Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Архитектура вычислительных систем

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту
на тему

«Программирование под Raspberry Pi»

Студент гр. 753501
П. Н. Каменко
Руководитель
Ст. преподаватель кафедры информатики
А. В. Леченко

Минск 2019

Содержание

Введение

В настоящее время ведется активная научная деятельность, направленная на уменьшение и удешевление персональных компьютеров при сохранении достаточной производительности и быстродействия для большинства практических задач. Одним из результатов таких действий стало семейство одноплатных компьютеров Raspberry Pi. Изначально созданный для обучения информатике в школах данный компьютер нашел применение в различных областях и сферах деятельности. В связи со своей дешевизной малыми размерами и энергопотреблением он может успешно использоваться для небольших сервисов и инструментов. Целью данной работы является демонстрация возможности применения данного компьютера в роли сервера для телеграмм-бота с простым функционалом, таким как: прогноз погоды для местности пользователя, хранение персональных файлов и их загрузка в мессенджер, загрузка и демонстрация расписания учебной группы пользователя.

1. Анализ предметной области

1.1. Краткие теоретические сведения

Raspberry Pi - семейство одноплатных компьютеров построенных на базе процессоров семейства ARM. Помимо основного ядра, BCM2835 включает в себя графическое ядро с поддержкой OpenGL и Full-HD видео. Также на плате присутствует несколько разъемов USB, Wi-Fi- и Bluetooth-модули. Основными операционными системами на данных компьютерах являются системы, основанные на ядре Linux. Главными преимуществами данных систем являются малые размер и энергопотребление. На компьютере, использованном для работы установлена операционная система Raspbian, основанная на одном из дистрибутивов Linux - Debian.

Рисунок 1.1. Внешний вид Raspberry Pi

Linux — семейство Unix-подобных операционных систем на базе ядра Linux, включающих тот или иной набор утилит и программ проекта GNU, и, возможно, другие компоненты. Как и ядро Linux, системы на его основе как правило создаются и распространяются в соответствии с моделью разработки свободного и открытого программного обеспечения. Linux-системы распространяются в основном бесплатно в виде различных дистрибутивов — в форме, готовой для установки и удобной для сопровождения и обновлений, — и имеющих свой набор системных и прикладных компонентов, как свободных, так, возможно, и собственнических. Появившись как решения вокруг созданного в начале 1990-х годов ядра, уже с начала 2000-х годов системы Linux являются основными для суперкомпьютеров и серверов, расширяется применение их для встраиваемых систем и мобильных устройств, некоторое распространение системы получили и для персональных компьютеров.

За счёт использования свободного программного обеспечения и привлечения волонтёров каждая из систем Linux обладает значительными программными возможностями, трудно реализуемыми в прочих моделях разработки

ARM - Архитектура процессоров, отличающаяся от более привычной x86 сниженным энергопотреблением. Это связано с использованием отличного от x86 набора команд RISC. Преимущество данного набора в изначально небольшом наборе простых команд, которые обрабатываются с минимальными затратами. Однако снижение энергопотребления также означает и снижение производительности - компьютеры, построенные на базе процессоров с архитектурой ARM, могут использоваться только для выполнения простейших задач и программ. В основном процессоры семейства завоевали сегмент массовых мобильных продуктов (сотовые телефоны, карманные компьютеры) и встраиваемых систем средней и высокой производительности (от сетевых маршрутизаторов и точек доступа до телевизоров). Отдельные компании заявляют о разработках эффективных серверов на базе кластеров ARM процессоров, но пока это только экспериментальные проекты с 32-битной архитектурой.

В настоящее время значимыми являются несколько семейств процессоров ARM:

- ARM7 (с тактовой частотой до 60-72 МГц), предназначенные, например, для недорогих мобильных телефонов и встраиваемых решений средней производительности. В настоящее время активно вытесняется новым семейством Cortex.
- ARM9, ARM11 (с частотами до 1 ГГц) для более мощных телефонов, карманных компьютеров и встраиваемых решений высокой производительности.
- Cortex A — новое семейство процессоров на смену ARM9 и ARM11.
- Cortex M — новое семейство процессоров на смену ARM7, также призванное занять новую для ARM нишу встраиваемых решений низкой производительности. В семействе присутствуют четыре значимых ядра:
  - Cortex-M0, Cortex-M0+ (более энергоэффективное) и Cortex-M1 (оптимизировано для применения в ПЛИС) с архитектурой ARMv6-M;
  - Cortex-M3 с архитектурой ARMv7-M;
  - Cortex-M4 (добавлены SIMD-инструкции, опционально FPU) и Cortex-M7 (FPU с поддержкой чисел одинарной и двойной точности) с архитектурой ARMv7E-M;
  - Cortex-M23 и Cortex-M33 с архитектурой ARMv8-M ARMv8-M .

Telegram - кроссплатформенный мессенджер, позволяющий обмениваться сообщениями и медиафайлами многих форматов. При помощи

специального API сторонние разработчики могут создавать «ботов», специальные аккаунты, управляемые программами. Типичные боты отвечают на специальные команды в персональных и групповых чатах, также они могут осуществлять поиск в интернете или выполнять иные задачи, применяются в развлекательных целях или в бизнесе. [3]

1.2. Краткий обзор существующих аналогов

1.2.1. Orange Pi Prime

Отличается от Raspberry Pi 3 наличием большего количества ОЗУ (2 Гбайт) и встроенным в SoC AllWinner H5 видеоускорителем Mali-450 GPU, позволяющим воспроизводить 2K видео. Также присутствует ИК-приемник, поэтому платой можно управлять с пульта ДУ или с некоторых моделей сотовых телефонов со встроенным ИК-светодиодом, например, Redmi Note 7. Из нестандартного оборудования есть также встроенный микрофон и видеоинтерфейс CSI, поддерживающий видеопоток до 1080p на скорости 30 fps.

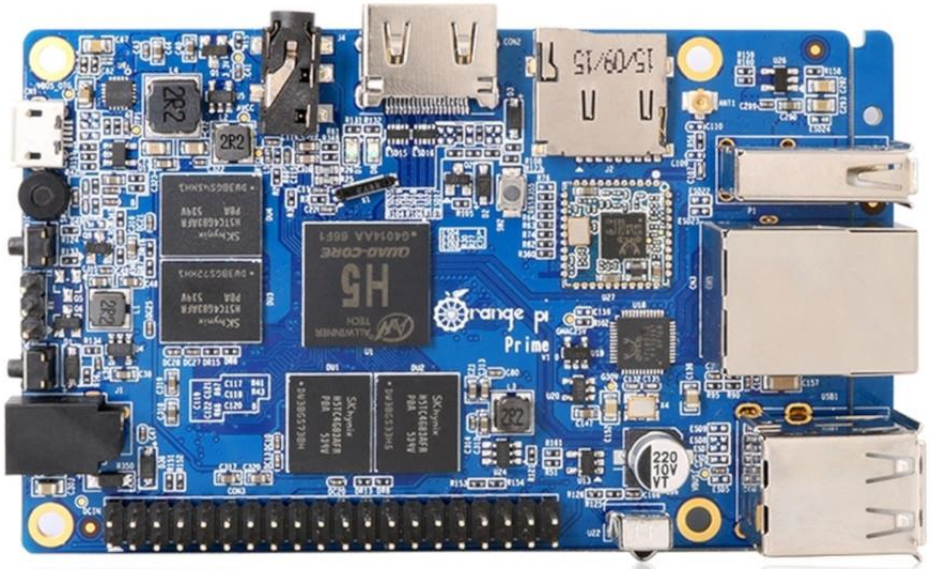

Рисунок 1.2. Внешний вид Orange Pi Prime.

Размер печатной платы - 98 × 60 мм присутствуют разъем для карт памяти (до 32 Гбайт), чип Wi-Fi 802.11 b/g/n, Bluetooth 4.0, четыре USB-входа (три USB 2.0 Host и один USB 2.0 OTG) и сорокаконтактная GPIO-гребенка. Также есть отдельно выведенный UART с TTL уровнями. Из аудио оборудования, кроме упомянутого выше микрофона, есть еще линейный выход и аудиовыход в HDMI. Видеоускоритель поддерживает OpenGL ES 2.0 и OpenVG 1.1. Среди поддерживаемых ОС присутствуют Ubuntu, Debian и Android 5.1.

1.2.2. Banana Pi M3

Banana Pi является разработкой китайской компании SinoVoip. Данное семейство одноплатных компьютеров включает модели Banana Pi M1, M1 Plus, M2 Plus, M2 Ultra, M2 Zero, M3.

Рис 1.3. Внешний вид Banana Pi M3.

Наиболее продвинутая версия - Banana Pi M3 построена на базе восьмиядерного SoC Allwinner A83T (процессоры ARM Cortex-A7, графический процессор PowerVR SGX544MP1), разгоняемого до 1.8 ГГц и работающего в окружении 2 Гбайт ОЗУ и 8 Гбайт флэш-памяти. Кроме гигабитного Ethernet, двух USB, Wi-Fi 802.11 b/g/n, Bluetooth 4.0 и HDMI, на плате присутствует SATA. Так же, как и у Orange Pi Prime, у M3 есть ИК-приемник, видеоинтерфейс CSI, отладочный UART, микрофон, линейный выход и аудиовыход в HDMI. В отличие от Orange, у Banana есть интерфейс дисплея MIPI DSI, объединенный с I2C для сенсорного экрана. Естественно, есть и сорокаконтактная GPIO-гребенка.

1.2.3. Rock64

Одноплатный компьютер Rock64 комплектуется 4 Гбайт ОЗУ, обслуживающими 64-х битный ARM Cortex A53, видеоподсистема достаточна для того, чтобы справится с потоком 4K на частоте 60 fps. Устройство способно питаться от POE. Графическая подсистема ARM Mali 450MP2 соответствует OpenGL ES 2.0, OpenVG1.1. На Rock64 портированы ОС Debian, Cent OS, Fedora и Android 8, нужно отметить, что разработчики и энтузиасты этого компьютера портировали на него большое количество ОС, базирующихся на Linux. У Rock64 обильная, подробная документация.

Данный одноплатный компьютер может рассматриваться как замена Raspberry Pi 3 в проектах с повышенными требованиями к вычислительной мощности.

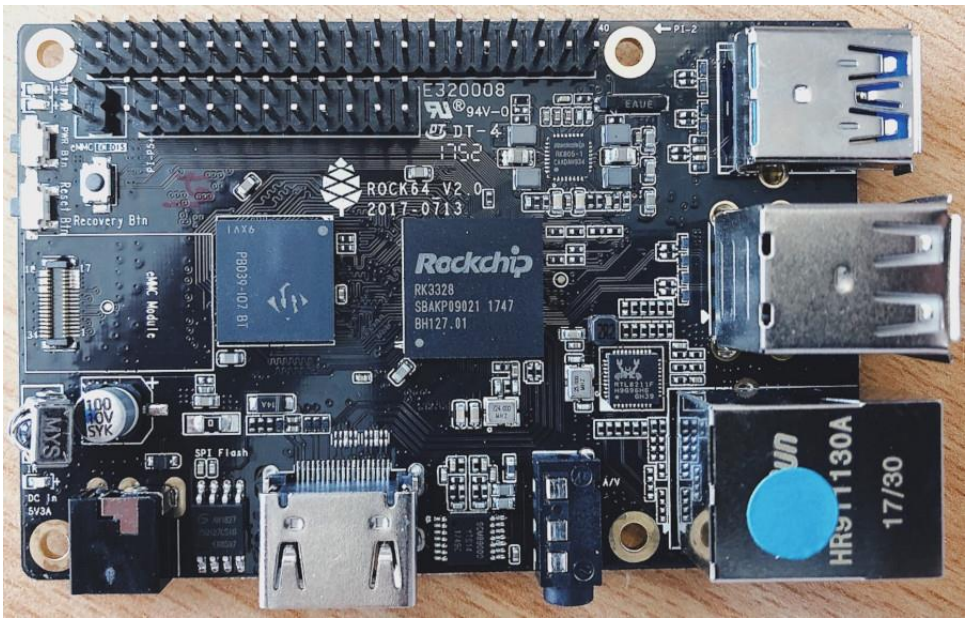

Рисунок 1.4. Внешний вид Rock64.

На плате компьютера кроме прочего присутствуют 64 контакта GPIO, с выведенными на них сигналами Ethernet. Кроме того, есть USB3.0.

1.2.4. Asus Tinker board S

Tinker построен на базе SoC Rockchip RK3288 с счетверенным процессором ARM Cortex-A17 и работает под управлением TinkerOS на базе Debian, можно установить Android. Видеопроцессор Mali-T760 MP4 поддерживает OpenGL ES 3.1, OpenCL 1.1, Renderscript и Direct3D 11.1.

Также существовала младшая модель в семействе Asus Tinker board S - Asus Tinker board без набортной флэш-памяти, но особого успеха onf не имел и сейчас практически исчезла из продажи.

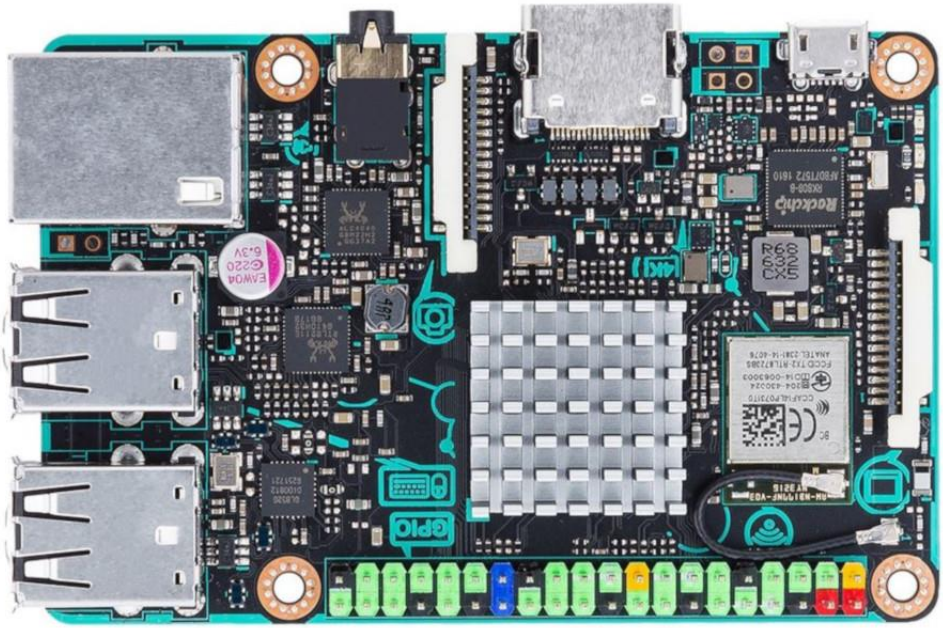

Рисунок 1.5. Внешний вид Asus Tinker board S.

В данном компьютере гребенка GPIO получила цветовую кодировку, для более удобной рабты и уменьшения шанса ошибки при подсоединении. Также модель снабжена небольшим пассивным радиатором, наклеиваемый на корпус процессора.

1.2.5. Libre Computer Renegade

Renegade, или Libre Computer ROC-Rk3328-CC Renegade, конструктивно разработан настолько похожим на Raspberry, насколько это только возможно; например, они являются абсолютно идентичными по размеру и форме.

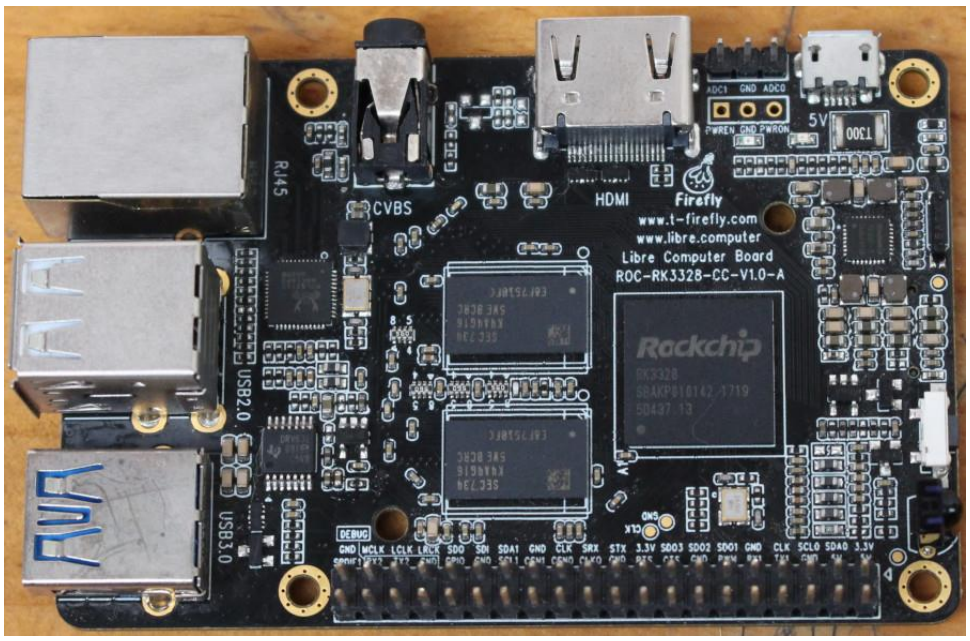

Рисунок 1.6. Внешний вид Libre Computer Renegade.

SoC RK-3328 построена на основе счетверенного 64-х битного процессора ARM Cortex-A53 с рабочей частотой до 1.5 ГГц. SoC такая же, как и в Rock64, так что здесь вы тоже имеете тот же GPU Mali 450MP2 с рабочей частотой 500 МГц. Существуют версии с различным объемом ОЗУ, вы можете выбрать 1 Гбайт DDR4, 2Гбайт или 4Гбайт. Из операционных систем на настоящий момент доступны Ubuntu 18.04, Debian 9, OpenMediaVault 4, Station OS и Android 7.1.

1.2.6. Odroid H2

Odroid H2, построен на базе 64-х битного 4-х ядерного Intel Celeron Gemini Lake J4105, вполне может может быть сопоставим по производительности с решениями на базе ARM. Плата размером 110 × 110 мм, с пассивным охлаждением, графический ускоритель - GPU Intel UHD Graphics 600, присутствует шина PCI-E gen2 и сдвоенный SATA 6 Гбайт/с.

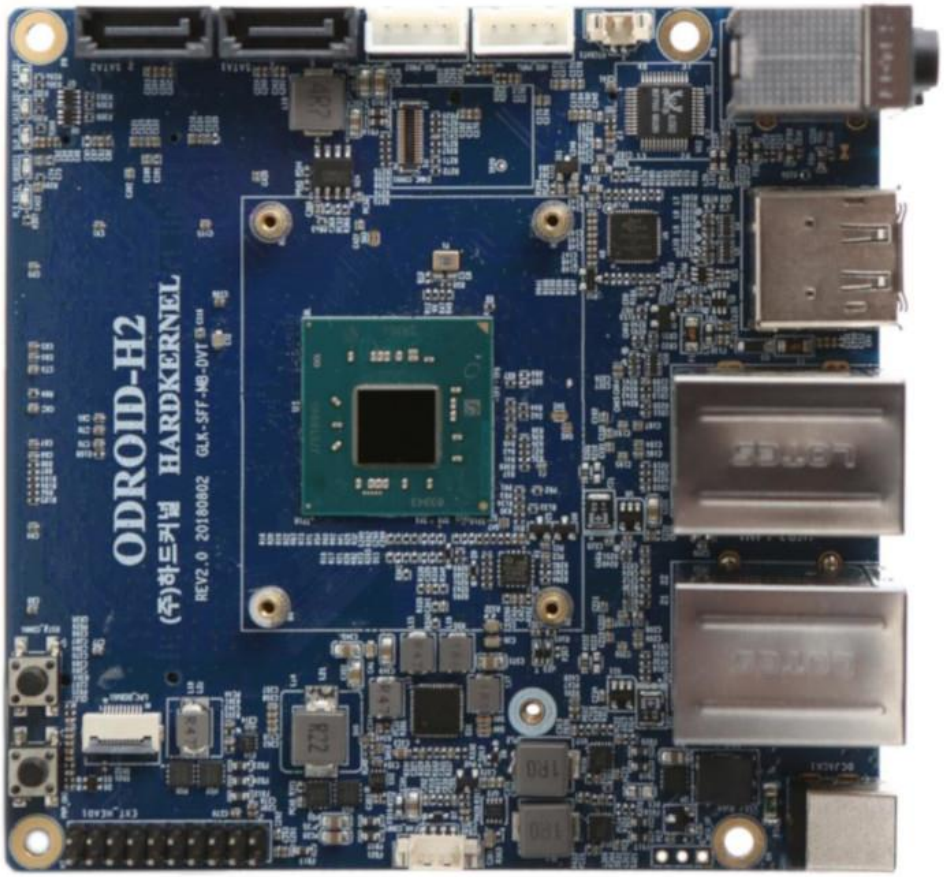

Рисунок 1.7. Внешний вид Odroid H2.

По спецификации Intel максимальный размер ОЗУ равен 8 Гбайт. Плата поддерживает Windows 10 / Linux x64, DirectX 12, OpenGL 4.3, OGL ES 3.0, OpenCL 2.0. [1]

Итоговая информация о данных компьютерах предоставлена в таблице ниже:

Модель	SoC	Процессор	Графика	Ядра	Частота
Raspberry Pi 3B+	Broadcom BCM2837B0	ARM Cortex A53	Broadcom VideoCore IV	4	1.4 ГГц
Raspberry Pi Zero	Broadcom BCM2835	ARM1176JZF -S	Broadcom VideoCore IV	1	1.0 ГГц
Raspberry Pi Zero W	Broadcom BCM2835	ARM1176JZF -S	Broadcom VideoCore IV	1	1.0 ГГц
Banana Pi M3	Allwinner A83T	ARM Cortex-A7	PowerVR 544MP1	8	1.8 ГГц
Banana Pi M2 Zero	Allwinner H2	ARM Cortex-A7	Mali400 MP2	4	1.0 ГГц
Rock64	Rockchip RK3328	ARM Cortex A53	Mali 450MP2	4	1.5 ГГц
Asus Tinker board S	Rockchip RK3288	ARM Cortex-A17	Mali T760 MP4	4	1.8 ГГц
Libre Computer Renegade	Rockchip RK-3328	ARM Cortex-A53	Mali 450MP2	4	1.5 ГГц

Libre Computer Renegade Elite	Rockchip RK3399	ARM Cortex-A72 + Cortex-A53	Mali-T860	6	2.0 ГГц
Odroid H2	-	Intel Celeron J4105	Intel UHD Graphics 600	4	2.3 ГГц
Arduino Mega	-	ATmega2560	-	1	16 МГц

1.3. Технологии и средства разработки

В качестве языка реализации выбран Java. При написании исходного кода и тестировании использовалась среда IntelliJ IDEA 2019 Community Edition.

IntelliJ IDEA — интегрированная среда разработки программного обеспечения для многих языков программирования, в частности Java, JavaScript, Python, разработанная компанией JetBrains.

Первая версия появилась в январе 2001 года и быстро приобрела популярность как первая среда для Java с широким набором интегрированных инструментов для рефакторинга, которые позволяли программистам быстро реорганизовывать исходные тексты программ. Дизайн среды ориентирован на продуктивность работы программистов, позволяя сконцентрироваться на функциональных задачах, в то время как IntelliJ IDEA берёт на себя выполнение рутинных операций.

Начиная с шестой версии продукта IntelliJ IDEA предоставляет интегрированный инструментарий для разработки графического пользовательского интерфейса. Среди прочих возможностей, среда хорошо совместима со многими популярными свободными инструментами разработчиков, такими как CVS, Subversion, Apache Ant, Maven и JUnit. В феврале 2007 года разработчики IntelliJ анонсировали раннюю версию плагина для поддержки программирования на языке Ruby.

Начиная с версии 9.0, среда доступна в двух редакциях: Community Edition и Ultimate Edition. Community Edition является полностью свободной версией, доступной под лицензией Apache 2.0, в ней реализована полная поддержка Java SE, Kotlin, Groovy, Scala, а также интеграция с наиболее популярными системами управления версиями. В редакции Ultimate Edition,

доступной под коммерческой лицензией, реализована поддержка Java EE, UML-диаграмм, подсчёт покрытия кода, а также поддержка других систем управления версиями, языков и фреймворков.

В данной работе исполняемая программа для «бота» написана на языке Java, который является строго типизированный объектно-ориентированный языком программирования, транслируемым в специальный байт-код. Преимуществами данного языка являются:

- Объектно-ориентированность
- Высокоуровневость и простой синтаксис
- Платформонезависимоть
- Автоматическое управление памятью
- Высокая производительность Just-In-Time компилятора

Также для упрощения работы со сторонними сервисами был использован сторонний фреймворк Spring Framework. Данный фреймворк реализует модель разработки, основанную на лучших стандартах индустрии, и делает её доступной во многих областях Java. Таким образом к достоинствам Spring можно отнести:

1. Относительная легкость в изучении и применении фреймворка в разработке и поддержке приложения.
2. Внедрение зависимостей (DI) и инверсия управления (IoC) позволяют писать независимые друг от друга компоненты, что дает преимущества в командной разработке, переносимости модулей и т.д.
3. Spring IoC контейнер управляет жизненным циклом Spring Bean и настраивается наподобие JNDI lookup (поиска).
4. Проект Spring содержит в себе множество подпроектов, которые затрагивают важные части создания софта, такие как веб сервисы, веб программирование, работа с базами данных, загрузка файлов, обработка ошибок и многое другое. Всё это настраивается в едином формате и упрощает поддержку приложения.

Для сборки Java-проекта использовалась популярная система автоматической сборки Gradle. Ее преимуществами перед другимим системами автоматической сборки является то, что он не использует XML. Вместо этого он использует разработанный DSL, основанный на Groovy (JVM-Based язык программирования). В результате билд скрипты стали короче и чище, чем написанные для Ant и Maven. Количество кода конфигурации уменьшилось, поскольку Gradle был разработан с целью разрешения специфической проблемы: провести программное обеспечение через его жизненный цикл, от компиляции через статический анализ и тестирование до упаковки и развертывания. Gradle использует Apache Ivy для управления jar зависимостями (загрузка из удаленного репозитория).

2. Описание структуры программы

Структура программы представляет из себя базовое Java приложение - связанный набор классов, описывающих сущности и Gradle-скрипт для сборки. Внутри приложения обмен данными происходит с помощью сообщений.

2.1. Описание классов

- *LightbulbchanApplication* - класс, инициализирующий работу фреймворка Spring Framework (серверная часть, база данных и т.д.), а также всех остальных классов приложения.
- *Core* - главный класс приложения. Он отвечает за основную логику приложения, принимает сообщения от остальных сервисов и перенаправляет их в классы TelegramGateaway, Weather, ScheduleManager и Files.
- *Weather* - сервис, ответственный за получение прогноза погоды из стороннего ресурса. Данный класс при обращении к нему делает HTTP запросы к сервису “Яндекс Погода” для получения актуальной температуры и скорости ветра в локации пользователя.
- *ScheduleManager* - сервис, ответственный за получение расписания учебной группы, введенной пользователем. Он делает HTTP запросы к ресурсу БГУИР iis.bsuir.by для получения подробного расписания введенной пользователем группы.
- *Files* - сервис, ответственный за работу с файлами. При вызове он производит поиск файлов на вставленном в Raspberry Pi носителе, с учетом расширения, введенного пользователем (.txt, .pdf, .mp3).
- *TelegramGateway* - сервис, необходимый для получения программой сообщений от пользователя. Он работает с API Телеграма и при получении сообщения отправляет его в виде запроса в класс Core для дальнейшей маршрутизации.
- *BotSettings* - класс, содержащий конфигурацию “бота” для TelegramGateway. Осуществляет работу с токеном Телеграма и именем бота в мессенджере.
- *LBCMessage* - класс, описывающий структуру сообщений, циркулирующих внутри программы. Каждое сообщение имеет тип, который помогает определить откуда пришло сообщение, собственно само сообщение в формате строки, день создания сообщения, идентификатор чата и другую служебную информацию.
- *LBCMessageType* - служебный класс, представляет из себя перечисление (Enum) типов LBCMessage. В программе существует 4 ттипа сообщений:
    - WEATHER – сообщения, необходимые для обработки погоды
    - FILES – сообщения, необходимые для работы с файлами пользователя

        - SCHEDULE – сообщения, необходимые для работы с расписанием занятий
        - CONFIG – служебные сообщения
- *MainMenuKeyboard* - класс, описывающий клавиши клавиатуры, ответсвенной за основной функционал программы. Описывает три клавиши:
    - WEATHER_BUTTON
    - FILES_BUTTON
    - SCHEDULE_BUTTON
- *ScheduleKeyboard* - класс, описывающий клавиши клавиатуры для запроса пользователя на получение расписания. Содержит клавиши для каждого дня недели, ближайшего дня, и смены текущей группы.
- *FilesKeyboard* - класс, описывающий клавиши клавиатуры для выбора формата загружаемых файлов. Содержит описание клавиш для каждого расширения файлов.
- *DaySchedule* – класс, описывающий расписание занятий одного определенного дня.
- *FullSchedule* – класс, описывающий общее расписание занятий пользователя. Необходим для обработки ответа сервиса IIS BSUIR и выбора конкретного дня, интересующего пользователя.
- *SubjectSchedule* – вместе с предыдущими двумя классами, образует систему классов, являющихся DAO для расписания занятий, полученных с IIS.
- *UserData* - класс служебной информации о пользователе. Используется для хранения его учебной группы, местоположения и т.п.
- *WeatherResponse* - класс являющийся DAO для ответа ресурса “Яндекс Погода”. Обрабатывает ответ данного сервиса и выбирает необходимую пользователю информацию.
- *Response* - внутренний класс для определения успешности выполнения запроса
- *Employee* - класс, описывающий структуру информации о преподавателе, который ведет занятие (Имя, фамилия, отчество, ученая степенб).

2.2. Иные файлы и сторонние ресурсы

Для получения информации о погоде и расписании, данной программе необходимо отправлять запросы к определенным сторонним ресурсам. Взаимодействия с ними реализованы с помощью предоставляемых данными ресурсами API. Осуществляется взаимодействие со следующими сервисами:

- **Яндекс Погода**. Сервис для просмотра погоды в различных городах России, Беларуси, ближнего и дальнего зарубежья. Разрабатывется компанией “Яндекс”. Предоставляет текущие климатические условия, а также прогноз на ближайшие 9 дней. Для некоторых городов

предоставляется информация о метеостанциях и времени замеров. Для работы с данным сервисом необходимо получить так называемый “токен” – криптографический ключ для авторизации. Непосредственно погода получается от сервиса с помощью отправки GET-запроса, содержащего местоположение и токен. После этого из ответа сервиса выбирается интересующая нас информация.

- **IIS BSUIR**. IIS BSUIR является интегрированной информационной системой университета БГУИР. Данный сервис предоставляет возможность работы со всей, необходимой студентам информацией: расписания, оценки, экзамены, электронные журналы и т.д.
  Для работы с данным сервисом токены не используются. Достаточно просто отправить URL-запрос с номером группы, и выделить расписание из ответа сервиса.
- **Telegram**. Мессенджер, внутри которого и происходит работа бота. Изначально, для создания бота и работы с API необходимо получить токен авторизации с помощью общедоступного бота BotFather. Дальнейшее взаимодействие происходит с помощью HTTP-запросов в стиле REST. Однако для упрощения написания кода была использована сторонняя библиотека, преобразующая вызываемые функции в HTTP-запросы, а также производящую постоянные запросы на обновление сообщений.

3. Разбор работы программы

Для начала работы с приложением необходимо в мессенджере Telegram найти пользователя с никнеймом lightbulbchan и отправить ему сообщение с базовой командой начала диалога /start. (Рис. 2.1.)

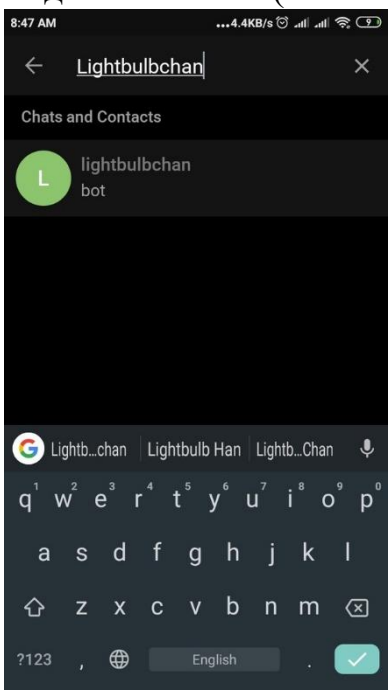

Рисунок 2.1.Пользователь бота в поиске Telegram.

После этого перед пользователем вместо стандартной клавиатуры возникает главное меню выбора действия, состоящее из 3 пунктов: “Погода”, “Загрузка файлов”, “Расписание”. (Рис. 2.2.)

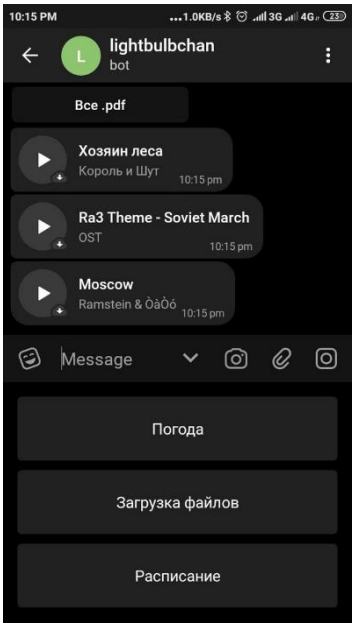

Рисунок 2.2.Основное меню.

Рассмотрим работу первого пункта - “Расписание”. При первом нажатии данной клавиши, бот предлагает пользователю ввести номер своей группы. После ввода номера группы предлагается выбрать день текущей недели, для которого необходимо отобразить расписание. (Рис. 2.3.)

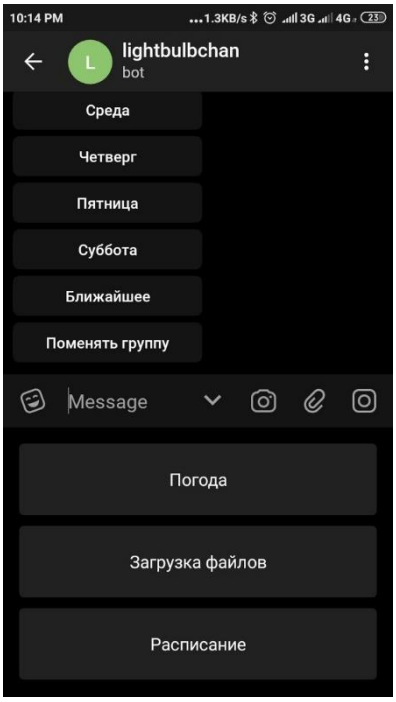

Рисунок 2.3.Выбор дня недели.

Соответственно после нажатия, отображается расписание занятий на данный день недели, включающее в себя название, тип, время и преподавателя каждого занятия. Или констатация того, что в данный день занятий нет. (Рис. 2.4.)

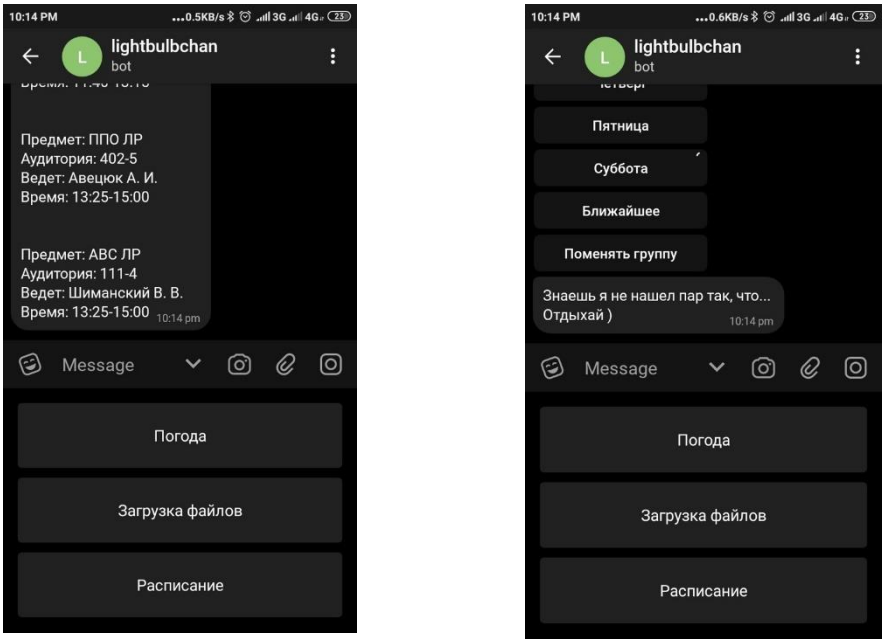

Рисунок 2.4.Расписание для дня недели.

Второй пункт стартового меню - “Погода”. При нажатии на данную клавишу, бот запросит у пользователя разрешение на получение текущей геолокации. После ее получения программа производит запрос в сторонний сервис ЯндексПогода для получения актуальной информации. После получения ответа на запрос, бот отправит текущие температуру и скорость ветра в данном регионе пользователю (отдельно предупредив в случаях, когда температура или скорость ветра принимают слишком высокие или низкие значения). (Рис. 2.5.)

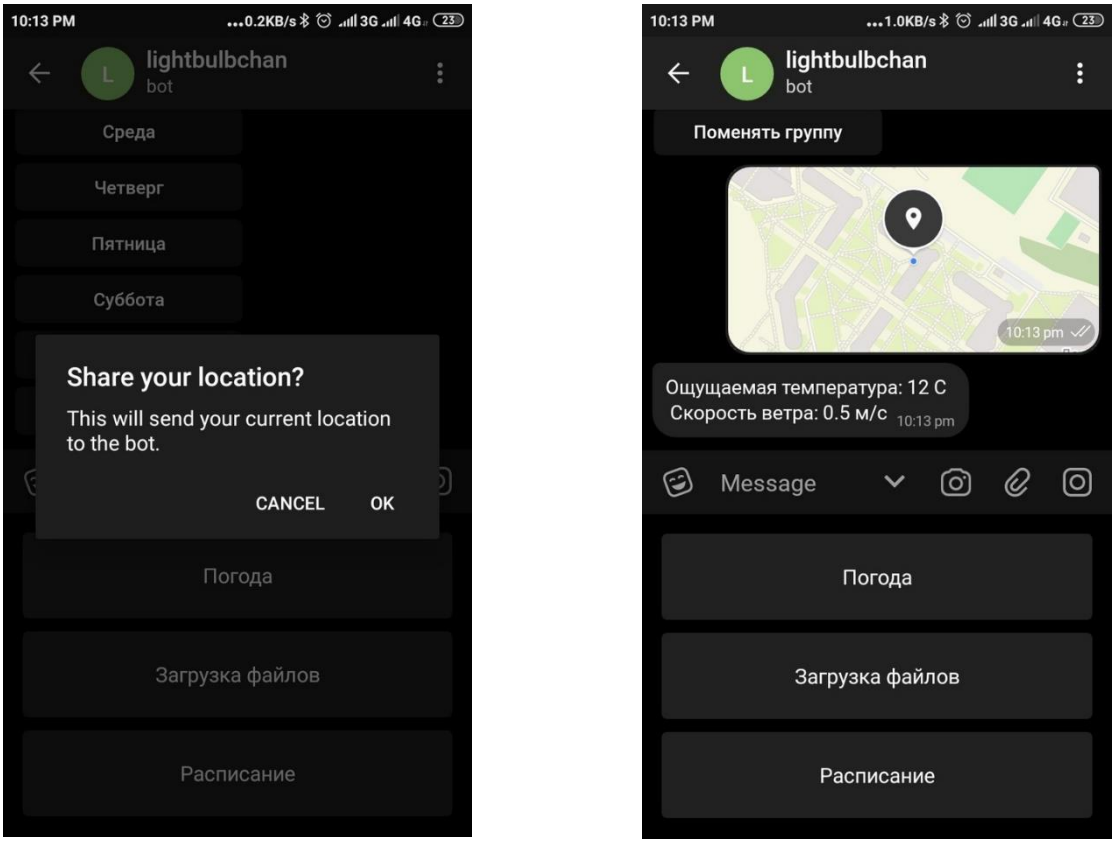

Рисунок 2.5.Запрос геолокации и демонстрация погоды.

Третий пункт в меню бота - “Загрузка файлов”. Данный сервис позволяет отправлять файлы, находящиеся на внешнем флеш-накопителе, подключенном к Raspberry Pi через один из нескольких USB-портов, в виде телеграм-сообщений. При выборе данной опции пользователю предлагается выбрать расширение файлов, на наличие которых будет сканироваться флеш-накопитель (.txt, .pdf, .mp3). При выборе определенного расширения бот сканирует флеш-накопитель начиная с корневой директории и отправляет пользователю все найденные файлы с возможностью скачать их. (Рис. 2.6.)

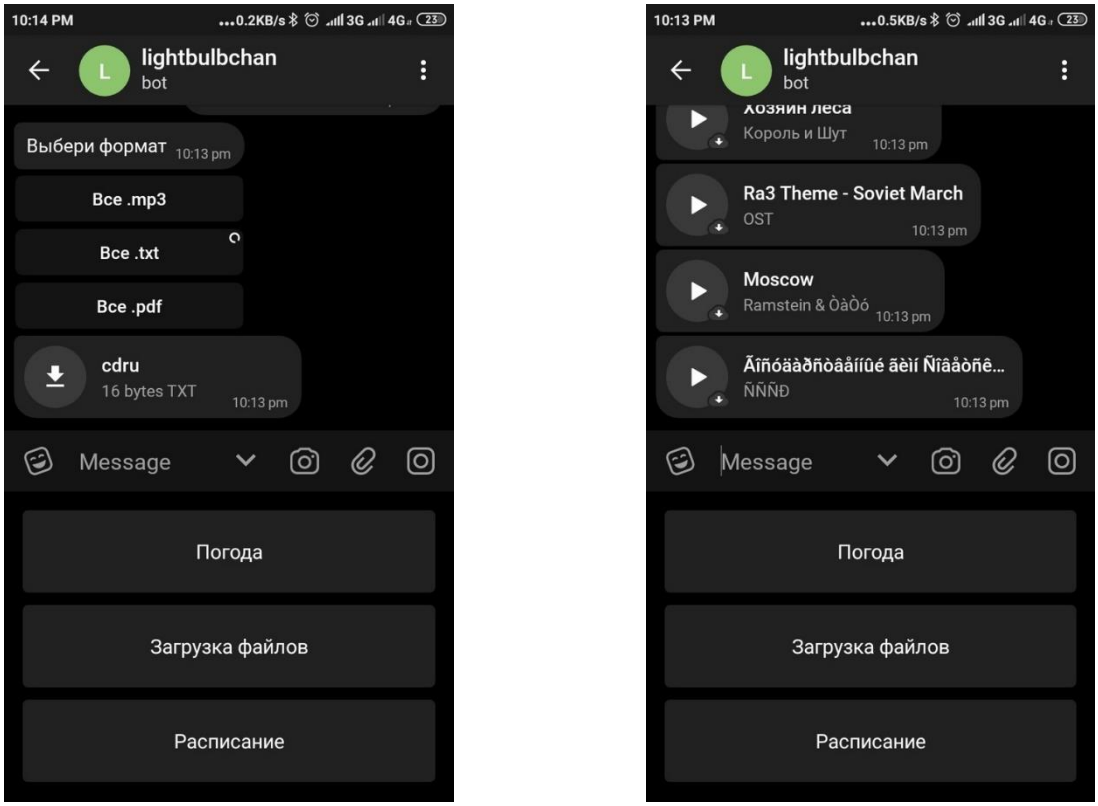

Рисунок 2.6.Запрос типа файла и их загрузка.

Если в любой момент работы бота пользователь введет неподходящие данные или попытается обойти предоставляемую клавиатуру, сработает механизм валидации ввода и будет выведено сообщение, предупреждающее о неверном вводе. Так же существует предупреждение пользователя о том, что какой-то из сервисов не смог получить актуальную информацию из стороннего ресурса.

Так как Телеграм является кроссплатформенным мессенджером, весь данный функционал так же возможен и с персональных компьютеров. За исключением сервиса погоды (в связи с невозможностью получения геолокации). (Рис. 2.7)

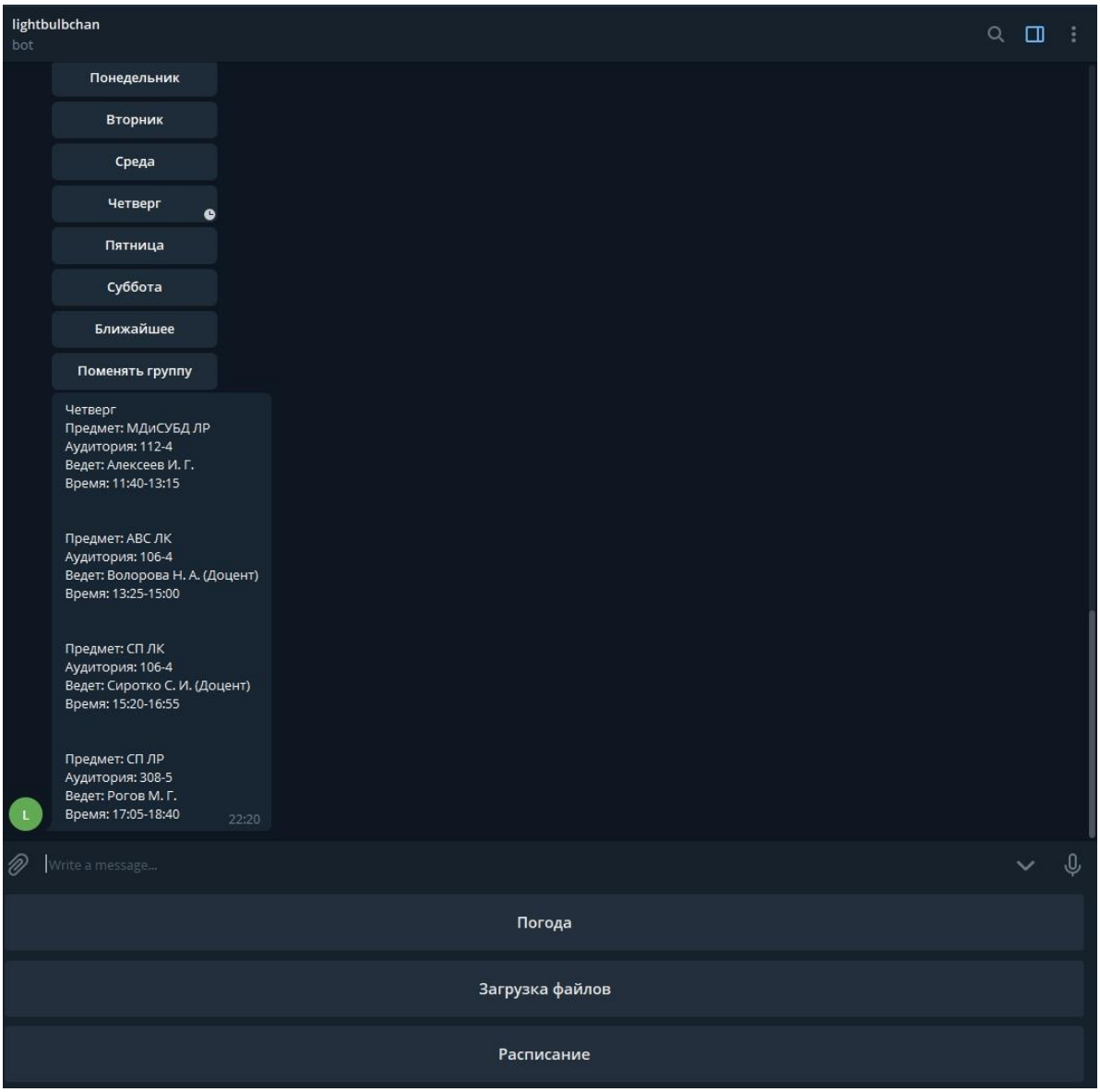

Рисунок 2.7.Просмотр расписания с ПК.

Заключение

В результате проделанной работы был реализован небольшой сервис, демонстрирующий, что одноплатный компьютер Raspberry Pi вполне способен выполнять простые программы, в режиме онлайн осуществлять обращения к сторонним ресурсам, для приобретения необходимой пользователю информации. Была показана возможность загрузки файлов с внешнего носителя в сеть интернет.

Так же в связи с маленькими размерами и низким потреблением энергии данная платформа может постоянно находиться в работающем состоянии, готовая к обработке запросов, в чем и заключается главная особенность платформ на архитектуре ARM.

Список использованной литературы

1. Альтернативы Raspberry Pi. [электронный ресурс] - https://habr.com/ru/post/457666/
2. Система автоматической сборки Gradle [Электронный ресурс] - https://gradle.org/
3. Боты – информация для разработчиков [Электронный ресурс] - https://tlgrm.ru/docs/bots
4. API iis.bsuir руководство пользования [Электронный ресурс] - https://iis.bsuir.by/api/rules

Приложение. Текст программы

Class Core

```
package com.light.bulb.chan.lightbulbchan;

import com.light.bulb.chan.lightbulbchan.models.*;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.io.File;

@Service
public class Core implements MessageReceiver {
    @Autowired
    private TelegramGateway gateway;
    @Autowired
    private Weather weather;
    @Autowired
    private ScheduleManager scheduleManager;
    @Autowired
    private Files files;

    @Override
    public void OnMessage(LBCMessage message) {
        if (message.getType() == LBCMessageType.WEATHER) {
            Response<WeatherResponse> response = weather.getWeather(message.getLon(), message.getLat());
            if (response.isStatus()) {
                gateway.sendMessage(Weather.parseWeather(response.getValue()), message.getChatId());
            }
        }

        if (message.getType() == LBCMessageType.SCHEDULE) {
            Response<FullSchedule> response = scheduleManager.getSchedule(message.getGroupId());
            if (response.isStatus()) {
                if (message.getDay() > 5) {
                    gateway.sendMessage(ScheduleManager.parseSchedule(response.getValue()), message.getChatId());
                } else {
                    gateway.sendMessage(ScheduleManager.parseForDay(response.getValue(), message.getDay()),
message.getChatId());
                }
            }
        }

        if (message.getType() == LBCMessageType.FILES) {
            for (File file : files.getAllWithExpansion(message.getExpansion())) {
                gateway.sendFile(message.getChatId(), file);
            }
        }
    }
}
```

Class BotSettings

```
package com.light.bulb.chan.lightbulbchan;

import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.stereotype.Component;

@Component
@ConfigurationProperties(prefix = "lightbulbchan.bot")
```

```
public class BotSettings {
    private String name;
    private String token;

    public String getName() {
        return name;
    }

    public BotSettings setName(String name) {
        this.name = name;
        return this;
    }

    public String getToken() {
        return token;
    }

    public BotSettings setToken(String token) {
        this.token = token;
        return this;
    }
}
```

Class Files

```
package com.light.bulb.chan.lightbulbchan;

import org.springframework.stereotype.Service;

import java.io.File;
import java.util.ArrayList;
import java.util.List;

@Service
public class Files {
    public Files() {
    }

    private File[] getAllFiles() {
        return File.listRoots()[2].listFiles();
    }

    public File[] getAllWithExpansion(final String expansion) {
        List<File> list = new ArrayList<>();
        for (File file : getAllFiles()) {
            if (file.getName().contains(expansion)) {
                list.add(file);
            }
        }
        File[] arr = new File[list.size()];
        list.toArray(arr);
        return list.toArray(arr);
    }
    public File[] getAllMp3() {
        return getAllWithExpansion(".mp3");
    }

    public File[] getAllTxt() {
        return getAllWithExpansion(".txt");
    }

    public File[] getAllPdf() {
        return getAllWithExpansion(".pdf");
    }
```

```
}
```

Class LightbulbchanApolication

```
package com.light.bulb.chan.lightbulbchan;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.telegram.telegrambots.ApiContextInitializer;
import org.telegram.telegrambots.meta.TelegramBotsApi;
import org.telegram.telegrambots.meta.exceptions.TelegramApiRequestException;

@SpringBootApplication
public class LightbulbchanApplication {
	static {
		ApiContextInitializer.init();
	}
	public static void main(String[] args) throws TelegramApiRequestException {
		SpringApplication.run(LightbulbchanApplication.class, args);
	}

}
```

Class SheduleManager

```
package com.light.bulb.chan.lightbulbchan;

import com.light.bulb.chan.lightbulbchan.models.FullSchedule;
import com.light.bulb.chan.lightbulbchan.models.Response;
import com.light.bulb.chan.lightbulbchan.models.SubjectSchedule;
import org.springframework.stereotype.Service;
import org.springframework.web.client.RestClientException;
import org.springframework.web.client.RestTemplate;

import java.util.Calendar;
import java.util.Date;

@Service
public class ScheduleManager {
   private final RestTemplate restTemplate;

   public static String parseSchedule(final FullSchedule fullSchedule) {
      StringBuilder result = new StringBuilder();
      Calendar c = Calendar.getInstance();
      c.setTime(new Date());
      int dayOfWeek = c.get(Calendar.DAY_OF_WEEK);
      dayOfWeek = (dayOfWeek + 5) % 7;
      for (int i = 0; i < 7; ++i) {
         if ((i + dayOfWeek) % 7 == 6) {
            continue;
         }
         if (fullSchedule.getSchedules()[(i + dayOfWeek) % 7].getSchedule() == null) {
            continue;
         }

         SubjectSchedule[] subjects = fullSchedule.getSchedules()[(i + dayOfWeek) % 7].getSchedule();
         StringBuilder temp = new StringBuilder();
         for (int j = 0; j < subjects.length; ++j) {
            for (Long x : subjects[i].getWeekNumber()) {
               if (x.equals(fullSchedule.getCurrentWeekNumber())) {
                  temp.append("Предмет: ").append(subjects[j].getSubject()).append(" ")
                        .append(subjects[j].getLessonType()).append("\n");
                  if (subjects[j].getAuditory().length > 0) {
                     temp.append("Аудитория: ").append(subjects[j].getAuditory()[0]).append("\n");
```

```
                }
                if (subjects[j].getEmployee().length > 0) {
                    temp.append("Ведет: ").append(subjects[j].getEmployee()[0].getFio()).append("\n");
                }
                temp.append("Время: ").append(subjects[j].getLessonTime()).append("\n\n\n");
                break;
            }

        }
    }

    if (temp.length() > 0) {
        result.append("Ближайший учебный день: ").append(fullSchedule.getSchedules()[(i + dayOfWeek) %
7].getWeekDay()).append("\n").append(temp);
        return result.toString();
    }
  }

  result.append("Знаешь я не нашел пар так что... Отдыхай )");
  return result.toString();
 }

 public static String parseForDay(final FullSchedule fullSchedule, final int day) {
  StringBuilder result = new StringBuilder();

  if (fullSchedule == null || fullSchedule.getSchedules() == null || fullSchedule.getSchedules().length <= day) {
    result.append("Знаешь я не нашел пар так, что... Отдыхай )");
    return result.toString();
  }

  SubjectSchedule[] subjects = fullSchedule.getSchedules()[day].getSchedule();
  StringBuilder temp = new StringBuilder();

  for (int j = 0; j < subjects.length; ++j) {
    for (Long x : subjects[j].getWeekNumber()) {
      if (x.equals(fullSchedule.getCurrentWeekNumber())) {
        temp.append("Предмет: ").append(subjects[j].getSubject()).append(" ")
            .append(subjects[j].getLessonType()).append("\n");
        if (subjects[j].getAuditory().length > 0) {
          temp.append("Аудитория: ").append(subjects[j].getAuditory()[0]).append("\n");
        }
        if (subjects[j].getEmployee().length > 0) {
          temp.append("Ведет: ").append(subjects[j].getEmployee()[0].getFio()).append("\n");
        }
        temp.append("Время: ").append(subjects[j].getLessonTime()).append("\n\n\n");
        break;
      }

    }
  }

  if (temp.length() > 0) {
    result.append(fullSchedule.getSchedules()[day].getWeekDay()).append("\n").append(temp);
    return result.toString();
  }

  result.append("Знаешь я не нашел пар так, что... Отдыхай )");
  return result.toString();
 }

 public ScheduleManager() {
  restTemplate = new RestTemplate();
 }
```

```
    public Response<FullSchedule> getSchedule(final String groupId) {
        String url = String.format("https://journal.bsuir.by/api/v1/studentGroup/schedule?studentGroup=%s", groupId);
        try {
            FullSchedule response = restTemplate.getForObject(url, FullSchedule.class);
            return new Response<>(response, true);
        } catch (RestClientException e) {
            e.getLocalizedMessage();
            return new Response<>(null, false);
        }
    }

}
```

Class TelegramGateaway

```
package com.light.bulb.chan.lightbulbchan;

import com.light.bulb.chan.lightbulbchan.models.*;
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.telegram.telegrambots.bots.TelegramLongPollingBot;
import org.telegram.telegrambots.meta.TelegramBotsApi;
import org.telegram.telegrambots.meta.api.methods.send.SendDocument;
import org.telegram.telegrambots.meta.api.methods.send.SendMessage;
import org.telegram.telegrambots.meta.api.objects.CallbackQuery;
import org.telegram.telegrambots.meta.api.objects.Location;
import org.telegram.telegrambots.meta.api.objects.Message;
import org.telegram.telegrambots.meta.api.objects.Update;
import org.telegram.telegrambots.meta.api.objects.replykeyboard.ReplyKeyboard;
import org.telegram.telegrambots.meta.exceptions.TelegramApiException;
import org.telegram.telegrambots.meta.exceptions.TelegramApiRequestException;

import java.io.File;
import java.util.HashMap;
import java.util.Map;

@Service
public class TelegramGateway extends TelegramLongPollingBot {
    private static final Logger log = LogManager.getLogger(TelegramGateway.class);
    private static TelegramBotsApi telegramBotsApi = new TelegramBotsApi();
    private final BotSettings settings;
    private final Map<Long, String> chatToGroup = new HashMap<>();

    @Autowired
    private Core core;

    @Autowired
    public TelegramGateway(BotSettings settings) throws TelegramApiRequestException {
        this.settings = settings;
        telegramBotsApi.registerBot(this);
    }

    public Response sendMessage(final String text, final Long chatId, final ReplyKeyboard keyboard) {
        SendMessage sendMessageRequest = new
SendMessage().setText(text).setChatId(chatId).setReplyMarkup(keyboard);
        try {
            execute(sendMessageRequest);
        } catch (TelegramApiException e) {
            e.printStackTrace();
            log.error("Cannot send message. Text: " + text + " ChatId: " + chatId);
            new Response(null, false);
```

```
        }

        return new Response(null, true);
    }

    public Response sendMessage(final String text, final Long chatId) {
        return sendMessage(text, chatId, new MainMenuKeyboard());
    }

    public Response sendFile(final Long chatId, final File file) {
        SendDocument sendDocument = new SendDocument().setChatId(chatId).setDocument(file);
        try {
            execute(sendDocument);
        } catch (TelegramApiException e) {
            e.printStackTrace();
            log.error("Cannot send file. ChatId: " + chatId);
            new Response(null, false);
        }

        return new Response(null, true);
    }

    @Override
    public void onUpdateReceived(Update update) {
        if (update.hasMessage()) {
            Message message = update.getMessage();

            if (message.getLocation() != null) {
                Location location = message.getLocation();
                core.OnMessage(new LBCMessage().setType(LBCMessageType.WEATHER)

.setLat(Float.toString(location.getLatitude())).setLon(Float.toString(location.getLongitude())).setChatId(message.ge
tChatId()));
            } else if (message.getText().equals(MainMenuKeyboard.SCHEDULE_BUTTON_TEXT)) {
                if (!chatToGroup.containsKey(message.getChatId())) {
                    sendMessage("Введи свою группу пожалуйста :)", message.getChatId());
                } else {
                    sendMessage("Выбери опцию", message.getChatId(), new ScheduleKeyboard());
                }
            } else if (message.getText().equals(MainMenuKeyboard.FILES_BUTTON_TEXT)) {
                sendMessage("Выбери формат", message.getChatId(), new FilesKeyboard());
            } else if (message.getText().length() == 6) {
                chatToGroup.put(message.getChatId(), message.getText());
                core.OnMessage(new
LBCMessage().setType(LBCMessageType.SCHEDULE).setChatId(message.getChatId())
                        .setGroupId(chatToGroup.get(message.getChatId())));
            } else {
                sendMessage("Используйте меню для ввода!", message.getChatId());
            }
        }

        if (update.hasCallbackQuery()) {
            CallbackQuery callbackQuery = update.getCallbackQuery();
            Message message = callbackQuery.getMessage();
            for (String ext : FilesKeyboard.EXT) {
                if (ext.equals(callbackQuery.getData())) {
                    core.OnMessage(new
LBCMessage().setType(LBCMessageType.FILES).setChatId(message.getChatId()).setExpansion(ext));
                    return;
                }
            }
            int day = Integer.parseInt(callbackQuery.getData());
            if (day == -1) {
```

```
            sendMessage("Введи свою группу пожалуйста :)", message.getChatId());
            return;
        }
        core.OnMessage(new
LBCMessage().setType(LBCMessageType.SCHEDULE).setChatId(message.getChatId())
                .setGroupId(chatToGroup.get(message.getChatId())).setDay(day));
    }
  }


  @Override
  public String getBotUsername() {
    return settings.getName();
  }

  @Override
  public String getBotToken() {
    return settings.getToken();
  }
}
```

Class Weather

```
package com.light.bulb.chan.lightbulbchan;

import com.light.bulb.chan.lightbulbchan.models.LBCMessage;
import com.light.bulb.chan.lightbulbchan.models.Response;
import com.light.bulb.chan.lightbulbchan.models.WeatherResponse;
import org.springframework.http.HttpEntity;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpMethod;
import org.springframework.stereotype.Service;
import org.springframework.web.client.RestClientException;
import org.springframework.web.client.RestTemplate;

@Service
public class Weather {
  private final RestTemplate restTemplate;
  private final HttpHeaders headers;

  public static String parseWeather(final WeatherResponse response) {
    StringBuilder result = new StringBuilder();
    Long temp = Long.parseLong(response.getFact().getFeelsLike());
    Float cloudness = Float.parseFloat(response.getFact().getCloudness());
    Float prec = Float.parseFloat(response.getFact().getPrecStrength());
    Float precType = Float.parseFloat(response.getFact().getPrecType());

    if (temp < 0) {
      result.append("Кажется на улице холодно. Советую одется по теплее!\n");
    }

    if (temp > 25) {
      result.append("ЖАААРКО. Шорты твой выход)\n");
    }

    if (cloudness > 0) {
      result.append("☁\n");
    }

    if (prec > 0) {
      result.append("Имеются осадки!\n");
    }
```

```
        result.append("Ощущаемая температура: ").append(temp).append(" C\n Скорость ветра: " +
response.getFact().getWindSpeed()).append(" м/с");
        return result.toString();
    }

    public Weather() {
        restTemplate = new RestTemplate();
        headers = new HttpHeaders();
        headers.add("X-Yandex-API-Key", "f43e6136-6717-4212-905c-73aa0e9385b7");
    }

    public Response<WeatherResponse> getWeather(final String lan, final String lon) {
        String url =
String.format("https://api.weather.yandex.ru/v1/forecast?lat=%s&lon=%s&lang=ru_RU&limit=1&extra=1", lan,
lon);
        try {
            HttpEntity<String> request = new HttpEntity<String>(headers);
            WeatherResponse response = restTemplate.exchange(url, HttpMethod.GET, request,
WeatherResponse.class).getBody();
            return new Response<>(response, true);
        } catch (RestClientException e) {
            e.getLocalizedMessage();
            return new Response<>(null, false);
        }
    }
}
```

Class weatherResponce

```
package com.light.bulb.chan.lightbulbchan.models;

import com.fasterxml.jackson.annotation.JsonInclude;
import com.fasterxml.jackson.databind.PropertyNamingStrategy;
import com.fasterxml.jackson.databind.annotation.JsonNaming;

@JsonInclude(JsonInclude.Include.NON_NULL)
@JsonNaming(PropertyNamingStrategy.SnakeCaseStrategy.class)
public class WeatherResponse {
    private Fact fact;

    public Fact getFact() {
        return fact;
    }

    public WeatherResponse setFact(Fact fact) {
        this.fact = fact;
        return this;
    }
}
```

Class UserData

```
package com.light.bulb.chan.lightbulbchan.models;

public class UserData {
    private String lat;
    private String lon;
    private String group;
    private Long chatId;

    public String getLat() {
        return lat;
    }
```

```
    public UserData setLat(String lat) {
        this.lat = lat;
        return this;
    }

    public String getLon() {
        return lon;
    }

    public UserData setLon(String lon) {
        this.lon = lon;
        return this;
    }

    public String getGroup() {
        return group;
    }

    public UserData setGroup(String group) {
        this.group = group;
        return this;
    }

    public Long getChatId() {
        return chatId;
    }

    public UserData setChatId(Long chatId) {
        this.chatId = chatId;
        return this;
    }
}
```

Class SubjectShedule

```
package com.light.bulb.chan.lightbulbchan.models;

import com.fasterxml.jackson.annotation.JsonInclude;

@JsonInclude(JsonInclude.Include.NON_NULL)
public class SubjectSchedule {
    private Long[] weekNumber;
    private String lessonType;
    private String[] auditory;
    private String subject;
    private Employee[] employee;
    private String lessonTime;

    public Long[] getWeekNumber() {
        return weekNumber;
    }

    public SubjectSchedule setWeekNumber(Long[] weekNumber) {
        this.weekNumber = weekNumber;
        return this;
    }

    public String getLessonType() {
        return lessonType;
    }

    public SubjectSchedule setLessonType(String lessonType) {
        this.lessonType = lessonType;
```

```
        return this;
    }

    public String[] getAuditory() {
        return auditory;
    }

    public SubjectSchedule setAuditory(String[] auditory) {
        this.auditory = auditory;
        return this;
    }

    public String getSubject() {
        return subject;
    }

    public SubjectSchedule setSubject(String subject) {
        this.subject = subject;
        return this;
    }

    public Employee[] getEmployee() {
        return employee;
    }

    public SubjectSchedule setEmployee(Employee[] employee) {
        this.employee = employee;
        return this;
    }

    public String getLessonTime() {
        return lessonTime;
    }

    public SubjectSchedule setLessonTime(String lessonTime) {
        this.lessonTime = lessonTime;
        return this;
    }
}
```

Class SheduleKeyboard

```
package com.light.bulb.chan.lightbulbchan.models;

import org.telegram.telegrambots.meta.api.objects.replykeyboard.InlineKeyboardMarkup;
import org.telegram.telegrambots.meta.api.objects.replykeyboard.buttons.InlineKeyboardButton;

import java.util.ArrayList;
import java.util.List;

public class ScheduleKeyboard extends InlineKeyboardMarkup {
    private static final String[] days = {"Понедельник", "Вторник", "Среда", "Четверг", "Пятница", "Суббота"};

    public ScheduleKeyboard() {
        List<List<InlineKeyboardButton>> allButtons = new ArrayList<>(){};
        for (int i = 0; i < 6; ++i) {
            List<InlineKeyboardButton> buttonList = new ArrayList<>();
            buttonList.add(new InlineKeyboardButton().setText(days[i]).setCallbackData(Integer.toString(i)));
            allButtons.add(buttonList);
        }

        List<InlineKeyboardButton> buttonList = new ArrayList<>();
        buttonList.add(new InlineKeyboardButton().setText("Ближайшее").setCallbackData(Integer.toString(6)));
        allButtons.add(buttonList);
```

```
        buttonList = new ArrayList<>();
        buttonList.add(new InlineKeyboardButton().setText("Поменять группу").setCallbackData(Integer.toString(-
1)));
        allButtons.add(buttonList);
        setKeyboard(allButtons);
    }
}
```

Class Response

```
package com.light.bulb.chan.lightbulbchan.models;

public class Response<T> {
    private T value;
    private boolean status;

    public Response(T value, boolean status) {
        this.value = value;
        this.status = status;
    }

    public T getValue() {
        return value;
    }

    public Response<T> setValue(T value) {
        this.value = value;
        return this;
    }

    public boolean isStatus() {
        return status;
    }

    public Response<T> setStatus(boolean status) {
        this.status = status;
        return this;
    }
}
```

Class MainMenuKeyboad

```
package com.light.bulb.chan.lightbulbchan.models;

import org.telegram.telegrambots.meta.api.objects.replykeyboard.ReplyKeyboardMarkup;
import org.telegram.telegrambots.meta.api.objects.replykeyboard.buttons.KeyboardButton;
import org.telegram.telegrambots.meta.api.objects.replykeyboard.buttons.KeyboardRow;

import java.util.ArrayList;
import java.util.List;

public class MainMenuKeyboard extends ReplyKeyboardMarkup {
    public static final String WEATHER_BUTTON_TEXT = "Погода";
    public static final String FILES_BUTTON_TEXT = "Загрузка файлов";
    public static final String SCHEDULE_BUTTON_TEXT = "Расписание";

    public MainMenuKeyboard() {
        List<KeyboardRow> buttons = new ArrayList<>();
        KeyboardRow row = new KeyboardRow();
        row.add(0, new KeyboardButton().setText(WEATHER_BUTTON_TEXT).setRequestLocation(true));
        buttons.add(row);
        row = new KeyboardRow();
        row.add(0, new KeyboardButton().setText(FILES_BUTTON_TEXT));
```

```
        buttons.add(row);
        row = new KeyboardRow();
        row.add(0, new KeyboardButton().setText(SCHEDULE_BUTTON_TEXT));
        buttons.add(row);
        setKeyboard(buttons);
    }
}
```

Class LBCMessage

```
package com.light.bulb.chan.lightbulbchan.models;

public class LBCMessage {
    private LBCMessageType type;
    private String data;
    private String lat;
    private String lon;
    private Long chatId;
    private String groupId;
    private int day;
    private String expansion;

    public LBCMessageType getType() {
        return type;
    }

    public LBCMessage setType(LBCMessageType type) {
        this.type = type;
        return this;
    }

    public String getData() {
        return data;
    }

    public LBCMessage setData(String data) {
        this.data = data;
        return this;
    }

    public String getLat() {
        return lat;
    }

    public LBCMessage setLat(String lat) {
        this.lat = lat;
        return this;
    }

    public String getLon() {
        return lon;
    }

    public LBCMessage setLon(String lon) {
        this.lon = lon;
        return this;
    }

    public Long getChatId() {
        return chatId;
    }

    public LBCMessage setChatId(Long chatId) {
```

```
        this.chatId = chatId;
        return this;
    }

    public String getGroupId() {
        return groupId;
    }

    public LBCMessage setGroupId(String groupId) {
        this.groupId = groupId;
        return this;
    }

    public int getDay() {
        return day;
    }

    public LBCMessage setDay(int day) {
        this.day = day;
        return this;
    }

    public String getExpansion() {
        return expansion;
    }

    public LBCMessage setExpansion(String expansion) {
        this.expansion = expansion;
        return this;
    }
}
```

Class FullSchedule

```
package com.light.bulb.chan.lightbulbchan.models;

import com.fasterxml.jackson.annotation.JsonInclude;

@JsonInclude(JsonInclude.Include.NON_NULL)
public class FullSchedule {
    private DaySchedule[] schedules;
    private Long currentWeekNumber;

    public DaySchedule[] getSchedules() {
        return schedules;
    }

    public FullSchedule setSchedules(DaySchedule[] schedules) {
        this.schedules = schedules;
        return this;
    }

    public Long getCurrentWeekNumber() {
        return currentWeekNumber;
    }

    public FullSchedule setCurrentWeekNumber(Long currentWeekNumber) {
        this.currentWeekNumber = currentWeekNumber;
        return this;
    }
}
```

Class FilesKeyboard

```
package com.light.bulb.chan.lightbulbchan.models;
```

```
import org.telegram.telegrambots.meta.api.objects.replykeyboard.InlineKeyboardMarkup;
import org.telegram.telegrambots.meta.api.objects.replykeyboard.buttons.InlineKeyboardButton;

import java.util.ArrayList;
import java.util.List;

public class FilesKeyboard extends InlineKeyboardMarkup {
    public static final String[] EXT = {".mp3", ".txt", ".pdf"};

    public FilesKeyboard() {
        List<List<InlineKeyboardButton>> allButtons = new ArrayList<>();
        for (int i = 0; i < 3; ++i) {
            List<InlineKeyboardButton> buttonList = new ArrayList<>();
            buttonList.add(new InlineKeyboardButton().setText("Все " + EXT[i]).setCallbackData(EXT[i]));
            allButtons.add(buttonList);
        }

        setKeyboard(allButtons);
    }
}
```

Class Fact

```
package com.light.bulb.chan.lightbulbchan.models;

import com.fasterxml.jackson.annotation.JsonInclude;
import com.fasterxml.jackson.databind.PropertyNamingStrategy;
import com.fasterxml.jackson.databind.annotation.JsonNaming;

@JsonInclude(JsonInclude.Include.NON_NULL)
@JsonNaming(PropertyNamingStrategy.SnakeCaseStrategy.class)
public class Fact {
    private String temp;
    private String feelsLike;
    private String condition;
    private String windSpeed;
    private String precType;
    private String precStrength;
    private String cloudness;
    private String season;

    public String getTemp() {
        return temp;
    }

    public Fact setTemp(String temp) {
        this.temp = temp;
        return this;
    }

    public String getFeelsLike() {
        return feelsLike;
    }

    public Fact setFeelsLike(String feelsLike) {
        this.feelsLike = feelsLike;
        return this;
    }

    public String getCondition() {
        return condition;
    }
```

```
    public Fact setCondition(String condition) {
        this.condition = condition;
        return this;
    }

    public String getWindSpeed() {
        return windSpeed;
    }

    public Fact setWindSpeed(String windSpeed) {
        this.windSpeed = windSpeed;
        return this;
    }

    public String getPrecType() {
        return precType;
    }

    public Fact setPrecType(String precType) {
        this.precType = precType;
        return this;
    }

    public String getPrecStrength() {
        return precStrength;
    }

    public Fact setPrecStrength(String precStrength) {
        this.precStrength = precStrength;
        return this;
    }

    public String getCloudness() {
        return cloudness;
    }

    public Fact setCloudness(String cloudness) {
        this.cloudness = cloudness;
        return this;
    }

    public String getSeason() {
        return season;
    }

    public Fact setSeason(String season) {
        this.season = season;
        return this;
    }
}
```