



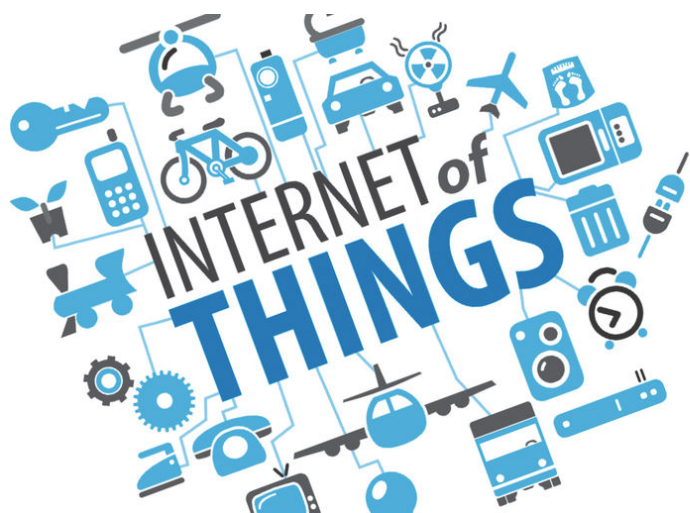
# Выпускная квалификационная работа

Тема: 1. Разработка интерфейса системы расширения функционального взаимодействия компонентов интернета вещей. Суханов К.Д.

Тема: 2. Разработка серверной части системы расширения функционального взаимодействия компонентов интернета вещей. Алексеев В.Ж.

- Актуальность задачи
- Цель работы
- Задачи работы
- Обзор аналогичных комплексов
- Отличительные особенности
- Структура системы
- Взаимодействие компонентов системы
- Структура проектов на C#
- Хост
- Возможность установки на одноплатные ПК
- Интерфейс взаимодействия плагинов и хоста
- Клиентская библиотека
- Генератор интерфейса к устройствам
- Сравнение кода
- IoT-устройства Xiaomi
- Расширение функционала MI Home
- Процесс инъекции кода
- Динамическая загрузка кода
- Классы-обертки
- Пример объявления устройства
- Описание функционала устройства
- Упрощенная схема работы с устройствами Xiaomi
- Интеграция камеры
- Плагин MiHomePlatform
- Результаты работы
- Заключение
- Библиография

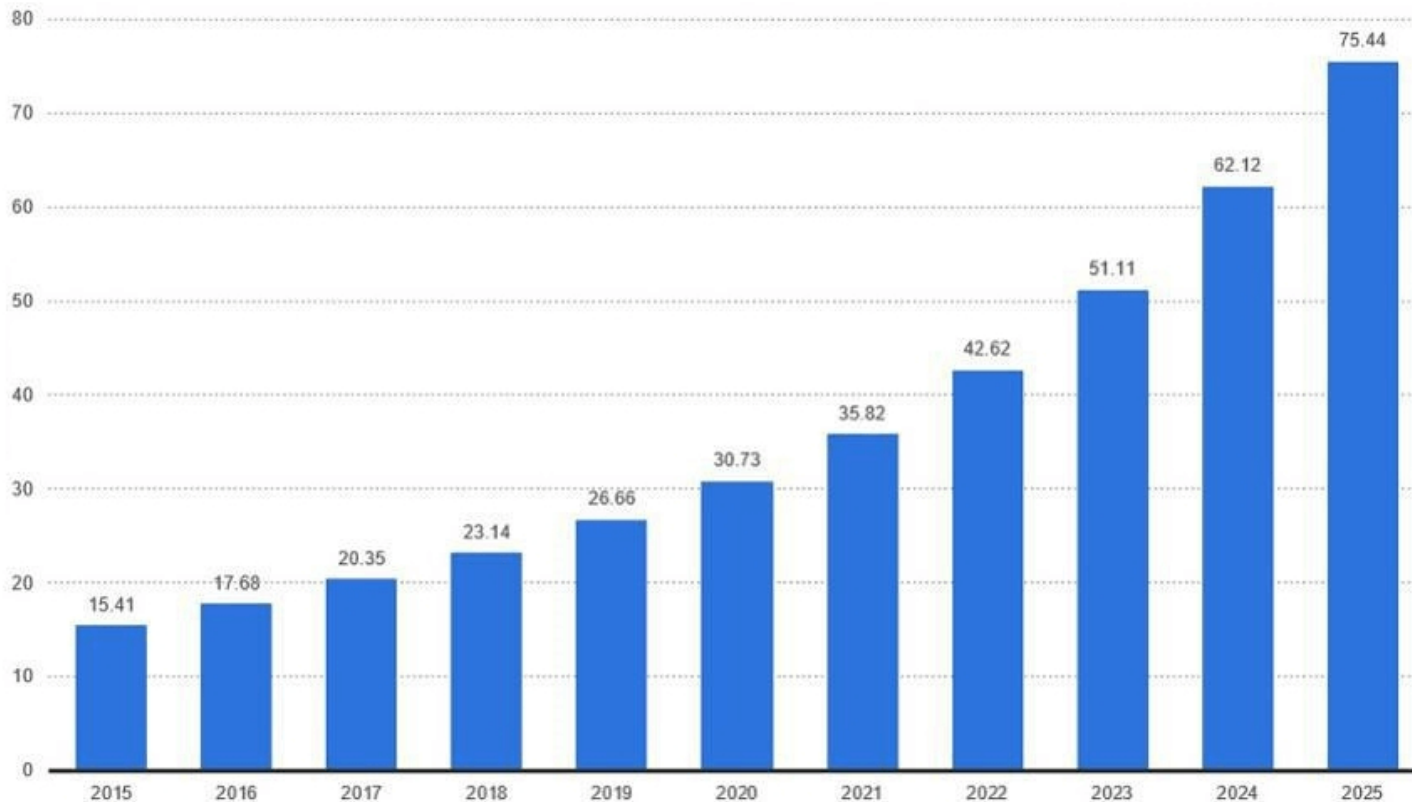
# Актуальность задачи



Интернет вещей — это технологии, встроенные в разные окружающие нас вещи, собирающие и обрабатывающие информацию, помогающие связывать вещи. Это тесная интеграция реального и виртуального миров, где производится взаимодействие между людьми и устройствами.

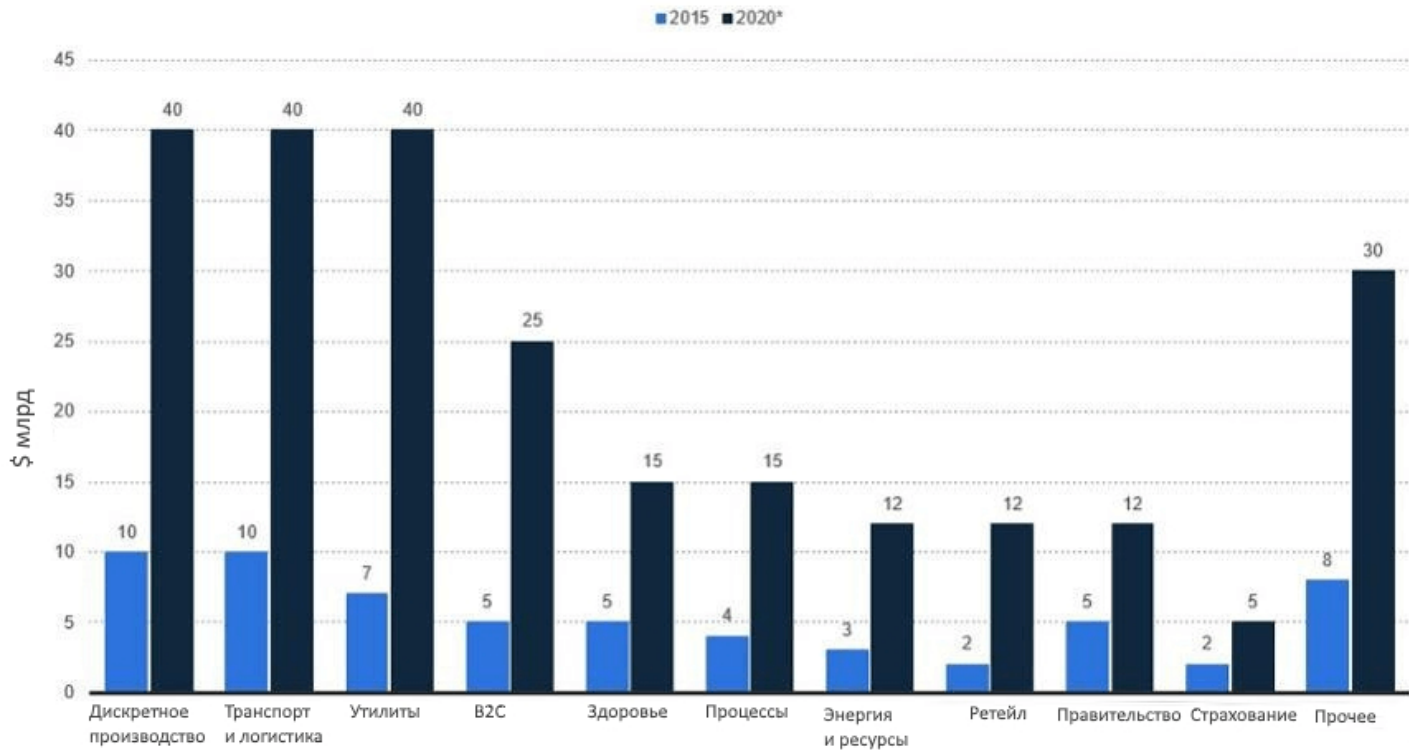
# Актуальность задачи

Рост количества IoT-устройств, подключенных к общей сети (млрд)



# Актуальность задачи

Затраты на IoT в мире по отраслям



По данным ресурса Statista, к 2020 году на IoT будет потрачено 246 млрд. долларов. С помощью интернета вещей компании будут решать вопросы точности, скорости и уровня поставщиков.



# Актуальность задачи



# Цель работы

- Разработка расширяемой системы взаимодействия компонентов интернета вещей, позволяющей создавать сложные сценарии на языке программирования общего назначения.
- Разработка плагина для данной системы, позволяющего работать с IoT-устройствами экосистемы Xiaomi.
- Разработка документации для пользователей и разработчиков расширений.

# Задачи работы

Для осуществления поставленных целей необходимо реализовать следующие задачи:

1. **Реализация программы, предоставляющей единый интерфейс к плагинам – хост.**
2. **Реализации библиотеки на языке C#, позволяющей пользователям системы управлять устройствами и писать скрипты автоматизации**
3. **Реализовать генерацию кода на основании описания устройств для удобства пользователя системы**
4. **Реверс-инжиниринг приложения MI Home**
5. **Реализация плагина для расширяемой системы, путем инъекции кода в приложение MI Home**
6. **Написать документацию к программному комплексу**
7. **Произвести тестирование**

Результат работы — программный комплекс, позволяющий получить унифицированный доступ к IoT-устройствам различных производителей.



# Обзор аналогичных комплексов

Проведя исследование существующих программы продуктов, были выделены следующие аналоги:



Domoticz



openHUB



MajorDoMo



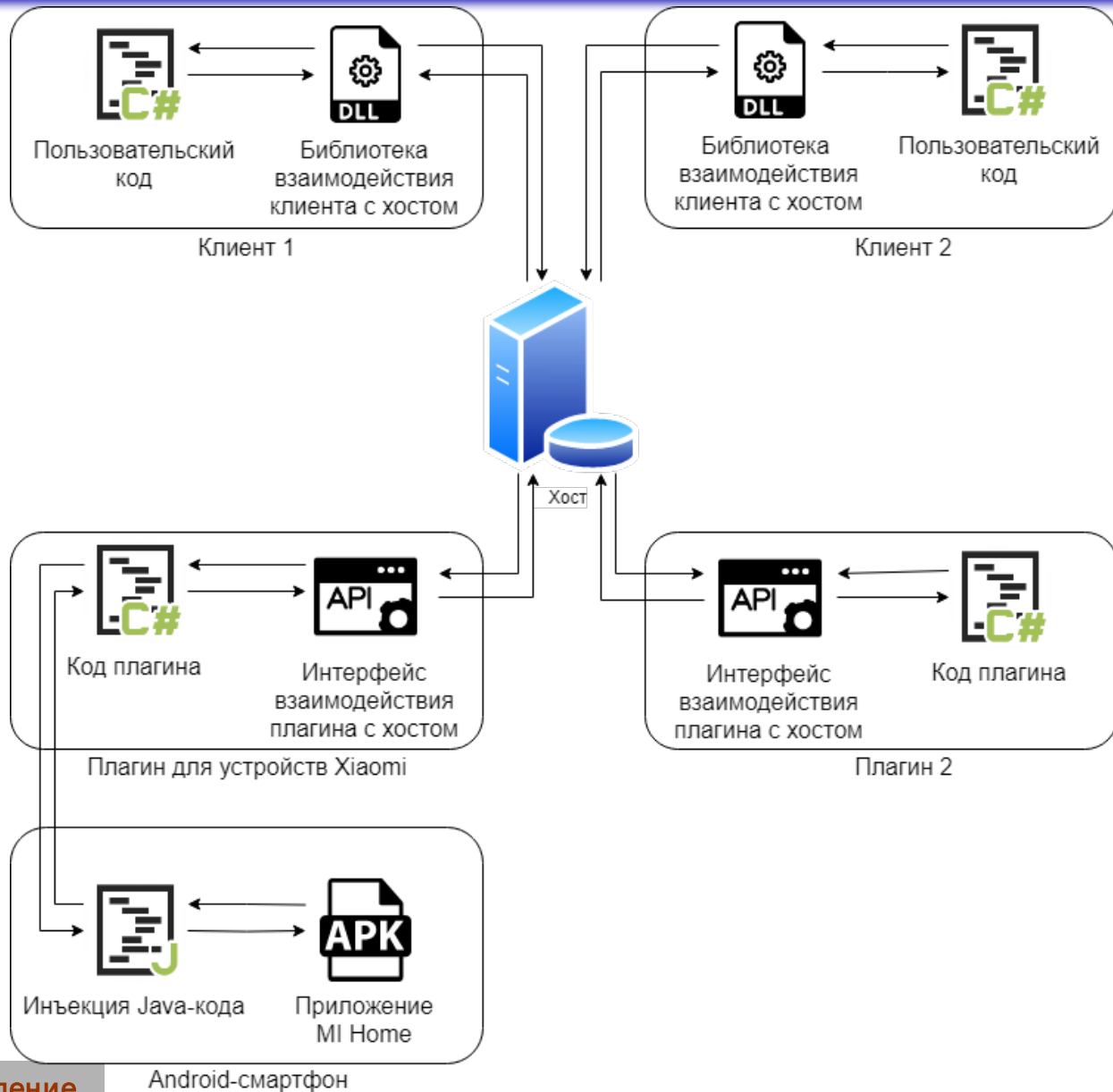
Home Assistant

## Отличительные особенности

Использование языка общего назначения для описания взаимодействия IoT-устройств, что позволяет создавать более сложные системы и поведение.

Отсутствие привязки к единому графическому интерфейсу пользователя, что позволяет реализовывать различные управляющие программы.

# Структура системы



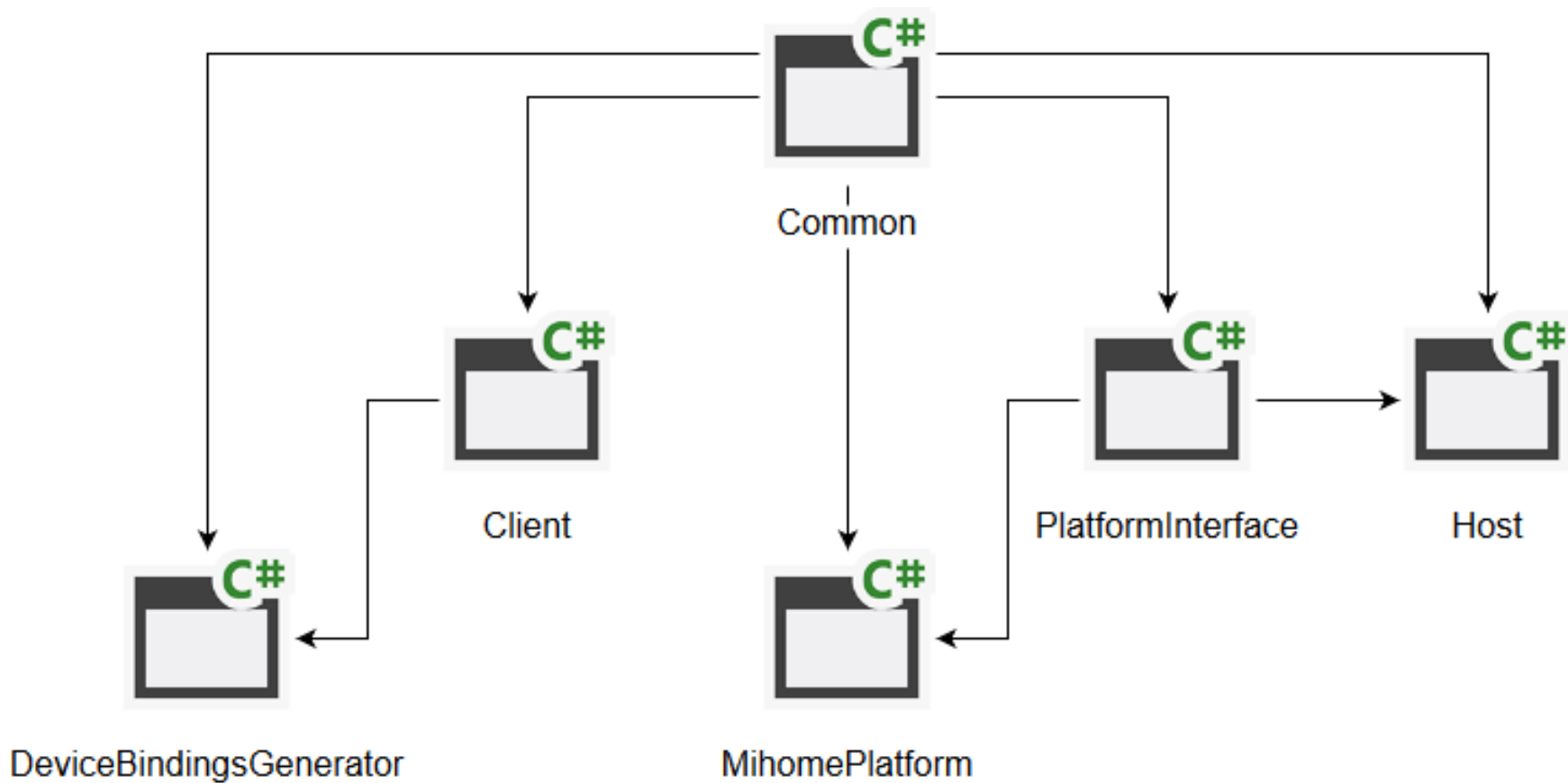
# Взаимодействие компонентов системы

Так как клиенты и хост могут находиться на разных машинах, то взаимодействие между клиентом и хостом должно осуществляться по сети.

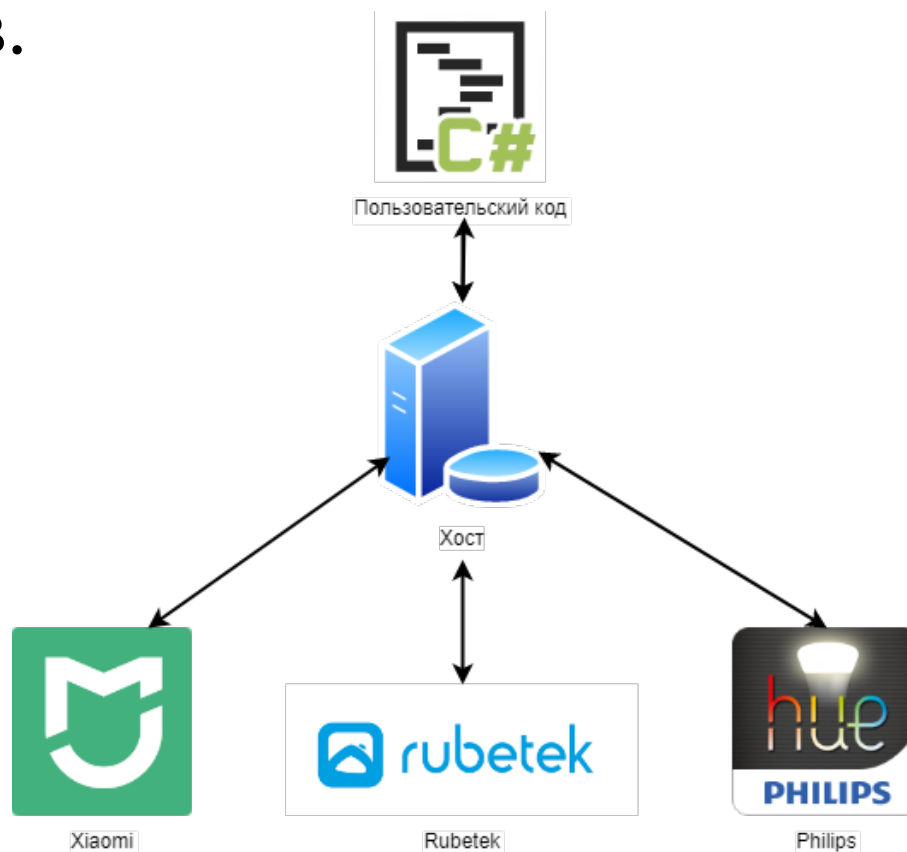


**ØMQ**

# Структура проектов на C#



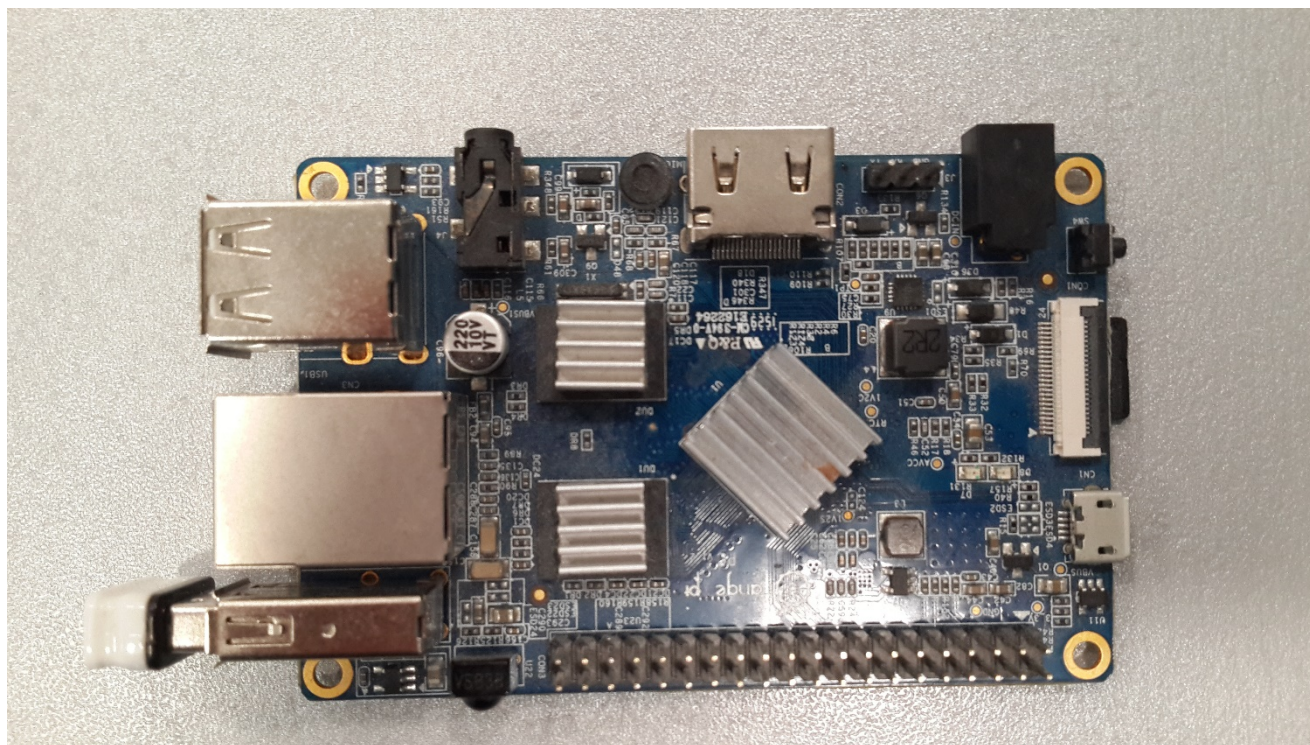
Хост позволяет объединить устройства различных производителей с помощью плагинов.



- Хост написан на языке С#.
- Хост динамически загружает плагины отвечающие за работу конкретных IoT-устройств.
- Каждый плагин представляет из себя сборку (библиотеку), написанную на языке С#.
- Хост принимает сообщения из сети от пользователей и перенаправляет их к IoT-устройству.

# Возможность установки на одноплатные ПК

Данный программный комплекс кроссплатформенный, а также может быть установлен на одноплатные компьютеры.





Функционал реализован в сборке PlatformInterface. Основой являются интерфейсы IPlatformFactory и IPlatform. Данные интерфейсы должны быть реализованы в каждом плагине.

IPlatformFactory отвечает за инициализацию плагина, а IPlatform – за управление плагином и его устройством.

# Клиентская библиотека

Для взаимодействия с системой, была разработана библиотека на языке С#, которую пользователь должен добавить в свой код.

Библиотека позволяет:

- получать список подключенных устройств
- вызывать методы устройств
- просматривать свойства устройств
- подписываться на события устройств

# Генератор интерфейса к устройствам

Для более удобного написания кода пользователем, был создан генератор C#-классов на основе описаний устройств. Это позволяет:

- Получать помощь от IDE в написании кода, работающего с устройствами
- Устранить ошибки, связанные, например, с обращением к несуществующим устройствам или методам

# Сравнение кода

```
Client client = new Client("192.168.1.2", 6667, «id»);  
Device device = await client.GetDevice("Mihome",  
"86796240");
```

```
//Вызов метода без использования генератора интерфейса  
await lamp.InvokeMethod("set_rgb", new object[]{new  
RGBColor(200, 0, 0)});
```

```
//Вызов метода с использованием генератора интерфейса  
YeelinkLightColor1 lamp = new YeelinkLightColor1(device);  
await lamp.SetRgb(new RGBColor(200, 0, 0));
```



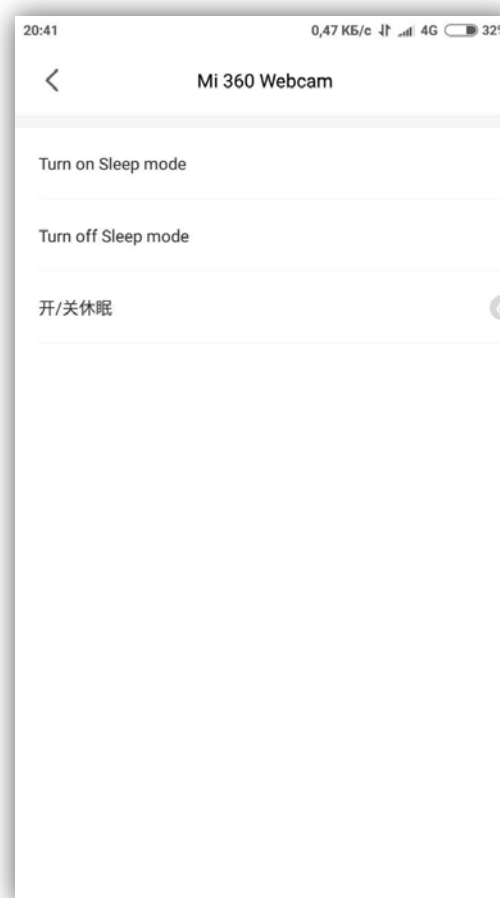
# Выпускная квалификационная работа

Тема: 2. Разработка серверной части системы расширения функционального взаимодействия компонентов интернета вещей. Алексеев В.Ж.

Тема: 1. Разработка интерфейса системы расширения функционального взаимодействия компонентов интернета вещей. Суханов К.Д.

# IoT-устройства Xiaomi

Единственный способ взаимодействия с устройствами Xiaomi – официальное приложение MI Home.



# Расширение функционала Mi Home

- Для возможности интеграции устройств Xiaomi, необходимо расширить функционал приложения Mi Home.
- Это достигается за счет модифицирования официального приложения путем инъекции кода.

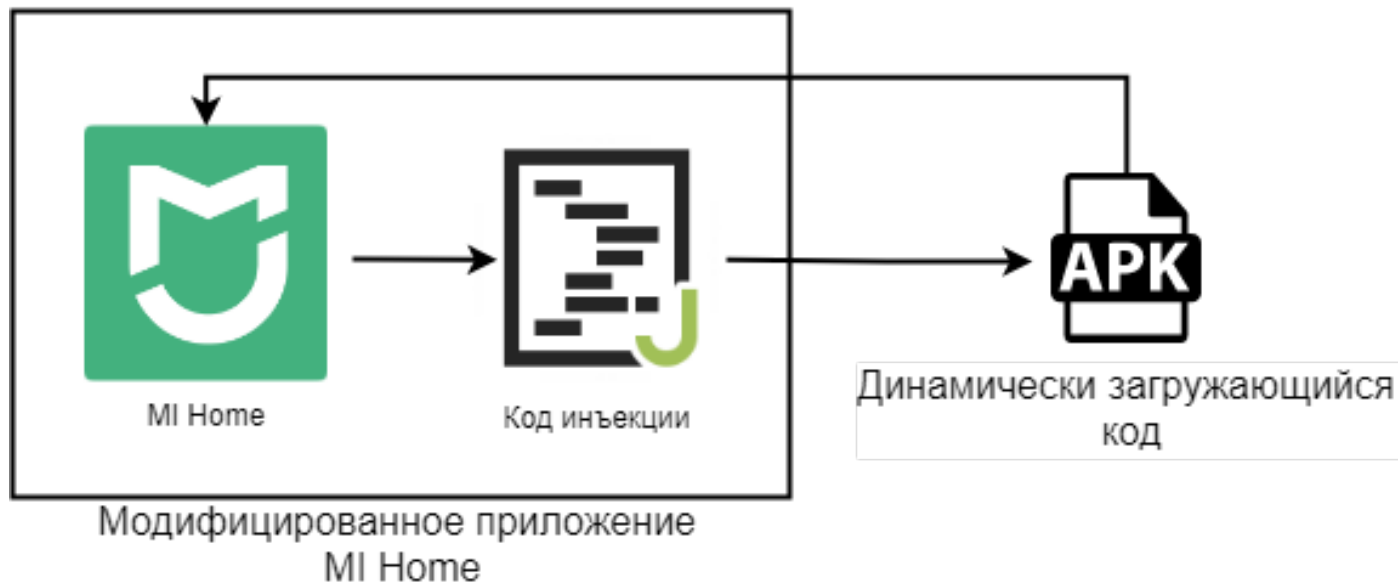
# Процесс инъекции кода

- 1) Дизассемблировать оригинальное приложение. Результат: smali-код — аналог ассемблера для Java-машины.
- 2) Написать необходимую инъекцию кода на Java.
- 3) Скомпилировать инъекцию кода в APK.
- 4) Дизассемблировать APK с инъекцией.
- 5) Перенести smali-код инъекции в smali-код оригинального приложения.
- 6) Собрать smali-код оригинального приложения с инъекцией в APK.



# Динамическая загрузка кода

Внедренная инъекция служит в качестве загрузчика основного кода модификации, находящегося в отдельном APK.



# Динамический вызов кода

- Так как Java-код динамически загружается в Mi Note, то на этапе компиляции нет доступа к классам и методам, объявленным в приложении Mi Note.
- Для работы с неизвестными во время компиляции классами используется Java Reflection API.

# Классы-обертки

- Для удобного доступа к классам приложения Mi Home были созданы классы-обертки, которые вызывают методы оригинальных классов через reflection, при этом предоставляя удобный интерфейс для вызова.

# Пример объявления устройства

```
@DeviceProps({
    @DeviceProp(name = "is_pressed", type = BoolParam.class)
})
@DeviceEvents({
    @DeviceEvent(name = "on_click", notificationText = "Click"),
    @DeviceEvent(name = "on_double_click", notificationText =
"DoubleClick")
})
public class SomeDevice extends Device
{
    @DeviceMethod(name = "set_power")
    public void setPower(MetaParam meta, @ParamName("value")
BoolParam value)
    {
        // реализация метода
    }
}
```

# Описание функционала устройства

- Для каждого устройства в системе создается формальное описание его функционала. Это необходимо для проверки корректности взаимодействия пользовательского кода с устройством и автоматической генерации классов.
- Описание устройства генерируется из аннотаций классов.

# Упрощенная схема работы с устройствами Xiaomi



- Интеграция камеры требует взаимодействия с плагином производителя устройства. Данный плагин загружается динамически, аналогично тому, как загружается наша модификация в приложение Mi Home.

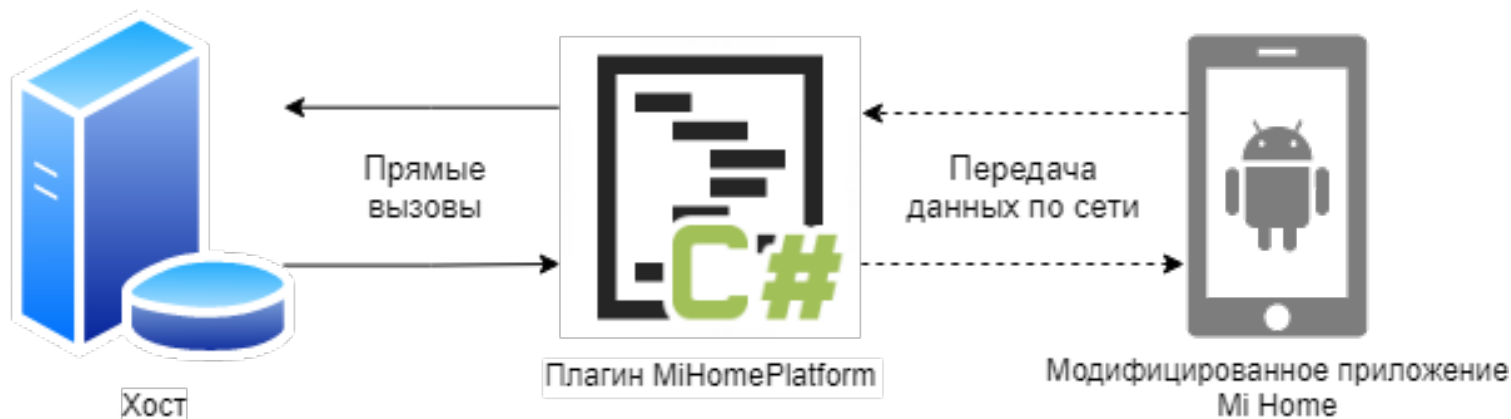
# Интеграция камеры





# Плагин MiHomePlatform

Для интеграции устройств Xiaomi в расширяемую систему был написан плагин MiHomePlatform, который загружается Хостом и общается с модифицированным приложением Mi Home по сети.



# Результаты работы

В результате работы была создана расширяемая система взаимодействия компонентов интернета вещей. Данная система может быть применима:

- 1) Создание умного дома из компонентов различных производителей
- 2) Использование в обучении создания автоматизированных систем
- 3) Коммерческое использование

# Результаты работы



# Заключение

В ходе выполнения дипломной работы:

- проведен анализ существующих программных комплексов для управления IoT-устройствами
- спроектирована структура программного комплекса
- реализована расширяемая система
- произведена модификация приложения для работы с устройствами Xiaomi
- произведено тестирование

# Библиография

Албахари Джозеф, Албахари Бен. С# 7.0. Справочник. Полное описание языка. /— М.: Вильямс, 2018. — 1024 с.

Shovic, John C. Raspberry Pi IoT Projects / — Apress, 2016. — 223 с.

Умный интернет вещей/ Хабрахабр. [Электронный ресурс]. — Режим доступа: <https://habr.com/ru/post/259243/>. — Заглавие с экрана. — (Дата обращения: 23.02.2019).

Интернет вещей: прогнозы по развитию рынка [Электронный ресурс]. — Режим доступа: <https://www.likeni.ru/analytics/internet-veshchey-prognozy-porazvitiyu-rynka/>. — Заглавие с экрана. — (Дата обращения: 05.03.2019).

Хорстманн К. Java. Библиотека профессионала. / Хорстманн К. — М.: Вильямс, 2018. — 864 с.