

**Аннотация**

Данная работа посвящена выбору, установке и настройке уже существующего программного обеспечения с целью превращения одноплатного компьютера в мультимедийное устройство для подключения к телевизору, а также созданию собственного ПО для организации оптимальной схемы управления этим мультимедийным устройством. Основной целью данной работы является расширение возможностей относительно старых телевизоров, чтобы придать им все необходимые функциональные возможности современных Smart TV с помощью дополнительного оборудования – медиапристаки. Ее роль в рамках данной реализации исполняет одноплатный компьютер Cubieboard3. Наиболее важной подзадачей для достижения поставленной цели является организация беспроводного управления с android-смартфона этим одноплатным компьютером по средствам локальной сети WiFi.

Отчёт по МКР содержит:

1. 5 изображений;
2. 19 электронных ресурсов.

**Annotation**

The coursework is dedicated to choosing, installation and configuration of existing software to turn the single-board computer into TV multimedia device and developing own software to organize optimal control scheme of the one. The main purpose of this work is to expand the capabilities of relatively old TVs to give t hem all the necessary functionality of modern Smart TV via additional equipment – Multimedia Set-Top Box. Its role in this implementation is performed by the single-board computer Cubieboard3. The most important subtask for achieving this goal is the organization of wireless control of this single-board computer from the android-smartphone using the local WiFi network.

The report contains:

1. 5 images;
2. 19 internet resources.

**Оглавление**

[Введение 7](#_Toc482402603)

1. [Разработка ПО для удаленного управления 8](#_Toc482402604)
   1. [Анализ существующих решений 8](#_Toc482402605)
   2. [Процесс подключения телефона к медиаприставке 8](#_Toc482402606)
   3. [Внутренний протокол прикладного уровня 9](#_Toc482402607)
   4. [Серверное приложение 11](#_Toc482402608)
   5. [Клиентское приложение 13](#_Toc482402609)
2. [Настройка стороннего ПО 15](#_Toc482402610)
   1. [Выбор дистрибутива для Cubieboard 15](#_Toc482402611)
   2. [Медиаплеер Kodi 16](#_Toc482402612)
   3. [Альтернативы 18](#_Toc482402613)

[Заключение 18](#_Toc482402614)

[Глоссарий 19](#_Toc482402615)

[Приложение 19](#_Toc482402616)

[Использованные источники 20](#_Toc482402617)

# **Введение**

В наше время старые ЭЛТ-телевизоры уже практически вышли из употребления, да и сам телевизор перестал быть только устройством для просмотра эфирного телевидения. Поэтому большинство моделей на рынке оборудованы функциями Smart TV и их темпы продаж только растут [1]. Современные Smart TV работают под управлением таких операционных систем, как Android TV, Firefox OS или OS Tizen, которые обладают довольно широкими возможностями, отлично работают и способны управляться со смартфона благодаря специальному приложению. Однако так было не всегда: модели времен, когда Smart TV только зарождалось, имеют на борту довольно медлительные и неудобные операционные системы. И что же делать владельцам таких моделей, если телевизор как устройство их еще устраивает, но получить все современные прелести Smart TV тоже хочется? Очевидно, что покупка нового телевизора – довольно расточительное и, для многих, дорогое решение, поэтому предлагается альтернатива – TV-приставка. Мультимедийные TV-приставки – это небольшие компьютеры, которые значительно расширяют возможность телевизора. В данной работе будет рассматриваться реализация такой приставки, главной особенностью которой будет возможность управления с устройства Android.

# **Разработка ПО для удаленного управления**

## **Анализ существующих решений**

Существует уже довольно много написанных программ для удаленного управления компьютером, однако в основном они реализуют какой-либо протокол удаленного рабочего стола, например, RDP, и по сути превращают смартфон в тонкий клиент. Это совсем не то, что требуется для управления медиаприставкой. Нам необходима только эмуляция устройств ввода: мыши и клавиатуры. Было рассмотрено два приложения, которые предоставляют такие функции: «Unified Remote» [2] и «WiFi мышь» [3]. Первое распространяется в двух вариантах: бесплатно, но с ограничениями, или за плату, но без ограничений. В бесплатной версии есть возможность только удаленного управления мышью, ввод же с клавиатуры доступен только после оплаты. Второе приложение, хоть и содержит рекламу, никак не ограничивает возможности управления, однако было выявлено, что эмуляция ввода с клавиатуры не работает с кириллическими символами.

На основе этого было принято решение разработки собственного приложения для удаленного управления, которое бы никак не ограничивало пользователя в том числе и при работе с кириллицей.

В качестве подключения используется локальная WiFi-сеть. В перечень используемых технологий вошли:

1. libev для асинхронной работы с сетью.
2. Xlib, libxtest, libxdo для эмуляции устройств ввода.
3. Android SDK для разработки Android приложения.

## **1.2 Процесс подключения телефона к медиаприставке**

В первую очередь, между смартфоном и ТВ-приставкой необходимо установить соединение. Очевидно, что напрямую вводить IP-адрес – это самое ужасное решение, потому что требует от конечного пользователя дополнительных усилий в виде получения этого IP-адреса. В связи с этим была реализована следующая схема: серверное приложение на стороне приставки при запуске создает UDP-сокет, которые слушает сообщения, пришедшие на широковещательный адрес текущей сети. В IP-сетях существует такой механизм, как broadcasting. Его суть заключается в том, что при отправке сообщения на специальный широковещательный адрес оно будет перенаправлено всем устройствам в текущей локальной сети. В нашем случае, при получении контрольного broadcast-сообщения сервер отправляет на адрес, с которого было послано это сообщение, свой текущий адрес в локальной сети, то есть адрес медиаприставки. Соответственно, после этого клиентское приложение на стороне android-устройства получает адрес медиаприставки и может установить устойчивое TCP-соединение (Рис. 1.).

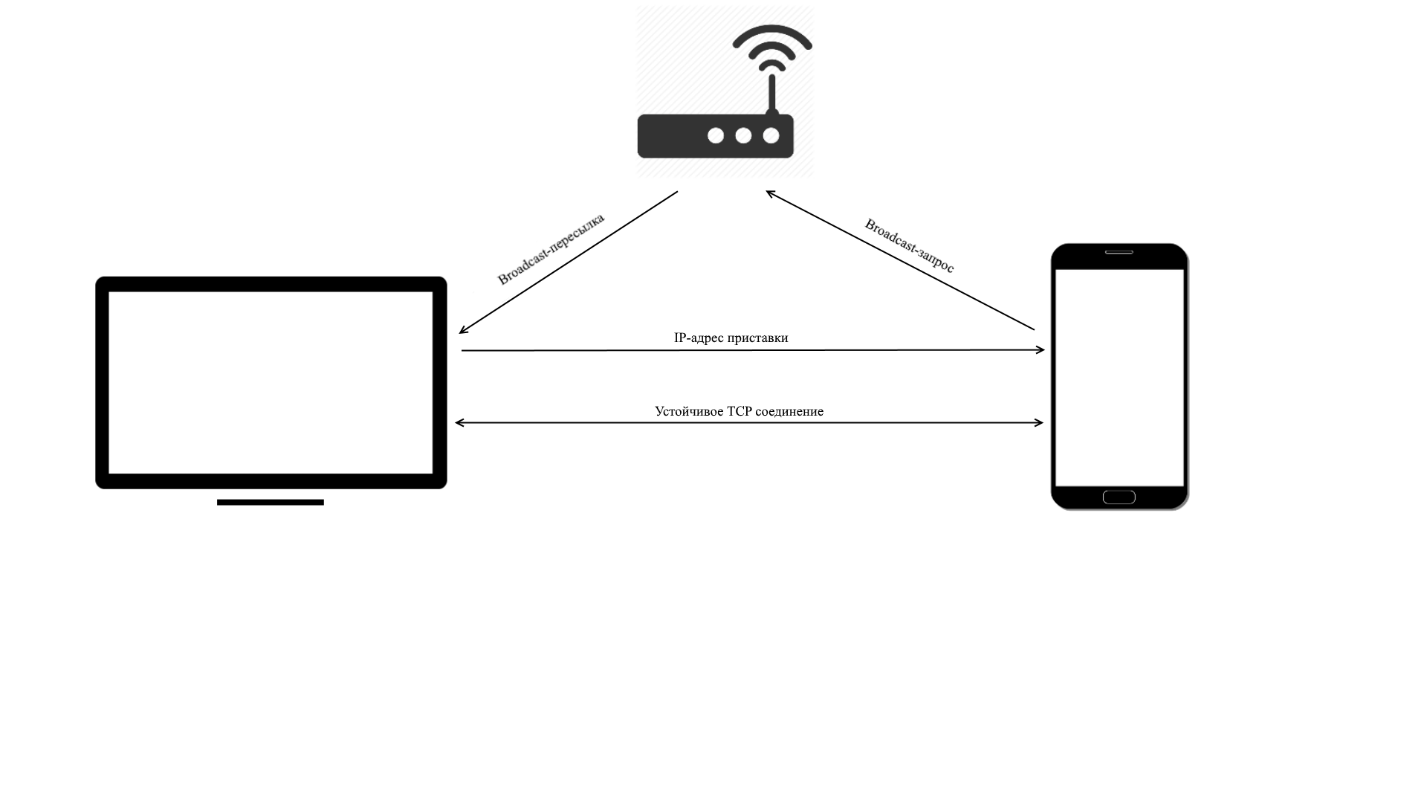


Рис. 1. Схема установления соединения приставки и смартфона.

## **1.3 Внутренний протокол прикладного уровня**

Для организации постоянного соединения телефона и медиаприставки на транспортном уровне стека TCP/IP был выбран протокол TCP. Вариант с UDP в теории мог бы только улучшить скорость работы за счет пропуска некоторых пакетов, однако, как показала тестовая реализация, использование UDP-сокета приводит к нестабильному поведению курсора в некоторых моментах. Скорее всего это вызвано не столько пропуском некоторых пакетов, сколько нарушением их порядка.

Передача информации с обычных устройств ввода, как правило, происходит довольно быстро, даже когда дело касается беспроводных устройств. Однако в нашем случае для взаимодействия используется локальная WiFi-сеть, а не собственный радиоканал, как в большинстве беспроводных устройств, поэтому вопрос быстродействия стал наиболее приоритетным вопросом при написании данного приложения.

Обычно для пересылки подобных сообщений от клиента к серверу используется связка HTTP и JSON или HTTP и XML, однако, при эмуляции устройства ввода объем передаваемых данных достаточно мал, а HTTP и JSON только увеличат размер сообщения. Также стоит принять во внимание, что как генерация и разбор HTML заголовков, так и самого JSON-файла, занимает достаточно много времени, что только навредит быстродействию системы.

Учитывая эти факты, был разработан собственный (Рис. 2.) протокол прикладного уровня стека TCP/IP. Сообщение в рамках такого протокола включает две составляющие: заголовок и тело, причем последнее может отсутствовать. Заголовок представляет собой 1 байт, в младших 2-х битах которого содержится информация о типе сообщения, в старших байтах может располагаться дополнительная служебная информация. Всего имеется 4 типа сообщения:

1. «Команда» – сообщение содержит команду для нажатия какой-либо одной клавиши или их короткой комбинации. Код необходимой кнопки записывается в старшие биты заголовка. Так как 2 бита заголовка уже заняты, возникает ограничение в возможных команды. Общий размер сообщений данного типа всегда равен размеру заголовка и составляет ровно один байт.
2. «Вертикальный скролл» – сообщение содержит команду для эмуляции вертикального пролистывания. В данном случае в старших битах ничего не содержится (и это, стоит заметить, гарантия корректности сообщения), зато имеется тело – 16-разрядное число, хранящее вертикальное смещение для прокрутки.
3. «Движение» – сообщение, которое вызывает эмуляцию движения курсора относительно своего предыдущего положения. Как и в предыдущем случае, в старших разрядах заголовка ничего не содержится, а тело состоит из двух 16-разрядных чисел – смещение курсора по вертикальной и горизонтальной оси относительно текущей позиции. На большинстве android-смартфонах установлены экраны разрешением не больше 1920 на 1080, поэтому максимальное теоретическое смещение, которое может возникнуть на таком экране – 1920 пикселей. Чтобы хранить такое число, как 1920, уже недостаточно 1 байта. Также существует естественное ограничение, что минимальной единицей записи в сокет является 1 байт. Поэтому 16 бит, то есть два байта – это единственное возможное решение в данном случае, хотя стоит заметить, что получить смещение в 1920 пикселей на практике практически невозможно, так как замеры происходят с достаточно высокой частотой.
4. «Текст» – сообщение, которые вызывает эмуляцию ввода произвольной последовательности клавиш. Тело данного типа сообщений состоит из 256 байт и содержит последовательность клавиш для нажатия.

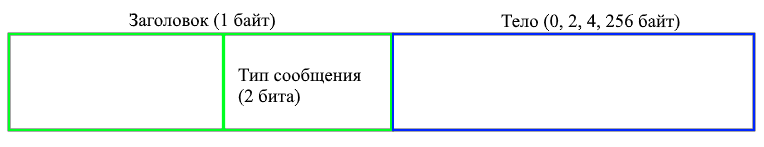


Рис. 2. Структура сообщения по внутреннему протоколу.

## **1.4 Серверное приложение**

С целью повышения быстродействия языком программирования был выбран язык C. Для сетевого взаимодействия использовались сокеты Беркли совместно с библиотекой libev [4] для задания асинхронного режима работы. Если рассматривать простые случаи работы с сетью, то обычно применяются либо многопоточный-синхронный подход, либо асинхронный. Суть первого подхода состоит в том, чтобы в вечном цикле слушать запросы на новые подключения, и, как только такой запрос возникает, создать новый поток исполнения с другим бесконечным циклом, в котором уже и происходит работа с клиентом. Это довольно простой подход, однако создание новых нитей занимает достаточно много времени и ресурсов. Смысл асинхронного подхода состоит в том, чтобы задавать в коде колбеки на определенные события, такие как чтение или запись, а затем передать управление, по сути, ядру операционной системы. Пока приложение не работает ни с одним клиентом, ядро ОС усыпляет его, то есть отсутствует трата процессорного времени, а как только в сокеты, ассоциированные с этим приложением, начинает приходить информация, ядро пробуждает процесс. Именно второй подход используется в данной работе, так как не создает лишней нагрузки, которая могла бы возникнуть при первом подходе.

В самом начале работы программы происходит открытие двух сокетов: один для прослушивания новых подключений, второй для прослушивания broadcast-сообщений. Как только постоянное подключение будет установлено, вотчеры библиотеки libev, отвечающие за асинхронное чтение из этих двух сокетов, будут остановлены, но не удалены. Это значит, что серверное приложение перестанет отвечать на broadcast-сообщения и принимать новые подключения до тех пор, пока вотчеры не будут снова запущены. Это необходимо для того, чтобы предотвратить подключение множества android-устройств к одной приставке.

Так как серверное приложение предполагает фоновую работу, для завершения работы предполагается использование сигналов UNIX. Соответствующий хендлер переопределен таким образом, чтобы перед завершением процесса закрыть все существующие подключения и очистить используемую память.

Все, что связано с сетью, описано в файлах AntouchServer.h и AntouchServer.c (см. приложение).

Для эмуляции устройств ввода использовались библиотеки xlib, xtest и libxdo. Библиотека xlib выступает основной библиотекой для взаимодействия с графической составляющей большинства Linux-based систем. Дело в том, что за все, что связано с графическим интерфейсом пользователя, в GNU/Linux отвечает xorg – свободная реализация системы X Window [5], а библиотека xlib предоставляет API для взаимодействия с ней. Вообще говоря, существуют и альтернативные варианты организации графического пользовательского интерфейса в Linux-based системах. Самые известные из них – это Wayland и проект Canonical – Mir. Ввиду того, что xorg разрабатывался очень давно, в нем накопилось достаточно много проблем, спорных решений и тому подобного. Wayland является переосмыслением xorg в данном контексте. Данная система является основной для рабочего стола Gnome, однако в других проектах особой популярности не нашла. Что касается Mir – это система, разрабатываемая Canonical специально для портирования своей ОС Ubuntu на мобильные устройства. Но совсем недавно стало известно, что Canonical сворачивает ряд своих разработок, таких как Unity, Ubuntu для мобильных устройств, а соответственно и Mir [6]. Поэтому выбор xorg и xlib более чем очевиден. Библиотека xtest является модулем библиотеки xlib, созданной как раз для эмуляции устройств ввода. На основе последней и реализованы все необходимые функции по эмуляции устройств ввода (см. файлы XlibWrapper.h и XlibWrapper.c в приложении). Библиотека libxdo является основой для консольной утилиты xdo, которая позволяет автоматизировать некоторые действия с устройствами ввода. Эта библиотека используется только для ввода произвольной последовательности символов (команда «Текст»). Ввод произвольной последовательности символов можно провести и в рамках xlib, однако, сама система xorg разрабатывалась так давно, что не поддерживает юникод, что очень сильно усложняет эмуляцию ввода символов национальных алфавитов. Поэтому для этих целей используется libxdo, которая учитывает этот недостаток xlib и добавляет необходимые функции.

Рассмотрим подробнее как происходит обработка запроса клиента. Во-первых, как только в сокет приходит сообщение, срабатывает заданный колбек (response\_cb в AntouchServer.c), и в нем с помощью модуля ProtoAtci происходит парсинг команды и передача данных в обертку над xlib (XWrapper.h), а далее управление передается библиотеке xlib, которая и эмулирует устройства ввода.

## **1.5 Клиентское приложение**

Основным языком разработки для android является Java, поэтому он и был выбран для написания клиентского приложения. Целевой платформой был выбран Android 5.0, так как версии ниже 5.0 являются уже морально устаревшими, к тому же, версии с 5.0 по 7.1 охватывают больше половины Android-рынка [7], что достаточно для успешного продвижения приложения.

Разработанное приложение включает в себя три основных модуля:

1. Обработка touch-жестов (см. TouchListener.java в приложении);
2. Обработка нажатий клавиш (см. ButtonsListener.java в приложении);
3. Сетевое взаимодействие и обертка над внутренним протоколом (см. Networking.java и ProtoAtci.java в приложении).

Изначально планировалось использовать привычные жесты для современных тачпадов, например, тап двумя пальцами для вызова контекстного меню (правая клавиша мыши). Однако, позднее был проведен анонимный онлайн опрос (Рис. 3.), который выявил, что большинство людей держат телефон одной рукой. Также можно вспомнить режимы управления одной рукой в последних смартфонах кампаний Apple и Samsung. Иначе говоря, было принято решение адаптировать управление для одной руки. Поэтому контекстное меню вызывается не тапом двух пальцев, а долгим нажатием (long press), что кстати является более привычным для этого жестом в системе android. Для скролла используется 10% справа от области тач панели, как это делается до сих пор в некоторых моделях ноутбуков.



Рис. 3. Результаты анонимного интернет опроса.

Также хочется упомянуть, что при разработке приложения, были учтены советы из гайдлайнов по MaterialDesign [8]. Основное и единственное активити приложения изображено на рисунке 4.

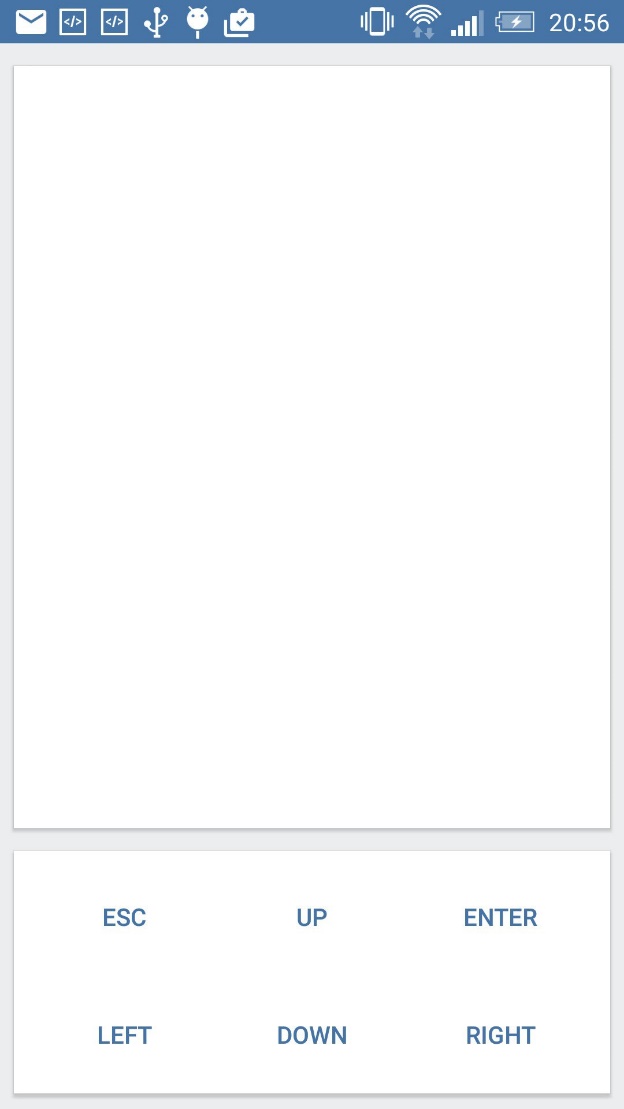


Рис. 4. Основное активити мобильного приложения.

# **Настройка стороннего ПО**

## **2.1 Выбор дистрибутива для Cubieboard**

Данная плата поддерживает только Android и Linux-based операционные системы. Ввиду того, что Android изначально являлся только мобильной операционной системой, он не предполагает изменения своей системной части, поэтому эмуляция устройств ввода на нем проходила бы достаточно проблематично. Поэтому для данной работы использовались Linux-based система. На официальном сайте cubieboard [9] предлагаются различные серверные дистрибутивы и только один дистрибутив с графическим окружением – cubieez – дистрибутив Linux, основанный на Debian GNU/Linux Wheezy. Wheezy – достаточно старая версия вышедшая в 2013-м году [10]. Поэтому в качестве альтернативы также рассматривались сборки от проекта armbian [11], основанные на Ubuntu. В последних релизах в качестве основы используется Ubuntu 16.04, то есть актуальный LTS-выпуск, что одновременно и достаточно современно, и надежно. Согласно политике Canonical, LTS (Long Term Support) [12] версии предполагаются как надежные и проверенные версии операционной системы Ubuntu, в то время как версии между LTS используются для тестирования новых пакетов, что конечно же сказывается на стабильности системы. Выпуски armbian также подразделяются на две ветки: legacy и mainline. Их главное отличие состоит в используемых ядрах. В legacy-версии используется старое ядро Linux 3.4.x, а в mainline – актуальное. Это связано с тем, что ядро версии 3.4.x – это ядро, разработанное производителем SoC данной платы [13], поэтому оно поддерживает такие функции, как перенос ОС с SD-карты в NAND, графический драйвер Mali и аппаратное ускорение. В связи с отсутствием аппаратного ускорения и необходимого графического драйвера, версии armbian с legacy-ядром рекомендуется использовать в том случае, если необходимо графическое окружение, а mainline-ядро, если cubieboard будет использоваться только в качестве какого-либо сервера [14].

## **2.2 Медиаплеер Kodi**

Написание своего графического интерфейса для медиаприставки – сложная и многомерная задача, требующая довольно много времени и профессиональной команды разработчиков, поэтому в рамках данной работы было принято взять готовое решение.

В качестве графической составляющей был выбран открытый медиа плеер kodi (бывший XBMC) [15]. Основными его преимуществом является то, что это наиболее развитый медиаплеер, способный превратить компьютер в полноценную медиаприставку. Также стоит упомянуть об его расширяемости – существует огромная библиотека модулей, которые способны добавить практически любые функции данному плееру.

Разумеется, так как нам необходимо графическое окружение, была взята legacy-версия armbian. Однако, это не помогло. При попытке запуска kodi, последнее падало с ошибкой, связанной с обработкой графики. После компиляции! необходимых библиотек [16] проблема не была решена.

Далее решено было попробовать установить kodi в cubieez. В связи с устаревшей базой пакетов для Debian Wheezy, kodi обозначен там еще как xbmc. Однако опять же, даже после сборки библиотек, kodi отказывался запускаться (Рис. 5).



Рис. 5. Ошибка при запуске xbmc, несмотря на то, что необходимые модули запущены.

На официальной вики странице сообщества linux-sunxi, которое поддерживает программное обеспечение для устройств на основе Allwinner SoC, было обнаружено, что проблема запуска kodi на Cubieboard3 имеет более глобальный характер. Это связано с тем, что производитель чипа – Allwinner – нарушил условия лицензии GPL [17] и скрыл исходный код библиотек, необходимый разработчикам медиаплеера kodi. Поэтому команда разработчиков данного медиаплеера не рекомендует приобретать устройства на базе Allwinner A1x и A2x. Также на той же вики-странице сказано, что идет реверс-инжиниринг библиотек Allwinner и попытки внедрить необходимые компоненты в mainline версию ядра, однако о дальнейшем развитии событий ничего не известно.

## **2.3 Альтернативые медиа ПО и SoC**

Данные проблемы с медиаплеером Kodi это проблема исключительно данной SoC, так как установка Kodi на любую x86 или x86-64 систему не вызывает абсолютно никаких трудностей. Это же справедливо и для других более популярных одноплатных компьютеров, например, таких как raspberry pi. Для последнего существуют даже отдельные дистрибутивы, нацеленные только на использование Kodi [18], поэтому содержат минимальное количество дополнительных программ. Также существует проект RasPlex [19], который предоставляет достойную альтернативу Kodi это Plex Home Theater.

# **Заключение**

В результате работы Д.А. Коптуровым и М.В. Альхимовичем было разработано клиент-серверное приложение для организации удаленного управления Linux-based компьютера с android смартфона. В ходе выполнения МКР Д.А. Коптуровым исследованы особенности работы графической подсистемы X11, разработан собственный протокол прикладного уровня для передачи команд между клиентом и сервером, реализован механизм автоматического установления соединения между устройствами в пределах одной локальной сети. М.В. Альхимовичем выполнен анализ аналогов приложений управления компьютером, а также особенностей дистрибутивов Linux для одноплатных компьютеров и проведена установка одного из них. На аппаратную платформу cubieboard 3 был собран модифицированный графический двайвер для Mali-400 для обеспечения стабильной работы графического стека, что в перспективе позволит реализовать функциональность медиаплеера Kodi, которая сейчас технически невозможна.

# **Глоссарий**

**Вотчер (watcher) –** конкретный объект одной из специальных структур, определенных в библиотеке libev. Данная структура хранит информацию о файловом дескрипторе, событиях для этого дескриптора, при которых необходимо вызвать тот или иной колбек и, непосредственно, указатели на эти колбеки.

**Колбек (callback) –** в данном контексте это функция с определенной сигнатурой, которая вызывается при срабатывании того или иного события.

**Хендлер сигнала (signal handler) –** функция, которая должна быть незамедлительно вызвана при срабатывании определенного сигнала UNIX. Основным отличием от колбека является то, что хендрел сигнала прерывает текущее исполнение кода и незамедлительно исполняется, причем в таком случае состояние программного стека неопределенно.

**Гайдлайн (Guideline) –** набор правил и советов по правильному применению той или иной технологии.

**Активити (Activity)** – одно окно, в котором отрисовываются все элементы, с которыми пользователь может взаимодействовать в данный момент.

**Реверс-инжиниринг (обратная разработка, reverse engineering)** – исследование готового программного или аппаратного продукта с целью установления принципа его внутренней работы.

# **Приложение**

Весь исходный код программы-сервера доступен по ссылке <https://github.com/IsThisLoss/antouch_server>.

Весь исходный код программы-клиента также доступен по ссылке <https://github.com/IsThisLoss/AntouchClient>.

Код обеих программ задокументирован, документация доступна в папках docs в корне каждого проекта.

# **Использованные источники**

1. Dailycomm.ru [Электронный ресурс]. URL: <http://www.dailycomm.ru/m/36066/> (Дата обращения 01.05.2017).
2. Unified Remote. // Google Play [Электронный ресурс] URL: <https://play.google.com/store/apps/details?id=com.Relmtech.Remote> (Дата обращения: 01.05.2017
3. WiFi Mouse. // Google Play [Электронный ресурс] URL: <https://play.google.com/store/apps/details?id=com.necta.wifimousefre> (Дата обращения: 01.05.2017)
4. Libev docs [Электронный ресурс]. URL: <http://pod.tst.eu/http://cvs.schmorp.de/libev/ev.pod> (Дата обращения 01.05.2017).
5. X.Org project [Электронный ресурс]. URL: <https://www.x.org/wiki/> (Дата обращения 01.05.2017).
6. Canonical возвращает Ubuntu на GNOME и прекращает развитие Unity 8 и Ubuntu Phone // opennet.ru [Электронный ресурс]. URL: <https://www.opennet.ru/opennews/art.shtml?num=46326> (Дата обращения 01.05.2017).
7. Статистика версий Android на март 2017 // iguides.ru [Электронный ресурс]. URL: <https://www.iguides.ru/main/gadgets/google/statistika_versiy_android_na_mart_2017/> (Дата обращения 01.05.2017).
8. Material design [Электронный ресурс]. URL: <https://material.io/guidelines/> (Дата обращения 01.05.2017).
9. CubieBoard3/Image [Электронный ресурс]. URL: <http://dl.cubieboard.org/model/CubieBoard3/Image/> (Дата обращения 11.05.2017).
10. Информация о выпуске Debian “wheezy” // debian.org [Электронный ресурс]. URL: <https://www.debian.org/releases/wheezy/> (Дата обращения 12.05.2017).
11. Armbian [Электронный ресурс]. URL: <https://www.armbian.com/> (Дата обращения 12.05.2017).
12. LTS // Ubuntu Wiki [Электронный ресурс]. URL: <https://wiki.ubuntu.com/LTS> (Дата обращения 12.05.2017).
13. Kernel versions // [linux-sunxi.org](file:///C:\Users\Dima\AppData\Local\Temp\linux-sunxi.org) [Электронный ресурс]. URL: <http://linux-sunxi.org/Linux#Kernel_versions> (Дата обращения 12.05.2017).
14. Legacy or mainline kernel? // armbian documentation [Электронный ресурс]. URL: <https://docs.armbian.com/Hardware_Allwinner/#legacy-or-mainline-kernel> (Дата обращения 12.05.2017).
15. Kodi [Электронный ресурс]. URL: <https://kodi.tv/> (Дата обращения 01.05.2017).
16. Mali binary driver // [linux-sunxi.org](file:///C:\Users\Dima\AppData\Local\Temp\linux-sunxi.org) [Электронный ресурс]. URL: <https://linux-sunxi.org/Mali_binary_driver> (Дата обращения 12.05.2017).
17. Kodi // [linux-sunxi.org](file:///C:\Users\Dima\AppData\Local\Temp\linux-sunxi.org) [Электронный ресурс]. URL: <http://linux-sunxi.org/Kodi#History> (Дата обращения 12.05.2017).
18. OpenELEC // Kodi Wiki [Электронный ресурс]. URL: <http://kodi.wiki/view/OpenELEC> (Дата обращения 12.05.2017).
19. A Plex Client for the Raspberry Pi computer // RasPlex [Электронный ресурс]. URL: <http://www.rasplex.com/> (Дата обращения 12.05.2017).