**Руководство оператора**

1. **Назначение программы**

Основное назначение программного комплекса (client+ai) – управление площадкой умных вещей (в данной задаче openhab) посредством голоса.

1. **Условия выполнения программы**

Данный программный разработан разработан в среде разработки Nodepad++ и Android studio ide на операционной системе Windows 8.1.

**- Серверная часть**

Серверная часть комплекса предназначена к применению на компьютерах со следующим установленным ПО : nodejs, npm .

**- Клиентская часть**

Клиентская часть комплекса предназначена к применению на мобильных устройствах под управлением ос Android 4.1 и выше.

1. **Выполнение программы**

- **Серверная часть**

Для запуска программы необходимо запустить файл server.js. После программа инициализирует веб-сервер по указанному оператором порту(по умолчанию 3000) выход из программы осуществляется через принудительное завершение операции в консоли(ctrl + c).

**- Клиентская часть**

Для запуска клиента необходимо установить непосредственно сам клиент. Далее запускаем приложение, открываем меню и выбираем настройки (settings). В настройках указываем адрес площадки openhab (openhab url) – для ipv4, openhab remote url – в случае подключения по ipv6.Для подключения AI площадки в настройках нашего клиента указываем адрес(ipv4 или ipv6) в пункте AI Remote URL.Более подробную информацию о настройке openhab и подключении к нему клиента можно узнать на официальном сайте (openhab.org).

1. **Создание диалогов**

Создать диалог можно несколькими способами : через обычный веб-браузер (во вкладке admin), через клиент(меню-> Ai panel), или редактируя сам файл разметки(файл openhab.aiml, хранимый в корне проекта).Все доступные команды представлены в виде разметки по спецификации A.I.M.L, но с возможностью многоступенчатой беседы(до 3 колена).Расширенная разметка выглядит следующим образом:

<category>

<pattern>VV</pattern> - условие первого паттерна

<template>HI </template> - вывод на условие

<subcategory>

<pattern>MM</pattern> - условие второго колена данной категории

<template>VU</template>

</subcategory>

<subcategory>

<pattern></pattern> - условие третьего колена данной категории

<template></template>

</subcategory>

</category>

В данном примере общение построено следующим образом: Входящие сообщения(от оператора) VV->MM, где конечный ответ на последний шаблон ответа (template) будет VU, и именно данный конечный ответ и будет передан на площадку openhab. Система на уровне api оценивает, есть ли дальнейшее развитие диалога. В данном примере диалог построен до 2 колена (третье колено пустое, однако данная реализация предполагает собой обязательное условие разметки 2 сабкатегорий, а именно второго и третьего коленей, даже если они будут пустыми). На графическом уровне все диалоги представлены в виде иерархического дерева, работающего на rest api. Для редактирования выбираем блок, его колено и затем щелкаем на редактируемый тип (template или pattern) и соответственно изменяем его. Для удаления конкретного блока щелкаем на знак минус, расположенный в крайнем правом углу удаляемого блока. Для добавления блока щелкаем на знак плюс в верхней панели в правом углу. В случае, если необходимо удалить конечное колено конкретного блока, достаточно удалить текст паттерна(pattern), в таком случае колено выше будет являться конечным.

1. **Rest API**

В случае интеграции с другими продуктами отличными от openhab, серверная часть программного комплекса была выполнена на rest api.

**Общение**

Для общения используется вкладка talk и обращение к ней идет методом GET. Система принимает несколько параметров: sentence – предложение пользователя, s2 – следующее предложение пользователя(второе колено), s3 – конечное третье колено. Данные отдаются в формате json c двумя параметрами: response(ответ на запрос пользователя), exit(1 – если нет дальнейшего развития диалога, либо 0 – если есть следующее колено).

**Редактирование**

Редактирование происходит методом POST во вкладке admin-rest.

Параметры для редактирования конкретного блока:value (новое значение),dataid(номер редактируемого блока, или его место по порядку в файле habmind.aiml начиная с единицы),typeid(pattern или template),datasub(колено, начинается с единицы. Если это корень, то параметр не нужен).

Параметры для создания нового блока: add (значение по умолчанию true).

Параметры для удаления конкретного блока: dataid (номер удаляемого блока, или его место по порядку в файле habmind.aiml начиная с единицы)

Параметры для получения всех блоков: fetch (значение по умолчанию true).

**Пример создания простой логики (моста между openhab и ai)**

В openhab все управление идет через систему называемую правила. Правила удобно задавать через openhab designer (подробнее на официальном сайте openhab https://github.com/openhab/openhab/wiki/Samples-Item-Definitions#how-to-set-up-voice-control-for-use-with-habdroid).

Имя правила в openhab – это есть конечная команда в ai (а именно последнее колено). Например, если в правиле его имя light, а в ai дерево выглядит как turn -> me -> light, то при дохождении до light (последнего колена) клиент посылает запрос на это правило по его имени.

**Нужен ли для этого интернет?**

Для распознания текста по голосу (stt speech-to-text) гугл использует свои базы в режиме online, однако для определенных языков эти базы могут быть доступны в режиме offline. Подробнее <http://blinddroid.ru/blog/android-and-samsung/zagruzka-komponenta-dlya-raspoznavaniya-rechi-offlain>.

**Демо конфигурация**

В конфиге прописано при исполнении голосовой команды (turn->me->light) включается весь свет.