



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Информационные системы и телекоммуникации»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
НА ТЕМУ:

«Разработка устройства для автоматического тестирования
платежных терминалов»

Студент группы ИУ3-82

(Подпись, дата)

С. А. Шепелев

Руководитель ВКР

(Подпись, дата)

И. М. Сидякин

Нормоконтролер

(Подпись, дата)

...

2020 г.

АННОТАЦИЯ

Расчетно-пояснительная записка 95 с., 11 рис., 1 табл., 11 источников.

POS-ТЕРМИНАЛ, ARDUINO, ПРОГРАММНЫЙ ИНТЕРФЕЙС
БАНКОВСКОГО ПРИЛОЖЕНИЯ.

Объектом разработки является устройство автоматического тестирования платежных терминалов.

Цель работы – устройство автоматического тестирования платежных терминалов, предназначенное для разработчиков банковских программ.

Полученная цель достигается за счет использования современной методологии проектирования ПО, технологий разработки программного обеспечения.

СОДЕРЖАНИЕ

АННОТАЦИЯ	5
ВВЕДЕНИЕ.....	7
1 Исследовательская часть	9
1.1 Предметная область	9
1.1.1 Общая информация об электронных платежах	9
1.1.2 Кредитные карты.....	16
1.1.3 POS – терминалы.....	25
1.2 Arduino	27
2 Конструкторская часть	40
2.1 Выбор конструкции	40
2.2 Программный интерфейс банковского приложения JPAУ	47
2.2.1 Протокол обмена данными с банковским приложением.....	47
2.2.2 Формат данных и ограничения интерфейса.....	48
2.2.3 Команды	48
2.3 Разработка архитектуры системы	79
2.4 Выбор необходимых инструментов программирования	82
2.4.1 Выбор языка программирования и фреймворка.....	82
2.4.2 Выбор среды разработки	83
2.5 Выбор серверной платформы	84
3 Технологическая часть	87
3.1 Разработка программной части	87
3.2 Выводы.....	91
ЗАКЛЮЧЕНИЕ	92
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	93
ПРИЛОЖЕНИЕ А Графическая часть выпускной квалификационной работы.....	95

ВВЕДЕНИЕ

Пластиковые карты были впервые введены в США в 1920-х годах. Ранние пластиковые карты были предназначены для использования исключительно в определенных магазинах или местах. American Express первым запустил пластиковую карту, как мы ее знаем сегодня.

Платежи по пластиковым картам стали более широко распространенными в течение многих лет, и сегодня некоторые города и даже страны стремятся стать безбумажными - поэтому полагаются только на платежи с помощью карты, мобильного телефона или виртуальные платежи вместо наличных.

Платежи по карте можно использовать в любом месте с соответствующей настройкой. Для личных карточных платежей требуется устройство для чтения карт, которое по сути «считывает» информацию, содержащуюся в магнитной полосе или микрочипе карты.

Это устройство называется платежный терминал, в настоящее время оно используется повсеместно. Создание и тестирование таких устройств является трудоемкой задачей. Компании необходимо протестировать свой терминал для его сертификации. В большинстве случаев тестирование осуществляется вручную, что накладывает большие временные издержки.

Актуальность. В связи с этим возникает необходимость в разработке устройства, которое способно освободить разработчиков от рутинных действий для тестирования платежных терминалов и автоматизировать тестирование в целом.

Цель и задачи. Цель работы – устройство автоматического тестирования платежных терминалов, предназначенное для разработчиков банковских программ.

Задачи для достижения поставленной цели:

- 1) Ознакомление с общей структурой платежных терминалов и электронных платежей.
- 2) Изучение программного интерфейса банковского приложения JPAУ.
- 3) Разработка архитектуры системы.
- 4) Выбор инструментов и технологий для разработки системы.
- 5) Разработка и тестирование системы

1 Исследовательская часть

1.1 Предметная область

1.1.1 Общая информация об электронных платежах

Когда дело доходит до вариантов оплаты, нет ничего более удобного, чем электронный платеж. Вам не нужно выписывать чек, смахивать кредитную карту или обрабатывать бумажные деньги; все, что вам нужно сделать, это ввести некоторую информацию в ваш веб-браузер и щелкнуть мышью. Неудивительно, что все больше и больше людей обращаются к электронным платежам - или электронным платежам - в качестве альтернативы отправке чеков по почте.

Электронный платеж - это любой безналичный платеж, который не требует бумажного чека. Методы электронных платежей включают кредитные карты, дебетовые карты и сеть ACH (Automated Clearing House). Система ACH включает в себя прямой депозит, прямой дебет и электронные чеки (электронные чеки).

Для всех этих способов электронных платежей существует три основных типа транзакций:

- 1) Единовременный платеж от поставщика к поставщику обычно используется при совершении покупок в Интернете на сайте электронной коммерции, например, Amazon. Вы нажимаете на значок корзины покупок, вводите данные своей кредитной карты и нажимаете кнопку «Оформить заказ». Сайт обрабатывает информацию о вашей кредитной карте и отправляет вам электронное письмо с уведомлением о получении вашего платежа. На некоторых веб-сайтах вы можете использовать электронный чек вместо кредитной карты. Чтобы оплатить с помощью электронного чека, вы вводите номер своего счета

и номер маршрутизации вашего банка. Продавец разрешает оплату через банк клиента, который затем инициирует электронный перевод средств (EFT) или распечатывает чек и отправляет его поставщику по почте.

- 2) Вы делаете регулярный платеж от клиента к поставщику, когда оплачиваете счет путем регулярного прямого списания средств со своего текущего счета или автоматического списания средств с вашей кредитной карты. Этот тип плана оплаты обычно предлагается страховыми компаниями, телефонными компаниями и компаниями по управлению кредитами. Некоторые долгосрочные контракты (например, в спортзалах или фитнес-центрах) требуют такого рода автоматических графиков платежей.
- 3) Чтобы использовать автоматическую оплату от банка к поставщику, ваш банк должен предложить услугу, которая называется онлайн оплата счетов. Вы заходите на веб-сайт своего банка, вводите информацию о поставщике и уполномочиваете свой банк переводить деньги со своего счета в электронном виде для оплаты счета. В большинстве случаев вы можете выбрать, делать ли это вручную для каждого цикла выставления счетов, или счета будут автоматически оплачиваться в один и тот же день каждого месяца.

Электронный платеж очень удобен для потребителя. В большинстве случаев вам необходимо вводить информацию о вашей учетной записи - например, номер вашей кредитной карты и адрес доставки - один раз. Затем информация сохраняется в базе данных на веб-сервере продавца. Когда вы возвращаетесь на веб-сайт, вы просто входите под своим именем пользователя и паролем. Завершить транзакцию так же просто, как щелкнуть мышью: все, что вам нужно сделать, это подтвердить свою покупку, и все готово.

Электронный платеж снижает расходы для бизнеса. Чем больше платежей они могут обрабатывать в электронном виде, тем меньше они тратят на бумагу и почтовые расходы. Предложение электронных платежей также может помочь компаниям улучшить удержание клиентов. Клиент, скорее всего, вернется на тот же сайт электронной коммерции, где его или ее информация уже была введена и сохранена.

При всех преимуществах электронных платежей неудивительно, что их использование растет. В 2004 году было произведено более 12 миллиардов платежей АСН, что на 20 процентов больше, чем в 2003 году. В исследовании платежей Федеральной резервной системы за 2004 год отмечалось, что с 2000 по 2003 год электронные платежи росли по мере сокращения платежей по чекам, что свидетельствует о том, что электронные платежи заменяют чеки.

Чтобы лучше обслуживать своих клиентов, банки стремительно предлагают услуги по оплате счетов через Интернет. Опрос руководителей банков, проведенный Грантом Торнтоном в 2005 году, показал, что 65 процентов общинных банков и 94 процента крупных банков предлагают оплату счетов в режиме онлайн 24/7. Большинство из этих услуг являются бесплатными для участников и легко координируются с персональными программами, такими как Quicken или MS Money. В качестве альтернативы, потребители могут подписаться на услуги онлайн-оплаты счетов, такие как Paytrust или Yahoo! Оплата счета. Эти услуги взимают ежемесячную плату в обмен на удобство безбумажной оплаты счетов.

Основными недостатками электронных платежей являются опасения за конфиденциальность и возможность кражи личных данных. К счастью, существует множество способов защиты вашей конфиденциальной личной информации от попадания в чужие руки.

Вы можете защитить себя от кражи личных данных, используя программное обеспечение для защиты от вирусов и брандмауэр на своем компьютере. Вам также следует убедиться, что вы отправляете информацию о своей кредитной карте через защищенный сервер. Ваш интернет-браузер будет уведомлять вас о безопасности сервера, показывая значок замка или ключа. Кроме того, URL-адрес защищенного сайта обычно обозначается префиксом «https» вместо «http». Розничные продавцы вносят свой вклад, используя шифрование данных, которое кодирует вашу информацию таким образом, что только владелец ключа может ее расшифровать.

Помимо вопросов конфиденциальности, некоторые люди просто не любят делать электронные платежи. Они находят настройку слишком трудоемкой и не хотят запоминать больше логинов и паролей. Другие просто предпочитают знакомство с написанием чеков и отправкой конвертов по почте. Независимо от этих опасений, электронные платежи, вероятно, будут продолжать расти в популярности.

Допустим, у вас есть небольшой бизнес, и вы хотите настроить онлайн-платежи через свой веб-сайт. Ваше первое решение заключается в том, стоит ли отдавать ваше платежное решение на аутсорсинг или обрабатывать его самостоятельно. Для тех, кто хочет комплексное решение, такие сервисы, как PayPal и ProPay, позволяют легко принимать кредитные карты и другие формы электронных платежей с вашего сайта. Когда клиент вводит свою информацию на вашем сайте, ваша платежная служба авторизует транзакцию и переводит средства на ваш счет. Эти услуги взимают плату за обработку транзакции.

Если вы предпочитаете обрабатывать платежи внутри компании, первое, что вам нужно сделать, это настроить защищенный сервер. Это компьютер, использующий шифрование, чтобы злоумышленникам было трудно перехватить конфиденциальную информацию. Технология Secure Socket Layer (SSL)

используется для шифрования данных. Вы можете подать заявку на SSL-сертификат онлайн.

Получив сертификат SSL, вам необходимо зарегистрировать свой сайт в службе цифровой аутентификации. Цифровой сертификат подтверждает, что сайт, получающий информацию ваших клиентов, является правильным. Это гарантирует клиентам, что ваш сайт является законным и что их информация зашифрована.

Теперь, когда у вас есть защищенный сервер, вам нужно будет создать или купить программное обеспечение для покупок, которое позволит покупателю выбирать товары с вашего сайта и добавлять их в виртуальную корзину. Когда клиенты готовы выполнить свои заказы, они нажимают на ссылку «оформить заказ», которая приводит их на ваш защищенный сервер, где они вводят данные своей кредитной карты.

Наконец, вам нужна система для обработки платежей по кредитным картам и счет интернет-продавца в банке. Услуги по обработке платежей по кредитным картам доступны через интернет-компании, такие как Verisign. Они предоставляют вам программное обеспечение, которое проверяет данные кредитной карты вашего клиента на вашем защищенном сервере. Некоторые предприятия также принимают электронные чеки от клиентов.

Другим потенциальным источником информации является Национальная ассоциация автоматизированного клиринга (NACHA), также известная как Ассоциация электронных платежей. Теперь давайте посмотрим, что делает эта группа и какую помощь она предлагает потребителям и малому бизнесу.

Национальная ассоциация автоматизированных клиринговых центров (NACHA), также известная как Ассоциация электронных платежей, помогла расширить использование электронных платежей и электронных чеков. NACHA

управляет общенациональной сетью автоматизированных клиринговых центров (АСН). Через эту сеть 11 000 банков-членов НАСНА и других финансовых учреждений предлагают прямой депозит, прямой дебет и электронные чеки для потребителей и предприятий.

Эта активность довольно незаметна, когда вы проверяете остатки на банковских счетах в Интернете, совершаете покупки в интернет-магазинах с помощью дебетовой карты или оплачиваете счета с веб-сайта вашего банка. Но роль НАСНА важна. Некоммерческая ассоциация разрабатывает правила работы и методы ведения бизнеса для сети АСН, чтобы обеспечить ее эффективность, надежность и безопасность, сохраняя таким образом и ваши электронные платежи.

НАСНА также предлагает инструменты и ресурсы, чтобы помочь своим организациям-членам упростить электронные платежи. Кроме того, ассоциация разрабатывает методы электронных платежей за пределами сети АСН для таких областей, как интернет-коммерция, обмен электронными финансовыми данными (EDI) и международные платежи.

В качестве одной из услуг НАСНА отслеживает рост использования электронных платежей в квартальных и годовых отчетах. Например, сеть АСН обработала почти 16 миллиардов платежей на общую сумму 30,3 триллиона долларов в 2006 году, что на 14,5 процента больше, чем в 2005 году, согласно статистике НАСНА. Это включает в себя прямые платежи по заработной плате, пособия по социальному обеспечению, возврат налогов, оплату 8 миллиардов счетов потребителей и многое другое. Показатель показывает, что объем электронных платежей продолжает удваиваться каждые пять лет.

Хотя большинство предложений NACHA предназначены для финансовых учреждений-членов, ассоциация предлагает помощь для потребителей и малых предприятий через интерактивный веб-сайт.

На веб-сайте вы можете посмотреть виртуальные демонстрации работы прямого депозита, прямой оплаты и проверки чеков. Вы также найдете объяснения различных типов электронных платежей, а также информацию о том, как решить, являются ли для вас подходящим вариантом прямой депозит и прямая оплата счетов.

Бизнес-раздел веб-сайта содержит анализ затрат и выгод прямого депозита и прямых платежей для компаний разных размеров, наборы маркетинговых инструментов для сотрудников и клиентов предприятий, а также предлагает ответы на вопросы клиентов о конвертации чеков.

В рамках своей инициативы Pay It Green NACHA поощряет потребителей получать и оплачивать счета в электронном виде, а не на бумаге, чтобы сэкономить деревья, топливо и воду. Альянс объединяет NACHA, Федеральный резерв США и лидеров финансовой и потребительской индустрии выставления счетов.

В этой инициативе приводятся данные опроса Javelin Strategy and Research, проведенного в 2007 году, согласно которому, если бы все домохозяйства США получали и оплачивали свои счета в электронном виде, Соединенные Штаты:

- Экономят 16,5 миллионов деревьев каждый год, обеспечивая достаточное количество пиломатериалов для 216 054 домов на одну семью.
- Сократить количество токсичных загрязнителей воздуха на 3,9 миллиарда тонн эквивалента углекислого газа, что эквивалентно изъятию 355 015 автомобилей из дороги.

- Сократить на 1,6 миллиарда фунтов твердых отходов, образующихся каждый год, на 1,6 миллиарда фунтов, вес 56 000 полностью загруженных мусоровозов.

1.1.2 Кредитные карты

Кредитная карта - это тонкая пластиковая карта, обычно размером 3-1 / 8 дюймов на 2-1 / 8 дюймов, которая содержит идентификационную информацию, такую ​​как подпись или изображение, и уполномочивает лицо, указанное на ней, взимать плату за покупки или услуги с на его счету - расходы, за которые ему будут периодически выставляться счета. Сегодня информация на карте читается банкоматами, банкоматами, банковскими и интернет-компьютерами.

Согласно Encyclopedia Britannica, использование кредитных карт возникло в Соединенных Штатах в 1920-х годах, когда отдельные компании, такие как гостиничные сети и нефтяные компании, начали выдавать их клиентам для покупок, совершаемых на этих предприятиях. Это использование значительно возросло после Второй мировой войны.

Первая универсальная кредитная карта - та, которая могла использоваться в различных магазинах и предприятиях - была представлена ​​Diners Club, Inc. в 1950 году. С помощью этой системы компания, выпускающая кредитные карты, взимала с держателей карт ежегодную плату и выставляла им счета. ежемесячно или ежегодно. Еще одна важная универсальная карта - «Не выходи из дома без нее!» - была основана в 1958 году компанией American Express.

Позже появилась банковская система кредитных карт. В соответствии с этим планом банк зачисляет средства на счет продавца при получении квитанций о продаже (это означает, что продавцам платят быстро - то, что им нравится!) И собирает платежи, которые должны быть выставлены на счет держателя карты в

конце расчетного периода. Владелец карты, в свою очередь, выплачивает банку либо весь остаток, либо ежемесячными платежами с процентами (иногда называемыми текущими расходами) [1].

Первым планом национального банка был BankAmericard, который был начат на всей территории штата в 1959 году Банком Америки в Калифорнии. Эта система была лицензирована в других штатах, начиная с 1966 года, и была переименована в Visa в 1976 году.

Затем последовали другие основные банковские карты, включая MasterCard, ранее Master Charge. Чтобы предлагать расширенные услуги, такие как питание и проживание, многие мелкие банки, которые ранее предлагали кредитные карты на местной или региональной основе, установили отношения с крупными национальными или международными банками.

Хотя телефонные компании, газовые компании и универмаги имеют свои собственные системы нумерации, стандарт ANSI X4.13-1983 является системой, используемой большинством национальных систем кредитных карт.

Вот что означают некоторые цифры:

Первая цифра в номере вашей кредитной карты обозначает систему:

- 3 - карты для путешествий / развлечений (например, American Express и Diners Club).
- 4 – Visa.
- 5 – MasterCard.
- 6 – Discover Card.

Структура номера карты зависит от системы. Например, номера карт American Express начинаются с 37; Carte Blanche и Diners Club с 38.

- American Express - Цифры три и четыре - это тип и валюта, цифры с 5 по 11 - номер счета, цифры с 12 по 14 - номер карты в счете, а цифра 15 - контрольная цифра.
- Visa - Цифры от двух до шести - это номер банка, цифры от 7 до 12 или от 7 до 15 - это номер счета, а цифры 13 или 16 - это контрольные цифры.
- MasterCard - Цифры два и три, два-четыре, два-пять или два-шесть являются номером банка (в зависимости от того, является ли цифра два 1, 2, 3 или другим). Цифры после банковского номера до цифры 15 - это номер счета, а цифра 16 - это контрольная цифра.

Полоса на обратной стороне кредитной карты представляет собой магнитную полосу, которую часто называют магнитной полосой. Магнитная полоса состоит из крошечных магнитных частиц на основе железа в подобной пластику пленке. Каждая частица действительно представляет собой крошечный стержневой магнит длиной около 20 миллионов долей дюйма.

Магнитная полоса может быть «написана», потому что крошечные стержневые магниты могут намагничиваться в направлении северного или южного полюса. Магнитная полоса на обратной стороне карты очень похожа на кусок кассеты (подробнее см. Как работают кассеты).

Считыватель магнитных полос (вы, возможно, видели, как тот, кто подключен к чьему-либо компьютеру на базаре или ярмарке), может понять информацию на трехполосной полосе. Если банкомат не принимает вашу карту, возможно, ваша проблема также:

- Грязная или поцарапанная магнитная полоса.
- Стертая магнитная полоса (Наиболее распространенные причины стертых магнитных полос - это воздействие магнитов, таких как

маленькие, которые используются для хранения заметок и фотографий на холодильнике, а также воздействие размагничивающего устройства в магазине (EAS) в магазине).

Есть три следа на магнитной полосе. Каждая дорожка имеет ширину около одной десятой дюйма. Стандарт ISO / IEC 7811, который используется банками, определяет:

- Первый трек равен 210 бит на дюйм (bpi) и содержит 79 6-битных символов плюс бит четности только для чтения.
- Второй трек имеет 75 бит / дюйм и содержит 40 4-битных символов плюс бит четности.
- Третий трек имеет 210 бит / дюйм и содержит 107 4-битных символов плюс бит четности.

Ваша кредитная карта обычно использует только треки один и два. Третий трек - это трек для чтения / записи (который включает в себя зашифрованный PIN-код, код страны, денежные единицы и разрешенную сумму), но его использование не стандартизировано среди банков.

Информация по первой дорожке содержится в двух форматах: А, который зарезервирован для частного использования эмитента карты, и В, который включает в себя следующее:

- Старт часового - один персонаж.
- Код формата = "В" - один символ (только альфа).
- Основной номер счета - до 19 символов.
- Разделитель - один символ.
- Код страны - три символа.
- Имя - от двух до 26 символов.
- Разделитель - один символ.

- Срок годности или разделитель - четыре символа или один символ
- Дискреционные данные - достаточно символов для заполнения максимальной длины записи (всего 79 символов).
- Конечный страж - один персонаж.
- Продольная проверка избыточности (LRC) - один символ LRC является формой вычисленного контрольного символа.

Формат для второй дорожки, разработанный банковской индустрией, выглядит следующим образом:

- Старт часового - один персонаж.
- Основной номер счета - до 19 символов.
- Разделитель - один символ.
- Код страны - три символа.
- Срок годности или разделитель - четыре символа или один символ

Дискреционные данные - достаточно символов для заполнения максимальной длины записи (всего 40 символов).

- LRC - один персонаж.

Для получения дополнительной информации о формате дорожки см. Стандарты карт с магнитной полосой ISO [2].

Существует три основных метода определения того, будет ли ваша кредитная карта платить за то, что вы снимаете:

- Продавцы с небольшим количеством транзакций каждый месяц проводят голосовую аутентификацию, используя телефон с тональным набором.
- Терминалы для считывания карт с магнитной полосой с электронным захватом данных (EDC) становятся все более

распространенным явлением - так же, как и при проверке вашей карты.

- Виртуальные терминалы в интернете.

Вот как это работает: после того, как вы или кассир проведете свою кредитную карту через считывающее устройство, программное обеспечение EDC на терминале торговой точки (POS) набирает сохраненный телефонный номер (с помощью модема) для вызова эквайера. Эквайер - это организация, которая собирает запросы по кредитной аутентификации от продавцов и предоставляет продавцам гарантию оплаты.

Когда компания-эквайер получает запрос проверки подлинности кредитной карты, она проверяет транзакцию на достоверность и запись на магнитной полосе для:

- Идентификатор продавца.
- Действительный номер карты.
- Дата окончания срока.
- Лимит кредитной карты.
- Использование карты.

Транзакции с одним набором номера обрабатываются со скоростью от 1200 до 2400 бит / с, в то время как прямое подключение к Интернету использует намного более высокие скорости по этому протоколу. В этой системе владелец карты вводит персональный идентификационный номер (ПИН) с клавиатуры.

ПИН-кода нет на карте - он зашифрован (скрыт в коде) в базе данных. (Например, перед тем, как получить наличные в банкомате, банкомат зашифровывает ПИН-код и отправляет его в базу данных, чтобы узнать, есть ли совпадение.) ПИН-код может храниться на компьютерах банка в зашифрованном виде (в виде шифра). или зашифрованы на самой карте. Преобразование,

используемое в этом типе криптографии, называется односторонним. Это означает, что можно легко вычислить шифр с учетом ключа банка и PIN-кода клиента, но с вычислительной точки зрения невозможно получить обычный шифр PIN-кода из шифра, даже если ключ известен. Эта функция была разработана для защиты держателя карты от того, что кто-то имеет доступ к компьютерным файлам банка.

Аналогичным образом, связь между банкоматом и центральным компьютером банка зашифрована, чтобы не допустить прослушивания потенциальными ворами телефонных линий, записи сигналов, отправляемых в банкомат, чтобы разрешить выдачу наличных, а затем подачи этих же сигналов в банкомат для обмануть его в несанкционированной выдаче наличных.

Если этой защиты недостаточно для того, чтобы успокоить ваш разум, теперь есть карты, которые используют еще больше мер безопасности, чем ваша обычная кредитная карта: смарт-карты.

«Умная» кредитная карта - это инновационное приложение, которое включает в себя все аспекты криптографии (секретные коды), а не только аутентификацию, которую мы описали в предыдущем разделе. В смарт-карту встроен микропроцессор. Криптография имеет важное значение для функционирования этих карт несколькими способами:

Пользователь должен подтверждать свою личность на карте каждый раз, когда совершается транзакция, почти так же, как PIN-код используется в банкомате.

Карта и считыватель карт выполняют последовательность зашифрованных обменов, аналогичных знакам / встречным знакам, чтобы убедиться, что каждый имеет дело с законным контрагентом.

Как только это установлено, сама транзакция выполняется в зашифрованном виде, чтобы никто, включая владельца карты или продавца, чье

устройство для чтения карт было задействовано, не «подслушивал» обмен, а затем выдавал себя за другую сторону за мошенничество в системе.

Этот сложный протокол составлен таким образом, что он невидим для пользователя, за исключением необходимости ввода PIN-кода для начала транзакции.

Смарт-карты впервые получили широкое распространение во Франции в 1984 году. Сейчас они являются популярным товаром, который, как ожидается, заменит простые пластиковые карты, которые большинство из нас использует сейчас. Visa и MasterCard лидируют в США благодаря технологиям смарт-карт [3].

Чипы на этих картах способны на множество видов транзакций. Например, вы можете совершать покупки с вашего кредитного, дебетового или сохраненного значения, которое можно перезагружать. Расширенные возможности памяти и обработки смарт-карт во много раз выше, чем у традиционных карт с магнитной полосой, и могут вместить несколько различных приложений на одной карте. Он также может хранить идентификационную информацию, отслеживать ваше участие в программе близости (лояльности) или предоставлять доступ к вашему офису. Это означает, что вам больше не придется перетасовывать карты в вашем кошельке, чтобы найти нужную - смарт-карта будет единственной, которая вам нужна!

Несмотря на то, что цифры растут, потребители все еще не используют свои кредитные карты в Интернете почти так, как хотелось бы электронным торговцам (электронным магазинам). Вот почему многие кибер-продавцы продолжают предлагать бесплатный номер заказа, чтобы покупатели могли выбирать, куда их заказывать. Кибер-шопинг может быть удобным - и некоторые люди делают все покупки в Интернете - но с помощью кредитной карты Мошенничество всегда представляет угрозу, как в Интернете, так и в реальном мире. Хакеры нашли способы украсть номера кредитных карт с веб-сайтов.

Независимо от того, какую карту и план вы выберете, у вас должен быть доступ к следующей информации в соответствии с федеральным Законом о правде в кредитах, чтобы вы могли сравнить один кредит с другим:

- Расходы на финансирование в долларах и в процентах годовых.
- Эмитент кредита или компания, предоставляющая кредитную линию.
- Размер кредитной линии.
- Продолжительность льготного периода, если таковой имеется, до оплаты.
- Минимальный платеж требуется.
- Ежегодные сборы, если применимо.
- Сбор за страхование кредита (если есть), который погашает ваш кредит, если вы умрете до того, как долг будет полностью погашен.

Есть в основном три типа кредитных карт:

- Банковские карты, выпущенные банками (например, Visa, MasterCard и Discover Card).
- Карты для путешествий и развлечений (Т & Е), такие как American Express и Diners Club.
- Карточки для дома, которые хороши только в одной сети магазинов (самый крупный из них - Sears, за ними следуют нефтяные компании, телефонные компании и местные универмаги.) Карты Т & Е и национальные карточки на дом имеют одинаковые условия, где бы вы ни действовали [9].

1.1.3 POS – терминалы

POS-терминал - это аппаратная система для обработки карточных платежей в точках продаж. Программное обеспечение для чтения магнитных полос кредитных и дебетовых карт встроено в аппаратное обеспечение. Портативные устройства (то есть не терминалы, привязанные к счетчику), как собственные, так и сторонние, а также бесконтактные возможности для новых форм мобильных платежей, представляют системы POS следующего поколения. POS-терминалы предназначены для обработки транзакций при финансовых расчетах с использованием пластиковых карточек. POS-терминалы позволяют процесс обслуживания операций с пластиковыми картами.

Когда для оплаты чего-либо используется кредитная или дебетовая карта, обычный кассовый терминал (POS) сначала считывает магнитную полосу, чтобы проверить, достаточно ли средств для передачи продавцу, а затем выполняет перевод. Сделка купли-продажи записывается, а квитанция распечатывается или отправляется покупателю по электронной почте или в текстовом виде. Торговцы могут купить или арендовать POS-терминал в зависимости от того, как они предпочитают управлять денежными потоками. Покупка системы предполагает более высокие первоначальные затраты, в то время как аренда выравнивает ежемесячные платежи, хотя общие арендные платежи могут в конечном итоге стать более чем разовой покупкой в течение срока полезного использования системы.

В настоящее время наблюдается тенденция отхода от традиционного проприетарного оборудования и программных POS-систем, которые можно загружать в планшет или другое мобильное устройство. Чтобы идти впереди, производители POS-терминалов представляют свои собственные версии портативных и мобильных POS-устройств.

Такие устройства можно увидеть в оживленных розничных магазинах и ресторанах, где владельцы осознают тот факт, что покупатели обычно не любят ждать, чтобы заплатить за товар или еду. Цена, функциональность и удобство являются важными критериями для покупателей POS-систем. Чрезвычайно важным в растущем взаимосвязанном мире является безопасность систем. Некоторые громкие взломы данных клиентов произошли через POS-терминалы, которые не имели обновленных операционных систем.

Первая система продаж была разработана Национальным кассовым аппаратом (NCR) - компанией, ответственной за большинство кассовых аппаратов в мире сегодня. Компания интегрировала новые технологии, такие как штрих-коды и сканеры, разработанные в 1980-х годах, для преобразования ручных кассовых аппаратов в мобильные системы продаж.

Square, Inc. была новатором в области PoS в последнее время. Она представила аппаратное и программное обеспечение, «чтобы преобразовать процесс оформления заказа и продвинуть цифровую и мобильную коммерцию, развязав продажи с длинных очередей и устаревших кассовых аппаратов», говорится в заявлении компании в форме S-1.

Его системы взаимодействуют напрямую с сетями платежных карт, снимая бремя поддержания соответствия правилам и нормам платежной индустрии с плеч продавцов. Бизнес-аналитика в POS-системах компании также является еще одной привлекательной особенностью. Тем не менее, это поле с относительно низкими входными барьерами - площадь, возможно, была пионером, но есть много конкурентов. [4]

1.2 Arduino

Arduino - это электронная платформа с открытым исходным кодом, основанная на простом в использовании аппаратном и программном обеспечении, используемом для создания проектов электроники.

Все платы Arduino имеют одну общую черту - микроконтроллер. Микроконтроллер - это маленький компьютер.

С Arduino вы можете проектировать и создавать устройства, которые могут взаимодействовать с окружающей средой. Платы Arduino - это в основном инструмент для управления электроникой. Они могут считывать входные данные с помощью встроенного в них микроконтроллера (например, свет на датчике, объект рядом с датчиком) и превращать его в выходной сигнал (приводить в движение мотор, звонить в сигнализацию, включать светодиод, отображать информацию на ЖК-дисплее),

Однако для этого вам сначала нужно запрограммировать плату Arduino.

Как вы программируете плату Arduino? Вы используете приложение под названием Arduino IDE (интегрированная среда разработки), о котором мы расскажем позже.

С Arduino производители и электрики могут легко создавать прототипы своих продуктов и воплощать их идеи в жизнь.

Существует много электронных плат, зачем использовать плату Arduino? Ну, есть много причин, которые делают этот микроконтроллер особенным. Преимущества использования Arduino включают в себя:

1) Дешево

- Всякий раз, когда вы покупаете что-то, вы всегда будете сначала смотреть на стоимость.
- Для Arduino они очень доступны и экономичны!

- Вы можете получить официальную полную версию Arduino UNO Rev3 всего за \$ 24,95 или нашу собственную Seeeduino V4.2, которая является Arduino-совместимой платой, основанной на микроконтроллере ATmega328P (аналогично Arduino UNO), всего за \$ 6,90!
- Не нарушая свой кошелек, вы можете легко получить Arduino для себя!

2) Прост в использовании и идеально подходит для начинающих

- С Arduino простое в использовании программное обеспечение IDE для начинающих, Arduino легче учиться программировать, поскольку оно использует упрощенную версию C ++ по сравнению с другим программным обеспечением.
- Кроме того, сообщество Arduino очень большое, и многие пользователи и организации используют его. В Интернете доступны разнообразные учебные пособия и проекты, которые предварительно кодируются для того, чтобы вы могли учиться и создавать их с помощью Arduino, что позволяет начинающим легко начать работу.
- Даже если у вас возникнут какие-либо проблемы, просто попросите помощи в комментариях или на форумах, где пользователи помогают друг другу решать проблемы.
- Вы опытный электрик и боитесь, что Arduino будет легким для вас? Не беспокойтесь, так как программное обеспечение Arduino (IDE) также является гибким для продвинутых пользователей, чтобы воспользоваться ими!

3) Кросс-платформенная

- Arduino IDE также является кроссплатформенным, что означает, что вы можете запускать его в Windows, Macintosh OSX, а также в операционных системах Linux по сравнению с другими системами микроконтроллеров, которые могут работать только под Windows.

4) Широкое разнообразие

- В Arduino есть много вариантов, из которых вы можете выбрать тот, который больше всего подходит вашему проекту!
- Есть ограничения по пространству? Вы можете приобрести себе Arduino Nano размером всего 43,18 мм на 18,54 мм!
- Требовать больше памяти и вычислительной мощности? Вы можете получить себе Arduino Mega!

Arduino в основном состоит из двух компонентов, о которых вам следует знать: аппаратного и программного обеспечения. Для аппаратной части Arduino он состоит из физической платы, а также датчиков и экранов, используемых для взаимодействия с платой.

Arduino Hardware

Arduino Board

Физическим оборудованием Arduino является сама плата. Однако, когда речь идет о платах Arduino, существует множество разновидностей с различными функциональными возможностями.

Для примера мы рассмотрим Seeeduno V4.2, которая имеет те же функции, что и одна из самых популярных плат Arduino - Arduino UNO. Большинство плат Arduino будут иметь следующие общие компоненты, которые мы собираемся перечислить:

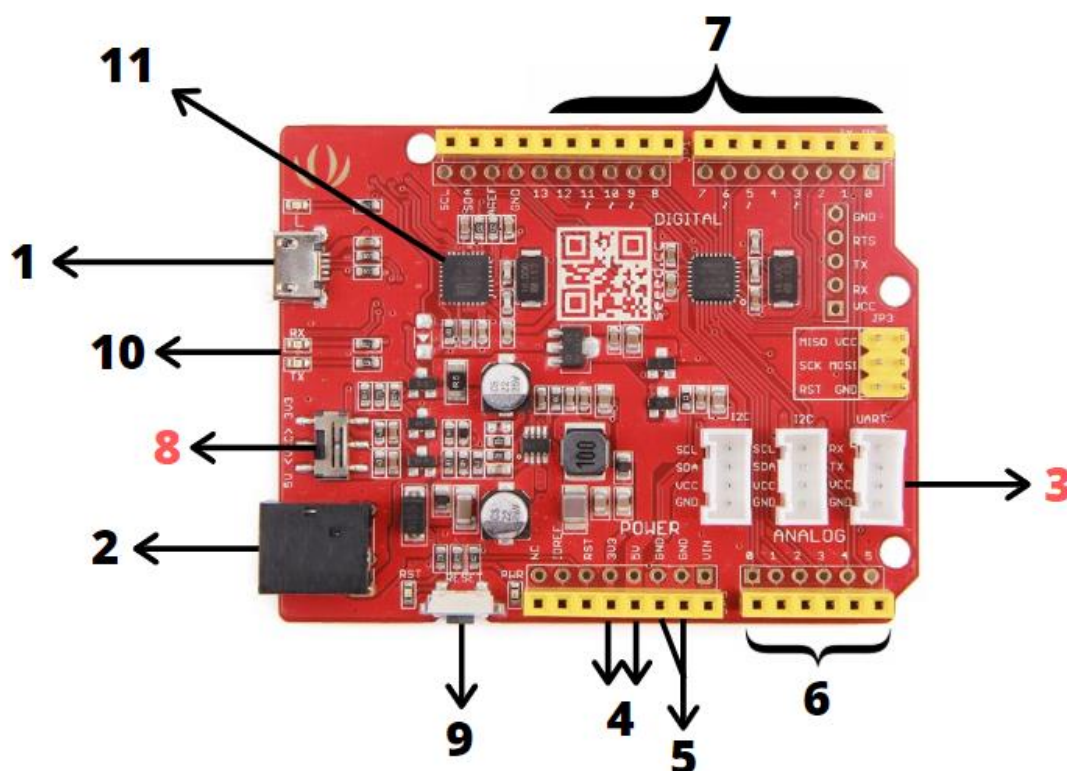


Рисунок 1 – Устройство платы Arduino

По сравнению с Arduino UNO, он имеет некоторые дополнительные функции, которые выделены красным, которые можно найти только на наших досках Seeeduino!

1) USB-вход.

- Порт USB используется для подключения платы к компьютеру для программирования и включения платы Arduino.
- Это USB-соединение важно, так как оно будет через этот порт, куда вы будете загружать свой код на плату Arduino.
- Чтобы узнать больше о том, как загрузить код на Arduino, вы можете обратиться к нашему руководству о том, как загрузить код на Arduino.

2) Вход постоянного тока.

- Разъем питания постоянного тока позволяет питать вашу плату Arduino от настенного адаптера, чтобы вы могли обеспечить больше энергии для вашего проекта, если это необходимо.

3) Соединители Grove.

- Эти разъемы Grove можно найти только на наших платах Seeeduino.
- SeeedStudio имеет множество датчиков / устройств, которые могут использовать это соединение I2C или UART.
- С нашими разъемами Grove вы можете легко подключать модули для использования с Arduino без каких-либо паяльных или перемычек.

3.3 В и 5 В Контакты

4) 3.3 В и 5 В Контакты.

- Как следует из названия, контакты 3,3 В и 5 В подают напряжение на ваши модули. Контакт 3,3 В подает 3,3 вольт, а контакт 5 В - 5 В.

5) Выводы GND.

- С этим выводом GND (Земля) они используются для заземления вашей цепи.
- GND означает, что этот вывод находится под нулевым напряжением относительно источника питания и плоскости заземления монтажной платы.

6) Аналоговые контакты

- Аналоговые контакты позволяют Arduino считывать сигналы с аналогового датчика, такого как датчик освещенности, и преобразовывать его в цифровое значение.
- Несмотря на то, что основная функция аналоговых выводов для большинства пользователей Arduino заключается в считывании аналоговых датчиков, аналоговые выводы также обладают всеми функциями выводов ввода / вывода общего назначения (GPIO).

7) Цифровые Пины.

- В Sceduino или Arduino UNO цифровые контакты находятся на контактах от 0 до 13.
- Они позволяют Arduino считывать цифровые входы, такие как нажатие кнопки, и цифровой вывод, например, включать светодиод.

8) Выключатель питания системы.

- Этот системный выключатель питания можно найти только на наших платах Sceduino.
- Этот ползунковый переключатель используется для изменения логического уровня и рабочего напряжения платы либо на 5 В, либо на 3,3 В, что полезно, так как если вы хотите сэкономить энергию, вы можете установить его на 3,3 В.

9) Кнопка сброса.

- Эта кнопка сброса позволяет сбросить плату и перезапустить любой код, загруженный на вашу плату Arduino. После нажатия контакт сброса будет временно подключен к земле.

- Эта кнопка сброса очень полезна для ваших проектов, если ваш код не повторяется, но вы хотите протестировать его несколько раз.
- Эта кнопка удобно расположена сбоку, чтобы вы могли сбросить плату Sseeduino, даже если щит находится сверху. Это не относится к другим платам Arduino, где кнопка расположена сверху, что затрудняет доступ к ней.

10) Индикатор RX / TX.

- Также известные как индикаторы передачи и приема, светодиодные индикаторы TX и RX подключены к TX и RX микросхемы USB-to-UART.
- Они работают автоматически и сообщают вам, когда доска отправляет или получает данные, например, когда вы загружаете программу на свою плату Arduino.

11) Микроконтроллер

- На Sseeduino V4.2 и Arduino UNO они основаны на микроконтроллере: ATmega328P
- Это основной чип, который действует как мозг вашей платы Arduino.
- Они позволяют вам запрограммировать Arduino, чтобы он мог выполнять команды и решения на основе кода.
- Вы должны будете знать, какой тип микроконтроллера использует ваша плата, прежде чем загружать новую программу из Arduino Software.
- Хотя микроконтроллер на платах Arduino отличается, их разница невелика. Единственное отличие, которое вы можете заметить, - это разное количество встроенной памяти.

С вашей платой Arduino вы определенно ничего не сможете с ней поделаться.

Это где датчики Arduino и щиты входят:

О Grove

- Grove - это модульная стандартизированная система прототипирования соединителей. Гроув использует строительный подход для сборки электроники. По сравнению с системой, основанной на перемычках или припоях, легче подключать, экспериментировать и строить, что упрощает систему обучения, но не до такой степени, что она становится тупой.
- Некоторые из других прототипных систем доводят уровень до уровня строительных блоков, но система Grove позволяет создавать реальные системы. Требуется некоторое обучение и экспертиза, чтобы соединить вещи.
- Каждый модуль Grove, который у нас есть, обычно предназначен для одной функции, такой как простая кнопка или более сложный датчик сердечного ритма.
- Вам не нужен базовый блок для подключения к модулям Grove. Вы можете использовать кабель (Grove to Pin Header Converter) для удобного подключения от контактов Arduino к встроенным разъемам Grove.
- У вас нет доски Seeeduino? Это тоже хорошо! Вы можете получить наш Базовый экран, который имеет 16 разъемов роуи, которые вы можете подключить и играть!
- Таким образом, эта система идеально подходит для новичков в мире электроники и Arduino.

Без дальнейших церемоний, давайте прыгнем прямо в расширенное семейство Arduino: сенсоры и щиты.

Arduino sensors

Имея несколько строк кода на Arduino, вы можете поиграть и управлять множеством датчиков и создавать потрясающие проекты. Наши датчики могут

измерять свет, ультразвуковое расстояние, влажность, температуру, влажность, газ, давление, движение, звук, осязание и многое другое! Что бы вы ни думали, наши датчики Grove могут это почувствовать! Не говоря уже о том, что все наши сенсорные модули Arduino совместимы с нашей системой Grove, что идеально подходит для начинающих.

Arduino shields

Arduino shields - это предварительно смонтированные печатные платы, которые легко подключаются к верхним контактам Arduino для расширения его возможностей.

С Arduino, добавление Bluetooth, подключение к Wi-Fi, GPS, драйвер двигателя может быть затруднено, если вы новичок в Arduino. С помощью щитов вы можете избежать всех проблем и легко подключить щит на задней панели вашего Arduino.

Подобно сенсорам, экраны имеют различные функции, такие как подключение к Wi-Fi, Ethernet, управление двигателями и управление ими, камера, память, сенсорный экран, E-Ink Display и многое другое!

Arduino Software

Зная аппаратное обеспечение Arduino, вам потребуется программное обеспечение и программы, чтобы оживить Arduino и позволить ему взаимодействовать с различными датчиками и экранами.

Для программирования вашего Arduino вам потребуется программное обеспечение Arduino IDE.

Об Arduino IDE

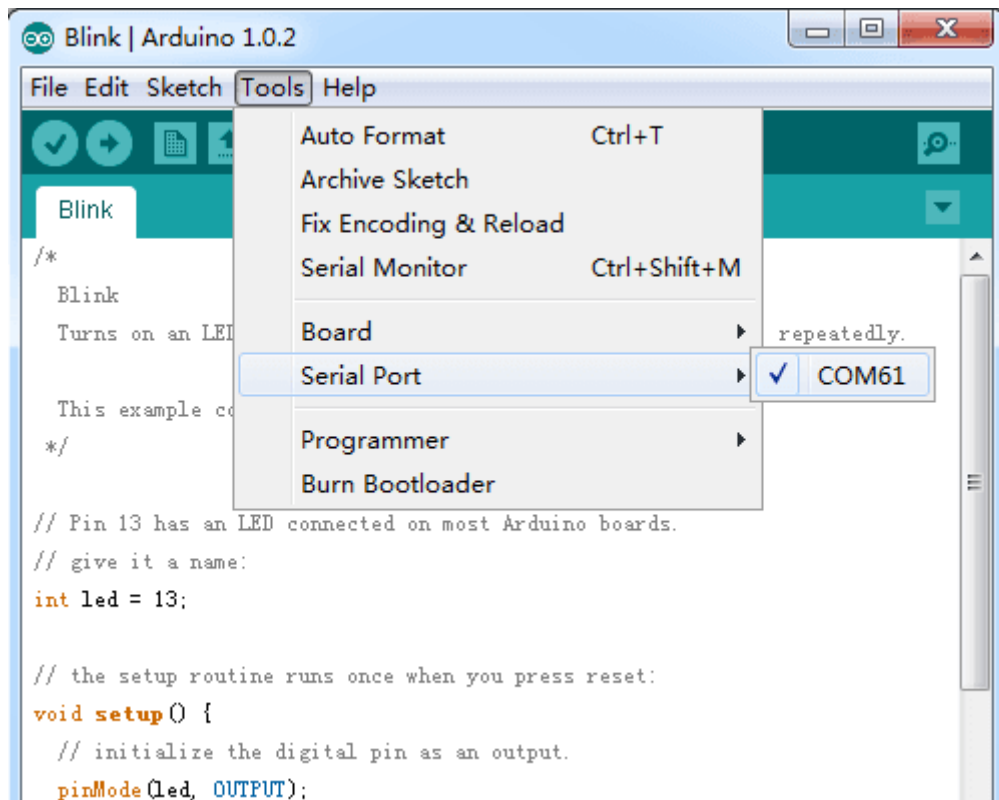


Рисунок 2 – Arduino IDE software

- Arduino IDE позволяет легко писать код и загружать его на свою плату Arduino.
- Эта программа является кроссплатформенной, что означает, что она может работать в Windows, Mac OS X и Linux по сравнению с другими системами микроконтроллеров, которые могут работать только под Windows.
- Это программное обеспечение может использоваться с любой платой Arduino, такой как Seeeduino V4.2, Arduino UNO и т. Д.
- Среда написана на Java и основана на обработке и другом программном обеспечении с открытым исходным кодом.
- Эта программа использует упрощенную версию C ++ с подсветкой синтаксиса и другими функциями, облегчающими обучение программированию, что идеально подходит для начинающих изучать программирование и кодирование!
- После того, как вы закончили писать код, вы можете легко загрузить его в Arduino IDE с помощью USB-кабеля одним нажатием кнопки.

Теперь, когда вы поняли аппаратную и программную часть Arduino, пришло время выбрать собственную плату Arduino! Тем не менее, вы можете заметить, что существует множество вариаций от официальных плат Arduino до плат, совместимых с Arduino, каждая с различными возможностями и ценовыми показателями.

Сегодня мы собрали некоторые из них, которые являются наиболее подходящими для начинающих! Они есть:

1) Arduino Uno Rev3

- Arduino Uno - это идеальная доска для начала работы с электроникой, веселых и увлекательных проектов. Эта доска - ваш вход в уникальный опыт Arduino: отлично подходит для изучения основ работы датчиков и исполнительных механизмов, а также является важным инструментом для ваших нужд быстрого прототипирования.
- Arduino Uno Rev3 также является наиболее используемой и документированной платой в семействе Arduino. Есть много учебных пособий и проектов, доступных онлайн, с инструкциями для начала.
- Он имеет 14 цифровых входов / выходов (из которых 6 могут использоваться в качестве ШИМ-выходов), 6 аналоговых входов, кварцевый генератор 16 МГц, разъем USB, разъем питания, разъем ICSP и кнопку сброса.
- Он содержит все необходимое для поддержки микроконтроллера; просто подключите его к компьютеру с помощью USB-кабеля или включите адаптер переменного тока в постоянный ток или батарею, чтобы начать работу.

2) Seeduino V4.2

- Seeeduino V4.2 основан на загрузчике Arduino UNO. Наш Seeeduino V4.2, по сути, намного дешевле Arduino UNO с большим количеством функций!
- Он имеет такое же аппаратное обеспечение и функции, с некоторыми дополнительными функциями, которые есть только в нашем Seeeduino V4.2, такими как:
- Переключитесь, чтобы выбрать напряжение питания системы, 3,3 В или 5 В, что очень полезно, если вы хотите установить систему на 3,3 В для экономии энергии.
- Три встроенных интерфейса Grove позволяют вашей плате легко подключаться к нашим модулям Grove. (Более подробно расскажу о Grove в нашем секторе «Сенсоры и щиты»!)
- Использование преобразователя постоянного тока вместо LDO (регулятор Low DropOut) для повышения эффективности

3) Seeduino Nano

- Хотите меньше Arduino UNO или Seeeduino V4.2 для ваших потребностей проекта? Тогда Seeduino Nano будет идеальным для вас!
- Seeduino Nano - это компактная плата, похожая на Seeeduino V4.2 / Arduino UNO, и она полностью совместима с Arduino Nano по разводке и размерам.
- Имея размеры 43,18 мм × 18,54 мм и размер менее четверти по сравнению с Seeeduino V4.2, размер Seeduino Nano, а также надежность позволяют легко интегрировать их во многие проекты, такие как носимые устройства, мини-роботы и многие другие!
- Кроме того, Seeduino Nano имеет 1 встроенный интерфейс Grove, который позволяет вашей плате легко подключаться к нашим модулям Grove.

4) Seeduino Mega(ATmega2560)

- Хотите больше, лучше и мега Arduino? Мега Seeeduino определенно вписывается в эту категорию.
- Seeeduino Mega - мощный микроконтроллер, производный от Arduino Mega. Он оснащен процессором ATmega2560, который имеет большое количество выводов ввода / вывода.
- Он имеет до 70 цифровых входов / выходов, 16 аналоговых входов, 14 ШИМ и 4 аппаратных последовательных порта.
- По сравнению с Arduino Mega мы сократили объем Arduino Mega как минимум на 30% и сделали его на 100% совместимым с продуктами Sseed Shield.
- С этой платой она очень подходит для проектов, которые требуют много цифровых входов или выходов, таких как светодиоды, кнопки и т. Д. [5]

Выводы

В исследовательской части была изучена предметная область работы, которая включает в себя обзор структуры электронных платежей, кредитных карт, POS-терминалов и Arduino.

2 Конструкторская часть

2.1 Выбор конструкции

Для разработки данного решения было выдвинута идея с робот-манипуляторами. Робот-манипулятор – устройство для имитации действий объектов из реального мира. Для наших нужд достаточно имитация руки, что и было выбрано в качестве каркаса проекта. Роботизированная рука двигается с помощью сервоприводов. Сервопривод – механический прибор автоматической коррекцией состояния через внутреннюю отрицательную обратную связь, в соответствии с параметрами, заданными извне. В качестве роботизированной руки мы решили выбрать готовое решение. В качестве устройства управления и программирования сервоприводов было выбрана платформа на основе Arduino.

Путем анализа рынка были выявлено несколько готовых решений.

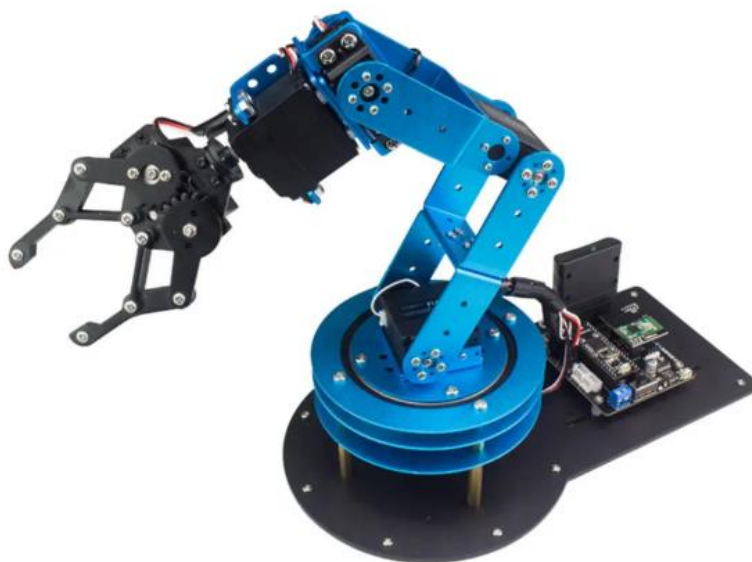


Рисунок 3 – Конструкция от компании Hiwonder LOBOT

Достоинства данной конструкции:

- Хорошая конструкция, весьма точная и устойчивая.
- В наличие сервоприводы, конструкция, ардуино.
- Есть три вида управления: программное, с помощью джостика, с компьютера или мобильного приложения.
- Возможность программирования программирование.

Недостатки:

- Ожидание сборки и доставки.
- Высокая цена.
- Плохая документация для разработки.



Рисунок 4 – Конструкция от компании ZYMini

Достоинства:

- Цена.
- В комплект входит все необходимое.
- Быстрая доставка.
- Возможность управление через Bluetooth или непосредственное программирование.

Недостатки:

- Маленький, слабая точность и не берет по вертикали (захватывает только по горизонтали).



Рисунок 5 – Конструкция от компании DIY

Достоинства:

- Плохая конструкция.
- Быстрая доставка.

Недостатки:

- Цена.
- В комплекте нет модуля Arduino.

После длительного анализа и проработки решения был выбран первый экспонат, так как он самый подвижный и точный, а все остальное можно улучшить и расширить в дальнейшем самостоятельно.

При первоначальном тестировании возникли проблемы с программированием стандартного модуля Arduino, который шел в комплекте с роботизированной рукой, так как этот модуль был кастомизированным. Для него была плохая документация. В результате было принято решение заменить этот модуль на широко применяемый в данной среде разработке. Был произведен анализ рынка для выбора подходящей платы.



Рисунок 6 – Плата Arduino Uno

Arduino Uno является стандартной платой Arduino и возможно наиболее распространенной. Она основана на чипе ATmega328, имеющем на борту 32 КБ флэш-памяти, 2 Кб SRAM и 1 Кбайт EEPROM памяти. На периферии имеет 14 дискретных (цифровых) каналов ввода / вывода и 6 аналоговых каналов ввода / вывода, это очень разносторонне-полезные девайсы, позволяющие покрывать большинство любительских задач в области микроконтроллерной техники. Данная плата контроллера является одной из самых дешевых и наиболее часто

используемых. При планировании нового проекта, если вы незнакомы, с платформой Arduino, советуем начать с Uno.



Рисунок 7 – Плата Arduino Leonardo

Та же Arduino Uno, но с другим микроконтроллером, который находится в том же классе, но имеет некоторые отличия положительного характера. Большее количество аналоговых входов (12 против 6) для сенсоров, больше каналов ШИМ (7 против 6), больше пинов с аппаратным прерыванием (5 против 2), отдельные независимые serial-интерфейсы для USB и UART. Arduino Leonardo может притворяться клавиатурой или мышью (HID-устройством) для компьютера. Это позволяет легко сделать своё собственное устройство ввода. Из-за распиновки чуть отличной от Arduino Uno возможна несовместимость с некоторыми платами расширения.

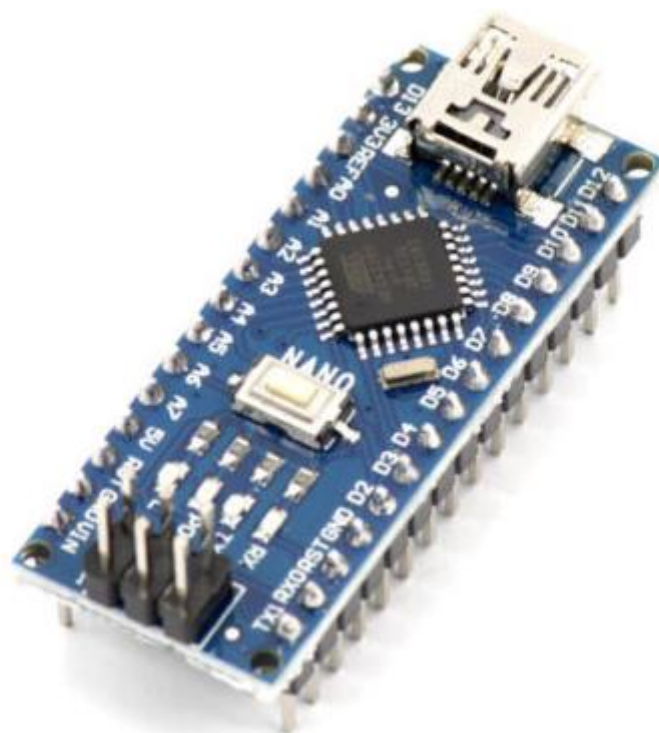


Рисунок 8 – Плата Arduino Nano

Arduino Nano — это функциональный аналог Arduino Uno, но размещённый на миниатюрной плате. Отличие заключается в отсутствии собственного гнезда для внешнего питания, использованием чипа FTDI FT232RL для USB-Serial преобразования (либо CH340G, требуется установить соответствующие драйвера) и применением mini-USB кабеля для взаимодействия вместо стандартного. В остальном, начинка и способы взаимодействия совпадают с базовой моделью. Платформа имеет штырьковые контакты, что позволяет легко устанавливать её на макетную плату. Используйте Arduino Nano там, где важна компактность, а возможностей Arduino Mini либо недостаточно, либо не хочется заниматься пайкой.

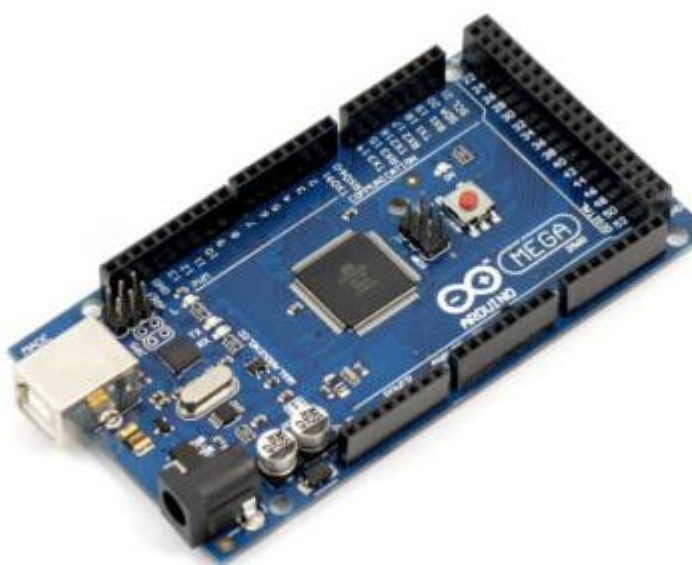


Рисунок 9 – Плата Arduino Mega

Как Arduino Uno, но на базе более мощного микроконтроллера той же архитектуры. Отличный выбор «на вырост» или если Arduino Uno перестала справляться. В разы больше памяти: 256 КБ против 32 КБ постоянной и 8 КБ против 2 КБ оперативной. В разы больше портов: 60 из них 16 аналоговых и 15 с ШИМ. Немного длиннее базовой Arduino Uno: 101×53 мм против 69×53 мм.

В результате анализа рынка и сравнения всех плат была выбрана плата Arduino Uno R3, так как она является самой документированной и распространенной платой, имеет достаточно входов для сервоприводов, оснащен входом для внешнего источника питания.

2.2 Программный интерфейс банковского приложения JPAУ

2.2.1 Протокол обмена данными с банковским приложением

Для обмена данными с банковским приложением используется протокол ТСР/IP. Клиент подключается по умолчанию к порту 4433.

Последовательность обработки одной команды:

1. Клиент устанавливает ТСР соединение с банковским приложением.
2. Передает команду.
3. Получает сообщения от приложения.
4. Отвечает на сообщения.
5. Получает ответ.
6. Закрывает соединение.

Пункты 2 и 3 используются для обеспечения обратной связи во время выполнения некоторых команд. Например, клиент может получить сообщение keep-alive во время исполнения длительной операции.

2.2.2 Формат данных и ограничения интерфейса

Клиент и банковское приложение обмениваются пакетами. Пакет может содержать команду, сообщение, ответ на команду или ответ на сообщение. Каждый пакет начинается с 4 двоичных байтов в которых указана длина пакета без учета этих 4х байтов. Первый из байтов длины старший. Далее следует текст в формате XML в кодировке ASCII. Имя корневого тэга команды (или сообщения) это имя команды (или сообщения). Имя корневого тэга ответа должно совпадать с именем указанным в запросе.

Функции настройки терминала, отчеты и транзакции с вводом данных карты вручную доступны только администратору терминала.

Для перехода в режим настройки приложению передается пароль состоящий из 8 цифр. В ответ клиент получает токен - случайное 16 байтовое число в формате HEX ASCII, которое далее включается в список параметров вызова "защищенных" функций. Пароль по умолчанию 12345678. Клиент должен сохранить полученный при входе в режим администратора токен и использовать его в командах до выхода из режима настройки. Текущий пароль администратора, дополнительно к токену, указывается при смене пароля администратора.

Сверка итогов, печать отчетов и исполнение транзакций, в которых данные карты вводятся вручную оператором, доступны только по паролю администратора. Схема работы с токеном на них не распространяется.

2.2.3 Команды

В описании команд приводятся примеры значений параметров. В ответе на команду или сообщение всегда содержится тэг с кодом ошибки status. Список кодов ошибок:

- ok - операция завершена успешно.
- failed - операция завершена с ошибкой.
- communicationerror - ошибка связи в т.ч. ошибка установки соединения по сети.
- notimplemented - запрашиваемая функция не реализована.
- ormaterror - ошибка представления данных.
- notauthorized - неопустимый логин или пароль.
- fileioerror - ошибка доступа к файлу.
- verificationfailed - ошибка проверки контрольной суммы или сертификата.
- missingdata - отсутствуют данные необходимые для выполнения операции.
- systemerror - системная ошибка
- timeout - превышено время ожидания завершения операции.
- invalidarguments - указаны недопустимые значения параметров операции.
- transactionnotfound - отменяемая транзакция отсутствует в логе.
- notreversible - транзакция не может быть отменена.
- alreadyreversed - транзакция уже отменена.
- notapproved - отменяемая транзакция не одобрена.
- emverror - ошибка ядра EMV. В т.ч. ошибка инициализации.
- notdetected - ошибка обнаружения карты.
- notallowedinterface - использование интерфейса запрещено для данной операции.
- cancelled - операция отменена.
- tryagain - ошибка чтения карты; повторить.
- usechip - требуется провести контактную emv транзакцию.

- batchuploadfailed - ошибка загрузки лога транзакций.
- readerdisabled - устройство чтения карт отключено.
- preprocessingfailed - ошибка предварительной обработки транзакции в ядре EMV.
- readernotavailable - устройство чтения карт отсутствует.
- readererror - ошибка устройства чтения карт.
- tryanotherinterface - ошибка чтения карты; используйте другой интерфейс.
- pinpadmalfunctionornotpresent – ошибка ввода ПИН-кода.
- pinentrybypassed – ввод ПИН-кода отменен пользователем.

Тэги могут включаться в ответ или исключаться из него в зависимости от значения тега status [10]. Например, в ответе на команду login в случае ошибки авторизации отсутствует тэг token:

```
<login>
    <status>notauthorized</status>
</login>
```

В ответе на команду может содержаться лог ошибок. Он помещается в тег error-stack. Каждое сообщение об ошибке помещается в тег error. Для преобразования ошибки в текст следует преобразовать значение тега error из формата HEX ASCII в байты и затем полученную двоичную последовательность в строку UTF8:

```
<loadmasterkeys>
    <status>connectionerror</status>
    <error-stack>
        <error>
            486F7374206E616D65206973206E6F7420666F756E642E
        </error>
    </error-stack>
</loadmasterkeys>
```

Сообщение в теге error = "Host name is not found".

Клиент, вместо ответа на команду, может получить от приложения сообщение. Имя тэга сообщения отличается от ожидаемого в ответе имени команды. Клиент должен обязательно ответить на сообщение. Если клиент не знает, как обработать сообщение, он должен в ответе на сообщение указать код статуса notimplemented [6]. На сообщение keeplive следует ответить, указав status=ok.

```
<keeplive/>
```

Ответ:

```
<keeplive>
```

```
<status>ok</status>
```

```
</keeplive>
```

и снова перейти к ожиданию ответа на команду от приложения. Сообщения может отправлять только приложение. Клиент может только отвечать на поступающие сообщения. Клиент после отправки команды приложению должен дожидаться ответа, прежде чем отправлять следующую команду.

1) Вход в режим администратора

Пример команды:

```
<login>
```

```
<password>12345678</password>
```

```
</login>
```

Пример ответа:

```
<login>
```

```
<status>ok</status>
```

```
<token>DE9773A8CB888560AB0F89C07623FE03</token>
```

```
</login>
```

2) Выход из режима администратора

Пример команды:

```
<logout>  
    <token>DE9773A8CB888560AB0F89C07623FE03</token>  
</logout>
```

Пример ответа:

```
<logout>  
    <status>ok</status>  
</logout>
```

3) Смена пароля администратора

Пример команды:

```
<changepassword>  
    <token>  
        DE9773A8CB888560AB0F89C07623FE03  
    </token>  
    <password>12345678</password>  
    <newpassword>12345678</newpassword>  
</changepassword>
```

Пример ответа:

```
<changepassword>  
    <status>ok</status>  
</changepassword>
```

4) Проверка и загрузка обновлений конфигурации с сервера конфигураций

Пример команды:

```
<updateconfiguration>  
    <token>  
        DE9773A8CB888560AB0F89C07623FE03  
    </token>  
</updateconfiguration>
```

Пример ответа:

```
<updateconfiguration>  
    <status>ok</status>  
</updateconfiguration>
```

5) Загрузка рабочих ключей

Пример команды:

```
<loadworkkeys>  
    <token>  
        DE9773A8CB888560AB0F89C07623FE03  
    </token>  
</loadworkkeys>
```

Пример ответа:

```
<loadworkkeys>
    <status>ok</status>
    <mac-change-receipt>
        <rrn>123456789123</rrn>
    </mac-change-receipt>
    <net-change-receipt>
        <rrn>023456789120</rrn>
    </net-change-receipt>
</loadworkkeys>
```

В ответе передаются два чека содержащие rrn операций загрузки ключа MAC и NET (РЕК), если это предусмотрено протоколом сервера авторизации.

б) Загрузка мастер ключей

Пример команды:

```
<loadmasterkeys>
    <token>
        DE9773A8CB888560AB0F89C07623FE03
    </token>
</loadmasterkeys>
```

Пример ответа:

```
<loadmasterkeys>
    <status>ok</status>
    <receipt>
        <pkcv>123456</pkcv>
        <mkcv>789ABC</mkcv>
    </receipt>
</loadmasterkeys>
```

- pkcv - KCV мастер ключа PIN.
- mkcv - KCV мастер ключа MAC.

7) Проверка связи с сервером авторизации

Пример команды:

```
<testconnection>
    <token>
        DE9773A8CB888560AB0F89C07623FE03
    </token>
</testconnection>
```

Пример ответа:

```
<testconnection>
    <status>ok</status>
</testconnection>
```

8) Сведения о приложении и терминале

Пример команды:

```
<getparameters>
    <token>
        DE9773A8CB888560AB0F89C07623FE03
    </token>
</getparameters>
```

Пример ответа:

```
<getparameters>
    <status>ok</status>
    <parameters>
        <sn>0000000000009</sn>
```

<app>1.0.67.6</app>
<firmware-mcu>1.5.3</firmware-mcu>
<firmware-boot>2.0.1</firmware-boot>
<os>CS10_V1.07_181127PK</os>
<sdk>1.0.4</sdk>
<tid>1000000001</tid>
<mid>243423434122313</mid>
<tconf>19-04-01.01</tconf>
<ntconf>cname</ntconf>
<cconf>18-10-25.06</cconf>
<ncconf>cname_test</ncconf>
<econft>19-04-13.02</econft>
<neconf>2can_jibe_emv</neconf>
<kconf>18-08-30.04</kconf>
<nkconf>combined_ca_database</nkconf>
<acqid>twocan</acqid>
<cccert>24.03.2027</cccert>
<csca>20.11.2037</csca>
<cshost>192.168.0.185</cshost>
<accert>24.03.2027</accert>
<asca>12.10.2020</asca>
<ashost>192.168.0.2</ashost>
<kcccert>24.03.2027</kcccert>
<ksca>none</ksca>
<devid>M2100-0000005164</devid>

</parameters>

</getparameters>

- sn - серийный номер терминала.
- app - версия приложения.

- `firmware-mcu` - версия защищенного ядра терминала.
- `firmware-boot` - версия загрузчика.
- `os` - версия операционной системы.
- `sdk` - версия SDK (aar).
- `tid` - идентификатор (номер) терминала.
- `mid` - идентификатор мерчанта.
- `tconf` - версия конфигурации терминала.
- `ntconf` - имя конфигурации терминала.
- `ssconf` - версия общей конфигурации.
- `psconf` - имя общей конфигурации.
- `esconf` - версия EMV конфигурации.
- `pesconf` - имя EMV конфигурации.
- `kconf` - версия списка ключей платежных систем конфигурации.
- `nkconf` - имя списка ключей платежных систем конфигурации.
- `acqid` - идентификатор эквайера.
- `ccert` - дата окончания действия клиентского сертификата для подключения к серверу конфигурации.
- `csca` - дата окончания действия СА сервера конфигурации.
- `cshost` - имя хоста или IP сервера конфигурации.
- `accert` - дата окончания действия клиентского сертификата для подключения к серверу эквайера.
- `asca` - дата окончания действия СА сервера эквайера.
- `ashost` - имя хоста или IP сервера эквайера.
- `kccert` - дата окончания действия клиентского сертификата для подключения к серверу загрузки ключей.
- `ksca` - дата окончания действия СА сервера загрузки ключей.
- `devid` - Идентификатор устройства для эквайинга.

9) Транзакция

Примеры команд:

```
<transaction>
```

```
    <type>purchase</type>
```

```
    <currency>643</currency>
```

```
    <amount>000000012300</amount>
```

```
</transaction>
```

```
<transaction>
```

```
    <type>refund</type>
```

```
    <amount>000000012300</amount>
```

```
    <rrn>120157645346</rrn>
```

```
</transaction>
```

```
<transaction>
```

```
    <type>purchase-with-cashback</type>
```

```
    <amount>000000012300</amount>
```

```
    <amount-other>000000000100</amount-other>
```

```
</transaction>
```

```
<transaction>
```

```
    <type>void</type>
```

```
    <number>000008</number>
```

```
    <amount>000000000100</amount>
```

```
</transaction>
```

```
<transaction>
```

```
    <type>purchase</type>
```

```
    <amount>000000012300</amount>
```

```
    <pan>4000000010000001</pan>
```

```
    <expired>1805</expired>
```

```
    <password>12345678</password>
```

```
    <cardholder>JOHN SMITH</cardholder>
```

</transaction>

- type - тип операции:
 - 1) purchase - оплата
 - 2) refund - возврат
 - 3) purchase-with-cashback - оплата с кэшбеком
 - 4) void - отмена
- amount - сумма (12 цифр копеек). Для транзакции void сумма может отсутствовать. В этом случае отмена производится на всю сумму отменяемой транзакции.
- amount-other - сумма кэшбэка (12 цифр копеек) для операции purchasewithcashback.
- currency - код валюты, если он отличается от кода валюты по умолчанию, указанного в конфигурации.
- rrn - retrieval refernce number (12 символов; необязательный параметр операции refund).
- number - номер чека (6 цифр).
- pan - номер карты. Указывается при вводе данных карты вручную.
- expired - окончание срока действия карты (4 цифры, ггмм). Указывается при вводе данных карты вручную.
- password - пароль администратора (8 цифр). Указывается при вводе данных карты вручную.
- cardholder - имя владельца карты.

Пример ответа:

<transaction>

<status>ok</status>

<receipt>

<header>

<line>Merchant Name</line>

<line>Merchant Address</line>

```

</header>
<tid>1000000001</tid>
<mid>012345678912345</mid>
<seq>000009</seq>
<type>purchase</type>
<state>active</state>
<tstatus>approved</tstatus>
<amount-authorized>
                                000000012300
</amount-authorized>
<amount-other>000000012300</amount-other>
<currency>643</currency>
<aid>A00000000031010</aid>
<appname>VISA Classic</appname>
<pan>*****1234</pan>
<aed>2003</aed>
<rrn> 120157645346</rrn>
<resp-code>000</resp-code>
<auth-number>AFK045</auth-number>
<datetime>180328120133</datetime>
<tsi>EF00</tsi>
<tvr>2040300000</tvr>
<decline-reason/>
<acquirer-tag>abc</acquirer-tag>
<bank-name>demo</bank-name>
<footer>
                                <line>Byte</line>
</footer>
<record>A1234DF3...</record>

```

</receipt>

<error-stack/>

</transaction>

- status - код ошибки.
- header - строки заголовка чека.
- tid - идентификатор (номер) терминала.
- mid - идентификатор владельца терминала (Merchant ID).
- seq - номер чека.
- type - тип транзакции.
- state - значение этого поля в чеке всегда active.
- tstatus - статус транзакции approved или declined.
- amount-authorized - сумма транзакции в копейках.
- amount-other - сумма кэшбека в копейках (для транзакции purchasewithcashback).
- currency - код валюты.
- aid - идентификатор приложения EMV карты (AID).
- appname - имя приложения EMV карты.
- pan - последние 4 цифры номера карты.
- aed - дата окончания срока действия карты (ГГММ).
- rrn - retrieval reference number.
- resp-code - код ответа сервера авторизации.
- auth-number - код авторизации.
- datetime - дата и время проведения транзакции (yymmddHHMMSS).
- tsi - значение EMV тэга Transaction Status Information.
- tvr - значение EMV тэга Terminal Verification Result.
- footer - строки внизу чека.

- `decline-reason` - значение этого тега используется только в случае если транзакция отклонена. Возможные значения: `unabletogoonline`, `offlinedeclined`, `systemerror`, `onlinedeclined`, `cardprocessingerror`.
- `acquirer-tag` - значение, используемое для привязки к платежной системе. Копируется в чек из конфигурации [JCS].
- `bank-name` – имя банка. Копируется в чек из конфигурации [JCS].

Транзакция успешно выполнена и одобрена, если в ответе имеется чек и тэг чека `tstatus = approved`. Чек в ответе может отсутствовать если значение тэга `status` отличается от `ok`.

Во время исполнения транзакции терминал может передавать сообщения `keepalive`, `display`, `dex`, `getonlinepin`, `getofflinepin`, `selectaid` и `getlanguage`. Если в ответ на сообщение `keepalive` терминалу возвращается `status=cancelled` терминал прерывает обработку транзакции. Сообщение `display` содержит информацию для отображения на экране. Код сообщения передается в теге `code`. Значение тега `code` указано в таблицах [EMVA, Table 9-5] [7] и [EMV4, Table 8] [8]. Дополнительно к указанным в этих таблицах используются следующие коды:

- D1 - подключение к хосту банка
- D2 - Повторное подключение
- D3 - Нет ответа от хоста банка
- D4 - Ответ получен
- D5 - Превышен счетчик попыток ввода ПИН-кода

<display>

<code>03</code>

<language>ruen</language>

<msg>ОДОБРЕНО</msg>

<status>02</status>

<hold-time>000000</hold-time>

```

    <value-qualifier>00</value-qualifier>
    <value>000000000000</value>
    <currency>643</currency>
</display>

```

В теге lang передается список предпочтительных языков, для отображения сообщения. Каждый язык представлен парой символов в формате ISO-639. Описание параметров сообщения приведено в [EMVC2, A.1.194]. Если value-qualifier = 10 (AMOUNT), то value содержит сумму транзакции для отображения на экране, а currency - код валюты транзакции.

Сообщение dex приходит от терминала во время исполнения транзакции, если это разрешено в общих настройках терминала [JCS] data-exchange = enabled. Кернел отправляет терминалу теги, указанные в EMV настройках [L2EMV]. Для указания списка тегов в конфигурации MCL используется тег DF8112 (Tags to Read), в остальных платежных системах тег DF811E (Data Exchange Tag List).

```

<dex>
    <kernel-tags>
        <tag name="5A" value=01234567890ABCDEF><code>
    </kernel-tags>
</dex>

```

Терминал должен вернуть в ответ status = ok и, если требуется, теги, которые кернел изменит/добавит в свои данные перед тем как продолжить исполнение транзакции. Формат значений этих тегов должен соответствовать спецификации EMV.

```

<dex>
    <status>ok</status>
    <terminal-tags>
        <tag name="9F06" value="000000001000"><code>
    </terminal-tags>

```

</dex>

Сообщение `getonlinepin` содержит запрос на ввод онлайн ПИН-кода. Клиент должен обеспечить безопасный ввод ПИН-кода и вернуть ПИН-блок в ответе на сообщение.

В параметрах сообщения передается PAN необходимый для формирования ПИН-блока:

<getonlinepin>

<pan>1234567890123456</pan>

</getonlinepin>

Ответ содержит зашифрованный рабочим РЕК ключом ПИН-блок в формате ISO-9564-0:

<getonlinepin>

<status>ok</status>

<pin-block>D42082B4AE20F603</pin-block>

</getonlinepin>

Рабочий ключ РЕК используется по умолчанию или задается в настройках приложения. Тег `status` должен содержать значение `ok` в случае успешного ввода ПИН-кода, `pinentrybypassed` в случае, если ввод ПИН-кода отменен пользователем или `pinpadmalfunctionornotpresent` в случае, если ввод ПИН-кода не возможен по техническим причинам.

Сообщение `getofflinepin` содержит запрос на ввод оффлайн ПИН-кода контактной карты. Клиент должен обеспечить безопасный ввод ПИН-кода и вернуть ПИН-блок в ответе на сообщение.

В параметрах сообщения передается счетчик оставшихся попыток ввода ПИН-кода `ptc`:

<getofflinepin>

<ptc>03</ptc>

</getofflinepin>

Ответ содержит зашифрованный сессионным ключом РЕК ключом ПИН-блок в формате ISO-9564-2:

```
<getofflinepin>
```

```
<status>ok</status>
```

```
<pin-block>D42082B4AE20F603</pin-block>
```

```
</getofflinepin>
```

Сессионный ключ РЕК используется по умолчанию или задается в настройках приложения. Клиент должен создавать новый сессионный ключ каждый раз перед вводом ПИН-кода.

Тег status должен содержать значение ok в случае успешного ввода ПИН-кода, pinentrybypassed в случае, если ввод ПИН-кода отменен пользователем или pinpadmalfunctionornotpresent в случае, если ввод ПИН-кода не возможен по техническим причинам.

Сообщение selectaid содержит запрос выбора приложения на контактной карте. Сообщение содержит список имен приложений составленный из тегов Preferred Name, Application Name или Application Identifier.

```
<selectaid>
```

```
<aids>
```

```
<aid>Mir</aid>
```

```
<aid>Maestro</aid>
```

```
</aids>
```

```
</selectaid>
```

Клиент должен обеспечить вывод на экран терминала списка приложений в том порядке, в котором они перечислены в сообщении, выбор одного из сообщений и передачу порядкового номера выбранного приложения в теге selected-aid в ответе на сообщение. Если сообщение не выбрано, (пользователь отменил выбор) следует вернуть 0.

```
<selectaid>
```

```
<status>ok</status>
```

```
<selected-aid>1</selected-aid>
</selectaid>
```

Сообщение getlanguage содержит запрос выбора языка для обработки транзакции по контактной карте. Сообщение содержит список предпочитаемых языков для отображения сообщений в формате ISO-639.

```
<getlanguage>
    <preferred-language>ruen</preferred-language>
</getlanguage>
```

Клиент должен обеспечить выбор языка и вернуть его в ответе на сообщение.

```
<getlanguage>
    <status>ok</status>
    <language>ru</language>
</getlanguage>
```

10) Отчет

Пример команды:

```
<report>
    <password>12345678</password>
    <report-type>brief</report-type>
</report>
```

- password - пароль (8 цифр)
- report-type - вид отчета (brief - краткий, full - полный)

Пример ответа:

```
<report>
    <status>ok</status>
    <xreport>
        <header>
            <mid>M123456789012345</mid>
```

```

<tid>1000000001</tid>
<title>
    <line1>Header line 1</line>
</title>
<cur>643</cur>
<time>189329120002</time>
</header>
<transactions>
    <t>
        <status>approved</status>
        <state>active</state>
        <type>refund</type>
        <seq>000009</seq>
        <aa>000000001000</aa>
        <ao>000000000100</ao>
        <rrn>647394847384</rrn>
        <time>180322121408</time>
        <apn>AFJ879</apn>
        <arc>000</arc>
        <cur>643</cur>
        <pan>*****1234</pan>
        <aex>2112</aex>
        <aid>A0000000031010</aid>
        <tvr>0000000000</tvr>
        <tsi>000000</tsi>
    </t>
    ...
</transactions>
<totals>

```

```

<card-type>
  <rid>A000000003</rid>
  <name>VISA</name>
  <purchase-count>100</purchase-count>1
  <purchase-sum>
    000000010000
  </purchase-sum>
  <refund-count>100</refund-count>
  <refund-sum>000000010000</refund-sum>
  <purchase-with-cashback-count>
    100
  </purchase-with-cashback-count>
  <purchase-with-cashback-sum>
    000000010000
  </purchase-with-cashback-sum>
  <purchase-reversal-count>
    100
  </purchase-reversal-count>
  <purchase-reversal-sum>
    000000010000
  </purchase-reversal-sum>
  <refund-reversal-count>
    2
  </refund-reversal-count>
  <refund-reversal-sum>
    000000010000
  </refund-reversal-sum>
  <total-card-sum>
    C000000010000

```

```

        </total-card-sum>
        <total-card-count>400</total-card-count>
    </card-type>
    ...
</totals>
<grand-totals>
    <purchase-total-count>100</purchase-total-count>1
    <purchase-total-sum>
        000000010000
    </purchase-total-sum>
    <refund-total-count>100</refund-total-count>
    <refund-total-sum>
        000000010000
    </refund-total-sum>
    <purchase-with-cashback-total-count>
        100
    </purchase-with-cashback-total-count>
    <purchase-with-cashback-total-sum>
        000000010000
    </purchase-with-cashback-total-sum>
    <purchase-reversal-total-count>
        100
    </purchase-reversal-total-count>
    <purchase-reversal-total-sum>
        000000010000
    </purchase-reversal-total-sum>
    <refund-reversal-total-count>
        2
    </refund-reversal-total-count>

```

```

        <refund-reversal-total-sum>
        0000000010000
        </refund-total-reversal-sum>
        <total-sum>C0000000010000</total-sum>
        <total-count>400</total-count>
    </grand-totals>
</xreport>
</report>

```

- status - код ошибки.
- tid - идентификатор (номер) терминала.
- mid - идентификатор владельца терминала (Merchant ID).
- title - строки заголовка чека.
- cur - код валюты.
- time - дата и время составления отчета (уymmddHHMMSS).
- transactions - список транзакций в полном (full) отчете. Для каждой транзакции указывается:
 - status - код ошибки.
 - state - значение этого поля в чеке всегда active.
 - type - тип транзакции.
 - seq - номер чека.
 - aa - сумма транзакции в копейках.
 - ao - сумма кэшбека в копейках (для транзакции purchasewithcashback).
 - rrn - retrieval reference number.
 - time - дата и время проведения транзакции (уymmddHHMMSS).
 - apn - код авторизации.
 - arc - код ответа сервера авторизации.
 - cur - код валюты.
 - rap - последние 4 цифры номера карты.

- aex - дата окончания срока действия карты (ГГММ).
- aid - идентификатор приложения EMV карты (AID).
- tvr - значение тэга terminal verification result.
- tsi - значение тэга transaction status information.
- totals - итоговые данные отчета сгруппированные по типу карты.

Для каждого типа карты указывается:

- rid - registered application provider identifier (RID) карты.
- name - название карты.
- purchase-count - количество операций оплата.
- purchase-sum - общая сумма всех операций оплата.
- refund-count - количество операций возврат.
- refund-sum - общая сумма всех операций возврат.
- purchase-with-cashback-count - количество операций оплата с кэшбэком.
- purchase-with-cashback-sum - общая сумма всех операций оплата с кэшбэком.
- purchase-reversal-count - количество операций отмены оплаты.
- purchase-reversal-sum - общая сумма всех операций отмены оплаты.
- refund-reversal-count - количество операций отмены возврата.
- refund-reversal-sum - общая сумма всех операций отмены возврата.
- total-card-count - общее количество операций по карте.
- total-card-sum - баланс всех операций по карте. Баланс это 12 цифр со знаком. Знак плюс обозначается символом 'D'. Знак минус символом 'C'.
- grand-totals - итоговые данные отчета:
- purchase-total-count - количество операций оплата.
- purchase-total-sum - общая сумма всех операций оплата.
- refund-total-count - количество операций возврат.

- refund-total-sum - общая сумма всех операций возврат.
- purchase-with-cashback-total-count - количество операций оплата с кэшбэком.
- purchase-with-cashback-total-sum - общая сумма всех операций оплата с кэшбэком.
- purchase-reversal-total-count - количество операций отмены оплаты.
- purchase-reversal-total-sum - общая сумма всех операций отмены оплаты.
- refund-reversal-total-count - количество операций отмены возврата.
- refund-reversal-total-sum - общая сумма всех операций отмены возврата.
- total-count - общее количество операций.
- total-sum - баланс всех операций. Баланс это 12 цифр.

11) Сверка итогов

Пример команды.

```
<settlement>
    <password>12345678</password>
</settlement>
```

Пример ответа

```
<settlement>
    <status>ok</status>
    <sreport>
        <resp-code>000</resp-code>
        <approval-number>ASD002</approval-number>
        <rrn>837495759322</rrn>
        <orig-amount>D003030001000</orig-amount>
        <amount>D003030001000</amount>
```



```

        <datetime>180322102354</datetime>
        <tid>1000000001</tid>
        <mid>123456789012345</mid>
        <cur>643</cur>
        <from>190101100000</from>
        <to>190102110000</to>
        <tnum>100</tnum>
        <batch-upload>passed</batch-upload>
        <art-resp-code>00</art-resp-code>
        <cov-resp-code>00</cov-resp-code>
        <cov-rrn>123456789012</cov-rrn>
        <settlement-result>passed</settlement-result>
    </sreport>
</settlement>

```

- status - код ошибки.
- resp-code - код ответа сервера авторизации.
- approval-number - код авторизации.
- rrn - retrieval reference number.
- orig-amount - итоговая сумма подсчитанная на терминале.
- amount - итоговая сумма переданная сервером авторизации.
- datetime - дата и время проведения транзакции (yyymmddHHMMSS).
- tid - идентификатор (номер) терминала.
- mid - идентификатор владельца терминала (Merchant ID).
- cur - код валюты.
- from - дата и время первой транзакции в сверке.
- to - дата и время последней транзакции в сверке.
- tnum - количество транзакций в сверке.

- batch-upload - в случае если суммы при сверке не совпали требуется загрузка транзакций. В этом теге указывается результат такой загрузки. passed - загрузка выполнена успешно, failed - загрузка транзакций не удалась. Этот тег может отсутствовать.
- art-resp-code - Код ответа на запрос завершающий загрузку транзакций. Этот тег может отсутствовать.
- sov-resp-code - код возврата операции очистки журнала. Тег включается в ответ, если после сверки итогов выполняется операция очистки журнала (cutover). Для этого в конфигурации надо указать настройку settlement cutover="true".
- sov-rtn - RRN операции очистки журнала. Присутствует в случае если операция очистки журнала успешно выполнена.
- settlement-result - результат операции сверки итогов. passed - сверка завершена успешно. failed - операция не выполнена.

12) Очистка журнала

Эта операция удаляет все записи из локального журнала транзакций и выполняет операцию Cutover.

Пример команды.

```
<clear>
    <password>12345678</password>
</clear>
```

Пример ответа

```
<clear>
    <status>ok</status>
    <sreport>
        <resp-code>000</resp-code>
        <approval-number>ASD002</approval-number>
```

```
<rrn>837495759322</rrn>
<orig-amount>D003030001000</orig-amount>
<amount>D003030001000</amount>
<datetime>180322102354</datetime>
<tid>1000000001</tid>
<mid>123456789012345</mid>

</sreport>

</clear>
```

- status - код ошибки.
- resp-code - код ответа сервера авторизации.
- approval-number - код авторизации.
- rrn - retrieval reference number.
- orig-amount - итоговая сумма подсчитанная на терминале.
- amount - итоговая сумма переданная сервером авторизации.
- datetime - дата и время проведения транзакции (yyymmddHHMMSS).
- tid - идентификатор (номер) терминала.
- mid - идентификатор владельца терминала (Merchant ID).

13) Сброс пароля

Пример команды:

```
<resetpassword/>
```

Пример ответа:

```
<resetpassword>
    <status>ok</status>
</resetpassword>
```

14) Инициализация терминала

Для запуска процедуры инициализации терминала используется команда `runreset`.

```
<runreset>
```

```
<token>DE9773A8CB888560AB0F89C07623FE03</token>
```

```
</runreset>
```

Приложение ожидает подключение программы активатора на порт 4434 и команду `reset` от активатора. Приложение периодически отправляет по каналу команды `runreset` сообщение

```
<keep-alive>
```

```
<reset-connected-indicator>0</reset-connected-indicator>
```

```
</keep-alive>
```

Значение тега `reset-connected-indicator` устанавливается в 1 когда к терминалу подключается активатор. Если в ответ на сообщение `keepalive` получен код ошибки, отличный от `ok` процедура инициализации прерывается.

Команда `reset`, в отличие от других команд, обрабатывается только в случае, если она передается через порт 4434.

```
<reset/>
```

В ответ на команду `reset` приложение

1. Сбрасывает свои настройки
2. Генерирует новый мастер ключ
3. Генерирует RSA ключ клиентского сертификата для сервера конфигураций
4. Отправляет сообщение

```
<signconfigclientcertificate>
```

```
<key>1234ABCD...</key>
```

```
<device-identifier>IMS0000000009</device-identifier>
```

```
</signconfigclientcertificate>
```

- key - Публичный RSA ключ для создания сертификата устройства.
- device-identifier - Символьный идентификатор устройства, включающий его серийный номер.

5. В ответ на это сообщение активатор возвращает:

```
<signconfigclientcertificate>
    <signed-cert>
        <cert>AB12C4ED ....</cert>
        <ca>56AE54C6 ...</ca>
    </signed-cert>
</signconfigclientcertificate>
```

- cert - подписанный сертификат терминала для доступа к серверу конфигураций. Сертификаты передаются в формате PEM. Двоичные образы PEM преобразуются в HEX ACSII и помещаются в тэги cert и ca.
- ca - корневой сертификат, которым подписан cert.

6. Терминал генерирует RSA ключ клиентского сертификата для загрузчика ключей и запрашивает у активатора этот сертификат, отправляя ему сообщение

```
<signkeyloaderclientcertificate>
...
</signkeyloaderclientcertificate>
```

Формат этого сообщения и формат ответа такие же как для сообщения signconfigclientcertificate. Ответ содержит сертификат клиента загрузчика ключей и корневой сертификат, которым он подписан.

7. Терминал генерирует RSA ключ клиентского сертификата для сервера авторизации и запрашивает у активатора этот сертификат, отправляя ему сообщение

```
<signacquirerclientcertificate>
...
```

</signacquirerclientcertificate>

Формат этого сообщения и формат ответа такие же как для сообщения *signconfigclientcertificate*. Ответ содержит сертификат клиента для сервера авторизации и корневой сертификат, которым он подписан. Таким образом, для каждого из трех серверов создается отдельный клиентский сертификат [11].

8. Терминал отправляет активатору запрос параметров сервера конфигурации

<queryconfigcredentials/>

9. Активатор возвращает:

<queryconfigcredentials>

<confdata>

<cert>AB1234...</cert>

<sign>CDEF5678..<sign>

<url>https://myconfig.server</url>

</confdata>

</queryconfigcredentials>

- *cert* - корневой сертификат сервера конфигураций.
- *sign* - цифровая подпись сертификата *cert*. Эта подпись проверяется публичным RSA ключом, встроенным в код приложения.
- *url* - URL сервера конфигураций.

10. На этом инициализация заканчивается. Приложение закрывает соединение с активатором и возвращает код ошибки в ответ на команду *runreset*:

<runreset>

<status>ok</status>

</runreset>

2.3 Разработка архитектуры системы

Разрабатываемая система будет представлять собой два консольных приложения и одно приложения для Arduino. Приложение для Arduino будет при включении выполнять начальную настройку, а дальше будет считывать данные с интерфейса USB, и в случае появления данных будет приводить в движение роботизированную руку посредством приведения в движение сервоприводов для прикладывания карты к терминалу. Одно из консольных приложений будет играть роль клиента к Arduino, которое подключается к Arduino и в необходимое время посылает данные по USB. Второе приложение будет клиентом к терминалу, первым делом оно выполняет подключение к терминалу, считывает все тесты из конфигурационного файла, которые должны быть сделаны, далее отправляет команду терминалу с типом транзакция, следующим шагом приложение дает команду клиентскому приложению для Arduino отправить данные, чтобы роботизированная рука поднесла карту к терминалу, далее приложение ждет, когда банковское приложение вернет ответ, анализирует этот результат и в случае успешного теста, продолжает тестирование терминала, в ином случае останавливается и выводит сообщение ошибки от терминала.

На рисунке 10 представлена архитектура разрабатываемой системы.

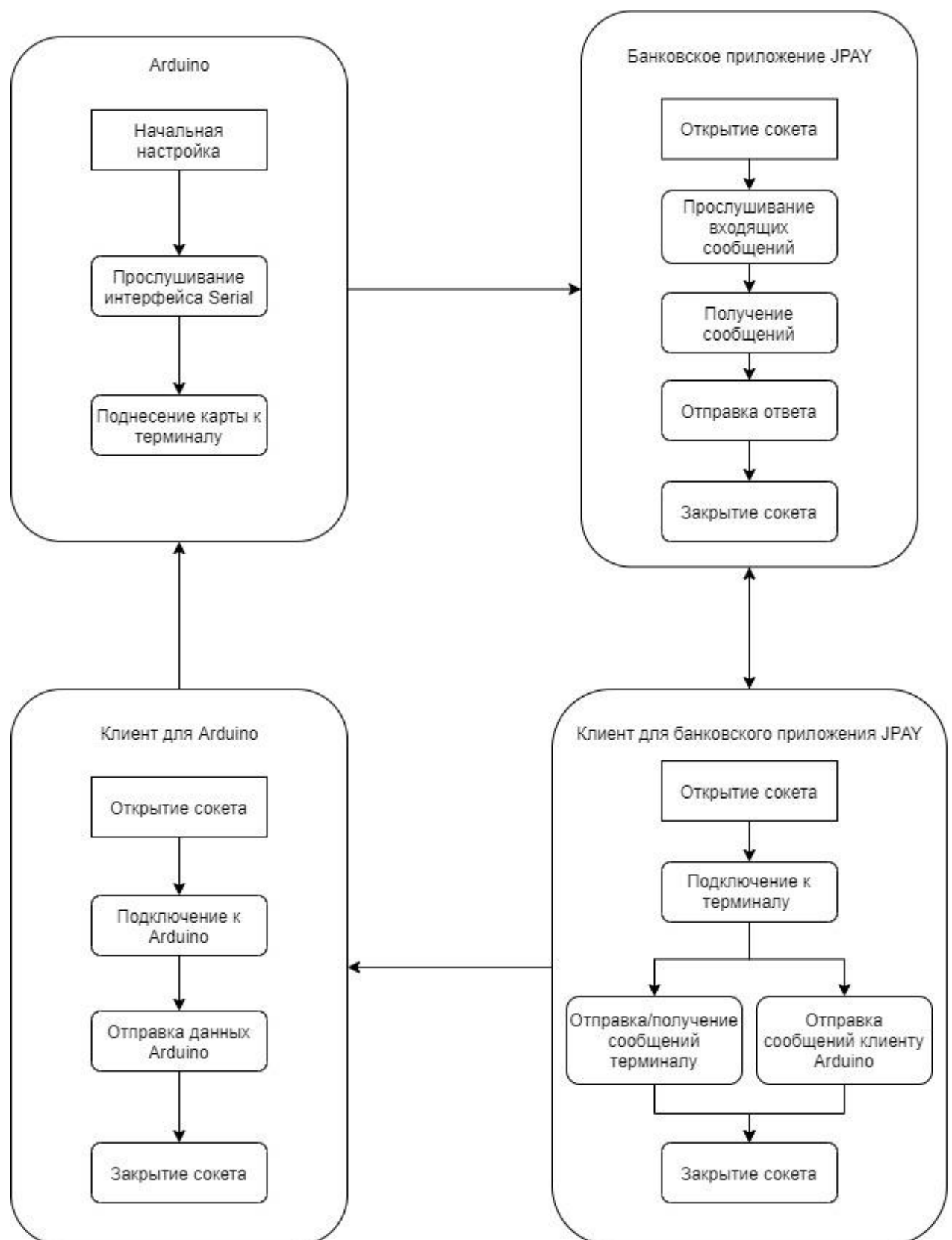


Рисунок 10 – Архитектура разрабатываемой системы

Разработанная архитектура помогает понять структуру системы, принцип взаимодействия между ее компонентами, а также продумать план разработки системы.

2.4 Выбор необходимых инструментов программирования

2.4.1 Выбор языка программирования и фреймворка

Язык программирования будет выбираться исходя из особенностей разрабатываемой системы и имеющихся инструментов для данного языка в виде библиотек или фреймворком.

Фреймворк – многоуровневая структура, показывающая, какие программы можно или нужно создавать и как они будут взаимосвязаны. Некоторые каркасы компьютерных систем также включают в себя реальные программы, задают программные интерфейсы или предлагают инструменты программирования для использования каркасов. Каркас может быть для набора функций в системе и того, как они взаимосвязаны; уровни операционной системы; уровни прикладной подсистемы; как коммуникация должна быть стандартизирована на некотором уровне сети; и так далее. Структура, как правило, является более всеобъемлющей, чем протокол, и более директивной, чем структура.

Наиболее популярные языки программирования и их фреймворки представлены в таблице 1.

Таблица 1 – Языки программирования и фреймворки

Язык программирования	Фреймворк
Golang	Echo
	Gin
C#	.NET
Python	Django
	Flask
Java	Spring
	Spark

В качестве языков разработки было выбрано два языка Golang и Си.

Язык Си был выбран, так как это самый распространенный язык программирования для Arduino и у него есть библиотека для работы с сервоприводами <Servo.h>.

Язык Golang был выбран для разработки клиентских приложений, так как из коробки он имеет все необходимое для нашей работы, к тому же он быстро работает и имеет хорошую поддержку многопоточности. Многопоточность нам необходима для работы с сообщениями от терминала, терминал может отправить не ответ на транзакцию, а служебные сообщения, которые мы должны либо обрабатывать, либо игнорировать в другом потоке, а в главном потоке ждать сообщение с типом транзакция. Также разработка на языке Golang занимает меньше времени, в результате мы на выходе получаем быстро разрабатываемую программу с хорошей поддержкой и быстрой скоростью работы.

2.4.2 Выбор среды разработки

Интегрированная среда разработки (IDE) - это программное обеспечение для создания приложений, которое объединяет общие инструменты разработчика в единый графический интерфейс пользователя (GUI). IDE обычно состоит из:

- Редактор исходного кода: текстовый редактор, который может помочь в написании программного кода с такими функциями, как подсветка синтаксиса с помощью визуальных подсказок, обеспечение автозавершения для конкретного языка и проверка ошибок во время написания кода.

- Локальная автоматизация сборки. Утилиты, которые автоматизируют простые повторяющиеся задачи в рамках создания локальной сборки программного обеспечения для использования разработчиком, например, для компиляции исходного кода компьютера в двоичный код, упаковки двоичного кода и запуска автоматических тестов.
- Отладчик: программа для тестирования других программ, которые могут графически отображать местоположение ошибки в исходном коде. Для разработки программного обеспечения .NET существует один из самых серьезных в этой сфере продукт производства Microsoft, который называется Visual Studio.

Для разработки под Android был выбран Arduino IDE, так как это официальный продукт для разработчиков под Arduino, который поддерживает почти все платы и включает в себя все необходимое для разрабатываемой системы.

Для разработки клиентских приложений на языке Golang в данной работе был выбран Visual Studio Code, который представляет из себя простой редактор кода без отладчика и локального сборщика кода, так как стандартные команды Linux предоставляют нам все эти возможности.

2.5 Выбор серверной платформы

В реализации данной работы используется клиент-серверная архитектура.

Клиент-серверная архитектура - это компьютерная архитектура производитель / потребитель, в которой сервер выступает в роли производителя, а клиент - в качестве потребителя. Сервер содержит и предоставляет высокопроизводительные вычислительные услуги клиенту по

требованию. Эти услуги могут включать доступ к приложениям, хранение, совместное использование файлов, доступ к принтеру и / или прямой доступ к необработанным вычислительным ресурсам сервера.

Клиент-серверная архитектура работает, когда клиентский компьютер отправляет запрос ресурса или процесса на сервер через сетевое соединение, которое затем обрабатывается и доставляется клиенту. Серверный компьютер может одновременно управлять несколькими клиентами, тогда как один клиент может быть подключен к нескольким серверам одновременно, каждый из которых предоставляет свой набор услуг. В своей простейшей форме Интернет также основан на архитектуре клиент / сервер, где веб-серверы одновременно обслуживают множество пользователей данными веб-сайта.

Для реализации данной архитектуры используется взаимодействие посредством протокола передачи данных TCP, работающим на транспортном уровне модели OSI/ISO.

Transmission Control Protocol (TCP) - протокол связи, ориентированный на установление соединения, который облегчает обмен сообщениями между вычислительными устройствами в сети. Это наиболее распространенный протокол в сетях, которые используют Интернет-протокол (IP); вместе они иногда упоминаются как TCP / IP.

TCP принимает сообщения от приложения / сервера и разделяет их на пакеты, которые затем могут быть отправлены устройствами в сети - коммутаторами, маршрутизаторами, шлюзами безопасности - в пункт назначения. TCP нумерует каждый пакет и собирает их перед тем, как передать их получателю приложения / сервера. Поскольку он ориентирован на установление соединения, он обеспечивает установление и поддержание соединения до завершения обмена между приложением / серверами, отправляющими и получающими сообщение.

Выводы

В ходе выполнения конструкторской части был разобран интерфейс взаимодействия с банковским приложением JPAУ. Был проведен анализ существующий конструкций роботизированных рук на рынке и выбрано лучшее решение. Далее была разработана архитектурна будущей системы и выбраны наиболее подходящие языки программирования и инструменты разработки.

3 Технологическая часть

3.1 Разработка программной части

Первым делом разберем программу для платы Arduino. Программа имеет две части функции `setup()` и `loop()`.

`Setup()` производит начальную настройку и инициализацию сервоприводов, а также инициализирует последовательное соединение и задает скорость передачи данных. Пример фрагмента кода:

```
void setup() {  
    Serial.begin(9600);  
    Servo1.attach(10);  
    Servo1.write(80);  
}
```

`Serial.begin(9600)` открывает последовательный порт и устанавливает скорость 9600 бит/с, `Servo1.attach(10)` подключает первый сервопривод к 10 выводу и осуществляет управление приводом, `Servo1.write(80)` передает значение 80 в ранее указанный вывод для изменение угла наклона роботизированной руки.

`Loop()` выполняет в бесконечном цикле проверку значений в последовательном соединении и при появлении данных изменяет положение роботизированной руки в нужное положение и очищает буфер последовательного интерфейса. Пример фрагмента кода:

```
void loop() {  
    if (Serial.available() > 0) {  
        ... // изменение значений сервоприводов  
        Serial.read();  
    }  
}
```

`Serial.available` получает данные доступные для чтения из последовательного интерфейса. При появлении данных происходит изменение положения роботизированной руки и очистка буфера.

Вторая программа является клиентом для платы Arduino, которая записывает данные в последовательный интерфейс, когда об этом сигнализирует третья программа, но о ней позже. Для начала этой программе необходимо подключиться к интерфейсу для передачи данных. Эта задача производится с помощью библиотеки `sio` и функции `sio.Open(arg1, arg2)`, первый аргумент – это устройство в системе, в нашем случае порт ввода/вывода USB, вторым аргументом является скорость передачи данных по этому интерфейсу. В результате функция возвращает порт, по которому мы можем записывать данные.

Второй функцией этой клиентской программы выступает непосредственно запись данных в порт, которая осуществляется с помощью функции `port.Write([]byte("somedata"))`. Нам не важно какие данные будут записываться, главное их наличие в последовательном интерфейсе в нужный момент.

Третья программа является главной программой во всех взаимодействиях, она осуществляет подключение к терминалу, считывает транзакции из конфигурационного файла, посылает транзакции терминалу, вызывает функцию клиентское программы к плате Arduino, чтобы осуществить поднесение роботизированной руки к терминалу, а также обрабатывает ответ от терминала.

Для начала программа подключается к терминалу по протоколу TCP для реализации клиент-серверного взаимодействия, где терминал выступает в роли сервера, а наша программа в виде клиента. Эта задача решается функцией из стандартного пакета `net`, а конкретнее `net.Dial("tcp", viper.GetString("terminalIP")+viper.GetString("terminalPort"))`, данная функция открывает сокет, подключается к сокету терминала для обмена данными между ними и возвращает `net.Conn` (абстракция соединения сокетов), через которое в дальнейшем будут отправляться данные и приниматься ответы от терминала.

Далее программа считывает из заранее подготовленного конфигурационного файла транзакции, которые являются тестовыми случаями для терминала. Фрагмент конфигурационного файла:

```
<transactions>
  <transaction>
    <type>purchase</type>
    <currency>643</currency>
    <amount>000000032100</amount>
  </transaction>
  <transaction>
    <type>purchase</type>
    <currency>643</currency>
    <amount>000000021300</amount>
  </transaction>
</transactions>
```

Конфигурационный файл является списком транзакций с настройками для каждой транзакции в виде хaml разметки. Это выполнено специально для удобства, так как банковская программа JPAУ принимает данные в виде хaml разметки, парсит и обрабатывает их. Также, так как в стандартном пакете языка Golang encoding/xml есть парсер xml, то были подготовлены структуры данных для взаимодействия.

```
type Transactions struct {
    XMLName xml.Name      `xml:"transactions"`
    Items   []TransactionRequest `xml:"transaction"`
}
```

Структура Transactions необходима для парсинга конфигурационного файла.

```

type TransactionRequest struct {
    XMLName xml.Name `xml:"transaction"`
    Type    string  `xml:"type"`
    Currency string  `xml:"currency"`
    Amount  string  `xml:"amount"`
}

```

Структура TransactionsRequest необходима для парсинга запроса терминалу.

```

type TransactionResponse struct {
    XMLName xml.Name `xml:"transaction"`
    Status   string  `xml:"status"`
    Receipt  Receipt `xml:"receipt"`
    ErrorStack ErrorStack `xml:"error-stack"`
}

```

```

type Receipt struct {
    Tid    string `xml:"tid"`
    Mid    string `xml:"mid"`
    State  string `xml:"state"`
    Tstatus string `xml:"tstatus"`
    Currency string `xml:"currency"`
    RespCode string `xml:"resp-code"`
}

```

```

type ErrorStack struct {
    XMLName xml.Name `xml:"error-stack"`
    Error   string  `xml:"error"`
}

```

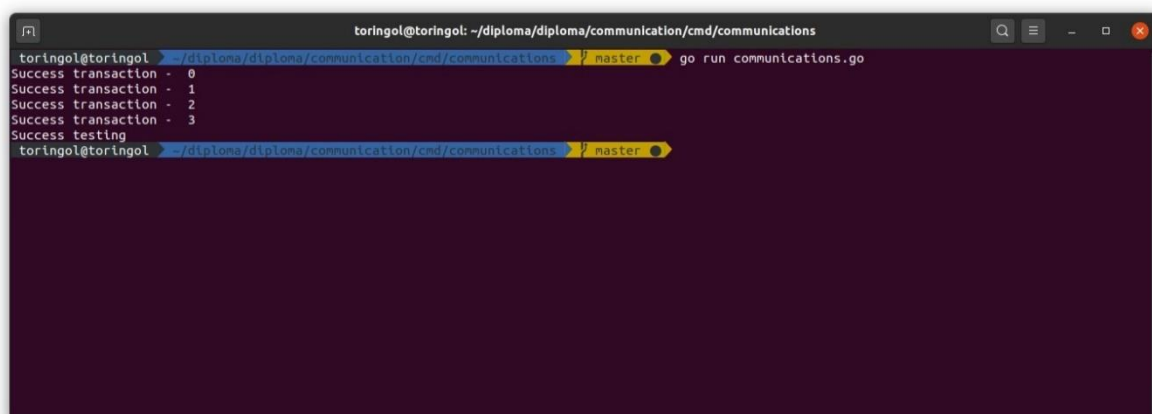
Структура TransactionResponse необходима для парсинга ответа от терминала для последующего анализа.

После считывания всех транзакций с конфигурационного файла программа синхронно пробегается по всем транзакциям в цикле, на каждой итерации отправляется пакет транзакции, тип которой терминал в дальнейшем будет ожидать, когда терминал включается для принятия транзакции с нужными настройками подается сигнал второй программе для поднесения карты к терминалу, дальше запускает горутину (асинхронная функция), которая читает ответ, пока не придет нужным нам ответ о прохождении транзакции. Когда приходит нужный ответ он обрабатывается. В случае ошибки выводится сообщение об ошибке и тестирование приостанавливается.

Методика и результаты тестирования

Для проверки корректности работы разработанной системы было произведено тестирование с заранее подготовленными данными.

Пример корректного вывода при тестировании 4 транзакций:



```
toringol@toringol: ~/diploma/diploma/communication/cmd/communications
Success transaction - 0
Success transaction - 1
Success transaction - 2
Success transaction - 3
Success testing
toringol@toringol: ~/diploma/diploma/communication/cmd/communications
```

Рисунок 11 – Корректный вывод системы

3.2 Выводы

В технологической части была разработана система по ранее составленной архитектуре. В итоге было разработано 3 программы (1 программа для платы Arduino, 2 программа клиент к плате Arduino, 3 программа разработана для клиент-серверного взаимодействия с банковской программой JPAУ), которые корректно работают, что показывает моделирование и тесты.

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы, целью которой было разработать устройство автоматического тестирования платежных терминалов, предназначенное для разработчиков банковских программ, были выполнены все поставленные ранее задачи.

Сначала была изучена общая информация об электронных платежах и терминалах, а также было изучено общая структура плат Arduino. Далее было изучены прикладная информация для разрабатываемой системы. Был изучен программный интерфейс банковского приложения JPAУ, были разобраны все возможные команды. Был произведен анализ рынка для поиска готового решения для манипулятора движений, была подобрана плата для работы манипулятора.

В последней части была разработана архитектура разрабатываемой системы, выбраны необходимые и самые удобные инструменты для ее реализации. В результате был сделан минимально жизнеспособный продукт для полностью автоматизированного тестирования терминалов, что в дальнейшем может помочь более быстрой разработке и сертификации терминалов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Credit Card Processing: How it Works [Электронный ресурс]: статья — Режим доступа: <https://www.cardfellow.com/blog/how-credit-card-processing-works>, свободный.
2. How Electronic Payment Work [Электронный ресурс]: статья — Режим доступа: <https://money.howstuffworks.com/personal-finance/onlinebanking/electronic-payment1.htm>, свободный.
3. How Credit Cards Work [Электронный ресурс]: статья — Режим доступа: <https://money.howstuffworks.com/personal-finance/debt-management/credit-card10.htm>, свободный.
4. Point-of-Sale Terminal [Электронный ресурс]: статья — Режим доступа: <https://www.investopedia.com/terms/p/point-of-sale-terminal.asp>, свободный.
5. Introduction to the Arduino – What is Arduino [Электронный ресурс]: статья — Режим доступа: <https://www.seeedstudio.com/blog/2019/12/04/introduction-to-the-arduino-what-is-arduino/>, свободный.
6. EMVA, EMV Contactless Specifications For Payment Systems [Электронный ресурс]: документация — Режим доступа: https://www.emvco.com/wp-content/uploads/2017/05/Book_A_Architecture_and_General_Rqmts_v2_6_Final_20160422011856105.pdf, свободный.
7. EMV4, EMV Integrated Circuit Card Specifications for Payment Systems [Электронный ресурс]: документация — Режим доступа: https://www.emvco.com/wp-content/uploads/2017/05/EMV_v4.3_Book_4_Other_Interfaces_20120607062305603.pdf, свободный.
8. EMVC2, EMV Contactless Specifications for Payment Systems [Электронный ресурс]: документация — Режим доступа: https://www.emvco.com/wp-content/uploads/2017/05/C-2_Kernel_2_V2.6_final-for_EMVCo_20160923093817522.pdf, свободный.
9. Перлин Ю.В. Пластиковые карты [Текст]: Ю.В. Перлин, Д.В. Сахаров — М: Издательская группа «БПЦ-пресс», 2005.

10. L2EMV, EMV настройки терминала [Текст]: LIBL2.
11. JCS, Настройки банковского приложения JPAY [Текст]: JCS.

ПРИЛОЖЕНИЕ А

Графическая часть выпускной квалификационной работы

В графическую часть выпускной квалификационной работы входят:

- актуальность работы;
- цель и задачи;
- схема электронной платежной системы;
- используемые инструменты и технологии;
- интерфейс банковского приложения JPAУ;
- архитектура разрабатываемой системы;
- алгоритм работы системы;
- тестирование системы;