Государственное бюджетное образовательное учреждение
высшего образования Московской области
Университет «Дубна»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

БАКАЛАВРСКАЯ РАБОТА

Тема: Разработка элементов системы «Умный дом»

Ф.И.О. студента Федоров Николай Андреевич

Группа 4251 **Направление подготовки** 09.03.04 Программная инженерия

Направленность (профиль) образовательной программы Разработка программно-информационных систем

Выпускающая кафедра распределенных информационных вычислительных систем

Руководитель работы _____ /ст. преп. Бархатова И.А. /

Консультант (ы) _____ /_____/

_____ /_____/

_____ /_____/

Рецензент _____ /доц. Сычев П.П. /

Выпускная квалификационная работа допущена к защите «____»_____**20**___**г.**
(дата)

Заведующий кафедрой _____ /_Кореньков В. В.____/

г. Дубна, 2021

Государственное бюджетное образовательное учреждение
высшего образования Московской области
Университет «Дубна»

УТВЕРЖДАЮ
Заведующий кафедрой

_____ /Кореньков В. В. /
(Подпись) (Ф И О
)

«_____» _____20___ г.

З а д а н и е

на выпускную квалификационную работу – бакалаврскую работу

Тема Разработка элементов системы «Умный дом»

Утверждена приказом № _____ от _____

ФИО студента Федоров Николай Андреевич

Группа _4251 **Направление подготовки** 09.03.04 Программная инженерия

Направленность (профиль) образовательной программы Разработка программно-информационных систем

Выпускающая кафедра распределенных информационных вычислительных систем

Дата выдачи задания _____

Дата завершения бакалаврской работы_____

Исходные данные к работе

Существующие решения похожей тематики, документации Android, Arduino, ESP32-Cam, Node.js

Результаты работы:

1. Содержание пояснительной записки (перечень рассматриваемых вопросов)
Обзор существующих решений, формирование требований, проектирование архитектуры, реализация

2. Перечень демонстрационных листов
Презентация Powerpoint

Консультант(ы) _____ /_____/
_____ /_____/
_____ /_____/

Руководитель работы _____ / ст. преп. Бархатова И.А./

Задание принял к исполнению _____

(дата)

_____

(подпись студента)

Аннотация

Концепция «Интернета вещей» и мобильная разработка активно развиваются в современной сфере информационных технологий, в частности мобильные устройства под операционной системой *Android* занимают лидирующее положение по популярности среди других операционных систем для мобильных устройств. Целью данной работы является объединение этих двух направлений разработки в одном проекте для создания информационной системы для обнаружения признаков движения в помещении и оповещения пользователя.

Для формирования требований были рассмотрены как частные разработки, так и готовые коммерческие решения, из которых были выявлены общие преимущества и недостатки. В результате проектирования были созданы общая архитектура системы и графический дизайн мобильного приложения.

Результатом работы является система из 3 элементов: устройства с датчиком движения, сервера и мобильного приложения. Данное решение представляет собой недорогой и простой способ отслеживания движений в помещении с получением изображений, и оповещением пользователя.

Abstract

The "Internet of things" and mobile app development are actively expanding in the current field of IT, in particular mobile devices powered by the Android OS are leading in popularity among other operational systems for smartphones. The goal of this work is to group up these fields under one project the result of which is an information system that detects movement in a room and notifies the user.

In order to form a list of requirements several projects were analyzed from which general advantages and disadvantages were identified. As the result of system design the general architecture of the system and mobile app's GUI were created.

The end product is a system which consists of 3 elements: a device with a motion sensor, a server and a mobile app. This solution offers cheap and easy-to-make way to detect movement in a room with notifications through smartphone app while also receiving pictures from the device.

Содержание

Введение 6

Обзор существующих решений 10

ESP8266 PIR Home Security & Notifier 10

IoT based Home Security System Using PIR Sensor, NodeMCU ESP8266, and Adafruit IO 11

Extaum WiFi Burglar Alarm 13

Smart Home & Security Internal PIR / Camera 14

Выводы 15

Формирование требований 17

Функциональные требования 17

Нефункциональные требования 17

Требования к эргономике и технической эстетике 17

Дополнительные требования 17

Выходные данные 17

Перечень и описание выходных сообщений 17

Перечень и описание структурных единиц информации выходных сообщений 18

Входные данные 18

Перечень и описание входных сообщений 18

Перечень и описание структурных единиц информации входных сообщений. 18

Системные требования 18

Требования к структуре и функционированию системы 18

Требования к способам и средствам связи для информационного обмена между компонентами системы 18

Проектирование архитектуры 20

Разработка 23

Устройство 23

ESP32-CAM 23
Камера *OV2640* 25
Датчик движения *HC-SR501 PIR* 26
Программирование 28
Arduino IDE 28
Программатор *FTDI FT232RL* 28
Установка драйверов 29
Подключение 30
Алгоритм работы 30
Серверная часть 31
База данных 31
Node.js, Express 32
Хостинг 32
Приложение 33
Проектирование интерфейса приложения 33
Разработка приложения 35
Рассмотрение результатов 38
Перспективы развития 38
Корпус устройства 39
Заключение 43
Список литературы 44
Приложения 47

Введение

«Интернет вещей» – концепция, возникшая в 1999 году, и активно развивающаяся по сей день, суть которой заключается во взаимодействии «вещей» (физических и виртуальных устройств) между собой без необходимости непосредственного участия человека в данном процессе [1][2].Рынок решений, разработанных по данной концепции, активно растет: прогнозируется, что размер рынка интернета вещей достигнет 1.5 трлн долларов США к 2025 году (Рис. 1) [3].

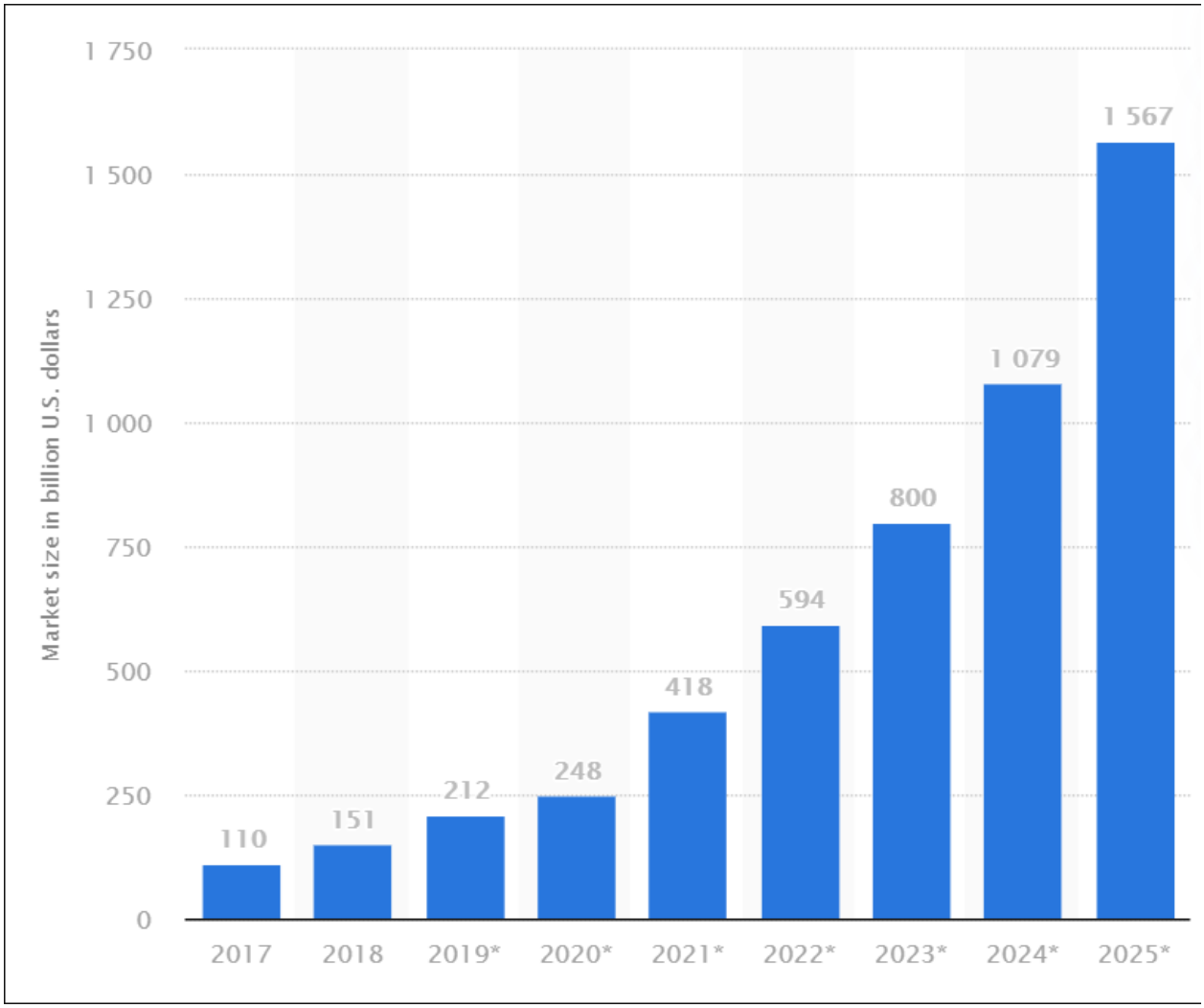


Рис. 1. Прогноз роста рынка интернета вещей

«Интернет вещей» является одним из приоритетных направлений разработки ПО в мире, которое предоставляет пользователям множество различных решений, в частности разработки т.н. технологии «Умный дом», общий смысл которой можно сформулировать как автоматизация повседневных дел человека. Примерами данной технологии служат системы автоматического включения/выключения света в помещении, зависящие от нахождения человека в комнате, системы автоматического полива и т.д.

Помимо «Интернета вещей» и «Умного дома» другим активно развивающимся направлением является мобильная разработка: в 2020 году число владельцев смартфонов

достигло отметки в 3.6 миллиарда человек (Рис. 2). Прогнозируется, что к 2023 году эта отметка достигнет 4.3 миллиардов [4].

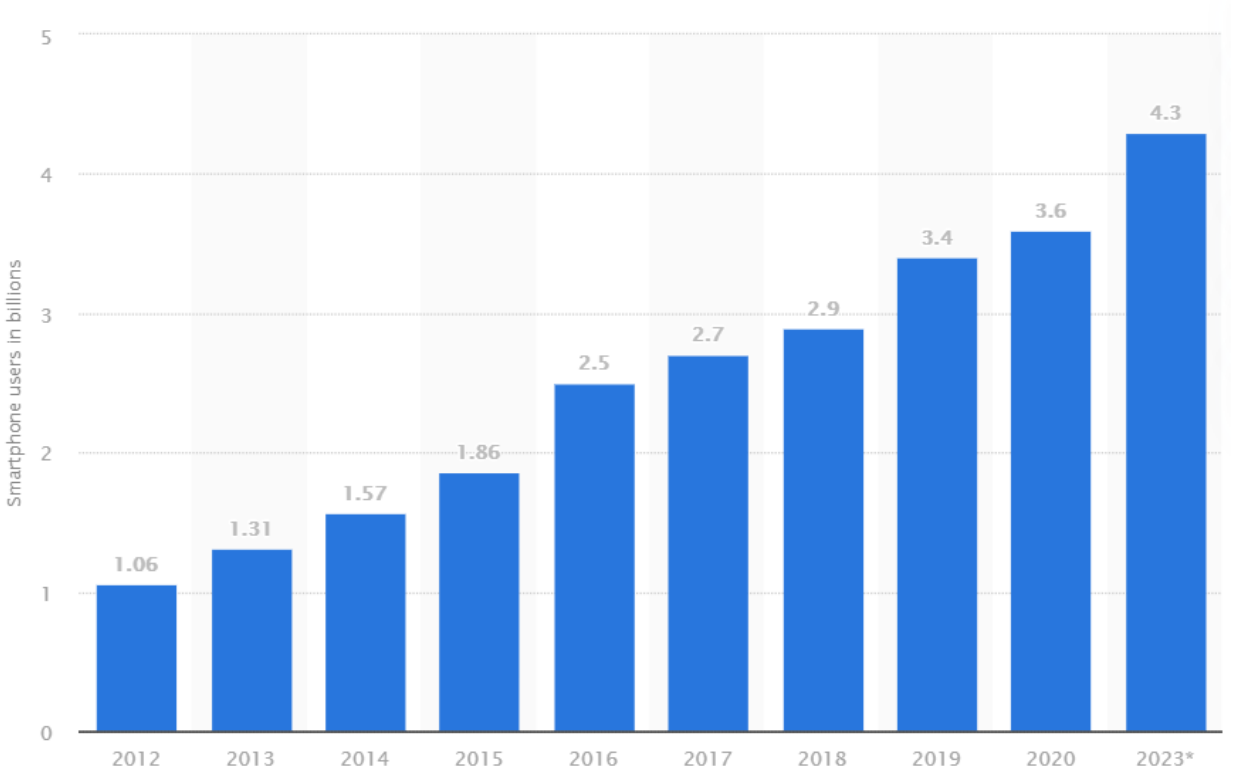


Рис. 2. Число владельцев смартфона за 2016-2021 года

Таким образом, разработка приложений для мобильных устройств охватывает огромное количество пользователей, делая ее приоритетным направлением разработки. Другим фактором, определяющим направление разработки, является операционная система, под которую будет выполняться разработка. В данном случае операционная система *Android* занимает большую часть рынка мобильных устройств – 71.9% всего рынка по статистике за февраль 2021 (Рис. 3, См. Приложение 1) [5]. Активная поддержка разработчиков под операционную систему *Android* со стороны компании *Google* делает данное направление наиболее привлекательным для разработчика, желающего начать написание приложений для мобильных устройств.

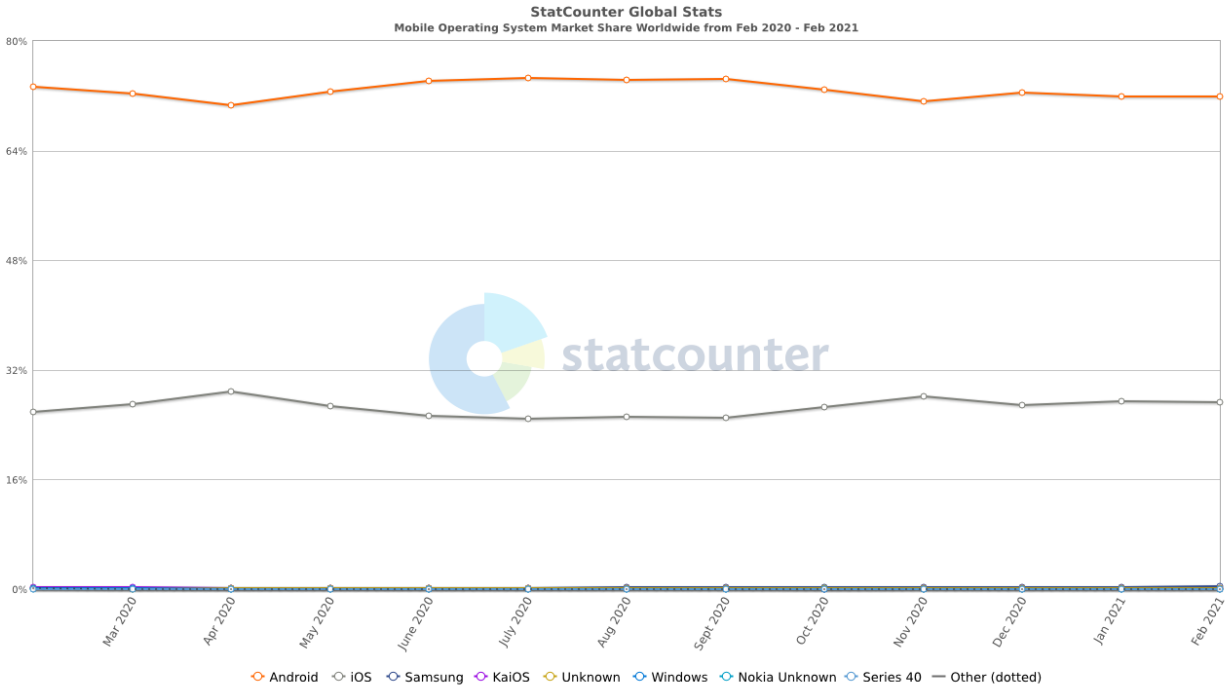


Рис. 3. Процентное соотношение операционных систем для смартфонов

В качестве языка разработки мобильных приложений для операционных систем *Android* активно набирает популярность язык *Kotlin*, который был выбран главным языком разработки для *Android* на конференции *Google* в 2019 году [6]. К списку преимуществ данного языка перед *Java* относят [6]:

- Возможность использования всех *Java* фреймворков и библиотек. Также возможна интеграция с *Maven, Gradle* и другими системами сборки;
- Открытый исходный код;
- *null*-безопасность;
- Значительная компактность кода по сравнению с *Java*;
- Простота в изучении;
- Обеспечивает взаимодействие между *Java* и *Kotlin*: позволяет вызывать написанный на *Java* код из *Kotlin* и наоборот;

Также, все примеры программ для *Android* предоставляют код на *Kotlin* в качестве основной опции в официальной документации [6].

Работа, которая будет рассматриваться в данном документе, объединяет вышеописанные направления: система «Умный дом» предназначена для постановки информации об обнаружении признаков движения в помещении. Пользователь будет информироваться с помощью мобильного приложения. Также идентификация «свой-чужой» каждого полученного сигнала будет также осуществляться пользователем в мобильном приложении. Система также является расширяемой в контексте получаемых пользователем данных с устройства посредством добавление новых датчиков.

Данная система состоит из 3 главных элементов: устройство, которое получает показания и отправляет их на сервер, непосредственно сам сервер, получающий данные с устройства и выполняющий их запись в базу данных, и мобильное устройство пользователя, отправляющее запрос на сервер на получение данных и подтверждение идентификации.

Обзор существующих решений

Из-за природы проекта существующие решения будут включать себя разработки, как крупных компаний, так и простых пользователей-любителей, из-за общей доступности и относительной дешевизны базовых комплектующих подобного рода проектов «Интернета вещей».

ESP8266 PIR Home Security & Notifier

Проект пользователя *rahuladitya303* [1] представляет собой систему оповещения пользователя об обнаружении признаков движения с помощью датчика движения *HC-SR501 PIR* и платы *NodeMCU* со встроенным модулем *Wi-Fi*. Для получения уведомлений на почту пользователю необходимо зарегистрироваться в стороннем приложении *Blynk App* (Рис. 4), после чего выдается уникальный токен, который указывается в коде для платы при отправке данных на сервер приложения.

Список компонентов проекта:

- Плата *NodeMCU* – средняя цена – от 155 рублей (на 05.05.2021)[2];
- Датчик движения *HC-SR501 PIR* – средняя цена – 75 рублей (на 05.05.2021)[3];
- Дополнительные кабели для подключения компонентов;

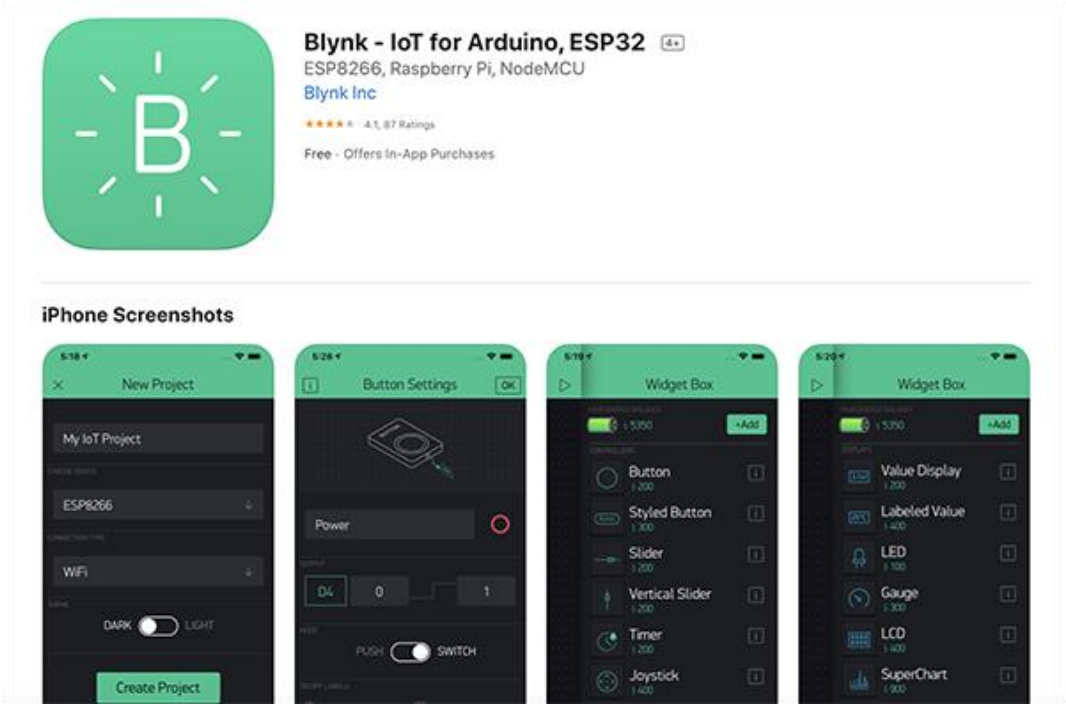


Рис. 4. Приложение *Blynk*

[1] https://www.hackster.io/rahuladitya303/esp8266-pir-home-security-notifier-b8245c
[2] https://aliexpress.ru/item/32665100123.html
[3] https://www.aliexpress.com/item/32699216549.html

Преимущества:

- Простота и низкая цена проекта (~230 рублей + затраты на кабели и/или плату разработки): все составляющие имеют низкую цену, код для платы находится в открытом доступе;
- Приложение работает с различными типами плат;
- Небольшие габариты устройства (49*24.5*13 мм);

Недостатки:

- Отсутствие камеры;
- Отсутствие истории;
- Требуется использование стороннего приложения;
- Оповещение происходит через электронную почту, а не через само приложение;
- Проект не предусматривает независимый источник питания (плата питается за счет подключенного к компьютеру кабеля micro *USB*);

IoT based Home Security System Using PIR Sensor, NodeMCU ESP8266, and Adafruit IO

Статья с сайта *Quartz Components*[4] описывает схожий с предыдущим проект по созданию охранной системы. Однако, в качестве обработчика полученных данных выступает сайт *Adafruit IO* (Рис. 5), созданный для различных проектов сферы «Интернета вещей».

Рис. 5. Главная страница сайта

[4] https://quartzcomponents.com/blogs/electronics-projects/iot-based-home-security-system-using-pir-sensor-nodemcu-esp8266-and-adafruit-io

После предварительной регистрации устройство будет отправлять показания на сервер, которые может просмотреть пользователь на сайте (Рис. 6).

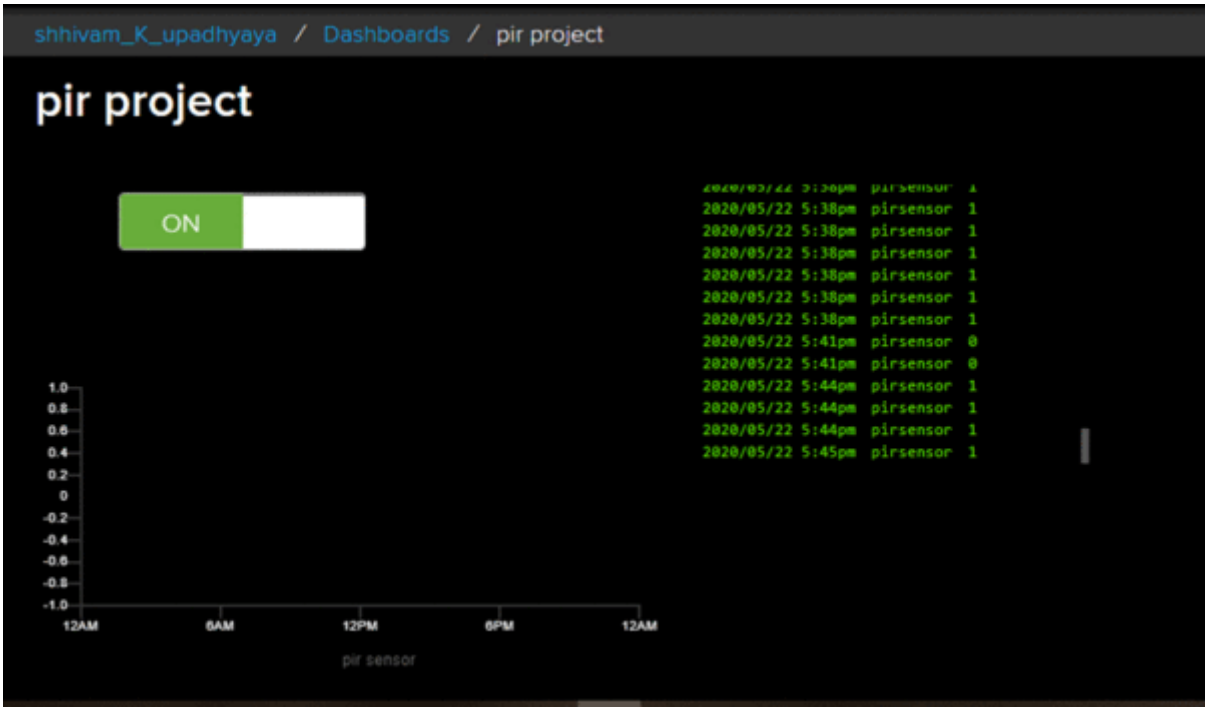


Рис. 6. Получение показаний

Преимущества:

- Сайт работает с различными типами плат;
- Гибкая настройка предоставляемой информации, возможность визуального представления показаний в виде графиков и т.д.;
- Простота и низкая стоимость проекта (одним из вариантов является использование составляющих предыдущего проекта, общая цена ~230 рублей + дополнительные затраты);
- Небольшие габариты устройства (49*24.5*13 мм);

Недостатки:

- Использование стороннего сервиса;
- Невозможность получения каких-либо уведомлений (получение данных только при переходе на сайт);
- Проект не предусматривает независимый источник питания (плата питается за счет подключенного к компьютеру кабеля *micro USB*);
- Отсутствие камеры;

Extaum WiFi Burglar Alarm

Данное решение компании *Extaum*[5] (Рис. 7) представляет собой охранное устройство, имеющее датчик движения *PIR* и использующее 3 *AAA* батарейки в качестве источника питания.

Рис. 7. Внешний вид устройства

Для оповещения пользователя об обнаружении сигнала используются стороннее приложение – *Smart Life/Tuya* (доступно для ОС *iOS* и *Android*) (Рис. 8).

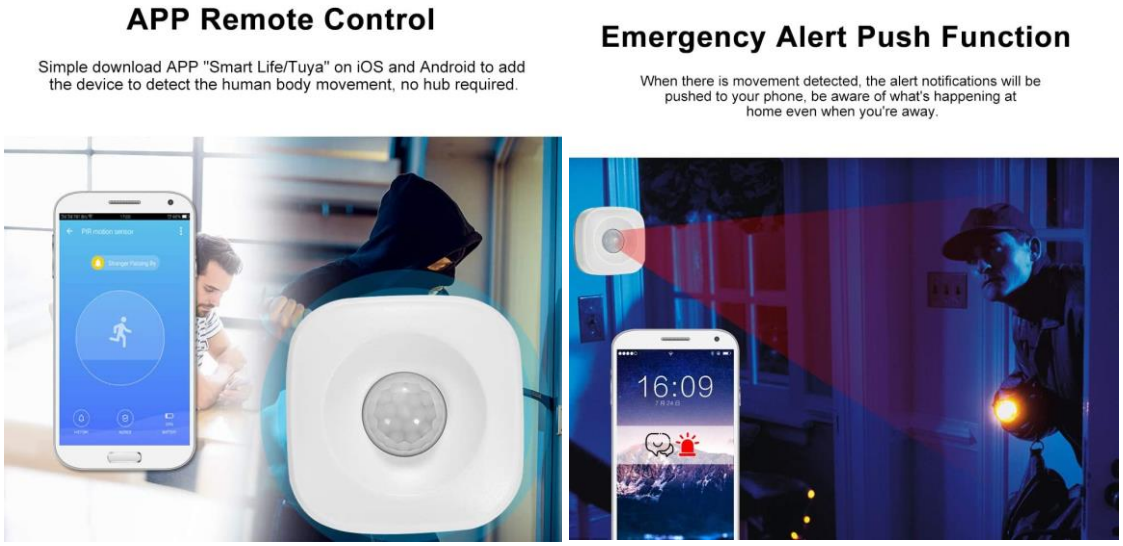


Рис. 8. Рекламные баннеры, демонстрирующие возможности устройства и приложения

Устройство имеет 2 режима работы: стандартный и экономный, с получением показаний каждые 2 минуты или каждые 4 минуты соответственно.

Стоит отметить следующие характеристики:

- Габариты: 6.5*6.5*2.9 см;
- Вес: 51 г;

[5] https://www.amazon.com/Extaum-Infrared-Detector-Compatible-Security/dp/B07TSGWGP7

- Цена: от 19.79 Долларов США (~1479.15 на 05.05.2021);

Преимущества:

- Готовый коммерческий продукт – от пользователя требуется только установка устройства и регистрация в приложении;
- Приложение поддерживает разные операционные системы;
- Получение уведомлений с приложения;
- Возможность просмотра истории полученных уведомлений;
- Разные настройки работы;
- Поддержка работы с *Alexa* и *Google Home;*

Недостатки:

- Относительно высокая стоимость по сравнению с рассмотренными выше проектами (~1479 рублей против ~225);
- Отсутствие камеры;
- Использование стороннего приложения;

Smart Home & Security Internal PIR / Camera

Продукт компании *SECURELY*[6] (Рис. 9) предоставляет возможность получать информацию об обнаружении признаков движения с помощью датчика движения, а также просматривать видеопоток с камеры на устройстве. Стоит отметить, что устройство идет в комплекте с фирменным программным обеспечением.

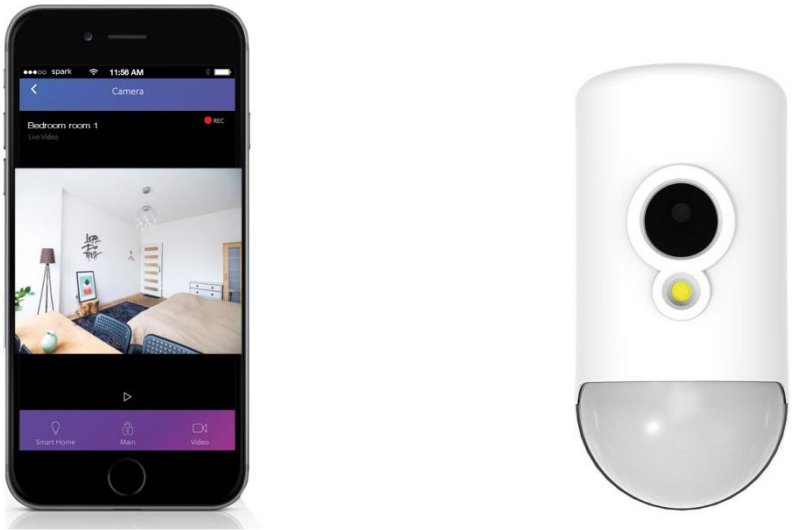


Рис. 9. Приложение и внешний вид устройства

[6] https://securely.nz/product/smart-home-security-internal-pir-camera/

Характеристики:

- Имеет свои приложение для смартфонов и веб-приложение для получения информации с устройства;
- Имеет светодиод, для освещения помещения;
- Вес: 640 г;
- Размеры: 180*128*77 мм;
- Цена: 280 Долларов США – ~ 20945.40 рублей (на 05.05.2021);
- Угол обзора камеры – 67 градусов, вместе с линзой – 90 градусов вертикально, 105 – горизонтально;

Преимущества:

- Наличие отдельных приложений для работы с устройством;
- Наличие камеры и возможность просмотра видеопотока;
- Наличие светодиода для освещения в случае, если в помещении уровень освещенности слишком низок для камеры;

Недостатки:

- Крупные габариты по сравнению с ранее рассмотренными вариантами;
- Цена превышает другие варианты больше, чем в 10 раз (в случае частных проектов – почти в 100 раз).

Выводы

Таким образом, после рассмотрения существующих решений были выделены следующие требования, которые должны будут учтены при разработке системы:

1. Наличие камеры – устройство должно иметь камеру для получения изображений в качестве дополнительной проверки от ложных сигналов;
2. Небольшие габариты – общие размеры устройства не должны превышать 70*70*30 мм;
3. Относительно низкая цена – цена всех компонентов не должна превышать 1000 рублей для обеспечения доступности данного решения;
4. Для получения данных и оповещения пользователя должно быть создано отдельное приложение, без использования сторонних сервисов;
5. Возможность просмотра пользователем всей истории полученных уведомлений.

Так как одним из поставленных требований является наличие камеры, в данной работе будет использоваться альтернатива платы *NodeMCU*, используемой в рассмотренных частных проектах, *ESP32-CAM*, благодаря наличию в ней разъема для подключения каме-

ры, из-за чего существенно облегчается работа с получением фотографий по сравнению с использованием отдельного камерного модуля с другими микроконтроллерами.

В качестве датчика движения в данном проекте будет использоваться пироэлектрический датчик *HC-SR501 PIR*, используемый в рассмотренных решениях.

Подробное описание каждого из компонентов будет дано в разделе «Устройство».

Формирование требований

Функциональные требования

Пользователями системы будут физические лица.

Возможности пользователя: просмотр последних сообщений с изображениями, полученных с камеры, просмотр истории всех полученных записей с сервера, очистка истории, включение/выключение уведомлений, включение/выключение получения сообщений с сервера.

Нефункциональные требования

Требования к эргономике и технической эстетике

Взаимодействие пользователей с мобильным приложением должно происходить с помощью графического интерфейса. Управление должно осуществляться посредством кнопок, наборов экранных меню и переключателей.

Все надписи экранных форм, а также сообщения, выдаваемые пользователю должны быть на русском языке.

Все экранные формы пользовательского интерфейса должны быть выполнены в едином графическом дизайне.

Интерфейс приложения должен поддерживать различные разрешения экрана мобильных устройств.

Дополнительные требования

Система должна эксплуатироваться на уже имеющемся у пользователя аппаратно-техническом комплексе, а именно на мобильном устройстве с операционной системой *Android* версии 10 и выше.

Также требуется наличием пользователем устройства – платы *ESP32-CAM* – оборудованным датчиком движения (*HC-SR501 PIR* или аналогом) и заранее зарегистрированным в системе.

Выходные данные

Перечень и описание выходных сообщений

Результатом успешной работы системы будут таблицы полученных сообщений с их соответствующим статусом подтверждения, который будет храниться на сервере, и с изображением, полученным с камеры в момент получения сигнала с датчика движения. Каждое сообщение также имеет свой идентификационный номер, который скрыт от пользователя и используется для последующего подтверждения сигнала пользователем.

Таблицы могут получаться пользователем в любое время по запросу.

Перечень и описание структурных единиц информации выходных сообщений

Формы таблиц представлены в Приложении 2 и в Приложении 3. Данные макеты интерфейса будут подробно рассмотрены в разделе «Проектирование интерфейса приложения».

Каждая запись имеет свои время и дату получения сигнала, изображение, и статус подтверждения. Дата и время имеют формат ДД.ММ.ГГГГ и ЧЧ:ММ соответственно, где ДД – номер дня месяца, ММ – номер месяца в году, ГГГГ – год, ЧЧ – час, ММ – минуты.

Изображение имеет формат *JPEG* и размеры 320*240 пикселей.

Входные данные

Перечень и описание входных сообщений

Идентификационный номер записи (скрыто от пользователя). Данный идентификационный номер присваивается сообщению по мере прибытия на сервер. Идентификационный номер используется для смены статуса записи при подтверждении ее пользователем.

Перечень и описание структурных единиц информации входных сообщений.

Каждая запись полученных показаний имеет свой идентификационный номер, который отправляется приложением при нажатии пользователем кнопки «Подтвердить», что принимается сервером как подтверждение сигнала.

Идентификационные номера задаются четырехзначными числами.

Системные требования

Требования к структуре и функционированию системы

Система должна быть реализована в виде комплекса программ, работающих на разных устройствах: устройство, которое представляет собой плату *ESP32-CAM,* для получения показаний с датчика движения *HC-SR501 PIR* и снимков с камеры, сервер, использующий *Express Node.js* для создания программного интерфейса, мобильное приложение для устройств с операционной системой *Android* версии 10 и выше. Система должна обрабатывать сигналы с устройства и предоставлять данные пользователю с их сохранением на сервере.

Система должна обновлять записи по мере их подтверждения пользователем и отображать их соответствующим образом.

Требования к способам и средствам связи для информационного обмена между компонентами системы

Система должна быть реализована на устройстве на базе платы *ESP32-CAM*, сервере и мобильном устройстве пользователя с операционной системой *Android* версии 10 и

выше. Передача данных между элементами системы должна осуществляться посредством сети Интернет.

Проектирование архитектуры

Для наглядного представления работы системы были построены диаграммы в нотации *UML*: диаграмма вариантов использования, диаграмма классов и диаграмма последовательности, определяющие главные элементы данной системы и связи между ними.

На диаграмме использования (Рис. 10) представлены все действующие лица системы (акторы) и выполняемые ими действия в контексте системы и связи между ними.

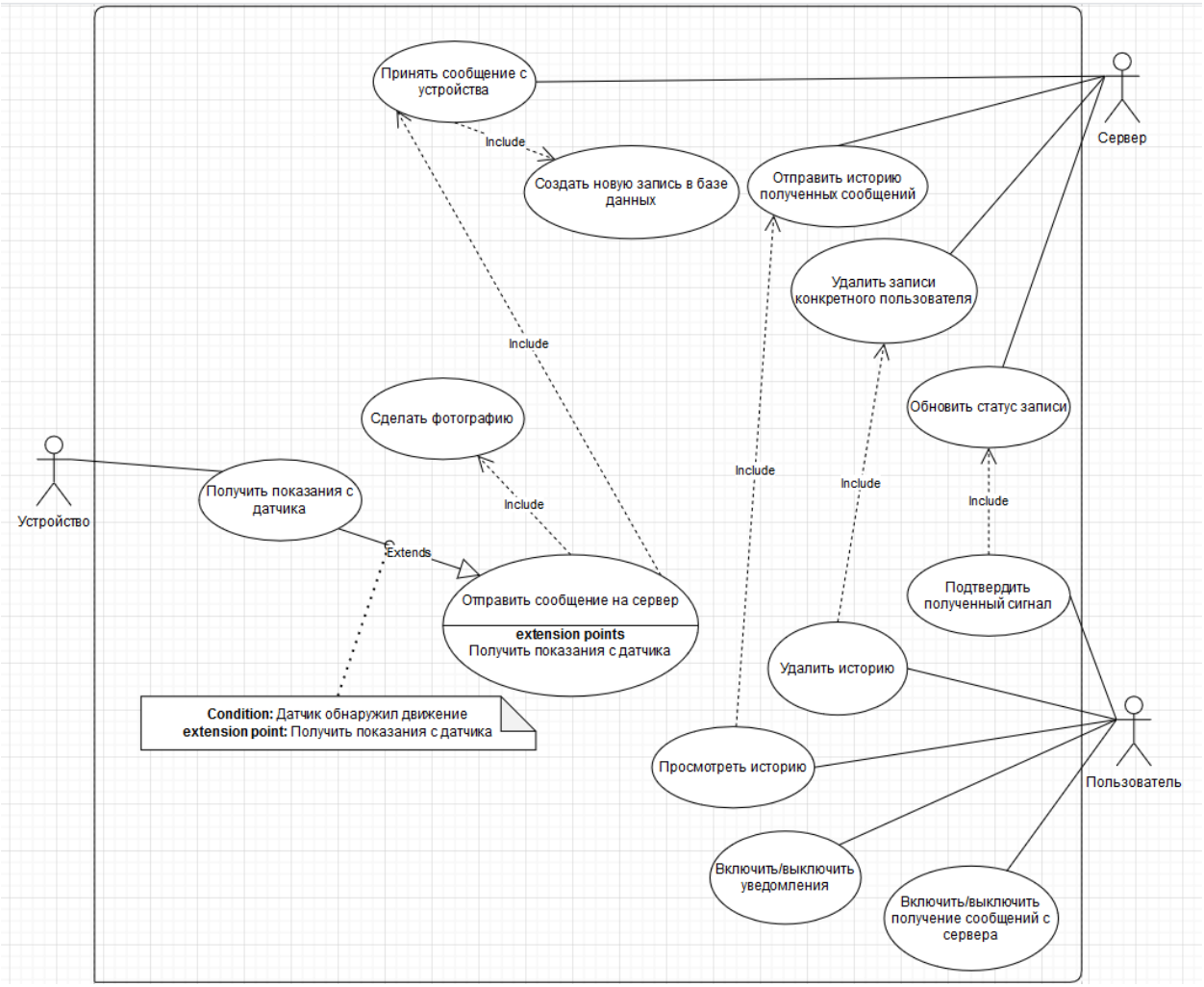


Рис. 10. Диаграмма вариантов использования

Диаграмма классов (Рис. 11) представляет статическую картину системы.

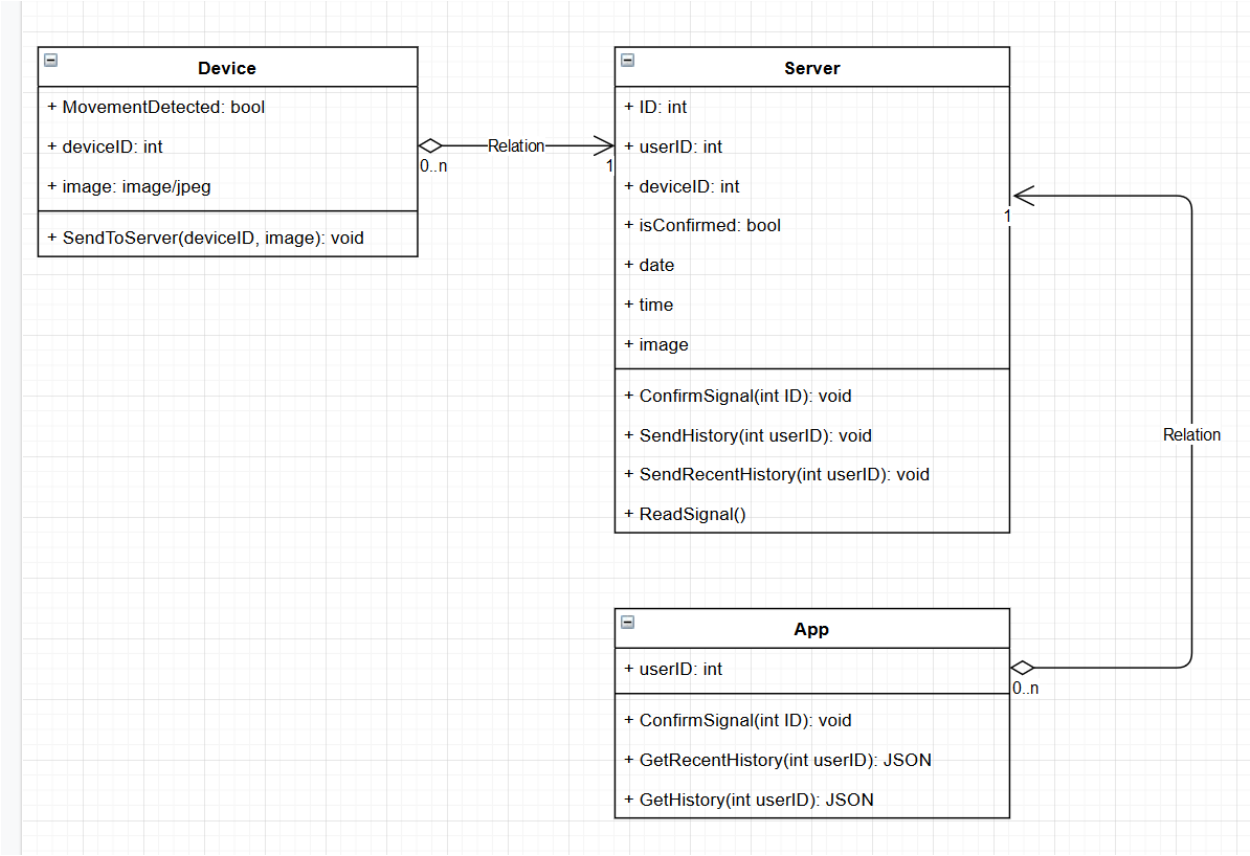


Рис. 11. Диаграмма классов

Диаграмма состоит из 3 классов:

- *Device* – класс устройства. Содержит в себе поля для идентификационного номера устройства, изображения и статуса обнаружения движения. Также имеет метод отправки сообщений на сервер.
- *Server* – класс сервера. Содержит в себе все полученные сообщения с устройств со статусом подтверждения их пользователем, а также, к какому пользователю они принадлежат. Содержит методы для получения сообщений, отправки истории пользователю и обновление статуса записи.
- *App* – класс приложения. Содержит идентификационный номер пользователя и методы для получения истории и подтверждения записи.

Диаграмма последовательности (Рис. 12) описывает процессы, происходящие в системе с момента получения показания с датчика движения устройством.

В бесконечном цикле устройство получает показания с датчика движения через некий промежуток времени. При получении положительного сигнала устройство делает снимок и отправляет сообщение на сервер, где создается новая запись в базе данных. Далее по запросу сервер отправляет в приложении историю полученных сообщений (за весь период активности или за последний день) или обновляет статус записи на «Подтверждено» после соответствующего действия пользователя.

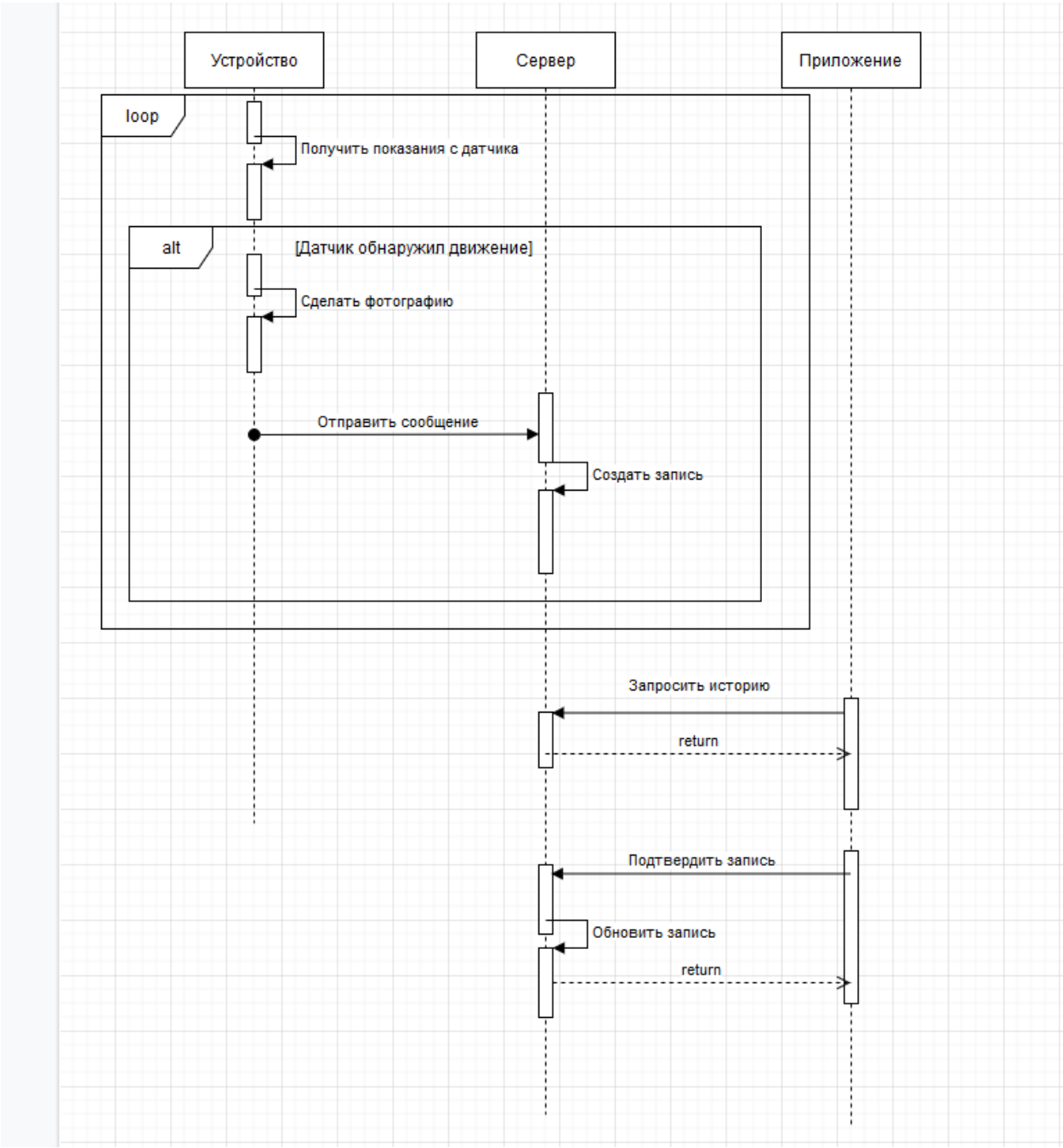


Рис. 12. Диаграмма последовательности

Разработка

Устройство

Концепция «интернета вещей» тесно связана с понятием «встраиваемая система» (*Embedded systems*) – специализированная микропроцессорная система, программный код которой разработан на выполнение конкретной функции как независимая система или как подсистема [7]. В данной работе встраиваемой системой будет являться устройство, получающее показания с датчика движения и передающее данные на сервер.

ESP32-CAM

В качестве устройства был выбран одноплатный модуль *ESP32-CAM*, специально разработанный для разработки устройств «интернета вещей» и работающий на базе *ESP8266* (Рис. 13) – микроконтроллера с интерфейсом *Wi-Fi*.

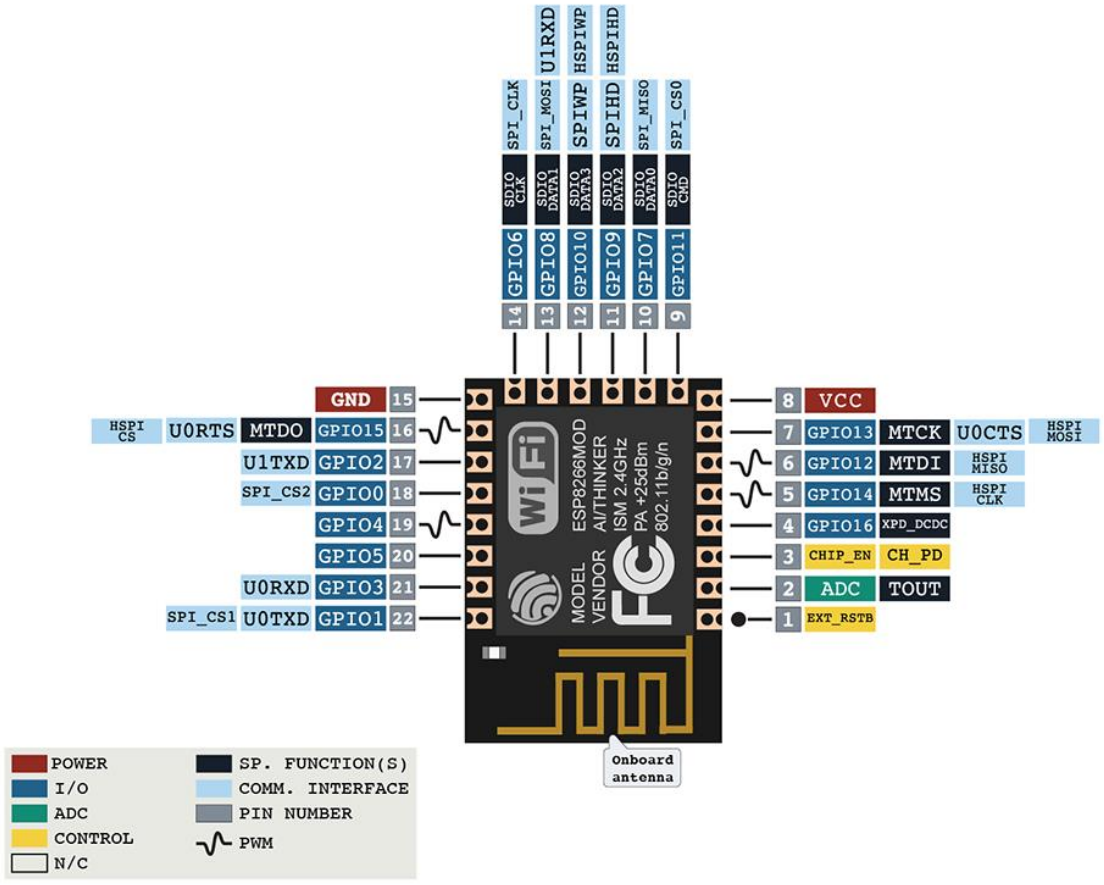


Рис. 13. *ESP8266*

Отличительной особенностью *ESP32-CAM* (Рис. 14, 15), из-за которой он был выбран для использования в данной работе, является специально выделенный разъем для камеры. Устройство также имеет разъем для карт типа *MicroSD*, однако, добавление карты памяти также блокирует доступ к цифровым выходам платы, так как они выделяются для работы с картой.

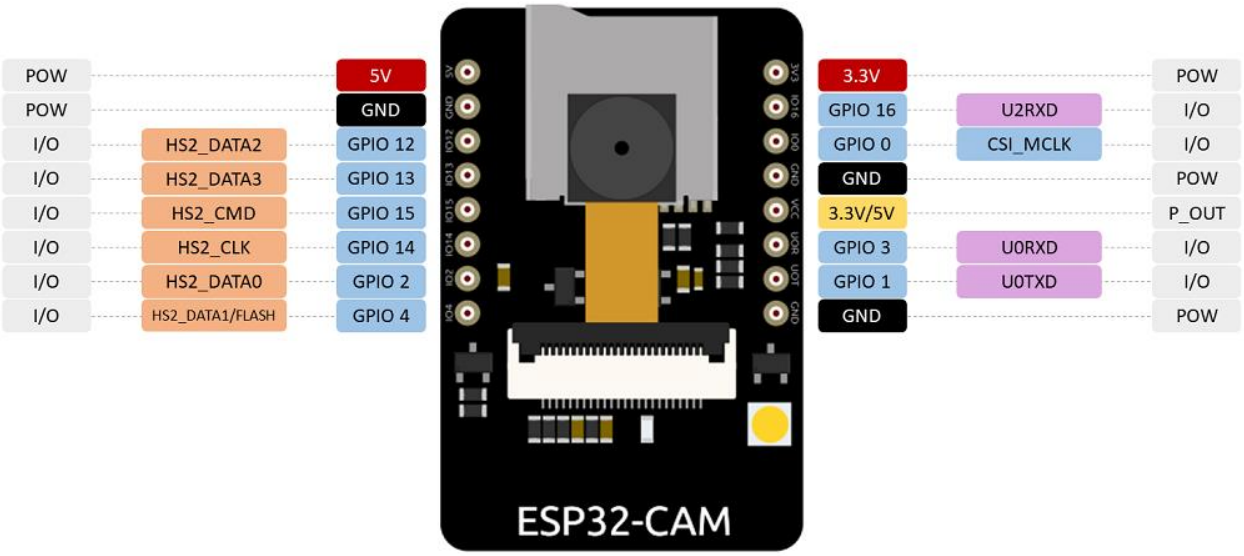


Рис. 14. *ESP32-CAM*

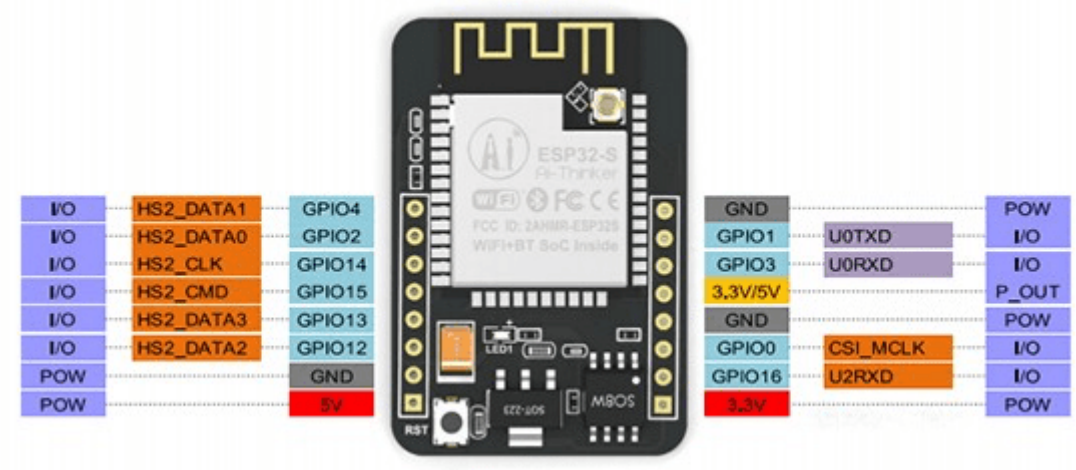


Рис. 15. *ESP32-CAM* – обратная сторона

Характеристики [8][9]:

- Вес: 5 г;
- Габариты (Рис. 16): 27*40.5*4.5 мм;
- Напряжение питания: 5 В;
- Ток потребления:
  - 180 мА;
  - при включенной вспышке – 310 мА;
  - в режиме глубокого сна – 6 мА;
- Процессор: 32-разрядный *Tensilica Xtensa LX6*;
- Тактовая частота процессора: 2 ядра*160 МГц;
- Память: 520 КБ *SRAM, 4МБ PSRAM*;
- Беспроводная связь: *Bluetooth 4.2 BR/EDR, Wi-Fi 802.11 b/g/n* 2.ГГц;
- Интерфейсы: *UART, I2C, SPI, PWM*;

- Микроконтроллер *ESP8266* имеет встроенный интерпретатор языка *LUA*;

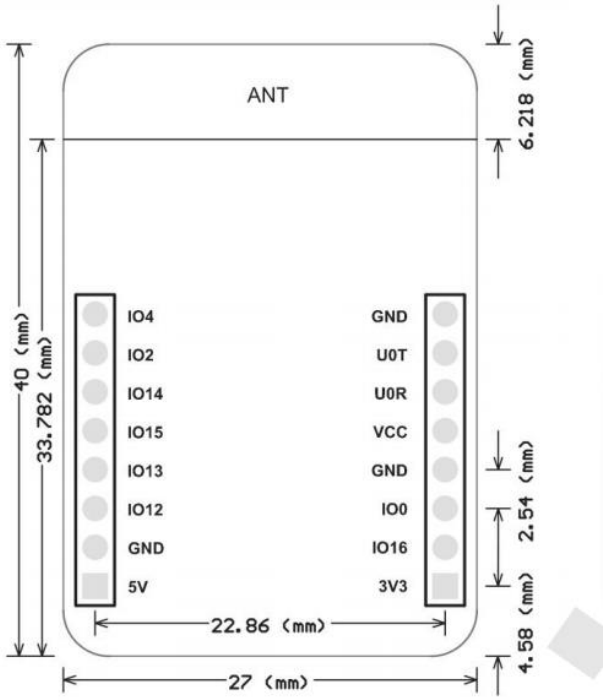


Рис. 16. Размеры платы

Камера *OV2640*

В комплекте с *ESP32-CAM* часто идет камера *OV2640* (Рис. 17), которая будет использоваться в данном проекте.

Рис. 17. Камера *OV2640*

Характеристики [10][11]:

- 2-мегапиксельная камера;
- автоматический контроль экспозиции, автоматический контроль усиления, автоматический баланс белого, автоматическое устранение световых полос,

автоматическая калибровка черного уровня. Контроль качества изображения, включая насыщенность цвета, оттенок, гамма, резкость;

- *ISP* с шумоподавлением и компенсация мертвых пикселей;
- Разрешение матрицы датчика изображения: 1600*1200 (*UXGA*);
- Питание: 3.3 В;
- Интерфейс *SCCB*, совместимый с интерфейсом *I2C*;
- Уровни напряжений *IO*: 1.7 В ~ 3.3 В (постоянный ток);
- Рабочая температура: от -30 до 70 ℃;
- Рекомендуемая температура: от 0 до 50 ℃;
- Угол обзора: 70 градусов;
- Выходные форматы:
  - *YUV*(422/420)/*YCnCr*422;
  - *RGB*565/555;
  - 8-*bit* сжатые данные;
- Максимальная скорость передачи изображения:
  - *UXGA/ SXGA* - 30 кадров в секунду;
  - *SVGA* - 30 кадров в секунду;
  - *CIF* (*Common Intermediate Format*) - 60 кадров в секунду;

Датчик движения *HC-SR501 PIR*

В качестве датчика движения в данном проекте будет использоваться пироэлектрический датчик *HC-SR501 PIR* (Рис. 18, 19) – малогабаритный, простой в использовании и недорогой вариант датчика движения, что является причиной его выбора.

Характеристики датчика [12][13][14]:

- Работает при разных напряжениях: от 4 до 12В (рекомендуемое напряжение – 5В и выше);
- Различает движения объектов и движения человека;
- Имеет два режима работы: *Repeatable(H)* – получение показаний через настраиваемый промежуток времени – и *Non-Repeatable(L)* – при обнаружении движений человека датчик будет подавать сигнал пока человек не выйдет из его поля зрения;
- Угол обзора – 120 градусов, максимальное расстояние – 7 метров;
- Низкое потребление тока – 65 мА;
- Работает при температурах от -20 до +80 градусах Цельсия;

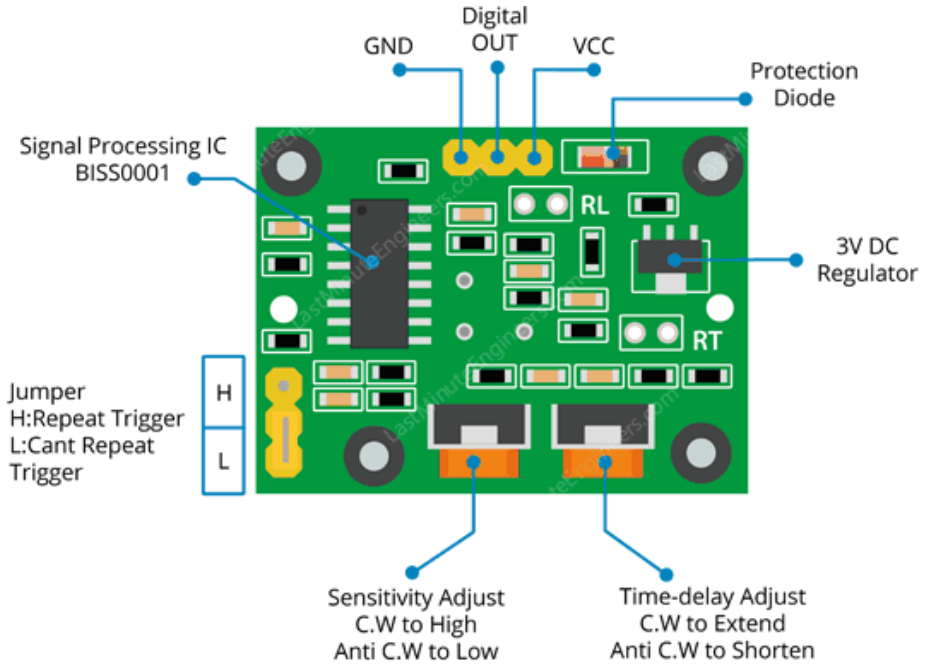


Рис. 18. Датчик движения *HC-SR501 PIR*

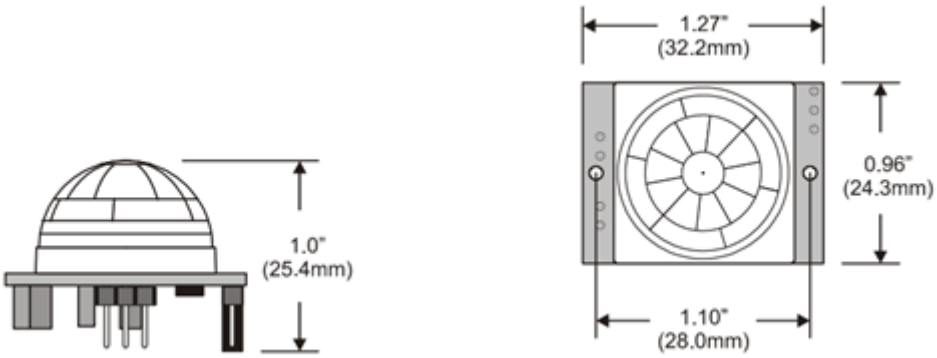


Рис. 19. Габариты датчика

Датчик имеет 3 пина: питание, земля и цифровой выход. Подключение к плате *ESP32-CAM* осуществляется следующим образом (Рис. 20).

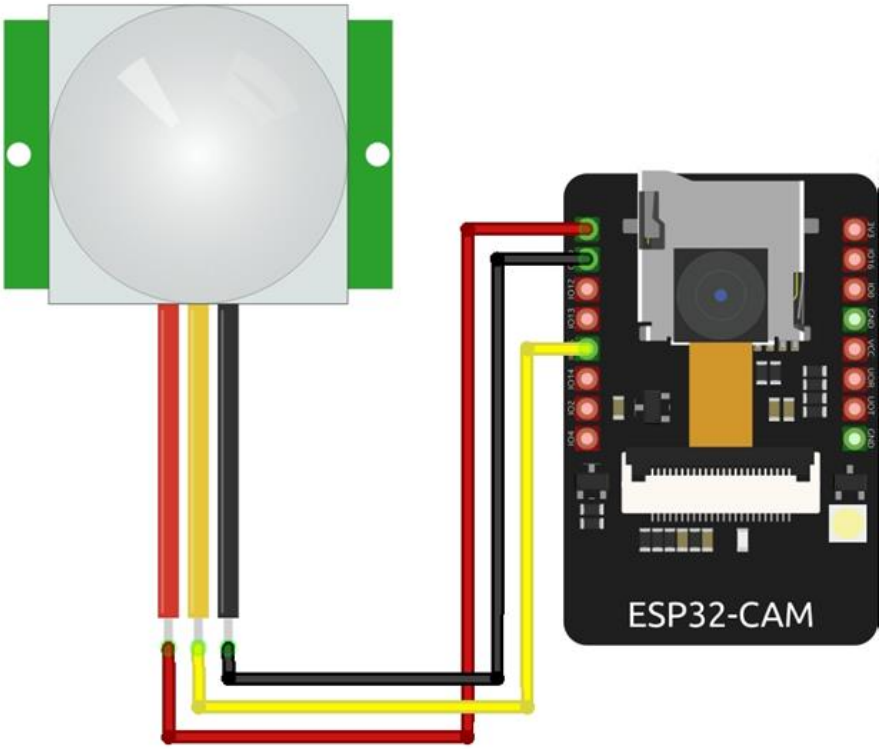


Рис. 20. Схема подключения

Программирование

Arduino IDE

В качестве интегрированной среды разработки для программирования устройства была использована *Arduino IDE* (См. Приложение 4). Изначально разработанная только для программирования микроконтроллеров *Arduino*, данная среда разработки поддерживает другие виды плат встраиваемых систем путем добавления соответствующих библиотек. Программный код пишется на *C/C++* с добавлением дополнительных функций *Arduino*, которые также известны как *Arduino API*.

Несмотря на то, что обычно для плат *ESP32-CAM* код пишется на языке программирования *LUA*, они также поддерживают код, написанный в *Arduino IDE*. Поддержка работы с платами *ESP32-CAM* осуществляется с помощью добавления соответствующих библиотек (См. Приложение 5), после чего следует выбрать соответствующий тип платы в списке устройств (См. Приложение 6).

Программатор *FTDI FT232RL*

Так как ESP32-CAM не имеет разъема для подключения к компьютеру, для программирования платы требуется программатор.

Конвертер *USB-UART* (Рис. 21) на основе микросхемы *FT232RL* предназначен для организации виртуального *COM*-порта с уровнями *TTL* для взаимодействия с внешними устройствами [15].

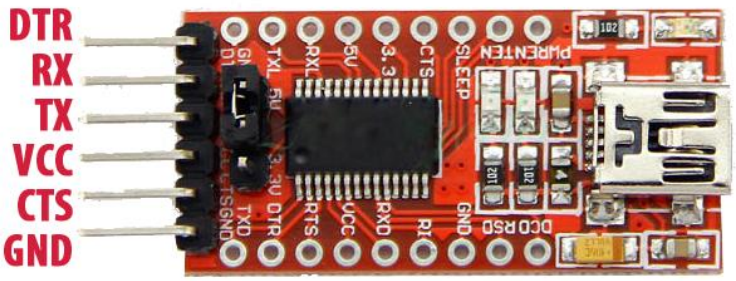


Рис. 21. *FTDI FT232RL*

Характеристики [15][16]

- Вес: 5 г;
- Габариты: 43 × 18 × 12 мм;
- Микросхемы:
  - *FT232RL*;
  - *SN75176*;
- Рабочее напряжение: 3,3 и 5 В;
- Потребляемый ток: 50 мА;
- Шаг: 2.54 мм;
- Интерфейс: *mini USB*;

Установка драйверов

При подключении к компьютеру чип *FT232RL* не распознается как *COM*-порт (что требуется для загрузки кода на плату) (Рис. 22). Для этого требуется драйвер для распознавания устройства как виртуальный *COM*-порт, который доступен для скачивания на сайте производителя чипа – фирмы *FTDI* [17]. После скачивания и выбора драйверов в «Диспетчере устройств», *FT232RL* готово к работе (Рис. 23) [18].

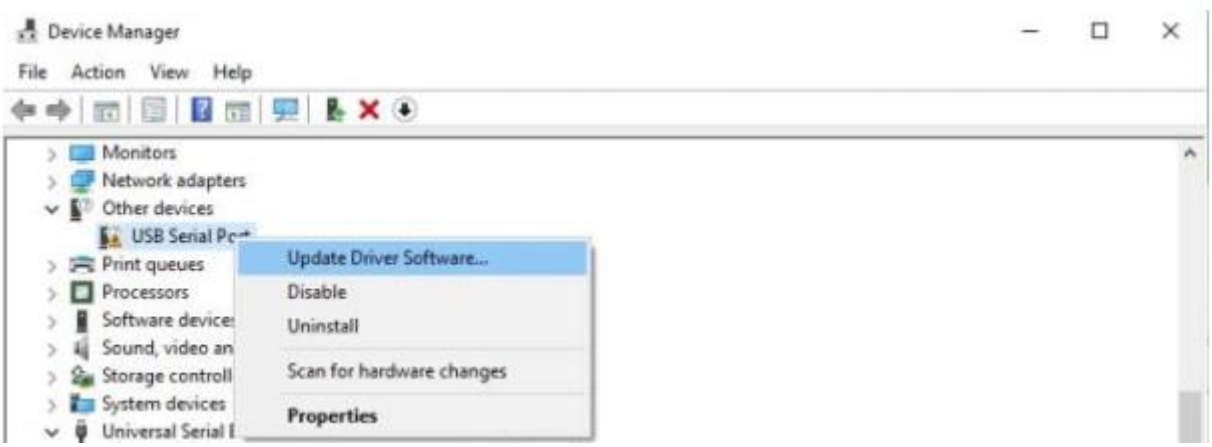


Рис. 22. Нераспознанное устройство при первом подключении

Рис. 23. Распознавание *FT232RL* как *COM*-порт после установки драйверов

Подключение

Для подготовки устройства к программированию помимо подключения к программатору, требуется замкнуть пины *IO0* и *GND* на плате. Также важно, чтобы выбранное напряжение на плате совпадало с напряжением на программаторе – оба напряжения должны быть 5 В или 3.3 В (Рис. 24) [9][16].

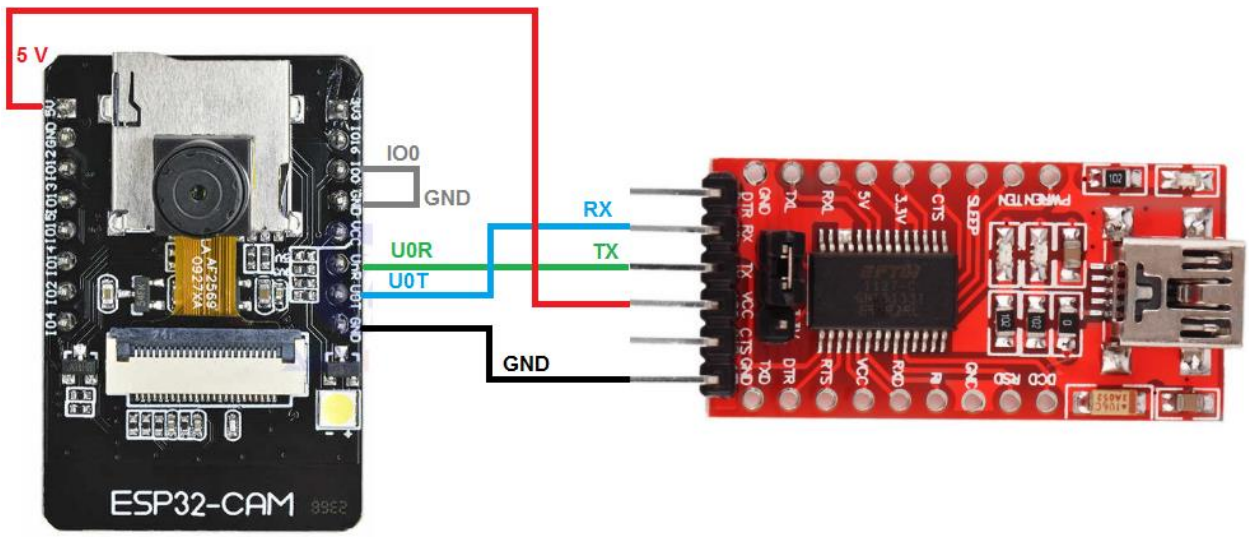


Рис. 24. Схема подключения

Алгоритм работы

Алгоритм работы устройства представлен в виде блок-схема на Рис. 25. Важно отметить, что программный код встроенных систем выполняется в бесконечном цикле: прекращение выполнения программы произойдет только в случае получения ошибки.

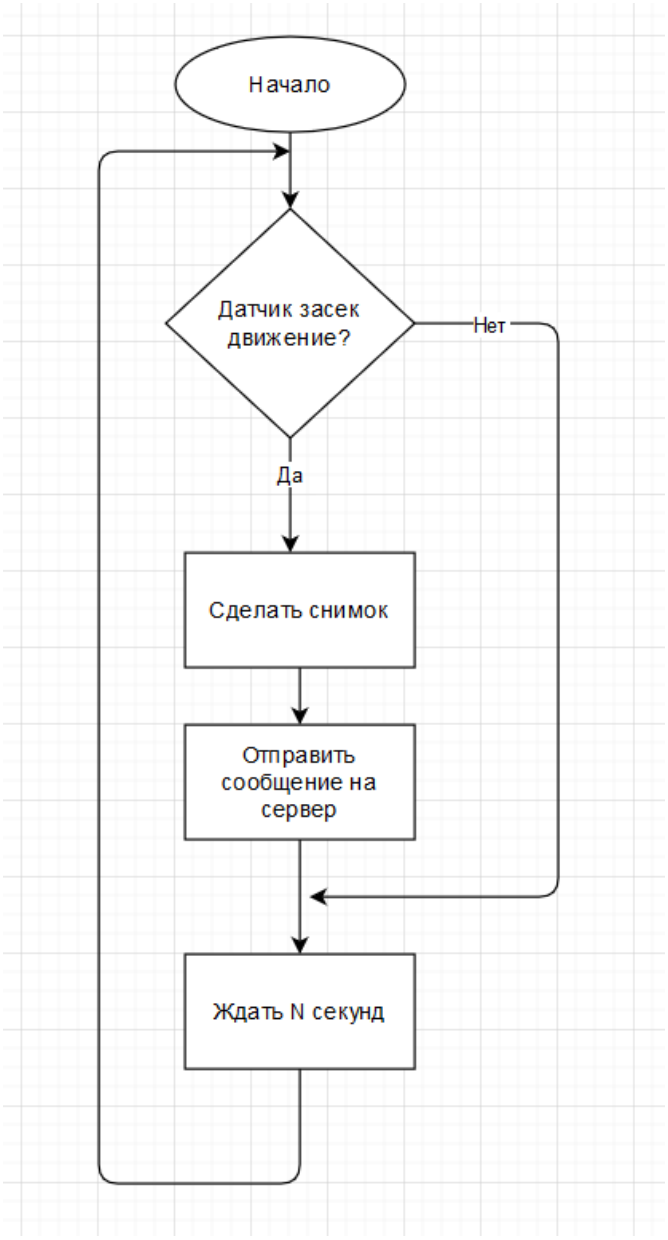


Рис. 25. Блок-схема алгоритма работы устройства

Серверная часть

База данных

База данных приложения состоит из двух таблиц (Рис. 26). Первая хранит список пользователей с закрепленными за ними устройствами. Это было сделано за тем, чтобы в будущем система была расширяемой и поддерживала работу со многими пользователями. Вторая таблица хранит историю всех событий: от какого устройства поступил сигнал, время поступления сигнала и был ли сигнал подтвержден пользователем (данное поле имеет логический тип данный и имеет значение по умолчанию *False*).

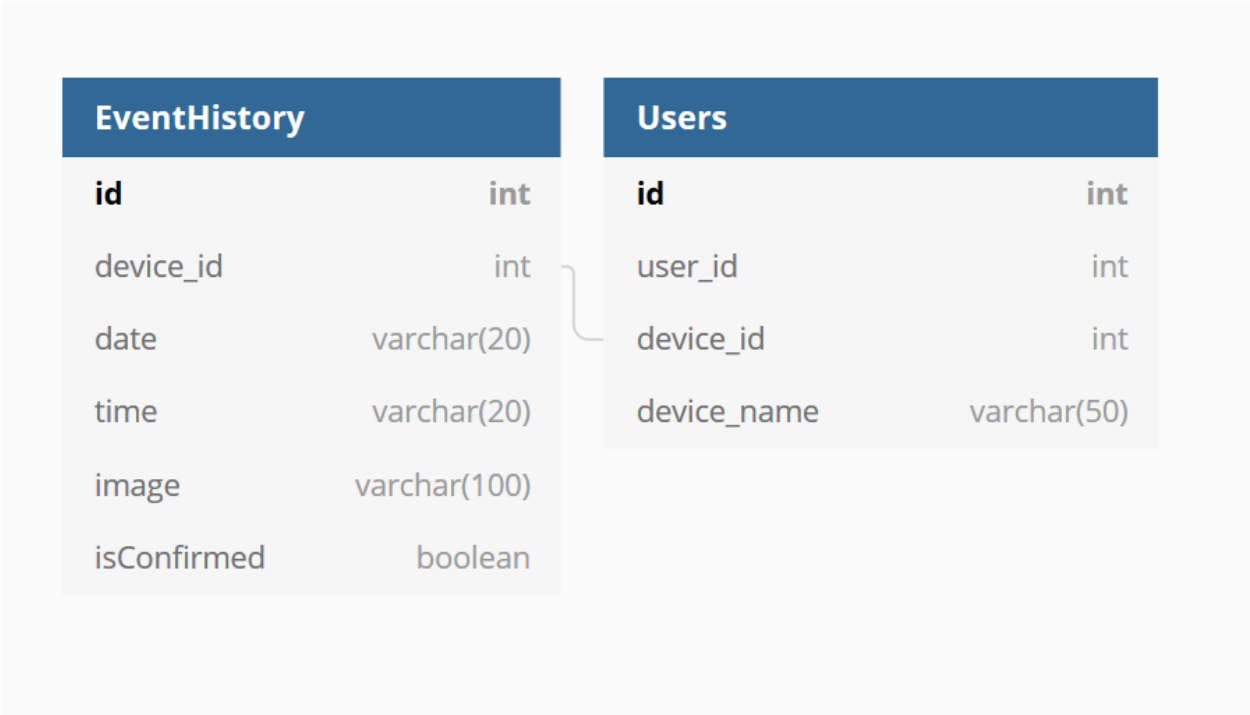


Рис. 26. База данных

В качестве СУБД в данном проекте будет использоваться *PostgreSQL* – свободная объектно-реляционная СУБД, находящаяся в открытом доступе.

Node.js, Express

Node.js представляет среду выполнения кода на *JavaScript*, которая построена на основе движка *JavaScript Chrome V8*, который позволяет транслировать вызовы на языке *JavaScript* в машинный код. *Node.js* прежде всего предназначен для создания серверных приложений на языке *JavaScript*.

В свою очередь, *Express* является фреймворком веб-приложений для *Node.js* и в данном проекте будет использоваться для создания *API* для сервера. Причиной выбора данной технологии для создания *API* обуславливается относительной простотой разработки.

Хостинг

Для упрощения развертывания работы сервера было принято решение об использовании хостинга. В данном проекте будет использоваться хостинг *vdsina*[7], который был выбран из-за относительно низкого тарифа и простоты работы.

В качестве операционный системы сервера был выбран *Ubuntu* 20.04 (Рис. 27).

[7] https://vdsina.ru/

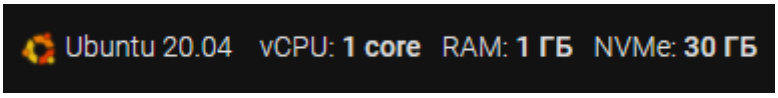


Рис. 27. Характеристики сервера

Приложение

Проектирование интерфейса приложения

В качестве средства для дизайна интерфейса приложения был использован онлайн-сервис *Figma*[8] - векторный графический редактор и средство прототипирования. В нем были созданы следующие дизайны основных экранов приложения.

Главный экран, который видит пользователь после открытия приложения (См. Приложение 2), отображает полученные устройством сигналы за последний день с соответствующими кнопками для их подтверждения и с полученным изображением с камеры в момент получения сигнала. При подтверждении конкретного сигнала, кнопка заменяется соответствующим символом, отображающим, что подтверждение получено.

В левом верхнем углу находится кнопка навигационного меню, при нажатии на которую отображается меню для навигации по приложению (Рис. 28): пункты «Последние события» – главный экран приложения, «История событий» и «Настройки».

[8] https://www.figma.com/

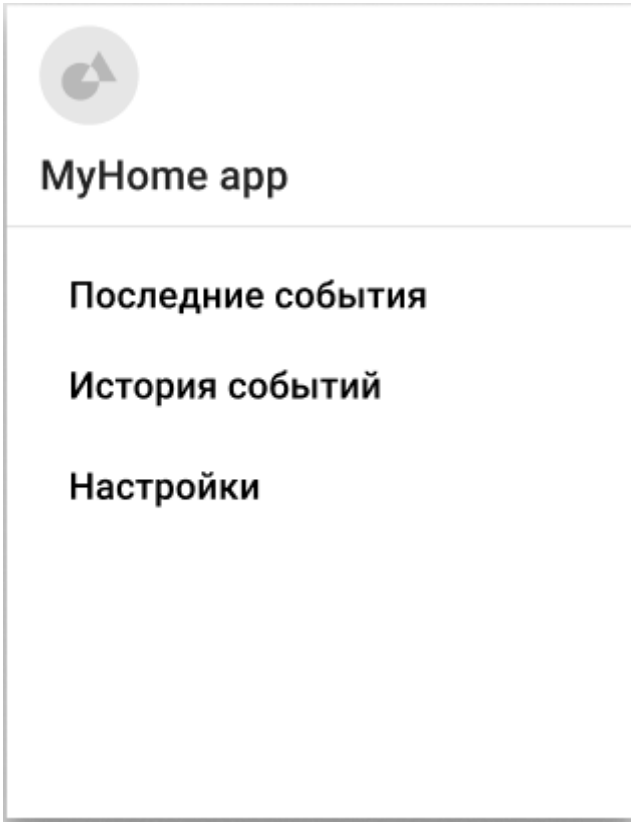


Рис. 28. Меню навигации

История событий (См. Приложение 3) содержит в себе всю историю полученных сигналов с их статусом подтверждения, отсортированную по времени и дате.

Настройки (Рис. 29) позволяют отключить получение сигналов с устройства, включить уведомления на смартфоне и содержат в себе опцию очистки истории.

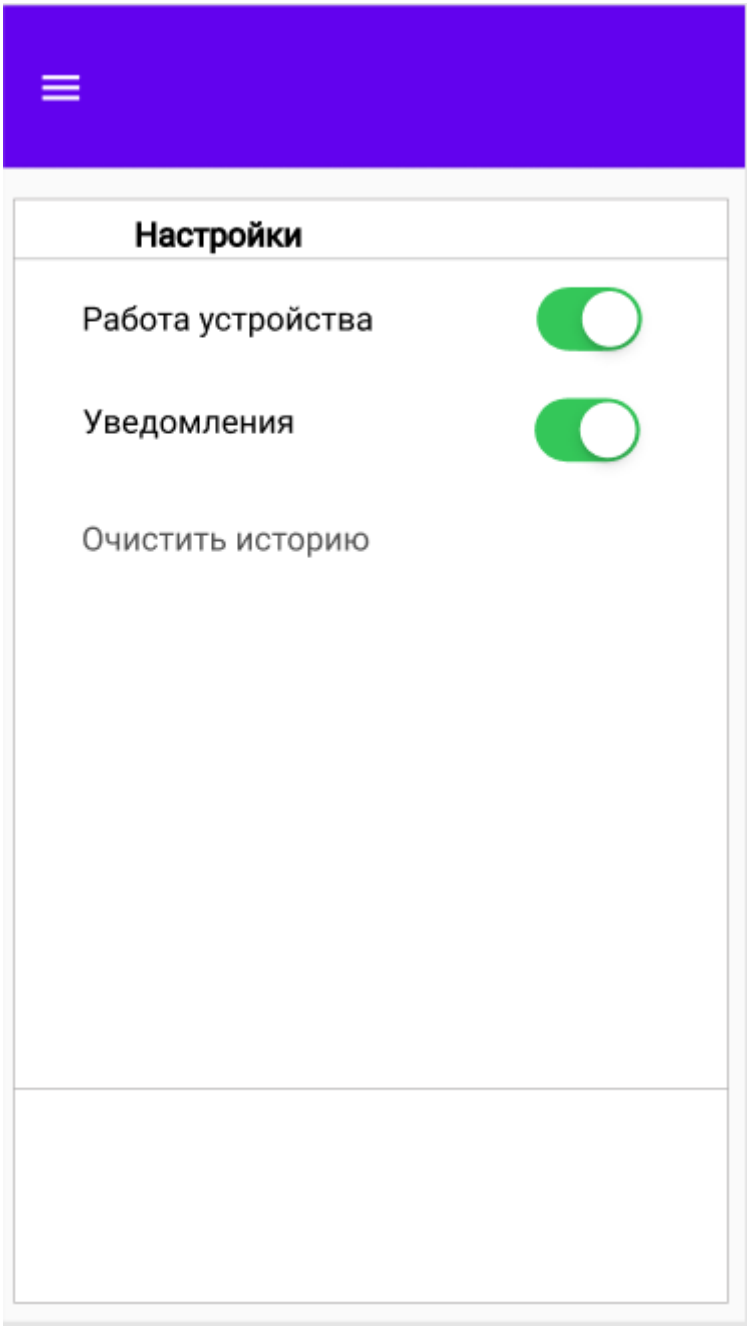


Рис. 29. Экран настроек

Разработка приложения

Как и было указано в разделе «Введение» разрабатываемое приложение было написано на языке программирования *Kotlin*. В качестве среды разработки была выбрана Android Studio (См. Приложение 7), которая является стандартным выбором при разработке мобильных приложений на языках *Java* и *Kotlin* из-за общего удобства разработки и встроенного эмулятора *Android* (См. Приложения 8, 9).

При разработке приложения использовался шаблон проектирования *Model-View-ViewModel (MVVM)*, суть которого заключается в разделении объектов на 3 группы (Рис. 30) [19]:

- *Model* – в данной группе хранятся вся бизнес-логика и данные;

- *View* – отвечает за презентацию данных и средств управления на экране;
- *View model* – преобразует данные из модели в значения, которые может отобразить *View*;

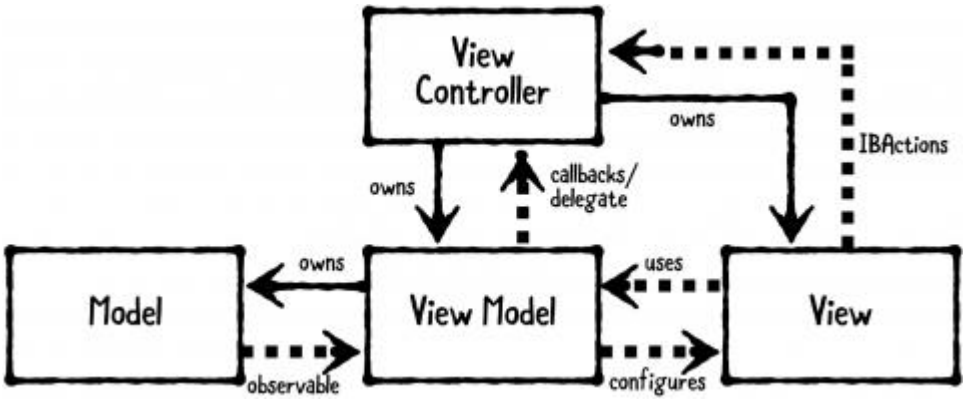


Рис. 30. Паттерн *MVVM*

В приложении этот паттерн осуществляется через использование классов *Fragment* и *ViewModel*. Класс *Fragment* отвечает за презентационный слой, *ViewModel* – по назначению идентичен одноименной группе объектов из паттерна *MVVM*.

Для отправки запросов на сервер была использована библиотека *Retrofit* – типобезопасный *HTTP* клиент, который значительно упрощает получение данных благодаря преобразованию *HTTP API* в интерфейс *Java/Kotlin* и обработке данных, полученных от сервера в формате *JSON* [20].

Для загрузки изображений с сервера была использована библиотека *Glide,* использующаяся для асинхронного получения изображений из сети, ресурсов, файловой системы, их кэширования и отображения [21].

Общий дизайн графического интерфейса приложения не претерпел крупных изменений: внешний вид главного экрана, страницы истории и настроек мало отличаются от разработанных макетов (См. Приложения 10, 11, 13). Стоит отметить добавление выпадающего списка на страницах с отображением историй сообщений (Рис. 31), позволяющего отображать записи с конкретного устройства (а не только со всех принадлежащих пользователю по умолчанию). Это было сделано для большего удобства использования приложения, так как при большом количестве устройств пользователя может затрудниться восприятие полученных данных.

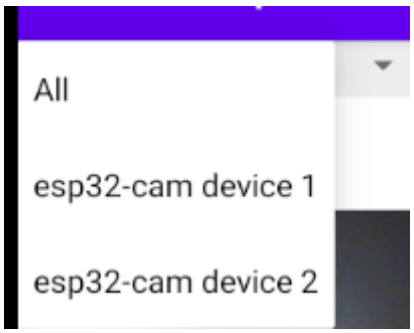


Рис. 31. Выпадающий список устройств

При включенной настройке уведомлений приложение также будет отправлять уведомления при работе в фоновом режиме (Рис. 32).

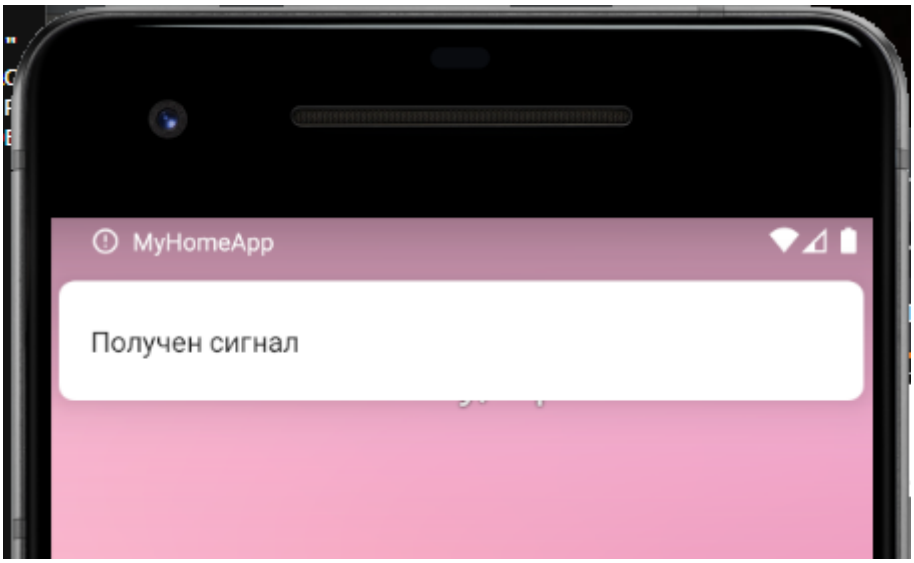


Рис. 32. Уведомление

Рассмотрение результатов

В результате разработки была создана система, полностью соответствующая изначальным требованиям:

- Размеры устройства не превышают поставленные 70*70*30 мм (*HC-SR501 PIR*: 32.2*24.3*25.4 мм, *ESP32-CAM*: 27*40.5*4.5 мм);
- Цена не превышает 1000 рублей (*ESP32-CAM*: ~500 рублей[9], *HC-SR501 PIR*: ~75 рублей[10], *FTDI FT232RL* – 289 рублей [15]);
- Устройство снабжено камерой;
- Для просмотра данных было разработано отдельное приложение;

Разработанная система успешно справляется с поставленной задачей, позволяя пользователю выполнять установленные в требованиях действия такие, как просмотр последних сообщений устройства с изображениями с камеры, полученными в момент получения сигнала с датчика движения, просмотр истории записей, подтверждение сообщений, сортировка записей по устройству, настройка уведомлений, включение/выключение получения новых сообщений с сервера, очистка истории сообщений.

Для работы системы требуется подключение к сети Интернет всех ее элементов. Также обязательна регистрация пользователя и его устройств перед использованием системы.

Перспективы развития

Несмотря на соответствие поставленным требованиям, данная система имеет моменты, которые могут быть улучшены: например, добавление новых элементов интерфейса приложения, позволяющие отсортировать полученный список по статусу («Подтвержденные»/ «Не подтвержденные»), добавление настройки типа уведомлений («Компактные»/ «Подробные» (с уменьшенным изображением)) и т.д.

Также стоит отметить, что на данный момент в системе отсутствует средство регистрации помимо прямого редактирования базы данных на сервере. Имплементация подобного элемента обязательна в случае применения системы не только для личного пользования. Регистрацию пользователя можно реализовать в виде отдельной формы на сервере, либо при первом входе в приложение. Одним из вариантов регистрации устройства является получение идентификационного номера, генерируемого на сервере, что приме-

[9] https://www.aliexpress.com/item/1005001978372728.html
[10] https://www.aliexpress.com/item/32699216549.html

нялось в рассмотренном решении *ESP8266 PIR Home Security & Notifier*. Данный номер, так же, как и в проекте, далее следует указать непосредственно в коде самого устройства.

Так как серверной части не принципиален тип устройства, с которого получаются сообщения, одним из вариантов развития работы является поддержка других типов плат, таких как *Arduino, NodeMCU* и *Raspberry Pi*, которые рассматривались в качестве используемого в проекте типа платы. Однако следует взять во внимание, что не все типы камер поддерживаются каждой из плат: *Raspberry Pi* имеет свой тип камеры, который не поддерживается *Arduino* и *NodeMCU*.

Весь исходный код проекта будет загружен в открытый репозиторий на *GitHub*[11], что позволит другим пользователям использовать и модифицировать код данной работы.

Корпус устройства

На данном этапе разработки существенным недостатком устройства является отсутствие какого-либо корпуса для обеспечения целостности конструкции, защиты компонентов от физических повреждений, упрощения установки и общего улучшения внешнего вида устройства.

Для решения данной проблемы был спроектирован вариант дизайна корпуса устройства, который не является финальным. Для моделирования деталей использовалась программа *SolidWorks*. Главной задумкой данного дизайна была его максимальная простота для возможности *3D* печати частей корпуса пользователем. Также корпус должен обеспечить достаточное пространство для устройства и всех его компонентов и независимого источника питания в виде трех *AAA* батареек.

Корпус состоит из двух деталей: задней (Рис. 33) и передней (Рис. 34) крышек. Сборка осуществляется посредством простой вставки крышек друг в друга. На задней крышке расположены отверстия для крепления (также для крепления возможно использовать двусторонний скотч). Передняя крышка содержит два отверстия – для датчика движения и для камеры.

[11] https://github.com/Kola14/MyHomeApp

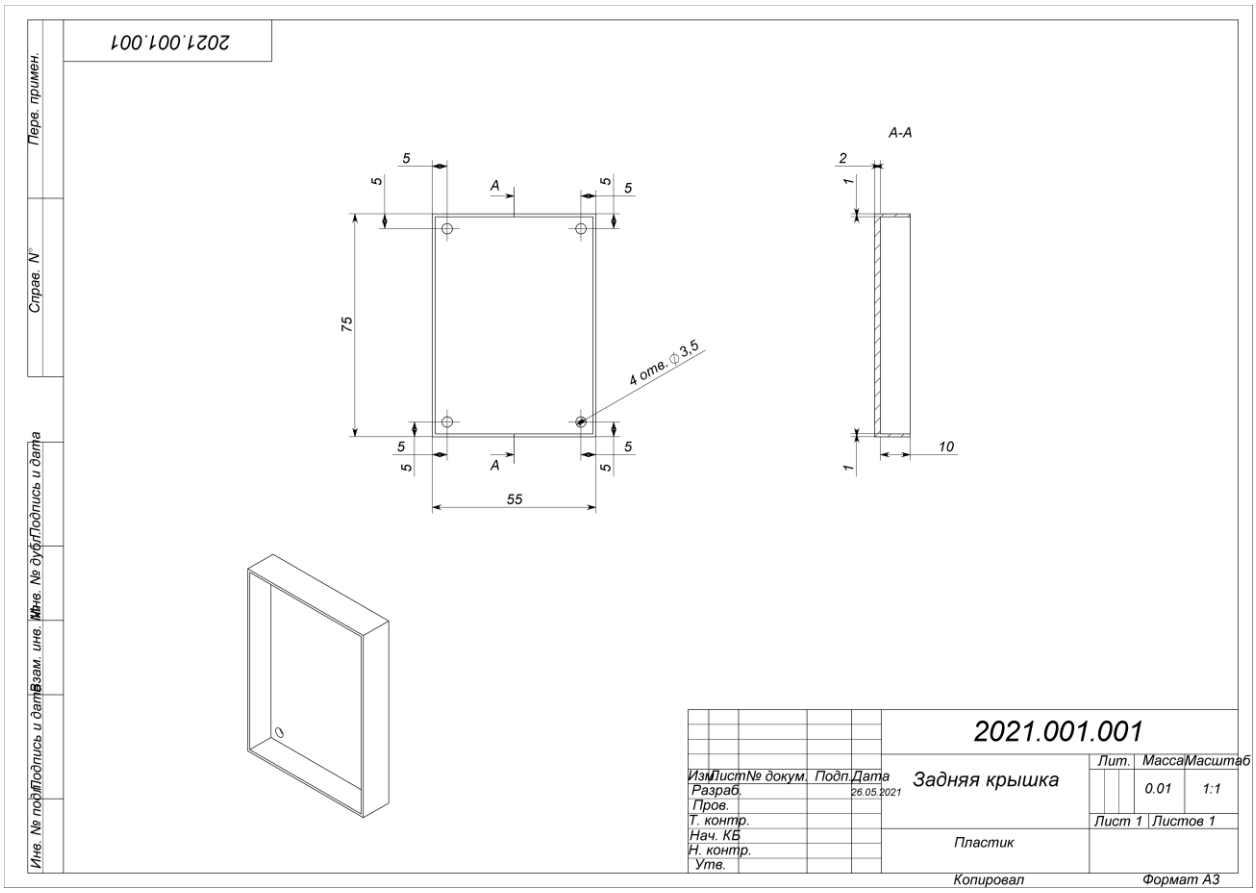


Рис. 33. Задняя крышка корпуса

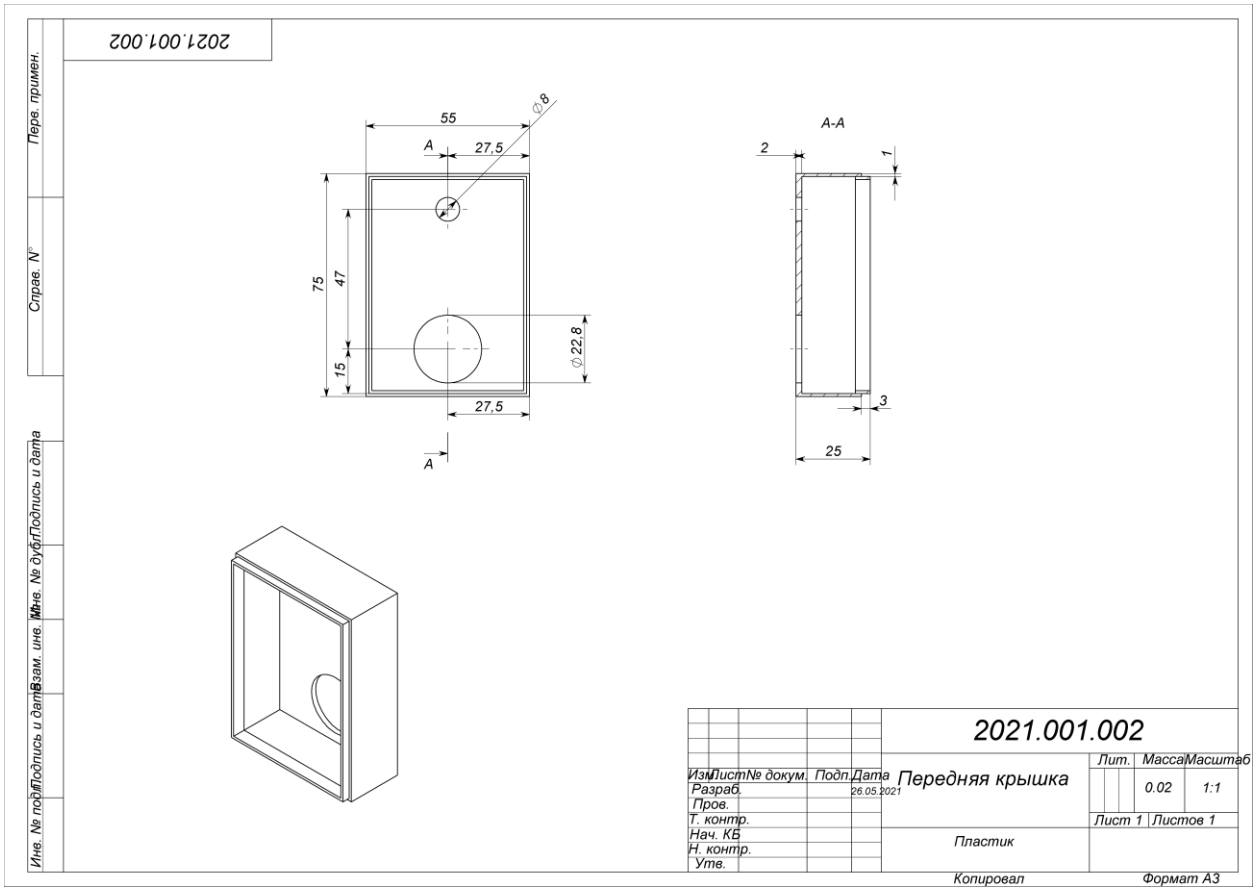


Рис. 34. Передняя крышка корпуса

Недостатком данного дизайна корпуса является превышение установленных в требованиях ограничений по размеру: общий размер корпуса (Рис. 35) 75*55*34 мм, высота превышает требуемые 70 мм на 5 мм, глубина превышает установленные 30 мм на 4 мм.

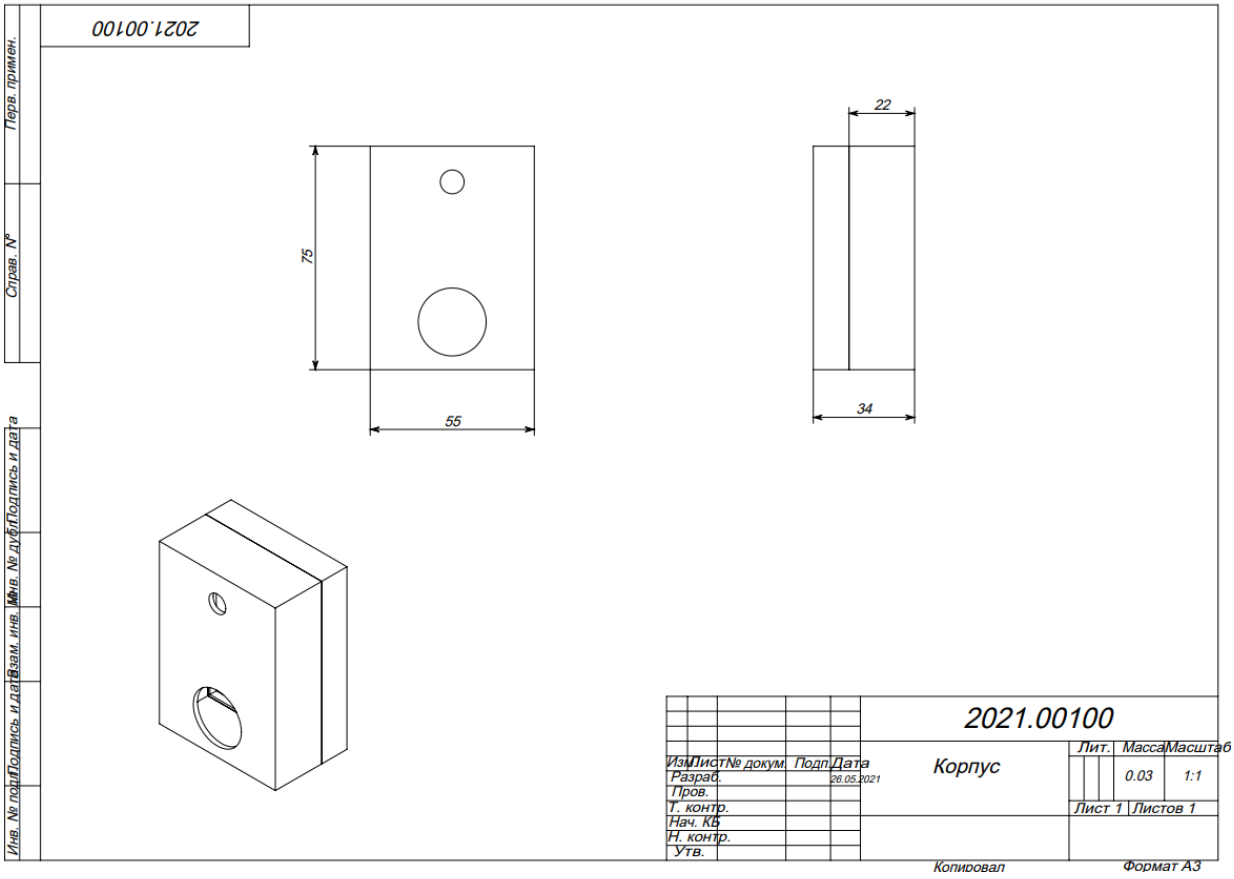


Рис. 35. Общие размеры корпуса

Общий вид корпуса представлен на Рис. 36. Передняя крышка была сделана полупрозрачной ради демонстрации внутреннего вида корпуса и возможного положения независимого источника питания.

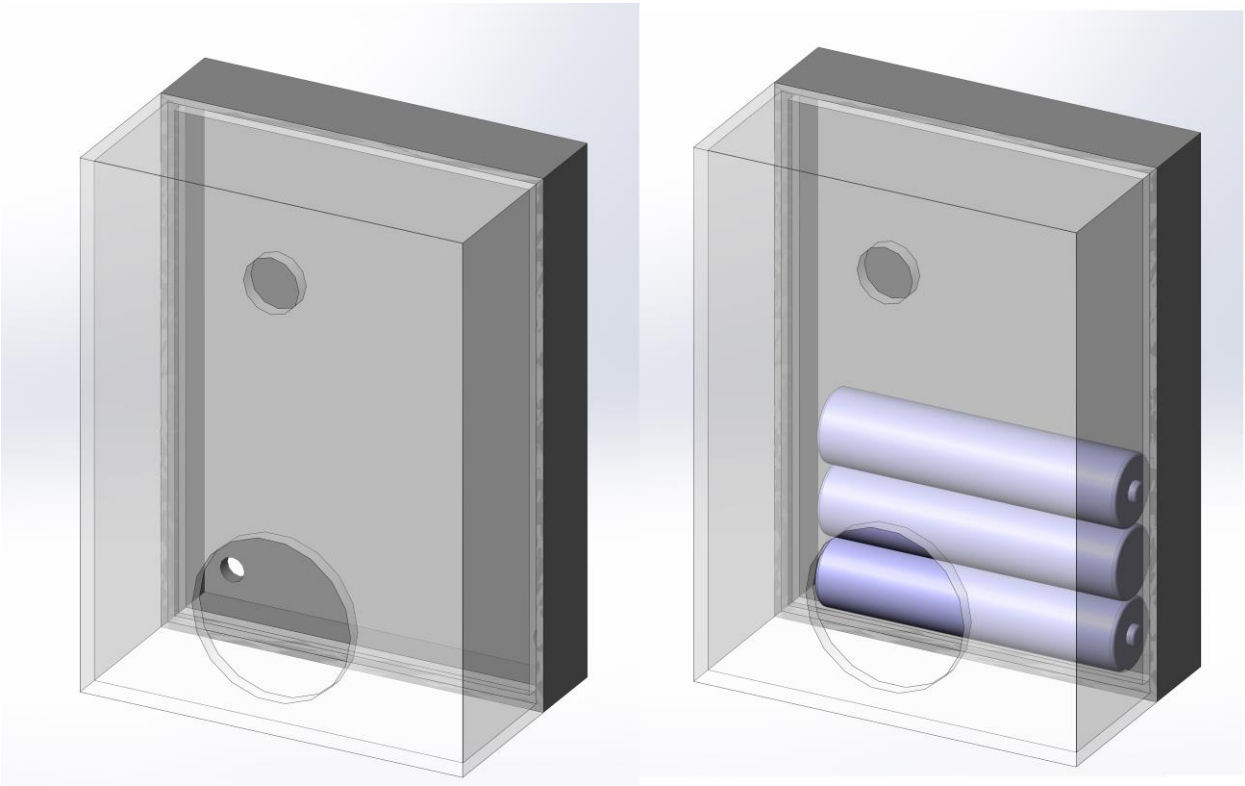


Рис. 36. Собранный корпус

Созданные модели деталей будут также выложены в свободный доступ для *3D* печати любыми желающими[12].

[12] https://github.com/Kola14/MyHomeAppExtra

Заключение

Таким образом, созданная в ходе работы система справляется со своей задачей в предоставлении пользователю сообщений об обнаружении признаков движения в помещении с отображением соответствующего изображения, полученного устройством в момент получения сигнала с датчика движения. Устройство состоит из недорогих и общедоступных компонентов, которые может приобрести любой желающий для имитации данного проекта. От пользователя лишь требуется создание собственного сервера с использованием сервисов хостинга (как это было сделано в данном проекте) или использования отдельной уже имеющейся машины в качестве сервера.

В ходе работы были рассмотрены уже имеющиеся на данный момент решения как коммерческие, так и некоммерческие, которые стали основой для формирования списка требований для разрабатываемой системы. Были описаны компоненты системы, их характеристики и принципы работы.

Разработанная система имеет множество потенциалов развития такие, как доработка возможностей графического интерфейса приложения, поддержка других типов устройств таких, как *Raspberry Pi* и другие и т.д. Любой пользователь может просмотреть исходный код приложения и использовать его и/или внести правки в программный код.

Список литературы

1. Alexander S. Gillis. internet of things (IoT) [Электронный ресурс] — Электрон. текст. — 2020. — Режим доступа: https://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT, свободный (дата обращения: 28.05.21).
2. International Telecommunication Union. ITU-T Y.4000/Y.2060 [Электронный ресурс] — Электрон. текст. — 2012. — Режим доступа: http://handle.itu.int/11.1002/1000/11559, свободный (дата обращения: 28.05.21).
3. Lionel Sujay Vailshery. Forecast end-user spending on IoT solutions worldwide from 2017 to 2025 [Электронный ресурс] — Электрон. текст. — 2019. — Режим доступа: https://www.statista.com/statistics/976313/global-iot-market-size/#:~:text=The%20global%20market%20for%20Internet,around%201.6%20trillion%20by%202025, свободный (дата обращения: 09.03.21).
4. S. O'Dea. Number of smartphone users worldwide from 2016 to 2023 [Электронный ресурс] — Электрон. текст. — 2020. — Режим доступа: https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/, свободный (дата обращения: 16.05.21).
5. StatCounter. Mobile Operating System Market Share Worldwide [Электронный ресурс] — Электрон. текст. — 2021. — Режим доступа: https://gs.statcounter.com/os-market-share/mobile/worldwide#, свободный (дата обращения 09.03.21).
6. Android Developers. Android’s Kotlin-first approach [Электронный ресурс] — Электрон. текст. — 2021. — Режим доступа: https://developer.android.com/kotlin/first, свободный (дата обращения 22.04.21).
7. OmniSci, Inc. Embedded Systems [Электронный ресурс] — Электрон. текст. — 2021. — Режим доступа: https://www.omnisci.com/technical-glossary/embedded-systems, свободный (дата обращения 28.05.21).
8. Электронный магазин «Вольтик». ESP32-CAM – плата разработки ESP32 с камерой OV2640 [Электронный ресурс] — Электрон. текст. — 2021. — Режим доступа: https://voltiq.ru/shop/esp32-cam-wi-fi-module/, свободный (дата обращения: 05.05.21).
9. Shenzhen Ai-Thinker Technology Co., Ltd. ESP32-CAM Module [Электронный ресурс] — Электрон. текст. — 2017. — Режим доступа: https://loboris.eu/ESP32/ESP32-CAM%20Product%20Specification.pdf, свободный (дата обращения: 05.05.21).
10. Электронный магазин «Robotclass». Камера OV2640 [Электронный ресурс] — Электрон. текст. — 2021. — Режим доступа:

https://shop.robotclass.ru/index.php?route=product/product&product_id=1559, свободный (дата обращения: 05.05.21).

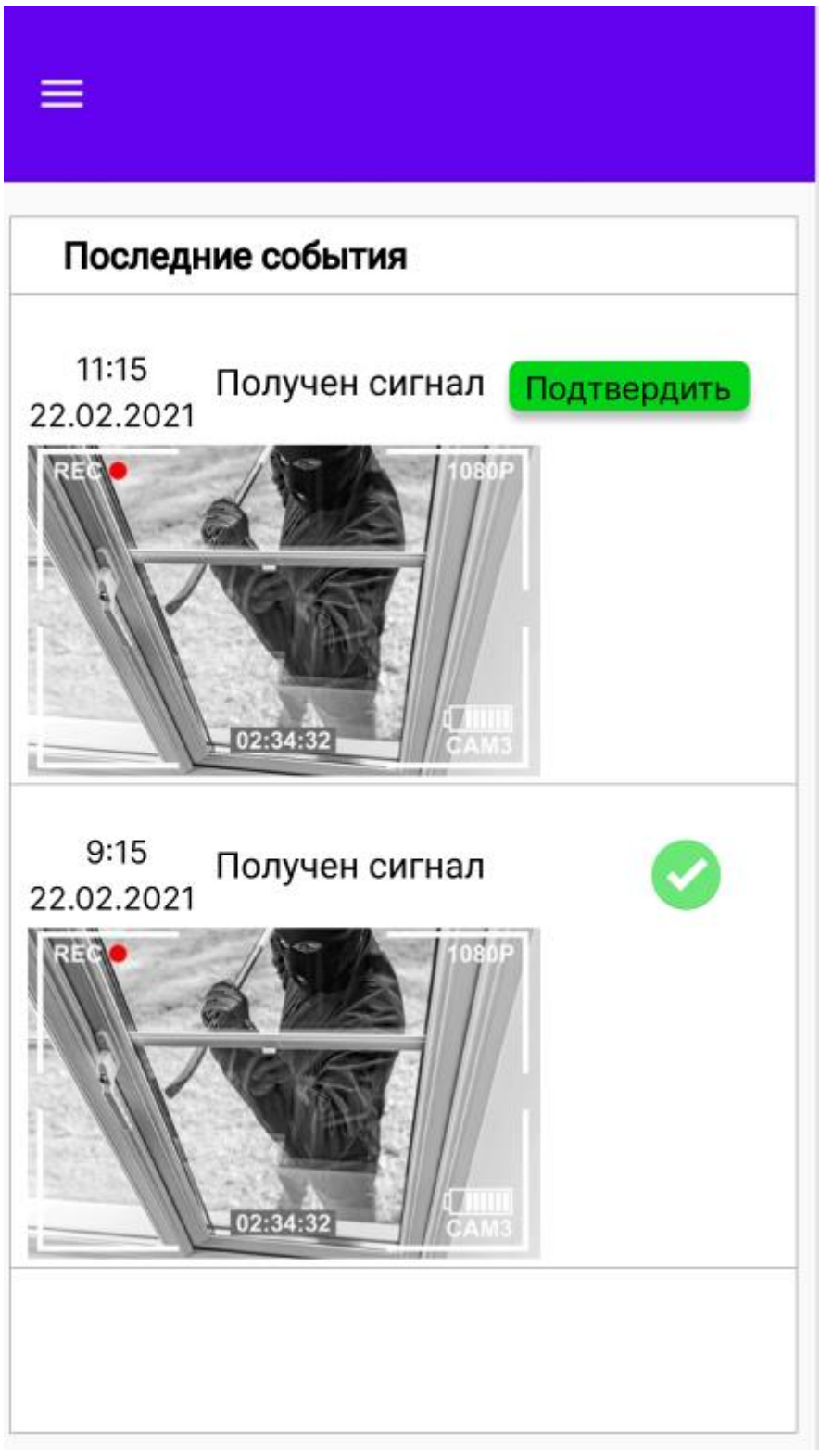
11. OmniVision Technologies. OV2640 Color CMOS UXGA (2.0 MegaPixel) CameraChip with OmniPixel2 Technology [Электронный ресурс] — Электрон. текст. — 2006. — Режим доступа: https://www.waveshare.com/w/upload/6/6b/OV2640_DS%281.6%29.pdf, свободный (дата обращения: 05.05.21).
12. Adafruit Industries. PIR Motion Sensor [Электронный ресурс] — Электрон. текст. — 2014. — Режим доступа: https://learn.adafruit.com/pir-passive-infrared-proximity-motion-sensor, свободный (дата обращения: 05.05.21).
13. Components101. HC-SR501 PIR Sensor [Электронный ресурс] — Электрон. текст. — 2017. — Режим доступа: https://components101.com/sensors/hc-sr501-pir-sensor, свободный (дата обращения: 05.05.21).
14. Adafruit Industries. PIR Motion Sensor [Электронный ресурс] — Электрон. текст. — 2020. — Режим доступа: https://cdn-learn.adafruit.com/downloads/pdf/pir-passive-infrared-proximity-motion-sensor.pdf, свободный (дата обращения: 08.05.21).
15. Электронный магазин «Вольтик». FTDI FT232RL – USB – UART TTL преобразователь [Электронный ресурс] — Электрон. текст. — 2021. — Режим доступа: https://voltiq.ru/shop/usb-uart-converter-ftdi-ft232rl/, свободный (дата обращения: 05.05.21).
16. Future Technology Devices International Ltd. FT232R USB UART IC Datasheet [Электронный ресурс] — Электрон. текст. — 2020. — Режим доступа: https://ftdichip.com/wp-content/uploads/2020/08/DS_FT232R.pdf, свободный (дата обращения: 06.05.21).
17. Future Technology Devices International Ltd. Drivers [Электронный ресурс] — Электрон. текст. — 2021. — Режим доступа: https://ftdichip.com/drivers/vcp-drivers/, свободный (дата обращения: 06.05.21).
18. Future Technology Devices International Ltd. Application Note AN_396 FTDI Drivers Installation Guide for Windows 10 [Электронный ресурс] — Электрон. текст. — 2016. — Режим доступа: https://ftdichip.com/wp-content/uploads/2020/08/AN_396-FTDI-Drivers-Installation-Guide-for-Windows-10.pdf, свободный (дата обращения: 06.05.21).
19. Jay Strawn. Design Patterns by Tutorials: MVVM [Электронный ресурс] — Электрон. текст. — 2018. — Режим доступа: https://www.raywenderlich.com/34-design-patterns-by-tutorials-mvvm, свободный (дата обращения: 08.05.21).

20. Square, Inc. Retrofit - A type-safe HTTP client for Android and Java [Электронный ресурс] — Электрон. текст. — 2021. — Режим доступа: https://square.github.io/retrofit/, свободный (дата обращения: 08.05.21).
21. Sam Judd. Glide [Электронный ресурс] — Электрон. текст. — 2021. — Режим доступа: https://github.com/bumptech/glide, свободный (дата обращения: 08.05.21).

Приложения

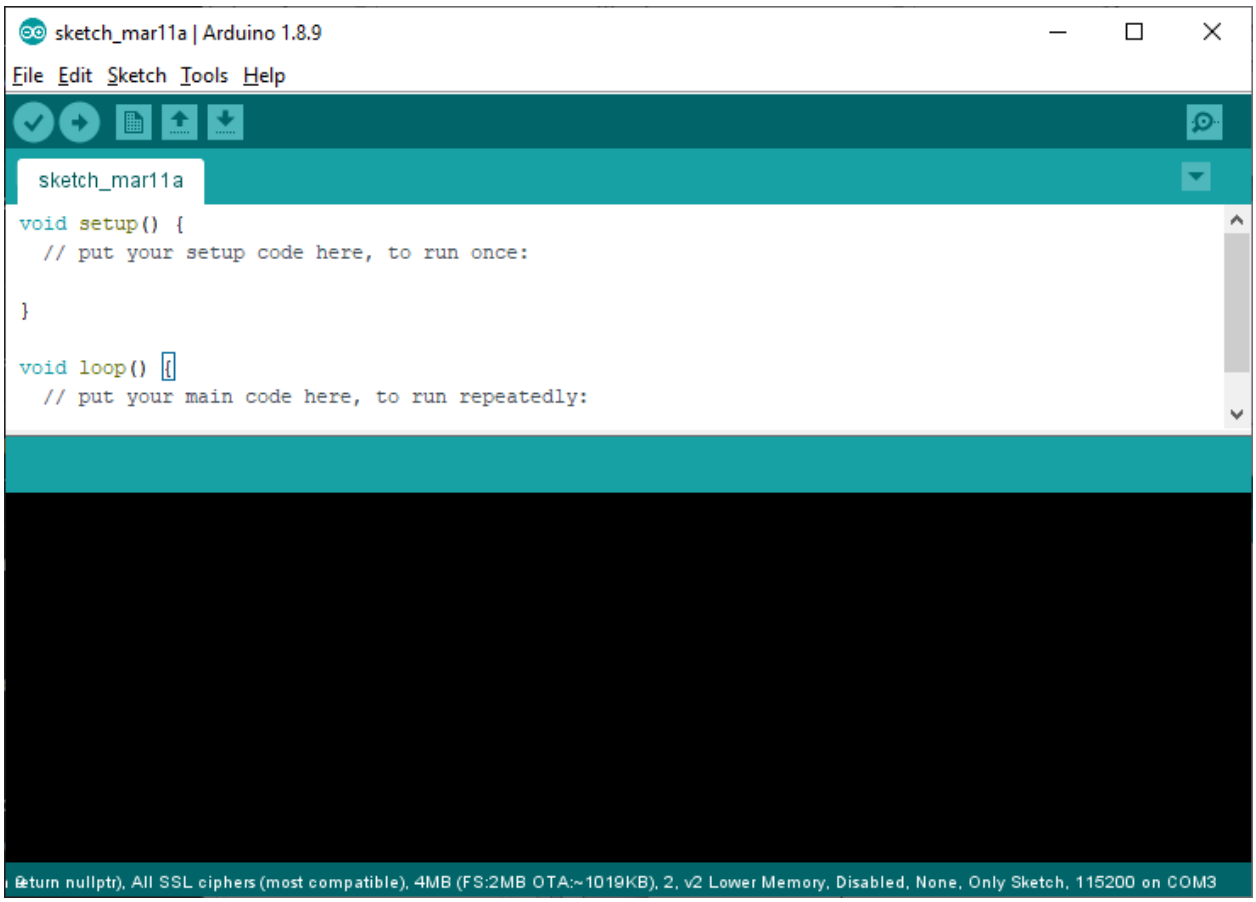
Date	*Android*	*iOS*	*Samsung*	*KaiOS*	*Unknown*	*Windows*	*Nokia Unknown*	*Series 40*	*Linux*	*Tizen*	*BlackBerry OS*	*Other*
2020-02	73.3	25.89	0.18	0.23	0.13	0.12	0.03	0.03	0.03	0.03	0.02	0.02
2020-03	72.26	27.03	0.16	0.32	0.06	0.1	0	0	0.01	0.02	0.01	0.01
2020-04	70.68	28.79	0.17	0.12	0.09	0.07	0.02	0.02	0.01	0.02	0.01	0.01
2020-05	72.6	26.72	0.21	0.2	0.11	0.03	0.02	0.02	0.02	0.02	0.01	0.01
2020-06	74.14	25.26	0.21	0.13	0.11	0.04	0.02	0.02	0.02	0.02	0.01	0.01
2020-07	74.6	24.82	0.21	0.1	0.13	0.04	0.02	0.02	0.02	0.02	0.01	0.01
2020-08	74.25	25.15	0.23	0.08	0.13	0.03	0.03	0.03	0.02	0.02	0.01	0.01
2020-09	74.44	24.98	0.22	0.08	0.14	0.03	0.03	0.02	0.02	0.02	0.01	0.02
2020-10	72.92	26.53	0.22	0.07	0.13	0.03	0.02	0.02	0.02	0.01	0.01	0.01
2020-11	71.18	28.19	0.24	0.13	0.14	0.03	0.02	0.02	0.02	0.01	0.01	0.01
2020-12	72.48	26.91	0.23	0.13	0.14	0.02	0.02	0.02	0.02	0.01	0.01	0.01
2021-01	71.93	27.47	0.28	0.1	0.13	0.02	0.02	0.02	0.02	0.01	0.01	0.01
2021-02	71.9	27.33	0.39	0.14	0.14	0.02	0.02	0.02	0.02	0.01	0.01	0.01

Приложение 1. Процентное соотношение операционных систем для смартфонов

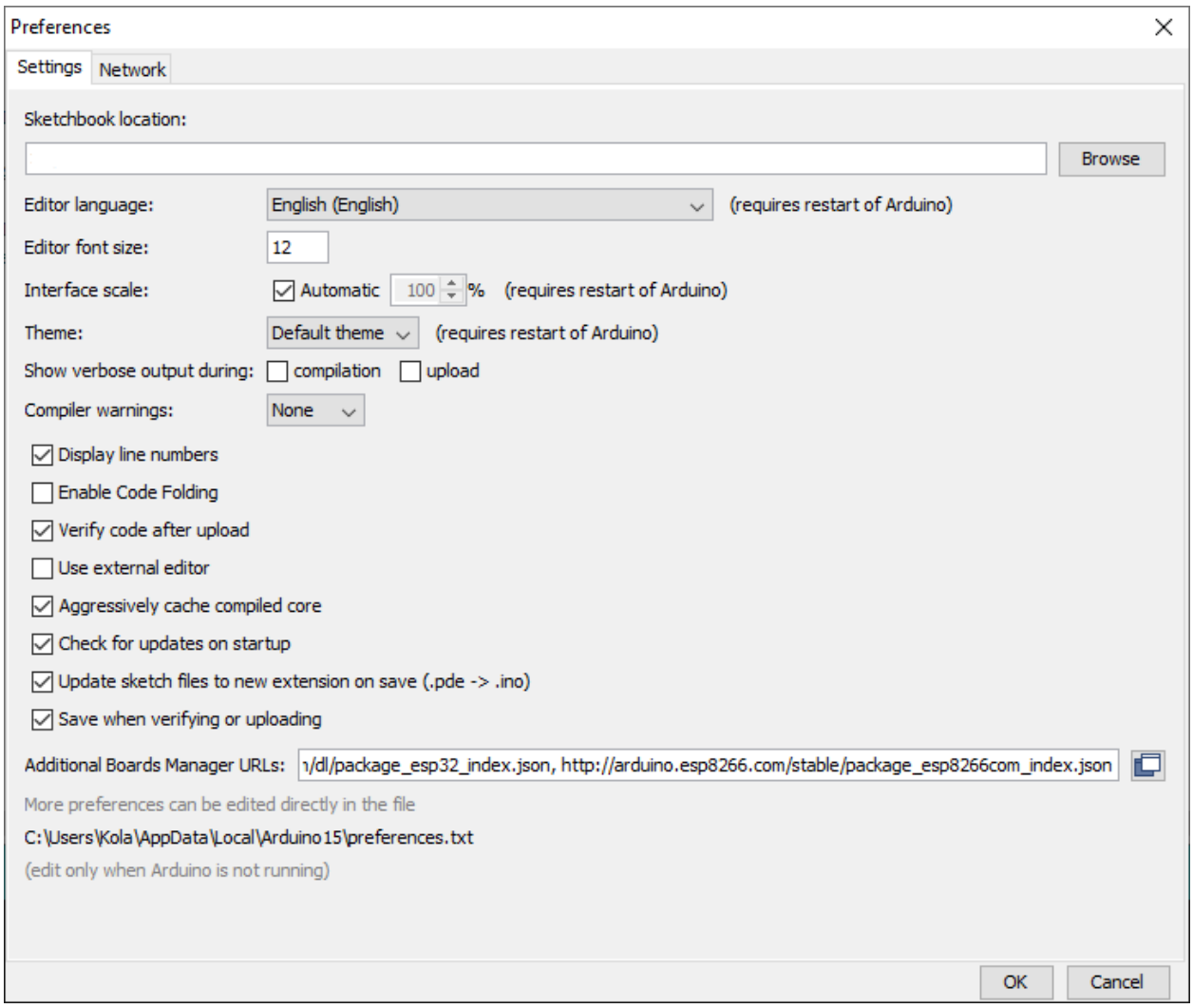


Приложение 2. Главный экран

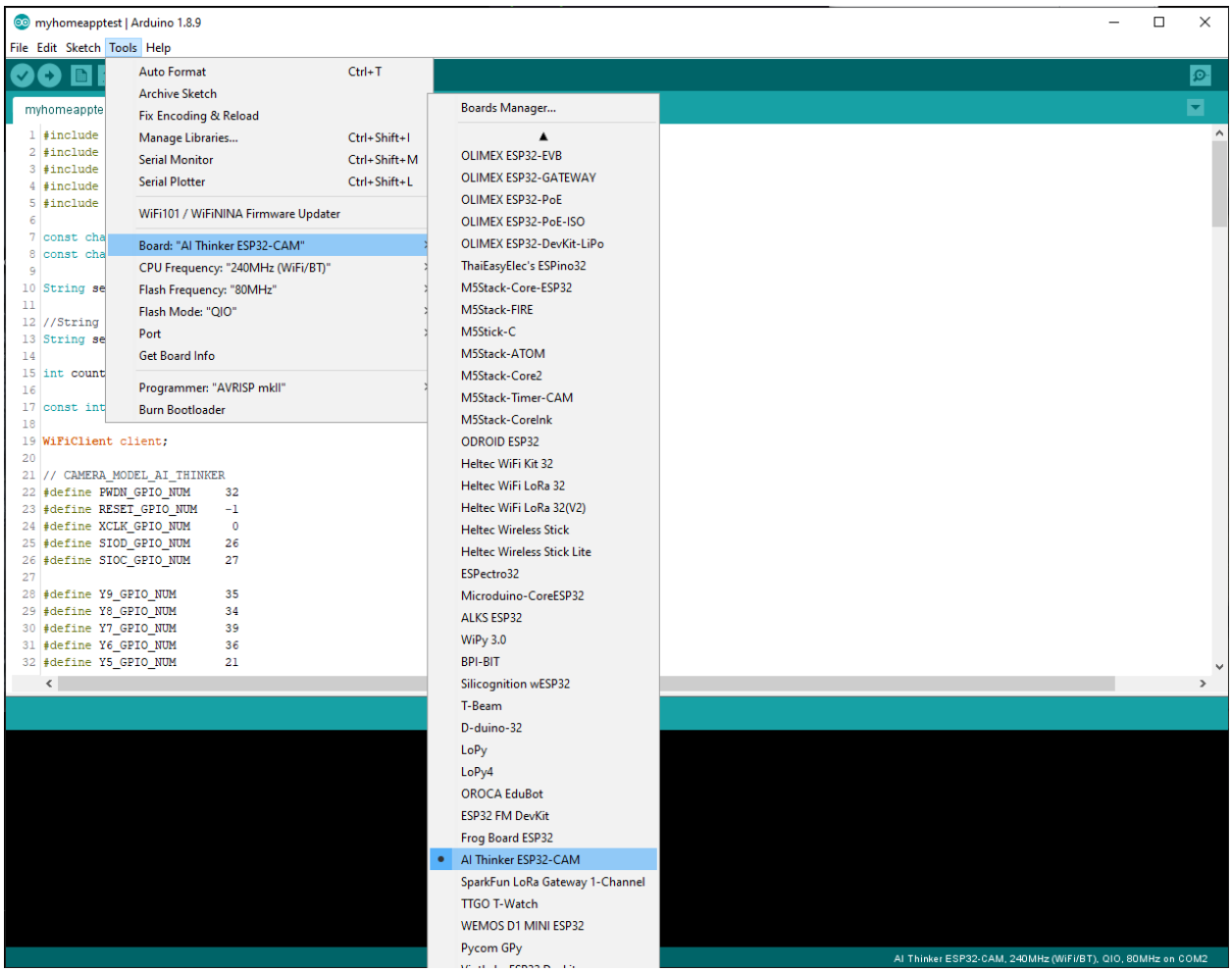
Приложение 3. История событий



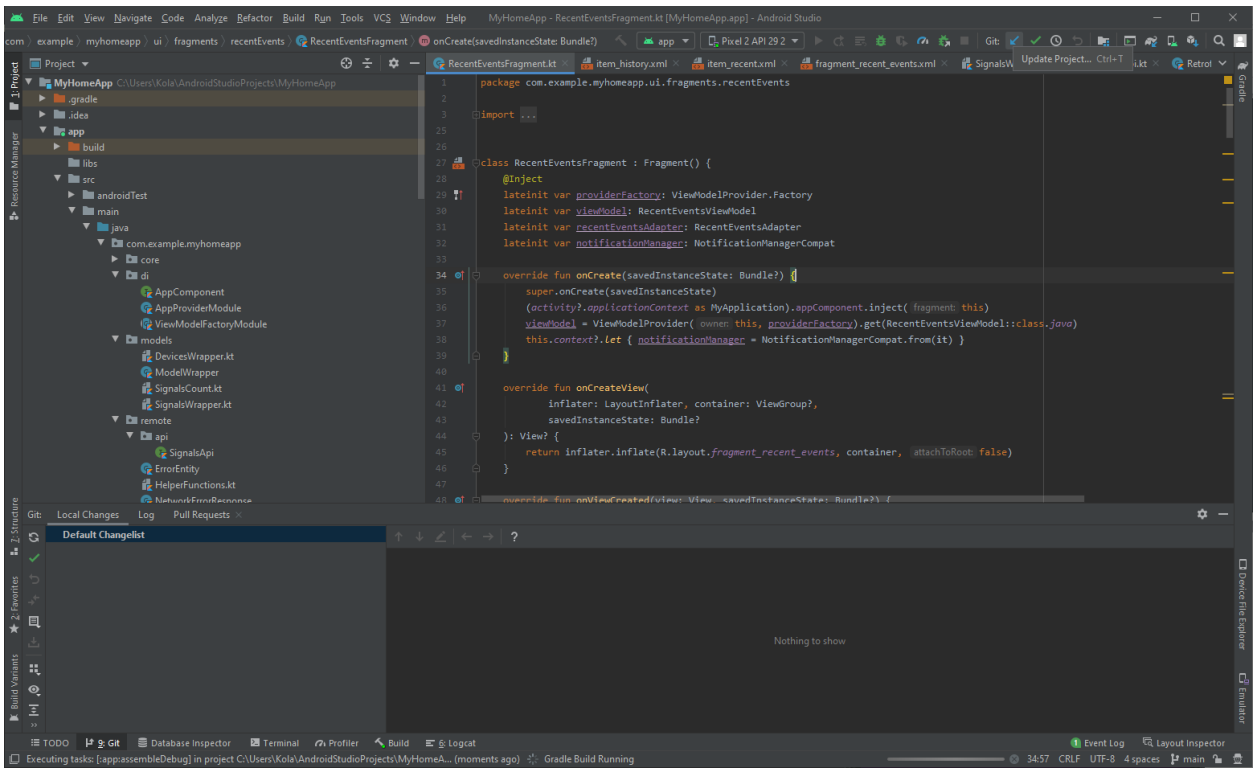
Приложение 4. Интерфейс программы



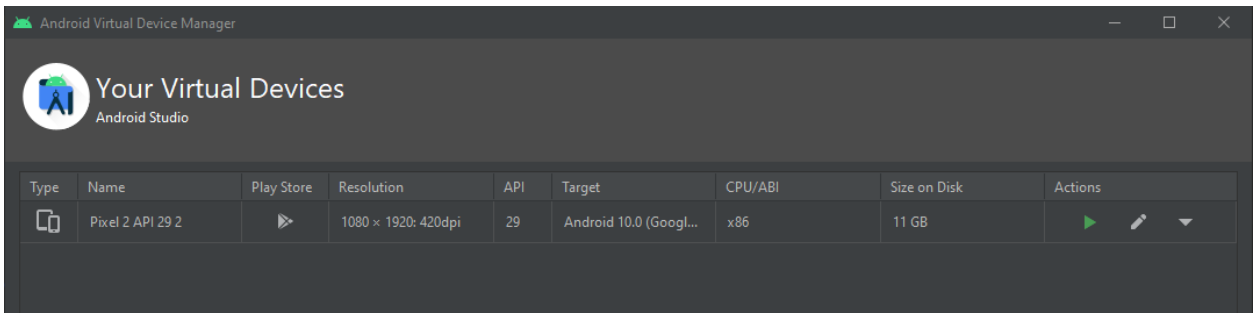
Приложение 5. Загрузка библиотек для работы с платой *ESP32-CAM*



Приложение 6. Выбор типа устройства

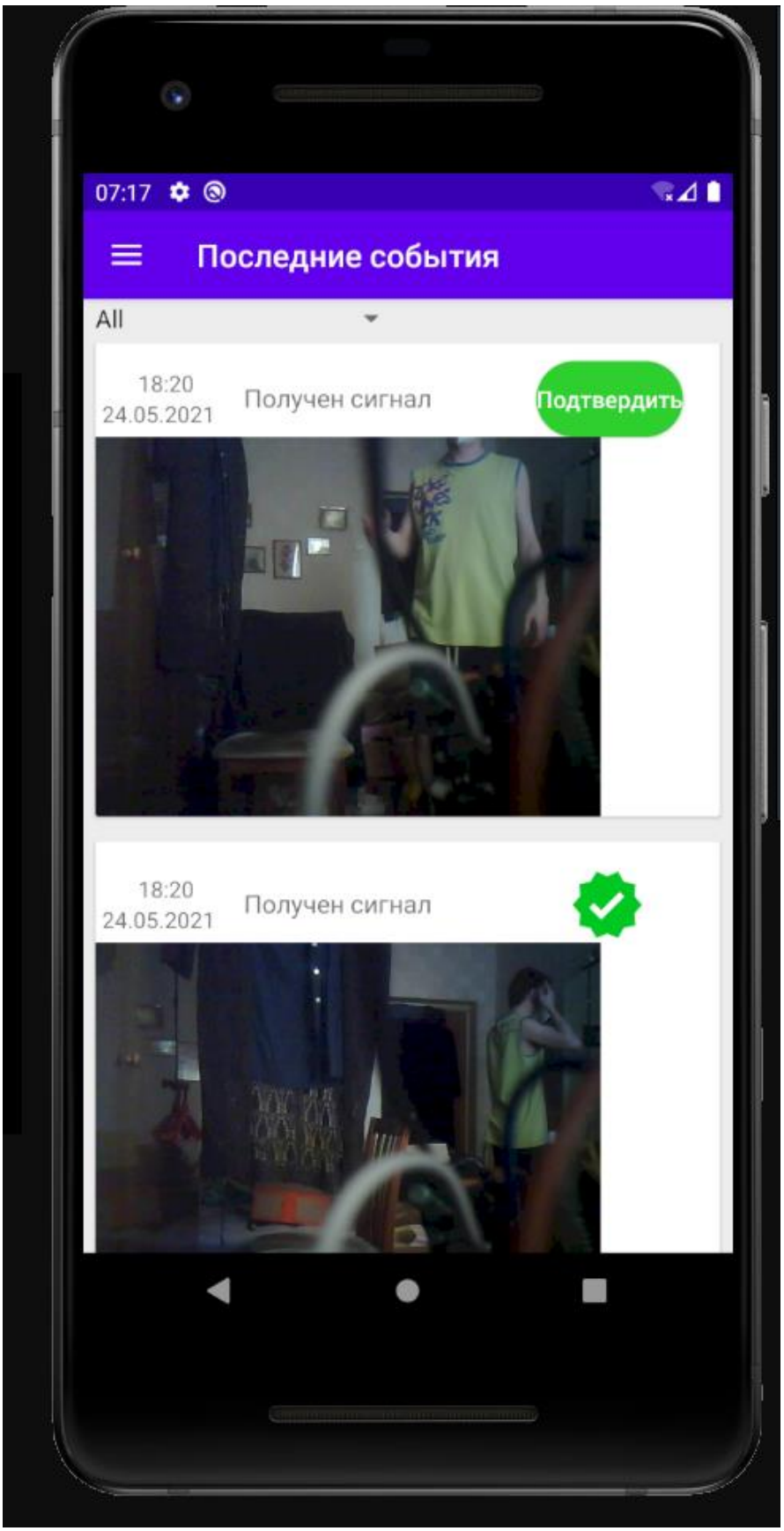


Приложение 7. Интерфейс *Android Studio*

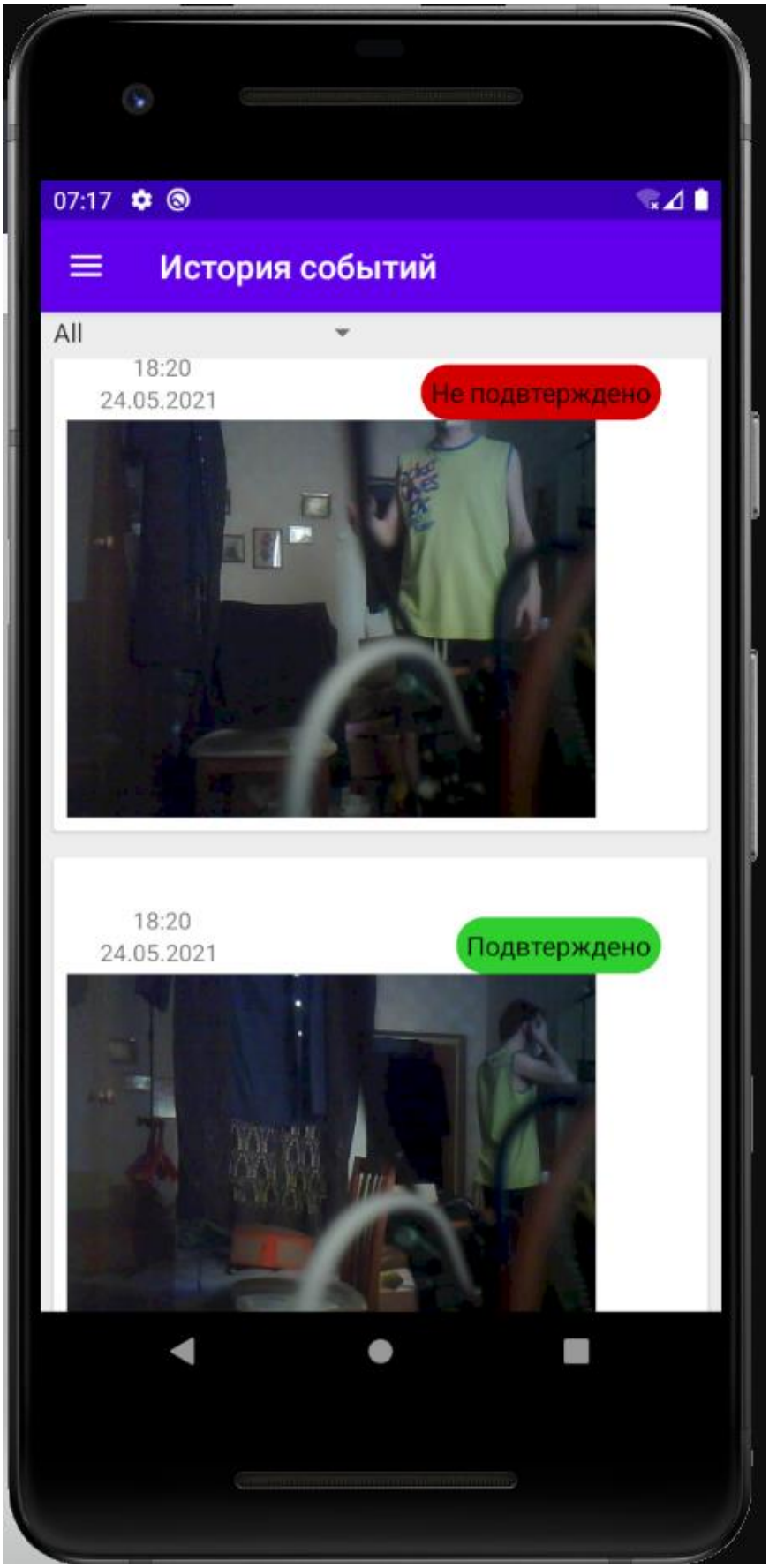


Приложение 8. Создание виртуальной машины

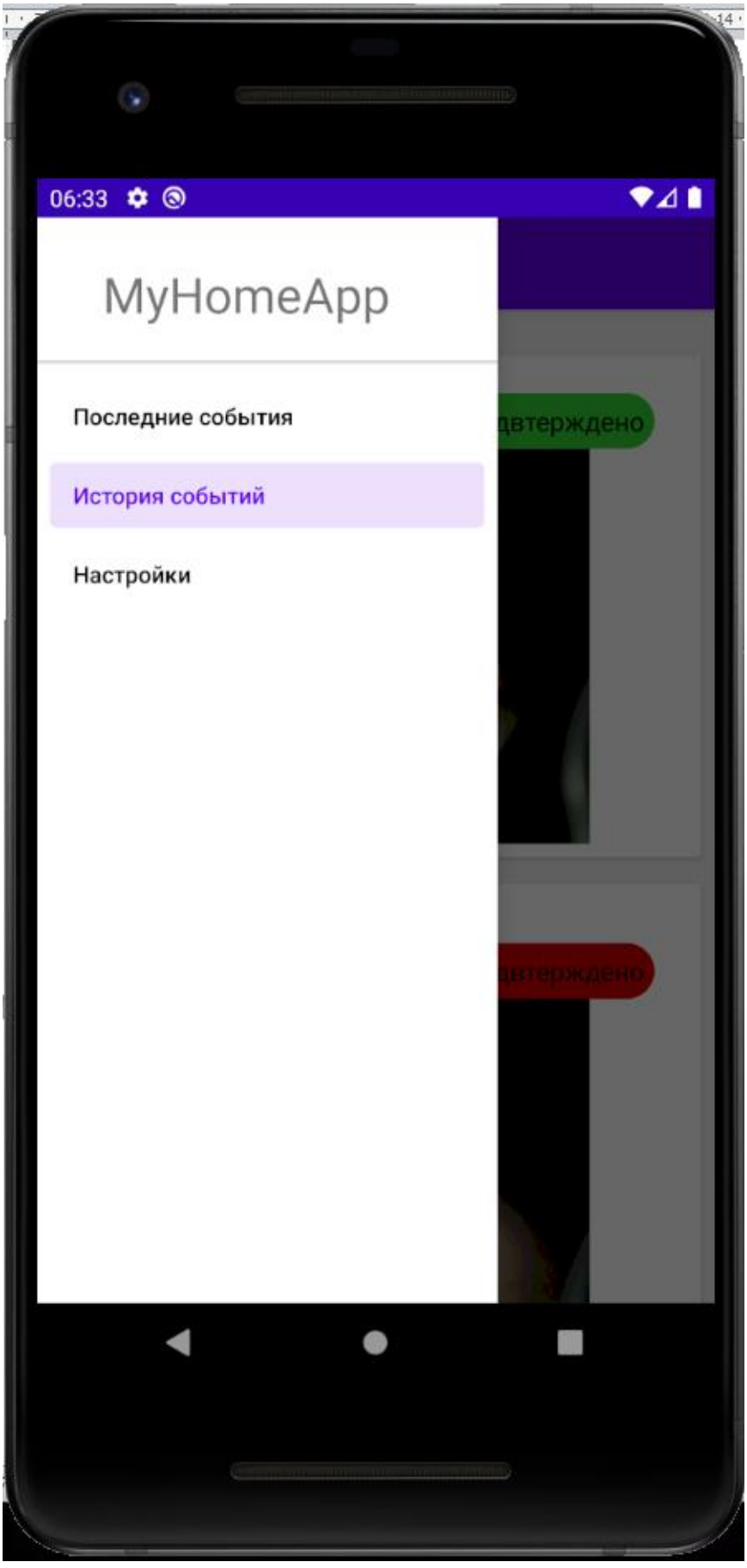
Приложение 9. Внешний вид виртуальной машины



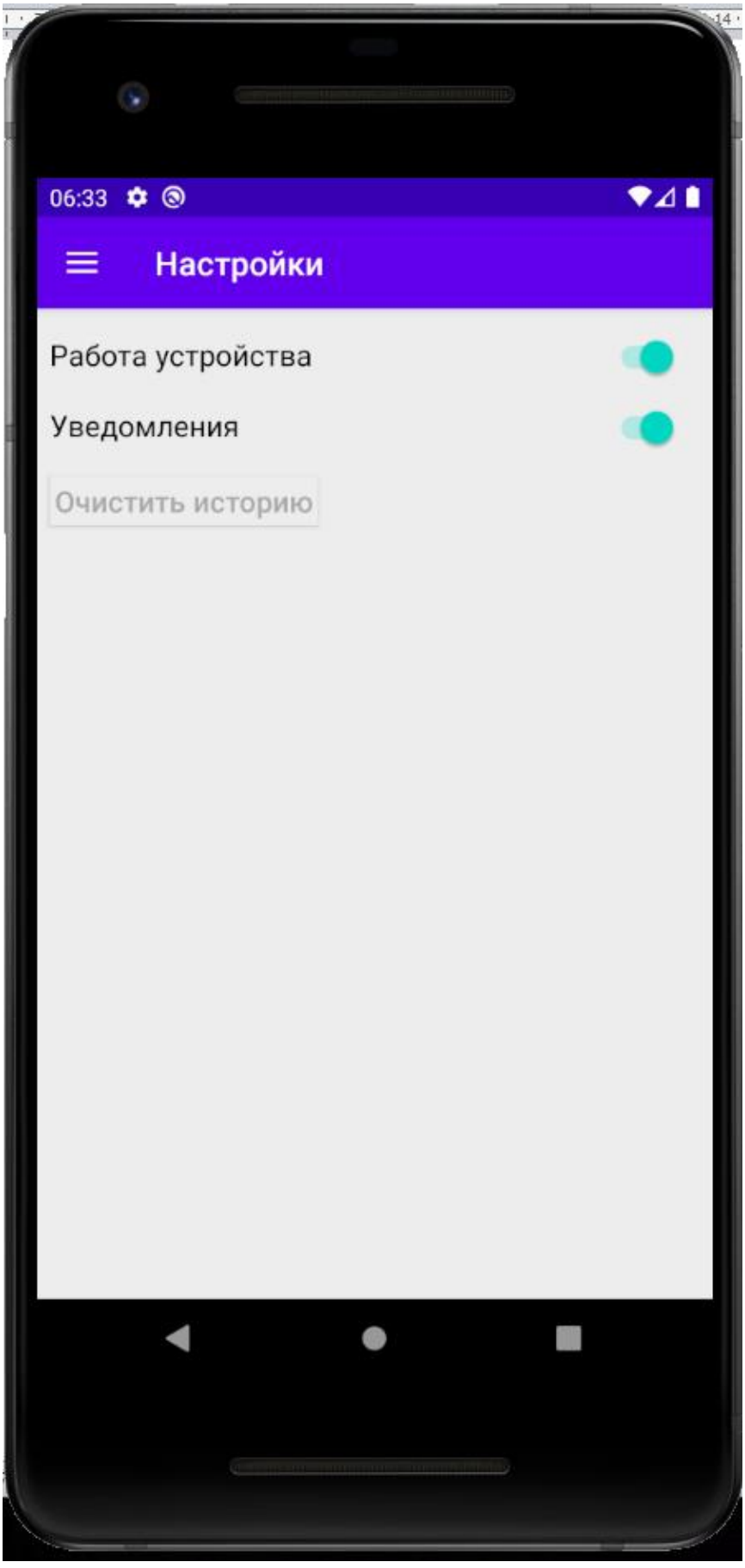
Приложение 10. Получение последних событий в приложении



Приложение 11. История событий



Приложение 12. Навигационное меню



Приложение 13. Окно настроек