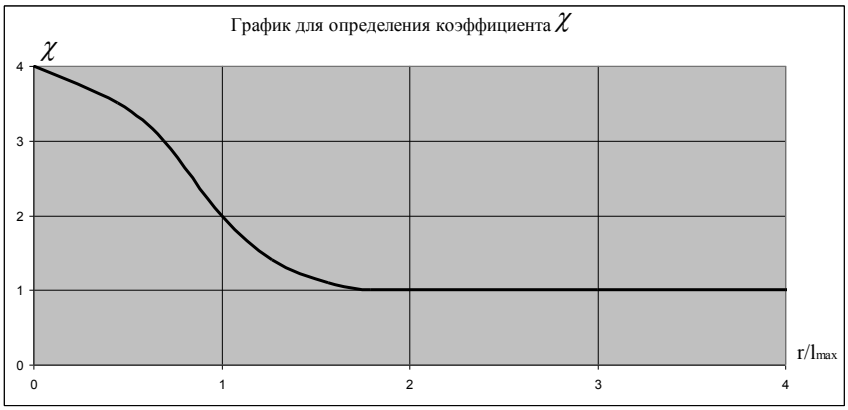


Рисунок 5.1 – График коэффициента χ

Для дальнейшего расчета применяется следующая формула:

$$\sum_{i=1}^{m} \frac{\Delta_i}{S_i} \quad (5.3)$$

Таким образом, производятся следующие вычисления:

Для частоты 63 Гц: $\left(\frac{1}{486,32} + + \frac{1}{692,37} + \frac{1}{566,77}\right) \times 1,99 \times 10^{10} = 1,05 \times 10^8$

Для 125 Гц: $\left(\frac{1}{486,32} + + \frac{1}{692,37} + \frac{1}{566,77}\right) \times 3,16 \times 10^{10} = 1,66 \times 10^8$

Для 250 Гц: $\left(\frac{1}{486,32} + + \frac{1}{692,37} + \frac{1}{566,77}\right) \times 0,25 \times 10^{10} = 1,316 \times 10^7$

Для 500 Гц: $\left(\frac{1}{486,32} + + \frac{1}{692,37} + \frac{1}{566,77}\right) \times 0,316 \times 10^{10} = 1,664 \times 10^7$

Для 1000 Гц: $\left(\frac{1}{486,32} + + \frac{1}{692,37} + \frac{1}{566,77}\right) \times 0,1995 \times 10^{10} = 1,05 \times 10^7$

Для 2000 Гц: $\left(\frac{1}{486,32} + + \frac{1}{692,37} + \frac{1}{566,77}\right) \times 0,1 \times 10^{10} = 5,265 \times 10^6$

Для 4000 Гц: $\left(\frac{1}{486,32} + + \frac{1}{692,37} + \frac{1}{566,77}\right) \times 0,6 \times 10^9 = 3,159 \times 10^6$

Для 8000 Гц: $\left(\frac{1}{486,32} + + \frac{1}{692,37} + \frac{1}{566,77}\right) \times 0,316 \times 10^9 = 1,664 \times 10^6$

ψ – коэффициент нарушения диффузности звукового поля в помещении, принимаемый по опытным значениям, а при их отсутствии – по графику на рисунке 5.2. По графику определяется, что при $B/S_{огр} = 1 \rightarrow \psi = 0,52$;

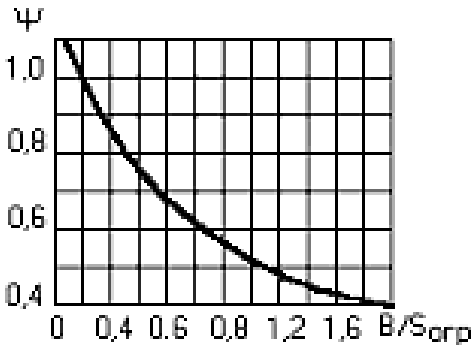


Рисунок 5.2 – Коэффициент нарушения диффузности звукового поля

B – постоянная помещения в м^2, определяемая по формуле:

$$B = B_{1000} \times \mu \qquad (5.4)$$

где B_{1000} – постоянная на среднегеометрической частоте 1000 Гц;
μ – частотный множитель.

Следует определить постоянную помещения B_{1000}. Выбирается тип помещения 1 – с небольшим количеством людей (металлообрабатывающие цехи, вентиляционные камеры, машинные залы, генераторные, испытательные стенды и т.п.), тогда

$$B_{1000} = V / 20 = 3200 / 20 = 160 \text{ м}^2$$

Приводится значение частотного множителя μ в таблице 5.2 для объема помещения V=3200 м^3.

Таблица 5.2 – Значение частотного множителя для V>1000.

Частота	63	125	250	500	1000	2000	4000	8000
M	0,5	0,5	0,55	0,7	1	1,6	3	6

Далее производится расчет $B=B_{1000} \cdot \mu$, где для V = 3200 (м^3) и для частоты 63 Гц → μ = 0,5. Тогда значения B принимают вид:

$$B_{63} = B_{125} = 160 \times 0,5 = 80$$

$$B_{250} = 160 \times 0{,}55 = 88$$

$$B_{500} = 160 \times 0{,}7 = 112$$

$$B_{1000} = 160 \times 1 = 160$$

$$B_{2000} = 160 \times 1{,}6 = 256$$

$$B_{4000} = 160 \times 3 = 480$$

$$B_{8000} = 160 \times 6 = 960$$

Далее нужно произвести расчет второй части формулы (5):

$$\frac{4\psi}{B_{63}} \times \sum_{i=1}^{5} \Delta_i = \frac{4 \times 0{,}52}{80} \times 1{,}995 \times 10^{10} = 5{,}187 \times 10^{8}$$

$$\frac{4\psi}{B_{125}} \times \sum_{i=1}^{5} \Delta_i = \frac{4 \times 0{,}52}{80} \times 3{,}16 \times 10^{10} = 8{,}216 \times 10^{8}$$

$$\frac{4\psi}{B_{250}} \times \sum_{i=1}^{5} \Delta_i = \frac{4 \times 0{,}52}{88} \times 0{,}25 \times 10^{10} = 5{,}909 \times 10^{7}$$

$$\frac{4\psi}{B_{500}} \times \sum_{i=1}^{5} \Delta_i = \frac{4 \times 0{,}52}{12} \times 0{,}316 \times 10^{10} = 5{,}869 \times 10^{7}$$

$$\frac{4\psi}{B_{1000}} \times \sum_{i=1}^{5} \Delta_i = \frac{4 \times 0{,}52}{160} \times 0{,}1995 \times 10^{10} = 2{,}594 \times 10^{7}$$

$$\frac{4\psi}{B_{2000}} \times \sum_{i=1}^{5} \Delta_i = \frac{4 \times 0{,}52}{256} \times 0{,}1 \times 10^{10} = 8{,}125 \times 10^{6}$$

$$\frac{4\psi}{B_{4000}} \times \sum_{i=1}^{5} \Delta_i = \frac{4 \times 0{,}52}{480} \times 0{,}6 \times 10^{9} = 2{,}6 \times 10^{6}$$

$$\frac{4\psi}{B_{8000}} \times \sum_{i=1}^{5} \Delta_i = \frac{4 \times 0{,}52}{960} \times 0{,}316 \times 10^9 = 6{,}487 \times 10^5$$

Затем, подставив значения, находится $L_{общ}$:

$$L_{общ}(B_{63}) = 10 \times \log(1{,}05 \times 10^8 + 5{,}187 \times 10^8) = 87{,}95 \text{ дБ}$$

$$L_{общ}(B_{125}) = 10 \times \log(1{,}664 \times 10^8 + 8{,}216 \times 10^8) = 89{,}948 \text{ дБ}$$

$$L_{общ}(B_{250}) = 10 \times \log(1{,}316 \times 10^7 + 5{,}909 \times 10^7) = 78{,}588 \text{ дБ}$$

$$L_{общ}(B_{500}) = 10 \times \log(1{,}664 \times 10^7 + 5{,}869 \times 10^7) = 78{,}77 \text{ дБ}$$

$$L_{общ}(B_{1000}) = 10 \times \log(1{,}05 \times 10^7 + 2{,}594 \times 10^7) = 75{,}616 \text{ дБ}$$

$$L_{общ}(B_{2000}) = 10 \times \log(5{,}265 \times 10^6 + 8{,}125 \times 10^6) = 71{,}268 \text{ дБ}$$

$$L_{общ}(B_{4000}) = 10 \times \log(3{,}159 \times 10^6 + 2{,}6 \times 10^6) = 67{,}603 \text{ дБ}$$

$$L_{общ}(B_{8000}) = 10 \times \log(1{,}664 \times 10^6 + 6{,}847 \times 10^5) = 63{,}708 \text{ дБ}$$

Необходимо определить требуемое снижение шума $\Delta L_{тр}$, приняв нормативные уровни звукового давления в расчетной точке по таблице из методических указаний (таблица 3): рабочие место – постоянные рабочие места и рабочие зоны в производственных помещениях и на территории предприятий.

$$\Delta L_{тр} = L_{общ} - L_{доп}, \text{ дБ} \qquad (5.5)$$

где $L_{общ}$ – октавный уровень звукового давления в расчетной точке от всех источников шума, дБ;

$L_{доп}$ – указаны в таблице 5.3.

Таблица 5.3 – Допустимые уровни звукового давления

Среднегеометрические частоты октавных полос, Гц	63	125	250	500	1000	2000	4000	8000
$L_{доп}$	99	92	86	83	80	78	76	74

Тогда $\Delta L_{тр i}$ равно:

$$\Delta L_{тр\ 63} = L_{общ} - L_{доп} = 87,95 - 99 = -11,05 \text{ дБ}$$

$$\Delta L_{тр\ 125} = L_{общ} - L_{доп} = 89,948 - 92 = -2,052 \text{ дБ}$$

$$\Delta L_{тр\ 250} = L_{общ} - L_{доп} = 78,588 - 86 = -7,412 \text{ дБ}$$

$$\Delta L_{тр\ 500} = L_{общ} - L_{доп} = 78,77 - 83 = -4,23 \text{ дБ}$$

$$\Delta L_{тр\ 1000} = L_{общ} - L_{доп} = 75,616 - 80 = -4,384 \text{ дБ}$$

$$\Delta L_{тр\ 2000} = L_{общ} - L_{доп} = 71,268 - 78 = -6,732 \text{ дБ}$$

$$\Delta L_{тр\ 4000} = L_{общ} - L_{доп} = 67,603 - 76 = -8,397 \text{ дБ}$$

$$\Delta L_{тр\ 8000} = L_{общ} - L_{доп} = 63,708 - 74 = -10,292 \text{ дБ}$$

Таблица 5.4 – Окончательная таблица

Вел-на	Среднегеометрическая частота октавной полосы, Гц							
	63	125	250	500	1000	2000	4000	8000
L_{pi}, дБ	103	105	94	95	93	90	88	85
Δ_i	$1,99 \cdot 10^{10}$	$3,16 \cdot 10^{10}$	$0,25 \cdot 10^{10}$	$0,316 \cdot 10^{10}$	$0,12 \cdot 10^{10}$	$0,1 \cdot 10^{10}$	$0,6 * 10^{9}$	$0,316 \cdot 10^{9}$
μ	0,5	0,5	0,55	0,7	1	1,6	3	6
B	80	80	88	112	160	256	480	960
$L_{общ}$, дБ	87,95	89,948	78,588	78,77	75,616	71,268	67,603	63,708
$L_{доп}$, дБ	99	92	86	83	80	78	76	74
$\Delta L_{тр}$, дБ	-11,05	-2,052	-7,412	-4,23	-4,384	-6,732	-8,397	-10,292

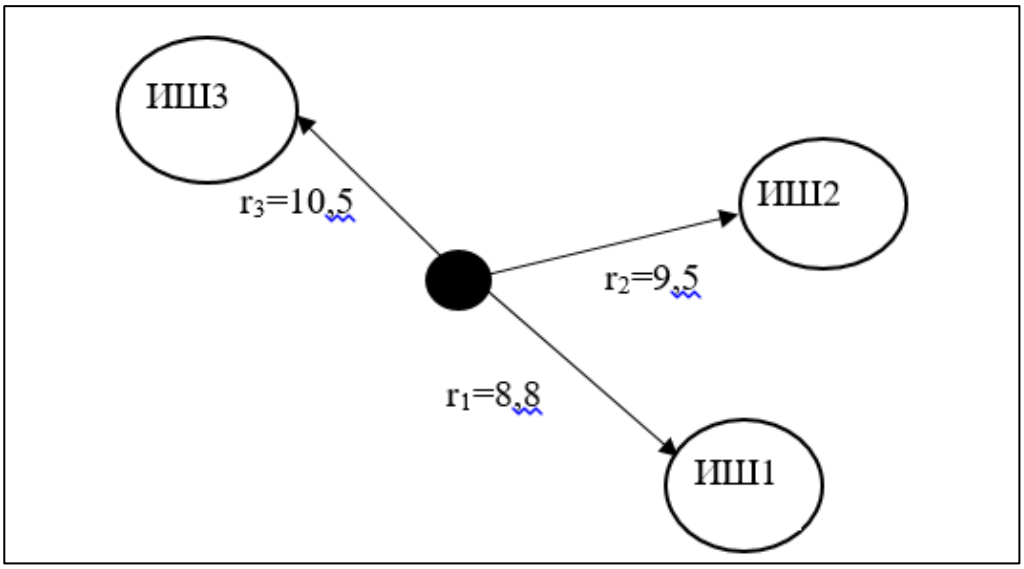


Рисунок 5.3 – Схема расположения расчетной точки и источников шума

Вывод по безопасности жизнедеятельности**:** по данным расчета, шум от роботизированного сборщика совсем не значительный, но следует произвести мероприятия по его максимальному снижению.

5.3 Расчет мероприятий для снижения шума

Спроектировать стену (с окном и дверью) и перекрытием кабины наблюдения, имеющего размер 15*10*5 м. Площадь глухой стены S_1 и перекрытия кабины наблюдения S_2 соответственно равны 75 и 150 м^2, площадь двери S_3 = 4 м^2, окна S_4= 3 м^2. Суммарный уровень звуковой мощности $L_{рСУМ}$, излучаемой всеми источниками шума, приведен в таблице 5.5.

Таблица 5.5 – Суммарный уровень звуковой мощности

Среднегеометрические частоты октавных полос, Гц	63	125	250	500	1000	2000	4000	8000
$L_ш$	93	90	91	108	117	116	115	117

Требуемую звукоизолирующую способность каждого элемента наблюдательной кабины рассчитывается по формуле:

$$R_{тр} = L_{ш} - lgB_и + 10lgS_i - L_{доп} + 10lg\ n \qquad (5.6)$$

где n = 3;

$L_ш$ – октавный уровень звукового давления вне защищаемого от шума помещения, указанного в таблице 5.4;

B_u – постоянная защищаемого от шума помещения, м^2, определяемая по формуле:

$$B = B_{u1000} * \mu \qquad (5.7)$$

где B_{u1000} – постоянная на среднегеометрической частоте 1000 Гц;

μ – частотный множитель. Определим постоянную помещения B_{u1000}.

Из методических указаний, выбирается тип помещения 2 – с жесткой мебелью и небольшим (или большим) количеством людей (лаборатории, кабинеты и т.д.), тогда:

$$B_{1000} = V / 10 = 15*10*5 / 10 = 75\ м^2 \qquad (5.8)$$

Следует привести значение частотного множителя μ в таблице 7 для V = 750 м^3 (Данные взяты из таблицы 3.9 методических указаний).

Таблица 5.6 – Значение частотного множителя для V = 200…1000.

Частота	63	125	250	500	1000	2000	4000	8000
μ	0,65	0,62	0,64	0,75	1	1,5	2,4	4,2

Далее производится расчет $B = B_{1000} \cdot \mu$, где для V=750 ($м^3$) и для частоты 63 Гц → μ = 0,65;

Тогда значения B равны:

$$B_{63} = 75 \times 0{,}65 = 48{,}75$$

$$B_{125} = 75 \times 0{,}62 = 46{,}5$$

$$B_{250} = 75 \times 0{,}64 = 48$$

$$B_{500} = 75 \times 0{,}75 = 56{,}25$$

$$B_{1000} = 75$$

$$B_{2000} = 75 \times 1{,}5 = 112{,}5$$

$$B_{4000} = 75 \times 2{,}4 = 180$$

$$B_{8000} = 75 \times 4{,}2 = 315$$

Для частоты 63:

$$R_{тр}(S_1) = 93 - \lg(48{,}75) + 10\lg(75) - 99 + 10\lg(3) = 15{,}834$$

$$R_{тр}(S_2) = 93 - \lg(48{,}75) + 10\lg(150) - 99 + 10\lg(3) = 18{,}844$$

$$R_{тр}(S_3) = 93 - \lg(48{,}75) + 10\lg(4) - 99 + 10\lg(3) = 3{,}104$$

$$R_{тр}(S_4) = 93 - \lg(48{,}75) + 10\lg(3) - 99 + 10\lg(3) = 1{,}854$$

Для частоты 125:

$$R_{тр}(S_1) = 90 - \lg(46{,}5) + 10\lg(75) - 92 + 10\lg 3 = 19{,}854$$

$$R_{тр}(S_2) = 90 - \lg(46{,}5) + 10\lg(150) - 92 + 10\lg 3 = 22{,}865$$

$$R_{тр}(S_3) = 90 - \lg(46{,}5) + 10\lg(4) - 92 + 10\lg 3 = 7{,}124$$

$$R_{тр}(S_4) = 90 - \lg(46{,}5) + 10\lg(3) - 92 + 10\lg 3 = 5{,}875$$

Для частоты 250:

$$R_{тр}(S_1) = 91 - \lg(48) + 10\lg(75) - 86 + 10\lg 3 = 26{,}841$$

$$R_{тр}(S_2) = 91 - \lg(48) + 10\lg(150) - 86 + 10\lg 3 = 29{,}851$$

$$R_{тр}(S_3) = 91 - \lg(48) + 10\lg(4) - 86 + 10\lg 3 = 14{,}111$$

$$R_{тр}(S_4) = 91 - \lg(48) + 10\lg(3) - 86 + 10\lg 3 = 12{,}861$$

Для частоты 500:

$$R_{тр}(S_1) = 108 - \lg(56{,}25) + 10\lg(75) - 83 + 10\lg 3 = 46{,}772$$

$$R_{тр}(S_2) = 108 - \lg(56{,}25) + 10\lg(150) - 83 + 10\lg 3 = 49{,}782$$

$$R_{тр}(S_3) = 108 - \lg(56{,}25) + 10\lg(4) - 83 + 10\lg 3 = 34{,}042$$

$$R_{тр}(S_4) = 108 - \lg(56{,}25) + 10\lg(3) - 83 + 10\lg 3 = 32{,}792$$

Для частоты 1000:

$$R_{тр}(S_1) = 117 - \lg(75) + 10\lg(75) - 80 + 10\lg 3 = 58{,}647$$

$$R_{тр}(S_2) = 117 - \lg(75) + 10\lg(150) - 80 + 10\lg 3 = 61{,}657$$

$$R_{тр}(S_3) = 117 - \lg(75) + 10\lg(4) - 80 + 10\lg 3 = 45{,}917$$

$$R_{тр}(S_4) = 117 - \lg(75) + 10\lg(3) - 80 + 10\lg 3 = 44{,}667$$

Для частоты 2000:

$$R_{тр}(S_1) = 116 - \lg(112{,}5) + 10\lg(75) - 78 + 10\lg 3 = 59{,}471$$

$$R_{тр}(S_2) = 116 - \lg(112{,}5) + 10\lg(150) - 78 + 10\lg 3 = 62{,}481$$

$$R_{тр}(S_3) = 116 - \lg(112{,}5) + 10\lg(4) - 78 + 10\lg 3 = 46{,}741$$

$$R_{тр}(S_4) = 116 - \lg(112{,}5) + 10\lg(3) - 78 + 10\lg 3 = 45{,}491$$

Для частоты 4000:

$$R_{тр}(S_1) = 115 - \lg(180) + 10\lg(75) - 76 + 10\lg 3 = 60{,}267$$

$$R_{тр}(S_2) = 115 - \lg(180) + 10\lg(150) - 76 + 10\lg 3 = 63{,}277$$

$$R_{тр}(S_3) = 115 - \lg(180) + 10\lg(4) - 76 + 10\lg 3 = 47{,}537$$

$$R_{тр}(S_4) = 115 - \lg(180) + 10\lg(3) - 76 + 10\lg 3 = 46{,}287$$

Для частоты 8000:

$$R_{тр}(S_1) = 117 - \lg(315) + 10\lg(75) - 74 + 10\lg 3 = 64{,}024$$

$$R_{тр}(S_2) = 117 - \lg(315) + 10\lg(150) - 74 + 10\lg 3 = 67{,}034$$

$$R_{тр}(S_3) = 117 - \lg(315) + 10\lg(4) - 74 + 10\lg 3 = 51{,}294$$

$$R_{тр}(S_4) = 117 - \lg(315) + 10\lg(3) - 74 + 10\lg 3 = 50{,}044$$

Таблица 5.7 – Окончательная таблица

Вел-на	Среднегеометрическая частота октавной полосы, Гц							
	63	125	250	500	1000	2000	4000	8000
μ	0,65	0,62	0,64	0,75	1	1,5	2,4	4,2
Bu	48,75	46,5	48	56,25	75	112,5	180	315
Lш	93	90	91	108	117	116	115	117
Lдоп	99	92	86	83	80	78	76	74
Rтр1	15,834	19,854	26,841	46,772	58,647	59,471	60,267	64,024
Rтр2	18,834	22,865	29,851	49,782	61,657	62,481	63,227	67,034
Rтр3	3,104	7,124	14,111	34,042	45,917	46,741	47,537	51,294
Rтр4	1,854	5,875	12,861	32,792	44,667	45,491	46,287	50,044

Вывод: звукоизоляция в звуковой кабинке соответствует всем нормам и не нуждается в доработке.

5.4 Выводы по проделанной работе

В ходе выполнения расчета производственного шума были выполнены следующие задачи:

– был определен источник шума и его шумовые характеристики;

– определены допустимые уровни звукового давления для расчетных точек;

– найдены ожидаемые уровни звукового давления в расчетных точках до проведения мероприятий по снижению шума;

– рассчитаны требуемые снижения уровней звукового давления;

– выбраны мероприятия для обеспечения требуемого снижения уровней звукового давления в расчетных точках.

Заключение

В данном дипломном проекте была создана система «умный чемодан». В ходе работы были реализованы такие задачи как:

- сбор и анализ информации по разработке мобильных приложений;
- внедрение устройств в обычный чемодан;
- создание программного кода для системы;
- разработка мобильного приложения;
- реализация связи между мобильным устройством и чемоданом;
- анализ экономической стороны реализованной системы;
- анализ условия труда, при которых был создан «умный чемодан».

После проведения экономического анализа были выяснены следующие показатели:

- себестоимость системы - 724 301,57 тенге;
- прибыль - 181 075,39 тенге;
- договорная цена - 1 014 022,19 тенге.

В ходе проверки условия труда был определен источник шума. Для устранения шума были применены мероприятия.

Список литературы

1 Ахметов И. Г. Международный научный журнал. Выпуск № 50(184) - Казань: изд-во «Молодой учёный», 2017. – 104 с.

2 SHINE! Responsible travel. Путешествия. URL: http://shine.city/triangle/5-chemodanov-kotorye-prileteli-v-2017-iz-budushhego (дата обращения 01.02.2019)

3 habr// Компания Bluesmart закрылась из-за запрета «умного багажа». URL: https://habr.com/ru/company/madrobots/blog/412191/ (дата обращения 01.02.2019)

4 Электроника в дорогу// Умная сумка-рюкзак Paqsule сама дезинфицирует вещи. URL: http://www.mobipukka.ru/2017/05/04/umnaya-sumka-ryukzak-paqsule-sama-dezinficiruet-veshhi/ (дата обращения 01.02.2019)

5 24 Gadget новейшие технологии// Самодвижущийся чемодан Travelmate. URL: https://24gadget.ru/1161066070-samodvizhuschiysya-chemodan-travelmate.html (дата обращения 01.02.2019)

6 Популярная механика// Modobag: чемодан-скутер для опаздывающих на рейс. URL: https://www.popmech.ru/gadgets/369522-modobag-chemodan-skuter-dlya-opazdyvayushchih-na-reys/ (дата обращения 01.02.2019)

7 Arduino UNO.ru/ Arduino Uno/ Общие сведения. URL: http://arduino.ru/Hardware/ArduinoBoardUno (дата обращения 03.02.2019)

8 Arduino Plus/ Android Studio. URL: https://arduinoplus.ru/android-studio/ (дата обращения 04.02.2019)

9 Орлов П.И. Основы конструирования: Справочник: В 2-х книгах. — М.: Машиностроение, 2011.

10 ГОСТ Р ИСО/МЭК 15288 – 2008. Системная инженерия – Процессы жизненного цикла систем. – 2008.

11 Портал о современных технологиях мобильной и беспроводной связи// Что такое Bluetooth и как он работает. URL: http://1234g.ru/blog-of-wireless-technologies/bluetooth/chto-takoe-bluetooth-i-kak-on-rabotaet (дата обращения 09.02.2019)

12 ArduinoMaster// Система спутниковой навигации GPS – принцип, схема, применение. URL: https://arduinomaster.ru/datchiki-arduino/sistema-sputnikovoj-navigacii-gps/#_GPS-4 (дата обращения 10.02.2019)

13 Wellsoft// Проектирование мобильного приложения. URL: https://wellsoft.pro/blog/proektirovanie-mobilnogo-prilozheniya (дата обращения 10.02.2019)

14 Петин В.А., Биняковский А.А. Практическая энциклопедия Arduino. – М.: ДМК Пресс, 2017. – 152 с.

15 Днищенко В.А. 500 схем для радиолюбителей. Дистанционное управление моделями. – Спб.: Наука и техника, 2007. – 464 с.

16 Блум Дж. Изучаем Arduino: инструменты и методы технического волшебства: Пер. с англ. – Спб.: БХВ-Петербург, 2015. – 366 с.

17 Белов А.В. Создаем устройства на микроконтроллерах. – Спб.: Наука и техника, 2007. – 304 с.

18 Бегимбетова А.С., Жандаулетова Ф.Р. Безопасность жизнедеятельности. Защита от производственного шума. Методические указания к выполнению дипломного проекта, АУЭС. – Алматы, 2010.

19 СНиП II-12-77 Защита от шума. – М., 1978.

20 Лагунов Л.Ф., Осипов Г.Л. Борьба с шумом в машиностроении. – М.: Машиностроение, 1980.

21 Долин П.А. Справочник по технике безопасности. – М.: Энергоатомиздат, 1985.

22 Под ред. Юдина Е.Я. Справочник проектировщика. Защита от шума. – М., 1974.

23 Аманбаев У.А. Экономика предприятия. – А.: «Бастау», 2012.

24 Басовский Л.Е. Экономика отрасли. – М.: ИНФРА-М, 2009.

25 Буров В.П. Бизнес-план фирмы. – М.: «Инфра-М», 2011.

27 Коноплев С.П. Экономика организаций. – М.: Проспект», 2009.

28 Куатова Д.Я. Экономика предприятия. – А.: «Экономика», 2011.

Приложение А
(обязательное)

Технические задание на разработку системы «умный чемодан»

Полное наименование системы – «умный чемодан».

Цель разработки: создать удобный и многофункциональный чемодан.

Идеология системы: разработка системы поспособствует созданию усовершенствованной версии чемодана и обеспечит его владельцу комфортный переезд/перелет. При этом, владельцу такого чемодана больше не придется переживать о его краже или потери.

Последовательность создания системы: проектирование аппаратной и программной части; программирование микроконтроллера Arduino UNO на языке C++; подключение всех необходимых устройств к микроконтроллеру; создание мобильного приложения в среде Android Studio.

Выбор и обоснование модели жизненного цикла: для разработки системы следует использовать спиральную модель. Выбор данной модели основан на быстрых и качественных результатах, повышенной конкурентоспособности, предусматривает изменения требований.

Построение общей модели разрабатываемой системы:

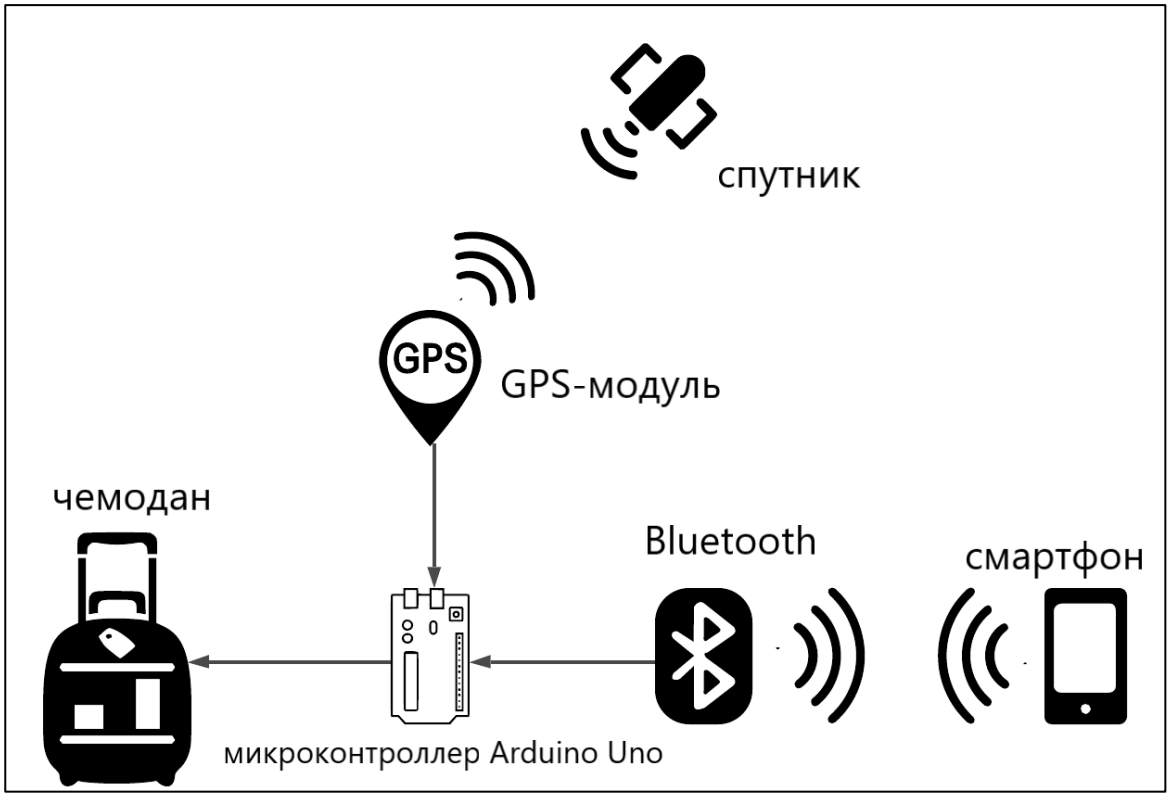


Рисунок А.1 – Общая модель системы «умный чемодан»

Продолжение приложения А

Архитектура разрабатываемой системы:

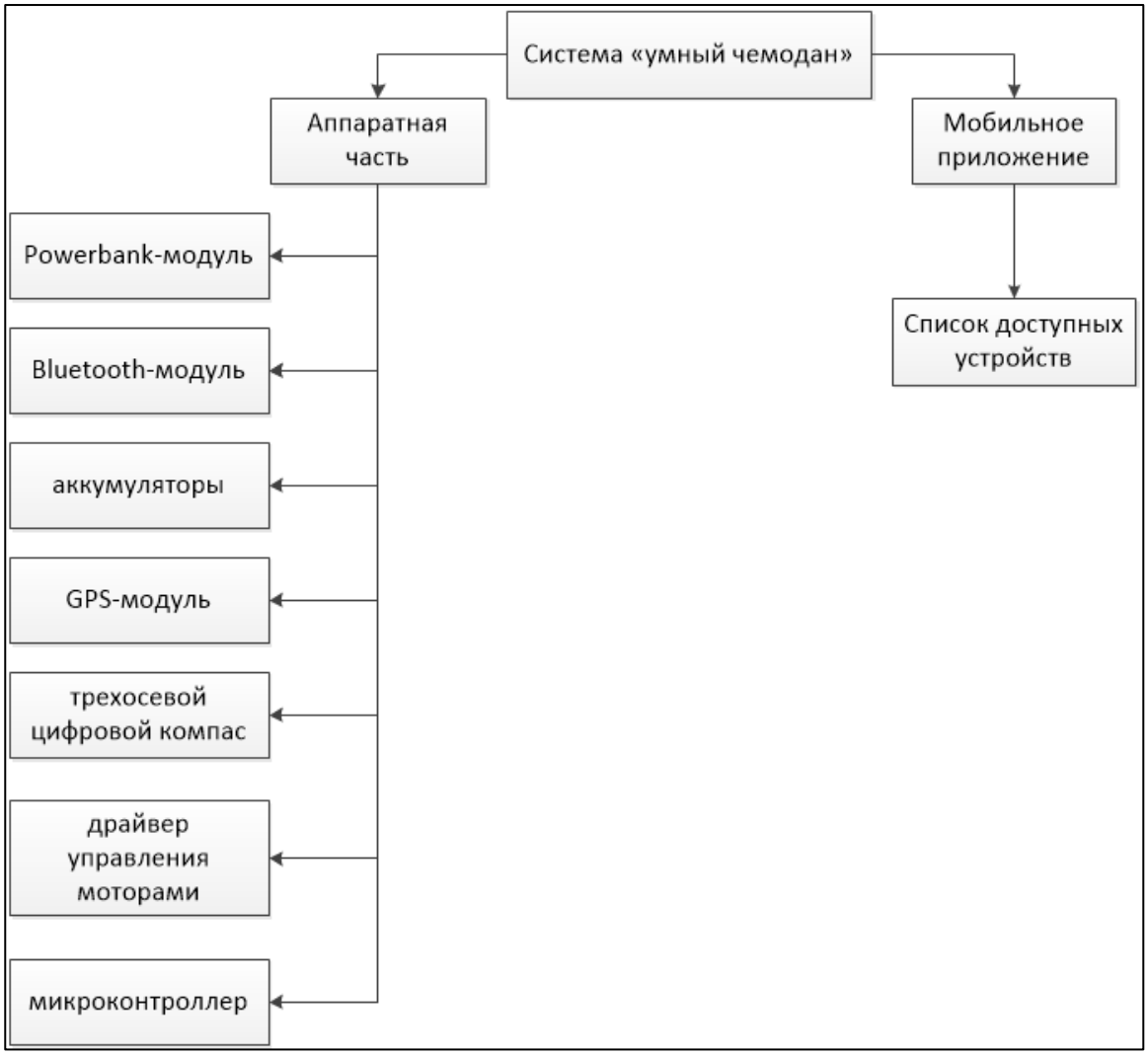


Рисунок А.2 – Архитектура системы «умный чемодан»

Выбор метода разработки: водопадный метод, при котором идет постепенная разработка системы без возвращения к пройденным этапам.

Выбор языка программирования: C++ для программирования микроконтроллера, Java для разработки мобильного приложения.

Предполагаемая аудитория: данная система рассчитана на людей, которые отправляются за пределы города/страны. Такие люди должны быть приспособлены к работе со смартфонами, так как чемодан управляется именно при помощи такого устройства.

Общий объем мобильного приложения составляет 24 МБ.

Порядок приемки: после полноценной разработки системы «умный чемодан» планируется продать её ИП «La Bella Vita» для дальнейшего расширения продаж.

Продолжение приложения А

Объем, состав текстовой и графической информации в мобильном приложении: данное приложение должно содержать возможность подключения к Bluetooth. При этом, на экране должны быть отображены устройства, к которым можно подключиться.

Семантика мобильного приложения: приложение следует обеспечить понятным простым в использовании графическим интерфейсом. Для подключения к чемодану необходимо минимальное количество кнопок и результаты подключения должны быть отражены.

Общая блок-схема мобильного приложения представлена на рисунке А.3.

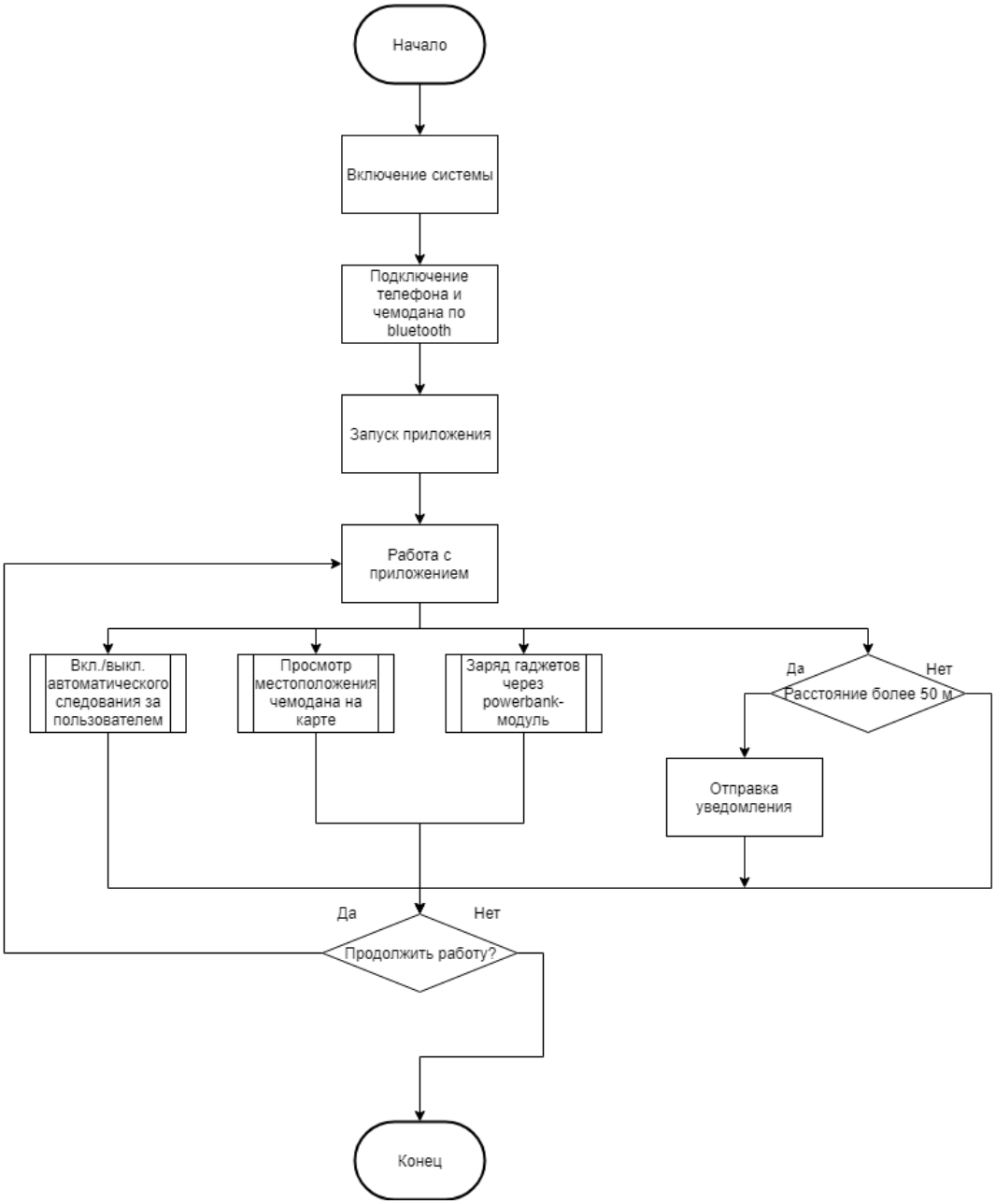


Рисунок А.3 – Блок-схема алгоритма работы мобильного приложения

Продолжение приложения А

Основной диапазон разрешения экранов, на которых будет просматриваться мобильное приложение: HD (1280 x 720), Full HD (1920 x 1080), Quad HD (2560 x 1440).

Операционные системы, на которых работает приложение: OS Android.

Обеспечение защищенности программного части системы: так как приложение мобильное, то вероятность его заражения вирусами очень мала, поскольку мобильные устройства на базе Android имеют хорошую защиту.

Разработка инструкции помощи пользования: для того, чтобы объяснить пользователю способы взаимодействия с системой, необходимо создать подробную инструкцию, содержащую всю необходимую информацию по работе с чемоданом и приложением.

Тестирование и отладка программного продукта: тестирование и отладка мобильного приложения должно происходить в среде разработки Android Studio при создании приложения.

Специфические требования к системе:

Адаптивность программного продукта: мобильное приложение должно иметь функцию приспособления к различным экранам телефонов на базе операционной системы Android. В приспособлении приложения к другим устройствам нет необходимости.

Интеллектуальное развитие: мобильное приложение должно быть функционально законченным и тогда ему не потребуются обновления.

Психологические особенности:

Дизайн: при разработке графического интерфейса приложения не стоит уделять большее внимание дизайну, стоит делать упор на функционал. Создавая сам чемодан стоит учесть его стиль для привлечения покупателей.

Расположение элементов интерфейса: при разработке мобильного приложения следует располагать различные элементы интерфейса так, чтобы их назначение было интуитивно понятно.

Экономическая эффективность системы: так как такая система является инновационной на территории Республики Казахстан, то вероятность ее окупаемости довольно высока. Также, при использовании наиболее бюджетных и качественных устройств, система будет иметь максимально приемлемую цену.

Приложение Б
(обязательное)

Листинг программы

```
// Imports
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_HMC5883_U.h>
#include <Servo.h>
#include <SoftwareSerial.h>
#include <TinyGPS++.h>
#include "./Definitions.h"

// GPS
TinyGPSPlus gps;
#define GPS_BAUD 4800

// Motors
Servo servoThrottleRight;
Servo servoThrottleLeft;

bool enabled = true;

// Bluetooth input
bool started = false;
bool ended = false;
char inData[80]; // creates an 80 character array called "inData"
byte index; //creates a variable type=byte called "index"
double Long; //variable for longitude coordinate
double Lat; //variable for latitude coordinate

SoftwareSerial bluetoothSerial(BLUETOOTH_TX_PIN, BLUETOOTH_RX_PIN);
SoftwareSerial gpsSerial(GPS_RX_PIN, GPS_TX_PIN); // TXD to digital pin 6

Adafruit_HMC5883_Unified mag = Adafruit_HMC5883_Unified(12345);

GeoLoc checkGPS() {
  Serial.println("Reading onboard GPS: ");
  unsigned long start = millis();
  while (millis() - start < GPS_UPDATE_INTERVAL) {
    if (feedgps())
      return gpsdump();
```

```
  }
```

Продолжение приложения Б

```
  GeoLoc robotLoc;
  robotLoc.lat = 0.0;
  robotLoc.lon = 0.0;

  return robotLoc;
}

GeoLoc gpsdump() {
  GeoLoc robotLoc;
  robotLoc.lat = gps.location.lat();
  robotLoc.lon = gps.location.lng();

  Serial.print(robotLoc.lat, 7); Serial.print(", "); Serial.println(robotLoc.lon, 7);
  return robotLoc;
}

bool feedgps() {
  while (gpsSerial.available() > 0) {
    if (gps.encode(gpsSerial.read()))
      return true;
  }
  return false;
}

void displayCompassDetails(void)
{
  sensor_t sensor;
  mag.getSensor(&sensor);
  Serial.println("------------------------------------");
  Serial.print("Sensor:       "); Serial.println(sensor.name);
  Serial.print("Driver Ver:   "); Serial.println(sensor.version);
  Serial.print("Unique ID:    "); Serial.println(sensor.sensor_id);
  Serial.print("Max Value:    "); Serial.print(sensor.max_value); Serial.println("
uT");

  Serial.print  ("Min Value:    "); Serial.print(sensor.min_value); Serial.println("
uT");
  Serial.print  ("Resolution:   "); Serial.print(sensor.resolution); Serial.println(" uT");
  Serial.println("------------------------------------");
  Serial.println("");
  delay(500);
```

```
}
```

Продолжение приложения Б

```
#ifndef DEGTORAD
#define DEGTORAD 0.0174532925199432957f
#define RADTODEG 57.2957795130823208767f
#endif

float geoBearing(struct GeoLoc &a, struct GeoLoc &b) {
  float y = sin(b.lon-a.lon) * cos(b.lat);
  float x = cos(a.lat)*sin(b.lat) - sin(a.lat)*cos(b.lat)*cos(b.lon-a.lon);
  return atan2(y, x) * RADTODEG;
}

float geoDistance(struct GeoLoc &a, struct GeoLoc &b) {
  const float R = 6371000; // radius of earth in metres
  float p1 = a.lat * DEGTORAD;
  float p2 = b.lat * DEGTORAD;
  float dp = (b.lat-a.lat) * DEGTORAD;
  float dl = (b.lon-a.lon) * DEGTORAD;

  float x = sin(dp/2) * sin(dp/2) + cos(p1) * cos(p2) * sin(dl/2) * sin(dl/2);
  float y = 2 * atan2(sqrt(x), sqrt(1-x));

  // returns distance in meters
  return R * y;
}

float geoHeading() {
  /* Get a new sensor event */
  sensors_event_t event;
  mag.getEvent(&event);

  // Hold the module so that Z is pointing 'up' and you can measure the heading with
x&y
  // Calculate heading when the magnetometer is level, then correct for signs of axis.
  float heading = atan2(event.magnetic.y, event.magnetic.x);

  // Offset
  heading -= DECLINATION_ANGLE;
  heading -= COMPASS_OFFSET;

  // Correct for when signs are reversed.
  if(heading < 0)
```

```
  heading += 2*PI;
```



```
 // Check for wrap due to addition of declination.
 if(heading > 2*PI)
  heading -= 2*PI;

 // Convert radians to degrees for readability.
 float headingDegrees = heading * 180/M_PI;

 // Map to -180 - 180
 while (headingDegrees < -180) headingDegrees += 360;
 while (headingDegrees >  180) headingDegrees -= 360;

 return headingDegrees;
}

void setSpeedMotorA(int speed) {
 servoThrottleRight.writeMicroseconds(speed + MOTOR_A_OFFSET);
}

void setSpeedMotorB(int speed) {
 servoThrottleLeft.writeMicroseconds(speed + MOTOR_B_OFFSET);
}

void stop() {
 // stop the motors
 servoThrottleRight.writeMicroseconds(RC_NEUTRAL);
 servoThrottleLeft.writeMicroseconds(RC_NEUTRAL);
}

void drive(int distance, float turn) {
 int fullSpeed = 230;
 int stopSpeed = 0;

 int s = fullSpeed;

 if ( distance < 8 ) {
  int wouldBeSpeed = s - stopSpeed;
  wouldBeSpeed *= distance / 8.0f;
  s = stopSpeed + wouldBeSpeed;
 }

 int autoThrottle = constrain(s, stopSpeed, fullSpeed);
```

```
  autoThrottle = 230;
```

Продолжение приложения Б

```
  float t = turn;
  while (t < -180) t += 360;
  while (t >  180) t -= 360;

  Serial.print("turn: ");
  Serial.println(t);
  Serial.print("original: ");
  Serial.println(turn);

  float t_modifier = (180.0 - abs(t)) / 180.0;
  float autoSteerA = 1;
  float autoSteerB = 1;

  if (t < 0) {
    autoSteerB = t_modifier;
  } else if (t > 0){
    autoSteerA = t_modifier;
  }

  Serial.print("steerA: "); Serial.println(autoSteerA);
  Serial.print("steerB: "); Serial.println(autoSteerB);
//  int speedA = (int) (((float) map(autoThrottle, stopSpeed, fullSpeed,
RC_NEUTRAL, RC_MAX)) * autoSteerA);
//  int speedB = (int) (((float) map(autoThrottle, stopSpeed, fullSpeed,
RC_NEUTRAL, RC_MIN)) * autoSteerB);
  int speedA = (int) (((float) map(autoThrottle, stopSpeed, fullSpeed,
RC_NEUTRAL, RC_MIN)) * autoSteerA);
  int speedB = (int) (((float) map(autoThrottle, stopSpeed, fullSpeed,
RC_NEUTRAL, RC_MAX)) * autoSteerB);

  setSpeedMotorA(speedA);
  setSpeedMotorB(speedB);
}

void driveTo(struct GeoLoc &loc, int timeout) {
  gpsSerial.listen();
  GeoLoc robotLoc = checkGPS();
  bluetoothSerial.listen();

  if (robotLoc.lat != 0 && robotLoc.lon != 0 && enabled) {
    float distance = 0;
```

```
   //Start move loop here
```

Продолжение приложения Б

```
   do {
     gpsSerial.listen();
     robotLoc = checkGPS();
     bluetoothSerial.listen();

     distance = geoDistance(robotLoc, loc);
     float bearing = geoBearing(robotLoc, loc) - geoHeading();

     Serial.print("Distance: ");
     Serial.println(distance);

     Serial.print("Bearing: ");
     Serial.println(geoBearing(robotLoc, loc));

     Serial.print("Heading: ");
     Serial.println(geoHeading());

     drive(distance, bearing);
     timeout -= 1;
   } while (distance > 1.0 && enabled && timeout>0);

   stop();
  }
}

void setupCompass() {
   /* Initialise the compass */
  if(!mag.begin())
  {
    /* There was a problem detecting the HMC5883 ... check your connections */
    Serial.println("Ooops, no HMC5883 detected ... Check your wiring!");
    while(1);
  }

  displayCompassDetails();
}

void setup()
{
  // Attaching motors
  servoThrottleRight.attach(PIN_SPEED_RIGHT);
```

```
  servoThrottleLeft.attach(PIN_SPEED_LEFT);
```

Продолжение приложения Б

```
  //Debugging via serial
  Serial.begin(115200);

  // Compass
  setupCompass();

  //GPS
  gpsSerial.begin(GPS_BAUD);

  //Bluetooth
  bluetoothSerial.begin(9600);
}

// Testing
void testDriveNorth() {
  float heading = geoHeading();
  int testDist = 5;
  Serial.println(heading);

  while(!(heading < 5 && heading > -5)) {
    drive(testDist, heading);
    heading = geoHeading();
    Serial.println(heading);
    delay(500);
  }

  stop();
}


void loop()
{
  while (bluetoothSerial.available() > 0)
  {
    char inChar = ((byte)bluetoothSerial.read());
    if (inChar == SOP)
    {
      index = 0;
      inData[index] = '\0';
      started = true;
      ended = false;
```

```
  }
```

Продолжение приложения Б

```
  else if (inChar == EOP)
  {
   ended = true;
   break;
  }
  else
  {
   if (index < 79)
   {
    inData[index] = inChar;
    index++;
    inData[index] = '\0';
   }
  }
}
if (started && ended)
{
  char *token = strtok(inData, ",");
  boolean first = true;
  while (token)
  {
   double val = atof(token);
   token = strtok(NULL, ",");
   if (first){
    Lat = val;
   }else{
    Long = val;
   }
   first = false;
  }

  started = false;
  ended = false;
  index = 0;
  inData[index] = '\0';
  GeoLoc phoneLoc;
  phoneLoc.lat = Lat;
  phoneLoc.lon = Long;
  Serial.print(phoneLoc.lat, 7); Serial.print(", "); Serial.println(phoneLoc.lon, 7);
  driveTo(phoneLoc, GPS_WAYPOINT_TIMEOUT);
}
```

}

Приложение В
(обязательное)

Акт внедрения системы «умный чемодан»

LA BELLA VITA

Исх. № 083 «6» мая 2019 г.

Акт внедрения системы «Умный чемодан»

Настоящий Акт свидетельствует, что система «умный чемодан», разработанная Усербаевым Асетом, внедрено в ИП «La Bella Vita».

Процесс внедрения проходил с 1 по 4 мая 2019 г.

Заявленные характеристики системы предполагали наличие следующих основных возможностей:

- Управление системой при помощи мобильного приложения;
- Автоматического следование за пользователем;
- Определение местоположения «умного чемодана»;
- Встроенное зарядное устройство для гаджетов.

В ходе опытной эксплуатации системы подтверждено, что оно обладает всеми заявленными возможностями.

На момент подписания настоящего Акта система находится в завершающей стадии разработки.

Директор ИП «La Bella Vita» Раточка А.Л.

La Bella Vita

050007, Республика Казахстан, г. Алматы, Ахмедсафина, 58, Тел./факс: 2502 693, 2507 402
58 Ahmedsafina str., Almaty, 050007, Republic of Kazakhstan, Tel./fax: 2502 693, 2507 402