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

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

РАБОТА ПРОВЕРЕНА
Рецензент
_____
«___»_____ 2019 г.

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

Разработка системы мониторинга транспорта с применением технологии RFID

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

Руководитель работы,
к.т.н., доцент каф. ЭВМ
_____И.Л.Кафтанников
«___»_____2019 г.

Автор работы,
студент группы КЭ-452
_____ Н.С.Алдохин
«___»_____2019 г.

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

Челябинск-2019

Аннотация

Н.С. Алдохин. Разработка системы мониторинга транспорта с применением технологии RFID. – Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ВШЭКН; 2019, 88 с., 45 ил., библиогр. список – 28 наим.

В рамках выпускной квалификационной работы производится разработка системы мониторинга транспорта, основанной на технологии радиочастотной идентификации. Осуществляется анализ текущей транспортной ситуации в городе, а также методов отслеживания подвижного состава общественного транспорта и идентификации объектов. Рассматриваются преимущества и недостатки систем спутниковой навигации и радиочастотной идентификации. Выполняется проектирование оптимальной для городских условий системы идентификации.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Мониторинг общественного транспорта в городе со временем становится необходимостью, так как всё чаще жилые районы отделяются от мест работы и центров досуга и развлечений, и появляется потребность в передвижении на значительные расстояния по достаточно строгому расписанию. Подвижной состав города-миллионника, например, Челябинска насчитывает около 500 единиц троллейбусов и автобусов, а также более 2000 маршрутных такси (микроавтобусов). Подобные цифры влекут за собой значительные расходы на отслеживающее оборудование. А при его установке непосредственно на машины эти расходы пропорциональны росту их количества.

RFID (Radio Frequency IDentification – радиочастотная идентификация) – это способ идентификации объектов, при котором чтение и запись данных с меток производится посредством радиосигналов. Такой способ учета позволяет обрабатывать большое множество объектов при отсутствии сложного электронного оборудования на них самих. При этом пропускная способность многих считывателей достигает тысяч меток в минуту. Метки, в свою очередь, являются альтернативой QR или штрихкодам, но при этом не требуют непосредственной видимости при считывании. Это означает, что водителям транспорта не обязательно проезжать под какой-либо камерой или считывающим устройством. Им достаточно быть в зоне считывания, на остановке.

Системы идентификации объектов широко используются в разных областях, например, в складской логистике, системах контроля доступа. Но для мониторинга движения транспорта в масштабах города они будут применяться впервые.

Цель исследования – создание программно-аппаратного комплекса учета движения городского транспорта на основе технологии RFID,

обеспечивающего доступ к информации о его местоположении в реальном времени.

Для достижения поставленной цели необходимо решить следующий ряд задач:

1. Изучить параметры радиоинтерфейсов.
2. Провести анализ существующих решений в области идентификации объектов, в том числе их местоположения.
3. Разработать структуру будущего программно-аппаратного комплекса.
4. Создать макет аппаратной части, взаимодействующей с программной частью.
5. Оценить работоспособность полученной системы.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 ОБЩЕСТВЕННЫЙ ТРАНСПОРТ В СОВРЕМЕННОМ ГОРОДЕ

Продолжительный процесс эволюции городов тесно связан с развитием транспорта с момента появления первых самоходных машин [1]. Итогом этого развития стало обширное децентрализованное городское пространство с доступным широкому слою населения маршрутным общественным транспортом.

Общественный транспорт позволяет предоставлять населению услуги перевозок. Это особенно важно в городе, так как по различным историческим, географическим, экономическим критериям существует малое количество возможностей компактно в одном месте организовать проживание, рабочую и иную деятельность человека, что, в свою очередь, связано с многоузловой организацией городских структур.

Грамотно организованный общественный транспорт является основой эффективной жизнедеятельности граждан, обеспечивает экономическую стабильность и продуктивность города в целом. Также недвижимость в районе обслуживания городского маршрутного транспорта имеет более высокую стоимость и общественное значение. Данный вид транспорта способствует притоку новых клиентов, удобному предоставлению услуг населению, развитию социальных отношений, создает ощущение комфортного проживания в городе [2]. Благодаря использованию общественного транспорта некоторые группы населения (например, пожилые люди) ввиду отсутствия возможностей управления личным автомобилем тоже могут участвовать в социальной жизни города.

Город предполагает наличие нескольких видов транспорта: автобусы, троллейбусы, трамваи, маршрутные такси (микроавтобусы). В крупных российских городах, например, в Челябинске доли муниципального транспорта (в большей степени это автобусы и другой крупный пассажирский транспорт) и компаний перевозчиков (микроавтобусы) в области оказания

услуг пассажирских перевозок часто разделены по 50% [3, 4]. Иногда наблюдается преобладание маршрутных такси по причине слабо подготовленной инфраструктуры именно для автобусов.

Таким образом, можно сказать, что общественный транспорт является неотъемлемой частью города, и его популярность, а значит и остальные аспекты эффективного существования города напрямую зависят от удобства использования. При этом удобством считается не только сам процесс достижения комфортной поездки, а также работа и с другими мелкими аспектами. Например, современные информационные технологии позволяют пассажирам отслеживать местоположение транспорта и детальнее планировать свою деятельность. Мониторинг только начинает становиться доступным широкому кругу потребителей, следовательно, предполагает рассмотрение и изучение различных технологий отслеживания.

1.2 ОБЗОР АНАЛОГОВ

1.2.1 Системы на основе спутниковой навигации

Мониторинг на основе ГЛОНАСС/GPS к 2019 году обустроен в основном на таких видах общественного транспорта, как троллейбус, автобус, трамвай.

Целью создания спутниковой навигации является точное определение местоположения объектов в заданной системе координат. Осуществляется это при помощи автоматического обмена и обработки навигационных параметров между спутником и отслеживаемым объектом.

Благодаря совместному использованию спутниковой навигации и мобильной связи появляется возможность отслеживать положение транспорта в режиме реального времени.

На данный момент наиболее популярными являются две системы навигации: GPS (Global Positioning System – система глобального позиционирования) и ГЛОНАСС (глобальная навигационная спутниковая

система). Обе системы изначально разрабатывались для военных целей, однако, в современное время предоставляют сервис по определению местоположения и скоростных характеристик для гражданского сектора в любой точке Земли.

Спутниковая навигация подразделяется на три сегмента: космический, наземный, пользовательский. Первый состоит из навигационных спутников, основной функцией которых является формирование и излучение радиосигналов, необходимых для навигационных определений потребителей, контроль бортовых систем спутника. Второй состоит из космодрома, наземного центра управления, командно-измерительного комплекса. Все эти элементы в совокупности обеспечивают вывод спутников на орбиту, их обслуживание, корректировку множества параметров. Обе системы содержат по 24 спутника, вращающихся на круговых орбитах Земли. ГЛОНАСС на высоте около 19100 км, а GPS на высоте примерно 20100 км [5]. Пользовательский сегмент состоит из аппаратуры потребителей. Устройства выполняют задачи приема сигналов от спутников, измерения навигационных параметров и обработки измерений.

Основой работы системы являются дальномерные измерения между навигационными спутниками и потребителем. То есть потребитель получает координаты спутников, при этом одновременно выполняется измерение дальностей до них, основанное на временных задержках принимаемого сигнала от спутника по сравнению с сигналом, генерируемым аппаратурой потребителя. Дополнительно вводится в решение задачи смещение часов потребителя относительно спутника. В итоге, с учетом трёх координат объекта и времени в процессе определения местоположения решается система уравнений с четырьмя неизвестными. Отсюда следует, что устройство потребителя должно иметь связь как минимум с четырьмя спутниками.

Потребительский сегмент представлен множеством устройств. Некоторые из них накапливают данные во встроенной памяти, а затем, при

подключении к компьютеру, передают историю поездок в его память. В результате, можно выявить маршрут по итогу рейса (множества рейсов) и нет никакой возможности просматривать данные в режиме онлайн. Другая группа устройств оснащается модемами мобильной связи, благодаря которым становится реальным отслеживание местоположения онлайн. В связи с повсеместным распространением сетей мобильного интернета, они практически вытеснили первую группу устройств, которые теперь чаще применяются в грузовых перевозках на дальние расстояния. Примером трекера может служить STARLINE M18, стоимость которого составляет 3800 руб [6], в дополнение к этому к этому оплата тарифов мобильного трафика порядка 200-300 руб ежемесячно.

Таким образом, рынок трекеров спутниковой навигации в настоящее время является довольно устоявшимся и предлагает достаточно стабильные в работе решения. Зачастую его функционал ресурсозатратный, когда речь заходит о мониторинге большого количества транспортных единиц. Стоимость ввода в эксплуатацию и дальнейшее обслуживание растет пропорционально количеству машин в автопарке. К тому же, сложные навигационные устройства могут выходить из строя, некоторые из них требуют вмешательства специалиста в систему питания транспортной единицы, могут быть неправильно настроены, вследствие чего данные искажаются либо не предоставляются вовсе.

1.3 АНАЛИЗ ОСНОВНЫХ ТЕХНОЛОГИЧЕСКИХ РЕШЕНИЙ

1.3.1 Обзор технологии радиочастотной идентификации

Технология RFID (англ. Radio Frequency IDentification, радиочастотная идентификация) – бесконтактной идентификации объектов, в котором посредством радиосигналов считываются или записываются данные, хранящиеся в RFID-метках, или по-другому – транспондерах [7].

Стандартизацией RFID систем занимается организация EPC Global [8]. Основным регулирующим документом в настоящее время является международный стандарт ISO 18000-6[9], дополняющий другой документ, ISO 18000.

RFID система состоит считывающего устройства, которое чаще называют считыватель или ридер (англ. reader – считыватель), метки, называемой транспондером (англ. transponder – передатчик-ответчик), а также системы сбора информации [10]. На рисунке 1.3.1.1 представлена приблизительная схема этих компонентов.

Рисунок 1.3.1.1 – Компоненты RFID-системы

Считывающее устройство включает в себя приёмник и передатчик (обычно это единое устройство), а также антенну. Транспондер – приёмопередающее устройство, которое отвечает заранее заданным сигналом на определённый принятый сигнал. Метка состоит из антенны для приёма и передачи информации и чипа, хранящего и обрабатывающего информацию. Некоторые виды меток предусматривают встроенный блок питания. В роли системы сбора информации может выступать рабочая станция при локальном соединении компонентов, или веб-сервер, когда необходима отправка данных от удалённых считывателей.

1.3.1.1 RFID-метки

Возможности системы в области идентификации объектов практически не ограничены. По своей сути RFID-метки напоминают штрихкоды, но для их считывания и прочих манипуляций нет необходимости иметь визуальный доступ к ним. Такую метку можно установить где угодно на объекте, который будет идентифицирован в некоторой области пространства. RFID-метки защищены от изменений освещённости, положения объекта в области считывания, а также при определённом исполнении и от неблагоприятных погодный условий.

Большинство RFID-меток состоит из двух частей – антенны и интегральной схемы. Благодаря первой происходит приём и передача сигнала, а также осуществляется питание интегральной схемы метки. А вторая необходима для хранения и обработки информации, модулирования и демодулирования радиочастотного сигнала и .

На рисунке 1.3.1.1.1 представлена метка с указанными компонентами.

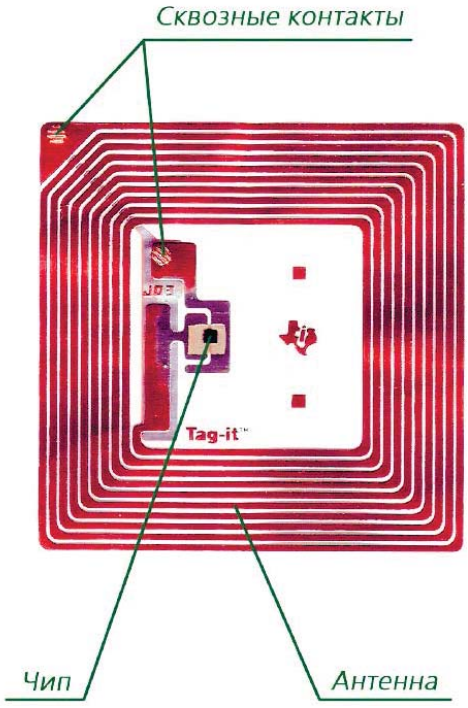


Рисунок 1.3.1.1.1 – Основные компоненты пассивной RFID-метки

По источнику питания метки бывают:

- пассивные RFID-метки. Не имеют встроенного источника энергии. Благодаря радиоволнам в антенне возникает электрический ток, который служит источником питания для чипа, а также приёма и передачи информации;
- активные RFID-метки. Энергии от считывателя не получают, потому что имеют встроенный источник питания. Такие метки могут считываться на гораздо большем расстоянии, так как генерируют выходной сигнал, а не отражают как пассивные. Они могут быть оснащены различными дополнительными датчиками;
- Полупассивные RFID-метки. Очень похожи на пассивные метки, однако, встроенное питание в них используется только для обеспечения функционирования чипа, а радиосигнал отражается, а не генерируется.

Также метки имеют различные виды встроенной в них памяти:

- только чтение (англ. Read Only) – такая память предусматривает запись информации при изготовлении метки без каких-либо дальнейших манипуляций. Метки с этим типом памяти пригодны для простого идентифицирования объектов. Недостаток в том, что она оказывается привязанной к этому объекту. Вариант решения проблемы - создание абстрактных меток с идентификаторами, и переназначение их в случае, когда понадобится установить метку на другой объект;
- однократная запись и многократное чтение (англ. Write Once Read Many) – кроме уникального идентификатора такие метки содержат блок однократно запис шем можно многократно читать;
- многократно перезаписываемая (англ. Read and Write) – такие метки содержат идентификатор и блок памяти для чтения (записи) информации. Данные в них могут быть перезаписаны многократно.

По раб частоте метки бывают:

- низкочастотные (125—134 кГц);
- средней частоты (13,56 МГц);
- высокочастотные (860—960 МГц).

Первые используются для введения меток людям, животным, рыбам. Но длина волны не позволяет считывать эти метки на большом расстоянии, а также появляется большое количество коллизий при чтении множества меток в ограниченном пространстве. Системы на частоте 13,56 МГц наиболее стандартизованы, имеют малую стоимость, широко применяются. Например, в системах идентификации личности, контроля доступа, платежах, обмене информацией между мобильными устройствами. Но расстояние чтения не позволяет их использовать на дистанциях более 1,5 метров, а также не решены проблемы коллизий. Системы высокочастотного диапазона могут взаимодействовать с объектами на расстояниях до 25 метров для пассивных меток, а также они оснащены антиколлизионными алгоритмами, что позволяет считывать большое количество меток на широкой площади.

Согласно стандартам EPC Global пассивные метки относят к классам 0 и 1. Метки типа класса 0 оснащаются не перезаписываемым уникальным идентификатором EPC (Electronic Product Code) при производстве, который в перспективе может быть только прочитан. Метки типа класса 1 могут получить этот идентификатор от пользователя единственный раз. Затем доступ к памяти метки будет так же только на чтение. На рисунке 1.3.1.2 можно увидеть схему размещения данных в памяти.

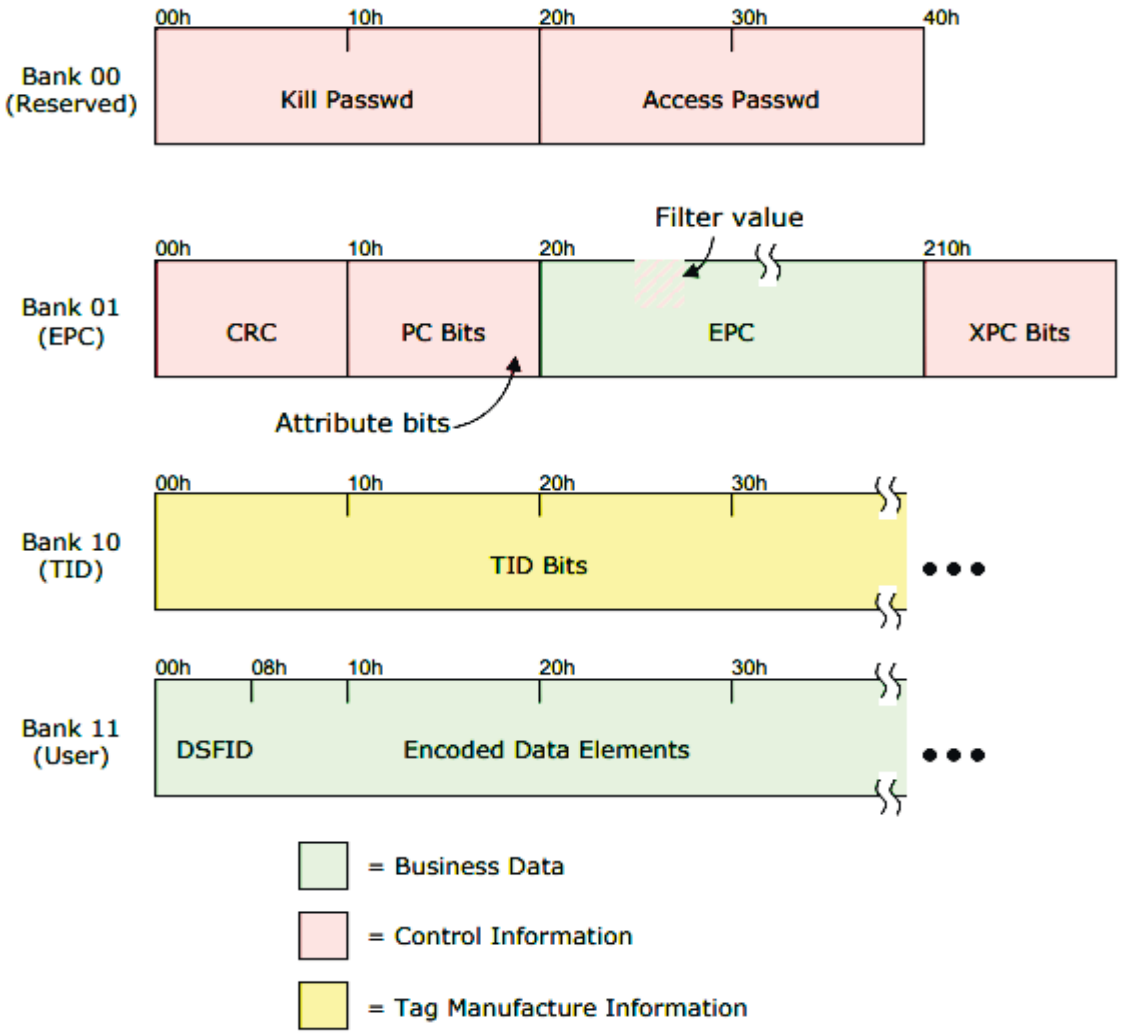


Рисунок 1.3.1.2 – схема данных в памяти метки

Условно она разделяется на 4 подраздела [11]:

1. Зарезервированный:

   1.1. Пароль на уничтожение метки (32 бита). При ненулевом значении этого поля, зная его, можно послать команду ликвидации метки, после чего та полностью перестанет функционировать.

   1.2. Пароль на доступ к метке (32 бита). Если это поле установлено, то доступ к ней будет возможен только при знании этого пароля.

2. EPC (англ. Electronic Product Code – электронный код продукта):

   2.1. CRC (англ. Cyclic Redundancy Check – циклическая проверка избыточности) – 16-разрядное значение, вычисленное на

основе содержимого памяти метки. Позволяет проводить проверку целостности данных

2.2. PC bits (англ. Protocol Control bits – биты управления протоколом). 16 бит под различные атрибуты протокола.

2.3. EPC. Глобальный уникальный идентификатор физического объекта размером 496 бит.

2.4. XPC bits (англ. Extended Protocol Control bits – биты расширенного управления протоколом). 16 бит для расширенных настроек. Доступны, если установлен бит 16h в EPC.

3. TID bits (англ. Transponder ID – идентификатор метки).

4. Encoded Data Elements (Зашифрованный раздел данных). Этот раздел может отсутствовать. Если есть, то обычно занимает от 32 до 512 бит.

1.3.1.2 RFID-считыватели

Считыватели (ридеры) – это приборы, которые читают информацию с меток и записывают в них данные. Эти устройства могут быть постоянно подключенными к учетной системе, или работать автономно.

Виды считывателей:

- стационарные. Устанавливаются на любые, зачастую неподвижные объекты: стены, столбы, иные конструкции. Они могут быть выполнены в виде замка, вмонтированы в стол или закреплены рядом с конвейером . По сравнению с переносными, считыватели такого типа обычно обладают большей зоной чтения и мощностью, как правило от 1 до 100 метров, реже – около 300 метров, и способны одновременно обрабатывать данные с нескольких десятков меток. Стационарные считыватели подключаются к программируемым логическим контроллерам или к ПК. Задача таких считывателей — поэтапно фиксировать перемещение маркированных

объектов в реальном времени, либо идентифицировать положение отмеченных меткой предметов в пространстве;

- мобильные. Обладают сравнительно меньшей дальностью действия и зачастую не имеют постоянной связи с программой контроля и учета. Мобильные считыватели имеют внутреннюю память, в которую записываются данные с прочитанных меток (потом эту информацию можно загрузить в компьютер) и, как и стационарные считыватели, способны записывать данные в метку (например, информацию о произведённом контроле). В зависимости от частотного диапазона метки, дистанция устойчивого считывания и записи данных в них будет различна. Но как правило это не более 10 метров.

Помимо приведенной классификации есть разные реализации считывателей: одноплатный модуль, взаимодействующий с микроконтроллером или другим управляющим устройством с помощью интерфейсов SPI или UART, а также готовое устройство с предустановленной операционной системой (например, Windows CE или Linux), с которым можно взаимодействовать по RS232 или RS485, а также они могут иметь порты USB и UTP Ethernet.

В качестве одного из примеров считывателя, реализованного на одной плате можно рассмотреть "SparkFun Simultaneous RFID Reader-M6E Nano". Он основан на модуле "M6E-Nano" от производителя ThingMagic[12].

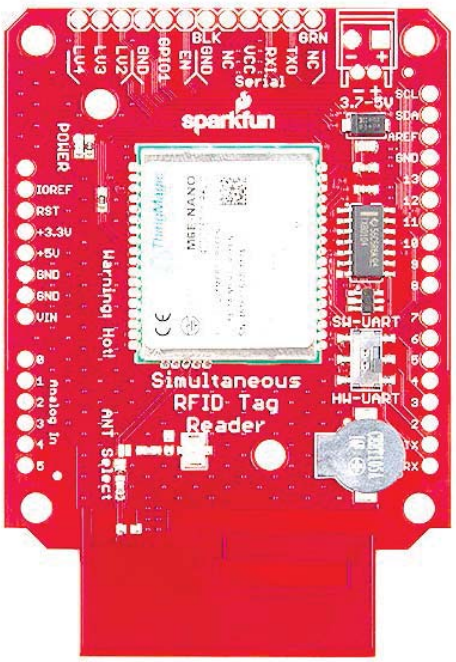


Рисунок 1.3.1.2.1 – плата "SparkFun M6E Nano"

Данный модуль позволяет читать несколько меток одновременно, а также работать с внешней антенной. Обмен данными с микроконтроллером происходит посредством UART интерфейса. Настройку мощности и производительности под заданные потребности можно производить с помощью программного обеспечения "Universal Reader Assistant". Эту же программу можно использовать для считывания информации. После настройки можно перейти к использованию с микроконтроллером. Плата создана специально в виде расширяемого модуля для «Arduino UNO». Производитель также предоставляет библиотеку, позволяющую работать с микроконтроллером в "Arduino IDE". Функционал библиотеки аналогично программе "Universal Reader Assistant" позволяет настроить считыватель.

Плата считывает метки "EPCglobal Gen 2" (соответствующие диапазону 860-960 МГц) со скоростью до 150 штук в секунду. Плата имеет регулируемую выходную мощность от 0 дБм до +27 дБм. Есть возможность считывать метки на расстоянии до 5 метров при подключении внешней антенны. Встроенная позволяет производить считывание на расстоянии до 60 см.

К плюсам данного считывателя относится его исполнение на одной плате, которая может подключаться как расширяемый модуль к Arduino UNO, либо просто использоваться с любым другим микроконтроллером, а также предоставляемая библиотека. Минусом устройства является достаточно малая дальность считывания – 5 метров. Это вплотную подходит для использования на остановках в городе, а иногда может быть недостаточно. Его стоимость составляет 14560 рублей, и её нельзя отнести к плюсам или минусам, потому что это средний показатель для такого рода устройств.

В качестве готового к использованию без дополнительных действий, не касающихся настройки непосредственно на месте установки, рассматривается считыватель "Motorola FX7400", представленный на рисунке 1.3.1.2.2.

Рисунок 1.3.1.2.2 – считыватель "Motorola FX7400"

Данное устройство обладает 4 портами для подключения антенн, выводом для GPIO (англ. general-purpose input/output – порты ввода/вывода), USB type-B, UTP Ethernet. На нём предустановлена операционная система Windows CE 5.0, а также есть доступ к различным настройкам по веб интерфейсу с другого устройства[13]. Можно отметить наличие API (англ. application programming interface – программный интерфейс приложения), поддерживающего языки Java и C (платформу .NET). Благодаря нему

происходит обращение к настройкам и функциям считывателя из других приложений.

Считыватель поддерживает стандарт ISO 18000-6C (EPC Class 1 Gen 2, высокочастотные метки), его выходная мощность от +15 до +30 дБм.

Плюсами данного устройства являются: его полноценная готовность к использованию по назначению, а также различного рода сопутствующее ПО для управления и наличие API. К минусам можно отнести избыточный функционал для изучаемой области мониторинга транспорта на остановках и соединение с сетью Интернет только через UTP Ethernet, а так же цену порядка 38000 рублей, которая обуславливается богатым функционалом.

1.3.1.3 RFID антенны

Иногда в считывателях уже установлена антенная, однако её дальность считывания не позволяет работать в масштабах остановки. Если идёт речь о считывании меток на расстоянии в десятки метров, то важным элементом аппаратного комплекса является антенна. На рисунке 1.3.1.3.1 представлен пример такой антенны.

Рисунок 1.3.1.3.1 – RFID антенна диапазона UHF CLOU CL7205A

Основными параметрами антенны являются[14]:

- ширина луча;
- направленность;
- тип поляризации.

Ширина луча определяется по диаграмме направленности как угол между двумя точками в одной плоскости, где излучение падает примерно на 3 дБ ниже максимальной точки, в пределах которого излучается наибольшая часть энергии сигнала. Его также можно рассматривать как пиковую эффективную величину излучаемой мощности основного направления волны. Существует две ширины луча – азимут (горизонтальный) и угол возвышения (вертикальный). Но чаще всего ширина луча рассматривается как горизонтальный угол на диаграмме направленности. Как правило, ширина луча ультравысокочастотных антенн колеблется в диапазоне от 60 до 80 градусов. На рисунке 1.3.1.3.2 представлен пример диаграммы направленности с подписанными элементами.

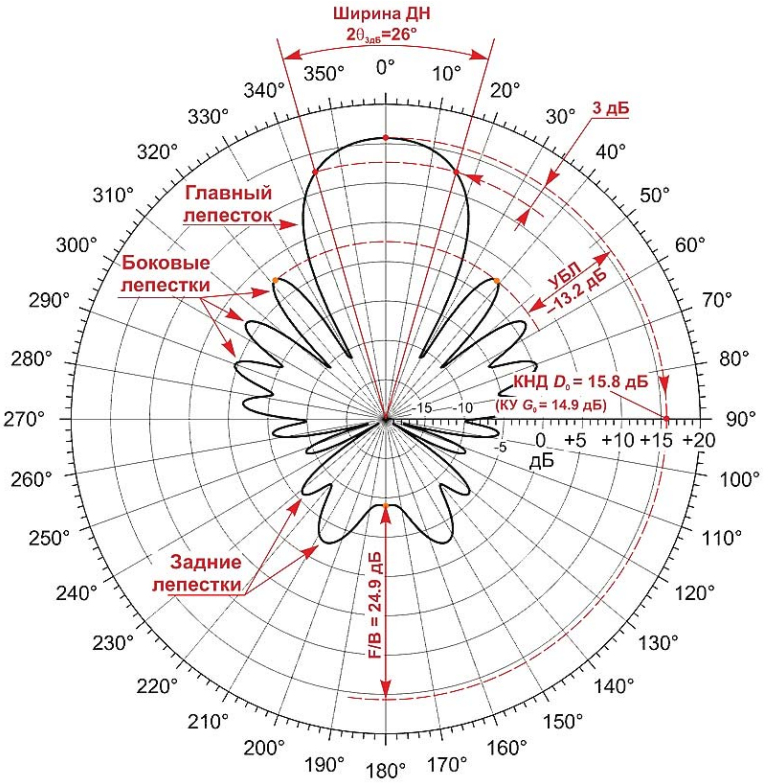


Рисунок 1.3.1.3.2 – Диаграмма направленности антенны

Направленность антенны определяется как ее способность фокусироваться в определенном направлении для передачи или приема энергии.

Третьим важным параметром упоминался тип поляризации. При линейной поляризации вектор амплитуды волны изменяется в одной плоскости (см. рисунок 1.3.1.3.3), и метка должна располагаться строго вдоль длины антенны. При круговой поляризации вектор описывает окружность в плоскости колебаний (см. рисунок 1.3.1.3.4), что даёт возможность располагать метку любым образом.

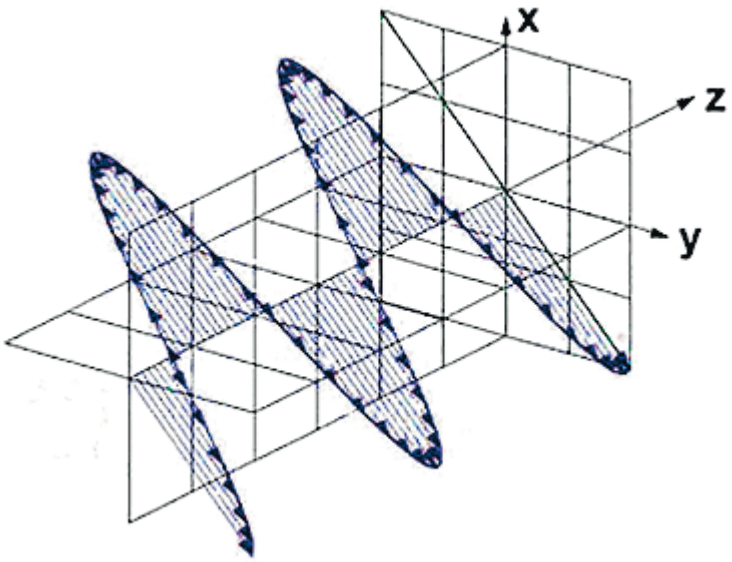


Рисунок 1.3.1.3.3 – колебание вектора амплитуды при линейной поляризации

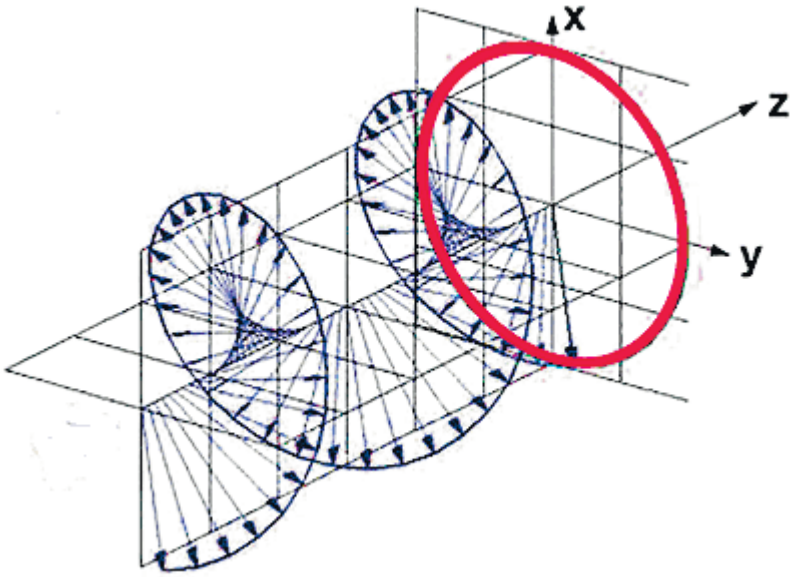


Рисунок 1.3.1.3.4 – колебание вектора амплитуды при круговой поляризации

Например, антенна CLOU CL7205A стоимостью 5 814 рублей [15] обладает следующим набором характеристик:

- ширина луча – 70 градусов;
- частота излучения – 840-960 МГц;
- тип поляризации – левая круговая.

1.3.2 Обзор инструментария для сервера

Веб-сервер.

На данный момент наиболее популярными веб-серверами являются Apache и Nginx [16]. И они конкурируют друг с другом во многих нюансах обработки соединений и запросов.

Например, Apache на запросы клиента создает процессы (или потоки). То есть, процесс следующий: клиент отправляет запрос, веб-сервер создает отдельный процесс, отвечает клиенту, блокирует процесс, затем следует ожидание закрытия соединения от клиента. Это достаточно простая и гибко настраиваема реализация, она хорошо подходит проектам с небольшим потоком запросов даже с учетом роста нагрузки на аппаратную часть при увеличении количества запросов.

Nginx обладает ведущим процессом, управляющим (и создающим) дочерними. Число дочерних обычно соответствует числу ядер аппаратной части. Запросы обслуживаются в бесконечном цикле, ненужные и обслуженные просто удаляются. При этом, загрузка CPU и использование памяти равномерны при широком диапазоне числа запросов. Тем не менее, это свойство чаще полезно в высоконагруженных системах.

Apache проигрывает по обработке статического контента Nginx примерно в 2 раза. Но проект мониторинга общественного транспорта предусматривает наличие большого объёма динамического контента, часто изменяющегося в базе данных. Производительность обработки динамического контента одинакова. Но минусом Nginx является то, что ему приходится делегировать обработку таких данных другим модулям. Иногда для этого используется связка Nginx и Apache. Поэтому Apache в области обработки динамического контента выигрывает.

Методы конфигурации различны, но по сложности не отличаются. Apache пользуется популярностью из-за возможности конфигурировать обработку соединений на уровне директорий. Производится это с помощью скрытого файла ".htaccess", позволяющего настраивать права доступа, авторизацию, аутентификацию, политику кеширования и многие другие правила. Nginx для этого имеет конфигурационный файл для всего проекта, который обрабатывает ведущий поток. При обновлении настроек необходимо отправить сигнал этому потоку, который в свою очередь перезагружает конфигурацию и плавно завершает работу дочерних процессов. Здесь оба веб-сервера конкурируют, но Apache имеет более классический и привычный множеству системных администраторов подход.

И тот, и другой сервер имеют десятки официальных и неофициальных модулей. Методы подключения и установки приблизительно одинаковы и не влияют на общий опыт использования.

Оба сервера совместимы с языком PHP, однако Nginx обрабатывает каждый виртуальный хост в отдельном процессе, значит может использовать разные версии php (и других языков веб среды) одновременно. Каждый процесс может иметь свою собственную независимую конфигурацию. Так как проект мониторинга не предполагает создания какой-либо сложной веб части, этот критерий сравнения не имеет большого влияния [17].

И Nginx, и Apache могут работать и с Unix-подобными системами, и с Windows. Однако, стоит учесть, что Nginx имеет неполную поддержку Windows.

Так как Apache разрабатывается дольше Nginx (1995 год против 2002) [18, 19], то у него сообщество и поддержка шире. Ответ на множество вопросов можно узнать у большого числа пользователей этого сервера. Официальная документация у обоих серверов исчерпывающая и доступная.

В вопросе выбора веб-сервера приоритетным является Apache, так как он имеет качественную поддержку и совместимость с операционной системой Windows (на которой будет создаваться сервис). А также данному выбору способствует невысокая потребность проекта относительно веб составляющей и наличие большого сообщества с документацией.

Язык программирования для веб

В области веб-программирования, особенно серверной части, PHP является одним из наиболее популярных сценарных языков. Несмотря на то, что он в последние годы уступает прогрессивно развивающимся языкам программирования, он продолжает входить в десятку самых используемых языков по версии организации TIOBE, занимающейся анализом этой области [20]. Если учитывать языки, ориентированные только на веб, то PHP становится лидером этой области, позволяющим реализовать проекты любой сложности [21].

Одним из плюсов является кроссплатформенность PHP – то есть можно разрабатывать Интернет-составляющую на любой операционной системе: Linux, Windows, Mac и т.д. Сложности переноса.

PHP является продуктом свободного программного обеспечения, не потребуется платить ни за использование языка в разработке, ни за большинство сопутствующих программ (редакторы, Web-серверы, базы данных). А также большинство программных продуктов, с которыми придется иметь дело, обладают доступным для изучения и модификации открытым исходным кодом.

Данный язык имеет низкий порог входа. Это означает, что начать создавать на нем готовые приложения намного проще, чем с использованием конкурирующих технологий (.NET, Python, Ruby, Go). При этом он не перекрывает возможность использования множества других технологий (веб-серверы, базы данных, библиотеки, вспомогательные языки) и не ограничивает в их выборе. Благодаря этому привлечение разработчиков к проекту, который использует PHP является простой задачей.

Составляющие PHP можно разделить следующим образом:

- компоненты – это библиотеки на PHP, которые собираются при помощи менеджера пакетов Composer;
- фреймворки – готовые сборки, как правило, из компонентов, при помощи которых в будущем можно создать сайт или веб-приложение любой сложности. Например, перечень наиболее популярных за последнее время фреймворков: Symfony, Laravel, Zend, Yii;
- готовые приложения – готовые к использованию разработки на PHP. Они позволяют управлять контентом, форумами, а также предоставляют веб-интерфейс управления базами данных [22]. Например, для управления базой данных выбирается разработка «phpMyAdmin».

Система управления базами данных

Для хранения набора различных типов данных применяется специально разработанное для этого хранилище – база данных. Для обеспечения хранения множества объектов, зависящих друг от друга применяется реляционная модель базы данных – это набор данных с предопределенными связями между ними. В ней данные организованны в виде набора таблиц, состоящих из столбцов и строк. В таблицах хранится информация об объектах, представленных в базе данных. В каждом столбце таблицы хранится определенный тип данных, в каждой ячейке – значение атрибута [23]. Далее рассматриваются системы управления базами данных (СУБД) и выбирается приоритетная для разрабатываемого проекта.

SQLite – встраиваемая в приложения СУБД, базирующаяся на файлах, то она предоставляет довольно широкий набор инструментов для работы с ней, по сравнению с сетевыми СУБД. При работе с этой СУБД обращения происходят напрямую к файлам, в которых хранятся данные, вместо портов и сокетов в сетевых СУБД.

Преимущества SQLite:

- файловая структура – вся база данных состоит из одного файла, поэтому её очень легко переносить на разные машины;
- использование стандартов SQL. Поддерживаются только основные особенности;
- масштабируемость в случае расширения функционала разработки.

Недостатки SQLite:

- отсутствие системы пользователей. Критично для сервисов с большим количеством участников;
- отсутствие инструментов улучшения производительности.

Таким образом, можно сделать вывод, что SQLite больше подходит для простых приложений с малым количеством пользователей редким

обращением к базе данных. Для сервиса, обрабатывающего запросы от большого массива датчиков она не подходит.

Следующей рассматриваемой СУБД является MySQL – это самая распространенная полноценная серверная СУБД. MySQL предоставляет широкий функционал, успешно работает с различными сайтами и веб приложениями, предоставляется бесплатно. У данной СУБД есть хорошо сформировавшееся сообщество, в котором можно найти поддержку.

Преимущества MySQL:

- простота в работе и установке MySQL на сервере. Дополнительные приложения (например, phpMyAdmin) позволяют легко работать с базой данных;
- данная СУБД поддерживает большинство функционала SQL;
- масштабируемость – MySQL легко работает с большими объемами данных и легко масштабируется;
- производительность при упрощении некоторых стандартов.

Недостатки MySQL:

- незначительные ограничения функционала;
- проблемы с надежностью связей, транзакций по сравнению с другими СУБД.

Благодаря широкой поддержке стандартов SQL, достаточно тонкой настройке и достойной производительности эту СУБД можно рассматривать как кандидата на использование для нужд программной части проекта.

Далее рассматривается СУБД PostgreSQL. Она свободно распространяется и максимально соответствует стандартам SQL. Вследствие этого она максимально приближена к профессиональным потребностям при создании сложных проектов.

Хотя PostgreSQL обладает меньшей популярностью в отличии от MySQL, но сейчас существуют приложения, облегчающие работу с

PostgreSQL. Её так же, как и MySQL несложно установить, используя стандартные менеджеры пакетов операционных систем.

Достоинства PostgreSQL:

- открытое бесплатное ПО с открытым исходным кодом;
- наиболее широкое соответствие стандартам SQL;
- большое количество дополнений, позволяющих разрабатывать данные для этой СУБД и управлять ими;
- СУБД является и реляционной, и объектно-ориентированной.

Недостатки PostgreSQL:

- PostgreSQL при некоторых операциях, например, чтении может быть медленнее других СУБД;
- невысокая популярность;
- часто хостинги могут не поддерживать данную СУБД.

Эту СУБД стоит использовать, когда предъявляются наивысшие требования к целостности данных, структура которых также достаточно сложна [24]. То есть для рядового проекта по обработке данных в базе с не самой сложной организацией она будет избыточна, ухудшит производительность, увеличит продолжительность настройки на сервере.

В качестве интерфейса для работы и администрирования СУБД рассматривается наиболее популярное приложение «phpMyAdmin». Это написанное на PHP приложение, обеспечивающее полноценную работу с базами данных MySQL через браузер[25]. Важным плюсом «phpMyAdmin» является то, что оно позволяет во многих случаях обойтись без непосредственного ввода команд SQL. Базой можно управлять через графический интерфейс, который преобразует действия оператора в необходимые команды, что ускоряет процесс разработки.

1.3 ВЫВОД

Технологии радиочастотной идентификации объектов на данный момент хорошо развиты и обладают сформировавшейся базой стандартов, которая продолжает совершенствоваться. Тем не менее, текущий прогресс в данной области позволяет создавать крупные системы мониторинга объектов с высокой производительностью и с минимальными ошибками в процессе считывания информации.

Сканирование объектов посредством радиочастотной идентификации является новшеством для обширной городской среды. Это влечет за собой разработку программно-аппаратного комплекса, способного правильно идентифицировать транспорт в месте посадки пассажиров, а также выполнять это в условиях хаотичного движения и неидеального положения этого транспорта относительно зон мониторинга.

Стоит учитывать финансовую выгоду от использования систем радиочастотной идентификации по сравнению с отслеживанием по спутниковой навигации. Например, для оборудования 100 остановок считывателями, парами антенн, соединительными элементами, корпусными частями необходимо приблизительно 3 миллиона рублей. Для оборудования 1000 машин GPS/ГЛОНАСС устройствами нужно около 3,8 миллионов рублей, плюс ежегодные расходы на связь примерно 250 тысяч рублей. Стоимость системы радиочастотной идентификации транспорта на остановках может быть меньше примерно в 2 раза при наладке производства отечественных комплектующих и разработке узконаправленных считывателей меток в будущем с не такой избыточной производительностью, как у существующих решений.

Учитывая, что разрабатываемый комплекс программно-аппаратный, а расположение считывателей на остановках в городе является достаточно рассредоточенным, его программной частью является веб-сервер Apache. Он позволит организовать получение информации от большого количества

датчиков в базе данных и её хранение. В качестве СУБД подходит MySQL, так как её функционал при хорошей производительности удовлетворяет потребностям разработки и не станет избыточным. Различные действия на остановке могут обрабатываться аппаратным комплексом и отправляться на сервер с записью в соответствующие таблицы базы данных посредством использования PHP скриптов.

2 ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ

Для реализации разрабатываемой системы потребуется набор компонентов:

1. RFID-модуль считывания с возможностью обрабатывать не менее 10 меток в секунду и с возможностью подключения внешней антенны, а также с поддержкой SPI или UART интерфейсов.
2. Микроконтроллер, который будет обеспечивать обмен данными с RFID-модулем и отправлять их в базу данных на сервер.
3. MySQL база данных, обеспечивающая хранение данных о местоположении транспорта в реальном времени, а также историю маршрутов (полезная информация для диспетчеров).
4. HTTP-сервер Apache предоставляющий работу с PHP скриптами. Сервер принимает данные от множества считывателей, расположенных в городе. С помощью веб-интерфейса предоставляется доступ к текущему местоположению машин. Также предоставляются расширенные данные (например, продолжительность остановки, общая длительность маршрута, время между двумя остановками) для диспетчеров, ответственных за определенный маршрут.
5. Графический интерфейс клиентов (для пассажиров и диспетчеров).

2.1 ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ

На примере диаграммы вариантов использования системы (см. рисунок 2.1.1) рассмотрим взаимодействующих с ней актеров.

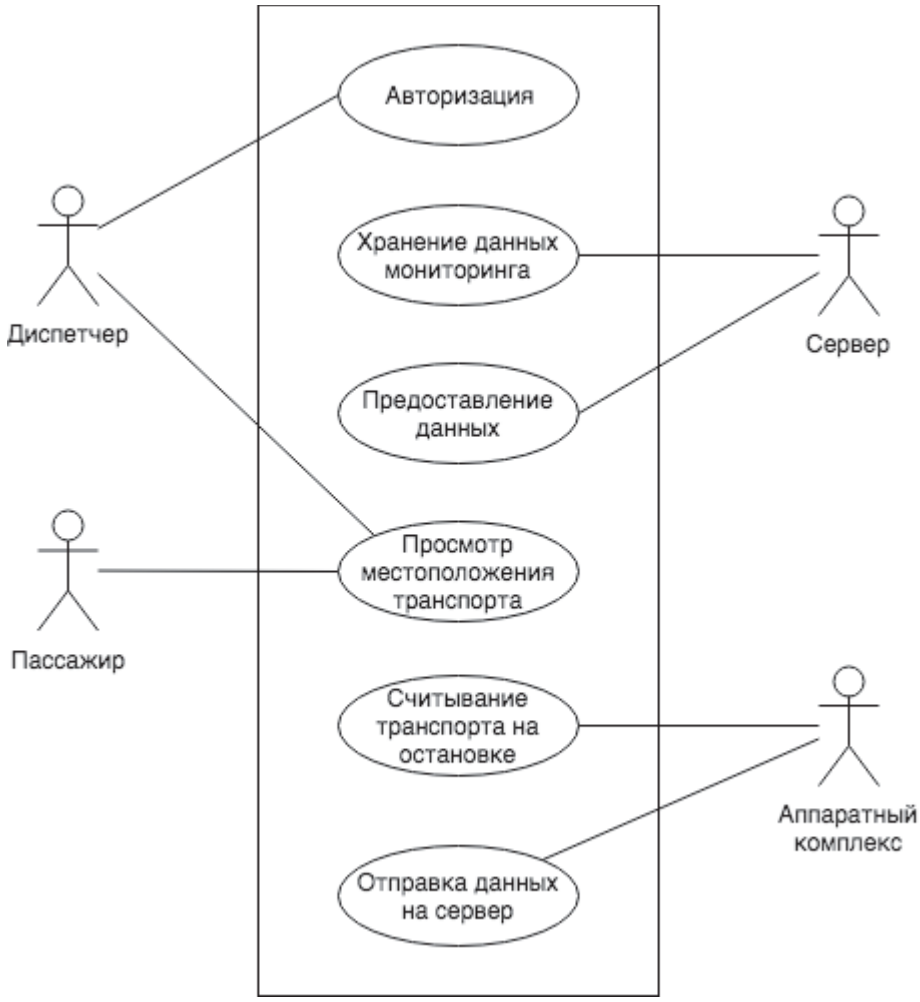


Рисунок 2.1.1 – Диаграмма вариантов использования системы мониторинга транспорта

Получается следующий перечень актеров:

1. Диспетчер – работник депо (компании перевозчика), осуществляющий деятельность по наблюдению за техническим состоянием транспорта и принятию решений по устранению неисправностей, а также исполняющий обязанности по логистике подвижного состава.

2. Пассажир – лицо, пользующееся услугами транспортных компаний.

3. Сервер – информационная система, собирающая информацию о местоположении транспорта в реальном времени, занимающаяся долгосрочным хранением таких данных, предоставляющая их пользователям и диспетчерам.

Все актеры имеют какое-либо отношение к прецедентам (возможностям) системы, которые рассматриваются далее.

Для доступа к большему количеству данных диспетчеру необходимо авторизоваться в системе. Обычным пользователям, для их удобства, нет необходимости это делать.

И пользователь, и диспетчер могут просматривать местоположение транспорта. Отличие состоит в том, что диспетчеру за счет его учетной записи предоставляется накопленная сервером история.

Аппаратный комплекс, устанавливаемый на остановках производит регулярное считывание области посадки и высадки пассажиров. При наличии в этой области какой-либо единицы транспорта с RFID меткой, он должен считать, что это за метка, то есть её уникальный идентификатор. Затем следует отправка данных идентификатора метки, а также остановки, на которой установлен аппаратный комплекс на сервер.

Сервер, в свою очередь, ответственен за хранение данных, а также за предоставление их обращающимся к нему клиентам.

2.1.1 Реализуемые задачи

Перечень задач:

1. Фиксация времени прибытия транспортной единицы на остановку и убытия с неё.
2. Вычисление средней скорости между остановками.
3. Вычисление общей продолжительности маршрут.

4. Вычисление плотности машин конкретного маршрута на остановке (машин в час).

5. Ведение и выдача данных диспетчеру и пассажирам в режиме онлайн.

Необходима интеграция с картографическим сервисом для визуального отображения машин на маршруте. Наилучшим вариантом предполагаются Яндекс карты за счет отличного сервиса на территории России, детализированных данных, доступного и простого API. Также для вычисления приблизительного местоположения транспортной единицы на интервале между двумя остановками потребуются данные о пробках от этого сервиса.

2.1.2 Требования к ПО

Программное обеспечение должно включать в себя серверную и клиентскую части. Клиенты ориентируются на пассажиров и диспетчеров. При этом клиенты должны быть реализованы в виде веб-приложения.

Серверная часть должна обеспечивать возможность использования следующих функциональных возможностей:

1. Регистрация (авторизация) диспетчеров.
2. Обращение к базе данных для получения отчетов.
3. Вычисление различных динамических параметров транспортных единиц (скорость, местоположение).

Клиентская часть диспетчера должна помогать осуществлять должностному лицу свои обязанности в области управления, назначения, снятия машин на маршрутах на основании данных о плотности машин на остановке, средней скорости движения, времени пребывания машин на остановках. А также должна иметь возможность привязывания к RFID-метке данных транспортной единицы.

Клиентская часть пассажира должна предоставлять ему данные о текущем местоположении машины интересующего маршрута. Для удобства пользователей доступ к сервису должен обеспечиваться без обязательной

регистрации. Клиент пассажира должен отображать как карту, так и простой список того, как движется ближайший к этой остановке транспорт. При использовании списка необходимо указание остановки, на которой происходит ожидание, затем будет получена информация о единице транспорта, где она находится. Перечень возможных сообщений:

1. №75 находится на остановке "ЮУрГУ".
2. №370 отправился с остановки "Доватора".
3. №92 находится (отправился) на (с) остановке (-и) "ДС Юность".

В соответствии с выше перечисленными требованиями, отчеты базы данных следующие:

1. Время пребывания данной машины на данной остановке.
2. Плотность машин (машин в час) между двумя указанными точками (остановками).
3. Количество циклов маршрута для данной машины.
4. Средняя длительность одного цикла.
5. Положение данной машины (сообщения в соответствии с перечнем текстового отображения процесса движения транспорта).

Перечень предоставляемых отчетов и данных может быть расширен по запросам диспетчеров и отзывам пользователей сервиса в будущем при тестировании системы.

2.2 НЕФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ

2.2.1 Условия эксплуатации

Город, административный центр области. Умеренный континентальный климат, средняя температура зимой около -15 градусов по Цельсию с минимумами до в -45 градусов раз в 5-7 лет, средняя температура летом примерно 18 градусов по Цельсию с максимумами до 35 градусов так же раз в 5-7 лет. Зима длительная, умеренно-холодная, снежная, метели наблюдаются в

течение до 30-и дней. Лето умеренно теплое, сухое, редко – дождливое. Среднегодовое количество осадков – до 430 мм.

Электромагнитный фон осложнен наличием электрического транспорта (троллейбусы, трамваи).

2.2.2 Требования к пользователям сервиса

Пользователи (пассажиры и диспетчеры, либо иные работники транспортных компаний, осуществляющих ввод машин в рейс и контролирующих текущую дорожную ситуацию) должны обладать базовыми навыками пользования компьютером и мобильной техникой, в частности, веб браузером.

Также сотрудники транспортных компаний обязаны своевременно предоставлять информацию об изменении номера маршрута определённой машины путём редактирования этой информации в базе данных

2.2.3 Характеристики сервера

Вычислительные мощности системы должны быть рассчитаны на обработку запросов от 100 до 300 остановок в секунду.

Для улучшения работы транспортной инфраструктуры и прогнозирования тенденций её развития в будущем необходимо предусмотреть пространство для архива данных за последние 5 лет.

2.2.4 Характеристики аппаратного комплекса на остановке

Устанавливаемый на остановки комплекс устройств должен безошибочно идентифицировать приезжающий транспорт. Для этого считыватели должны поддерживать антиколлизионные алгоритмы, а также необходимо исключить ошибочные считывания транспорта, не останавливающегося на данной остановке. Например, проезжающего в полосе движения противоположного направления.

2.2.5 Характеристики RFID-меток

Метки на подвижном составе должны быть защищены от проникновения и изменения к содержащейся в них информации третьими лицами. Это может обеспечиваться путём использования однократно записываемых меток, либо тех, память которых предназначена только для чтения.

3 ПРОЕКТИРОВАНИЕ

3.1 АРХИТЕКТУРА ПРЕДЛАГАЕМОГО РЕШЕНИЯ

Рассмотрим топологию объектов (см. рисунок 3.1.1), устанавливаемых на остановки, которые необходимы для считывания данных о транспорте на ней.

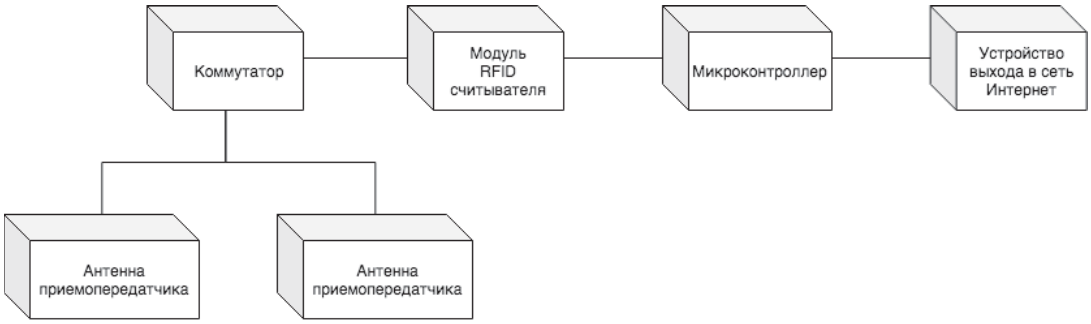


Рисунок 3.1.1 – топология объектов на остановке

Модуль RFID посредством антенн получает данные от меток и обработанную информацию передаёт на микроконтроллер, который формирует запрос для сервера и передаёт его устройству выхода в интернет. Таким устройством может быть GSM модуль, WiFi модуль, либо плата с разъёмом Ethernet.

Для центральной части города, где есть большое число публичных точек доступа беспроводной сети, наиболее оптимальным решением будет использование модуля WiFi. На остановочных комплексах с киосками и магазинами наиболее высока вероятность встретить проводное подключение к сети Интернет, что влечёт использование платы для коннектора RJ-45. В

остальных местах хорошим вариантом будет использование мобильной сети и модуля GSM.

Учитывая физические ограничения параметров RFID антенн, было спроектировано приблизительное их размещение на объектах остановочных комплексов. На рисунке 3.1.2 представлена одна из таких остановок.

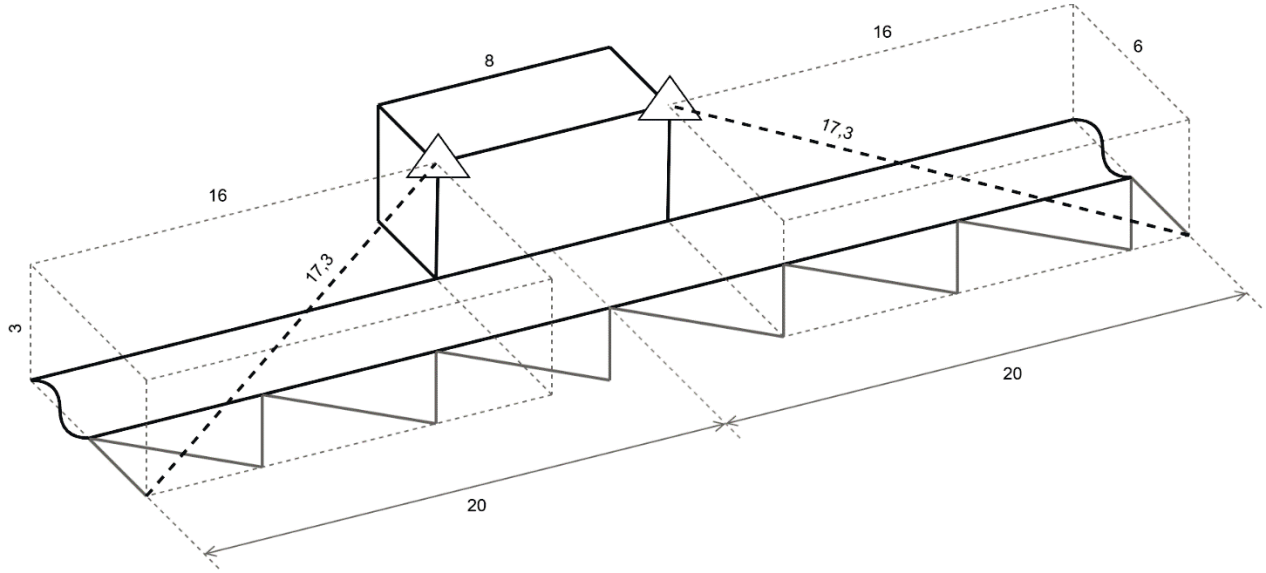


Рисунок 3.1.2 – размещение двух антенн на остановке

На этом рисунке антенны условно обозначены треугольниками в углах предполагаемой конструкции на остановке (параллелепипед на рисунке). С учетом ширины тротуара приблизительно 3 метра, и примерно такой же шириной полосы остановки (отмечена треугольной разметкой) получается ширина считываемого пространства 6 метров. Длина, за вычетом габаритов строения на остановке составляет 16 метров. Итого, до дальнего угла полосы остановки необходимая дальность считывания 17,3 метра.

Антенны не дают настолько четко ограниченной и ровной области считывания. Поэтому возможно принятие сигнала с полосы движения, следующей за остановочной. Зачастую это может быть плюсом, потому что транспорт не всегда идеально останавливается именно на разметке остановки. Не слишком большая ширина луча антенн (70-80 градусов) позволит направить их так, чтобы затрагивалась только полоса остановки и попутного движения, следующая за ней.

На рисунке 3.1.3 представлен вариант размещения одной антенны.

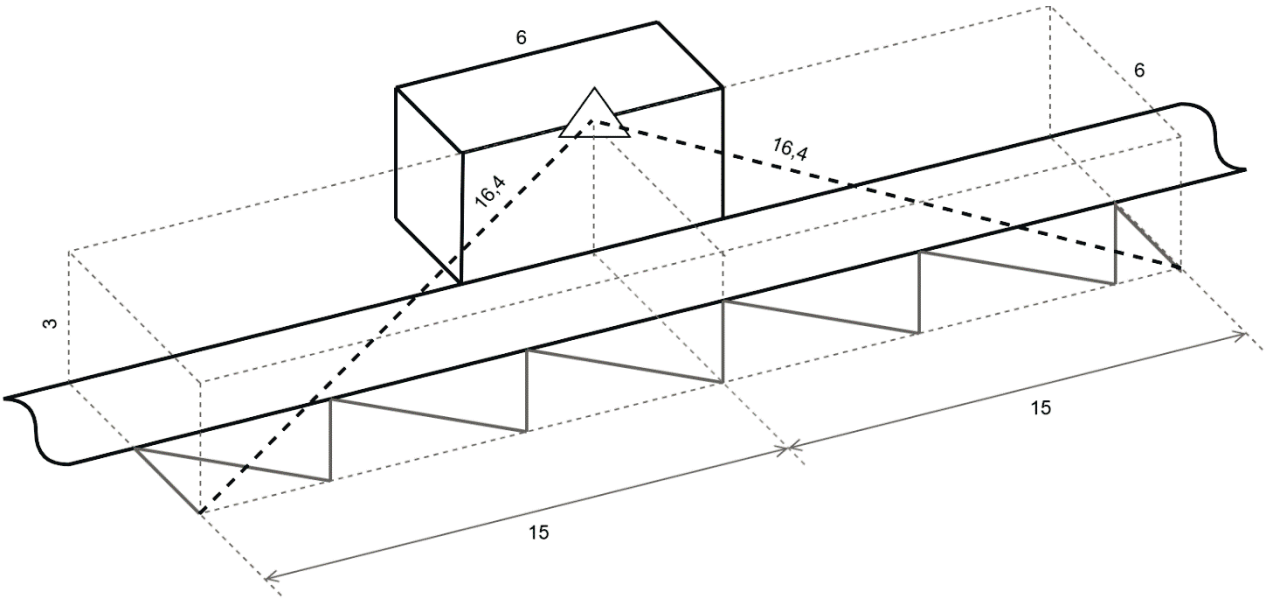


Рисунок 3.1.3 – вариант размещения одной антенны

У этого способа есть ряд недостатков. Например, антенна должна обладать очень широким лучом, чтобы обрабатывать всю полосу остановки, порядка 120 градусов. Для этого придётся уменьшить коэффициент усиления, что повлечёт уменьшение дальности считывания.

Такое решение можно применять на самых малых остановках, которые обслуживаются буквально не более, чем десятком маршрутов. При этом протяжённость остановочной полосы должна быть меньше примерно на 10-15 метров, чем на рисунке, где она составляет 30 метров (15 метров симметрично в обе стороны относительно остановки).

На рисунке 3.1.4 показаны расстояния считывания до меток (отмеченных квадратами), которые установлены на транспортных средствах (примерные габариты показаны параллелепипедами с пунктирной линией).

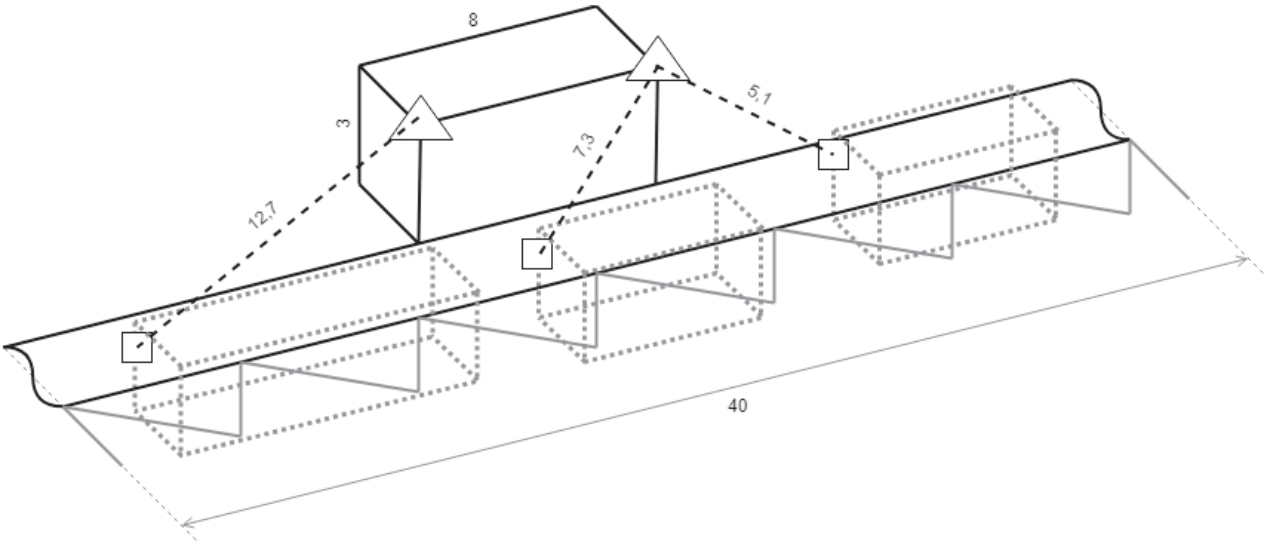


Рисунок 3.1.4 – считывание меток с транспорта

3.2 АЛГОРИТМЫ РЕШЕНИЯ ЗАДАЧИ

При разработке алгоритма считывания и отправки данных необходимо учитывать особенности работы RFID считывателей: они позволяют моментально обнаружить метку в поле считывания, после чего та зачастую становится неактивной при дальнейших опросах пока не покинет поле считывания. Это означает, что момент ухода из зоны сканирования вызывает ряд трудностей, которые можно решить путём разработки такого алгоритма, который позволит не только регулярно собирать метки из зоны считывания, но и запоминать набор транспондеров от предыдущей операции считывания.

Например, вводится два массива: в первом набираются метки с транспорта на остановке, во втором хранятся метки от предыдущего считывания. При этом, метки не переводятся в неактивное состояния, а регулярно опрашиваются снова, чтобы знать время отправки машины с остановки.

На рисунке 3.2.1 представлен укрупнённый алгоритм работы всей системы, от этапа сбора данных на остановке, до действий сервера.

Рисунок 3.2.1 – Укрупненный алгоритм работы системы

Основной смысл алгоритма в том, что аппаратный комплекс сканирует пространство посадки и высадки пассажиров на наличие меток и определяет, какие прибыли, и какие покинули место посадки пассажиров. Он регулярно посылает запрос на сервер с данными о наборе меток за прошедший короткий промежуток времени (в процессе разработки алгоритма выбранный интервал составляет 4 секунды), об остановке, на которой он установлен, а также о состоянии единицы транспорта (приехала она на остановку только что, либо уже начала движение к следующей).

На рисунке 3.2.2 представлен алгоритм основного цикла программы микроконтроллера.

Рисунок 3.2.2 – Алгоритм основного цикла программы

Сначала необходимо дождаться появления отслеживаемых системой меток. После того, как появляется какая-либо метка, необходимо удостовериться, что система может распознать её уникальный идентификатор. Затем предусматривается переменная для временного хранения метки в процессе проверки на наличие в массиве. Если она ранее не была добавлена, и это её первое сканирование, то её необходимо занести в упомянутый массив. Этот массив служит буфером, в который набираются метки в интервалах времени между запросами к серверу.

Создание подобного временного хранилища меток обуславливается особенностями работы RFID считывателей: они позволяют моментально обнаружить метку в поле считывания, но момент выхода её из зоны считывания приходится отслеживать разработкой алгоритмов и дополнительных элементов.

Простой записью идентификаторов меток работа с массивом текущего считывания не ограничивается. Дальнейшие действия с ним происходят в обработчике прерывания таймера микроконтроллера. Каждая метка из текущего сравнивается с массивом меток предыдущего считывания. Если проверяемый уникальный идентификатор уже был на прошлой итерации считывания, то на последовательный порт нет необходимости выводить какую-либо информацию, так как эта метка уже отмечена как прибывшая на остановку и сейчас находится на ней. В ином случае, когда идентификатор метки отсутствует в массиве прошлого считывания, нужно вывести информацию о том, что это только что прибывшее транспортное средство. Изначально по алгоритму предполагается отметить все метки массива прошлого считывания как убывшие с данной остановки. В случае нахождения совпадений с текущим информация исправляется, и метка считается некоторые время пребывающей в этом месте. После проведения этих операций для каждой из меток массива текущего считывания выполняется переход к меткам массива прошлого считывания. Все транспондеры, оставшиеся отмеченными как убывшие выводятся на последовательный порт с соответствующей информацией. Затем работа по определению прибывших и убывших меток считается выполненной, поэтому текущий массив копируется в массив прошлого считывания, и очищается для следующей итерации сканирования. Алгоритм обработчика представлен на рисунке 3.2.3.

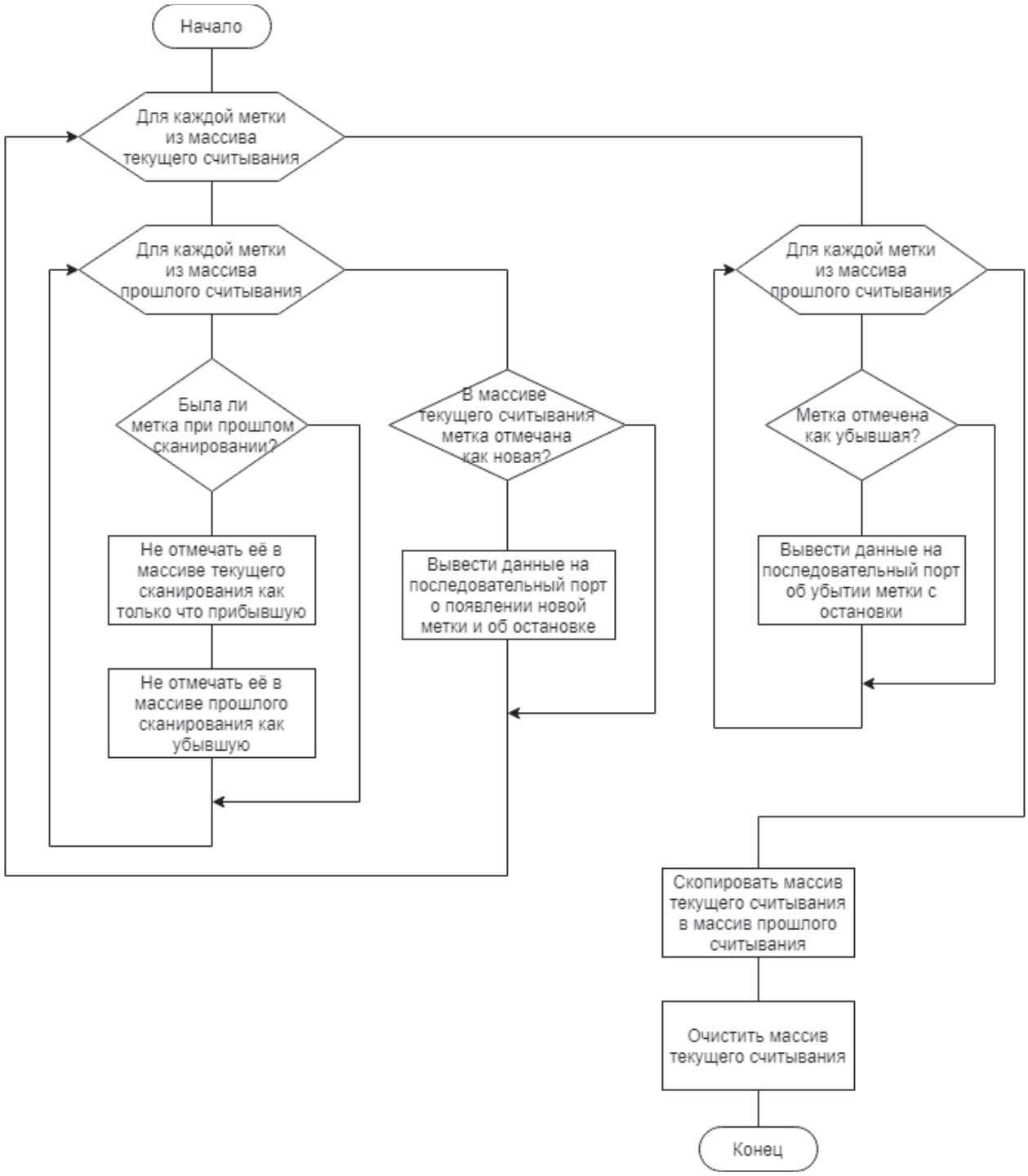


Рисунок 3.2.3 – алгоритм обработчика прерывания

Все передаваемые на последовательный порт данные обрабатываются устройством для соединения с сетью Интернет. Например, в данной работе рассматривается плата "NodeMCU" с Wi-Fi модулем "ESP-12E". На рисунке 3.2.4 представлен алгоритм работы этого устройства.

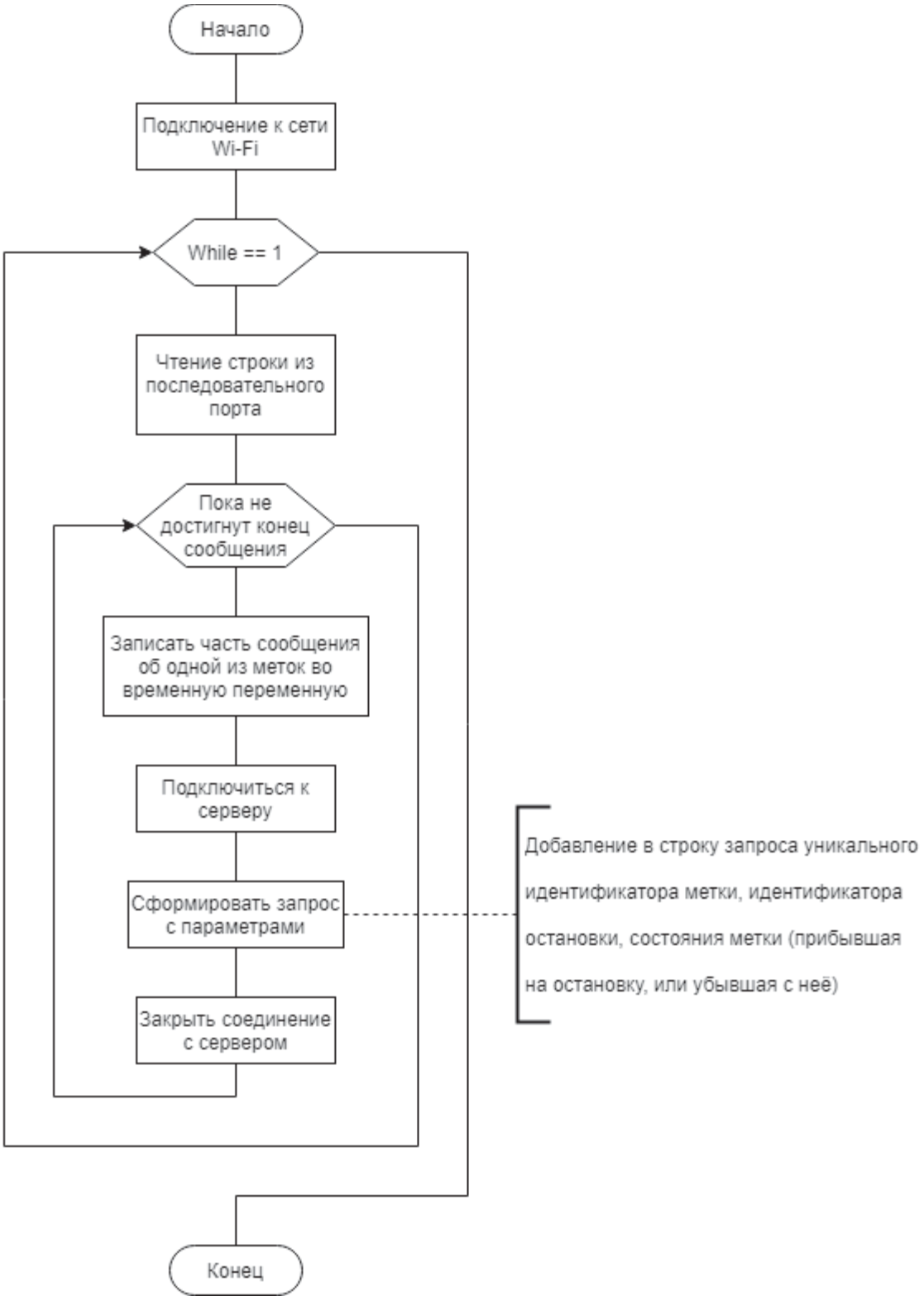


Рисунок 3.2.4 – алгоритм работы модуля беспроводного соединения

Беспроводной модуль подключается к сети Wi-Fi с указанными в коде названием сети и паролем. Затем, в бесконечном цикле устройство сначала ожидает доступные для чтения из последовательного порта байты. При

появлении данных в символьный массив записывается строка установленного формата (см. рисунок 3.2.5).

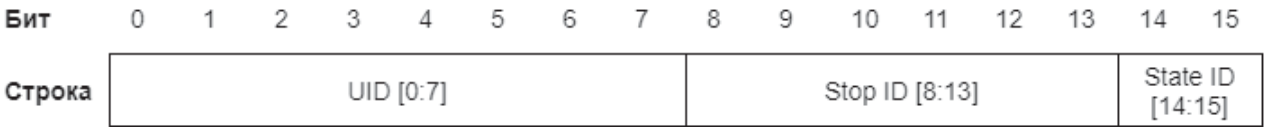


Рисунок 3.2.5 – формат обрабатываемой строки

В этой строке UID – уникальный идентификатор метки, Stop ID – идентификатор остановки, State ID – идентификатор состояния (машина только прибыла, либо уехала). Из полученной строки выбираются перечисленные части для записи их в параметры запроса на сервер. После чего соединение закрывается.

На сервере php скрипт обрабатывает полученный запрос. Принятые параметры записываются в переменные, считывается текущая дата для установленного часового пояса, формируется SQL-запрос вставки, либо обновления данных в таблице текущего местоположения транспортных единиц. Если транспортное средство ни разу не появлялось в этой таблице ранее, то применяется операция вставки. В ином случае обновляется статус, остановка и время.

На сервере также имеется скрипт для отображения информации по текущему местоположению транспорта, он производит выборку из нескольких таблиц, в результате которой получается информация о марке, номере маршрута, прибытии (или убытии), времени этого прибытия (убытия).

3.3 ОПИСАНИЕ ДАННЫХ

Для проверки работоспособности аппаратного комплекса, его взаимодействия с сервером, а также для организации хранения обработанной информации необходимо наличие базы данных. Общая схема базы с использованием UML диаграмм представлена на рисунке 3.3.1.

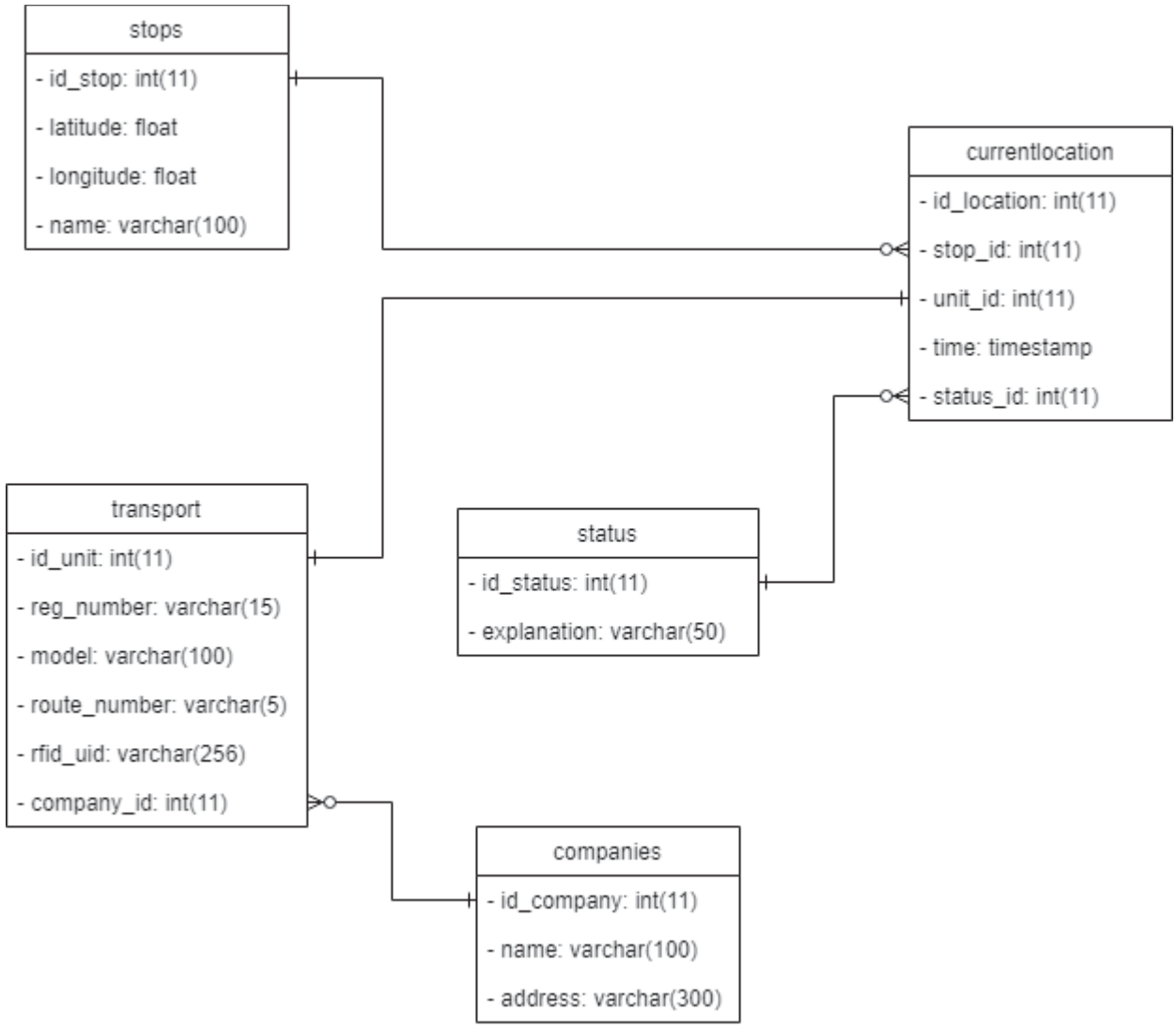


Рисунок 3.3.1 – схема разработанной базы данных

Для хранения информации о взаимодействующих с системой компаниях предназначена таблица "companies", в которой содержатся минимально необходимые данные (см. таблицу 3.3.1).

Таблица 3.3.1 – таблица "companies"

Название столбца	**Тип данных**	**Назначение**
id_company	int	Первичный ключ, уникальный идентификатор
name	varchar	Наименование компании
address	varchar	Адрес компании

Чтобы связать данные о транспорте, принадлежности машин компаниям, обслуживаемому маршруту, и уникальном идентификаторе RFID метки понадобится перечень этих транспортных средств (см. таблицу 3.3.2).

Таблица 3.3.2 – таблица "transport"

Название столбца	**Тип данных**	**Назначение**
id_unit	int	Первичный ключ, уникальный идентификатор
reg_number	varchar	Регистрационный знак транспортного средства необходимый для их учёта
model	varchar	Модель транспортного средства, выступает как одно из средств учёта для компании-перевозчика
route_number	varchar	Номер маршрута, по которому следует единица транспорта
frid_uid	varchar	Уникальный идентификатор RFID метки, позволяющий однозначно определить объект в системе радиочастотной идентификации
company_id	int	Внешний ключ таблицы «companies», позволяет определить, какой организации принадлежит транспортное средство

Для хранения данных об остановках и их местоположении необходима таблица "stops" (см. таблицу 3.3.3).

Таблица 3.3.3 – таблица "stops"

Название столбца	**Тип данных**	**Назначение**
id_stop	int	Первичный ключ, уникальный идентификатор

Продолжение таблицы 3.3.3

Название столбца	Тип данных	Назначение
latitude	float	Ширина, для местоположения
longitude	float	Долгота, для местоположения
name	varchar	Название остановки

Таблица "status" (см. таблицу 3.3.4) предназначена для указания состояния транспортной единицы на остановке: только что приехало, или отправилось к следующей остановке.

Таблица 3.3.4 – таблица «status»

Название столбца	Тип данных	Назначение
id_status	int	Первичный ключ, уникальный идентификатор
explanation	varchar	Описание статуса транспортного средства для определения, приехало оно на остановку, или уже отправилось к следующей

Для отображения информации о местоположении машин в реальном времени необходима таблица «currentlocation», в которой содержатся данные об идентификаторах объектов, которые участвуют в этом отображении (см. таблицу 3.3.5).

Таблица 3.3.5 – таблица «currentlocation»

Название столбца	Тип данных	Назначение
id_location	int	Первичный ключ, уникальный идентификатор
unit_id	int	Внешний ключ таблицы «transport», для определения, какому транспортному средству принадлежит конкретная запись в таблице

Продолжение таблицы 3.3.5

Название столбца	Тип данных	Назначение
stop_id	int	Внешний ключ таблицы «stops», благодаря которому можно узнать, от какой остановки исходит информация
time	timestamp	Дата и время прибытия (или убытия) на остановку
status_id	int	Внешний ключ таблицы «status», по нему определяется, какое последнее действие было на остановке: прибытие, или убытие к следующей.

4 РЕАЛИЗАЦИЯ

Для проверки работоспособности системы и взаимодействия её ключевых компонентов реализован макет с применением платы "Arduino UNO" на базе микроконтроллера "Atmel Mega 328P" [26], считывателя "RFID-RC522" [27] и платы "NodeMCU" на базе "ESP8266" с возможностью подключения к WiFi [28].

В роли сервера выступает сборка на Apache версии 2.4, СУБД MySQL версии 8.0 и языка PHP версии 7.3.2.

4.1 РЕАЛИЗАЦИЯ МАКЕТА

Чтобы проверить в действии алгоритмы работы системы, необходимо собрать макет с участием упомянутых выше плат.

Плата Arduino с микроконтроллером Atmega

Основой макета выступает плата Arduino UNO. Данное устройство предлагает всё необходимое для удобной работы с микроконтроллером: 14 цифровых входов/выходов (6 из них могут использоваться в качестве ШИМ-

выходов), 6 аналоговых входов, кварцевый резонатор на 16 МГц, разъём USB, разъём питания, разъём для внутрисхемного программирования (ICSP) и кнопка сброса. Внешний вид платы можно увидеть на рисунке 4.1.1.

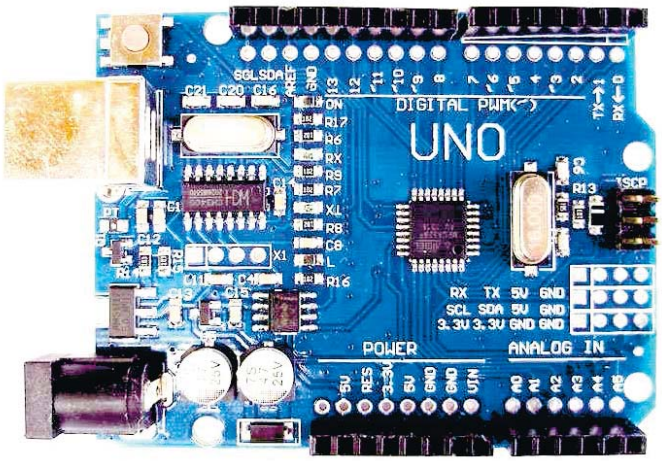


Рисунок 4.1.1 – Плата Arduino UNO

Выбор данной платы обуславливается тем, что среда продуктов Arduino (как аппаратная часть, так и программное обеспечение Arduino IDE) позволяет быстро начать разработку проекта, делая упор на проектирование алгоритма. Наличие множества библиотек для периферийных устройств помогает мгновенно пройти этап их настройки. В таблице 4.1.1 приведены характеристики платы.

Таблица 4.1.1 – характеристики платы Arduino UNO

Параметр	**Показатель**
Микроконтроллер	ATmega328P
Тактовая частота	16 МГц
Напряжение питания	USB – 5 В Разъём внешнего питания – 5-12 В
Цифровые порты ввода-вывода	14 штук
Цифровые ШИМ порты	6 штук
Аналоговый вход	6 штук
Ток на портах ввода-вывода	Постоянный 20 мА для 5 В Постоянный 50 мА Для 3.3 В

Продолжение таблицы 4.1.1

Параметр	**Показатель**
FLASH память	32 кБ (5 кБ занято загрузчиком)
SRAM	2 кБ
EEPROM	1 кБ
Размеры	68 x 53 мм

Плата считывателя RFID-RC522

Считыватель RC522 – периферийное устройство, позволяющее получать данные от меток. Его рабочая частота 13,56 МГц. Она отличается от той, которая должна применяться в городских условиях, но это не влияет на общее представление о работе системы, так как каждый из считывателей, независимо от его частотного диапазона, можно рассматривать как абстрактное периферийное устройство. Внешний вид платы считывателя представлен на рисунке 4.1.2.

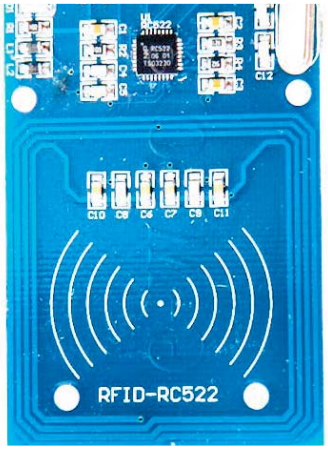


Рисунок 4.1.2 – Считыватель RC522

Работа с метками происходит по стандарту ISO 14443 Type A. Ему соответствуют пластиковые карты Mifare, а также брелки. Существует версия на русском языке ГОСТ Р ИСО/МЭК 14443. Карты и брелки имеют одинаковый рамочный тип антенны (см. рисунки 4.1.3, 4.1.4), как и сам модуль. Сигнал модуля воспринимается антенной метки, а также служит источником энергии для чипа.

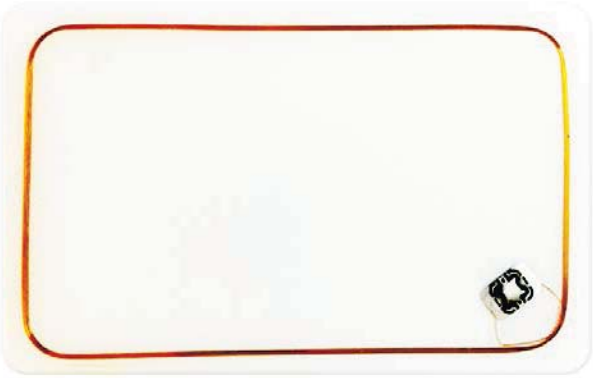


Рисунок 4.1.3 – Антенна карты Mifare

Рисунок 4.1.4 – Антенна брелка Mifare

Считыватель RFID RC522 срабатывает при поднесении метки. Он может обрабатывать информацию одновременно от нескольких меток. Для считывания информации достаточно ненадолго попасть карте в зону регистрации даже при перемещении на большой скорости. Полные характеристики считывателя RC522 представлены в таблице 4.1.2.

Таблица 4.1.2 – характеристики считывателя RC522

Параметр	**Показатель**
Напряжение питания	3,3 В
Ток потребления	Дежурный режим – 0,08 мА Режим ожидания – 12 мА Обычный режим – не более 26 мА Наибольший 30 мА

Продолжение таблицы 4.1.2

Параметр	**Показатель**

Частота HF	13,56 МГц
Частотная полоса	13,55–13,57 МГц
Расстояние считывания	0–25 мм
Сопровождаемые карты	классы S50, S70, Ultralight, Pro, DESFire типы Mifare S50, Mifare S70, Mifare UltraLight, Mifare Pro, Mifare DESfire
Скорость передачи информации	106, 212, 424, 848 кбит/с
Поддерживаемый интерфейс	SPI
Протокол	ISO 14443 A Mifare classic protocol
Шифрование	Security Features Mifare classic
Размеры	40 x 60 мм
Температура	Рабочая – 20…80 градусов Цельсия Хранения – 40…85 градусов Цельсия

Данный считыватель выбран по причине предоставляемого им интерфейса SPI, который применяется во множестве ультравысокочастотных считывателей, что позволит смоделировать систему максимально близкую к реальной (не учитывая рабочую частоту). Другим важным критерием является наличие библиотеки для работы с ним.

Плата NodeMCU для соединения WiFi

NodeMCU – плата (см. рисунок 4.1.5) для разработки на базе чипа ESP8266 (версия ESP12E), который представляет собой WiFi модуль, подключаемый по UART протоколу. Данная плата позволяет упростить разработку устройств, которым необходимо соединение с сервером т.к. на ней уже реализованы все необходимые компоненты: разъём micro-USB, стабилизатор напряжения питания, выводы чипа разведены на контактные ножки со стандартным шагом 2.54 мм, что позволяет использовать устройство в макетной плате. Кроме того, плата поставляется с прошивкой NodeMCU, что

позволяет программировать ее с помощью языка Lua или с помощью Arduino IDE. Полные технические характеристики перечислены в таблице 4.1.3.

Рисунок 4.1.5 – внешний вид платы NodeMCU

Таблица 4.1.3 – технические характеристики NodeMCU

Параметр	**Показатель**
Напряжение питания	4.5 – 9 В
Потребляемый ток	Во время обмена – 70-200 мА Режим ожидания – 0.2 мА
Стандарт WiFi	802.11 b/g/n
Цифровые порты ввода-вывода	11 штук, некоторые из них могут быть сконфигурированы под потребности протоколов передачи данных
Аналоговые порты ввода-вывода	1 штука
Ток портов ввода-вывода	15 мА
Поддерживаемые протоколы	UART, I^2C, SPI
Рабочая температура	От -40 до +125 градусов Цельсия
Размеры	47 x 31 мм

Сборка макета

После написания алгоритма приёма информации от считывателя и отправки на последовательный порт для WiFi модуля, а также программы непосредственно для самой WiFi платы, которая формирует запрос на сервер,

необходимо соединить все компоненты системы. Схема соединения представлена на рисунке 4.1.6.

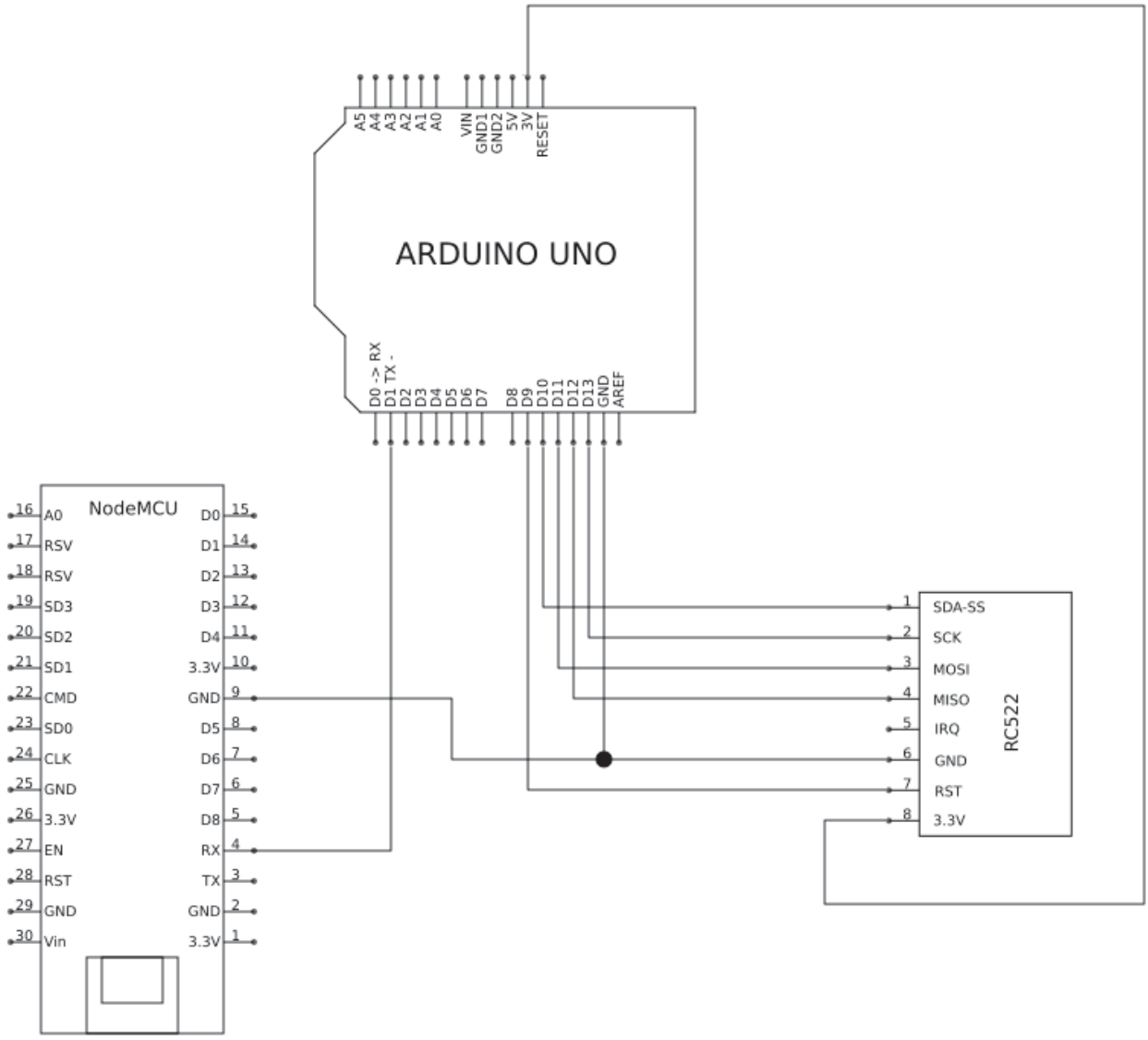


Рисунок 4.1.6 – схема соединения компонентов макета

По схеме можно увидеть, что микроконтроллер работает со считывателем по протоколу SPI (Serial Peripheral Interface – последовательный периферийный интерфейс), предназначение линий следующее:

- MOSI — Master Output / Slave Input. Выход ведущего / вход ведомого. Служит для передачи данных от ведущего устройства к ведомому;

- MISO – Master Input / Slave Output. Вход ведущего / выход ведомого. Служит для передачи данных от ведомого устройства к ведущему;
- SLK — Serial Clock. Сигнал синхронизации. Служит для передачи тактового сигнала всем ведомым устройствам;
- SS — Slave Select. Выбор ведомого устройства (необходим на случай, если к микроконтроллеру подключается несколько периферийных устройств, использующих этот протокол).

WiFi модулю микроконтроллер Atmega передаёт информацию по протоколу UART (англ. Universal Asynchronous Receiver-Transmitter – универсальный асинхронный приёмопередатчик).

В данном протоколе используется всего две линии: одна для передачи от ведущего к ведомому, вторая – от ведомого к ведущему. Называется асинхронным, потому что не использует сигнала синхронизации. Так как микроконтроллер Atmega только передаёт информацию ESP, то используется одна линия связи.

Программирование

Для программирования микроконтроллеров используется Arduino IDE (англ. Integrated Development Environment – интегрированная среда разработки). С помощью неё, а также встроенных библиотек гораздо быстрее и удобнее разрабатывать алгоритм работы микроконтроллера, чем при использовании обычных текстовых редакторов и других средств разработки.

Программы, написанные в этой среде называются скетчами. Язык программирования похож по синтаксису на C-подобные. К тому же, в момент загрузки скетча в микроконтроллер, но действительно конвертируется в C++ и компилируется. На рисунке 4.1.7 представлено окно программы Arduino IDE во время написания скетча.

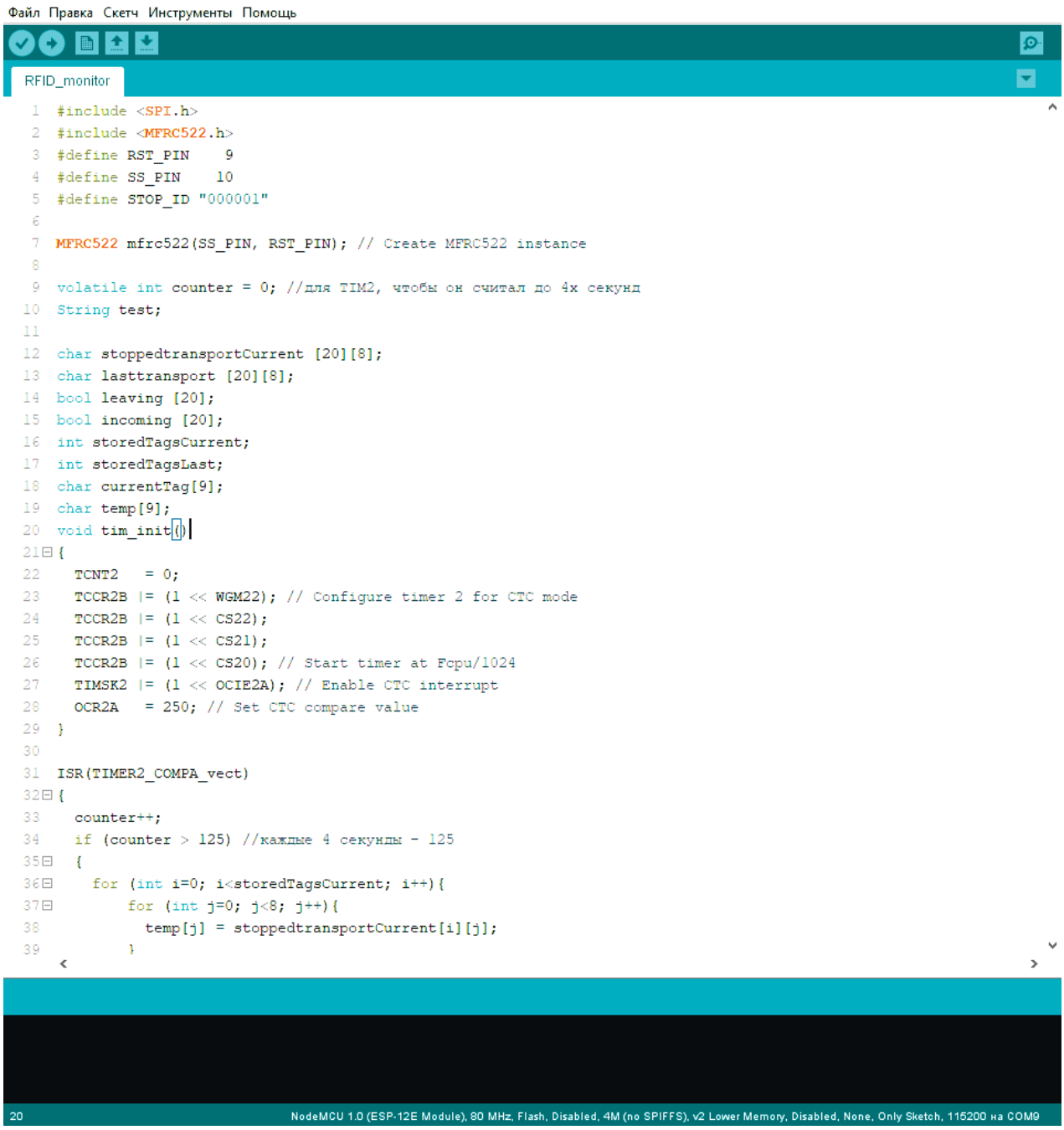


Рисунок 4.1.7 – окно программы Arduino IDE

Для работы со считывателем используется библиотека MFRC522. Чтобы реализовать спроектированный алгоритм понадобились функции PCD_Init, PICC_IsNewCardPresent, PICC_ReadCardSerial. PCD_Init инициализирует микроконтроллер считывателя RC522, приводит в исходное состояние служебные регисты, таймеры, утанавливает в активное состояние антенну. Функция PICC_IsNewCardPresent возвращает значение логической единицы при наличии активных карт в поле считывания. Функция PICC_ReadCardSerial

является оболочкой для функции выбора метки (при наличии нескольких меток в поле считывания) и получения уникального идентификатора и данных из памяти транспондера.

При написании кода для WiFi модуля использовались библиотеки "ESP8266WiFi" и "WiFiClient". Первая из перечисленных разработана на базе SDK ESP8266 и позволяет производить операции по подключению к точке доступа WiFi. Используется функция begin непосредственно для подключения, а проверка статуса подключения происходит с помощью функции status. Вторая библиотека необходима при подключении устройства как клиента к удалённому серверу. Используются функции connect для подключения, print для формирования строки запроса, stop для закрытия соединения.

Итогом реализации макета стало устройство, которое позволяет считывать уникальные идентификаторы меток и с интервалом времени отправлять данные на удалённый веб-сервер посредством беспроводного подключения к сети Интернет. На рисунке 4.1.8 изображен собранный макет.

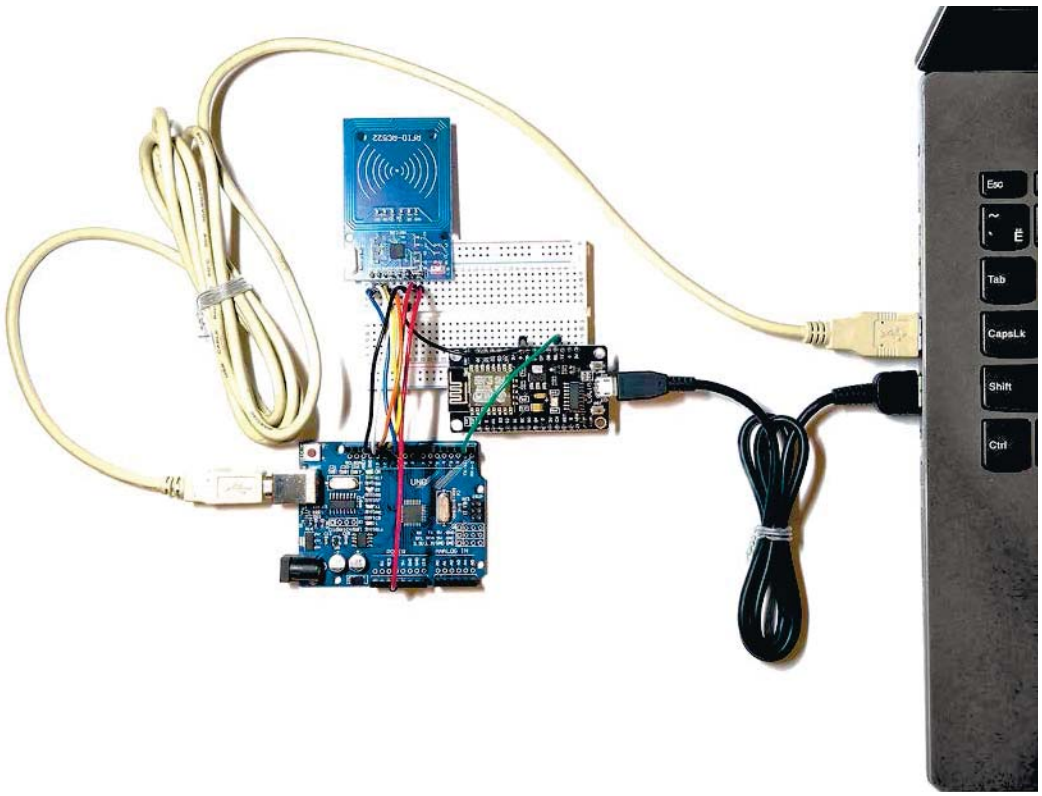


Рисунок 4.1.8 – фотография собранного макета

4.2 РЕАЛИЗАЦИЯ СЕРВЕРНОЙ ЧАСТИ

Для создания сервера на компьютере была создана директория Server, затем компоненты сервера Apache версии 2.4 были распакованы в директорию bin. В директории conf был отредактирован конфигурационный файл httpd.conf в соответствии с параметрами расположения сервера в файловой системе компьютера.

Затем произведена установка и настройка СУБД MySQL версии 8.0. Для этого в директорию bin были распакованы файлы MySQL из распространяемого разработчиками архива, и осуществлена настройка конфигурационного файла в соответствии с расположением. Затем произведена инициализация и установка из командной строки Windows. Итогом этого процесса является создание в каталоге сервера директории баз данных и прочих настроек.

Для установки PHP потребовалось добавить содержимое архива с версией 7.3.2 в директорию bin сервера. После добавления в конфигурационный файл httpd.conf информации о добавленном пакете перезапускается сервер.

Аналогичным образом устанавливается phpMyAdmin для управления базой данных посредством веб-интерфейса. На рисунке 4.2.1 представлен снимок экрана с созданной базой данных под названием «publictransport».

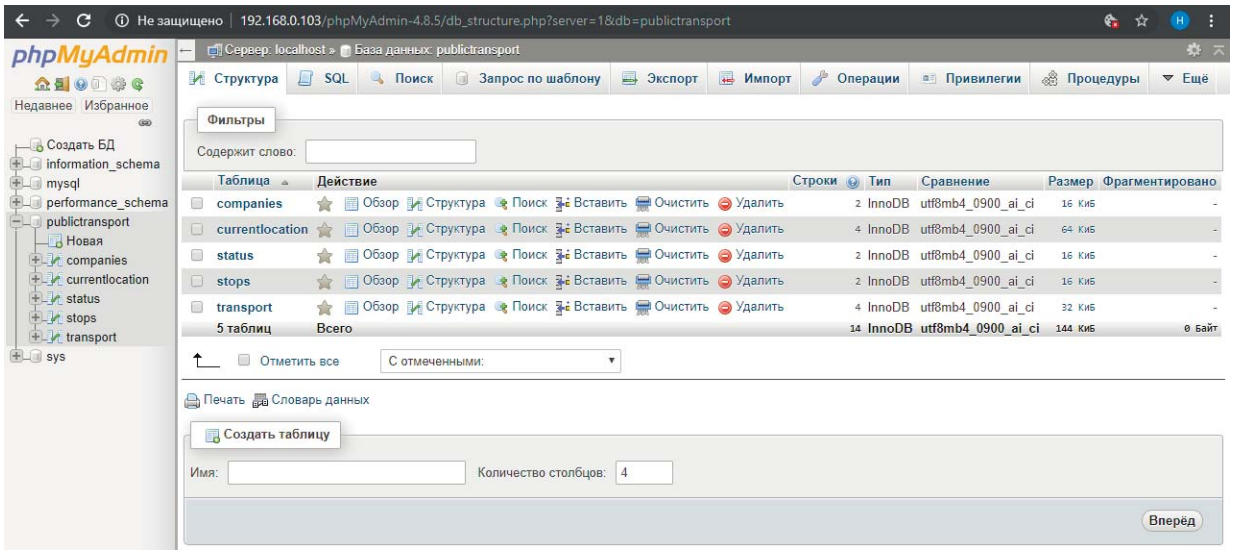


Рисунок 4.2.1 – интерфейс phpMyAdmin

Для того, чтобы принимать от клиентов (комплексов, установленных на остановках города) данные об уникальных идентификаторах меток на останавливающихся транспортных средствах необходимо написать php скрипт, который добавит (или обновит) данные о машине в таблице онлайн местоположения.

Для реализации онлайн монитогинга понадобится запрос вида: “server_address/ CurrentLocation.php?uid=value&stop=id_stop&status=id_status”. Формально это GET запрос с параметрами. Где параметры uid, stop и status это уникальный идентификатор метки, идентификаторы остановки и состояния машины соответственно.

Принцип работы таков, что сначала выполняется select-запрос с условием поиска машины с заданным UID в базе. Затем такой же запрос, но теперь с идентификатором машины из базы к таблице, в которой хранятся текущие местоположения машин, чтобы выяснить, была ли уже для неё какая-либо запись. Если была, то данные просто обновятся (операция update), в ином случае происходит вставка записи с информацией о машине (операция insert).

Для проверки корректности отправляемой информации и реализации функции онлайн просмотра местоположения машин был разработан второй php скрипт, показывающий таблицу текущего местоположения транспорта. Формат строки запроса следующий: “server_address/ShowAllUnitsOnline.php”. Этот скрипт просто применяет select-запрос к таблицам текущего местоположения, информации о транспортных единицах, остановках, состояниях и выводит её в табличном виде со следующими столбцами: модель, маршрут, действие, остановка, время.

Итогом реализации серверной части стал веб-сервер Apache версии 2.4 с СУБД MySQL версии 8.0 и поддержкой языка PHP 7.3.2. Написанные скрипты обеспечивают просмотр текущего местоположения машин и связь с аппаратным обеспечением на остановках посредством добавления или обновления в базе данных информации о местоположении транспорта.

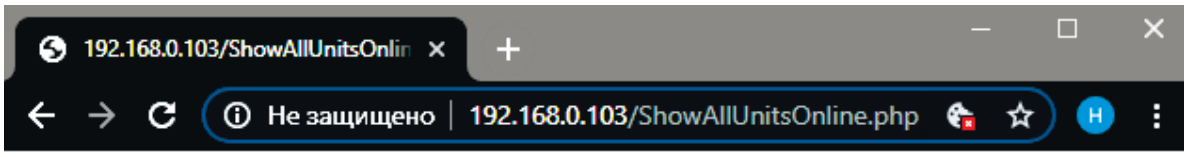
5 ТЕСТИРОВАНИЕ

5.1 МЕТОДОЛОГИЯ ТЕСТИРОВАНИЯ

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

5.2 ПРОВЕДЕНИЕ ПРОЦЕДУРЫ ТЕСТИРОВАНИЯ

Для проверки работы скриптов запросы сначала проводились в веб-браузере. На рисунке 5.1 показан результат выполнения запроса на отображение текущего местоположения транспорта.



Текущее местоположение транспорта

Модель	Маршрут	Действие	Остановка	Время
ПАЗ-3205	102	Уехал с остановки	ЮУрГУ	2019-05-21 19:27:10
ГАЗель NEXT	370	Уехал с остановки	ЮУрГУ	2019-05-21 19:21:15
Ford Transit	75	Уехал с остановки	ЮУрГУ	2019-05-21 21:30:04

Рисунок 5.1 – текущее местоположение транспорта

Можно заметить, что в таблице присутствуют по одной машине маршрутов с номерами 102, 370, 75. Проверка добавления новой для этой таблицы машины выполняется следующим запросом: “http://192.168.0.103/CurrentLocation.php?uid=03e14d1b&stop=2&status=1”. Здесь 03e14d1b – UID метки на машине с номером маршрута 58, 2 – идентификатор остановки «ПКиО им. Гагарина», 1 – идентификатор состояния «на остановке». На рисунке 5.2 видно, что выполнение запроса прошло успешно, и данные добавлены.

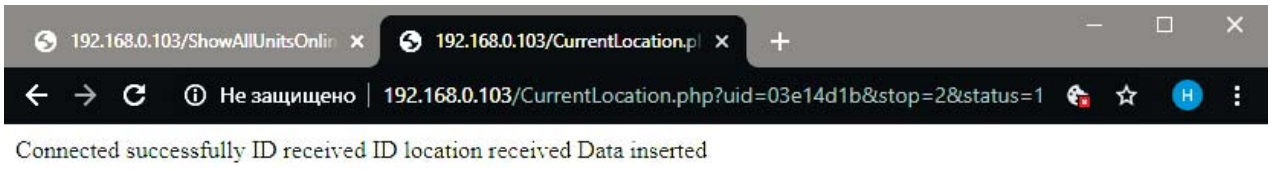


Рисунок 5.2 – выполнение добавления информации о местоположении новой машины

С помощью запроса на отображение местоположения проверяется факт добавления записи в таблицу. Проверка представлена на рисунке 5.3, на котором видно, что машина, обслуживающая 58-й маршрут стоит на остановке «ПКиО им. Гагарина» как и было задумано при составлении предыдущего запроса.

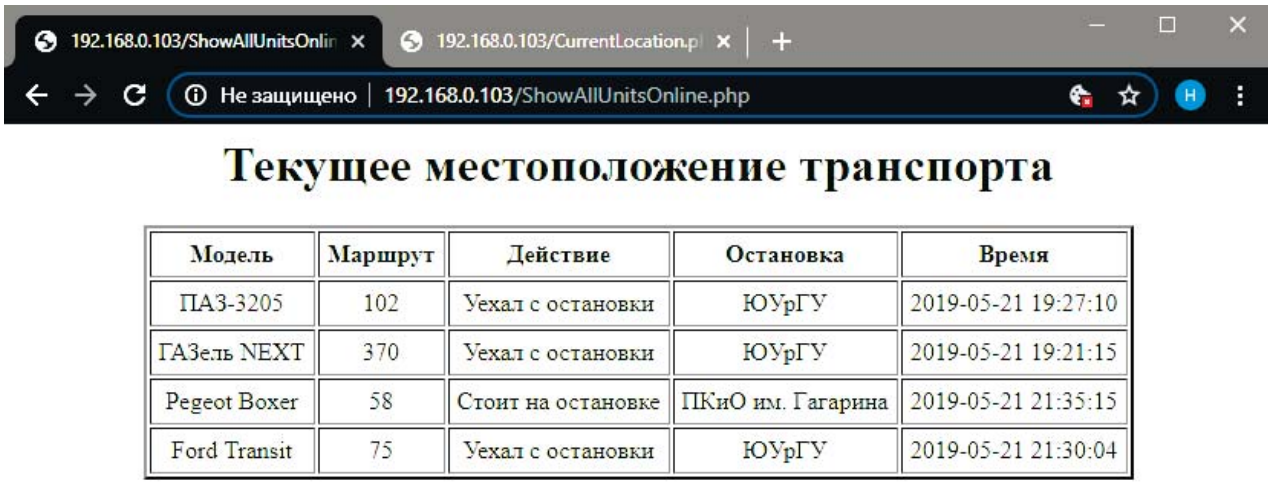


Рисунок 5.3 – проверка добавления машины

Затем данные об этой машине обновляются изменением состояния транспорта с «стоит на остановке» на «уехал с остановки». Это делает путём изменения параметра status в строке запроса с 1 на 2. На рисунке 5.4 видно, что данные именно обновлены, а не вставлены в таблицу, так как запись о транспортном средстве уже есть.

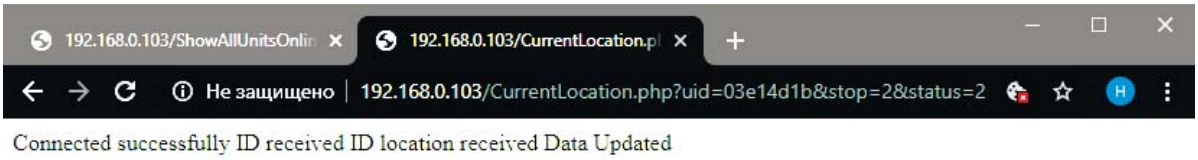
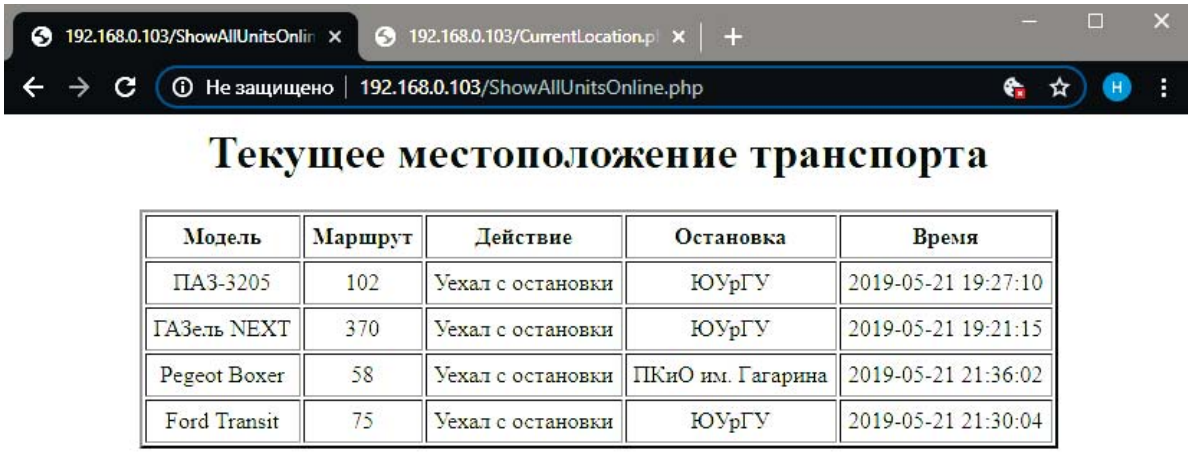


Рисунок 5.4 – обновление данных о машине

Проверка данного факта отображена на рисунке 5.5.



Модель	Маршрут	Действие	Остановка	Время
ПАЗ-3205	102	Уехал с остановки	ЮУрГУ	2019-05-21 19:27:10
ГАЗель NEXT	370	Уехал с остановки	ЮУрГУ	2019-05-21 19:21:15
Pegeot Boxer	58	Уехал с остановки	ПКиО им. Гагарина	2019-05-21 21:36:02
Ford Transit	75	Уехал с остановки	ЮУрГУ	2019-05-21 21:30:04

Рисунок 5.5 – проверка местоположения после обновления данных

Так как элементы системы устанавливаются в открытом пространстве, и в поле считывания могут попасть любые другие случайные метки ультравысокого диапазона частот, проверяется обработка метки с неизвестным системе идентификатором. На рисунке 5.6 снимок экрана из программы phpMyAdmin, где представлена таблица со всеми внесёнными идентификаторами машин в таблицу.

	id_unit	reg_number	model	route_number	rfid_uid	company_id
Изменить Копировать Удалить	1	с543тр174	Ford Transit	75	f2b6cb1a	1
Изменить Копировать Удалить	2	у254но174	Pegeot Boxer	58	03e14d1b	1
Изменить Копировать Удалить	3	н383ко174	ПАЗ-3205	102	5b04730d	2
Изменить Копировать Удалить	4	м389оу174	ГАЗель NEXT	370	cbcabc0d	2

Рисунок 5.6 – таблица транспортных единиц, находящихся в системе

Для проверки выполняется запрос на добавление метки с неизвестным системе идентификатором. Например, abcdef74 (см. рисунок 5.7).

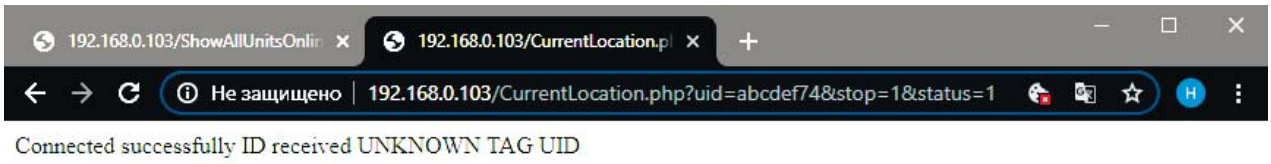


Рисунок 5.7 – добавление метки с несуществующим UID

Как видно из рисунка, подключение прошло успешно, параметр идентификатора считан в переменную, но сервер уведомляет о том, что он неизвестен системе. Это сообщение используется как справочная информация и игнорируется микроконтроллером на остановке, так как не представляет для него ценности. Однако, после обнаружения несоответствия сервер прекращает обработку данного запроса, и незнакомый UID не попадёт в базу данных.

Также проверяется ситуация сохранения целостности данных, когда может быть неправильно настроен микроконтроллер. Например, указан несуществующий идентификатор остановки, как в примере на рисунке 5.8. Для расшифровки ошибки вставки (обновления) данных предоставлено выделенное в конце сообщения пояснение.

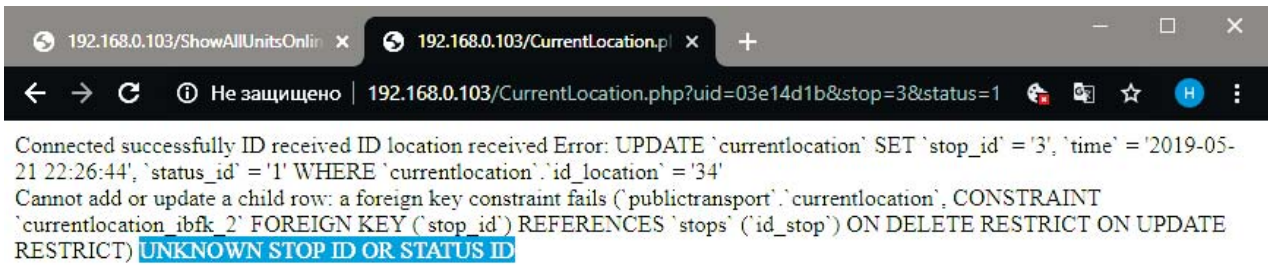


Рисунок 5.8 – проверка при некорректном идентификаторе остановки

После проверки работоспособности скриптов и корректной обработки поступающей информации необходимо проверить, какая информация проходит между элементами аппаратной части системы.

Например, с помощью встроенного в Arduino IDE монитора последовательного порта проверяется, правильного ли формата поступает

информация от микроконтроллера Atmega при попадании метки в зону считывания (см. рисунок 5.9).

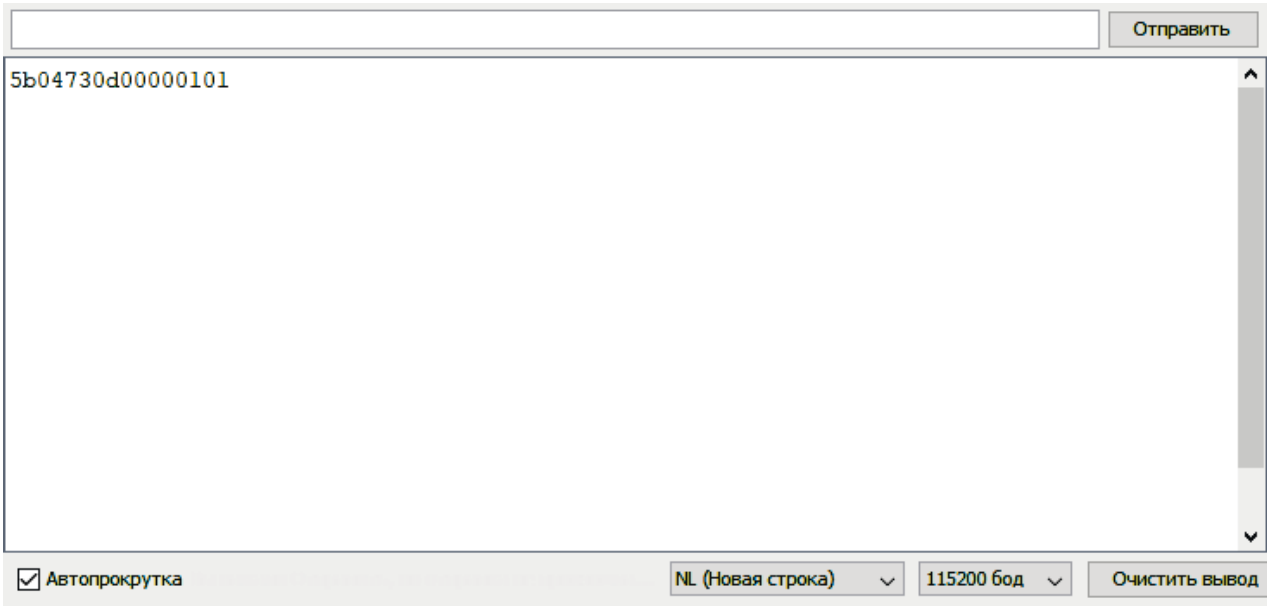


Рисунок 5.9 – мониторинг последовательного порта Arduino UNO

Параллельно проверяется вывод информации на веб сервере (см. рисунок 5.10)

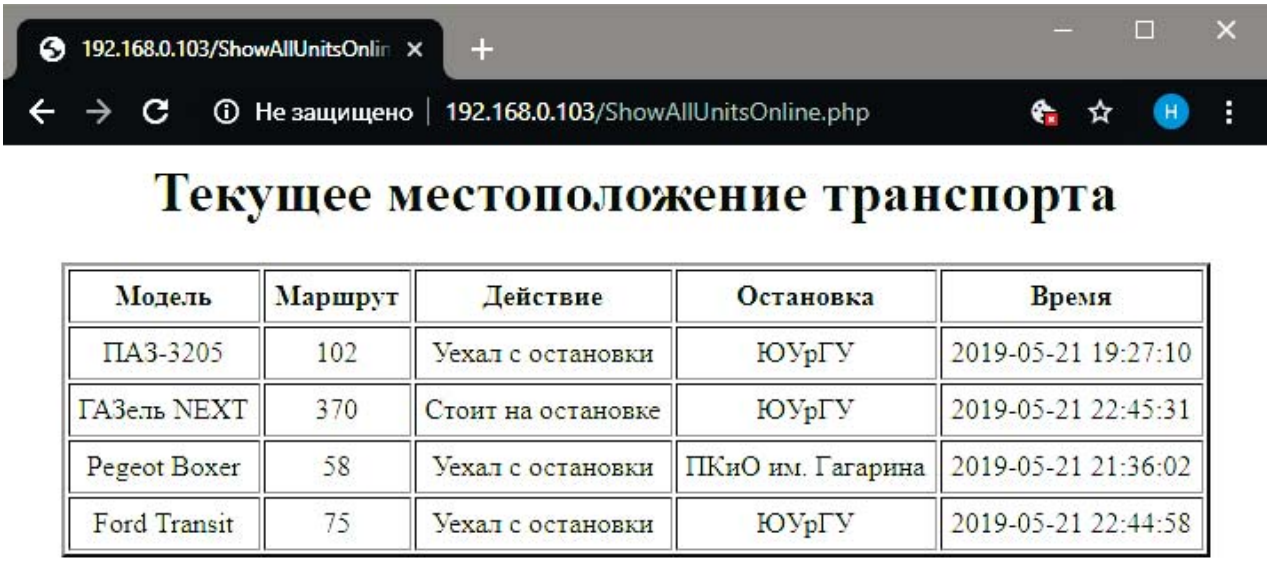


Рисунок 5.10 – проверка вывода информации от метки на веб сервере

Когда метка пропадает из зоны считывания (имитация убытия транспорта с остановки), это прослеживается в виде указания состояния с идентификатором 2 в конце выдаваемой строки (см. рисунок 5.11)

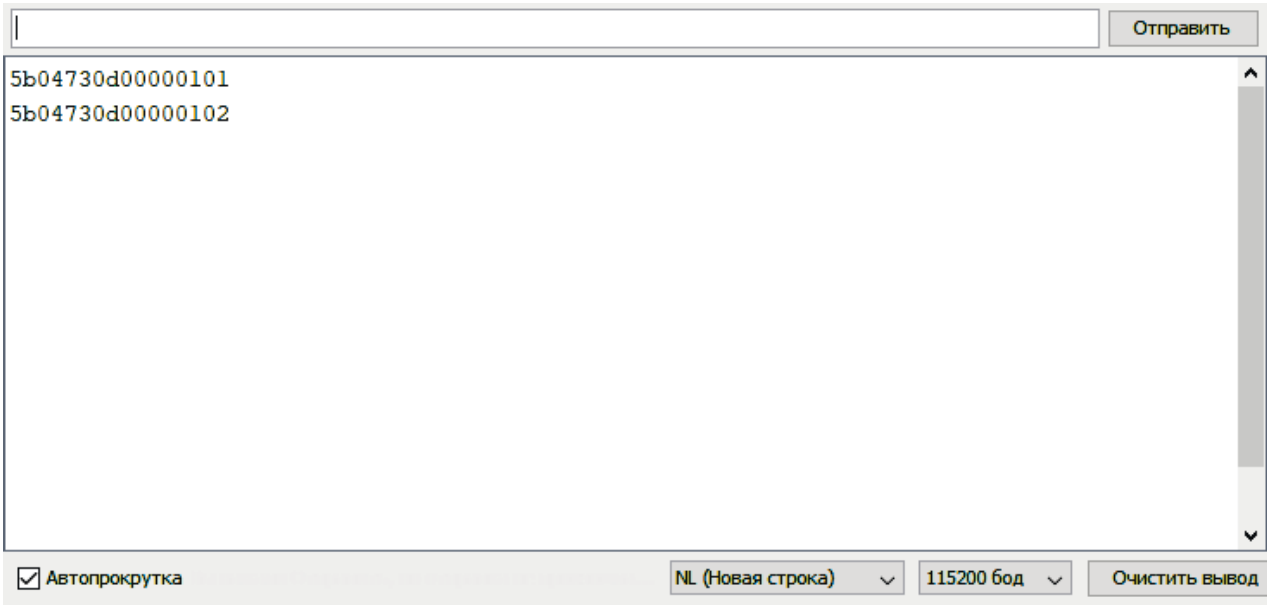


Рисунок 5.11 – проверка вывода данных при удалении метки из зоны считывания

На сервере отображается информация о том, что машина уехала с данной остановки (см. рисунок 5.12)

Текущее местоположение транспорта

Модель	Маршрут	Действие	Остановка	Время
ПАЗ-3205	102	Уехал с остановки	ЮУрГУ	2019-05-21 19:27:10
ГАЗель NEXT	370	Уехал с остановки	ЮУрГУ	2019-05-21 22:46:43
Pegeot Boxer	58	Уехал с остановки	ПКиО им. Гагарина	2019-05-21 21:36:02
Ford Transit	75	Уехал с остановки	ЮУрГУ	2019-05-21 22:44:58

Рисунок 5.12 – реакция сервера на удаление метки из зоны считывания

Проверяется так же работа, когда в области считывания находится несколько меток, например, две (см. рисунок 5.13).

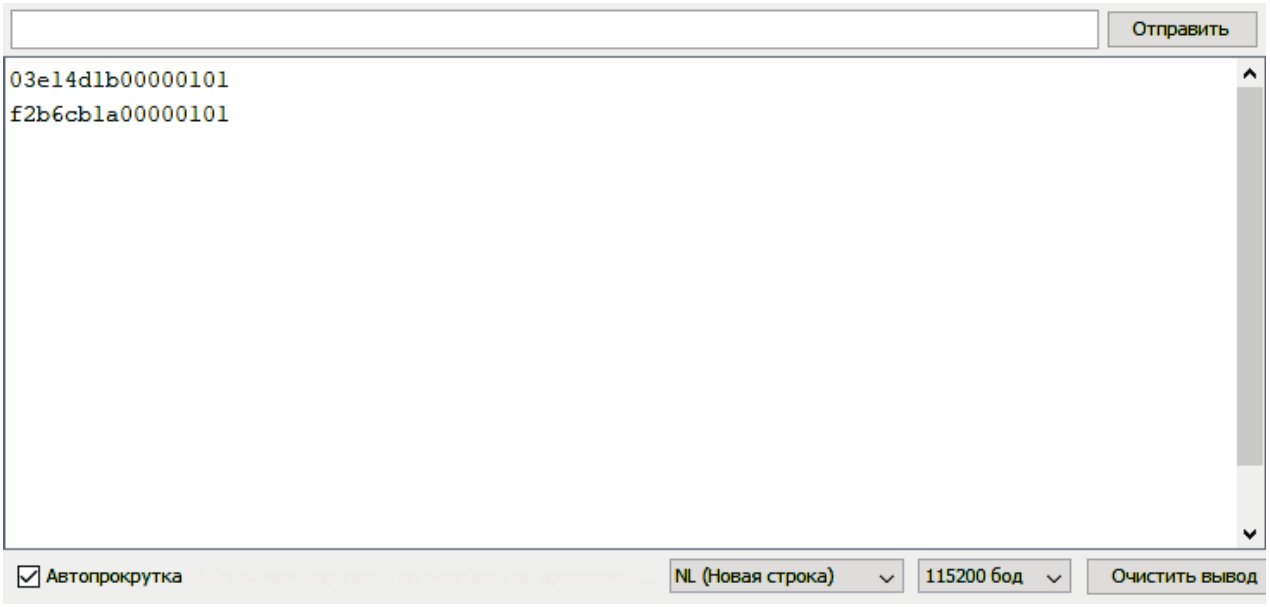
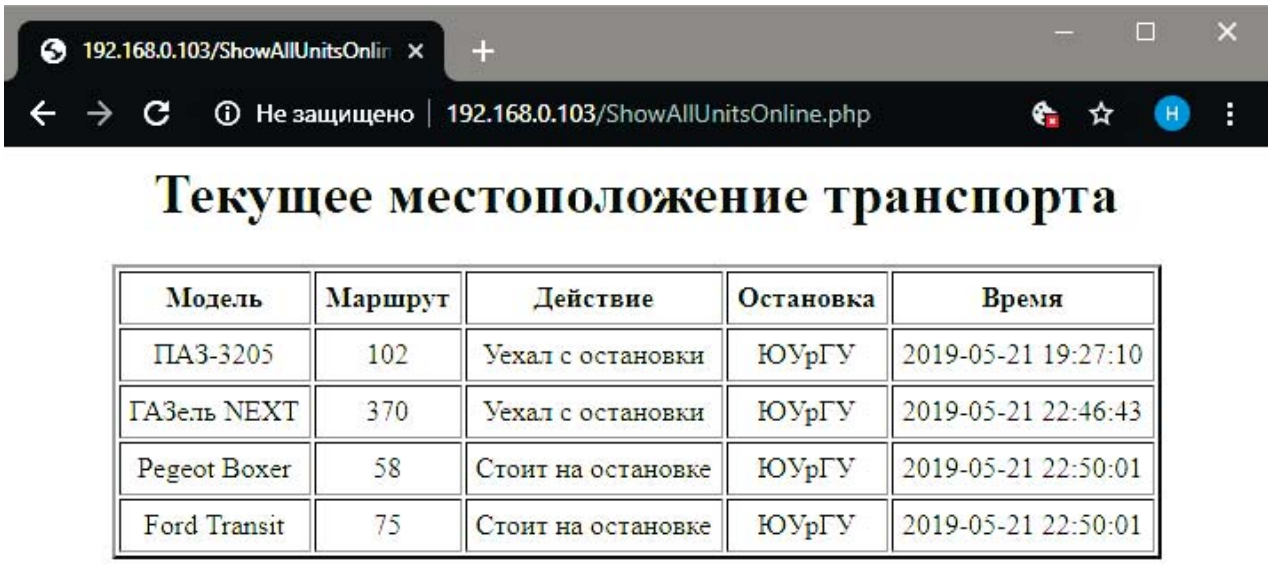


Рисунок 5.13 – вывод порта при вводе двух меток в зону считывания

По времени прибытия на сервере можно определить, что прибыло одновременно (с пренебрежимо малым интервалом времени) две метки (см. рисунок 5.14).



Текущее местоположение транспорта

Модель	Маршрут	Действие	Остановка	Время
ПАЗ-3205	102	Уехал с остановки	ЮУрГУ	2019-05-21 19:27:10
ГАЗель NEXT	370	Уехал с остановки	ЮУрГУ	2019-05-21 22:46:43
Pegeot Boxer	58	Стоит на остановке	ЮУрГУ	2019-05-21 22:50:01
Ford Transit	75	Стоит на остановке	ЮУрГУ	2019-05-21 22:50:01

Рисунок 5.14 – вывод информации на сервере при одновременном нахождении в зоне считывания двух меток

Таким же образом метки одновременно выводятся из области считывания, что можно определить по указанию состояния 2 в выводе последовательного порта (см. рисунок 5.15)

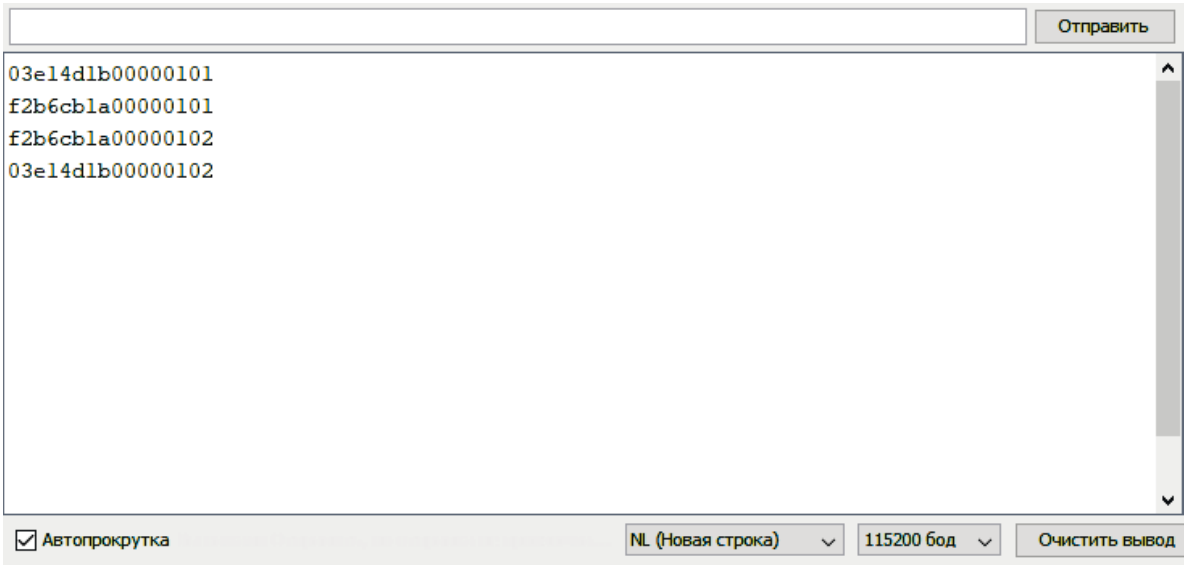


Рисунок 5.15 – одновременный вывод меток из зоны считывания

Аналогично вводу меток, при их удалении сервер отобразил одинаковое время убытия (см. рисунок 5.16).

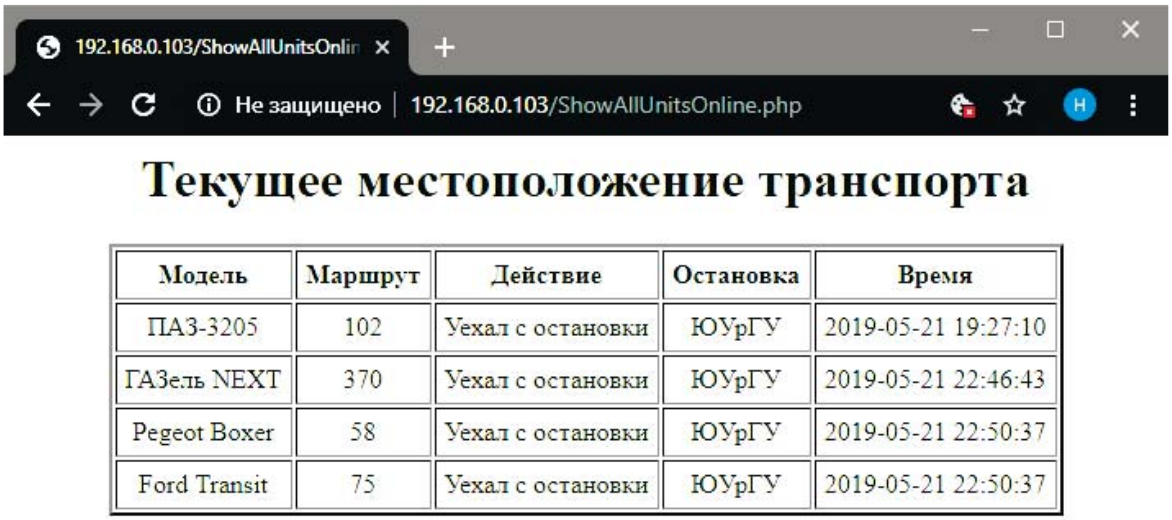


Рисунок 5.16 – показания в таблице при одновременном удалении двух меток из области считывания

В результате альфа-тестирования выяснилось, что система работает как предполагалось: считыватель получает информацию как от одной, так и от множества меток, данная информация поступает на устройство связи (плату с модулем WiFi), которое отправляет запрос на сервер. Веб-сервер, в свою очередь, принимает запросы и отклоняет заведомо ошибочные (с неправильными параметрами, либо вообще не связанными с системой данными).

6 ЗАКЛЮЧЕНИЕ

В выполненной работе была дана оценка роли общественного транспорта в современном городе, а также методам его мониторинга. В результате исследования и проектирования RFID системы выяснилось, что радиочастотная идентификация в ультравысоком диапазоне волн вполне может применяться в городской среде для контроля местоположения общественного транспорта на остановках города. Результат обзора существующих решений, обеспечивающих мониторинг по спутниковой навигации выявил, что они сложны в обслуживании в рамках большого количества единиц подвижного состава, а также требуют больших финансовых вложений по сравнению с разрабатываемой в данной работе системой.

В процессе проектирования и разработки аппаратной части системы возникла необходимость отказаться от стандартного алгоритма считывателя, при котором он однократно получает данные от метки, а затем отправляет её в состояние сна, после чего трудоёмко отследить время её выхода из зоны считывания. Для решения проблемы определения момента выхода транспондера из области действия антенн применялся алгоритм многократного считывания меток в области сканирования, накопления их во временных буфер, и отправки на сервер по строго заданным интервалам.

При выполнении работы одной из задач являлась реализация макета аппаратной части системы. Несмотря на то, что у макета рабочая частота (13,56 МГц) отличается от той, которая должна применяться в городских условиях (960 МГц), принципы работы не имеют больших различий. В качестве альтернативы пассивных меток ультравысокочастотного диапазона использовались карты и брелки Mifare.

Система прошла альфа-тестирование. Были проверены следующие функции: считывание уникального идентификатора метки и отправка на сервер, определение прибывшей машины на сервере и добавление данных о

текущем местоположении этой машины в таблицу базы данных. Реализован вывод информации о текущем местоположении транспорта.

Итогом выполнения работы является модель работоспособной системы, реализующей функции онлайн мониторинга транспорта на остановках.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Трубина Е.Г. Город в теории: опыты осмысления пространства – М.: Изд-во Новое литературное обозрение, 2011. – 520 с.

2 Тойменцева И.А. Стратегическое планирование транспортных услуг – М.: Изд-во Креативная экономика, 2009. – 160 с.

3 Веб-сайт «Первый областной». [Электронный ресурс]. URL: https://www.1obl.ru/news/o-lyudyakh/dolya-obshchestvennogo-transporta-v-chelyabinskikh-perevozkakh-dostignet-40-k-letu-2016-goda/. Дата обращения: 25.02.2019.

4 Веб-сайт «Доступ». [Электронный ресурс]. URL: https://dostup1.ru/society/K-2017-godu-dolyu-munitsipalnogo-transporta-v-Chelyabinske-uvelichat-do-50_82409.html. Дата обращения: 25.02.2019.

5 Соловьев Ю.А. Системы спутниковой навигации – М.: Изд-во Эко-Трендз, 2000. — 270 с.

6 Веб-сайт «Alarmstore». [Электронный ресурс]. URL: https://alarmstore.ru/catalog/poiskovo-ohrannye-sistemy/starline-m18. Дата обращения: 04.03.2019.

7 Сандип Лахири. RFID. Руководство по внедрению − М.: Изд-во Кудиц-Пресс, 2007. − 312 с.

8 Веб-сайт «GS1». [Электронный ресурс]. URL: https://www.gs1.org/epcglobal. Дата обращения: 15.02.2019.

9 ГОСТ Р ИСО/МЭК 18000-6. Идентификация радиочастотная для управления предметами. – М.: Изд-во Стандартинформ, 2014. – 13 с.

10 Веб-сайт «IntechOpen». [Электронный ресурс]. URL: https://www.intechopen.com/books/designing-and-deploying-rfid-applications/a-knowledge-based-approach-for-detecting-misuses-in-rfid-systems. Дата обращения: 11.03.2019.

11 Веб-сайт «RFID Center». [Электронный ресурс]. URL: http://rfidcenter.ru/blog/about_uhf_rfid_tag/. Дата обращения: 11.03.2019.

12 Веб-сайт «SparkFun». [Электронный ресурс]. URL: https://www.sparkfun.com/products/14066. Дата обращения: 15.03.2019.
13 Веб-сайт «IT проект». [Электронный ресурс]. URL: http://www.itproject.ru/oborudovanie_torgovlya_sklad/rfid_reader/rfid_schityvateli_stacionarnye/motorola/motorola_fx7400__rfid_. Дата обращения: 15.03.2019.
14 Веб-сайт «Go RFID». [Электронный ресурс]. URL: https://go-rfid.ru/novosti-i-statiy/novosti-oborudovaniya/princip-raboti-rfid-antenni. Дата обращения: 18.03.2019.
15 Веб-сайт «ReUnit». [Электронный ресурс]. URL: https://reunit.ru/id/uhf-rfid-antenna-clou-cl7205a-98.html. Дата обращения: 18.03.2019.
16 Веб-сайт «Habr». [Электронный ресурс]. URL: https://habr.com/ru/post/267721/. Дата обращения: 25.03.2019.
17 Веб-сайт «Записки Backend-разработчика». [Электронный ресурс]. URL: https://ekaterinagoltsova.github.io/posts/apache-vs-nginx/. Дата обращения: 25.03.2019.
18 Веб-сайт «Nginx». [Электронный ресурс]. URL: http://nginx.org/ru/. Дата обращения: 25.03.2019.
19 Веб-сайт «Apache». [Электронный ресурс]. URL: http://www.apache.org/foundation/. Дата обращения: 25.03.2019.
20 Веб-сайт «Индекс TIOBE». [Электронный ресурс]. URL: https://www.tiobe.com/tiobe-index/. Дата обращения: 25.03.2019.
21 Веб-сайт «PHP». [Электронный ресурс]. URL: https://www.php.net/. Дата обращения: 25.03.2019.
22 Веб-сайт «Сообщество специалистов Oracle Patches». [Электронный ресурс]. URL: https://oracle-patches.com/web/3652. Дата обращения: 29.03.2019.

23 Веб-сайт «Amazon Web Services». [Электронный ресурс]. URL: https://aws.amazon.com/ru/relational-database/. Дата обращения: 29.03.2019.
24 Веб-сайт «Devacademy». [Электронный ресурс]. URL: https://devacademy.ru/article/sqlite-vs-mysql-vs-postgresql/. Дата обращения: 29.03.2019.
25 Веб-сайт «phpMyAdmin». [Электронный ресурс]. URL: https://www.phpmyadmin.net/. Дата обращения: 30.03.2019.
26 Веб-сайт «Arduinoplus». [Электронный ресурс]. URL: https://arduinoplus.ru/arduino-vse-plati-sravnitelnaya-tablica/. Дата обращения: 17.04.2019.
27 Веб-сайт «Robocraft». [Электронный ресурс]. URL: http://robocraft.ru/blog/3004.html. Дата обращения: 17.04.2019.
28 Веб-сайт «Radioprog». [Электронный ресурс]. URL: https://radioprog.ru/shop/merch/59. Дата обращения: 17.04.2019.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД МИКРОКОНТРОЛЛЕРА ATMEGA

```
#include <SPI.h>
#include <MFRC522.h>
#define RST_PIN    9
#define SS_PIN    10
#define STOP_ID "000001"

MFRC522 mfrc522(SS_PIN, RST_PIN); //создание экземпляра mfrc522

volatile int counter = 0; //счетная перем. для TIM2, чтобы "считал" до 4х секунд
String tag;

char stoppedtransportCurrent [20][8];
char lasttransport [20][8];
bool leaving [20];
bool incoming [20];
int storedTagsCurrent;
int storedTagsLast;
char currentTag[9];
char temp[9];
void tim_init()
{
  TCNT2   = 0;
  TCCR2B |= (1 << WGM22); // конфигурация таймера 2 в режим сравнения
  TCCR2B |= (1 << CS22);
  TCCR2B |= (1 << CS21);
  TCCR2B |= (1 << CS20); // предделитель Fcpu/1024
  TIMSK2 |= (1 << OCIE2A); // разрешить прерывания по сравнению
  OCR2A   = 250; // установка регистра сравнения
}

ISR(TIMER2_COMPA_vect)
{
  counter++;
  if (counter > 125) //каждые 4 секунды обработчик попадает в эту область
  {
    for (int i=0; i<storedTagsCurrent; i++){
        for (int j=0; j<8; j++){
          temp[j] = stoppedtransportCurrent[i][j];
```

```
        }

        for (int l=0; l<storedTagsLast; l++){ //сравнивание массивов
          int sim = 0;
          for (int i=0; i<8; i++){
            if(temp[i] == lasttransport[l][i])
              sim++;
          }
          if(sim == 8){
            leaving[l] = true; //если в предыдущей итерации была такая метка, и она
есть сейчас
            incoming[i] = false; //метку нельзя назвать только что прибывшей
          }
        }
        if (incoming[i] == true){
          Serial.print(temp); //выводим прибывшие метки
          Serial.print(STOP_ID);
          Serial.println("01");
        }
      }
      for (int i=0; i<storedTagsLast; i++){
        for (int j=0; j<8; j++){
          temp[j] = lasttransport[i][j];
        }
        if (leaving[i] == false){ //оставшиеся от предыдущей итерации метки как
убывшие с остановки
          Serial.print(temp);
          Serial.print(STOP_ID);
          Serial.println("02");
        }
      }
      for (int i=0; i<20; i++){ //запоминаем данные этой итерации
        for (int j=0; j<8; j++){
          lasttransport[i][j] = stoppedtransportCurrent[i][j];
        }
        leaving[i] = false;
      }
      storedTagsLast = storedTagsCurrent;
    counter = 0;
```

```
    clearMass();
  }
}

String getTagUID(byte *buffer, byte bufferSize)
{
    String s;
    unsigned long uiddec = 0;
    unsigned long temp;
    char uid[8];
    for (byte m = (bufferSize > 4 ? (bufferSize - 4) : 0); m < bufferSize; m++) {
       //берем только последние 4 байта (UID) и переводим в десятичную систему
        unsigned long p = 1;
        for(int k = 0; k < bufferSize-m-1; k++) {
          p = p*256;
        }
        uiddec += p*buffer[m];
        s = s + (buffer[m] < 0x10 ? "0" : "");
        s = s + String(buffer[m], HEX);
    }
    s.toCharArray(uid, 8);
    return s;
}

void clearMass(){
  for (int i = 0; i<20; i++)
  {
    incoming[i] = true;
    for (int j=0; j<8; j++)
    {
      stoppedtransportCurrent[i][j] = '0';
    }
  }
  storedTagsCurrent = 0;
}

void clearMass2(){
  for (int i = 0; i<20; i++)
  {
```

```
    leaving[i] = false;
    for (int j=0; j<8; j++)
    {
      lasttransport[i][j] = '0';
    }
  }
  storedTagsLast = 0;
}
void setup() {
  clearMass();
  clearMass2();
  tim_init();
  Serial.begin(115200);   //иниц. последов. порта
  while (!Serial);        //ожидание готовности последовательного порта
  SPI.begin();
  sei();
  mfrc522.PCD_Init();
}

void loop() {
    if ( ! mfrc522.PICC_IsNewCardPresent()) { //false - когда нет карт
      return;
    }
    if ( ! mfrc522.PICC_ReadCardSerial()) { //выбор одной из считанных карт
      return;
    }
    tag = getTagUID(mfrc522.uid.uidByte, mfrc522.uid.size);
    tag.toCharArray(currentTag, 9);
    //в процессе между отправками данных
    bool isNew = true;
    for (int k = 0; k < storedTagsCurrent; k++){
      int similarity = 0;
      for (int i=0; i<8; i++){
        if (currentTag[i] != stoppedtransportCurrent[k][i]){
          break;
        }
        else{
          similarity++;
        }
```

```
      }
      if (similarity == 8)// проверяемая метка сходится с той, что в массиве
      { isNew = false; }
    }
    if (isNew){
      for (int i=0; i<8; i++){
        stoppedtransportCurrent[storedTagsCurrent][i] = currentTag[i];
      }
      storedTagsCurrent++;
    }
}
```

ПРИЛОЖЕНИЕ Б
ИСХОДНЫЙ КОД МИКРОКОНТРОЛЛЕРА ESP

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#define SERVER_ADDR "192.168.0.103"

const char* ssid     = "dlink";
const char* password = "00123638300";

WiFiClient client;

String receivedStr;
int messageLenght;
char temp[16];

void setup() {
  Serial.begin(115200);
  while (!Serial){
    //do nothing
  }
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.print("Connected to ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
}

void loop() {
  delay(100);
  if (Serial.available()){
    receivedStr = Serial.readString();
    //Serial.write(val);
    messageLenght = receivedStr.length();
    int i = 0;
    while (i<messageLenght){    //деление сообщения на метки
```

```
      for (int k=0; k<16; k++){
        temp[k] = receivedStr[k+i];
      }
      Serial.println(temp);
      i+=18;

      //здесь отправка каждого UID
      // CurrentLocation.php?uid=xxxxxxxx&stop=xxxx&status=xx
      if (client.connect(SERVER_ADDR, 80)) {
        client.print("GET /CurrentLocation.php?");
        client.print("uid=");
        for (int i=0; i<8; i++){
          client.print(temp[i]);
        }
        client.print("&stop=");
        for (int i=8; i<14; i++){
          client.print(temp[i]);
        }
        client.print("&status=");
        for (int i=14; i<16; i++){
          client.print(temp[i]);
        }
        client.println(" HTTP/1.1");
        client.print("Host: ");
        client.println(SERVER_ADDR);
        client.println("Connection: close");
        client.println();
        client.println();
        client.stop();
      }
      else {Serial.println("--> connection failed\n");}
    }
  }
}
```

ПРИЛОЖЕНИЕ В
PHP СКРИПТ ДОБАВЛЕНИЯ ИНФОРМАЦИИ О МЕСТОПОЛОЖЕНИИ

```
<?php
date_default_timezone_set("Asia/Yekaterinburg");

$servername = "localhost";
$username = "root";
$password = "";
$dbname = "publictransport";

$date = date('Y-m-d H:i:s');
$uid = $_GET['uid'];
$stop = $_GET['stop'];
$status = $_GET['status'];
//$stop = 1;
//$status = 2;

//создание соединения
$conn = new mysqli($servername, $username, $password, $dbname);

//проверка соединения
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully\r\n";

$sql1 = "SELECT id_unit FROM `transport` WHERE rfid_uid = '$uid'";
$result = $conn->query($sql1);

if ($conn->query($sql1) == TRUE) {
    echo "ID received\r\n";
} else {
    echo "Error: " . $sql1 . "<br>" . $conn->error;
}
if ($result->num_rows > 0) {
    while($row = $result->fetch_assoc()) {
              $unitid = $row["id_unit"];
              }
```

```
} else {
    die("UNKNOWN TAG UID");//echo "0 results";
}

$sql3 = "SELECT id_location FROM `currentlocation` WHERE unit_id = '$unitid'";
$result = $conn->query($sql3);
if ($conn->query($sql3) == TRUE) {
    echo "ID location received\r\n";
} else {
    echo "Error: " . $sql3 . "<br>" . $conn->error;
}
if ($result->num_rows > 0) {
    while($row = $result->fetch_assoc()) {
             $idlocation = $row["id_location"];
             $sql4 = "UPDATE `currentlocation` SET `stop_id` = '$stop', `time` =
'$date', `status_id` = '$status' WHERE `currentlocation`.`id_location` =
'$idlocation'";
             if ($conn->query($sql4) == TRUE) {
                    echo "Data Updated\r\n";
             } else {
                    echo "Error: " . $sql4 . "<br>" . $conn->error;
                    echo " UNKNOWN STOP ID OR STATUS ID";
             }
             }
} else {
      $sql2 = "INSERT INTO `currentlocation` (`id_location`, `unit_id`, `stop_id`,
`time`, `status_id`) VALUES (NULL, '$unitid', '$stop', '$date', '$status')";

      if ($conn->query($sql2) == TRUE) {
             echo "Data inserted\r\n";
      } else {
             echo "Error: " . $sql2 . "<br>" . $conn->error;
             echo " UNKNOWN STOP ID OR STATUS ID";
      }
}

$conn->close();
?>
```

ПРИЛОЖЕНИЕ Г
PHP СКРИПТ ОТОБРАЖЕНИЯ ИНФОРМАЦИИ О МЕСТОПОЛОЖЕНИИ

```
<?php
//SELECT model, route_number, explanation, name, time  FROM currentlocation,
transport, stops, status WHERE transport.id_unit = currentlocation.unit_id &&
stops.id_stop = currentlocation.stop_id && status.id_status =
currentlocation.status_id
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "publictransport";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: \n" . $conn->connect_error);
}

$sql = "SELECT model, route_number, explanation, name, time  FROM currentlocation,
transport, stops, status WHERE transport.id_unit = currentlocation.unit_id &&
stops.id_stop = currentlocation.stop_id && status.id_status =
currentlocation.status_id ORDER BY route_number";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    // output data of each row
       echo "<h1 align = \"center\"><b>Текущее местоположение транспорта</b></h1>";
       echo "<table border=\"2\" cellpadding=\"5\" align =
\"center\"><tr><th>Модель</th><th>Маршрут</th><th>Действие</th><th>Остановка</th><th>В
ремя</th></tr>";
    while($row = $result->fetch_assoc()) {
             echo "<tr>";
             echo   "<td align = \"center\">" . $row["model"] . "</td>";
             echo   "<td align = \"center\">" . $row["route_number"] . "</td>";
             echo   "<td align = \"center\">" . $row["explanation"] . "</td>";
             echo   "<td align = \"center\">" . $row["name"] . "</td>";
```

```
            echo   "<td align = \"center\">" . $row["time"] . "</td>";
           //for ($j = 0 ; $j < 3 ; ++$j) echo "<td>$row[$j]</td>";
        echo "</tr>";
    }
} else {
    echo "0 results";
}
$conn->close();
?>
```