МУНИЦИПАЛЬНОЕ БЮДЖЕТНОЕ ОБЩЕОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ СРЕДНЯЯ ОБЩЕОБРАЗОВАТЕЛЬНАЯ ШКОЛА №12

Инженерно-техническое направление

проект

ЭЛЕКТРОННЫЙ ПОВОДЫРЬ

«Умные очки»

Работу выполнила команда MEony:  
  
Соловьев Егор Юрьевич, уч.9 кл. МБОУ СОШ №12

**Научные руководители**:

Ефимов Павел Андреевич,

Руководитель направления Робототехники «Центра проектной деятельности» ДВФУ,

т. 8-914-402-69-44

Артемьева Олеся Григорьевна,

руководитель ШНОУ «Интеллект» МБОУ СОШ №12

т.8-929-407-40-07

Хабаровск 2019

# Содержание

[Содержание 2](#_Toc523397986)

[Введение 3](#_Toc523397987)

[Существующие решения 3](#_Toc523397988)

[Концепция проекта 4](#_Toc523397989)

[Техническое задание 5](#_Toc523397990)

[Организация рабочего процесса 5](#_Toc523397991)

[Разработка первого прототипа 8](#_Toc523397992)

[Причина разработки 8](#_Toc523397993)

[Получение и передача данных с Arduino Pro Mini 8](#_Toc523397994)

[Передача данных между процессами 9](#_Toc523397995)

[Воспроизведение аудио файлов 9](#_Toc523397996)

[Результат 10](#_Toc523397997)

[Создание устройства 11](#_Toc523397998)

[Моделирование «умных очков» 11](#_Toc523397999)

[Распознавание объектов 12](#_Toc523398000)

[База данных 14](#_Toc523398001)

[Распознавание голоса 15](#_Toc523398002)

[Результаты исследования работы кода для распознавания объектов 17](#_Toc523398003)

[Создание браслета 17](#_Toc523398004)

[Тестирование устройство 18](#_Toc523398005)

[План тестирования 18](#_Toc523398006)

[Результаты тестирования 18](#_Toc523398007)

[Дальнейшее развитие проекта 19](#_Toc523398008)

[Изменение устройства 19](#_Toc523398009)

[Экономическая часть 19](#_Toc523398010)

[Результат 20](#_Toc523398011)

[Приложение 20](#_Toc523398012)

# Введение

В России 108 тысяч человек страдают нарушением зрительного и визуального восприятия. Это число может в скором времени увеличиться в разы, так как ежедневно люди пренебрегают своим здоровьем, работая с компьютерной и мобильной техникой больше положенной нормы. Более того предрасположенность к некоторым болезням глаз передается на генетическом уровне, дефективность которого на данный момент исправить сложно. Ситуация обостряется тем, что инфраструктуры для таких людей очень мало, а именно: указатели со шрифтом Брайля, системы звукового оповещения, отсутствие наземных тактильных указателей. Они не могут жить полноценной жизнью и посещать свободно общественные места без лишних неудобств. По словам Яны Лантратовой, Председателя Координационного совета Всероссийской общественной организации «Союз добровольцев России», члена Президиума Совета при Президенте РФ по развитию гражданского общества и правам человека, только 32% инфраструктуры приспособлены для использования инвалидами по зрению. Поэтому даже простой поход в магазин для них – это непростое испытание. Из-за чего людям с дефектом зрения приходится посещать только знакомые места и ограничивать свой выбор. Чтобы исправить это и убрать поход в продуктовый магазин из списка невыполнимых задач, не изменяя инфраструктуры, мы решили разработать устройство электронный поводырь. Очки для слепых и слабовидящих людей.

# Существующие решения

На данный момент компаний, которые выпускают устройства для помощи ориентации в помещение, очень мало. Готовых устройств, которые уже можно заказать, единицы. Из того, что мы нашли, заказать можно только продукты от компаний: Oriense, Assisted Vision Smart Glasses, OrCam, Intel Edison. Все они выпускают «умные очки» с похожим функционалом, различия между устройствами представлены в таблице 1.

Таблица 1. Сравнение параметров аналогов

Большинство устройств позволяет перемещаться в помещение, не врезаясь в препятствия, имеют систему аудио оповещения и могут распознавать объект. Данного функционала достаточно для расширения возможности слабовидящего или слепого человека. Также они пользуются большим спросом в странах Европы и Америке. При этом устройства имеют ряд недостатков, что не позволяет использовать их постоянно и в России.

Во-первых, высокая ценна, стоимость большинства из них выше пособия по инвалидности 1 группы в 10 раз, 178$ по данным на 21.08.2018. Из-за этого люди не могут позволить купить себе устройство даже с минимальным функционалом, финансов хватает только на Intel Edition. Мало серийный аппарат приспособленный только для слабовидящих.

Во-вторых, размер. Очки, которые должны заменить человеку зрение, должны обрабатывать видеопоток и производить сложные вычисления, для этого нужны вычислительные мощности, а это занимает определенное место. Два из четырех устройств весят больше 600 грамм и внешне похожи больше на шлем, чем на очки, Oriense и Smart Glasses, рисунок 1.



Рисунок 1. Oriense и Smart Glasses

Способ обработки данных сразу на очках прост, но требует увеличение размера устройства.

В-третьих, функционал. Несмотря на то, что функционал большинство устройств обширен, позволяет обходить препятствия, распознавать людей, текст, но у них нет конкретных задач для выполнения. Если пойти в таких очках в больницу или магазин, то они не помогут вам найти кабинет или продукты.

# Концепция проекта

Разобрав ошибки существующих разработок, мы решили создать свои «умные очки» для помощи ориентации слепым людям в помещение, конкретнее в магазинах самообслуживания. Процесс создания устройства мы начали с разработки концепции работы готового продукта.

Перед тем как запустить «умные очки», мы строим карту помещения, чтобы затем прокладывать маршрут для человека. В начале работы пользователь должен произнести название продуктов, которые он собирается приобрести. Затем голосовой помощник повторяет список товаров и ждет утверждения от пользователя. Устройство, используя карту, находящуюся на сервере, прокладывает кратчайший маршрут до всех товаров. Произнося аудиосообщения в наушники, мы просим человека пройти в определенную сторону, параллельно с помощью вибромоторов сообщаем пользователю о препятствиях слева, справа и перед ним. Как только человек оказывается рядом с товаром, аудио ассистент просит его повернуться в том направление, где располагаются продукты. Камеры на «умных очках» ищут на ценниках QR код, по которому мы определяем наш ли это товар. Если это так, то камеры начинают искать руку человека и сам продукт. Аудио помощник сообщает о том, что товар перед пользователем, и с помощью, установленных на браслете вибромоторов, помогаем совместить руку и товар. Повторяем процесс до тех пор, пока товары в списке не закончатся. После помогаем человеку аудио сообщениям дойти до кассы. Работа устройства на этом заканчивается.

# Техническое задание

Для воплощения идеи, описанной выше, было составлено техническое задание. Оно включает в себя следующие пункты.

1. Два часа автономной работы

Если считать, что среднее время, проведенное в гипермаркете, равняется одному часу, то одна процедура заряда аккумулятора позволит использовать устройство дважды.

1. Хранение в базе данных до 1000 единиц товаров

В гипермаркете количество товаров в базе может доходить редко превышает 1000 видов, поэтому нет смысла расходовать ограниченную память.

1. Стоимость устройства до 400$

В России большинство инвалидов по зрению не получают зарплату в несколько раз выше МРОТ, 140$ на состояние 21.08.2018. Поэтому, если считать МРОТ с размером пенсии за инвалидность 1 группы, то выходит 318$, что позволит при накоплении средств, в течение года приобрести одно устройство. А компаниям приобретать партию «умных очков» за цену ниже цены одного заграничного аналога, что снизит стоимость аренды.

1. Вес до 300 грамм

Пользователь будет использовать очки в течение часа, поэтому они должны весить не больше, чем шлем виртуальной реальности от компании HTC. Кроме того очки должны быть компактны и удобны.

# Организация рабочего процесса

Для того чтобы создать, придуманное нами устройство, нужны обширные знания из разных областей, которыми мы не обладали. Кроме того время было сильно ограничено 1 месяцем. Для решения этой первой и очень важной проблемы мы внедрили систему Scrum менеджмента, взяв основную концепцию и изменив некоторые идеи для нашей команды.

Scrum - это методология управления проектами, которая построена на принципах тайм-менеджмета. Scrum менеджмент включает в себя команду Scrum Team, которая состоит из Scrum Master, Product Owner, Development Team.

Так как наша команда работает за идею и на себя, у нас нет человека, перед которым мы отчитываемся, то из Scrum Team мы убрали такого игрока, как Product Owner (заказчик). Роль Scrum Master занял капитан команды, так как уже имел опыт работы с этой системы. Development Team – вся команда.

Чтобы успеть сдать, проект мы разделили рабочий проект на 3 спринта. Спринт - это список работ, который определила и согласовала команда, на ближайший отчетный период. Каждый спринт длился 10 дней и заканчивался демонстрацией работы, готового продукта, на рисунке 2 представлены баннеры с задачами на каждый спринт.

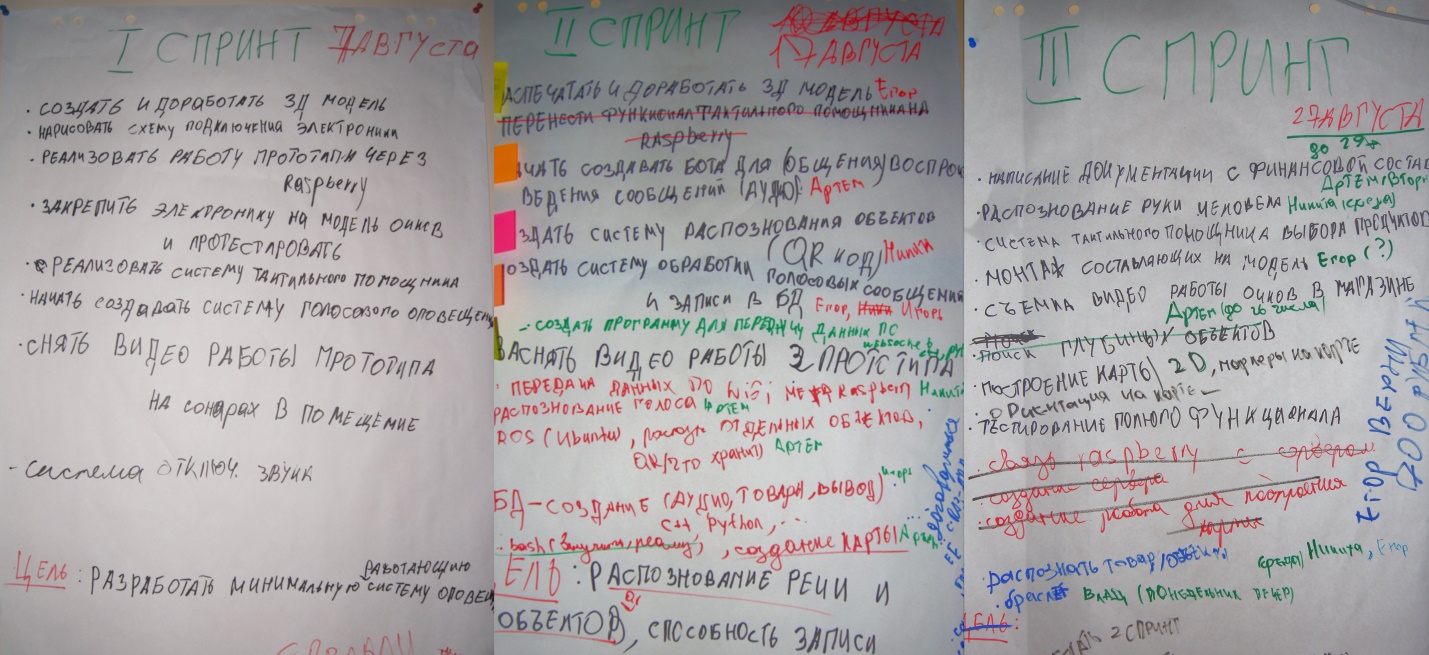


Рисунок 2. Scrum баннеры с задачами

Первый спринт. Цель первого спринта было создание прототипа, с помощью которого можно было перемещаться в помещение, не встречая препятствия. Прототип должен был работать на Arduino и Raspberry Pi 3, ориентируясь только с помощью датчиков. Для выполнения спринта мы поставили задачи:

1. Создать и доработать 3D модель
2. Нарисовать схему подключения электроники
3. Реализовать работу прототипа через Raspberry Pi
4. Закрепить электронику на модель протестировать
5. Реализовать систему тактильного помощника
6. Начать создавать систему голосового оповещения
7. Снять видео работы прототипа
8. Реализовать отключение аудио с помощью кнопки

Каждый сам выбирал себе задачу и расписывал её, размещая в последствие на Scrum доске. Каждое утро рядом с ней проходил сбор, где мы обсуждали, что будем делать сегодня, что сделали вчера, почему мы не выполнили определенные задачи и пытались их там решить, на рисунке 3 представлена Scrum доска на момент третьего спринта. Подробнее о первом прототипе и принципе работы написано в пункте «Разработка первого прототипа».

Второй спринт. Цель второго этапа – это создание системы записи списка продуктов в базу данных с помощью голосовых команд и нахождения нужных товаров на полке. Подробнее принцип работы каждой системы расписаны в пункте «Создание устойства». Задачи, поставленные для выполнения цели:

1. Напечатать и доработать 3D модель
2. Начать создавать голосового помощника
3. Написать код для распознавания QR кодов
4. Написать код для распознавания голоса
5. Создать БД и написать код для взаимодействия с ней
6. Написать код для передачи данных между процессами
7. Построить 2D карту помещения



Рисунок 3. Scrum доска

У каждого спринта было время выполнения и из-за того, что детали шли долго, некоторым членом команды пришлось покинуть город на определенное время, мы просрочили deadline. В нашей системе нет способов наказания за такое, поэтому мы изменили принцип работы. Мы начали более подробно расписывать задачи, чтобы не было ситуаций, когда пустует доска, а спринт не готов. Все чаще начали распараллеливать задачи. Сократили количество выходных за спринт до 0,5 дней вместо 2.

Третий спринт. Цель последнего этапа – собрать конечное устройство и протестировать его в одном из магазинов города. Это был самый сложный этап потому, что нужно было выполнить сразу 4 наисложнейшие и большие задачи, это были последние 10 дней каникул, количество выходных резко было сокращено голосованием команды, мы работали около 10-12 часов в день. Но эти шаги были вынуждены для того, чтобы выполнить следующие задачи:

1. Написать документацию
2. Снять видео презентацию
3. Написать программу распознавания руки человека
4. Создать браслет
5. Монтаж составляющих на модель
6. Ориентация на карте
7. Написать код для помощи выбора товаров
8. Собрать части программ работы отдельных узлов в один код

В результате использования системы Scrum мы смогли быстрее и эффективнее выполнить задачи и завершить проект.

# Разработка первого прототипа

## Причина разработки

Чтобы проверить, возможна ли ориентация в пространстве, используя вместо зрения датчики и вибромоторы, было решено разработать первый прототип с минимальным функционалом. Оповещение пользователя, если впереди препятствие не менее чем в 80 см с помощью вибромоторчиков, оповещение пользователя, если впереди препятствие не менее чем в 40 см с помощью аудио помощника. На прототипе должны были расположиться; 3 сонара, подключенные к 3 разным Arduino Pro Mini, и 1 Raspberry Pi Zero. Мини компьютер должен был получить и обработать данные с Arduino и проигрывать аудиозаписи.

## Получение и передача данных с Arduino Pro Mini

Чтобы понимать, как далеко находится препятствие, мы решили использовать ультразвуковой дальномер HY-SRF05. Максимальная дальность работы датчика равняется 2м, он потребляет 20мА в рабочем режим, компактен, дешев и распространен. Эти факторы повлияли на наш выбор датчика. Но одна из основных проблем – это угол измерения, 15 градусов. Чтобы решить проблему мы установили 3 датчика: слева, справа и спереди.

Следующая дилемма, перед которой мы встали – на каком устройстве обрабатывать данные с датчиков. Мы собираемся использовать в качестве основного железа Raspberry Pi Zero, к ней можно подключить датчик и не один, она может обрабатывать данные и передавать дальше их следующей программе. Но есть несколько причин, почему мы отказались от этого. Raspberry в дальнейшем должна обрабатывать видеопоток, работать с базой данной и включать аудиозаписи. Путем проб и ошибок было установлено, что если добавить нагрузки к плате линейки Zero, то проигрывает обработка видеопотока, что очень важно. Кроме того, время которое Raspberry тратит на опрос ультразвуковых датчиков больше, чем время на опрос SCL, SDA шины. Поэтому опрос датчиков происходит с помощью платы Arduino Pro Mini, код программы можно посмотреть в приложение.

Но если мы используем Arduino, то должны как-то передавать данные Zero, да еще и не с одной платы, а сразу с трех. Для этого мы использовали интерфейс I2C. Последовательный протокол обмена данными IIC (также называемый I2C – Inter-Integrated Circuits, межмикросхемное соединение) использует для передачи данных две двунаправленные линии связи, которые называются шина последовательных данных SDA(Serial Data) и шина тактирования SCL (Serial Clock**)**. Также имеются две линии для питания. Шины SDA и SCL подтягиваются к шине питания через резисторы.

В сети есть хотя бы одно ведущее устройство (Master), которое инициализирует передачу данных и генерирует сигналы синхронизации. В сети также есть ведомые устройства (Slave), которые передают данные по запросу ведущего. У каждого ведомого устройства есть уникальный адрес, по которому ведущий и обращается к нему. Адрес устройства указывается в паспорте (datasheet). К одной шине I2C может быть подключено до 127 устройств, в том числе несколько ведущих. К шине можно подключать устройства в процессе работы, т.е. она поддерживает «горячее подключение».

Raspberry Pi Zero мы программировали, как master, Arduino – slaver. Данные передаются пакетами по 8 бит, поэтому мы могли отправлять на raspberry массив типа byte, где хранится расстояние до препятствия, собирая на компьютере снова в число, а могли сравнивать расстояние с определенным значением, 80 и 40 сантиметров, и передавать только метки (0 и 1). Мы выбрали второй способ, из-за скорости, переложив на Arduino задачу по обработки данных и включения вибромоторов. Код работы master можно посмотреть в приложение.

## Передача данных между процессами

Так как прием данных с датчиков является важной частью работы устройства, то от скорости приема и обработки данных зависит, когда мы сообщим пользователю о препятствии. Именно поэтому мы писали эту часть кода на языке С++, но проблема заключается в том, что все остальные части программы написаны на языке Python3 (большинство используемых библиотек есть в этом языке и мы имеем больше опыта работы именно на Python). Программы работают параллельно и в разных процессах, но должны обрабатывать одни и те же данные. Есть несколько способов передавать данные из программы в программу. Первый записывать и считывать из файла. Такой способ подразумевает существования какого-либо файла, это может быть база данных, таблица excel, текстовой документ или что-либо другое, в который происходит запись данных из одной программы и считывание из другой. Но проблема заключается в том, что файл нельзя открывать одновременно двумя программами, вылезает ошибка чтения файла. Чтобы это исправить, нужно организовать очередь и синхронизировать программы. А это сложно и долго. Именно поэтому мы воспользовались вторым способом, передача данных через именованный канал (IPC).

IPC также использует способ записи данных в файл. Разница заключается в том, что этот FIFO файл создается с параметром, позволяющий работать с ним в разных процессах и открывать одновременно. Скорость передачи данных таким методом равна 755Мб/сек. Посмотреть код работы можно в приложение.

## Воспроизведение аудио файлов

Кроме установки вибромоторов для оповещения пользователя о препятствиях мы решили добавить аудио ассистента для экстренных случаев, расстояние до препятствия меньше 40 см, также ассистента мы планировали использовать в конечном устройстве. Воспроизводить аудиосообщения должна была Raspberry Pi Zero, но у линейки Zero отсутствует 3.5мм разъем Jack, что препятствует подключению наушников к плате. У нас был выбор подключить внешнюю звуковую карту через USB интерфейс или же подключать динамики напрямую к пинам GPIO. Так как звуковая карта занимает много места, мы отказались от неё ради уменьшения веса и размера.

У мини-компьютера Raspberry Pi есть 3 пина GPIO, которые подключены к звуковой карте, расположенной на плате, а именно 13,18,19. И нет никакой сложности, подключить наши наушники к плате напрямую без усилителя звука, но качество звука сильно страдает, даже в состоянии ожидания, сильные шумы. Чтобы исправить это, нужно было подключать их через делитель напряжения, на рисунке 4 представлена схема подключения.

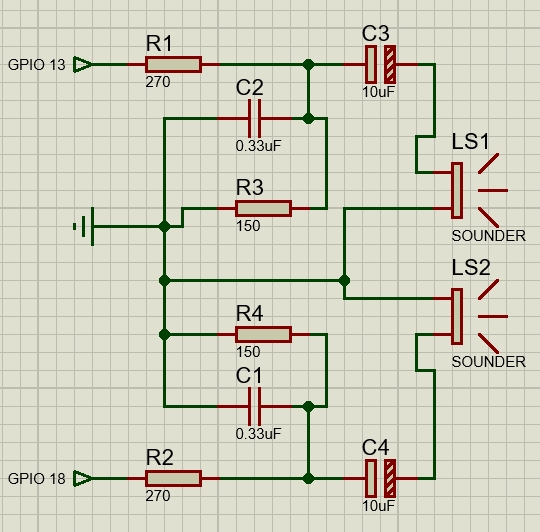


Рисунок 4. Схема подключения динамиков

После подключения осталось только написать SOFT для работы. Мы использовали библиотеку pygame для вывода звука, так как имели опыт работы с ней. Эта библиотека обладает обширной базой методов, что позволяет писать большие и сложные программы, но не все эти методы работают. Так как данные с датчиком передаются постоянно, то если препятствие будет сразу спереди и слева, следует произнести их последовательно. Это требует создания очереди, что очень плохо реализовано в библиотеке. Код реализации очереди можно посмотреть в приложение.

## Результат

Первый прототип собирался с целью тестирования концепции ориентации в пространстве с помощью датчиков, видео с тестированием можно посмотреть по ссылке (<https://yadi.sk/i/6gkuIrFq3aUjvH>) или в прикрепленном файле. В результате тестирования были обнаружены проблемы.

Переизбыток аудио команд. Слух – главное средство получения информации у слепых и слабовидящих людей. Именно он помогает им ориентироваться в пространстве, аудио оповещения отнимают такую способность у людей. Поэтому было решено снизить частоту воспроизведения аудио файлов и добавить кнопку отключения это функции.

Расположение комплектующих. На рисунке 5 изображен первый прототип, и по ним можно понять, что весь вес расположен на затылке и лбе. В ходе испытаний стало ясно,

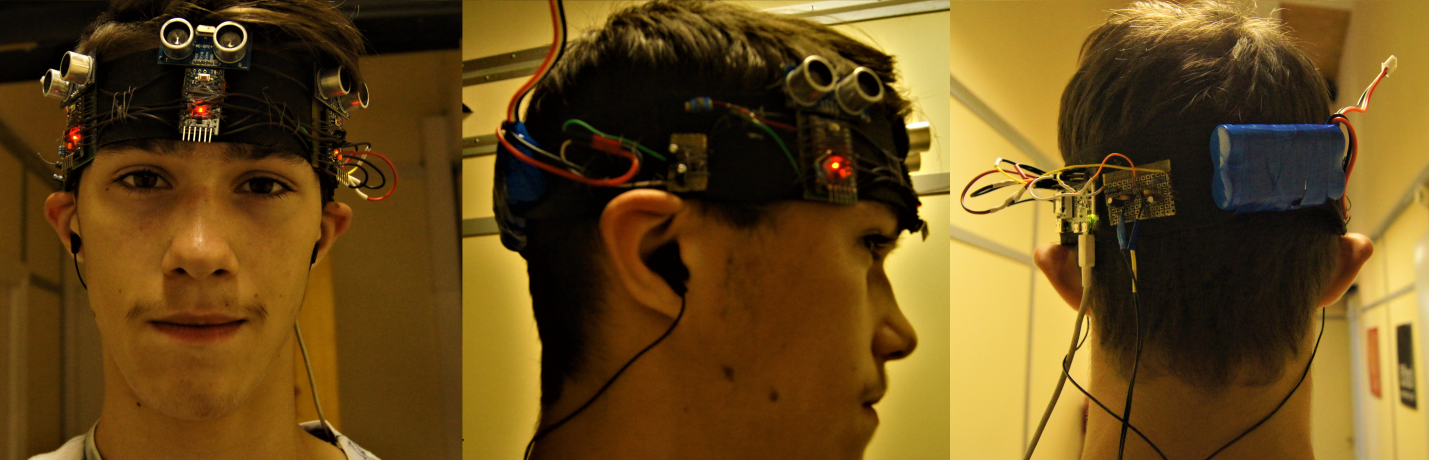


Рисунок 5. Первый прототип

что длительное использование устройства не представляется возможным при таком расположение компонентов. Было решено перенести вес на шею и плечи, оставив на голове только датчики.

Но, несмотря на это, очки справляются с задачей, из-за чего было сделано решение, продолжить разработку устройство, которое будет соответствовать техническому заданию и содержать функционал, описанный в концепции идеи.

# Создание устройства

## Моделирование «умных очков»

Разработку устройства мы начали с моделирования. Была поставлены требования для модели:

1. Возможность поместить электронику: 2 камеры, 2 сонара, 1 аккумулятор, 2 Raspberry Pi Zero
2. Не больше, чем горнолыжные очки

Иначе устройство занимает половину лица и неудобно

1. Большая часть электроники должна располагаться на шее
2. Должны быть линзы

Иначе очки выглядят странно и отталкивают людей

На основе требований были разработаны две модели «умных очков» Егором Соловьевым. Оба проекта состоят из двух частей: очки и крепление для электроники. Созданные модели отличаются внешне только частью «очки». Первая модель, которая изображена на рисунке 6, имеет размеры 150мм длина , 75мм высота, 10мм ширина.

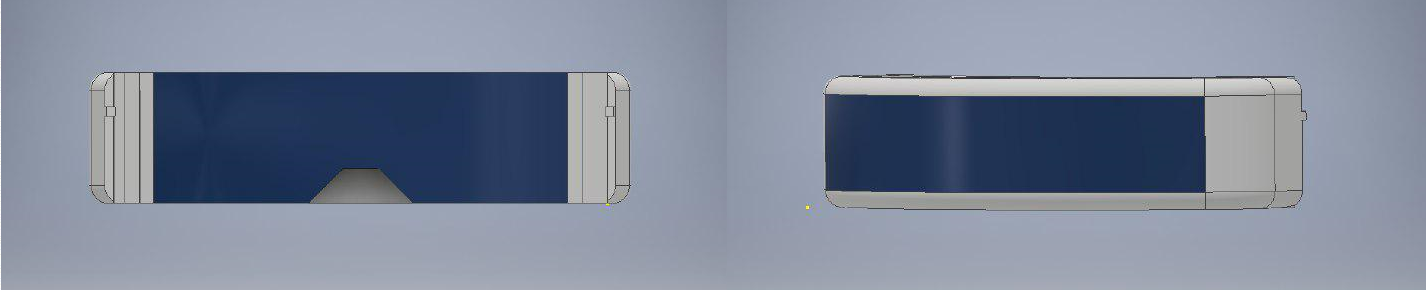


Рисунок 6. Модель «умных очков» первый вариант

Очки имеют округлую форму, вырез под нос и единое стекло, из-за чего должны удобно сидеть на голове. Но на очках не расположены отверстия по датчики, так как планировалось разместить их под стекло. Проблема заключается в том, что датчики, которые должны были там находится, мы не можем себе позволить, а именно лидар Mini size LiDAR. Поэтому нам пришлось отказаться от этой разработки и выбрать вторую, которая изображена на рисунке 7. Вторая модель имеет отверстия для двух сонаров и камеры. Но чтобы компоненты поместились, нам пришлось сделать выше и толще конструкцию. Очки имеют крепления для душек, которые позволят держать очки.

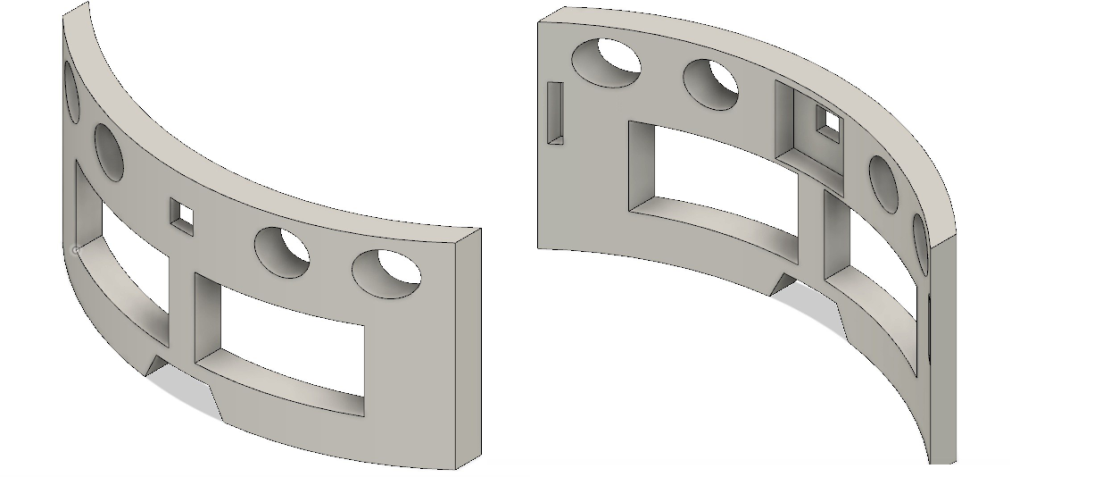


Рисунок 7. Модель «умных очков» конечный вариант

Все остальные компоненты мы разместились в отдельном модуле на шее. Вес конструкции с электроникой должен составить 273 грамма, что подходит под наше техническое задание. Очки и модуль с электроникой мы напечатали на 3D принтере из пластика FLEX, пластик гибкий из-за чего не разрушается и не деформируется при нагрузке и при сгибе.

## Распознавание объектов

Наше устройство должно работать в продуктовых магазинах самообслуживания. Для этого ему нужно распознавать товары и ценники. Так как в техническом задание сказано, что мы уже знаем, где находится нужный товар, то единственное, что нам нужно распознать – это его местоположение на полке. Считывать надписи на ценниках сложно и долго, поэтому мы пошли на небольшую хитрость. Посмотрев, как работают европейские магазины, решили добавить к ценнику QR код с информацией о товаре, который и будем считывать, впоследствии дополнив это более сложным и оптимизированным кодом для работы напрямую с ценниками без QR. Этой частью разработки занимался Важничин Никита.

Для распознавания QR мы использовали библиотеку OpenCV и Pyzbar. Кроме того, что код нужно найти, также его следует прочесть. Но обычно для этого следует писать большой и сложный алгоритм, который будет еще и не оптимизирован. Поэтому мы опустили этот этап, оставив библиотеке Pyzbar данную задачу. Так же за пределами нашей работы находится и нахождения QR, так как в библиотеке уже есть реализованный метод для поиска. Код нахождения и считывания QR кода можно посмотреть в приложение.

После того, как мы нашли QR код и получили подтверждение, что этот товар есть у нас в корзине, подробнее об этом в пункте «Создание базы данных», нам нужно найти сам товар на полке. Если принять, что ценник находится обычно под товаром и в районе вытянутой руки от него, то нам не нужно тратить мощности на обработку и распознавания объектов, а только искать их. Поэтому мы планируем искать товар на полке, как ближайшее облако точек на QR кодом. Код работы программы можно посмотреть в приложение, а алгоритм работы описан ниже.

Мы работаем с видеопотоком, поэтому, чтобы обрабатывать его, мы захватываем из него кадр. Полученное изображение мы изменяем алгоритмом Брэдли. Процесс бинаризации – это перевод цветного изображения в двухцветное черно-белое. Главным параметром такого преобразования является порог t – значение, с которым сравнивается яркость каждого пикселя. По результатам сравнения, пикселю присваивается значение 0 или 1. Существуют различные методы бинаризации, которые можно условно разделить на две группы – глобальные и локальные. В первом случае величина порога остается неизменной в течение всего процесса бинаризации. Во втором изображение разбивается на области, в каждой из которых вычисляется локальный порог, на рисунке 8 представлено изображение, обработанное нами для тестов, алгоритмом Брэдли.



Рисунок 8. Изображение после бинаризации

Главная цель бинаризации, это радикальное уменьшение количества информации, с которой приходится работать. После этого мы объединяем пиксели в вектора, соединяя близнаходящиеся вместе. Таким образом, после работы алгоритма визуально становится проще найти отдельные объекты. Программе же нужно знать координаты объектов относительно QR кода, размер, цвет, чтобы выбрать, какой из объектов является нашим. Проверка происходит последующим критерием:

1. Местоположение

Объект должен быть выше QR кода и не дальше чем в 0,5м от него

1. Размер

Объект не должен превышать средних по данному товары или же быть меньше

1. Цвет

Объект не должен отходить от нормы

Таким способом мы сокращаем количество возможных объектов до нескольких штук, выбирая затем самый близкий.

Но нам не достаточно знать, где находится нужный товар, нужно сообщить об этом пользователю. Использовать аудиосообщения не самый лучший вариант. Тесты показывают, что после таких указаний взять товар с первой попытки – мала. Тогда мы решили воспользоваться браслетом, который будет с помощью вибраций указывать, в какую сторону перемещать руку, подробнее о создание браслета в пункте «Создание браслета». Так как рука не имеет определенной формы и неравномерна, то прошлые методы сильно бы нагружали систему. Чтобы обойти это и упростить код мы решили сделать браслет определенного цвета, а именно красного. Программа так же, как и при распознавании объекта захватывает кадр из видеопотока для обработки, накладывая после на изображение маску красного цвета. Так как все изображение становится черным, кроме областей нужного цвета, то остается только следить за координатами центра области точек. Пытаясь затем помочь совместить руку с товаром.

## База данных

В документации не раз говорилось о том, что мы создаем аудио ассистента, также, так как мы работаем в магазине, то взаимодействуем с тысячами товаров. При этом многие части программы часто ссылаются на параметры товаров: цвет, размер, местоположение, название. Чтобы это все работало быстро и эффективно, мы решили хранить эти данные в базе данных.

База данных была создана в программе SQLite на языке SQL. База содержит две таблицы: basket и product. В таблице product содержатся все наименования продуктов, местоположение их на карте, цвет, размер и путь к аудиофайлу, в котором актриса произносит название продукта. В программе мы вызываем таблицу в начале работы кода, когда пользователь надиктовывает, какие товары он хочет купить, чтобы узнать, есть ли эти товары в магазине. Так же вызов таблицы происходит при каждом запуске аудио ассистента, через БД мы узнаем, где лежит аудиофайл с командой. Таблица basket содержит в себе только наименование товаров, которые были добавлены пользователем при входе в магазин. Нам нужна эта таблица для того, чтобы не создавать список в самой программе, позволяя работать со значениями разным потокам.

Запись и чтение значений из базы данных происходит программой, написанной на языке python3, с помощью библиотеки sqlite3, на рисунке 9 представлен код для работы с базой данных.

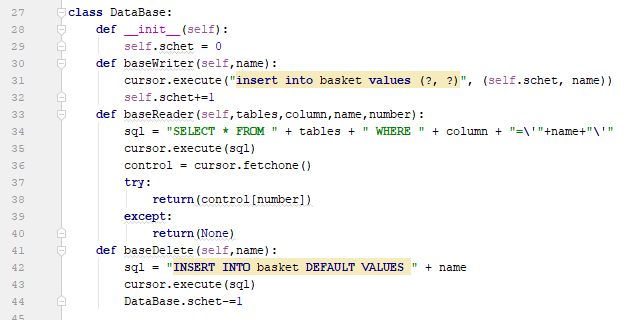


Рисунок 9. Класс DateBase

Реализованный класс позволяет записывать в базу данных значения в таблицу basket, с помощью метода baseWriter. Возможности метода ограничены только таблицей basket, так как только она имеет параметр на изменение значением. С этой же таблицей работает и метод baseDelete. Эту часть кода мы вызываем, если пользователь нашел нужный ему товар, удаляя тем самым строку из таблицы. Метод baseReader уже более универсальный. Он возвращает любое значение из таблицы, если такое существует, что позволяет узнать, существует ли товар или получить один из его параметров.

## Распознавание голоса

«Умные очки» должны не только помогать ориентироваться в пространстве, но и общаться с пользователем, чтобы точнее выполнять задачу. Задачи можно получать разными способами:

1. Использовать заранее записанный набор команд

Устройство израильской компании OrCam выполняет только одну задачу, чтение текста, который был указан пользователем. Мы могли тоже создать набор таких команд, когда пользователь указывает на предмет, и мы запускаем определенную часть кода. Но кол-во заданий ограничено, и пользователь должен выучить определенные движения, что не подходит для масштабируемой системы.

1. Записывание текстовых команд

Практически все компании, которые разрабатывают пользовательские ПО, давно перестали использовать вписанные пользователем команды для обработки. Перейдя на простой и интуитивно понятный пользовательский интерфейс. Конечно, нельзя использовать такой метод для слепых и слабовидящих людей, из-за невозможности ими увидеть наш интерфейс. Но также нельзя и использовать текстовые команды.

Во-первых, это заставляет людей держать постоянно в руках клавиатуру, или останавливаться

Во-вторых, команды могут содержать ошибки или быть некорректными потому, что люди редко могут ввести точную задачу.

Поэтому мы отказались от идеи использования текстовых задач

Мы решили сделать систему, с помощью которой, человек не отвлекаясь, может давать команды устройству, без использования дополнительных модулей, требующих контакта. Наш выбор пал на систему распознавания голоса.

Есть достаточное количество готовых библиотек, позволяющих реализовывать обработку голосовых команд. Одни из них работают только с подключением к интернету, библиотека SpeechRecognition, использующая API Google, или SpeechKit от компании Yandex Другие используют мощности локального компьютера для этого, pocketsphinx. У каждого из способов есть свои достоинства, из-за которых мы и начали тестировать сразу оба.

Обработка данных на сервере. Данный способ предполагает, что алгоритм для распознавания речи уже написан и находится в открытом доступе. Все что требуется сделать программе - это разрешить доступ к микрофону, записать аудио файл и отправить на сервер. Где уже алгоритм компании обработает его и вернет строку. Плюса способа: простота использования, точность обработки, возможность работы с русским языком. Минусы: время работы, требует подключения к интернету. Минусы есть, но они не особо сильно влияли на работоспособность устройство. Для работы мы использовали библиотеку SpeechRecognition.Он не требует ключа для работы и бесплатен, в отличие от библиотеки от компании Yandex.

Но есть еще и второй способ. Обработка данных на локальном компьютере. При таком способе обработке данных предполагается, что алгоритм распознавания уже готов, но нужно составить файл грамматики и словарь. Аудио файл будет обрабатываться на локальном компьютере, но по определенному плану, который будет прописан в файле грамматики. Грамматика – файл, в котором прописывается, какие слова будут использованы в запросе и в каком порядке, также там прописывается, слово – ключ для начала работы и окончания, чего нет в сервисе от Google. Словарь – это текстовый документ, в котором записаны все слова, которые может произнести пользователь, и их транскрипция. Плюсы способа: высокая скорость, возможность настройки. Минусы: низкая точность, отсутствие документации, возможность работы с русским языком. Для работы с этим способом мы использовали библиотеку pocketsphinx.

Мы пробовали писать код для работы, используя второй способ. Скорость обработки сообщений была выше, чем при работе с сервером, ответ приходил после 1 сек. В файле грамматике было описано задание, в котором программа ждала, пока не появится слово «добавить» или «купить». В словаре были записаны 100 разных наименований продуктов. Но из-за несовершенства встроенного алгоритма, результат в 30% был неверен. Было принято решение отказаться от этого способа.

Для распознавания голоса мы решили использовать готовое API, из-за точности распознования. Проблем с библиотекой SpeechRecognition у нас не было, код представлен на рисунке 10. Программа начинает слушать голос с момента запуска, поэтому возвращает весь текст, даже с лишним набором слов. Время обработки запросов равняется 5 секундам, что довольно-таки долго, чтобы его уменьшить нам следует использовать другой микрофон и увеличить скорость интернета.

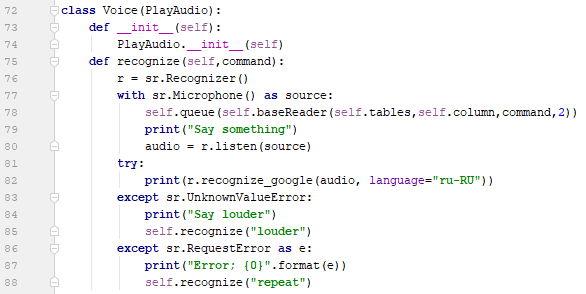
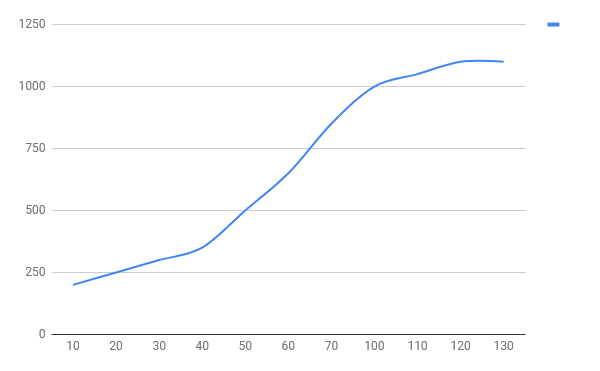


Рисунок 10. Распознавание голоса

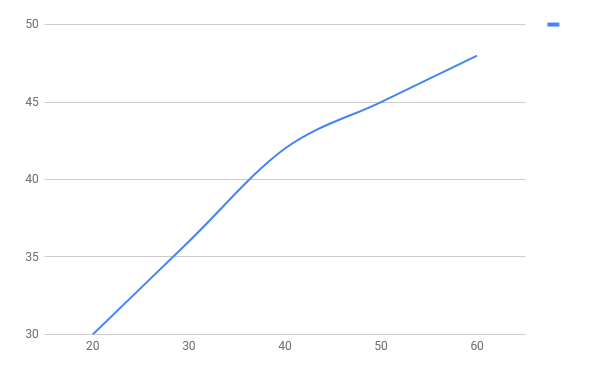
Также очки должны работать в тех помещениях, где нет wi-fi сети, для этого мы подключили к raspberry модуль GSM800L.

## Результаты исследования работы кода для распознавания объектов

В процессе работы над нашим проектом, мы провели исследование по определению качества распознавания QR кодов нашим устройством. Мы изменяли размеры QR кодов и проверяли, на каком максимальном расстоянии камера могла считать этот QR код. Результаты данного исследования вы можете увидеть на графике №1

График №1 (Вертикальная шкала обозначает расстояние (мм), горизонтальная шкала обозначает размер QR кода (мм).)

Этот график показывает, зависимость распознавания QR кода от определенной дистанции от размера QR кода. В результате исследования мы установили, что 85% QR кодов размером 110Х110 мм было распознано с расстояния 1050 мм.

Далее мы провели исследование по выявлению максимального градуса распознавания QR кода с разной дистанции удаления. Для этого был взят QR код размером 60Х60 мм. Результаты данного исследования вы можете увидеть на графике №2

График№2 (Вертикальная шкала обозначает градус поворота QR кода по отношению к камере, горизонтальная шкала обозначает расстояние удаления камеры (мм).)

## Создание браслета

В ходе испытаний стало ясно, что взять нужный продукт без посторонней помощи тяжело. Для этого был разработан браслет. Браслет имеет контрастный красный цвет, чтобы его было проще взять пользователю. В него вмонтированы 4 вибромотора, которые сообщают пользователю, где находится продукт, на рисунке 11 изображен браслет.

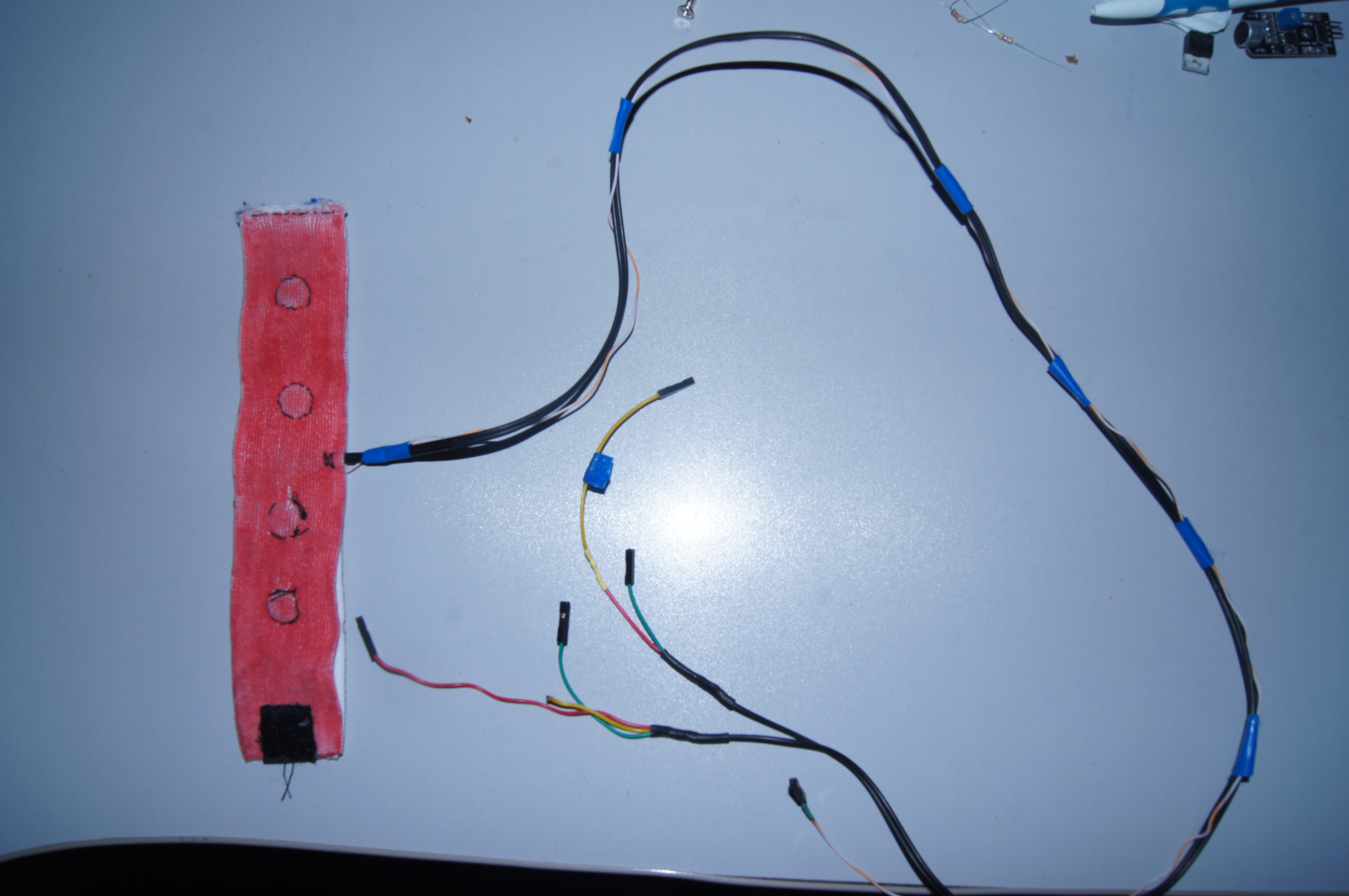


Рисунок 11. Браслет

Команды передаются по проводам, что будет изменено в дальнейшем. Браслет крепится к руке с помощью липучек, а также нанизан на резинку для эластичности.

# Тестирование устройства

## План тестирования

Для тестирования наших очков мы выбрали магазин «Самбери». В нем мы проверили и убедились, что наше устройство можно использовать в любых магазинах и даже в таких больших.

Перед тем, как идти за покупками, мы должны были сообщить нашим очкам: купить продукты. Выбранные нами продукты находятся на разных уровнях, а также абсолютно в разных местах. Наша задача была: найти, взять продукты и дойти до кассы.

Перед тем, как тестировать очки, мы хотели приготовить площадку для QR кода. На каждой полке, где находится товар, мы думали прикрепить QR код, но из-за строгих правил магазина и невозможности напечатать нужное качество кодов, нам пришлось держать их в руках.

Каждый QR код не только название продукта или направление выхода, но и отдела, стеллажа, для простоты ориентации и масштабируемости проекта.

## Результаты тестирования

В результате тестирования нам удалось найти продукты и дойти до кассы без столкновений, но появились некоторые проблемы, которые нам нужно было решить.

Мы предполагали разместить QR коды на каждом стеллаже, повороте и у товаров, но нам не удалось это сделать. Поэтому мы сами держали QR коды для испытуемого. По возможному пути мы располагали ложные QR, для того, чтобы человек с очками не поворачивал в какую-либо сторону только тогда, когда очки сообщают о нахождении QR ,чтобы избежать потасовки результатов. Кроме того мы держались вдали от испытуемого, чтобы он не мог использовать нас как ориентир, сонары находят препятствия, расположенных в 80 см от них . QR код был большего размера чем нужно потому, что не было возможности, распечатать меньшего размера.

Испытуемый с первого раза прошел весь маршрут, не зная его. Артём должен был, ориентируясь только по виброметрам, и голосовым сообщения, которые активировались в случае распознавания QR сам найти себе маршрут.

Так же мы столкнулись с проблемой передвижных препятствий: люди, тележки, грузы. Бывали случаи, что испытуемый был в полуметре от человека, который, видя его, не уходил, но очки помогали избежать столкновений.

Стало ясно, что использование дешевых наушников – плохой вариант. Они быстро вышли из строя и играли очень тихо, из-за чего приходилось придерживать их.

Камеру, которая должна была крепиться к очкам, мы перенесли на грудь из-за сломанного шлейфа, который порвался при перевозке.

Видео тестирование можно посмотреть по ссылке (<https://www.youtube.com/watch?v=UryrFpvN-0c&t=1s>) или в прикрепленном файле.

# Дальнейшее развитие проекта

## Изменение устройства

В будущем мы планируем внести модификации в конструкцию самих очков: сделать их меньшего веса и размера, для большего удобства их использования. Печатать мы будем очки типа «циклоп», которые описаны в пункте «Моделирование». Затем необходимо разместить вместо двух сонаров один лидар и две камеры для создания карты глубины. Помимо этого, планируется изменения типа работы браслета с проводного на дистанционный. Эту нужно для большего удобства ношения, уменьшения шанса поломки путем отрыва провода. Необходимо заменить наушники на более качественные и поставить усилитель звука, так как оказалось, что в людных местах очень шумно и той громкости, которую дает Raspberry не хватает (порой бывает сложно услышать команду). Так же мы планируем обрабатывать все данные, которые поступают от камеры и датчиков на специально созданном сервере. В дальнейшем производстве данных очков, мы хотим отказаться от производства всех деталей оправы на 3D принтере, так как это очень долго и не эффективно. Мы планируем штамповать все детали из пластика на специальном станке. Это существенно сократит время изготовки деталей оправы очков.

## Экономическая часть

Наш проект по составленному техническому заданию должен стоить не больше 400$. Поэтому мы использовали не оригинальные компоненты и пытались уместить их все в маленьком корпусе, в таблице 2 стоимость компонентов.

|  |  |  |
| --- | --- | --- |
| Наименование | Количество | Цена |
| Raspberry Pi Zero | 1 | 29.5$ |
| Виборомоторы | 5 | 6$ |
| Камера и шлейф | 1 | 32$ |
| Наушники и микрофон | 1 | 10$ |
| Сонары | 2 | 3$ |
| Arduino Pro mini | 2 | 5$ |
| Доп. компоненты | 1 | 21.3$ |

Таблица 2. Стоимость компонентов

В результате себестоимость нашего проекта 106.8$. Что ниже заявленного на 293.2$ и дешевле ближайших аналогов с таким же функционалом в 14 раз. В проект не закладывалась стоимость разработки из-за того, что разработка не закончена и при количестве устройств в 1000 штук будет в рамках 400$.

# Результат

В результате нашего проекта были созданы очки, помогающие людям с плохим или отсутствующим по разным причинам зрением ориентироваться на определенной местности.

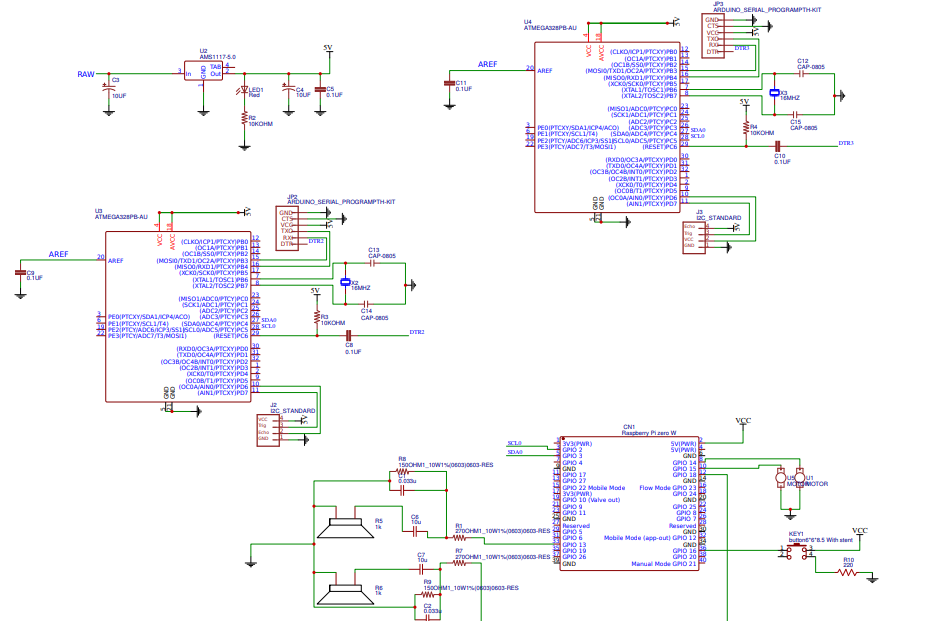
Было создано два прототипа: первый тестовый, на котором мы отработали некоторые функции, и узнали о возможных сложностях, которые нам удалось избежать. Второй прототип получился компактнее и удобнее в использовании. Данный прототип успешно прошел тестирование, как в условиях кабинета, площадь которого равна 20 квадратных метров, так и в условиях огромного супермаркета с посетителями. Для того чтобы провести испытания, нам необходимо было наладить контакты с представителями сети магазинов «Самбери», для получения разрешения проводить эксперимент на протяжении не определенного времени внутри магазина. В ходе этого тестирования, испытатель прошел из одного конца продуктового супермаркета сети «Самбери» в другой его конец, совершив по пути необходимые покупки. Во время этого прохождения испытатель ни разу не врезался в препятствие в виде прилавков, холодильников, а так же покупателей различного роста. В результате долгой и плодотворной работы у нас удалось создать готовое устройство, с работающим функционалом, описанным в техническом задание.

Освещение данного проекта в СМИ (<https://www.youtube.com/watch?v=TEyKxV3CQiI&feature=youtu.be>)

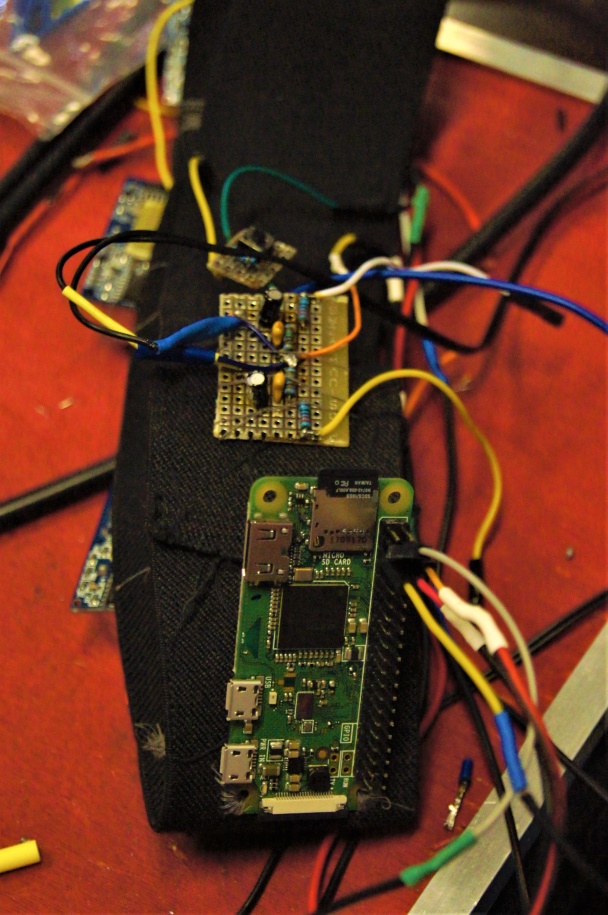
# Приложение

Код работы устройства прикреплен к проекту и находится в архиве “code”.

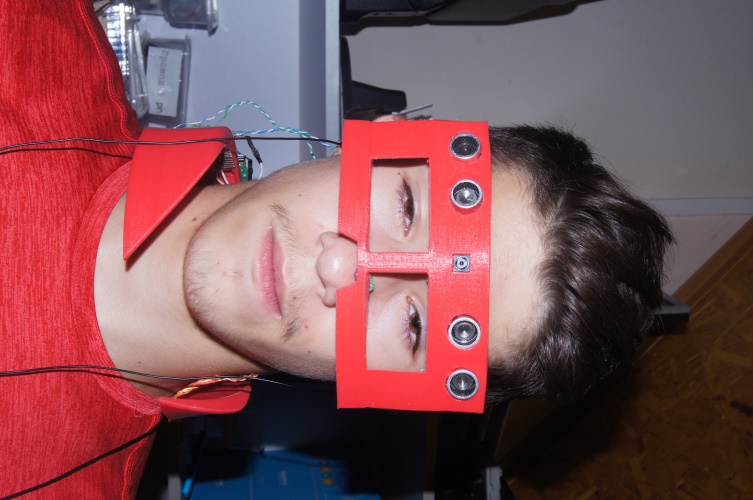
Фотографии работы команды и дополнительные фото устройств расположены в yandex disk (https://yadi.sk/d/6RnhsE5T3aDwQ7)



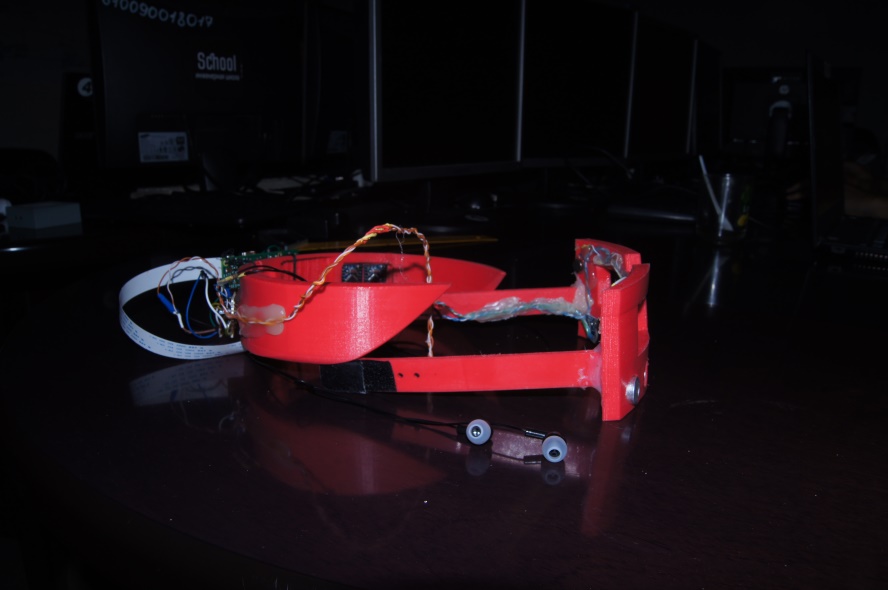
Электронная схема



Первый прототип



Устройство на испытателе

Устройство

# Ссылки на используемы материалы

<https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_tutorials.html>

<https://www.pygame.org/docs/>

<https://pypi.org/project/SpeechRecognition/>

SQLite Forensics [Paul Sanderson](https://www.amazon.com/Paul-Sanderson/e/B07D1QH7BP/ref=dp_byline_cont_book_1) – May 12, 2018

Полный справочник по C++ Герберт Шилдт -2002 г

Справочник программиста c++ Герберт Шилдт 2000г

<http://raspberrypi.ru/doc/>

<https://docs.python.org/3/library/multiprocessing.html>