Так же пользователь может добавлять новые маршруты. Для этого ему нужно перейти в пункт меню «Добавить маршрут». После открытия формы пользователь должен выбрать начало пути, ввести название кабинета и добавить все точки маршрута. Пример добавления показан на рисунке 3.

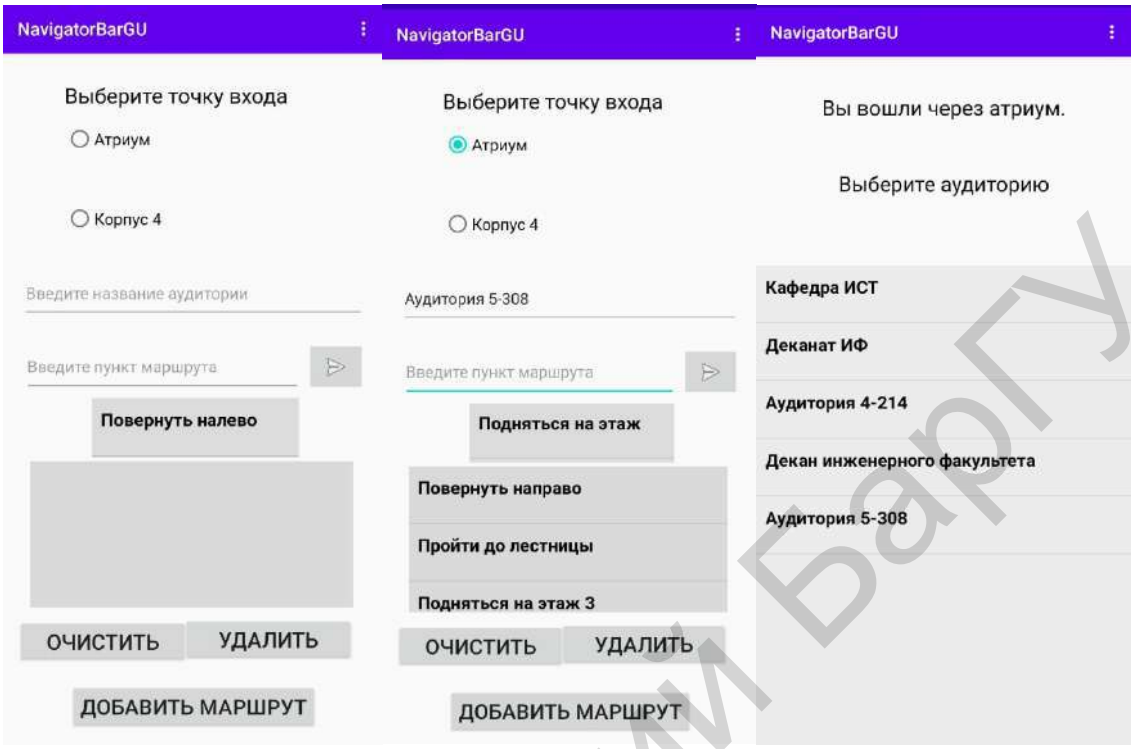

Рисунок 3 — Пример добавления нового маршрута

Заключение. В результате исследования было разработан мобильный путеводитель по специализированным кабинета Барановичского государственного университета. Оно поможет гостям, студентам и преподавателям университета находить нужные им кабинеты быстрее.

Список цитируемых источников

1. *Дарвин, Я. Ф.* Android. Сборник рецептов / Я. Ф. Дарвин — 2-е изд. — М. : Вильямс, 2017. — 768 с.
2. *Васильев, А. Н.* Программирование на Java для начинающих / А. Н. Васильев — М. : Эксмо,2017. —704 с.
3. Coderlessons [Электронный ресурс]. — 2021. — Режим доступа : https://coderlessons.com/tutorials/mobilnaia-razrabotka/uchitsia-android/android -baza-dannykh-sqlite . — Дата доступа : 07.05.2021.

УДК 621.311

А. В. Кульша
Учреждение образования «Барановичский государственный университет», Барановичи, Республика Беларусь

ПРОЕКТ «АВТОМАТИЧЕСКАЯ СВЕЧА» НА ARDUINO

Введение. Среда Arduino была разработана, чтобы быть простым в использовании для начинающих, которые не имеют опыта работы с программным обеспечением или электроникой. С помощью Arduino можно создавать объекты, которые может реагировать и/или управлять светом, звуком, касанием и движением. Arduino используется для создания удивительного разнообразия вещей, включая музыкальные инструменты, роботы, световые скульптуры, игры, интерактивная мебель и даже интерактивная одежда [1].

Основная часть. Целью данного исследования является создание физического устройства, с помощью которого можно дистанционно, по щелчку пальцев, зажечь свечу.

Инструменты и материалы, применимые в исследовании [2]:

– транзистор IRFZ44N;
– Arduino Nano V3;

– модуль звукового датчика;
– макетная плата;
– резисторы;
– перемычки;
– спички.

В процессе исследования были пройдены следующие пункты:

1. Сборка макета устройства. Сначала собирается устройство на базе макетной платы. В качестве «пламени» будет использоваться светодиод. Дальше устанавливается на макетную плату Ардуино, звуковой модуль, транзистор и светодиод.

2. Регулировка чувствительности звукового датчика. Теперь нужно отрегулировать чувствительность звукового датчика. Светодиод должен светиться, когда пользователь щелкает пальцами. На датчике есть регулировочный винт. Поворачивая винт нужно отрегулировать так, чтобы он реагировал на щелчок пальцами.

3. Зажигаем огонь. Теперь нужно заменить светодиод нитью накаливания и попытаться зажечь горючий материал. Мастер вытаскивает из провода одну медную жилу. Вместо светодиода устанавливает соединительные провода. На концы проводов наматывает медную жилу. Проверяет работу устройства.

Прототип устройства представлен на рисунке 1. Устройство в действии представлено на рисунке 2.

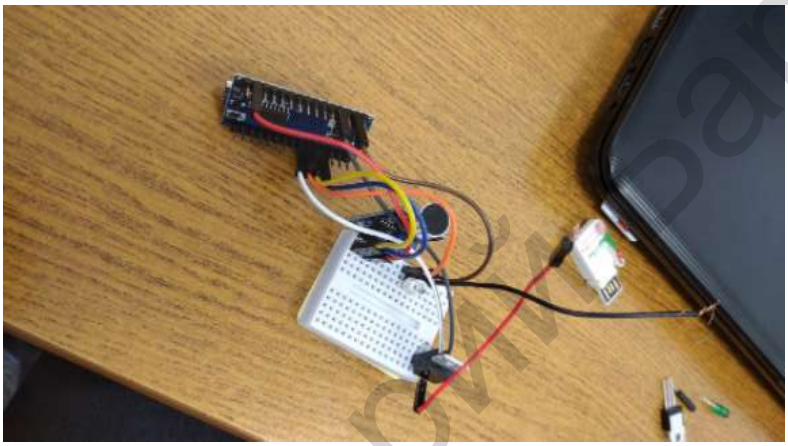

Рисунок 1 — Прототип устройства «Автоматическая свеча»

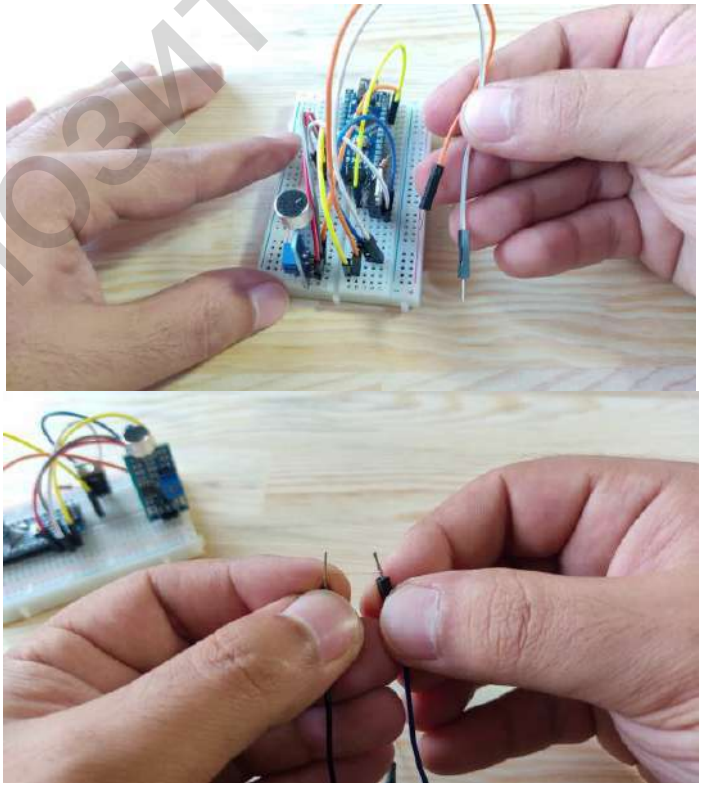

Рисунок 2 — Устройство в действии

Заключение. Плата спроектирована таким образом, чтобы на ней могла разместится толстая свеча. Через свечу нужно протянуть два провода и соединить их возле фитиля перемычкой. Рядом с перемычкой в свечу устанавливается спичечная головка (нужно, чтобы сера касалась перемычки).

В данном научном проекте предметной областью является макетный проект на базе Arduino UNO.

Список использованных источников

1. *Блум, Дж.* Изучаем Arduino : инструменты и методы технического волшебства / Дж. Блум. — СПб. : БВХ-Петербург, 2015. — 336 с.
2. *Монк, С.* Программируем Arduino : профессиональная работа со скетчами. — СПб. : Питер, 2017. — 272 с.

УДК 004.6

А. В. Кухта, А. И. Калько
Учреждение образования «Барановичский государственный университет», Барановичи, Республика Беларусь

ПРОЕКТИРОВАНИЕ ЗАШИФРОВАННОГО ОБЛАЧНОГО ФАЙЛОВОГО ХРАНИЛИЩА

Введение. Стремительное технологическое развитие и рост зависимости бизнеса от IT подталкивают компании к более внимательному и вдумчивому подходу к вопросам информационной безопасности, ведь чем сложнее информационная система, тем больше существует возможных рисков. Наиболее актуальным это является для компаний, ведущих какую-либо коммерческую деятельность онлайн или предоставляющих сервисы в Интернете.

В большинстве случаев данные, хранящиеся в кампании, составляют коммерческую тайну и подлежат защите в соответствие с законами РБ. Это является основной предпосылкой для заинтересованности предприятия в использовании защищенных хранилищ информации.

Основная часть. Предмет исследования является клиент-серверное программное приложение, имплементирующее функционал облачного хранилища.

Целью проекта является проектирование клиент-серверного приложения для хранения файлов на удаленном сервере.

Требуется спроектировать функционал клиент-серверного приложения для работы с удаленным файловым хранилищем.

Серверная часть программного продукта должно быть предназначено для операционной системы Linux в виде консольного приложение с логированием событий в логовый файл. Одним из главнейшых пунктов является отказоустойчивость сервера: в случае возникновения ошибок времени выполнения, приложение должно продолжать функционировать.

Клиентская часть программного продукта представляет собой кроссплатформенное приложение с интуитивно понятным интерфейсом.

Функциональные возможности программного продукта:

- регистрация пользователя в ОХ;
- аутентификация пользователя;
- создание файловой директории в файловой системе ОХ;
- выбор и загрузка файлов в ОХ;
- установка времени хранения файлов в ОХ по истечению которого файлы будет удалены;
- выбор и загрузка директории и файлов для загрузки из ОХ на ПЭВМ.

Механизм сетевых взаимодействий построен на основе протокола TCP/IP [1]. Приложения обмениваются информацией между собой путем отправки и получения блока данных, который содержит в себе идентификатор вызываемой функции класса RequestHandler и все необходимые аргументы, а также вызовом удаленных функций. Длина каждого блока не является фиксированной, поэтому перед каждым блоком всегда приходит блок фиксированного размера в 8 байт, который содержит в себе размер последующего блока данных динамического размера.

Пример блока данных, которыми обмениваются приложения, представлен на рисунке 1 (для удобства восприятия здесь и далее блок будет представлен в символьно-десятичном виде, и переменная, хранящая длину строковых переменных, хранится в 1 байте).

Рисунок 1 — Пример блока данных