ОБЩЕСТВО С ОГРАНИЧЕННОЙ ОТВЕТСТВЕННОСТЬЮ
«ФАБМИКРО»

ВЫЗЫВНАЯ ПАНЕЛЬ
IP-ДОМОФОН «ЕРМАК»

(SIPHomePhone версия 2.x)

Руководство по эксплуатации.

Руководство администратора.

Руководство программиста.

Приложения.

Выполнил:
Залата Р.Н.
Мальцев В.В.
Эзау М.Я.

Тюмень 2023

АННОТАЦИЯ

Данный документ содержит описание изделия Вызывная Панель IP-ДОМОФОН «ЕРМАК», разработанного в ООО «Фабмикро» по заказу ООО «ДатаЦентр» (договор на ОКР N 4Р/19 от 01.08.2019), инструкцию по эксплуатации данного изделия, инструкцию по установке, настройке и администрированию. Также в данном документе представлено описание программной части, включая API всех программных модулей и структуру их взаимодействия в процессе функционирования в составе изделия.

СОДЕРЖАНИЕ

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Дата внесения изменений	Текст изменений	Перечень разделов

ПЕРЕЧЕНЬ ТЕРМИНОВ И СОКРАЩЕНИЙ

API

Application Program Interface, интерфейс программного взаимодействия - описание способов (набор классов, процедур, функций, структур или констант), которыми одна компьютерная программа или модуль может взаимодействовать с другой программой.

JSON

JavaScript Object Notation, открытый независимый формат представления данных с использованием человеко-читаемого текста. Содержит типизированные пары «аттрибут-значение», из которых могут быть сформированы списки, массивы и более сложные вложенные структуры.

DHCP

Dynamic Host Configuration Protocol, протокол для автоматического конфигурирования устройств в IP-сетях. Позволяет устройству в процессе загрузки получить IP-адрес и другие параметры сети, необходимые для функционирования.

RFID метка

Radio Frequanfy Indentification, способ идентификации с помощью радио-передающих меток. RFID метка содержит антенну и микросхему трансивера. Метка, помещенная в высокочастотное магнитное поле, используя антенну и посредством модуляции поля, передает в приемное устройство свой уникальный, запрограммированный при её производстве, идентификатор (число), который далее используется для идентификации посетителей.

ВП

Вызывная панель, устройство устанавливаемое на входной двери офисного здания или МКД для предоставления посетителям возможности осуществить разговор с человеком в помещением. Позволяет идентифицировать посетителя,

отпирать замок двери по команде с удаленного переговорного устройства или по RFID ключу.

Конфиг

Консолидированный конфигурационный файл определенного формата и структуры (в данном случае - JSON структура).

МКД

Многоквартирный жилой дом.

Модуль

Часть программного комплекса, функционально ограниченная определенным кругом задач, имеющая строго специфицированный интерфейс взаимодействия (API).

СнК

Система-на-кристалле, микросхема сверхвысокой степени интеграции (СБИС), содержащая на одном кремниевом кристалле вычислительное ядро (микропроцессор), различные периферийные устройства, постоянную и оперативную память. Большинство современных микропроцессоров являются системами-на-кристалле.

ПЕРЕЧЕНЬ РИСУНКОВ И ТАБЛИЦ

1. Руководство по эксплуатации

1.1. Описание изделия

Изделие Вызывная Панель «ЕРМАК» (далее «Панель») предназначено для обеспечения двусторонней аудио и видео связи в системах контроля доступа в помещения, в том числе жилые, офисные или производственные. Помимо связи, Панель обеспечивает контроль датчиков состояния дверей и управление дверными замками, позволяет производить отображение полезной информации (или рекламных объявлений) на цветном ЖК-дисплее. Вызывная Панель может функционировать как в одиночном режиме, так и в составе системы подобных изделий совместимых со стандартом SIP 2.0. Панель может взаимодействовать с серверным программным обеспечением по протоколу HTTPS (JSON): запрашивать с сервера необходимую конфигурационную информацию и список RFID ключей, загружать объявления, а также отправлять на сервер информацию о произошедших событиях (нотификации) и получать команды на отпирание замков. Вызывная панель может быть интегрирована в существующую аналоговую домофонную сеть через специализированный Коммутатор, связь с которым осуществляется через интерфейс RS-485/Modbus, таким образом позволяя осуществить плавный переход от устаревших аналоговых домофонных сетей к цифровым технологиям.

Функциональная схема вызывной панели представлена на рис. 1.0. Вызывная панель представляет собой микропроцессорное устройство, построенное на базе системы-на-кристалле (СнК) ***A20*** производства компании ***Allwinner Technologies*** (КНР) (версии 1.х изделия использован СнК A13), в составе которой находятся два вычислительных ядра ARM Cortex-A7, а также ряд периферийных модулей, аппаратных энкодеров видеосигнала и цифровых интерфейсов, обеспечивающих функционирование изделия. Панель работает под управлением ОС Linux (ядро версии 5.18, набор системных утилит на основе дистрибутива Devuan). Загрузка операционной системы осуществляется либо со встроенного носителя eMMC, либо с внешней microSD-карты, устанавливаемой в слот, расположенный на задней стенке Панели в отсеке для разъемов. Загрузка операционной системы осуществляется сразу после подачи питания, которое может подаваться либо через разъем +12В, либо через кабель Ethernet согласно PoE 802.3af. На Панели, под управлением ОС Linux функционирует специализированное ПО, состоящее из ряда модулей

(процессов), обеспечивающих функционал Панели и логику обработки вызовов. Все модули имеют специфицированный интерфейс, описанный в разделе «3. *Руководство программиста*», что позволяет эксплуатанту создавать свои кастомизированные системы на основе готового решения, гибко расширять функционал и адаптировать работу устройства под свои потребности.

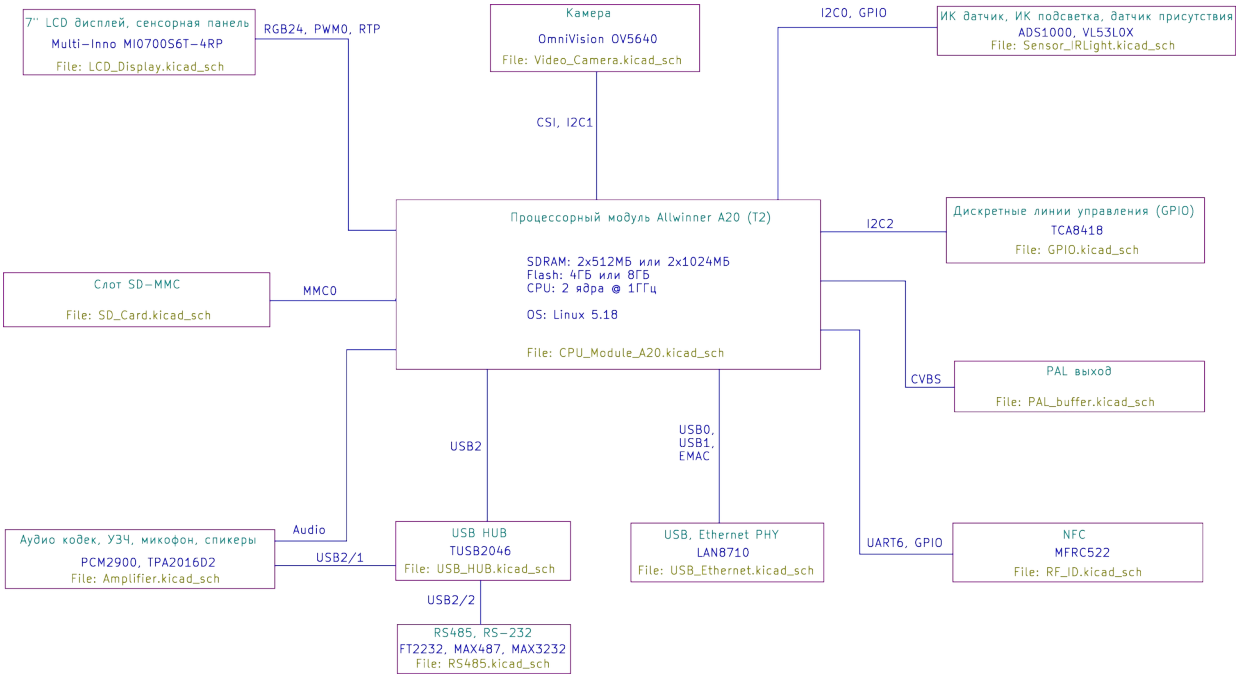


Рис.1.0. Структурная схема вызывной панели «ЕРМАК».

1.2. Органы управления и отображения информации

На рис. 1.1 ниже представлена лицевая сторона Вызывной Панели и схематично обозначены основные элементы органов управления.

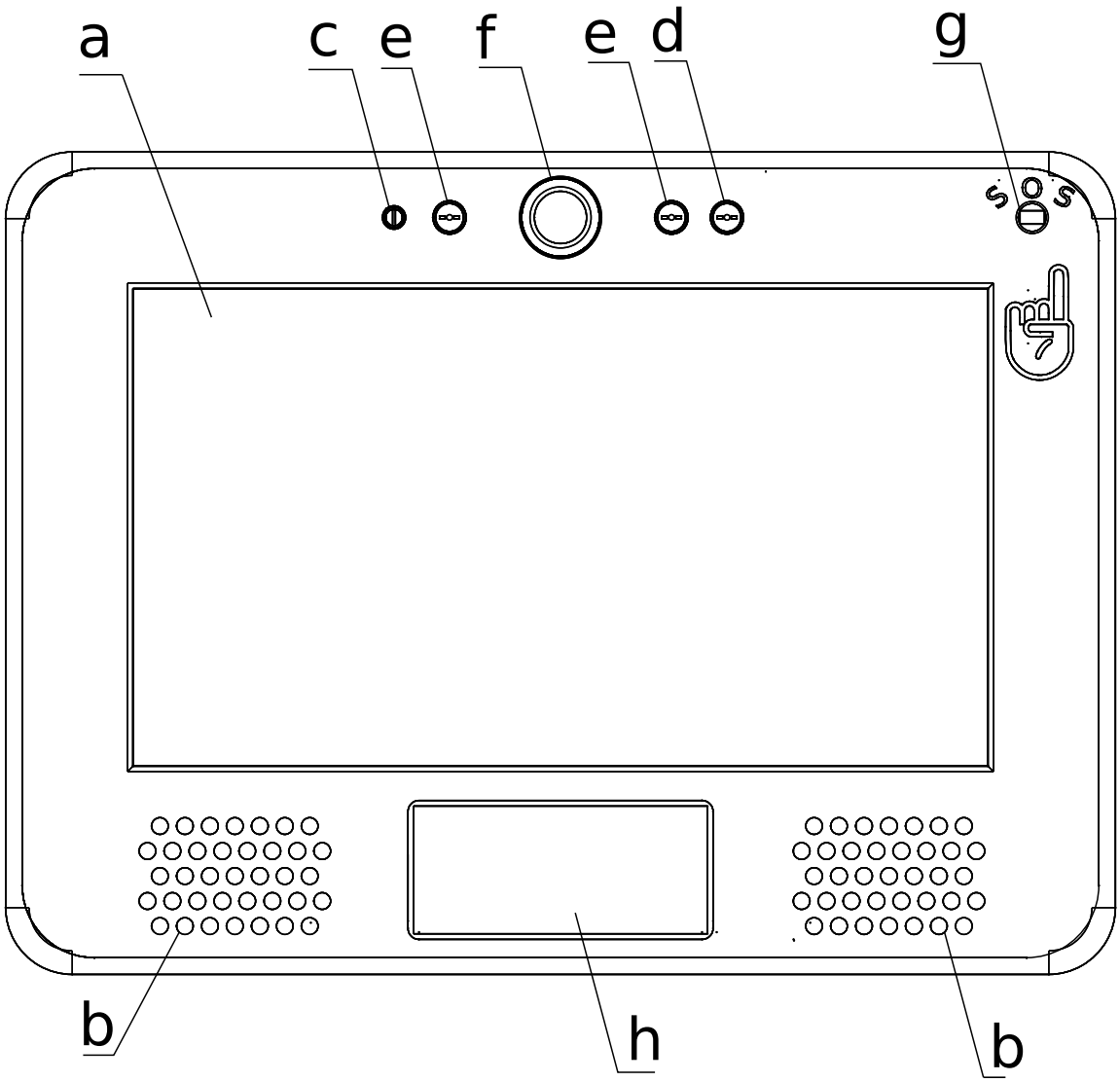


Рис.1.1. Лицевая сторона ВП. Органы управления и отображения информации.

К ним относятся:

a) цветной ЖК-дисплей диагональю 7“ с функцией резистивного сенсорного экрана (RTP);
b) два громкоговорителя мощностью по 1Вт каждый;
c) емкостной микрофон;
d) датчик освещенности (Ambient Sensor);
e) два ИК светодиода ночной подсветки;
f) видеокамера разрешением 1920x1080 @ 30 fps (или 15 fps для версии 1.x);
g) датчик присутствия (Proximity Sensor), также используется как кнопка «экстренного вызова» (SOS);
h) антенна считывателя RFID меток NFC/MIFARE (или Em-Marin) в версии 1.0).

Взаимодействие конечного пользователя с Вызывной Панелью осуществляется посредством графического интерфейса пользователя с сенсорным вводом. Описание графического интерфейса пользователя дано в разделе «*1.4. Руководство конечного пользователя*».

1.3. Внешние соединения

На рис. 1.2a представлена оборотная сторона Вызывной Панели, на которой располагаются четыре монтажных отверстия M5, крышка и отсек с разъемами. Схематичное расположение разъемов приведено на рис 1.2b, а их описание и назначение дано в таблице 1.2.

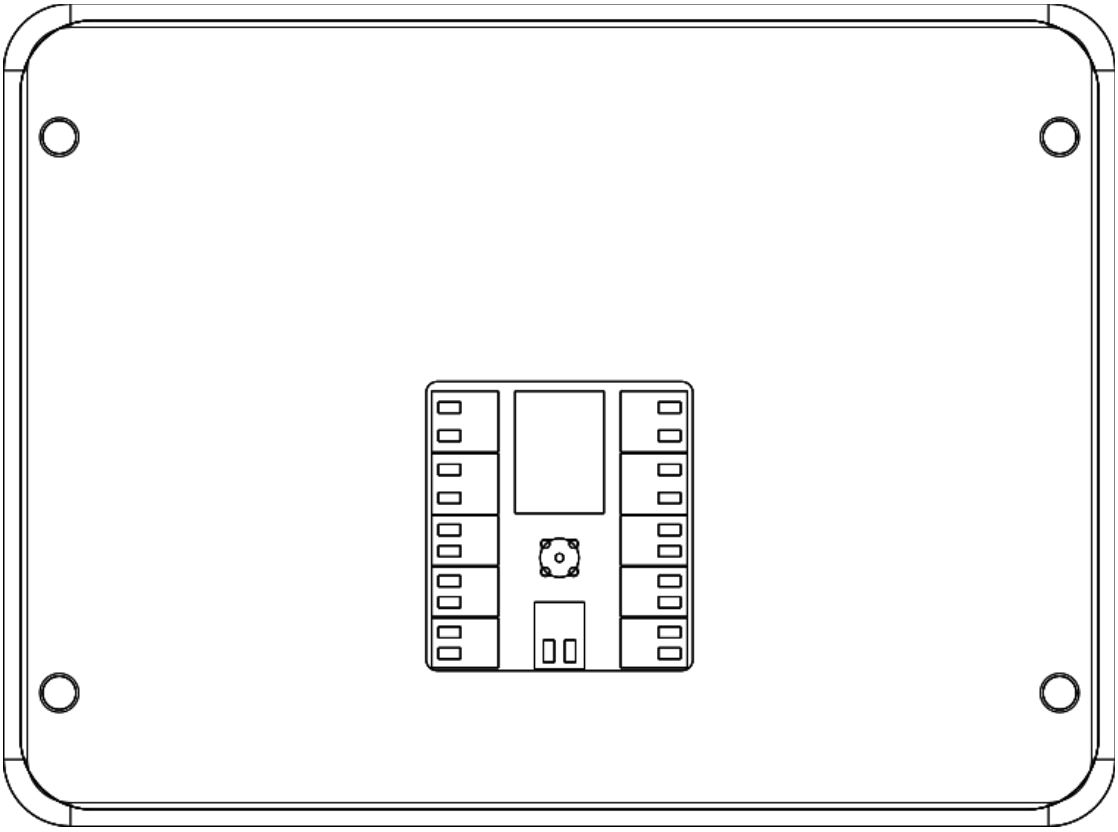


Рис.1.2a. Оборотная сторона ВП. Монтажные отверстия и отсек с клеммными соединителями.

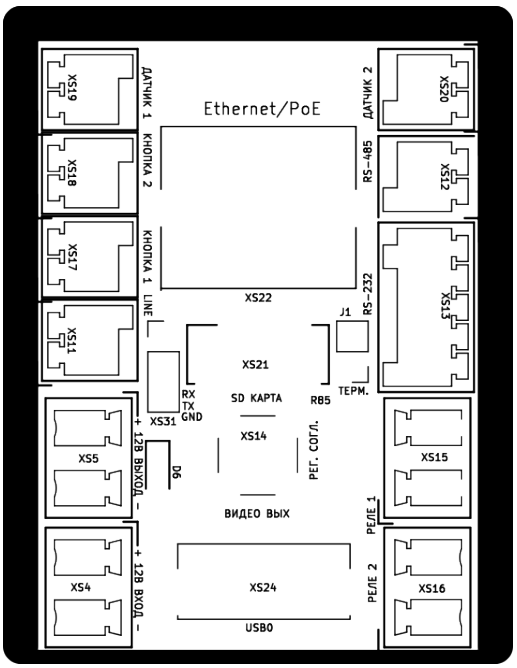


Рис. 1.2b. Расположение разъемов в отсеке и их обозначения для версии 2.x.

Таблица 1.2. Назначение разъемов.

Обозначение на плате	**Назначение и распиновка разъема**
XS31	Последовательный порт консоли. Режим порта 115200 8N1. Назначение пинов: 1. RXD 2. TXD 3. GND Уровень напряжений: +3.3В
XS22	Сеть FastEthernet (RJ-45) с функцией PoE 802.3af.
XS21	Разъём SD-карты.
XS19	Датчик состояния двери 1. Замыкание контактов друг на друга — дверь закрыта, размыкание — дверь открыта.
XS20	Датчик состояния двери 2. Замыкание контактов друг на друга — дверь закрыта, размыкание — дверь открыта.
XS17	Кнопка 1 отпирания замка. Замыкание контактов друг друга приводит к отпиранию замка.
XS18	Кнопка 2 отпирания замка. Замыкание контактов друг друга приводит к отпиранию замка.

XS12	Интерфейс RS-485/Modbus для подключения коммутатора и иной периферии. Назначение пинов: 1. линия A 2. линия B Скорость в порту и формат фрейма: 115200 8N1
XS13	Последовательный порт RS-232 для подключения периферии. Назначение пинов: 1. GND 2. RXD 3. TXD 4. Touch memory 5. +12B
XS5	Выход +12В для питания внешних устройств током до 1А. Назначение пинов: 1. +12В выход. 2. GND
XS4	Вход внешнего питания 12-14В. Потребляемый ток до 1А. Назначение пинов: 1. +12В вход. 2. GND
XS11	Аналоговая звуковая линия (LN) для подключения аналогового коммутатора. Назначение пинов: 1. GND 2. LN
XS15	Опто-реле отпирания двери 1.
XS16	Опто-реле отпирания двери 2.
XS24	USB type A (USB0)
XS14	Выход аналогового TV-сигнала (при наличии установленного PAL кодера).
R85	Потенциометр согласования аналогового шлейфа.

1.4. Руководство конечного пользователя

Изделие Вызывная Панель «ЕРМАК», функционируя в составе систем офисной или домашней системы видеосвязи, может одновременно выступать как в роли вызывного, так и ответного устройства. В «вызывном» режиме

изделие функционирует как обычный домофон или видеотелефон, позволяя пользователю осуществлять вызов удаленного абонента (инициировать исходящий SIP звонок) путем набора номера помещения (квартиры, офиса) или путем нажатия запрограммированной на-экранной клавиши с нанесенной на неё текстовой информацией о вызываемом абоненте. В «ответный» режим Панель переходит при поступлении входного SIP звонка на заданный в конфигурационном файле SIP URI. В «ответном» режиме Панель позволяет пользователю наблюдать видео-поток от удаленного абонента еще до момента соединения, позволяет принять соединение, отклонить соединение, игнорировать входной звонок (оставить без ответа) или послать команду удаленному устройство на отпирание двери.

Режим работы изделия, количество на-экранных клавиш, их обозначение и расположение на экране определяется настройками в конфигурационном файле и задается системным администратором, либо сервером при функционировании Панели в составе сложной сети.

В пассивном состоянии Панель находится в состоянии энергосбережения, погасив экран до 10% максимальной яркости (регулируется в конфигурационном файле). При обнаружении человека на расстоянии 60-90 см, включается полная подсветка экрана, сигнализируя о готовности к работе. Независимо от состояния подсветки и местонахождения человека происходит непрерывный показ информационных (рекламных) слайдов или объявлений, подаваемый в виде сменяющихся картинок. Их список, время просмотра, способ замены, также задается в конфигурационном файле и может загружаться с сервера.

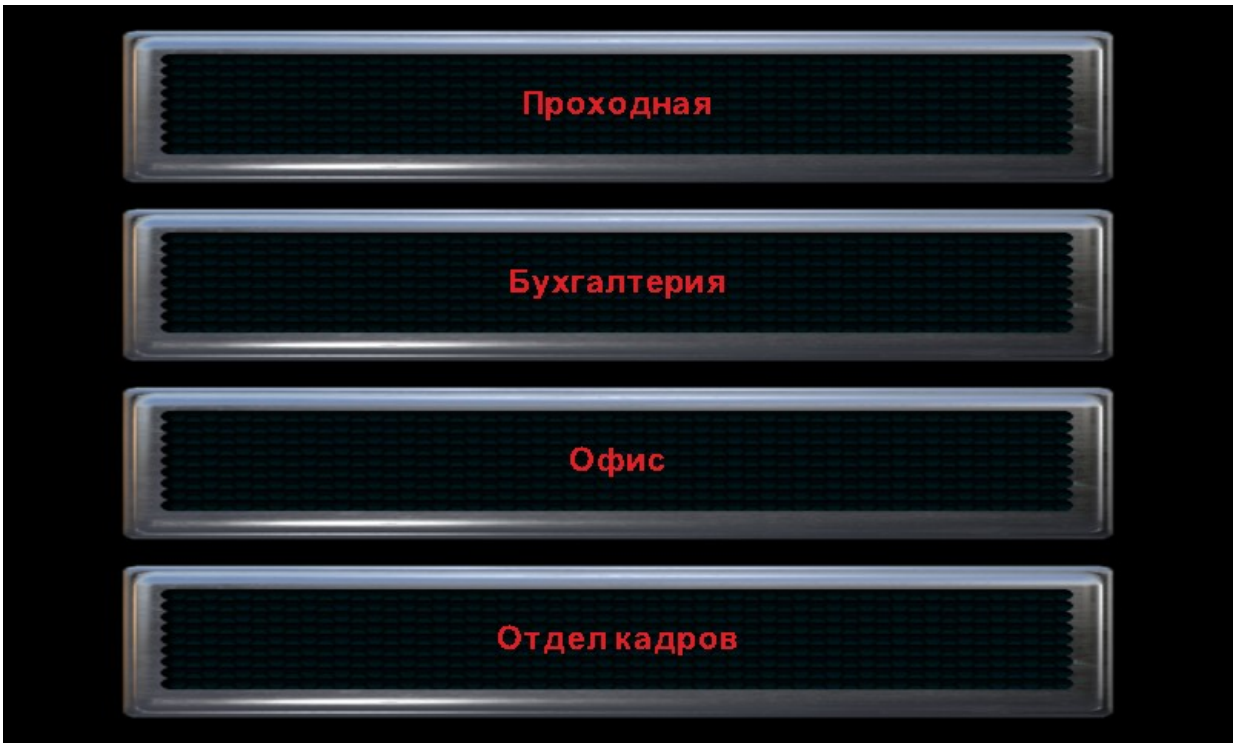
Подошедший человек может нажать кнопку SOS («экстренный вызов») путем удержания пальца возле датчика присутствия более 5 секунд. В этом случае Панель инициирует исходящий звонок на специально запрограммированный SIP URI, задаваемый в конфигурационном файле для зарезервированного номера помещения 9999112.

Нажатие в любое место сенсорного экрана вызывает переход в один из трех вариантов меню. В какое именно меню переходить, задано в конфигурационном файле в параметре ***BaseMenu*** секции ***GUI***.

Вариант 1 показывает меню набора номера помещения с цифровой на-экранной клавиатурой.

Этот вариант предназначен для эксплуатации в зданиях с большим числом офисных помещений или МКД.

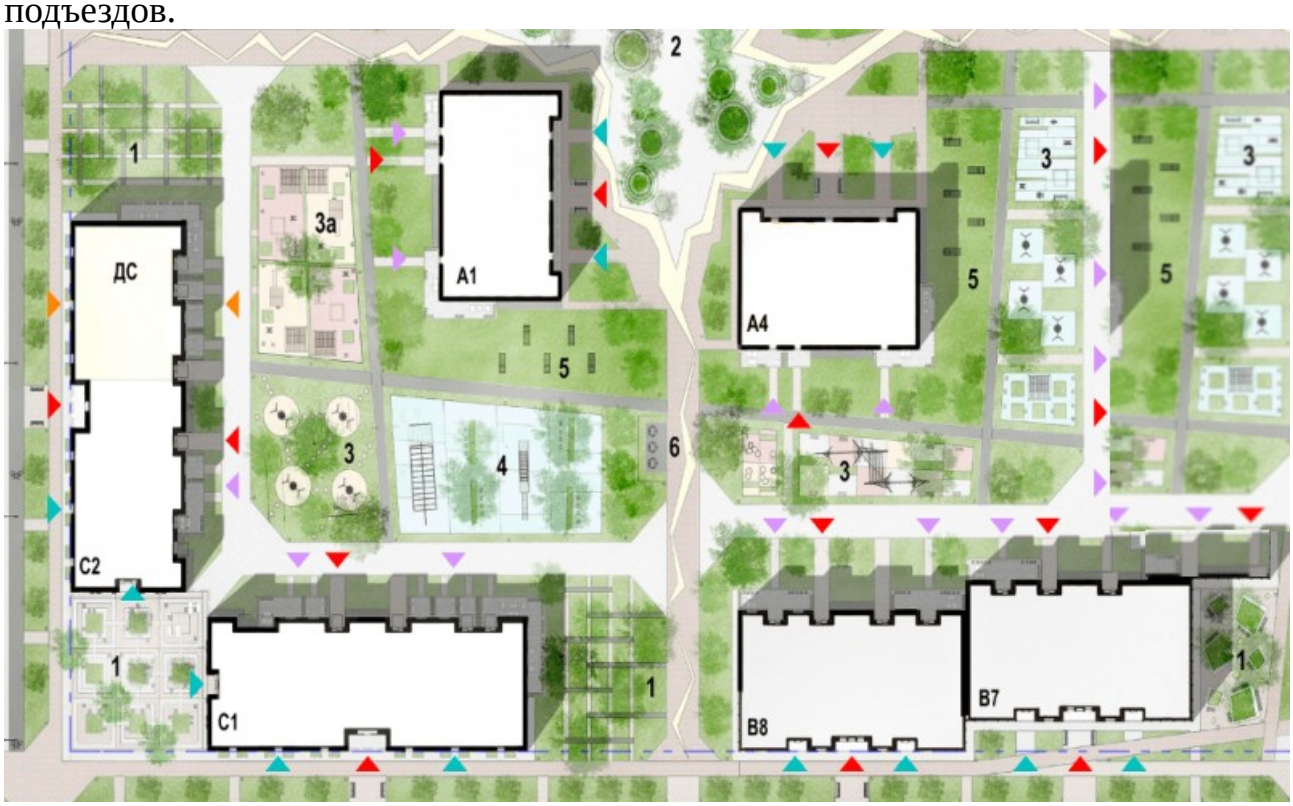
Вариант 2 дает возможность инициировать вызов поименованных помещений путем нажатия запрограммированных на-экранных кнопок:



Такой вариант подходит для зданий с небольшим числом офисных или жилых помещений, список которых может уместиться на экране Вызывной

Панели. На кнопках отображаются самые первые элементы из раздела ***Apartments*** конфигурационного файла. Для поименованных помещений предоставляется возможность задать произвольный текст, а также имя и размер шрифта, используемого при отображении этого текста на клавишах.

Вариант 3 предназначен специально для вызывной панели, установленной на калитке (входе) двора, в котором расположено сразу несколько зданий или подъездов.



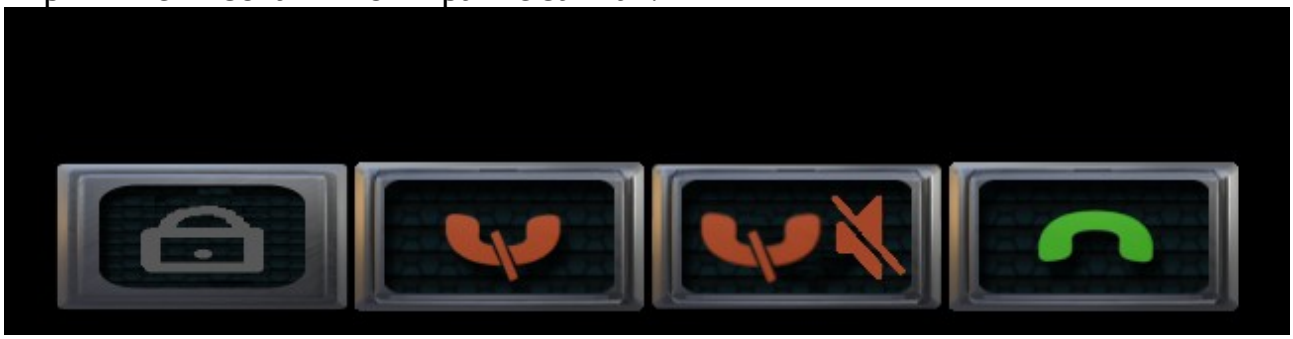
Выбрав одно из зданий, посетитель попадает в меню аналогичное варианту 1, для ввода номера помещения.

Для каждого помещения (поименованного или нумерованного) может быть задано одно и более направление для размещения исходящего вызова, таким образом, набор номера или нажатие клавиши, производит одновременный вызов по всем направлениям, указанным в списке доступных маршрутов для данного помещения. Вызов может быть SIP-звонком на любой SIP URI, на другую аналогичную Панель, установленную в квартире или офисе, или на аналоговое переговорное устройство (абонентская трубка). Соединение устанавливается с первым ответившим абонентом, с «неответившими» абонентами связь при этом разрывается (возможность «конференционных» звонков отсутствует).

Если вызываемая сторона поддерживает видео, то вызывная панель сразу начинает проигрывание входящего видеопотока, и пересылку ответного видеопотока со встроенной видеокамеры.

Появление RFID метки (ключа) возле считывателя в любой момент времени, вне зависимости от состояния Панели, вызывает запрограммированные события: открывание замка, проигрывание специально выбранного для данного RFID звукового оповещения или звукового оповещения «авария», если RFID метка не распознана.

Как отмечалось выше, Панель может принимать входящие SIP-вызовы. При получении такого вызова Панель переходит в режим «ответа». На экране отображается видео-картинка вызывающего абонента (если присутствует видеосигнал), и четыре пиктограммные клавиши: «отбой», «игнорирование», «принятие вызова» и «отпирание замка».



Клавиша «отбой» отбивает запрос и возвращает пользователя на главный экран.

Клавиша «игнорирование» — также возвращает на главный экран, но не делает отбой входящего звонка, оставляя вызывающего на линии в пред-ответном состоянии.

Клавиша «принятие» вызова включает двухстороннее аудио-соединение. В этом режиме оба абонента могут вести разговор. Если вызывающая сторона поддерживает видео, то абонент на принимающей стороне может, по желанию, нажать значок камеры (в правой нижней части экрана) для передачи видеосигнала со встроенной видеокамеры Панели.

Для окончания разговора нужно нажать кнопку «отбой» (в левой нижней части экрана).

Нажатие клавиши «отпирание замка» приводит к передаче на удаленное устройство команды на отпирание дверного замка (передача DTMF последовательности заданной в секции ***Common*** конфигурационного файла).

Нажатие клавиши «отпирание замка» возможно без установки соединения, т.е. в пред-ответном состоянии. Также имеется функция автоматического разъединения после нажатия клавиши «отпирание замка» (настраивается в конфигурационном файле), что избавляет пользователя от одного лишнего действия.

2. Руководство системного администратора

2.1. Установка и подключение ВП

Вызывная Панель «ЕРМАК» версии 2.х предназначена для монтажа на панели врезным методом, для чего изделие имеет на оборотной стороне четыре крепежных механизма (кронштейна) каждый их которых фиксируется винтом М4 и прижимается к монтажной панели винтом М5. Изделие может быть установлено без специальных приспособлений на входную металлическую дверь, а с кронштейном (поставляется отдельно) на стену или опору. При установке без кронштейна необходимо учитывать расположение коммуникационных кабелей, которые должны быть скрыты от посетителей, для чего рекомендуется устанавливать вызывную панель на неподвижную створку входной двери, предварительно проделав небольшой вырез по форме отсека с разъемами. Для облегчения монтажа в «*Приложение 1. Габаритный чертеж ВП*» приведен чертеж изделия с габаритными размерами и расположением разъемов.

После монтажа к вызывной панели требуется подвести и подключить коммуникации. Минимально достаточным является подключение к сети FastEthernet с питанием PoE согласно 802.3af. Другим способом питания изделия является внешний источник стабилизированного напряжения номиналом 12В, подключение которого осуществляется через клеммный разъем XS4 (см. таб. 1.2). Опционально к Вызывной Панели могут быть подключены одна или две кнопки отпирания двери (XS17 и XS18), один или два электромагнитных замка через встроенные опто-реле (клеммы XS15 и XS16), а также аналоговый коммутатор (клеммы XS12 - сигнальный и XS11 - звуковой). Процедура установки, настройки и подключения коммутатора приведена в разделе «*2.3 Подключение аналогового коммутатора*». Полная схема всех коммуникаций представлена на рис. 2.1. Описание назначения выводов в каждом разъеме дано в таблице 1.2.

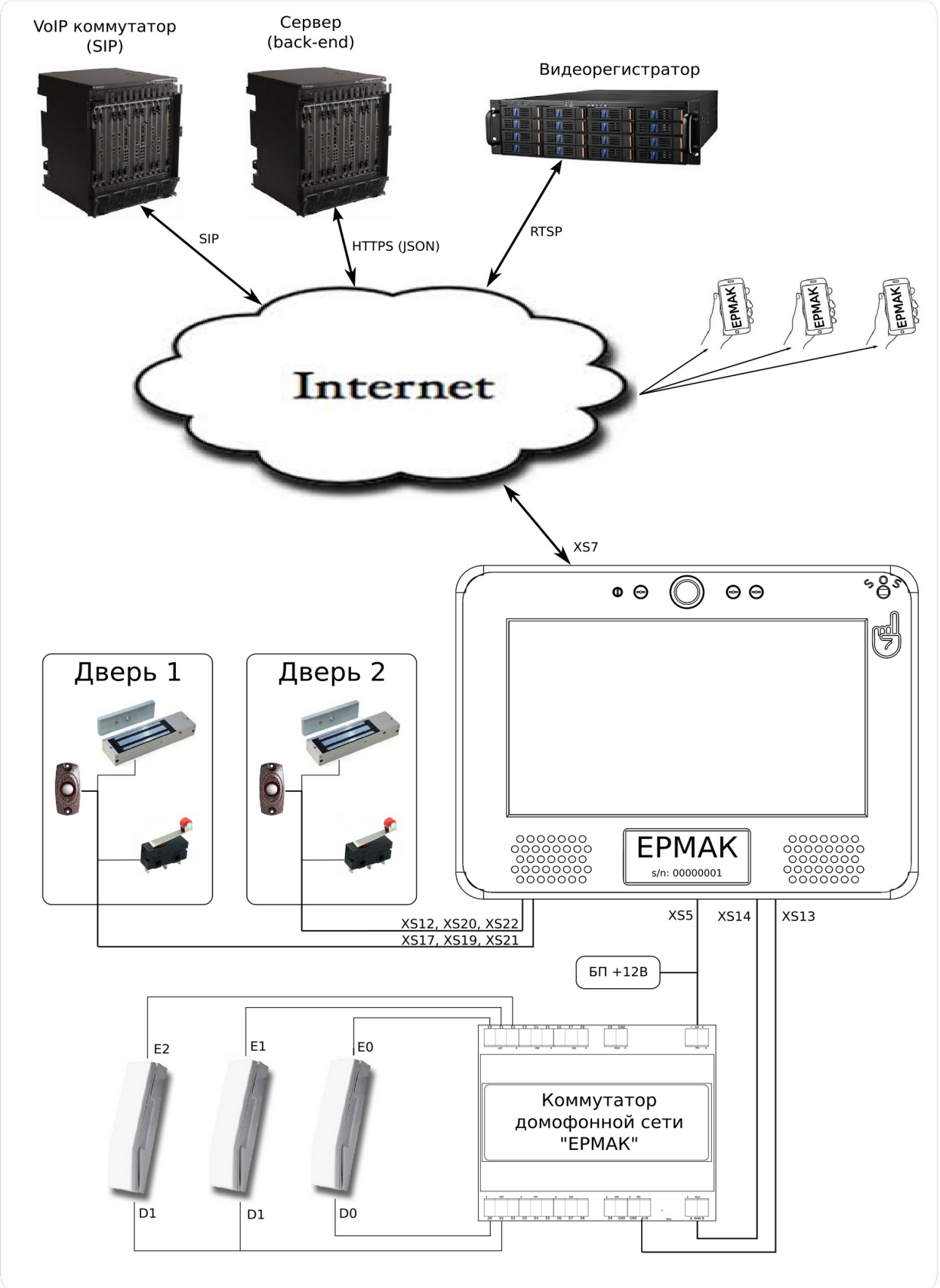


Рис. 2.1. Типовая схема домофонной сети «EPMAK».

2.1.1. Подключение кнопок отпирания и датчиков положения дверей

К Вызывной Панели могут быть подключены до четырех датчиков вида «сухой контакт» которые работают на замыкание или размыкание контакта. Два из этих датчиков обозначены как «Кнопка 1» и «Кнопка 2» (XS17 и XS18), которые предназначены для подключения кнопок отпирания замков. По-умолчанию, ВП настроена таким образом, что «Кнопка 1» замыкает опто-реле на порту XS15, а «Кнопка 2» - опто-реле на порту SX16. При срабатывании кнопки, соответствующее опто-реле будет замкнуто на интервал времени 5 секунд, по истечению которого будет обратно разомкнуто. Данное поведение кнопок и опто-реле может быть перенастроено в конфигурационной секции «DoorsAndButtons».

Еще два датчика обозначенных как «Датчик 1» и «Датчик 2» (XS19 и XS20 соответственно) могут быть использованы для контроля состояния дверей - «дверь закрыта» или «дверь открыта», посредством подключения индуктивного датчика положения. В случае необходимости, оба порта датчиков могут быт перенастроены для подключения дополнительных кнопок отпирания дверей. Схемы подключения кнопок и датчиков приведены ниже на рис. 2.1.1.

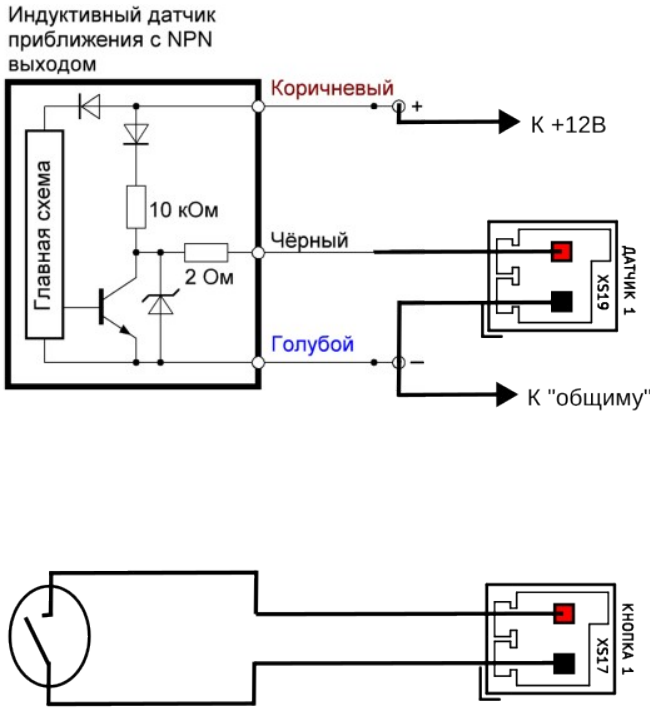


Рис. 2.1.1. Схема подключения кнопки и индуктивного датчика: вверху — с использованием индукционного датчика, внизу — обычной кнопкой или микровыключателем.

Перед подключениям кнопок или датчиков необходимо убедиться в том, что порты находятся в рабочем состоянии. Для этого можно воспользоваться любым прибором измерения постоянного напряжения (вольтметр или мультиметр). Переведите мультиметр в режим измерения напряжения до 20В (постоянный ток — DC) и установите щупы на клемы порта кнопки на включенной и загруженной Вызывной Панели. Мультиметр должен показывать напряжение +3,3В, что соответствует рабочему состоянию порта. Если данное напряжение не измеряется или его значения существенно ниже, то данный порт считается нерабочим и требуется осуществить ремонт ВП.

2.1.2. Подключение электромагнитного замка

Непосредственно к Вызывной Панели могут быть подключены до двух электромагнитных замков, для чего ВП снабжена двумя опто-реле выходы которых выведены на разъемы XS15 обозначенный как «РЕЛЕ 1» и XS16 - «РЕЛЕ 2» соответственно. Напряжение коммутации опто-реле не должно превышать 60В и **пиковый ток — 1,2 А**. В виду этих ограничений, при подключении мощного электромагнитного замка рекомендуется использовать внешние электромагнитные реле. На изображении рис. 2.1.2 приведены типовые схемы подключения электромагнитного замка.

ВАЖНО: Для гашения обратного выброса тока в схеме обязательно должен быть использован диод Шоттки обеспечивающий пропускания тока силой минимум в 2,5 раза больше чем сила тока удержания электромагнитного реле.

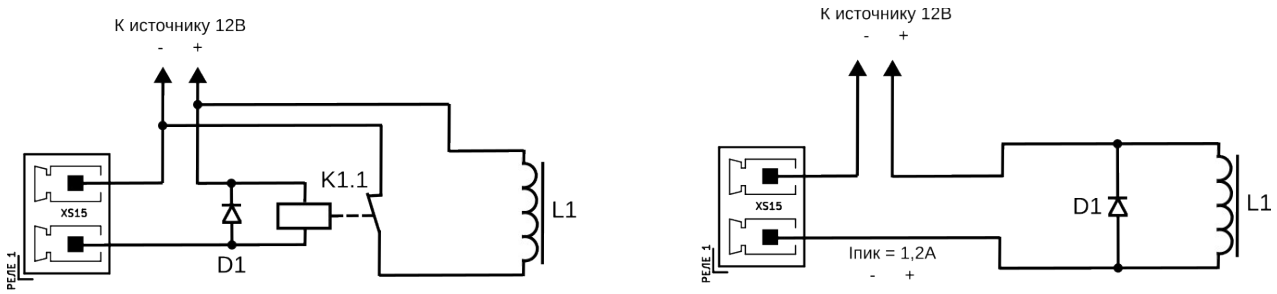


Рис. 2.1.2. Схема подключения Э/М замка: слева — через дополнительное э/м реле, справа — напрямую.

По-умолчанию, оба опто-реле настроены и работаю в состояние «нормаль-разомкнуто», т. е. реле постоянно находится в разомкнутом состоянии и замыкаются кратковременно при подаче управляющего воздействия. Данное поведение можно изменить на противоположное («нормально-замкнуто») путем

установки типа замка на «inversed» в конфигурационной секции «LockManager».

Перед подключением замка имеет смысл убедиться в том, что порты XS15 и XS16 настроены и находится в рабочем состоянии. Для этого необходимо установить щупы мультиметра на клеммы в разъема, перевести мультиметр в режим «прозвонки» и инициировать отпирание соответствующего замка кнопкой или RFID ключом. В момент отпирания мультиметр должен издавать звуковой сигнал в течении 5 секунд, что сигнализирует о настроенном и рабочем состоянии порта. Если звуковой сигнал отсутствует, то рекомендуется попробовать изменить полярность подключения щупов. Если ни в одном из положении звукового сигнала от мультиметра не наблюдается, при этом слышно голосовое оповещение от ВП, то необходимо проверить настройки замков в конфигурационной секции «LockManager», а так же настройки кнопок в секции «DoorsAndButtons» и правильно установить соответствие между кнопками и замками.

2.2. Первоначальная настройка ВП

2.2.1. Конфигурационный спул и консолидированный конфиг

Вызывная Панель «ЕРМАК» снабжена встроенным носителем информации типа NAND, подключенного по интерфейсу eMMC, на котором, помимо операционной системы Linux и программного обеспечения ВП, сохраняется конфигурация всех **модулей** вызывной панели в едином файле, представляющем собой JSON структуру — консолидированный конфигурационный файл. Программное обеспечение вызывной панели позволяет сохранять несколько различных конфигурационных файлов одновременно, предоставляя пользователю возможность выбора активного «конфига». Все конфиги располагаются в т.н. каталоге - «конфиг спуле», который расположен в файловой системе в подкаталоге **/var/configs**. На активный в данный момент конфиг создается временная символическая ссылка **/var/configs/all.json**. Эта ссылка создается модулем **Configurator** в момент загрузки ПО. Все конфиги, находящиеся в спуле, имеют уникальный номер серии (serial) — целое число, которое используется модулем Configurator для поиска самого последнего (самого свежего) конфига. Конфиг, серийный номер которого самый большой из имеющихся на данный момент в спуле, считается самым свежим, на него и создается символическая ссылка, которая далее используется остальными модулями ПО для загрузки своих конфигурационных данных.

Конфиг состоит из секций, по имени каждого модуля, которые содержат конфигурационные параметры для соответствующего модуля или подсистемы программного обеспечения ВП. Структура (JSON схема) каждой секции конфига приведена в «*Приложении 2*».

Программное обеспечение ВП содержит модуль WebAdmin, который позволяет производить манипуляции с конфигами через WEB интерфейс, в том числе имеются функции для редактирования различных параметров в секциях конфига, скачивания и загрузки конфига на ВП из локального файла, удаление устаревшего или некорректного конфига из спула.

2.2.2. Провизия и обеспечение настройками

Заводские настройки вызывной панели предполагают, что вызывная панель подключается сети FastEthernet с функционирующим сервисом Dynamic Host Configuration Protocol (DHCP), который должен обеспечить устройство по его

MAC-адресу динамическим IP-адресом и ссылкой (URL) на FTP или HTTP/HTTPS сервер, обеспечивающий вызывную панель конфигурационным файлом. Ссылка на конфигурационный файл задается DHCP параметром:

option siphomephone-config code 161 = string;

Данный параметр должен содержать URL сервера, предоставляющего конфигурационный файл. Если URL начинается со строки **ftp://** то используется анонимный FTP протокол для скачивания файла. В этом случае, при запросе файла, в имя файла подставляется уникальный **SID** вызывной панели, что позволяет хранить настройки нескольких вызывных панелей в одном и том же каталоге FTP сервера, не меняя при этом настройки DHCP.

Если URL начинается со строки **http://** или **https://** то используется специализированный HTTP API для запроса конфигурационного файла. Данный API предполагает прохождение процедуры авторизации и получения **авторизационной куки**. В процессе авторизации используется уникальный SID вызывной панели, а также пароль, заданный в файле **/var/configs/password**. Узнать SID вызывной панели, а также изменить авторизационный пароль можно через встроенный WEB интерфейс.

Если при загрузке ВП в момент инициализации сетевого интерфейса сервис DHCP по каким-либо причинам не обеспечил устройство IP-адресом, то на сетевом интерфейсе вызывной панели устанавливается статический **IP: 192.168.0.50 netmask 255.255.255.0.** При этом, запрос конфигурационного файла не осуществляется, а используется последний полученный конфиг. Если такового получено ранее не было (первое включение устройства), то используется конфигурационный файл с настройками по умолчанию: **/home/fabmicro/SIPHomePhone/all.json.default**.

2.2.3. Просмотр текущих сетевых настроек и SID

Для того, чтобы оперативно выяснить текущие сетевые настройки (IP-адрес и т.д.), а также узнать SID вызывной панели, в графическом интерфейсе предусмотрено информационное окно «**About**». Для отображения этого окна необходимо дождаться появления на экране ВП заставки, после чего последовательно кликнуть по одному разу в четырех углах сенсорного экрана согласно следующей последовательности: *левый верхний -> правый нижний -> правый верхний -> левый нижний*.

Окно «**About**» отобразится на 30 секунд, по истечению которых Вызывная Панель вернется в рабочий режим.

2.2.4. Использование WEB-интерфейса для задания начальных настроек

Как было сказано, в состав программного обеспечения ВП входит отдельный модуль **WebAdmin,** обепечивающий WEB-интерфейс администратора, для доступа к которому требуется использовать WEB-браузер с включенной поддержкой JavaScript.

Для доступа к WEB-интерфейсу вызывной панели используется HTTP протокол (HTTPS не поддерживается) с Web-авторизацией, по умолчанию используются следующие реквизиты для доступа:

URL: http://192.168.0.50/

Username: admin

Password: password

ВАЖНО: После проведения настроек, для обеспечения безопасности, модуль WebAdmin рекомендуется отключить. Для этого в конфиге в секции **SunBus** в подразделе модуля **WebAdmin** установить параметр **disabled** в значение **true**.

WEB-интерфейс предоставляет администратору следующие возможности, распределенные по системе меню:

Главная страница

Отображает следующие параметры:

- Текущий SID вызывной панели (строка SID длиной 32 символа) — используется как уникальный идентификатор изделия при авторизации на сервере.
- Серийный номер (Serial) текущего конфига — целое число, обычно отображающее UNIX time момента создания конфига.
- Источник текущего конфига (Source) — адрес сервера, webadmin или default_config, в зависимости от того откуда прибыл данный конфиг.

Меню [Сетевые настройки]

Позволяет просмотреть и задать сетевые настройки вызывной панели, в том числе:

- Просмотреть сетевые настройки - MAC, IP-адрес, netmask, gateway.
- Установить IP адрес/маску/шлюз/DHCP - IP-адреса, netmask, gateway, включение/выключение поддержки DHCP.
- Просмотреть DHCP lease - текущий DHCP lease и его параметров.
- Изменить имя пользователя и пароль для Web-интерфейса.
- Изменить адрес сервера конфигурации и пароль к нему.

Если параметр config_url не установлен, то запрос конфига не производится, а используется конфиг из спула с самым большим серийным номером. Если спул пуст, то используется конфиг по умолчанию с заводскими настройками.

Меню [Список конфигурационных разделов]

Позволяет просматривать и редактировать настройки модулей и подсистем в соответствующих секциях конфига, в том числе:

- Advertizer — секция конфига описывающая набор рекламных и/или информационных объявлений, отображаемых на экране вызывной панели. Каждое объявление - это ссылки (URL) с изображением PNG 800x480px.
- AmbientSensor — секция конфига, содержащая настройки модуля, обеспечивающего взаимодействие с датчиком освещенности. Задает диапазон считываемых значений АЦП для двух состояний: «день» и «ночь».
- Apartments — секция конфига, содержащая перечень помещений, их идентификаторы, имена владельцев, списки RFID ключей и таблицы маршрутизации вызовов (через SIP, через коммутатора, и т.д.).
- Common — секция конфига, содержащая настройки общие для различных подсистем.
- DoorsAndButtons — секция конфига, содержащая настройки модуля обслуживающего кнопки отпирания замков и датчики положения дверей.

- GUI — секция конфига, содержащая настройки модуля, обеспечивающего графический интерфейс пользователя. Задает тип главного меню вызывной панели и её амплуа. Более подробно см. раздел «*2.2.5. Настройка графического интерфейса пользователя*».
- HandsetState — секция конфига, содержащая настройки модуля, отслеживающего состояние аналоговой домофонной трубки. Позволяет задавать диапазон считываемых значений АЦП, отражающий положение трубки: «трубка висит», «трубка снята», «трубка снята и нажата кнопка отпирания замка».

  ВАЖНО: Данный модуль используется только в том случае, если аналоговая домофонная трубка подключена непосредственно к вызывной панели на порт XS11, минуя коммутатор. При использовании домофонного коммутатора, отслеживание состояние трубок производится через него модулем **RS485Master**.
- LockManager — секция конфига, содержащая настройки модуля, взаимодействующего с опто-реле управления ЭМ-замками.
- Notify — секция конифига, содержащая настройки для подсистемы оповещения о состоянии вызывной панели на сервер.
- ProximitySensor — секция конфига, содержащая настройки для модуля взаимодействующего с «датчиком присутствия», в том числе задает диапазоны значений показания датчика для трех состояний: approaching — к домофону приближается посетитель, dialing — посетитель взаимодействует с домофоном (набирает номер), fingering — посетитель поднес палец к датчику на близкое расстояние (экстренный вызов).

  ВАЖНО: задание данных параметров производится индивидуально для каждой вызывной панели после установки её по месту эксплуатации, так как показания датчика «присутствия» могут сильно разниться в зависимости от условий эксплуатации и присутствия объектов, вызывающих блики и переотражения сигнала.
- RFIDReader — секция конфига, содержащая настройки для модуля, взаимодействующего с аппаратным считывателем RFID меток.

- RS485Master — секция конфига для модуля, взаимодействующего с домофонным коммутатором и другими устройствами на шине RS-485/Modbus.
- Yard — секция конфига, описывающая конфигурацию «двора»: перечень зданий, доступные номера помещений в них и ссылка (URL) на файлы с графическим изображением двора и зданий (PNG 800x480px).

Подробное описание всех параметров и их доступных значений для каждой из конфигурационной секций следует искать в соответствующей JSON схеме. Описание всех JSON схем дано в «*Приложении 2. Структура конфигурационного файла*».

ВАЖНО: При изменении параметров в секциях конфигурационного файла требуется производить перезапуск всех модулей после каждой записи (сохранения) настроек, см. меню «*Системные операции*» ниже. Перезапуск модулей может занимать до 10 сек, в течение этого времени WEB-интерфейс может быть недоступен. Такое поведение ПО вызывной панели обусловлено необходимостью верифицировать сохраненный консолидированный конфиг всеми модулями.

Меню [Системные операции]

Позволяет производить ряд действий с вызывной панелью необходимых при настройке, пуско-наладке, в том числе:

- Просмотреть конфигурацинный спул
- Просмотреть конфигурационный файл
- Скачать конфигурационный файл
- Загрузить конфигурационный файл
- Просмотреть медиа спул
- Просмотреть параметры HTTP запроса
- Монитор системных событий
- Диагностика устройств
- Перезапусить все модули
- Перезагрузить систему
- Выключить питание

Меню [Показания датчиков]

Позволяет отследить текущие показания АЦП датчиков вызывной панели с целью подбора правильных рабочих значений для соответствующих модулей. В том числе доступны следующие датчики:

- Датчик присутствия
- Датчик освещенности
- Датчик напряжения в аналоговой линии
- Датчик положения дверей и кнопок

2.2.5. Настройка графического интерфейса пользователя

Графический интерфейс пользователя (GUI) служит для отображения информации на экран Вызывной Панели и обеспечивает ввод с сенсорного экрана, таким образом организовывая систему «меню» для взаимодействия с пользователем. В любой момент времени GUI отображает определенную картинку и обрабатывает скрипты ассоциированные с ней (далее по тексту этого документа каждую такую картинку будем называть просто — меню). Все меню описываются в файлах представляющих JSON структуру, эти файлы расположенных в папке ***menu/*** относительно рабочего каталога. Подробное описание файлов меню находится в разделе *3.5.3. Структура файлов в каталоге menu/.*

Настройка графического интерфейса пользователя производится заданием параметров в секции GUI консолидированного конфигурационного файла. Осуществить настройку графического интерфейса можно как через WEB интерфейс администратора, так и вручную, отредактировав JSON файл конфига и загрузив его на вызывную панель.

Краткое описание самых необходимых параметров:

- *"BackgroundColor":* цвет фона для всех меню (формат A8R8G8B8), например белый цвет в этом формате будет задан как 0xffffffff, красный - 0xffff0000.
- *"BaseMenu":* следующее меню, в которое ВП переходит из стартового. В данный момент существует несколько вариантов для этого меню, описание которых приведено в *1.4. Руководство конечного пользователя.* Параметр "BaseMenu" может принимать одно из следующих значений:

1. **"keyboard"** — задает меню набора номера помещения с цифровой на-экранной клавиатурой.
2. **"buttons_1"** — задает меню с одной запрограммированной кнопкой. Текст и тип шрифта для данной кнопки берется из первого элемента списка помещений (см. секцию *"**Appartments**")* из параметров *"descr"* и *"descr_font"*.
3. **"buttons_2"** — задает меню с двумя кнопками. Текст и тип шрифта для данных кнопок берется из первых четырех элементов в *"Appartments"* из параметров *"descr"* и *"descr_font"*. Подробнее, об *"Appartments"*, смотри в *2.2.7. Настройка «помещений»*.
4. **"buttons_4"** и **"buttons_8" —** аналогично меню **"buttons_2"** для меню с 4 и 8 запрограммированными на-экранными кнопками.
5. **"yard" —** данное меню предназначено для вызывной панели, установленной на входе во двор, в котором расположено сразу несколько зданий, либо одно здание с множеством подъездов. Данное меню позволяет выбрать одно из зданий двора и перейти к набору номера помещения для конкретно этого здания/подъезда.

- *"BrightnessDialing"* - яркость дисплея в режиме **dialing** (активный режим, посетитель перед вызывной панелью). Значения: 1 — 100 , значение 0 — дисплей погашен, 100 — максимальная яркость.
- *"BrightnessStandBye"* - яркость дисплея в режиме **stand-bye** (никого нет). Значения: 0 — 100.
- *"BrightnessApproaching"* - яркость дисплея в режиме **approaching** (вдали кто-то есть). Значения: 1 — 100.
- *"DelayBeforeStandBye"* - время неактивности, перед переключением в режим **stand-bye** (в секундах).
- *"TimeoutShowPicture"* - время по умолчанию, отображения картинки командой **show_picture** (в секундах). Подробнее о командах смотри в раздел *3.1.1 Набор команд.*
- *"TimeoutKeyboardMenu"* - время неактивности, перед возвратом из меню набора номера определенного в **keyboard.json** (в секундах).
- *"TimeoutButtonsMenu"* - время неактивности, перед возвратом из меню определенного в **buttons_N.json** (в секундах).

С целью задавать групповые настройки сразу нескольких параметров у элементов меню существует система алиасов. Определение параметров алиасов производится в подсекции **"Aliases"** секции **"GUI"** конфигурационного файла. На данный момент в системе алиасов доступны только **True Type** шрифты. Изменяя параметры у алиаса, можно поменять цвета и размеры сразу у всех текстовых надписей в интерфейсе пользователя.

ВАЖНО: При изменении параметров алиасов требуется обращать внимание на все файлы меню, в которых данный (изменяемый) алиас используется.

На данный момент предопределены алиасы FONT1 — FONT5. Для каждого из них можно отредактировать следующие параметры:

- *"ani":* имя графического файла для отображения. Для шрифтов это обычно файл с расширением .ttf содержащий True Type шрифт. По умолчанию он ищется в подкаталоге **images/** рабочего каталога.
- *"font_size":* размер символа в пикселях.
- *"color":* цвет символа в формате A8R8G8B8.

2.2.6. Настройка «двора»

Если для параметра *"BaseMenu"* установлено значение **"yard",** то Вызывная Панель будет функционировать в составе «двора» из нескольких зданий. На экране будут отображены эти здания и переход в меню набора номера помещения осуществится после выбора одного из зданий этого двора.

Параметры для описания двора находятся в секции **"Yard"** конфигурационного файла. Для описания двора задаются следующие параметры:

- *"allbuildings_url":* ссылка на фоновую картинку, заполняющую весь экран вызывной панели. Картинка должна содержать все здания, которые можно будет выбрать для двора. Пример такой картинки приведен на рис. 2.2.
- *"Buildings"*: подсекция описывающая отдельные здания:
  1. ***"name"*: имя элемента в конфигурационном файле yard.json. На данный момент предопределены следующие имена: "building1" - "building9",**

2. ***"aptnum_prefix"*: префикс для данного здания, который будет использоваться для поиска помещения в секции "Apartments". Указанный префикс будет подставляться перед введенным номером помещения, перед поиском элемента *"apt_num"*.** Например, если в префиксе указано "1#" и введен номер помещения 99, то будет осуществлен поиск элемента *"apt_num"* == "1#99".
3. **"url": ссылка на картинку данного здания. Пиксельные размеры этой картинки должны совпадать с размерами картинки, заданной в *allbuildings_url* (рекомендуется использовать размеры экрана: 800x480). В данной картинке требуемое здание должно находится на непрозрачном и черным фоне. Это необходимо для того, чтобы можно было определить область срабатывания нажатия на сенсорный экран при выборе данного здания. Здание можно изобразить немного бо́льшего размера или со смещением относительно того же здания на общей картинке, это приведет к эффекту анимации нажатия кнопки. Пример такой картинки приведен на рис. 2.3.**
4. ***"address":* географический адрес здания или иное наименование объекта. Данный текст будет отображаться в меню набора номера помещения.**
5. ***"min_aptnum":* минимальный номер помещения, разрешенный к набору пользователем, целое число от 0 до 9999.**
6. ***"max_aptnum":* максимальный номер помещения, разрешенный к набору пользователем, целое число от 0 до 9999.**

Все задаваемые в этой секции ссылки (URL) могут быть ссылками на ресурсы в сети (типа **http://**, **https://** или **ftp://**), либо на файл в локальной файловой системе вызывной панели (**file:///**).

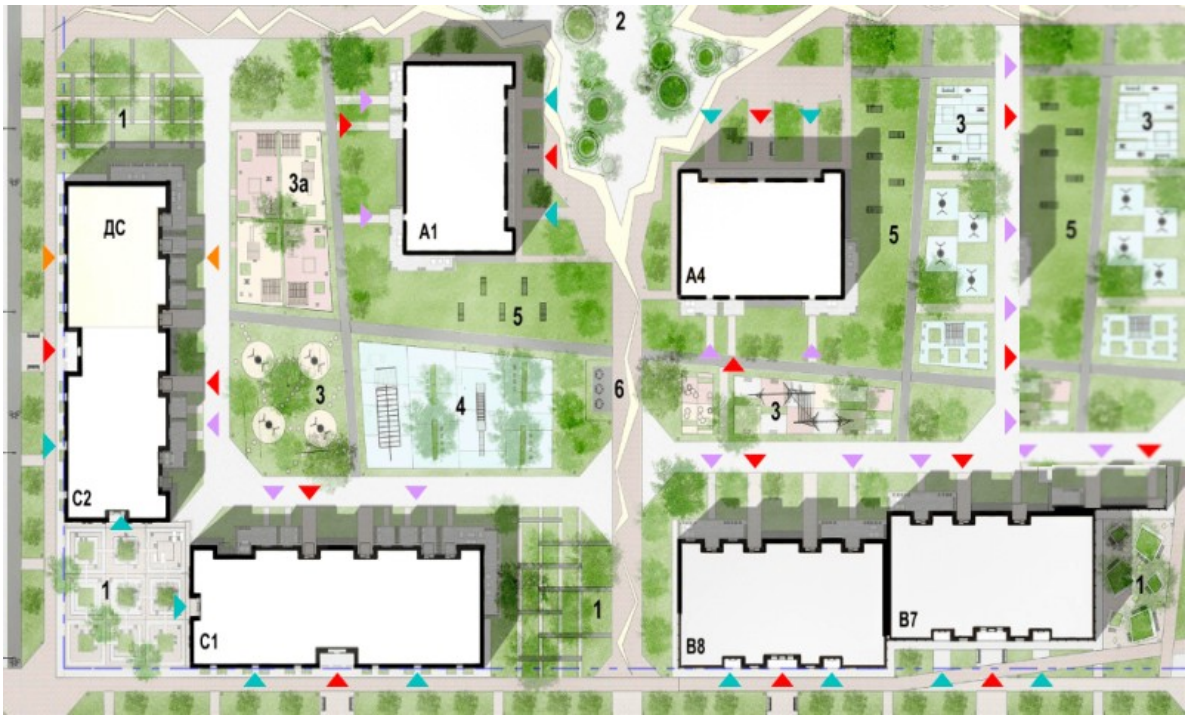


Рис. 2.2. Пример графического файла с отображением двора.

Рис. 2.3. Пример графического файла с отображением отдельного здания.

2.2.7. Настройка «помещений»

Основная функция Вызывной Панели - это доставка видео-звонков от посетителей в «помещения». Помещением может быть как офисное или производственное помещение, так и квартира в МКД. Помимо этого, помещение может быть виртуальным, т.е. существовать только для того, чтобы отправить на него вызов (например, мобильный офис или «экстренная служба 112»).

Список доступных к вызову помещений задается в секции «**Apartments**» конфигурационного файла. Каждый элемент списка представляет собой JSON структуру, описание всех параметров которой приведено в *Приложении 2*. Ниже рассмотри сокращенный вариант задания помещения.

Каждое помещение идентифицируется уникальным параметром **apt_num**, это может быть целое число (номер квартиры или офиса), строка символов, задающая номер помещения во дворе с несколькими зданиями, либо строка символов, кодирующая номер виртуального помещения.

Каждое помещение имеет наименование или ФИО собственника, это задается параметром **descr**, а так же параметром **descr_font,** определяющим алиас (шрифт) отображения данного текста.

Для помещения могут быть определены замки (параметр **locks**), которые разрешено отпирать собственнику данного помещения (сами замки описаны в секции «Locks», см. соответствующую JSON схему в *Приложении 2*). Также с данным помещением могут быть ассоциированны идентификаторы RFID ключей (параметр **keys**), которые могут отпирать указанные замки. Каждый ключ имеет уникальный идентификатор **key**, его тип **type** и текстовое обозначение владельца ключа **descr**.

ВАЖНО: На данный момент поддерживаются только ключи типа EM-Marin, значение ″type″:″emmarine″.

Для доставки голосовых и видео-вызовов в данное помещение (или к его собственнику) используется параметр **routes,** который представляет собой список возможных маршрутов. Каждый маршрут может быть одного из трех типов:

- **sip** — доставка вызова путем размещения SIP зонка;
- **switch** — доставка через аналоговый коммутатор;

- **direct** — доставка вызова на аналоговую трубку подключенную непосредственно к Вызывной Панели.

Для каждого из типов маршрутов требуется задавать свои параметры адресации:

- Для SIP звонков обязательно требуется указать **UriTo** — идентификатор получателя вызова и **UriFrom** - идентификатор источника вызова. Помимо этого, может потребоваться указать имя пользователя и пароль для прохождения авторизации на SIP сервера (параметры **authUsername** и **secret**), SIP прокси (параметр **proxy**) и текстовое обозначение источника звонка (параметр **display**).
- Для звонка через Коммутатор обязательно требуется указать адрес коммутатора в сети Modbus (параметр **modbus_address**) и физический адрес абонентской трубки: «единицы» (параметр **E**) и «десятки» (параметр **D**).
- Для вызова на трубку, подключенную напрямую к вызывной панели (маршрут типа **direct**), никаких дополнительных параметров указывать не требуется. Однако, для такого подключения требуется произвести точную настройку параметров модуля **HansetState** (см. описание JSON схемы для секции «HandsetSate» в *Приложении 2*) с целью правильного детектирования положения трубки.

Ниже приведен пример описания одного из помещений, доставка звонка в которое может осуществляться **одновременно по двум маршрутам**: через аналоговый коммутатор (direct) и через SIP:

```
{
        "apt_num":"101",
        "descr":"Квартира Ивановых",
        "descr_font":"FONT3",
        "keys":[
                {
                        "descr":"Мама",
                        "key":"576AC71EC2",
                        "type":"emmarine"
                },
                {
                        "descr":"Папа",
                        "key":"578AC71EA2",
                        "type":"emmarine"
                },
                {
```

```
                        "descr":"Дочка",
                        "key":"566AC71EB2",
                        "type":"emmarine"
                },
                {
                        "descr":"Жучка",
                        "key":"566AC70EC1",
                        "type":"emmarine"
                }
        ],
        "lock_open_delay":"3",
        "locks":["front_door_lock","back_door_lock"],
        "routes":[
                {
                        "UriFrom":"intercom201@sip.data72.ru",
                        "UriTo":"101@sip.data72.ru",
                        "display":"Домофон 201",
                        "type":"sip"
                },
                {
                        "D":"0x0001",
                        "E":"0x0001",
                        "modbus_address":"1",
                        "type":"switch"
                }
        ],

},
```

2.3. Подключение аналогового коммутатора

Вызывная панель «ЕРМАК» может быть интегрирована в существующую аналоговую домофонную сеть, что позволяет производить плавный переход к цифровым технологиям. Для интеграции Вызывной Панели в аналоговую сеть требуется использовать специализированное устройство — Блок Коммутации (далее, «коммутатор»), обеспечивающий доставку аналогового голосового сигнала между Вызывной Панелью и аналоговыми абонентскими трубками — УКП (устройство квартирное переговорное, далее «абонентская трубка») посредством матрицы проводников, ведущих к абонентским трубкам. Каждая абонентская трубка в такой матричной сети адресуется парой «единиц» (Ex) и «десятков» (Dx), в зависимости от физического подключения. Для того, чтобы звуковой сигнал от Вызывной Панели мог достигать абонентской трубки, коммутационное устройство должно, по команде от ВП, соединить соответствующие линии Ex и Dx со звуковым входом от ВП. Пара Exxxx/Dyyyy, представляющая два целых 16-битных числа, рассматривается как физический адрес абонентской трубки.

Для Вызывной Панели «ЕРМАК» был разработан специализированный Коммутатор «ЕРМАК», который управляется Вызывной Панелью по протоколу Modbus/RTU через интерфейс RS-485. Данный коммутатор по команде от ВП осуществляет коммутацию двухпроводной звуковой линии, идущей от ВП, к заданной абонентской трубке по переданной ему паре линий Exxxx/Dyyyy. Коммутатор позволяет производить одновременное соединение нескольких абонентских трубок с ВП, если все они имеют общий адрес Exxxx или Dyyyy, т.е. подключены к одной и той же линии «десятков» или к одной и той же линии «единиц». Также возможно одновременное подключение всех абонентских трубок - функция «Оповещение ГоиЧС». Для обслуживания такой высокой нагрузки, которая может достигать до 100 трубок на один коммутатор, в Коммутаторе «ЕРМАК» предусмотрен комплект звуковых усилителей «класса D».

2.3.1. Соединение коммутатора с ВП

На рис 2.1. видно, что Коммутатор «ЕРМАК» подключается к Вызывной Панели «ЕРМАК» по двум двухпроводным линиям: линия XS12 — обеспечивает управление коммутатором по протоколу Modbus/RTU, а линия XS11 служит для передачи звукового сигнала между ВП и абонентской трубкой.

При соединении линии XS12 следует соблюдать маркировку сигналов в ней: сигнал «**A»** Коммутатора подключается на соответствующий сигнал «**A»** Вызывной Панели, то же самое выполняется с сигналом «**B»**.

При подключении звуковой линии XS11 следует придерживаться следующего требования: сигнал «**LN**» (пин 2) в линии XS11 Вызывной Панели следует соединить с сигналом «**A.IN**» на Коммутаторе, а сигнал «**GND**» (пин 1) линии XS11 соединить с сигналом «**GND**» Коммутатора расположенного рядом с сигналом «**A.IN**».

2.3.2. Подключение абонентских трубок к Коммутатору

Аналоговые абонентские трубки подключаются к Коммутатору на линии «десятков» и «единиц» следующим образом:

- сигнал «**LN-**» абонентской трубки подключается к одной из линий «единиц» (E0-E9) Коммутатора;
- сигнал «**LN+**» абонентской трубки подключается к одной из линий «десятков» (D0-D9) Коммутатора.

При этом, необходимо отметить, что омическое сопротивление абонентской трубки в «снятом» состоянии должно составлять окло 50 Ом, а в «положенном» состоянии — не менее 600 Ом. При создании аналоговой сети необходимо придерживаться одного типа трубок, дабы избежать широкого разброса их параметров, что может привести к сложностям согласования импеданса.

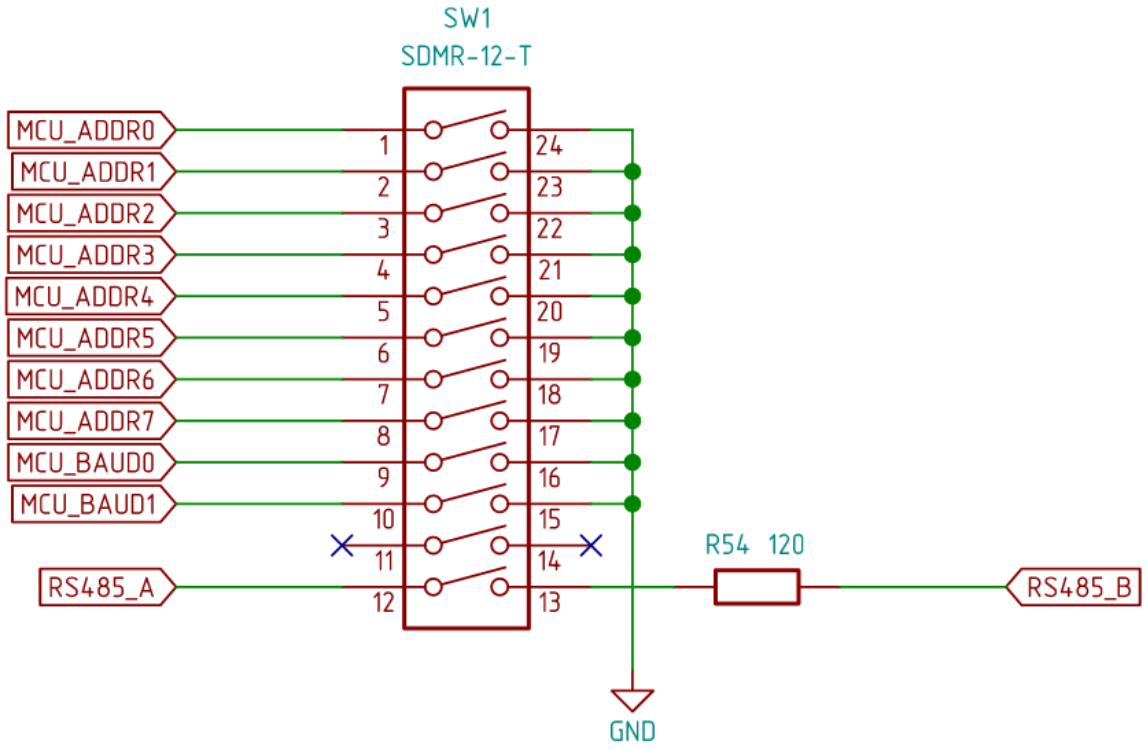
2.3.3. Задание адреса и параметров сети Modbus на Коммутаторе

Помимо физического соединения линий XS11/XS12, коммутатору нужно указать его адрес в сети Modbus. Это достигается выбором положения ряда DIP переключателей, установленных под крышкой Коммутатора: DIP переключатели с номера **1** по **8** задают биты адреса **A0-A7** соответственно. Допускается одновременная работа нескольких Коммутаторов в сети, при этом каждый коммутатор должен иметь уникальный адрес. Также необходимы выставить коммутатору скорость передачи в сети Modbus — DIP переключатели **9** и **10**. Обычно, это значение равно **115200** бод. Таблица соответствий значений DIP переключателей и скоростей приведена ниже.

Таблица 2.3.3.

DIP 9	DIP 10	Скорость передачи
OFF	OFF	9600
ON	OFF	38400
OFF	ON	57600
ON	ON	115200 (по умолчанию)

Также необходимо обратить внимание на положение DIP переключателя номер 12 который подключает/отключает терминирующий резистор 120 Ом к сигнальной линии. Если коммутаторов на линии несколько или линия имеет существенную протяженность, то необходимо включить терминирующий резистор (DIP12 = ON). В случае если коммутатор на линии один, то DIP 12 требуется отключить. Расположение и назначение DIP переключателей приведено на диаграмме ниже:



2.3.4. Согласование импеданса

При работе Вызывной Панели в аналоговой сети с большим количеством абонентов требуется обеспечить согласованность импедансов аналоговой линии

и аналогового входа ВП (XS12). Для этой цели в Вызывной Панели «ЕРМАК» предусмотрен регулировочный многооборотный потенциометр, расположенный в отсеке с разъемами и обозначенный как **R80**. Подстройку импеданса рекомендуется выполнять каждый раз при изменении числа абонентских трубок на коммутационном поле.

Процесс подстройки выглядит следующим образом:

1. Стандартными средствам Вызывной Панели произвести вызов аналогового абонента с предположительно самой большой длиной шлейфа (провода).
2. Когда абонент ответит на вызов (снимет трубку), необходимо, плавно вращая ротор потенциометра **R80** с помощью плоской и тонкой отвертки, добиться максимальной громкости и отчетливости двусторонней голосовой связи.
3. Произвести разъединение и повторное соединение, проверить качество голосовой связи после регулировки.

2.3.5. Настройка параметров аналогово абонента

После того как аналоговая абонентская трубка подключена к коммутатору её необходимо описать в разделе настроек «**routes»** у соответствующего помещения в конфигурационной секции настроек «**Apartments»**. При этом необходимо задать следующие параметры:

- **type** — установить в значение «switch»;
- **modbus_address** — задает адрес коммутатора в сети к которому подключена абонентская трубка;
- **E** — задает физический адрес абонентской трубки (единицы);
- **D** — задает физический адрес абонентской трубки (десятки);
- **handset_offhook_threshold** — задает порог напряжения (мВ) в линии при снятии трубки (ответ абонента);
- **button_press_threshold** — задает порог напряжения (мВ) в линии при нажатии абонентом кнопки «отпирания замка».

2.3.6. Подстройка параметров детектирования подъема трубки и нажатия кнопки

Как отмечалось а разделе 2.3.2 абонентские трубки устроены таким образом, чтобы выдавать на линию «LN» определенное омическое сопротивление в одном из трех положение: «трубка висит», «трубка снята», «кнопка нажата». Определение состояние вызываемой трубки производится коммутатором по среднеквадратичному напряжению в линии. Пороги срабатывания (в милливольтах) определены в программном обеспечение коммутатора и могут быть программно изменены для конкретного вызова со стороны Вызывной Панели по протоколу Modbus. В некоторых случаях (длинная линия или нестандартная трубка) требуется индивидуально настроить пороги срабатывания для определенного абонента чтобы избежать ложного детектирования подъёма трубки или несрабатывания нажатия кнопки. Для этих целей в секции **«routes»** помимо физических адресов **«E»** и **«D»** могут быть заданы параметры «**handset_offhook_threshold»** и «**button_press_threshold»** устанавливающие порогb срабатывания «подъём трубки» и «нажатие кнопки».

По умолчанию, если эти параметры не заданы в конфиге или установлены в пустое значение, используются запрограммированные в коммутатор значения порогов: **4700 мВ** и **9000 мВ** соответственно. При устранении проблем детектирования руководствуются следующими простыми правилами:

- Если при вызове на трубку происходит ложное определение ответа, то параметр «**handset_offhook_threshold»** необходимо постепенно увеличивать с шагом 200-250 мВ.
- Если при вызове на трубку не происходит детектирование состояние ответа (трубка поднята), то параметр «**handset_offhook_threshold»** следует постепенно уменьшать с шагом 200-250 мВ.
- Если при звонке на трубку происходит ложное детектирование нажатия кнопки отпирания замка, то параметр «**button_press_threshold»** следует постепенно увеличивать с шагом 200-250 мВ.
- Если при звонке на трубку не происходит детектирования нажатия кнопки отпирания замка, то параметр «**button_press_threshold»** следует постепенно уменьшать с шагом 200-250 мВ.

2.3.7. Пример: вызов на аналоговый абонентский аппарат подключенный к коммутатору «Ермак»

Ниже приведен фрагмент конфигурационного файла ВП для абонентской трубки подключенной к физическому адресу 34 на коммутаторе с адресом 1 в сети Modbus и с индивидуальными настройками порогов срабатывания:

```
{
        "apt_num":"104",
        "descr":"Вызов через коммутатора: кв 34",
        "descr_font":"FONT2",
        "locks":[
                "front_door_lock",
                "back_door_lock"
        ],
        "routes":[
                {
                        "type":"switch",
                        "D":"3",
                        "E":"4",
                        "modbus_address":"1",
                        "handset_offhook_threshold":"5100",
                        "button_press_threshold":"8700"
                }
        ]
},
```

Таким образом постепенно подбирая параметры порогов можно добиться правильного срабатывания трубки и кнопки индивидуально для каждого абонентского аппарата.

Важно! Если при вызове абонентского аппарата происходит моментальное (ложное) срабатывание кнопки отпирания звонка, то с большой вероятностью это указывает на то, что аппарат (трубка) физически не подключены, имеется обрыв или неверно указан адрес аппарата. По этому, перед тем как приступать к регулировке порогов срабатывания необходимо убедиться, что присутствует физическое подключение абонента к указанному адресу на коммутаторе и его аппарат исправен.

2.4. Взаимодействие Вызывной Панели с облачным сервисом

В случае, когда вызывная панель должна периодически получать свой конфиг из облачного сервиса (сервера находящегося в публичной сети), встает вопрос как произвести привязку конкретной панели к аккаунту пользователя внутри сервиса. Для этого предлагается следующий план действий:

- Не существует двух устройств (вызывных панелей) с одним и тем же SID, а значит SID можно использовать в качестве уникального идентификатора панелей внутри облачной системы. SID можно ассоциировать с пользователем сервиса, у каждого пользователя может быть один и более SID-ов (вызывных панелей которые он эксплуатирует).
- Так как сменить SID нельзя, то часто передавать его по публичным сетям не стоит, иначе кто-то подсмотрев SID сможет действовать от его имени, т.е. получать и подсматривать конфиг. Вместо этого, для всех операций взаимодействия устройства с сервисом необходимо использовать некую случайно сгенерированную авторизационную куку (**auth_cookie**).
- Чтобы сгенерировать уникальную куку предлагается передавать в сервис: SID, локальный IP-адрес устройства и еще одну строку символов, которая известна обоим сторонам - серверу и пользователю вызывной панели. Назовем эту строку **паролем**.
- Таким образом, чтобы осуществить привязку вызывной панели к сервису, необходимо:
    - Зарегистрироваться на сервисе (получить аккаунт);
    - Добавить в сервис SID устройства - его можно узнать подключившись к устройству через WEB интерфейс, по SSH или через консоль;
    - Сгенерировать для этого SID случайный пароль;
    - Еще раз подключиться к устройству через WEB, указать этот пароль и URL адрес сервера API;
    - Устройство далее автоматически подключится к серверу API, получит авторизационную куку с использованием указанного пароля и далее будет регулярно получать обновления конфига по имеющейся авторизационной куке, а так же отсылать на сервер оповещения с её помощью.

- Сервис может в любой момент аннулировать действующую авторизационную куку, тогда устройство повторит процесс получения куки по паролю. Желательно, что бы сервер периодически аннулировал авторизационные куки, но не слишком часто.
- Сервис может в любой момент аннулировать куку и пароль, тогда устройство больше не сможет получать обновления и взаимодействовать с сервисом.

2.5. Обновление ПО

Программное обеспечение Вызывной Панели состоит и двух частей: системного - ОС Linux, системные утилиты и библиотеки, и специализированного — набор программных модулей обеспечивающих весь «полезный» функционал Вызывной Панели. Таким образом, обновление ПО разделено на две части.

2.4.1. Обновление системного ПО и начальная прошивка eMMC

Обновление системного ПО во встроенное Flash (eMMC) памяти Вызывной Панели может быть осуществлено только путем загрузки с внешнего носителя - MicroSD карты, с последующей инициализацией eMMC данными либо с носителя с которого была произведена загрузка, либо из сети Internet путем выкачивания полного образа записи его на eMMC.

Для облегчения задачи установки (обновления) системного ПО был разработан shell скрипт, позволяющий автоматизировать этот процесс. Таким образом процесс инициализации eMMC выглядит следующим образом:

1. На рабочей машине загрузить с сайта производителя последнюю версию полного образа системы и записать её на свободную MicroSD карту:

   ```
   $ wget https://www.fabmicro.ru/pub/SIPHomePhone/A20-SIP_HomePhone-all.img.gz
   ```

2. Распаковать образ в текущий каталог:

   ```
   $ unzgip A20-SIP_HomePhone-all.img.gz
   ```

3. Записать распакованный образ на SD карту, где /dev/sdb — имя устройства SD карты.

   ```
   $ sudo dd if= A13-SIP_HomePhone-all.img of=/dev/sdb bs=8192 status=progress
   ```

4. Вставить SD карту в Вызывную Панель и загрузиться с неё.
5. Подключиться к Вызывной Панели через SSH или через последовательную консоль, как указано в разделе «*3.1. Доступ к оболочке ОС Linux на ВП*».
6. Исполнить инсталляционный shell скрипт:

```
# su
# /etc/install.sh
```

7. В появившемся меню выбрать п. 1 для переноса содержимого SD карты на eMMC, либо п. 4 — для инициализации eMMC образом из сети Internet с сайта производителя.
8. Дождаться завершения исполнения команды **dd**.
9. Погасить систему командой:

```
# poweroff
```

10. Отключить питание, вынуть загрузочную SD карту и загрузить систему с eMMC.

В случае, если в процессе инициализации eMMC памяти произошел сбой, то весь процесс следует повторить сначала, при это желательно произвести полную перезагрузку ВП (с отключением питания) с загрузочной SD карты.

При сбое может повредиться файловая система загрузочной SD карты. В этом случае можно попытаться устранить ошибку файловой системы с помощью утилиты **fsck** на рабочей машине, либо на Вызывной Панели загрузив её в монопольном (single user) режиме. Более подробно см. руководство по загрузчику U-Boot и адмиистрированию ОС Linux.

2.4.2. Обновление специализированного ПО

Обновление специализированного ПО, в том числе файлов меню, шрифтов и FSM скриптов можно выполнить через WEB интерфейс администратора из специализированных пакетов обновлений (т. н. tarball-ов) имя файлов которых имеет формат: **SIPHomePhone-*-YYYYMMDD.tar.gz**. Загрузить такие пакеты можно с сайта производителя по следующей ссылке:

`https://www.fabmicro.ru/pub/SIPHomePhone/`

Для установки пакетов обновлений, один или несколько пакетов необходимо поместить в спул обновлений используя меню WEB интерфейса: **«Системные операции» → «Загрузить обновление ПО»**. При входе в данное меню будет отображен список файлов находящихся на данный момент в спуле обновлений. Используя функцию **«Загрузить»** необходимо поместить файлы пакетов в спул,

после чего выполнить перезагрузку системы через меню **«Системные операции» → «Перезагрузить систему».**

Расположенные в спуле файлы пакетов будут распакованы и установлены при загрузке операционной системы в рабочий каталог **/home/fabmicro/SIPHomePhome/**, при этом будет создана резервная копия этого каталога с именем вида **/home/fabmicro/SIPHomePhome-123456789** где цифры в конце соответствуют UNIX time в момент выполнения резервной копии.

В нормальных условиях спул обновлений должен быть пустым. Если в спуле присутствует какой-то файл, то это означает, что данный файл не был распознан как пакет обновлений и его можно удалить нажатием на **[X]**.

Просмотреть версию установленных и загруженных на данный момент программных модулей можно через WEB меню: **«Системные операции» → «Просмотреть версии модулей»**.

ВАЖНО! Так как предыдущие версии ПО всегда сохраняются (не удалятся), то возможен случай, когда при слишком частом обновлении ПО в корневой файловой системе не останется свободного места. В этом случае необходимо выполнить подключение к системе ВП по SSH средствами ОС Linux удалить невостребованные резервные копии ПО.

Так же необходимо недопускать сбоев питания при выполнении обновлений, так как это может привести к ошибкам в файловых системах Вызывной Панели.

3. Руководство программиста

Программное обеспечение Вызывной Панели состоит из **системного** ПО (ядро ОС Linux, набор системных утилит и библиотек) и **специализированного** ПО - ряда модулей написанных на языках «C», «C++» и «Perl5», а также на специализированном скриптовом языке. В данном разделе руководства представлено описание специализированного ПО - программного интерфейса (API) модулей и принципа их взаимодействия друг с другом. Данное описание дано с целью предоставить эксплуатанту более глубокое понимание принципа функционирования изделия, а также дать возможность производить адаптацию функционала изделия под свои потребности и нужды.

3.1. Доступ к оболочке ОС Linux на ВП

Как отмечалось выше, Вызывная Панель функционирует под управлением ОС Linux на основе дистрибутива Devuan (Debina без systemd). Доступ к оболочке операционной системы можно осуществить двумя способами:

Путем подключение через последовательный порт консоли (см. описание разъема XS26 в Таблице 1.2). В этом случае на консоль выводится сообщение приветствия и промпт **login:** для авторизации пользователя. Для входа в оболочку системы таким способом следует использовать имя пользователя **root** и пароль: **toor**.

Путем подключения через сеть по протоколу SSH. Для этого потребуется приобрести RSA ключ, запросить который можно у официального дистрибьютора.

Предположим, что приватная часть RSA ключа находится в файле ***siphomephone_rsa***, тогда для доступа к оболочке можно использовать следующую команду с UNIX хоста в сети, из которой есть достижимость к устройству ВП:

```
$ ssh -i siphomephone_rsa fabmicro@192.168.0.50
```

где 192.168.0.50 это текущий IP-адрес вызывной панели. Узнать текущий IP-адрес ВП можно с на-экранного меню «**About**», см. раздел «*2.2.3. Просмотр текущих сетевых настроек и SID*»).

Важно указывать имя пользователя **fabmicro**, так как в домашнем каталоге этого пользователя в системе ВП располагается все специализированное

программное обеспечение и в настройках этого пользователя указана публичная часть RSA ключа для авторизации.

ВАЖНО: После входа в оболочку имеет смысл заменить пароль пользователя **root,** используя стандартные средства (команда **passwd**), а также заменить или добавить новый публичный RSA ключ для пользователя **fabmicro,** отредактировав содержимое файла ***/home/fabmicro/.ssh/authorized_keys***.

3.2. Структура файловой системы

Файловая система Вызывной Панели разбита на две части:

- rootfs (/) - корневая файловая система, которая при загрузке ОС всегда монтируется в режиме read-only, и
- variables (/var) — файловая система, содержащая часто изменяемые файлы (конфиг, DHPC lease, медиа файлы).

Все специализированное ПО и сопутствующие ресурсы (картинки, скрипты, JSON схемы) расположены в домашнем каталоге пользователя fabmicro в ***/home/fabmicro/SIPHomePhone*** — что является **рабочим каталогом** для всех модулей.

Структура рабочего каталога следующая:

./ - содержит исполняемые файлы модулей;

./menu/ - содержит файлы формата JSON, описывающие различные меню графического интерфейса пользователя;

./images/ - содержит графические файлы (PNG) и файлы шрифтов (TTF), используемые в меню;

./web_admin_docroot/ — содержит файлы скриптов (Perl, JavaScript) необходимые для функционирования модуля WebAdmin.

./web_admin_docroot/schema/ — содержит JSON схемы всех секций консолидированного конфига. Также используется модулем WebAdmin. При добавлении новых параметров или изменении диапазона значений для уже существующих, эти JSON схемы следуют отредактировать.

./SuperGlue/ — содержит текст программы модуля SuperGlue на языке Perl5 и набор файлов описания нескольких «машин состояний» (файлы *.fsm).

Для того, чтобы производить изменения в рабочем каталоге, т.е. модифицировать скрипты, заменять графические файлы или шрифты, необходимо перемонтировать корневую файловую систему в режим **read-write** следующей командой:

```
# mount -w -o remount /
```

Конфигурационные файлы и медиа-файлы, выкачиваемые из сети, складываются в подкаталог ***/var/configs*** и ***/var/media*** соответственно. Для работы с данными в этих подкаталогах перемонтировать корневую систему в режим read-only не требуется.

3.3. Структура взаимодействия модулей

Специализированное программное обеспечение Вызывной Панели состоит из ряда модулей, представленных в виде исполняемых ELF-файлов, расположенных в рабочем каталоге. Каждый модуль представляет собой один или несколько процессов в операционной системы, при этом каждый процесс может иметь множество нитей (POSIX threads).

Модули взаимодействуют между собой посредством системы обмена сообщениями **SunBus**, которая позволяет модулям отправлять друг-другу оповещения, команды и получать ответы на них - аналогично шине D-Bus, но в упрощенном варианте и с некоторыми дополнительными возможностями по отладке и «подглядыванию» за процессом. Все сообщения в системе SunBus представлены в виде JSON структур, помещенных в одну строку текста. Более подробно описание системы SunBus дано в разделе «*3.17. Менеджер сообщений SunBus*».

Среди ПО вызывной панели существует специальный модуль **SuperGlue**, который написан на языке Perl5 и представляет собой сложную много-сессионную «машину состояний» (FSM). Модуль SuperGlue является неким связующим звеном («клеем») между всеми остальными модулями, именно он задает логику функционирования вызывной панели и логику обработки звонков, управляет графическим интерфейсом, посылает команды на отпирания замков, отправляет нотификации на сервер и исполняет команды HTTP API.

На рис. 3.1 схематично изображена структура взаимодействия между модулями.

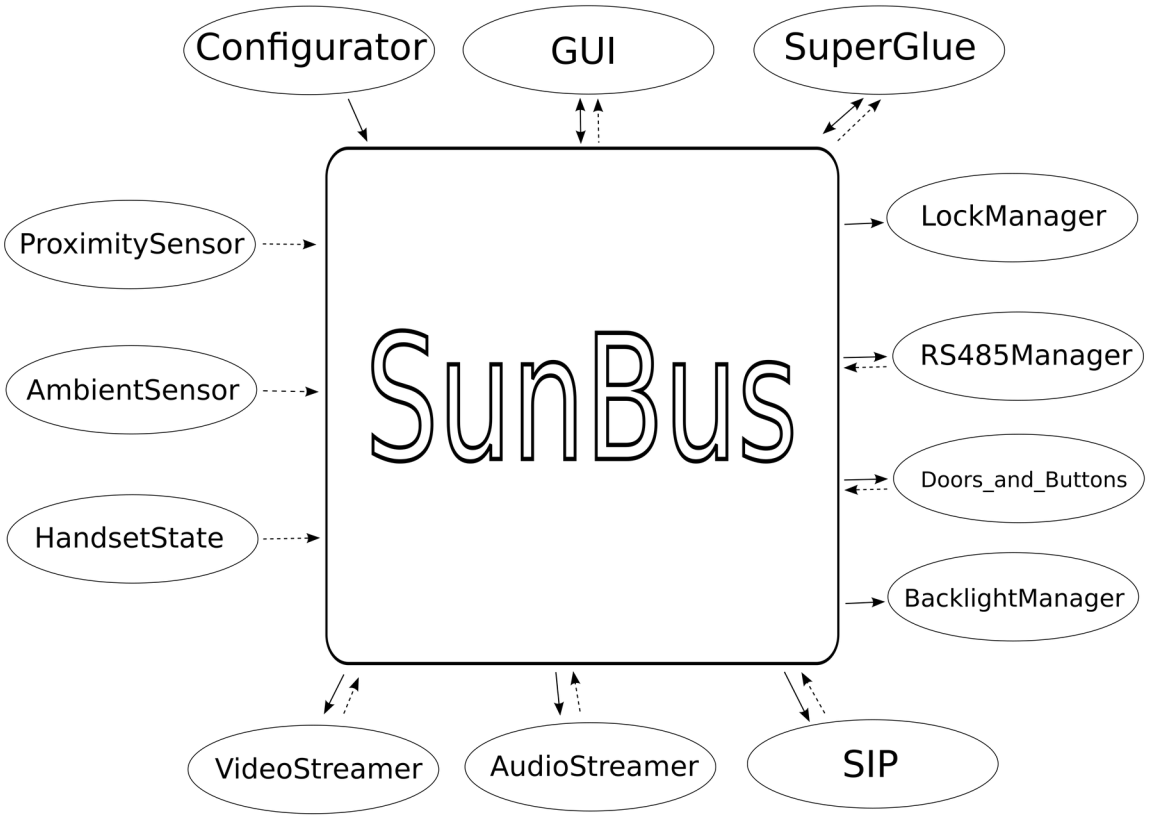


Рис. 3.1. Структура взаимодействия модулей в системе SunBus. Стрелками обозначено направление следования «команда» (сплошная линия) и «асинхронных событий» (пунктирная линия).

3.4. Перечень модулей и их краткое описание

В специализированном программном обеспечении Вызывной Панели задействованы следующие программные модули:

- SUNBUS (процесс SunBus) — обеспечивает обмен сообщениями между остальными модулями, командами и результатами их исполнения. Самостоятельно стартует все остальные модули и отслеживает их «жизнедеятельность», в случае «крэша» записывает отчет во временный файл и перезапускает все модули.
- CONFIGURATOR (процесс configurator) — предназначен для получения из сети самого свежего конфигурационного файла и регулярных обновлений к нем. Запускается первым (из SunBus-a), подготавливает и верифицирует консолидированный конфиг, по готовности сообщает об этом системе SunBus, что приводит к старту всех остальных модулей.
- AUDIO_STREAMER (процесс audio_streamer) — обеспечивает работу со звуковыми устройствами, реализует RTP для звукового трафика.
- VIDEO_STREAMER (процесс video_streamer) — обеспечивает работу с видео аппаратурой (дисплей, камера, аппаратное кодирование/декодирование), реализует RTP для видео трафика.
- AMBIENT_SENSOR (процесс ambient_sendor) — обеспечивает опрос датчика освещенности, преобразовывает его данные в поток событий в системе SunBus.
- PROXIMITY_SENSOR (процесс poximity_senso) — обеспечивает опрос датчика «приближения», преобразовывает его данные в поток событий в системе SunBus.
- BACKLIGTH_MANAGER (процесс backlight_manager) — обеспечивает работу с подсветкой LCD дисплея, экспортирует в систему SunBus набор команд позволяющих другим модулям управлять подсветкой экрана.
- HANDSET_STATE (процесс handset_state) — обеспечивает опрос АПЦ состояния аналоговой домофонной трубки подключенной напрямую к ВП, преобразует данные АЦП в поток событий в системе SunBus.
- DOOR_AND_BUTTONS (процесс doors_and_buttons) — обеспечивает опрос входных GPIO, связанных с датчиками положения дверей и кнопок, преобразует эти данные в поток событий в системе SunBus.

- RFID_READER (процесс rfid_reader) — обеспечивает опрос аппаратного считывателя RFID ключа, формирует события в системе SunBus при появлении ключа в области достижимости считывателя.
- RS485_MASTER (процесс rs485_master) — обеспечивает работу с устройствами на шине RS-485 (в том числе с домофонным Коммутатором «ЕРМАК»), преобразует запросы из системы SunBus в запросы Modbus, а ответы от Modbus в поток событий в системе SunBus.
- LOCK_MANAGER (процесс lock_manager) — обеспечивает управление линиями GPIO, к которым подключены опто-реле управления ЭМ замками, принимает и исполняет команды из системы SunBus.
- HTTP_REQUESTER (процесс http_requester) — предоставляет асинхронных API для исполнения HTTP запросов для других модулей в системе SunBus.
- SIP2 (процесс sip2) — реализует стек протокола SIP/2.0, принимает команды из системы SunBus на размещение и управление SIP звонками, преобразует внутренние состояния звонков в поток событий в системе SunBus.
- GUI (процесс gui) — обеспечивает графический интерфейс пользователя и систему мену.
- SuperGlue (процесс super_glue.pl) — обеспечивает логику работы Вызывной Панели, способы маршрутизации звонков, формирует оповещения на сервер, управляет отпиранием замков, и т.д.

Более подробное описание некоторых важных модулей будет приведено ниже.

3.5. Модуль графического интерфейса (GUI)

Модуль графического интерфейса представлен исполняемым файлом *SIPHomePhone/gui*, который предназначен для работы в системе обмена сообщений **SunBus**. Основная задача данного модуля - это отображение системы на-экранных «меню», снабженных соответствующей логикой, необходимой для выполнения функций вызывной панели. Логика «меню» реализуется набором скриптов, описываемых в виде строк текста в JSON файлах (файлы меню).

Помимо этого, модуль графического интерфейса предоставляет ряд команд в системе SunBus, позволяющих другим модулям взаимодействовать с системой меню.

3.5.1. Набор команд

Модуль графического интерфейса поддерживает описанный ниже набор команд. Каждая команда представляет собой JSON структуру в виде строки текста (содержание символов «перевод строки» не допускается).

Команда **show_picture** предназначена для перехода на специальное меню **picture.json** для отображения отдельной картинки. Возврат происходит на меню **BaseMenu** при нажатии на экран, либо по истечении времени «таймаута». Таймаут задается в секундах в параметре *timeout*, либо в конфиге в *"TimeoutShowPicture"*. Параметр *file_path* указывает на картинку в файловой системе вызывной панели.

Пример 1:

msg: user {"command":"show_picture", "file_path":"/home/fabmicro/SIPHomePhone/images/fail.png", "timeout":30},

Пример 2:

msg: user {"command":"show_picture", "url":"https://fabmicro.ru/SIPHomePhone/images/fail.png"},

Здесь параметр *url* указывает на адрес файла картинки в сети.

Команда **validate_config** служит для валидации (проверки синтаксиса) конфига. Пример:

msg: user {command:validate_config, config_name:"all.json.default"}

Команда **tap** служит для эмуляции нажатия на сенсорный экран. Пример:

msg: user TAG:GUI_CMD TEXT:{command:tap, x:400, y:320}

Команда **screenshot** служит для записи экранной копии в графический файл формата TGA. Пример:

msg: user TAG:GUI_CMD TEXT:{command:screenshot, file_name:"screen.tga"}

Существует иной способ сохранить экранную копию в графический файл - сделав тройное нажатие по сенсорному экрану, для этого требуется установить параметр ***EnableScreenShot*** в значение ***true*** в конфигурационном файле. Данный способ рекомендуется использовать только в целях отладки, при нормальной работе вызывной панели он должен быть отключен.

3.5.2. Калибровка сенсорной панели

Все сенсорные панели по своей природе имеют некоторый разброс по параметрам, поэтому при установке программного обеспечения на Вызывную Панель требуется проводить процедуру калибровки сенсорной панели. Для этого исполнительный файл ***gui*** можно запустить со следующими параметрами:

./gui -touch_test - чтобы протестировать работу сенсорного экрана.

./gui -touch_calibrate - чтобы выполнить калибровку сенсорного экрана.

Для корректной работы требуется, чтобы в текущем каталоге находился файл **touch.json**. В этом файле находится имя драйвера сенсорной панели, а также текущие данные калибровки сенсорной панели, которые будут **перезаписаны** после выполнения процедуры калибровки.

Для дальнейшей работы графической системы полученный в результате калибровки файл **touch.json** требуется сохранить в рабочем каталоге: ***/home/fabmicro/SIPHomeHome***.

Процедура калибровки состоит в последовательном нажатии сенсорного экрана в пяти точках, обозначаемых знаком «+» - четыре точки по углам экрана и одна в его центре. Для нажатия рекомендуется использовать карандаш или иной предмет с притупленным острием.

3.5.3. Структура файлов в каталоге *menu/*

Каждый файл в данном каталоге описывает отдельное на-экранное «меню». Ниже приведен перечень готовых «меню» и их назначение:

- **common.json —** содержит описания переменных, используемых во всех остальных меню. Требуется для передачи данных и обмена с основной частью модуля.
- **about.json -** меню «**About»** отображает значения сетевых настроек, некоторых параметров и версий модулей вызывной панели. Данное меню вызывается специальным сочетанием нажатий, см. раздел «*2.2.3. Просмотр текущих сетевых настроек и SID*».
- **call_5.json —** содержит меню исходящего звонка, отображает клавиатуру набора номера помещение, анимацию при исходящем вызове.
- **income.json** — содержит меню входящего звонка, отображает анимацию и элементы управления необходимые для принятия или отклонения входного вызова.
- **picture.json** — содержит меню отображения произвольной картинки по команде **show_picture**, используется как инструмент для модуля **SuperGlue**.
- **preload.json** — содержит меню отображаемое в процессе инициализации графической системы вызывной панели. Данное меню загружается и отображается первым, до загрузки всех остальных картинок и меню.
- **keyboard.json** — содержит меню ввода номера помещения.
- **yard.json** — содержит меню отображения «двора» и выбора здания.
- **keyboard_building.json** — содержит меню ввода номера помещения при работе в режиме «двора», отображает адрес выбранного дома.
- **buttons_1.json, buttons_2.json, buttons_3.json, buttons_4.json** — содержат меню с набором запрограммированных **«**кнопок», число и расположение кнопок определяется соответствующим файлом.
- **main.json** — содержит стартовое меню всей графической системы. По умолчанию в нем производится отображение рекламных объявлений, описанных в секции *"Advertizer"*. При любом нажатии на сенсорный

экран происходит переход в меню, описанное параметром *"BaseMenu"* - это может быть **keyboard.json, yard.json, buttons_?.json.**

Файлы **about.json**, **call_5.json**, **common.json**, **income.json**, **picture.json**, **preload.json** являются служебными и обязательно должны присутствовать в папке ***menu/***.

Внутренняя структура файлов «меню»

Каждый файл описания меню представляет собой JSON структуру, содержащую следующие параметры:

- Параметр *"name"* задает имя данного «меню», по которому можно ссылаться (переходить) на данное меню командой `ChangeMenu()`.
- Секция *"header"*:
  - ***"logical_screen_x"*:800, *"logical_screen_y"*:480 разрешение экрана. Всегда должно быть 800x480 для данной версии Вызывной Панели.**
  - **секция *"var"*: описывает список локальных переменных, используемых в данном меню. Пример: объявление двух переменных untap_time и inp_num, типа int и string соответственно:**
    - **"untap_time":"int",// имена переменных и их тип.**
    - **"inp_num":"string",**
- **Секция *"items":* содержит список элементов меню, каждый из которых описывает отдельный объект (item) на экране, содержащий следующие параметры:**
  - ***"name":* название элемента, для ссылок на данный элемент,**
  - ***"ani":* все поля начинающиеся на ani - это графика - картинки в виде графических файлов (поддерживается ',' '*' '?'), например, "b2.png, b3.png, b4.png" или "a?.png". Здесь же можно указывать имя алиасов из раздела *"aliase"*. Например, FONT2.**
  - ***"ani_to_touch":* анимация перехода в нажатое состояние.**
  - ***"ani_from_touch":* анимация выхода из нажатого состояния.**
  - ***"ani_touch":* нажатое состояние.**

Координаты задаются следующими переменными x, y, size_x, size_y, left, right, top, bottom (если размер отсутствует, то он берется равным размеру картинки).

- **"x", "y" координаты центра картинки. Их можно заменить на "left", "right", "top", "bottom".**

- **"size_x", "size_y" размеры картинки (для шрифтов это зона обрезания всей текстовой строки).**
- **"touch_size_x", "touch_size_y" размеры области для обработки нажатия**

Для анимаций можно также задать смещение:

- ***"shift_to_touch":"*1,1" при отрисовке анимации ani_to_touch все картинки будут отображаться со смещением x=1, y=1.**
- ***"shift_from_touch":"*1,1" при отрисовке анимации ani_from_touch все картинки будут отображаться со смещением x=1, y=1.**
- ***"shift_touch"*:"2,2 при отрисовке анимации ani_touch все картинки будут отображаться со смещением x=2, y=2.**
- ***"ani_speed"*: скорость смены кадров при анимации, в мс.**
- ***"prop":* перечисление опций данного элемента через символ «+»:**
  - **TEXT - если данный элемент содержит текст, а его анимация - «шрифт».**
  - **MONOSPACE - для шрифтов с равноразмерными буквами.**
  - **ALPHA — включает отображение альфа-канала для сглаживания краев картинки.**
  - **TOUCH_FROM_PICT — указывается если зону проверки нажатия нужно взять из альфа канала картинки.**
  - HIDE — указывает на то, что по-умолчанию данная картинка спрятана. Визуализация картинки осуществляется командой ShowItem().
- *"value":* текст для вывода на экран, если тип анимации **ani** — шрифт. Внутри программной части меню он может меняться или загружаться из конфигов при старте меню.
- ***"font_size"* размер символов шрифта в текстовой строке, если ani — шрифт.**
- ***"color"*: задает цвет символов шрифта в текстовой строке (пример: 0xffffffff — белый цвет).**

- **Секция *"commands"* содержит описание скриптов управляющих данным наэкранным меню.**

3.5.4. Язык программирования для описания скриптов

Для здания логики работы системы меню используется специализированный язык программирования скриптового типа. Команды этого языка находятся в секции *"commands"*. Каждый элемент этой секции является особым условным оператором:

- if_start — срабатывает только один раз при старте данного меню;
- if_tap — срабатывает при каждом нажатии на экран. В качестве параметра может принимать строку содержащую item_name, в таком случае сработает только если нажатие произошло на данный элемент меню;
- if_untap — срабатывает при каждом «отжатии» (убирании пальца от сенсорного экрана). В качестве параметра может принимать строку содержащую item_name, в таком случае сработает только если для данного элемента меню на экране;
- if_mrunner — срабатывает если пришло сообщение от сервиса межмодульного взаимодействия SunBus (ранее он назывался Mrunner), и принятый в сообщении JSON удовлетворяет условиям описанным в параметре. Например, параметр «TAG:PROXIMITY_SENSOR_RESP TEXT:{proximity_state!=fingering}» сработает только для сообщений от PROXIMITY_SENSOR_RESP и если proximity_state не равен «fingering»;
- if_var — вычисляет арифметическое выражение записанное в параметре. Если его результат не равен 0, то условие срабатывает.

У перечисленных операторов может быть дополнительное условие and_if_var и обязательно есть слово then со списком команд, выполняющихся если выполняется основное условие и дополнительное and_if_var.

Список команд является комбинацией вызовов функций и обычных арифметических выражений. Все вычисления являются строго типизированными, с двумя типами данных: string и int.

Встроенные функции языка:

- strlen(string str) - возвращает количество символов в строке.
- str2int(string str) - преобразует string в int.
- int2str(int i) - преобразует int в string.

- ToSunBus(string msg) — передает системе межмодульного взаимодействия SunBus строку msg.
- GetSunBusJsonItemString(string name) — возвращает значение поля name в строковом виде, из JSON строки полученной из системы межмодульного взаимодействия SunBus.
- GetSunBusJsonItemInt(string name) — возвращает значение поля name в целочисленном виде, из JSON строки полученной из системы межмодульного взаимодействия SunBus.
- GetConfigJsonString(string path) - ищет в конфиге строку разбирая путь path, заданный в виде GUI/Buildings[10]/aptnum_prefix или GUI/Buildings[aptnum_prefix=10]/min_kv
- GetConfigJsonInt(string path) - ищет в конфиге число распарсивая путь path, заданный в виде GUI/Buildings[10]/aptnum_prefix или GUI/Buildings[aptnum_prefix=10]/min_kv.
- SaveScreenshot(string file) — записывает скриншот экрана в виде файл file, в 32 битном формате tga.
- GetTapX() - возвращает позицию X последнего нажатия на экран.
- GetTapY() - возвращает позицию Y последнего нажатия на экран.
- GetTimeMs() - возвращает количество времени (мс), прошедшее с момента старта программы.
- ChangePictureAni(string item_name, string picture_file) — перезагружает картинку в анимацию ani из файла picture_file в item элементе с именем item_name.
- ChangeMenu(string new_menu) — переходит в новое меню new_menu.
- ChangeAni(string item_name, string new_ani) — меняет анимацию элемента с именем item_name на new_ani. Анимация может быть: ani, ani_touch, ani_to_touch, ani_from_touch, ani_birth.
- ChangeCadr(string item_name, int new_cadr) - меняет номер кадра текущей анимации у элемента с именем item_name, на new_cadr.
- HideItem(string item_name) — прекращает отрисовку элемента item_name.

- ShowItem(string item_name) — возобновляет отрисовку элемента item_name.
- SetItemText(string item_name, string value) — меняет текст у элемента item_name. Если item_name=TAPPED, то выбирается элемент на который в данный момент нажали на экране. Если UNTAPPED, то тот на котором был палец в момент отхода от экрана. Установленный текст отображается только у элементов шрифтов с prop:TEXT.
- GetItemText(string item_name, string value) - возвращает текст установленный элемента item_name. Если item_name=TAPPED, то выбирается элемент на который в данный момент нажали на экране. Если UNTAPPED, то тот на котором был палец в момент отхода от экрана.
- System(string cmd) — вызывается shell для выполнения команды.

3.6. Модуль обработки звукового потока (AUDIO_STREAMER)

Данный модуль основывается на одноименной библиотеке **libAudioStreamer.a** и предназначен для работы с аудио-интерфейсами и сетевыми аудио-потоками (RTP).

Специальные возможности:

- Поддержка множества гибко-конфигурируемых аудио-интерфейсов задается в **audio_streamer.json** (имя конфигурационного JSON файла можно передать первым параметром в командной строке).
- Использует ALSA для работы с PCM устройствами на низком уровне.
- Поддержка большого числа аудио-потоков, одно- и дву-направленных.
- Позволяет "на лету" линковать и перелинковать аудио-интерфейсы с аудио-потоками.
- Имеется возможность установить локальный заворот (loopback) как на аудио-интерфейсе, так и на сетевом потоке.
- Поддержка **STUN** протокола для сетевого аудио-интерфейса.
- Поддержка **keepalive** таймаута на сетевом потоке - поток не получающий данные в течении заданного времени помечается как **дохлый** о чем сообщается в виде сообщения SubBus.
- Простейший **jitter-буффер** для сортирования принимаемых RTP пакетов по полю **sequence number**.
- Поддерживаемые кодеки: **G711A**, **G711U**, **RAW16**.
- Проигрывание DTMF тонов на аудио-интрефейсе.

3.6.1. Архитектура классов

Для понимания работы модуля ниже приведены основные сущности (классы) содержащиеся внутри модуля.

- Класс **AudioInterface** (аудио-интерфейс) описывает аппаратный интерфейс, состоящий из одного звуко-записывающего устройства (**capture device**) и одного звуко-воспроизводящего устройства (**playback device**). Аудио-интерфейс может быть сконфигурирован на использование совершенно разных (не обязательно с одной и той же звуковой карты) устройств, главное, чтобы они логически были обращены в одну сторону, например, микрофон и громкоговоритель обращенный в сторону пользователя. Аудио-интерфейс содержит две нити: **AudioRecorder** и **AudioPlayer**, одна для записи данных во внутренний

буфер, другая - для проигрывания из внутреннего буфера на устройство (т.е. буферов тоже два - по одному в каждой ните). Также аудио-интерфейс содержит алгоритм акустического эхоподавления (**AEC** - из проекта **WebRTC**), который может быть включен или выключен по желанию в любой момент. Аудио-интерфейсы идентифицируются по уникальному имени, определенному в конфигурационном файле. Аудио-интерфейс может быть замьючен или размьючен (mute/unmute) в любой момент. На аудио-интрефейсе может быть установлена локальная звуковая петля (loopback) - т.е. копирование данных с capdure device на playback device. Данный класс является наследником абстрактного класса **AbstractAudioInterface**.

- Класс **MediaInterface** (медиа-интерфейс) описывает интрефейс аналогичный аудио-интерфейсу, но не с аппаратурой, а с файлом или сетевым ресурсом (**URL**). Доступ к файлу или сетевому ресурсу осуществляется через библиотеку **libcurl**, что позволяет использовать очень широкой формат сетевых путей, в том числе: http://, https://, ftp://, ftps://, file://. Объект этого класса представляет собой нить, которая открывает сетевой ресурс, парсит его и копирует данные во внутренний аудио-буфер. Данный класс также является наследником абстрактного класса **AbstractAudioInterface**.
- Класс **AudioInterfaceCross** (кросс) содержит одну нить, которая позволяет копировать данные из одного аудио- и медиа-интерфейсов друг в друга, таким образом достигается кроссирование (или коммутация) звуковых потоков между двумя интерфейсами наследниками **AbstractAudioInterface**. Идентификатором кросса является шестнадцатиричное число в символьном виде. Кросс может быть создан на одном и том же аудио-интерфейсе, тогда получается локальная звуковая петля (loopback). Кросс может быть выполнен между аудио-интерфейсом и медиа-интерфейсом, что приводит к проигрыванию содержимого сетевого ресурса на PCM устройстве. Кросс может быть замьючен или размьючен (mute/unmute) в любой момент. Кроссы нужно удалять после использования (команда **aif_uncross**).
- Класс **NetworkStream** (сетевой поток или стрим) работает с сетью по протоколу RTP. Сетевой поток содержит две нити **NetworkStreamTransmitter** и **NetworkStreamReceiver** для передачи и приема звукового трафика по протоколу RTP. Так же содержит алгоритмы

кодирования и декодирования (кодеки). К сетевому потоку может быть подключен только один аудио-интерфейс, либо, вообще, ни одного. Получаемые данные из сети декодируются и проигрываются на слинкованный аудио-интерфейс, а данные, получаемые от аудио-интерфейса, кодируются и отправляются в сеть. Линковка аудио-интерфейса с сетевым потоком может происходить в любой момент. Сетевой поток может быть поставлен на паузу (замьючен или размьючен - mute/unmute) в любой момент. На сетевой поток может быть установлена локальная цифровая петля (loopback) - т.е. данные поступающие из сети по RTP буду отправляться обратно без декодирования.

3.6.2. JSON API модуля AUDIO_STREAMER

Взаимодействие с модулем осуществляется через поток сообщений **SunBus**. Формат поддерживаемых сообщений и ответов приведен ниже:

- **{command:aif_mute, aif_name:FrontSide}** - Ставит на паузу (mute) обе нити записи и воспроизведения указанного аудио-интерфейса.
    - Ответ: **{response:ok, command:aif_mute, aif_name:FrontSide}**.
- **{command:aif_unmute, aif_name:FrontSide}** - Снимает с паузы (unmute) обе нити записи и воспроизведения указанного аудио-интерфейса.
    - Ответ: **{response:ok, command:aif_unmute, aif_name:FrontSide}**.
- **{command:aif_open, aif_name:FrontSide}** - Вызывает функции инициализации (методы **Open()**) у PCM устройств, входящих в указанный аудио-интерфейс. Используется только для служебных нужд, так как весь остальной API вызывает **Open()** неявно.
- **{command:aif_close, aif_name:FrontSide}** - Вызывает функции деинициализации и высвобождения ресурсов (метод Close()) у PCM устройств входящих в указанный аудио-интерфейс. Используется только для служебных нужд, так как предполагается, что все аудио-интрефейсы постоянно находятся в готовом (горячем) режиме, т.е. рекомендуется использовать механизм mute/unmute.
- **{command:aif_set_option, aif_name:FrontSide, option:1, value:2}** - Устанавливает специфическую для интерфейса опцию с номером **1** в значение **2**. На данный момент поддерживаются следующие опции:
    - **1** - установка bitrate,
    - **2** - включение/выключение AEC (1 - включено, 0 - выключего),

    - **3** - включение/выключение локальной петли (1 - петля включена, 0 - выключена).
- **{command:aif_cross, aif_name1:FrontSide, aif_name2:BackSide, aec_enable:true}** - Кроссирует звуковые потоки аудио-интерфейсов **FrontSide** и **BackSide** между собой. Фактически происходит создание объекта класса **AudioInterfaceCross,** внутри которого создается одна нить, занимающаяся копированием данных между внутренними буферами двух аудио-интерфейсов. //Если какой либо из аудио-интерфейсов не был предварительно открыт, то открывание вызовется автоматически.//
    - Параметр **aec_enable** типа **bool** указывает будет ли использован алгоритм акустического эхоподавления (на обоих) интерфейсах или нет. Программное эхоподавление хорошо нагружает CPU, на двух интерфейсах примерно до 18% на машине A13.
    - Ответ: **{response:ok, command:aif_cross, aif_name1:FrontSide, aif_name2:BackSide, aec_enable:true, cross_id:0x1a3e098}** где **cross_id** содержит уникальный идентификатор кросса, по которому будут производиться все дальнейшие операции с этим кроссом.
- **{command:aif_uncross, cross_id:0x01234567}** - Удаляет кроссировку с идентификационным номером указанным в **cross_id**.
    - Ответ: **{response:ok, command:aif_uncross, cross_id:0x01234567}**.
- **{command:aif_play_tone, aif_name:FrontSide, tone:asterisk}** - Синтезирует и проигрывает непрерывный DTMF или специализированный тон на заданном интерфейсе. Параметр **tone** задает символьное имя тона из следующего списка:
    - **zero** - DTMF 0
    - **one** - DTMF 1
    - **two** - DTMF 2
    - **three** - DTMF 3
    - **four** - DTMF 4
    - **five** - DTMF 5
    - **six** - DTMF 6
    - **seven** - DTMF 7
    - **eight** - DTMF 8
    - **nine** - DTMF 9
    - **a** - DTMF A
    - **b** - DTMF B
    - **c** - DTMF C

    - **d** - DTMF D
    - **asterisk** - DTMF Asterisk (*)
    - **hash** - DTMF Hash (#)
    - **ringback** - проигрывает стандартный КПВ
    - **noise** - проигрывает белый шум
    - **none** - отменяет проигрывание тонов, т.е. возвращает интерфейс в нормальное состояние.
- **{command:aif_play_url, aif_name:FrontSide, url:https://www.domain.com/somefile.wav, volume:100}** - Проигрывает звуковой файл в указанный аудио-интрефейс с громкостью 100%. На самом деле происходит следующее: создается новый кросс, к которому с одной стороны подлинковывается указанный аудио-интерфейс, а с другой - вновь созданный **MediaInterface,** работающий с заданым **url**. В качестве **url** можно задавать любой сетевой путь, в том числе **https://**, **ftp://** и т.д. Нужно учитывать, что медиа-интерфейс способен проигрывать только заданный формат звуковых файлов, см. ниже. Параметр **volume** задает программно определяемую громкость проигрывания в **процентах** (целое число), это означает, что амплитуды всех сэмплов будут умножены на заданное количество процентов. Возможно проигрывание с громкостью более 100%. Громкость 0% - проигрывание тишины. Громкость 100% - проигрывание с той громкостью, с которой записаны зуковые данные файла (т.е. проигрывание без изменений).
- **{command:aif_cross_mute, cross_id:0x01234567}** - Ставит на паузу кроссировку с идентификатором **cross_id**, т.е. перестает пересылать данные между двумя аудио-интерфейсами.
    - Ответ: **{response:ok, command:aif_cross_mute, cross_id:0x01234567}**.
- **{command:aif_cross_unmute, cross_id:0x01234567}** - Снимает с паузы кроссировку с идентификатором **cross_id**, т.е. возобновляет пересылку данных между двумя аудио-интерфейсами.
    - Ответ: **{response:ok, command:aif_cross_unmute, cross_id:0x01234567}**.
- **{command:netstream_create, dest_ip:1.2.3.4, dest_port:6007, mode:duplex, encoding:g711alaw, protocol:rtp, float_ip:true, rx_payload_type:0, tx_pyload_type:0, rx_telephony_payload_type:101, tx_telephony_payload_type:101, timeout:15}** - Создает новый сетевой поток исходя из следующего:

- **dest_ip** и **dest_port** - указывают куда слать RTP трафик;
- **mode** - режим работы из доступных: **duplex**, **send**, **receive**;
- **encoding** - какой из кодеков использовать, доступны: **g711ulaw**, **g711alaw**, **raw16**;
- **protocol** - выбор протокола инкапсуляции, на данный момент поддерживается только **rtp**;
- **float_ip** - если **true**, то в процессе работы будет разрешена подмена dest_ip/dest_port на адрес, с которого приходят RTP пакеты;
- **rx_payload_type** - задает номер payload-а для получаемых RTP пакетов, если получаемый пакет содержит номер отличный от этого и от **rx_telephony_payload_type**, то такой пакет будет скипнут;
- **tx_payload_type** - задает номер payload-а для отправляемых в сеть RTP пакетов;
- **rx_telephony_payload_type** - задает номер payload-а для принимаемых телефонных событий (по дефолту: 101);
- **tx_telephony_payload_type** - задает номер payload-а для отправляемых в сеть телефонных событий (по дефолту: 101);
- **timeout** - если приемник не получает пакетов из сети в течение указанного времени (в секундах), то данный сетевой поток помечается как дохлый, о чем сообщается специальным асинхронным событием (см. ниже), по умолчанию значение этого таймаута устанавливается в ноль и контроль приема не ведется;
- Все параметры являются опциональными, значения по умолчанию: **dest_ip:0.0.0.0, dest_port:0, mode:duplex, encoding:g711ulaw, protocol:rtp, float_ip:true**.
- Ответ: **{response:ok, command:netstream_create, dest_ip:0.0.0.0, dest_port:0, mode:duplex, encoding:g711alaw, protocol:rtp, float_ip:true, rx_payload_type:8, tx_pyload_type:8, rx_telephony_payload_type:101, tx_telephony_payload_type:101, timeout:15, stream_id:0x1e36a18, local_ip:192.168.174.154, local_port:55496, mapped_ip:217.116.57.130, mapped_port:55496}**, где:
- **stream_id** - идентификатор созданного сетевого потока;
- ** *_payload_type ** - идентификаторы полезной нагрузки (зависит от кодека и вычисляется автоматически, не указано явно);
- **local_ip** и **local_port** - адрес локального UDP сокета;

    - **mapped_ip** и **mapped_port** - адрес как нас видно из под NAT-а (запрашивается через STUN).
- **Команды установки типа полезной нагрузки**:
    - **{command:netstream_set_tx_payload_type, payload_type:88, stream_id:0x2a6a90}**,
    - **{command:netstream_set_rx_payload_type, payload_type:33, stream_id:0x2a6a90}**,
    - **{command:netstream_set_tx_telephony_payload_type, payload_type:55, stream_id:0x2a6a90}**,
    - **{command:netstream_set_rx_telephony_payload_type, payload_type:99, stream_id:0x2a6a90}**.
- **{command:netstream_start_all, stream_id:0x1e36a18}** - Снимает сетевой поток с паузы, запускает работу в обоих направления: прием и передачу.
- **{command:netstream_stop_all, stream_id:0x1e36a18}** - Ставит сетевой поток на паузу, останавливает прием и передачу.
- **{command:netstream_start_sending, stream_id:0x1e36a18}** - Снимает с паузы часть сетевого потока, ответственную за передачу данных в сеть, приемная часть - без изменений.
- **{command:netstream_start_receiving, stream_id:0x1e36a18}** - Снимает с паузы часть сетевого потока, ответственную за прием данных из сети, передающая часть - без изменений.
- **{command:netstream_stop_sending, stream_id:0x1e36a18}** - Ставит на паузу часть сетевого потока, ответственную за передачу данных в сеть, приемная часть - без изменений.
- **{command:netstream_stop_receiving, stream_id:0x1e36a18}** - Ставит на паузу часть сетевого потока, ответственную за прием данных из сети, передающая часть - без изменений.
- **{command:netstream_set_dest, stream_id:0x1e36a18, dest_ip:1.2.3.4, dest_port:1234}** - Настраивает сетевой поток на передачу данных в заданном направлении.
- **{command:netstream_destroy, stream_id:0x1e36a18}** - Удаляет сетевой поток, указанный в параметре **stream_id** и высвобождает все занимаемые им ресурсы. Слинкованный аудио-интрефейс при этом остается в прежнем состоянии.
- **{command:netstream_link_netstream, stream_id:0x1e36a18, other_stream_id:0x12345678}** - Делает так, чтобы все входящие RTP пакеты для потока **stream_id** переправлялись в поток **other_stream_id** и

уходили наружу. Т.е. производится однонаправленная линковка двух потоков. Чтобы сделать двунаправленную линковку требуется выполнить эту же команду, но поменять идентификаторы потоков местами.

- **{command:netstream_unlink_netstream, stream_id:0x1e36a18}** - Делает так, чтобы указанный в **stream_id** поток больше не был слинкован с другим потоком, т.е. будет сам обрабатывать полученные данные. Если имеется два потока симметрично слинкованных друг в друга, то для их полного разъединения требуется выполнить эту команду повторно для второго потока.
- **{command:netstream_link_aif, stream_id:0x1e36a18, aif_name:FrontSide, aec_enable:true}** - Подлинковывает аудио-интерфейс к указанному потоку, т.е. делает так, что все данные приходящие из сети по данному потоку будут декодироваться и проигрываться на слинкованном аудио-интерфейсе, и наоборот - все что записывается аудио-интерфейсом будет закодировано и отправлено в сеть. Параметр **aec_enable** если установлен в **true** указывает на то, что на данном аудио-интерфейсе нужно включить алгоритм **AEC**.
- **{command:netstream_set_params, stream_id:0x1e36a18, dest_ip:1.1.1.1, dest_port:111, tx_payload_type:8, rx_payload_type:8, tx_telephony_payload_type:101, rx_telephony_payload_type:102}** - Устанавливает текущие параметры для сетевого потока. При этом, нужно иметь в виду, что и **rx_payload_type** и **tx_payload_type** жестко привязаны к типу аудио-кодека, т.е. смена этих значений приведет к смене используемых кодеков. В указанном примере и для приема, и для передачи использован кодек 8 (PCMA). Если какой-то из указанных параметров отсутствует, то его значение не меняется для заданного потока.

Некоторые замечания:

- Теоретически, сетевой поток может быть одновременно слинкован с другим потоком и каким-нибудь аудио-интерфейсом. В этом случае, приоритетным является линковка с другим потоком, т.е. приходящие данные из сети по данному потоку будут без изменений уходить через другой поток, не попадая в аудио-интерфейс. Но, при этом, данные поступающие из аудио-интерфейса будут закодированы и отправлены в сеть через этот поток. Получается эдакое раздвоение личности потока.

Насколько это полезно с практической точки зрения выяснится в процессе эксплуатации.

- На слинкованных сетевых потоках декодирования трафика не проводится, **sequence number** и **timestamp** не проверяется, все что приходит на вход будет тут же тупо отправлено на выход.
- Не забываем делать **netstream_start_all** или **netstream_start_* **, иначе интерфейс будет молча стоять на паузе, даже будучи слинкованным.
- Класс **MediaInterfacess** на данный момент умеет принимать и парсить только файлы формата **RIFF/WAVE, PCM16, 8000 kHz, mono**.
- Кросс к которому подлинкован медиа-интерфейс удаляется автоматически при завершении медиа-интерфейса (конец файла, ошибка передачи по сети и т.д.), также за ним удаляется и сам медиа-интерфейс. Сделано это для удобства программирования, GUI не требуется отслеживать состояние проигрывателя файлов, но он по завершению проигрывания получит асинхронное событие **cross_playback_end**.
- Кросс к которому подлинкован аудио-интерфейс сам не удаляется, а может быть разрушен по команде **aif_uncross**.

3.6.3. Асинхронные события модуля AUDIO_STREAMER

Модуль AUDIO_STREAMER генерирует следующие асинхронные события, которые не являются прямым результатом исполнения команд:

- **{response:dead_stream, stream_id:0x2115b00, last_received:113695273, timestamp:469512751, timeout:15}** - Информирует пользователя о том, что в данный момент времени имеется сетевой поток, у которого истек таймаут в приемнике, т.е. поток не принимает данных из сети более чем указанное время таймаута. Тут есть важный момент: если поток стоит на паузе, то вычисление таймаута не ведется и начинается только после снятия потока с паузы.
- **{response:cross_playback_end, cross_id:0xa35970, url:https://www.fabmicro.ru/pub/8000Hz-16bits-1000Hz.wav, timestamp:4082656904}** - Информирует пользователя о том, что в данный момент времени существует кросс с медиа-интерфейсом, который закончил свое существование (полностью прочитан файл или возникла ошибка его чтения/передачи по сети). Никакие действия от пользователя не требуются, данный кросс и прилинкованный к нему медиа-интерфейс будут разрушены автоматически, использовать их больше нельзя.

- **{response:netstream_playback_end, stream_id:0xa35970, url:https://www.fabmicro.ru/pub/8000Hz-16bits-1000Hz.wav, timestamp:4082656904}** - Информирует пользователя о том, что в данный момент времени существует сетевой медиапоток с медиа-интерфейсом, который закончил свое существование (полностью прочитан файл или возникла ошибка его чтения/передачи по сети). Никакие действия от пользователя не требуются, медиа-интрефейс будет автоматически отлинкован и разрушен, а сетевой поток перейдет в нормальный режим работы.
- **{response:netstream_telephony_event, stream_id:0xc9ea70, event:6, volume:10, duration:200, rtp_timestamp: 107360, timestamp:3302779502}** - Информирует о том, что по сетевому потоку **stream_id** поступило телефонное событие согласно RFC-4733/RFC-2833. События с номера 0-16 это DTMF тоны (10 - это ** * **, 11 - ** # **).

3.6.4. Полезные примеры использования модуля AUDIO_STREAMER

Пример I: Исходящий SIP вызов (INVITE)

```
// Создаем сетевой поток для использования в исходящем SIP вызове (INVITE).
foo user TEXT:{command:netstream_create, timeout:15}
AUDIO_STREAMER user TAG:AUDIO_STREAMER_RESP TEXT: {response:ok,
command:netstream_create, dest_ip:0.0.0.0, dest_port:0, mode:duplex,
encoding:g711ulaw, protocol:rtp, float_ip:true, payload_type:0, timeout:15,
stream_id:0xe9ea18, local_ip:192.168.174.154, local_port:37983,
mapped_ip:217.116.57.130, mapped_port:37983}

// Устанавливаем IP/Port получателя нашего трафика после того как прибыл "180
Trying" с SDP содержащей IP/Port
foo user TEXT:{command:netstream_set_dest, stream_id:0xe9ea18,
dest_ip:192.168.168.103, dest_port:6007}
AUDIO_STREAMER user TAG:AUDIO_STREAMER_RESP TEXT: {response:ok,
command:netstream_set_dest, stream_id:0xe9ea18, dest_ip:192.168.168.103,
dest_port:6007}

// Запускаем сетевой поток.
foo user TEXT:{command:netstream_start_all, stream_id:0xe9ea18}
AUDIO_STREAMER user TAG:AUDIO_STREAMER_RESP TEXT: {response:ok,
command:netstream_start_all, stream_id:0xe9ea18}

// Линкуем аудио-интерфейс после того как соединение установлено: получили "200
OK".
foo user TEXT:{command:netstream_link_aif, aif_name:FrontSide,
stream_id:0xe9ea18}
AUDIO_STREAMER user TAG:AUDIO_STREAMER_RESP TEXT: {response:ok,
command:netstream_link_aif, stream_id:0xe9ea18, aif_name:FrontSide}

// Разрушаем сетевой поток после разрыва соединения: пришел CANCEL или BYE.
foo user TEXT:{command:netstream_destroy, stream_id:0xe9ea18}
AUDIO_STREAMER user TAG:AUDIO_STREAMER_RESP TEXT: {response:ok,
command:netstream_destroy, stream_id:0xe9ea18}
```

```
// Ставим аудио-интрефейс на паузу (не обязательно).
foo user TEXT:{command:aif_mute, aif_name:FrontSide}
AUDIO_STREAMER user TAG:AUDIO_STREAMER_RESP TEXT: {response:ok,
command:aif_mute, aif_name:FrontSide}
```

3.6.5. Дополнения

- Все команды в AudioStreamer могут принимать опциональный параметр "id", который тут же возвращается с ответом, чтобы можно было идентифицировать ответ в общем потоке команд, по аналогии с XMPP.
- В UDPSTUNSocket() добавлен параметр четности порта и номер желаемого порта. RTP сокет биндится в попытке найти четный порт из 20 попыток.
- У команды netstream_create появился опциональный параметр "local_port", который задает желаемый порт для RTP.

Фрагмент лога:

```
root@sip002emmc:/home/fabmicro/SIPHomePhone# ./audio_streamer
./audio_streamer - an Mrunner module to stream audio through RTP protocol, built
on: Aug 7 2020 at 20:38:18.
Copyright (C) 2020, Fabmicro, LLC., Tyumen, Russia.

Loading and parsing config file: audio_streamer.json
ZSyslog() will syslog to 217.116.57.130:514
SetStun() host = stun.12connect.com, port = 3478
GuessMyHostIP() Outbound iface: addr = 192.168.170.163, broadcast =
192.168.170.255, iface = eth0
UDPSocket() socket created.
UDPSocket::GetInterfaceAddress addr.Ptr=192.168.170.163, addr=192.168.170.163
UDPSocket::GetInterfaceAddress port.Ptr=60376, port=60376
StartLogging() syslog facility initialized, bound to 192.168.170.163:60376,
syslog host: 217.116.57.130:514
ZConnection::Start() SYSLOG initialized
AudioPlayer::AudioPlayer(FrontSide/plughw:CARD=sunxicodec) created.
AudioRecorder::AudioRecorder(FrontSide/plughw:CARD=sunxicodec) created.
AudioInterfaces::Add() New AIF added, aif = 0x4d94e8, name = FrontSide, playback
= plughw:CARD=sunxicodec, record = plughw:CARD=sunxicodec
AudioPlayer::AudioPlayer(BackSide/plughw:CARD=CODEC) created.
AudioRecorder::AudioRecorder(BackSide/plughw:CARD=CODEC) created.
AudioInterfaces::Add() New AIF added, aif = 0x4dcab0, name = BackSide, playback
= plughw:CARD=CODEC, record = plughw:CARD=CODEC
AudioPlayer::AudioPlayer(Pulse/pulse) created.
AudioRecorder::AudioRecorder(Pulse/pulse) created.
AudioInterfaces::Add() New AIF added, aif = 0x4dff70, name = Pulse, playback =
pulse, record = pulse
AUDIO_STREAMER core TEXT:ReadyForJob

T TEXT:{command:netstream_create}
UDPSTUNSocket() Attempt: 0, socket: 6, port: 60508
UDPSTUNSocket() socket opened, resolving stun: stun.12connect.com ...
UDPSTUNSocket() STUN: 77.72.169.212:3478
STUN: changed addr: 4D48A9D5, port: 3479
```

```
STUN: port1 = 60508, port2 = 60508, delta = 0
STUN: LocalPort = 60508
GuessMyHostIP() Outbound iface: addr = 192.168.170.163, broadcast =
192.168.170.255, iface = eth0
UDPSTUNSocket() Local: 192.168.170.163:60508

NetworkStreamTransmitter::NetworkStreamTransmitter() transmitter 0x4e4fe0
created.
NetworkStreamReceiver::NetworkStreamReceiver(0x4e3570) receiver 0x4e4990
created..
NetworkStreams::Add() New NetworkStream 0x4e3570 has been added.
UDPSTUNSocket::GetInterfaceAddress addr.Ptr=192.168.170.163,
addr=192.168.170.163
UDPSTUNSocket::GetInterfaceAddress port.Ptr=60508, port=60508
AUDIO_STREAMER user TAG:AUDIO_STREAMER_RESP TEXT: {response:ok,
command:netstream_create, dest_ip:0.0.0.0, dest_port:0, mode:duplex,
encoding:g711ulaw, protocol:rtp, float_ip:true, rx_payload_type:0,
tx_pyload_type:0, rx_telephony_payload_type:101, tx_telephony_payload_type:101,
timeout:0, stream_id:0x4e3570, local_ip:192.168.170.163, local_port:60508,
mapped_ip:217.116.57.130, mapped_port:60508, id:"<NONE>"}


T TEXT:{command:netstream_create, local_port:2226, id:some_id}
UDPSTUNSocket() Attempt: 0, socket: 10, port: 2226
UDPSTUNSocket() socket opened, resolving stun: stun.12connect.com ...
UDPSTUNSocket() STUN: 77.72.169.210:3478
STUN: changed addr: 4D48A9D3, port: 3479
STUN: port1 = 2226, port2 = 2226, delta = 0
STUN: LocalPort = 2226
GuessMyHostIP() Outbound iface: addr = 192.168.170.163, broadcast =
192.168.170.255, iface = eth0
UDPSTUNSocket() Local: 192.168.170.163:2226

NetworkStreamTransmitter::NetworkStreamTransmitter() transmitter 0x4e6630
created.
NetworkStreamReceiver::NetworkStreamReceiver(0x4e4b40) receiver 0x4e7178
created..
NetworkStreams::Add() New NetworkStream 0x4e4b40 has been added.
UDPSTUNSocket::GetInterfaceAddress addr.Ptr=192.168.170.163,
addr=192.168.170.163
UDPSTUNSocket::GetInterfaceAddress port.Ptr=2226, port=2226
AUDIO_STREAMER user TAG:AUDIO_STREAMER_RESP TEXT: {response:ok,
command:netstream_create, dest_ip:0.0.0.0, dest_port:0, mode:duplex,
encoding:g711ulaw, protocol:rtp, float_ip:true, rx_payload_type:0,
tx_pyload_type:0, rx_telephony_payload_type:101, tx_telephony_payload_type:101,
timeout:0, stream_id:0x4e4b40, local_ip:192.168.170.163, local_port:2226,
mapped_ip:217.116.57.130, mapped_port:2226, id:"some_id"}
```

3.7. Модуль обработки видеопотока (VIDEO_STREAMER)

Данный модуль осуществляет всю работу с аппаратурой по захвату видео: камера, ISP, аппаратный энкодер и декодер.

3.7.1. JSON API модуля VIDEO_STREAMER

Модуль поддерживает следующие команды:

```
{"command":"netstream_create", local_ip:"0.0.0.0", local_port:6005,
dest_ip:"1.1.1.1", dest_port:1234, mode:duplex, float_ip:true, timeout:15,
stun_host:"0.0.0.0", stun_port:"3478", rx_payload_type:96, tx_payload_type:96,
id:""},
{"response":"ok", "command":"netstream_create", local_ip:"1.2.3.4",
local_port:6005, dest_ip:"1.1.1.1", dest_port:1234, stream_id:0x123456,
mapped_ip:1.1.1.1, mapped_port:1111, id:"", rx_payload_type:96,
tx_payload_type:96},
{"response":"error", "command":"netstream_create", "error":"cannot_open_socket",
local_ip:"1.2.3.4", local_port:6005, dest_ip:"1.1.1.1", dest_port:1234, id:""}
```

Параметр **mode** может быть **duplex**, **send**, **receive**.

Для тестов (чтобы не линковать камеру отдельно) можно задавать сразу camera_name, camera_format, camera_bitrate, frame_rate, codec_key_interval

rx_payload_type, tx_payload_type, timeout, float_ip, local_port также необязательны, их можно задать не сразу, а позже командой:

```
{"command":"netstream_set_parameter", stream_id:1234, dest_ip:"1.1.1.1",
dest_port:1234, rx_payload_type:96, tx_payload_type:96, camera_format:"640x480",
camera_bitrate:"1024", frame_rate:"4", codec_key_interval:"15", id:""},
{"response":"ok", stream_id:1234, "command":"netstream_set_dest",
dest_ip:"1.1.1.1", dest_port:1234, rx_payload_type:96, tx_payload_type:96},

{"response":"error", "error":"stream_not_found", "command":"netstream_set_dest",
stream_id:1234}
```

Необязательно указывать все параметры в этой команде, можно установить только rx_payload_type, например.

Для записи файла JPEG предусмотрена команда 'write_jpeg'. Она приостанавливает текущий стрим в случае, если для записи требуется камера в другом разрешении. Команда открывает камеру в новом разрешении, делает снимок и возобновляет работу стрима.

```
{command:"write_jpeg", camera_name:"Internal", file_name:"test.jpeg",
quality:80, id:"aa"}
{"response":"ok", "command":"write_jpeg", camera_name:"Internal",
file_name:"test.jpeg", id:"aa"}
{"response":"error", "error":"prev_save_not_completed", "command":"write_jpeg",
id:"aa"}
```

Если камера стрима не была открыта в том же разрешении, то она открывается заново и после открытия камеры возвращает еще одно подтверждение об открытии камеры:

```
{response:ok, camera_open_command:"write_jpeg", camera_name:"Internal", id:"aa"}
{response:error, camera_open_command:"write_jpeg", camera_name:"Internal",
id:"aa"}
После записи картинки или неудачи выозвращается
{response:done, command:"write_jpeg", id:"aa"}
```

Т.к. камера открывается в течении 1-2секунды, то камеру для стримов можно открыть заранее командой:

```
{command:"camera_open", camera_name:"Internal", id:"aa"}
{"response":"ok", "command":"camera_open", camera_name:"Internal", id:"aa"}
{"response":"error", "error":"prev_camera_open_in_process",
"command":"camera_open", id:"aa"}
```

Если камера еще не успела открыться, а мы уже попытались ее использовать (например командой netstrean_link_camera), то возвращаемая команда поменяется:

```
{response:ok, camera_open_command:"netstrean_link_camera",
camera_name:"Internal", id:"aa"}
{response:error, camera_open_command:"netstrean_link_camera",
camera_name:"Internal", id:"aa"}
```

Для включения окон просмотра и выставления цвета фона(используются из GUI) поддерживаются команды:

```
{"command":"show_decoder", left:0, top:0, right:100,bottom:100, id:"aa"}
{"response":"ok", "command":"show_decoder", id:"aa"}
{"response":"error", "error":"Layer error", "command":"show_decoder", id:"aa"}

{"command":"hide_decoder", id:"aa"}
{"response":"ok", "command":"hide_decoder", id:"aa"}

{"command":"show_camera", left:0, top:0, right:100,bottom:100, id:"aa"}
{"response":"ok", "command":"show_camera", id:"aa"}
{"response":"error", "error":"Layer error", "command":"show_camera", id:"aa"}

{"command":"hide_camera", id:"aa"}
{"response":"ok", "command":"hide_camera", id:"aa"}

{"command":"show_background", background:"0xffffffff", id:"aa"}
{"response":"ok", "command":"show_background", id:"aa"}
{"response":"error", "error":"Layer error", "command":"show_background",
id:"aa"}

{"command":"hide_background", id:"aa"}
{"response":"ok", "command":"hide_background", id:"aa"}
```

Камера линкуется командой:

```
{"command":"netstream_link_camera", stream_id:1234, "camera_name":"Internal",
"camera_format":"640x480", camera_bitrate:"1024", frame_rate:"4",
codec_key_interval:"15"},
{"response":"ok", "command":"netstream_link_camera", stream_id:1234,
"camera_name":"Internal"},
{"response":"error", "error":"stream_not_found",
"command":"netstream_link_camera", stream_id:1234},

{"command":"netstream_unlink_camera", stream_id:1234},
{"response":"ok", "command":"netstream_unlink_camera", stream_id:1234},
{"response":"error", "error":"stream_not_found",
"command":"netstream_unlink_camera", stream_id:1234},
```

Запуск стрима на прием/передачу/или на то, и другое осуществляется командами:

```
{"command":"netstream_start_all",
stream_id:1234},/*netstream_start_sending,netstream_start_receiving*/
{"response":"ok", stream_id:1234, "command":"netstream_start_all"},
{"response":"error", "error":"stream_not_found",
"command":"netstream_start_all", stream_id:1234},

{"command":"netstream_stop_all",
stream_id:1234},/*netstream_stop_sending,netstream_stop_receiving*/
{"response":"ok", stream_id:1234, "command":"netstream_stop_all"},
{"response":"error", "error":"stream_not_found", "command":"netstream_stop_all",
stream_id:1234},
```

Дополнительные команды:

```
{"command":"netstream_link_netstream", stream_id:1234, other_stream_id:1234},
{"response":"ok", "command":"netstream_link_netstream", stream_id:1234,
other_stream_id:1234},
{"response":"error", "error":"stream_not_found",
"command":"netstream_link_netstream", stream_id:1234, other_stream_id:1234},

{"command":"netstream_unlink_netstream", stream_id:1234},
{"response":"ok", "command":"netstream_unlink_netstream", stream_id:1234},
{"response":"error", "error":"stream_not_found",
"command":"netstream_unlink_netstream", stream_id:1234},

{command:netstream_destroy, stream_id:1234},
{"response":"ok", "command":"netstream_destroy", stream_id:1234},
{"response":"error", "error":"stream_not_found", "command":"netstream_destroy",
stream_id:1234},
```

3.7.2. Профайлер модуля VIDEO_STREAMER

Если перед компиляцией установить export MICROPROFILE=1, то в код добавится встроенный профайлер. К нему можно коннектиться http://192.168.x.x:1338. И в реалтайм смотреть загрузку потоков, или счетчики.

3.8. Модуль обеспечения конфигурацией (CONFIGURATOR)

Модуль **CONFIGURATOR** (далее просто **конфигуратор**) предназначен для обслуживания основного конфигурационного файла Вызывной Панели. По замыслу, модуль запускается самым первым, находит и подготавливает рабочий конфиг, который будет использован всеми остальными модулями.

3.8.1. Концепция

В результате обсуждений, консультаций, собственных измышлений, проб и ошибок было принято решение использовать следующую концепцию настройки вызывной панели:

- Настройка вызывной панели НЕ производится с сенсорного экрана, так как:
    - Это небезопасно и требует дополнительных усилий по авторизации "настройщика", что сильно усложняет всю систему и делает её не пригодной для реальной эксплуатации!
    - Требует дополнительных усилий по реализации ряда экранных меню, реализации наэкранной буквенно-цифровой клавиатуры и прочих сложностей полнофункциональной графической оболочки.
    - Не может быть использовано в версии ВП без экрана.
- Настройка ВП может быть осуществлена по Сети либо в автоматическом режиме (DHCP provisioning), либо через встроенный Web сервер. Также возможны варианты: автоматического переноса конфига с SD-карты, доступ через консоль или SSH - но все это для «продвинутых» настройщиков.
- Все параметры настройки всех модулей, за исключением модуля CONFIGURATOR, хранятся в формате JSON в файле **all.json,** путь к которому передается из **SunBus** (ex. Mrunner) всем модулям при их запуске в качестве первого параметра командной строки.
- Файл all.json располагается в read-write файловой системе в т.н. каталоге-спуле (**/var/configs/all.json**) и является символической ссылкой на один из множества конфигурационных файлов, который является текущим конфигом на данный момент времени.
- Содержимое конфига может изменяться в зависимости от режимов работы устройства и пожеланий администратора.

- У конфигурационного файла обязательно присутствует серийный номер - 64-х битное число в символьном выражении (для этой цели предлагается использовать UNIX time), т.е. настоящее имя конфига всегда имеет вид: **all.json.XXXXXX**, где XXXXXX - серийный номер файла.
- У конфигурационного файла также имеется MD5 слепок, который используется для проверки файла на повреждения. Слепок хранится в этом же каталоге-спуле в файле: **all.json.XXXXXX.md5**
- У модуля CONFIGURATOR имеется свой независимый конфигурационный файл **configurator.json,** который не подлежит изменению, поэтому он и вынесен отдельно и находится в **read-only** файловой системе. **Если этот файл повредится, то вся система перестанет работать, так как модули не получат правильного конфигурационного файла!**
- Вызывная панель может функционировать в двух режимах: **инфраструктурном** и **самостоятельном**.
- В инфраструктурном режиме все настройки (т.е. содержимое файла **all.json**) передаются из Сети автоматически, для чего конфигуратор периодически производит опрос API сервера. Если данные от сервера получены и успешно валидированы, то они сохраняются в спуле как новый конфиг, для которого тут же вычисляется и сохраняется MD5 слепок и создается ссылка all.json указывающая на этот файл.
- В самостоятельном режиме конфигуратор никого не опрашивает, а находит в спуле самый свежий конфиг и так же устанавливает на него ссылку.
- Если конфигуратор не нашел подходящего конфига, то он устанавливает ссылку на дефолтный конфиг, всегда расположенный в read-only файловой системе.
- Конфигуратор при запуске всегда первым делом находит самый свежий конфиг и устанавливает ссылку, после чего приступает к "добыванию" нового конфига.

3.8.2. Инфраструктурный режим работы конфигуратора

По умолчанию конфигуратор устанавливает активный дефолтный конфиг, который не содержит ссылки на внешний сервер, а значит никаких запросов на изменения конфига не осуществляется.

Для того, что бы конфигуратор начал инициировать запросы в сервер, необходимо сообщить ему адрес сервера. Сделать это можно двумя способами:

- Внести изменения в основной конфигурационный файл (используя Web интерфейс, SSH или консоль) и установить значение параметра **confg_url** в секции **Config** указав URL адрес сервера.
- Передав по DHCP опцию содержащую строку адреса сервера, номер (или имя) опции должно совпадать со значением параметра **dhcp_option_config** в файле настроек конфигуратора (в файле **configurator.json**). Для обмена по DHCP предлагается использовать **опцию 161** в строковом виде, примеры DHCP с её использованием даны ниже.

Если конфигуратор получил адрес (URL) сервера, то он при запуске производит его опрос следующим образом:

- Если строка адреса начинается с префиксов ftp:// или tftp://, то к URL строке адреса прибавляется символ "-" за которым следует строка содержащая **SID** устройства и производится запрос на выкачивание статического файла, содержимое которого рассматривается как потенциальный конфиг. Из этого конфига извлекается параметр **Config/serial** и используется как серийный номер нового конфига. Данные сохраняются в соответствующий файл, для которого вычисляется MD5 слепок, слепок сохраняется рядом. При таком запросе авторизация **не производится**, т.е. предполагается, что администратор сети обеспечил доступ к безопасному внутреннему файл-серверу конфигураций. Этот механизм полностью позаимствован у IP-телефонов Cisco и имеет широкое хождение в корпоративных сетях, но не пригоден для использования на публичных сетях.
- Если строка адреса начинается с префикса http:// или https:// , то производится следующая последовательность запросов:
    - Прежде всего конфигуратор выясняет есть ли у него авторизационная кука (**auth_cookie**), для этого он заглядывает в файл указанный в параметре **auth_cookie_file_name** внутри спула -

первая строка текста в том файле считывается и используется далее как авторизационная кука.

- Если авторизационная кука отсутствует, а это произойдет при первом попадании вызывной панели в инфраструктурный режим, то конфигуратор делает HTTP или HTTPS запрос указанной выше URL с целью получить авторизационную куку по имеющемуся у него **SID** и **паролю**. SID - неизменяемый параметр, он считывается из CPU и устанавливается при изготовлении "железа", а вот пароль может быть задан через Web или по SSH. Пароль считывается из файла указанного в параметре **password_file_name,** расположенного внутри спула. Если файл с паролем отсутствует, то в качестве пароля передается пустая строка. Помимо SID и пароля при запросе авторизационной куки передается локальный IP адрес, на всякий случай.
- Получив авторизационную куку от сервера, конфигуратор сохраняет её в файл в спуле и далее всегда использует её для запроса конфига. Если по какой-то причине файл с авторизационной куой пропал, то процедура получения куки выполняется снова.

- Полученный конфиг анализируется, верифицируется (проверяется присутствие важных секций), из него извлекается серийный номер (параметр **Config/serial**) и сохраняется в спуле.
- Вычислется MD5 слепок и тоже сохраняется в спуле.
- Как только конфигуратор получил конфиг, серийный номер которого **больше** чем номер текущего конфига, он завершается, тем самым сигнализируя SunBus-у о том, что пришла пора перезапустить все модули (что бы они перечитали новый конфиг).
- Все HTTP или HTTPS запросы к серверу передаются в формате JSON, их список приведен ниже.

В конфигуратор добавлена проверка на TEXT:AllModulesStarted. Если к моменту завершения модуля CONFIGURATOR данное событие не наступило, то ЭТО сигнализирует о том, что текущий конфиг неправильный (т.е. один из модулей его не смог обработать), текущий конфиг маркируется как BAD - создается файл all.json.XXXXXXXX.bad, а следовательно при следующем старте данный конфиг игнорируется.

И наоборот, как только приходит TEXT:AllModulesStarted, создается маркерный файл all.json.XXXXXXX.good.

Данная функция позволит быстро узнавать, правильные конфиги или нет, а также позволит исключить повторное использование данных от сервера с невалидным конфигом.

3.8.3. Примеры работы конфигурационного API

Формат запроса авторизационной куки по известному паролю:

```
PUT /siphomephone/config.cgi HTTP/1.1
Host: www.fabmicro.ru
User-Agent: SIPHomePhone/1.3
Accept: application/json
Content-Type: application/json
charset: utf-8
Content-Length: 156

{"command":"request_auth_cookie", "sid":"16254251504E54313530303009410ACD",
"password":"4321", "local_ip":"192.168.174.160", "timestamp":"1291445126600288"}
```

Формат ответа на запрос куки:

```
HTTP/1.1 200 OK
Date: Tue, 04 Aug 2020 22:32:05 GMT
Server: Apache/2.4.12 (FreeBSD) OpenSSL/1.0.1l
Transfer-Encoding: chunked
Content-Type: application/json; charset=utf-8

{command:"request_auth_cookie", response:"ok",
"auth_cookie":"sdfSdfsdFSDSDFSDFsdfSDFSDFSDfSDFfSD"}
```

или

```
{command:"request_auth_cookie", response:"error", text:"authorization_failed",
"sid":"16254251504E54313530303009410ACD",}
```

Формат запроса конфига по известной авторизационной куки:

```
PUT /siphomephone/config.cgi HTTP/1.1
Host: www.fabmicro.ru
User-Agent: SIPHomePhone/1.3
Accept: application/json
Content-Type: application/json
charset: utf-8
Content-Length: 208

{"command":"request_config", "last_serial":"1596661124",
"auth_cookie":"sdfSdfsdFSDSDFSDFsdfSDFSDFSDfSDFfSD",
"config_version":"SIPHomePhone/1.3", "local_ip":"192.168.174.160",
"timestamp":"1296085287642558"}
```

Формат ответа на запрос конфига:

```
HTTP/1.1 200 OK
Date: Tue, 04 Aug 2020 22:33:31 GMT
Server: Apache/2.4.12 (FreeBSD) OpenSSL/1.0.1l
```

```
Transfer-Encoding: chunked
Content-Type: application/json; charset=utf-8


{command:request_config, response:ok,
data:
  {
     // здесь расположен JSON текст конфига
  }
}

или

{"command":"request_config", response:"error", text:"authorization_failed",
"auth_cookie":"sdfSdfsdFSDSDFSDFsdfSDFSDFSDfSDFfSD"}
```

3.8.4. Примеры DHCP

ВАЖНО: Далее под DHCP клиентом понимается **dhclient** от **Internet Systems Consortium DHCP Client 4.3.5,** работающий под ОС Linux.

Под DHCP сервером понимается **dhcpd** от **Internet Systems Consortium DHCP Server 4.2.4-P2,** работающий под ОС FreeBSD.

Для того, чтобы DHCP клиент начал запрашивать какую-то дополнительную опцию от сервера, необходимо ему об этом сообщить в конфиге следующим образом:

```
option siphomephone-config code 161 = string;

send host-name = gethostname();

request subnet-mask, broadcast-address, time-offset, routers,
        domain-name, domain-name-servers, domain-search, host-name,
        dhcp6.name-servers, dhcp6.domain-search, dhcp6.fqdn, dhcp6.sntp-servers,
        netbios-name-servers, netbios-scope, interface-mtu,
        rfc3442-classless-static-routes, ntp-servers, siphomephone-config;
```

DHCP клиент обычно сохраняет все лизы в специальный файл, который располагается в: **/var/lib/dhcp/dhclient.eth0.leases**, ссылку на этот файл требуется обязательно указать в конфиге конфигуратора - параметр **dhcp_lease_file**. После чего конфигуратор будет подглядывать в этот файл при запуске.

Содержимое **lease** файла выглядит вот так:

```
lease {
  interface "eth0";
  fixed-address 192.168.174.160;
  option subnet-mask 255.255.255.0;
  option routers 192.168.174.1;
  option dhcp-lease-time 600;
  option dhcp-message-type 5;
```

```
  option siphomephone-config "https://www.fabmicro.ru/siphomephone/config.cgi";
  option domain-name-servers 217.116.57.130;
  option dhcp-server-identifier 192.168.174.1;
  option netbios-name-servers 192.168.169.1;
  option domain-name "fabmicro.ru";
  renew 2 2020/08/04 22:48:18;
  rebind 2 2020/08/04 22:52:53;
  expire 2 2020/08/04 22:54:08;
}
```

Что бы DHCP сервер выдавал дополнительную опцию, его об этом тоже требуется попросить, в файле **/usr/local/etc/dhcpd.conf**:

```
option option-161 code 161 = string;

subnet 192.168.169.0 netmask 255.255.255.0
{
        range 192.168.169.101 192.168.169.254;
        option broadcast-address 192.168.169.255;
        option routers 192.168.169.1;
        option domain-name-servers 217.116.57.130;
        option domain-name "fabmicro.ru";
        option tftp-server-name "77.242.111.106";
        option netbios-name-servers 192.168.169.1;
        option option-161 "https://www.fabmicro.ru/siphomephone/config.cgi";
        option dhcp-parameter-request-list 3, 6, 15, 44, 161;
}
```

3.9. Модуль взаимодействия с шиной Modbus (RS485_MASTER)

Модуль RS485_MASTER предназначен для взаимодействия с устройствами на шине RS-485 по протоколу **Modbus/RTU**. Данный протокол подразумевает полудуплексный обмен по последовательной шине (RS-485) в режиме **запрос-ответ**, при этом запрашивать может только одно устройство - **мастер**, каковым в нашем случае является вызывная домофонная панель.

Запросы в Modbus/RTU адресуются подчиненным устройствам (**слэйвам**), которых на шине может быть до **255** шт, каждое подчиненное устройство имеет свой однобайтовый адрес, начиная с **1** (адрес **0** предполагает широковещательный запрос, то есть **всем**).

Запросы в Modbus/RTU это способ считать или записать состояние 16-битных регистров (переменных) в подчиненных устройствах, каждый регистр имеет свой 16-битный номер уникальный в пределах одного устройства. Есть способ передавать и читать сроки символов, но это уже отхождение от стандарта и, скорее всего, не потребуется.

Согласно стандарту, регистры в Modbus/RTU могут быть нескольких видов: регистры ввода (input register), регистры хранения (hold register), регистры состояния (coil register). Все пространство номеров регистров 1-65535 разбивается на группы в зависимости от типа регистра, но для нас оно будет единым, а для выбора способа обращения к регистру мы будем использовать параметр **method = ['input', 'hold', 'coil']**, при этом указывая номер регистра в параметре **reg**.

Таким образом, JSON команды для модуля RS485_MASTER будут иметь следующий вид:

- **{command:read, addr:1, method:hold, reg:1234}** — по этой команде происходит чтение регистра хранения **1234** из подчиненного устройства с адресом **1**.
- **{command:write, addr:2, method:coil, reg:123, value:1}** - по этой команде происходит запись в регистр состояния **123** булевого значения **1** (true) в устройстве с адресом **2**.

На каждый такой запрос есть три варианта исхода: успех, ошибка и timeout. Формат ответа от модуля следующий:

- **{response:ok, command:write, addr:1, method:hold, reg:123, value:222}** - команда на запись в регистр хранения **123** в устройство с адресом **1** выполнена успешно. Значение регистра установлено в **222**.
- **{response:ok, command:read, addr:1, method:hold, reg:123, value:333}** - команда на чтение из регистра хранения **123** в устройстве с адресом **1** выполнена успешно. Текущее значение регистра: **333**.
- **{response:error, command:write, addr:1, method:hold, reg:123, value:222}** - команда на запись в регистр хранения **123** в устройстве с адресом **1** выполнена с ошибкой или НЕ выполнена вообще. Текущее значение регистра не изменилось.
- **{response:error, command:read, addr:1, method:hold, reg:123}** - команда на чтение из регистра хранения **123** в устройстве с адресом **1** выполнена с ошибкой или НЕ выполнена вообще. Значение регистра не известно.
- **{response:timeout, command:write, addr:1, method:hold, reg:123, value:222}** - команда на запись в регистр хранения **123** в устройство с адресом **1** НЕ выполнена по причине timeout-а (устройство не существует или не отвечает). Значение регистра НЕ установлено.
- **{response:timeout, command:read, addr:1, method:hold, reg:123}** - команда на чтение из регистра хранения **123** в устройстве с адресом **1** НЕ выполнена по причине timeout-а (устройство не существует или не отвечает). Значение регистра не известно.

Этот модуль потребуется для взаимодействия с аналоговым домофонным коммутатором. sip-domophone:домофонный_коммутатор.

3.9.1. Пример использования

Пример команды для чтения **SCRATCH** регистра аналогового домофонного коммутатора приведен ниже. Данную команду можно отправить через отладочный сокет утилитой **telnet**.

```
FOO: user TAG:RS485_MASTER_CMD TEXT:{command:read, addr:1, reg:123, method:hold}
RS485_MASTER user TAG:RS485_MASTER_RESP TEXT: {response:timeout, command:read,
method:hold, addr:1, reg:123}
```

Здесь нам модуль ответил ошибкой **timeout**, что означает отсутствие ответа от подчиненного устройства с адресом **1** в отведенное для этого время. Причиной такой ошибки может быть неверный адрес подчиненного устройства, неверные параметры последовательного порта (скорость передачи), ошибки на шине

(шина длинная и нетерминированная), сбои подчиненного устройства, обрыв. **Иногда имеет смысла сделать несколько попыток чтения!**

Запись в 32-х битный регистр хранения

Строго говоря, данная опция не совместима со стандартом Modbus/RTU, но будет полезна нам при отправке в домофонный коммутатор команды на коммутацию абонента за один прием, а не за два. Так как абонент адресуется двумя 16-битными словами (вектор E - единицы/столбцы и вектор D - десятки/строки матрицы коммутации), то требуется сделать два запроса по Modbus, что не очень удобно - ведь на каждый запрос нужно еще дождаться правильный ответ, а это увеличивает число состояний и, соответственно, код программы. Чтобы упростить, был придумал нестандартный метод запись сразу 32 бита в один регистр (**method:hold32**), который далее уже может интерпретироваться внутри устройства как угодно. Формат такой команды следующий:

- **{command:write, addr:2, method:hold32, reg:123, value:1234567890}** по этой команде происходит запись в 32-х битный регистр хранения **123** значения **1234567890** в устройстве с адресом **2**.

Значение в поле **value**, да и в любом другом целочисленном, может быть записано в шестнадцатиричной или восьмиричной форме, например: **0x499602D2** или **011145401322**.

Пример команды в коммутатор на коммутацию абонента может выглядеть как запись в регистр REG_CMD_CONNECT_NORMAL (с номером 11), который является 32-битным регистром хранения:

```
FOO: user TAG:RS485_MASTER_CMD TEXT:{command:write, addr:1, reg:11,
method:hold32, value:0x00010002}
```

При этом коммутатор должен скоммутировать абонента с матричным адресом 0x0001/0x0002, т.е. D=1, E=2.

3.10. Модуль отслеживания состояния абонентской трубки (HANDSET_STATE)

Предназначен для определения состояния аналоговой абонентской (домофонной) трубки.

Вызывная Панель предусматривает функционирование в аналоговых домофонных сетях, при этом всю работу по коммутации и контролю состояния абонентской трубки берет на себя специальное согласующее устройство - **Коммутатор**. Вызывная панель производит опрос состояния ранее скоммутированной абонентской трубки, общаясь с коммутатором путем посылки команд по RS-485 линии (протокол Modbus/RTU), используя для этого модуль **RS485_MASTER**.

Однако, иногда требуется подключить всего одну (или две запараллеленные) абонентские трубки. Сделать это можно без коммутатора, подцепив трубку напрямую к вызывной панели на аналоговую линию **LN**. В этом случае встает вопрос определения состояния абонентской трубки, одного из трех: **трубка висит**, **трубка снята**, **трубка снята и кнопка отпирания замка нажата**. Для решения этой частной задачи внутри вызывной панели аналоговая линия LN, через буфферный операционный усилитель, заведена на низко-скоростной АЦП (**Low Rate ADC** или **LRADC**). С помощью LRADC можно узнать текущее напряжение в линии, а значит и состояние абонентской трубки, так как трубка имеет разное сопротивление в различных состояниях.

Чтобы считать значение из LRADC был создан специальный модуль-драйвер для ядра Linux 3.4 - **sunxi_lradc.ko**. Этот драйвер представляет собой **input device** и регулярно рассылает значения, считываемые из LRADC с частотой 30 измерений в сек. Ниже приведен пример как посмотреть поток этих измерений с помощью утилиты **evtest**:

```
root@devuan:/home/fabmicro/SIPHomePhone# evtest /dev/input/event3
Input driver version is 1.0.1
Input device ID: bus 0x19 vendor 0x1 product 0x1 version 0x100
Input device name: "sunxi-lradc"
Supported events:
  Event type 0 (EV_SYN)
  Event type 4 (EV_MSC)
    Event code 3 (MSC_RAW)
Properties:
Testing ... (interrupt to exit)
Event: time 1589241741.468469, type 4 (EV_MSC), code 3 (MSC_RAW), value 19
Event: time 1589241741.468477, -------------- SYN_REPORT ------------
Event: time 1589241741.503020, type 4 (EV_MSC), code 3 (MSC_RAW), value 19
Event: time 1589241741.503024, -------------- SYN_REPORT ------------
```

```
Event: time 1589241741.537573, type 4 (EV_MSC), code 3 (MSC_RAW), value 19
Event: time 1589241741.537576, ---------- SYN_REPORT ------------
Event: time 1589241741.572123, type 4 (EV_MSC), code 3 (MSC_RAW), value 19
Event: time 1589241741.572126, ---------- SYN_REPORT ------------
Event: time 1589241741.606672, type 4 (EV_MSC), code 3 (MSC_RAW), value 19
Event: time 1589241741.606674, ---------- SYN_REPORT ------------
Event: time 1589241741.641227, type 4 (EV_MSC), code 3 (MSC_RAW), value 19
Event: time 1589241741.641230, ---------- SYN_REPORT ------------
Event: time 1589241741.675792, type 4 (EV_MSC), code 3 (MSC_RAW), value 19
Event: time 1589241741.675795, ---------- SYN_REPORT ------------
Event: time 1589241741.710348, type 4 (EV_MSC), code 3 (MSC_RAW), value 19
Event: time 1589241741.710350, ---------- SYN_REPORT ------------
```

Значение **value** задается в единицах отсчета ADC. Чтобы перевести его в вольты, нужно провести несложные вычисления, но в данном случае смысла в этих вычислениях нет.

Далее, чтобы основному приложению вызывной панели не утруждаться с вычислениями разных состояний, предлагается использовать модуль **HEADSET_STATE** в системе SunBus. Данный модуль открывает указанное в параметрах устройство ввода (input device) от SUNXI LRADC, читает **сырые** данные с АЦП и переводит их в состояние абонентской трубки, после чего размещает соответствующее сообщение в поток SunBus-a.

По умолчанию, модуль размещает сообщения только при смене состояния трубки. Однако, задав параметр **-t** можно указать модулю, чтобы он повторял сообщения о текущем состояние трубки с определенной регулярностью.

```
root@devuan:/home/fabmicro/SIPHomePhone# ./headset_state
./headset_state - an Mrunner module to decode headset state using Low Rate ADC.
Copyright (C) 2020, Fabmicro, LLC., Tyumen, Russia.

Openning LRADC input device /dev/input/event3 ...

Thresholds are: headset_threshold_onhook = 23, headset_threshold_offhook = 32,
headset_threshold_button_pressed = 35

HEADSET_STATE core TEXT:ReadyForJob
HEADSET_STATE user TAG:HEADSET_STATE_RESP TEXT: {headeset_state:2,
headset_state_text:onhook, adc:25, timestamp:1589240687.896704}
HEADSET_STATE user TAG:HEADSET_STATE_RESP TEXT: {headeset_state:3,
headset_state_text:offhook_button_released, adc:32, timestamp:1589240701.134454}
HEADSET_STATE user TAG:HEADSET_STATE_RESP TEXT: {headeset_state:2,
headset_state_text:onhook, adc:29, timestamp:1589240703.381010}
HEADSET_STATE user TAG:HEADSET_STATE_RESP TEXT: {headeset_state:3,
headset_state_text:offhook_button_released, adc:32, timestamp:1589240704.521848}
HEADSET_STATE user TAG:HEADSET_STATE_RESP TEXT: {headeset_state:2,
headset_state_text:onhook, adc:30, timestamp:1589240705.731686}
HEADSET_STATE user TAG:HEADSET_STATE_RESP TEXT: {headeset_state:3,
headset_state_text:offhook_button_released, adc:32, timestamp:1589240707.252616}
HEADSET_STATE user TAG:HEADSET_STATE_RESP TEXT: {headeset_state:4,
headset_state_text:offhook_button_pressed, adc:35, timestamp:1589240708.496734}
```

```
HEADSET_STATE user TAG:HEADSET_STATE_RESP TEXT: {headeset_state:3,
headset_state_text:offhook_button_released, adc:34, timestamp:1589240709.879054}
HEADSET_STATE user TAG:HEADSET_STATE_RESP TEXT: {headeset_state:4,
headset_state_text:offhook_button_pressed, adc:35, timestamp:1589240710.915934}
HEADSET_STATE user TAG:HEADSET_STATE_RESP TEXT: {headeset_state:3,
headset_state_text:offhook_button_released, adc:34, timestamp:1589240712.505804}
HEADSET_STATE user TAG:HEADSET_STATE_RESP TEXT: {headeset_state:4,
headset_state_text:offhook_button_pressed, adc:35, timestamp:1589240713.473603}
HEADSET_STATE user TAG:HEADSET_STATE_RESP TEXT: {headeset_state:3,
headset_state_text:offhook_button_released, adc:34, timestamp:1589240715.478188}
HEADSET_STATE user TAG:HEADSET_STATE_RESP TEXT: {headeset_state:2,
headset_state_text:onhook, adc:29, timestamp:1589240716.169463}
```

Расшифровка состояний:

- **no_power** - не подано питания на трубку (сигнал PCM_MUTE = ON), либо не произведена коммутация.
- **onhook** - трубка висит.
- **offhook_button_release** - трубка снята, идет разговор, но кнопка отпирания замка не нажата.
- **offhook_button_pressed** - трубка снята, кнопка отпирания замка **нажата**.

Для того, что бы подать питание на трубку, нужно вывести логичекий **0** в соответствующее GPIO:

```
echo 0 > /sys/class/gpio/gpio1_pc3/value
```

Отключается питание трубки выводом логической **1** в этот же порт GPIO.

3.11. Модуль датчика освещенности (AMBIENT_SENSOR)

Аппаратное обеспечение Вызывной Панели содержит датчик освещенности на основе фотодиода. Логика его работы очень проста - фотоны попадают на активный элемент фотодиода (p-n переход) и создают в нем направленное движение электронов, т.е. электрический ток. Чем выше освещенность, тем больше создаваемый ток. Фотодиод, через операционный усилитель, подключен к периферийному аналого-цифровому преобразователю (**ADC** микросхема **TI ADS1000**), который в свою очередь преобразовывает напряжение, создаваемое током на нагрузочном резисторе, в последовательность измерений.

Для того, чтобы считывать эти измерения был создан специальный драйвер **ads1000.ko**, который экспортирует **HWMON** интерфейс в Linux систему. В простейшем случае считать текущее значение ADC можно из специализированного файла: **/sys/class/hwmon/hwmon1/device/in0_input**, например:

```
root@devuan:/home/fabmicro/SIPHomePhone# cat
/sys/class/hwmon/hwmon1/device/in0_input
2966
```

Для унификации и упрощения работы с этим драйвером, был создан модуль **AMBIENT_SENSOR**. Функция модуля очень простая: непрерывно считывать значения из ADC к которому подключен датчик освещенности и отслеживать его состояние. На данный момент имеется три состояния:

- **unknown** - состояние датчика не известно (происходит при запуске модуля);
- **day** - сейчас очень ярко, т.е. "день";
- **night** - сейчас недостаточно освещения, т.е. "ночь";

При запуске модулю передается имя конфигурационного файла формата JSON:

```
// This is a configuration file example for AmbientSensor module
{
        "AmbientSensor": {
                "ambient_threshold":2780,
                "ambient_hysteresis":30,
                "event_interval":0,
                "hwmon_device":"/sys/class/hwmon/hwmon1/device/in0_input"
        }
}
```

Где назначение параметров следующее:

- **event_interval** - интервал в секундах опроса датчика;

- **ambient_threshold** - порог (в миливольтах) перехода состояния день/ночь;
- **ambient_hysteresis** - гистерезис (так же в миливольтах) перехода состояния;
- **hwmon_device** - путь к специализированному файлу.

```
root@devuan:/home/fabmicro/SIPHomePhone# ./ambient_sensor ambient_sensor.json
./ambient_sensor - an Mrunner module to monitor ambient light sensor. Copyright
(C) 2020, Fabmicro, LLC., Tyumen, Russia.

Openning HWMON input device /sys/class/hwmon/hwmon1/device/in0_input ...

Threshold = 2700, Hysteresis = 50

AMBIENT_SENSOR core TEXT:ReadyForJob
AMBIENT_SENSOR user TAG:AMBIENT_SENSOR_RESP TEXT: {ambient_state:day,
voltage:2959, timestamp:1590072749.661467}
AMBIENT_SENSOR user TAG:AMBIENT_SENSOR_RESP TEXT: {ambient_state:day,
voltage:2976, timestamp:1590072752.702396}
AMBIENT_SENSOR user TAG:AMBIENT_SENSOR_RESP TEXT: {ambient_state:day,
voltage:2976, timestamp:1590072755.733362}
AMBIENT_SENSOR user TAG:AMBIENT_SENSOR_RESP TEXT: {ambient_state:day,
voltage:2959, timestamp:1590072758.764316}
AMBIENT_SENSOR user TAG:AMBIENT_SENSOR_RESP TEXT: {ambient_state:day,
voltage:2962, timestamp:1590072761.795280}
AMBIENT_SENSOR user TAG:AMBIENT_SENSOR_RESP TEXT: {ambient_state:day,
voltage:2962, timestamp:1590072764.826239}
```

3.12. Модуль управления подсветкой экрана (BACKLIGTH_MANAGER)

Данный модуль предназначен для плавного управления подсветкой LCD дисплея Вызывной Панели.

Подсветка дисплея управляется значением скважности ШИМ сигнала на линии **PWM0** микроконтроллера. Для изменения этого значения можно использовать прямую запись в служебный файл: **/sys/devices/virtual/pwm-sunxi/pwm0/duty_percent**. Данный модуль позволяет упростить работу графического интерфейса в следующих моментах:

- В зависимости от настроек в конфигурационном файле, записываемое значение скважности может быть как в прямой полярности, так и в инверсной, т.е. в единица (100% - X) - как раз для данного случая;
- Модуль отслеживает допустимые значения минимальной и максимальной скважности и сам растягивает (скейлит) шкалу, т.е. входной параметр **brighness** задает значение яркости в процентах, далее все вычисляется модулем;

- Модуль имеет возможность осуществления плавного гашения или зажигания экрана, путем задания параметра **fade_time,** который указывает на то, за какое примерно время (в секундах) должен осуществиться переход от текущей яркости к заданной. Если **fade_time** равен нулю, то переключение происходит мгновенно.

Минимальная яркость: **{brightness:0}**

Максимальная яркость: **{brightness:100}**

Плавное гашение экрана за 3 сек: **{brightness:0, fade_time:3}**

Пример конфига:

```
// This is a configuration file example for BacklightManager module

{
        "BacklightManager": {
                "pwm_file":"/sys/devices/virtual/pwm-sunxi/pwm0/duty_percent",
                "pwm_max":100,
                "pwm_min":1,
                "pwm_is_inverted":1
        }
}
```

Назначение параметров следующее:

- **pwm_min** - минимальное значение записываемое в файл отражающее **0%** яркости;
- **pwm_max** - максимальное значение записываемое в файл отражающее **100%** яркости;
- **pwm_is_inverted** - устанавливать в **1**, если действующее значение скважности имеет противоположное значение. Т.е. для того, что бы задать яркость свечения дисплея 80% необходимо записать значение 20;
- **pwm_file** - путь к управляющему файлу.

Пример работы с модулем:

```
BACKLIGHT_MANAGER core TEXT:ReadyForJob
foo user TEXT: {brightness:100, fade_time:1}
BACKLIGHT_MANAGER user TAG:BACKLIGHT_MANAGER_RESP TEXT: {response:ok,
brightness:100, timestamp:1591346146.79573}
foo user TEXT: {brightness:100, fade_time:1}
BACKLIGHT_MANAGER user TAG:BACKLIGHT_MANAGER_RESP TEXT: {response:ok,
brightness:100, timestamp:1591346166.170023}
foo user TEXT: {brightness:0, fade_time:1}
BACKLIGHT_MANAGER user TAG:BACKLIGHT_MANAGER_RESP TEXT: {response:ok,
brightness:0, timestamp:1591346183.925435}
foo user TEXT: {brightness:100, fade_time:1}
BACKLIGHT_MANAGER user TAG:BACKLIGHT_MANAGER_RESP TEXT: {response:ok,
brightness:100, timestamp:1591346202.772100}
```

```
foo user TEXT: {brightness:0, fade_time:5}
BACKLIGHT_MANAGER user TAG:BACKLIGHT_MANAGER_RESP TEXT: {response:ok,
brightness:0, timestamp:1591346219.317383}
foo user TEXT: {brightness:100, fade_time:4}
BACKLIGHT_MANAGER user TAG:BACKLIGHT_MANAGER_RESP TEXT: {response:ok,
brightness:100, timestamp:1591346245.92602}
```

3.13. Модуль управления ЭМ замками (LOCK_MANAGER)

Модуль **LOCK_MANAGER** предназначен для обслуживания процесса открывания/закрывания электромагнитных и электромеханических замков, а так же для отслеживания их состояния.

Основная идея состоит в следующем:

- Имеется некоторое количество **ЭМ** замков, каждый управляется твердотельным или обычным реле через линию GPIO микроконтроллера.
- Что бы открыть замок, необходимо изменить состояние порта GPIO на "1", удержать в течении небольшого промежутка времени (5-7 секунд) и вернуть обратно в "0".
- Существуют замки, которым нужно подавать для открытия "0", а для закрытия - "1", т.е. управляют инверсной полярностью.
- Замки могут быть в разных "состояниях по умолчанию", т.е. замок может быть "всегда закрыт" или наоборот - "всегда открыт".
- Открывание закрытого или закрывание открытого замка осуществляется кратковременно, после чего замок должен вернуться в исходное состояние (состояние по умолчанию).

Модуль LOCK_MANAGER скрывает от пользователя все тонкости отслеживания состояния и "полярности" замков. В рамках модуля, замки имеют уникальные имена задаваемые в конфиге, например **front_door**, **back_door**. У замка есть его **default_state** - состояние по умолчанию (**locked**, **unlocked**). Замку задается **timeout** - время в секундах до автоматического возвращения замка в дефолтное состояние. У замка так же есть параметр **polarity** - задает каким логическим уровнем отпирается замок: **normal** - отпирание происходит записью "1" и **inversed** - записью нуля.

Ниже приведен пример конфигурационного файла **lock_manager.json,** который считывается по умолчанию. Если в качестве параметра в командной строки указать строку, то она будет использоваться как имя конфигурационного файла.

```
// This is a configuration file example for LockManager module
```

```
{
        "LockManager": {
                "locks": [
                        {"name":"front_door", "default_state":"locked",
"polarity":"normal", "timeout":"5",
"gpio_file":"/sys/class/gpio/gpio108/value"},
                        {"name":"back_door", "default_state":"locked",
"polarity":"inversed", "timeout":7, "gpio_file":"/sys/class/gpio/gpio109/value"}
                ]
        }
}
```

Команды модуля:

- **{name:front_door, action:unlock, timeout=15}** - отпирает замок с именем **front_door** на 15 секунд. Если параметр **timeout** не указан, то отпирание происходит на время заданное в конфиге.
- **{name:front_door, action:lock}** - запирает замок с именем **front_door**. Параметр **timeout** в данном случае значения не имеет.

Если замок по умолчанию находится в состоянии **unlocked**, то смысл параметра **timeout** инвертируется. Т.е. замок **запирается** на указанное в **timeout** время, по истечению которого автоматически **отпирается**.

Пример работы с модулем LOCK_MANAGER:

```
root@a13-OLinuXino:/home/olimex# ./lock_manager
./lock_manager - an Mrunner module to control door locks. Copyright (C) 2020,
Fabmicro, LLC., Tyumen, Russia.

Loading and parsing config file: lock_manager.json
LOCK[0]: name = front_door, gpio_file = ./gpio1.txt, polarity = 0, default_state
= 0, timeout = 5
LOCK[1]: name = back_door, gpio_file = ./gpio2.txt, polarity = 0, default_state
= 0, timeout = 7
Openning GPIO control files...
Done

LOCK_MANAGER core TEXT:ReadyForJob
FOO user TEXT: {name:front_door, action:unlock}
LOCK_MANAGER user TAG:LOCK_MANAGER_RESP TEXT: {response:ok, name:front_door,
action:unlock, lockid:0, flag:1, timestamp:1591909654.116378}
LOCK_MANAGER user TAG:LOCK_MANAGER_RESP TEXT: {response:event, name:front_door,
action:lock, lockid:0, flag:0, timestamp:1591909659.132940}
FOO user TEXT: {name:front_door, action:unlock}
LOCK_MANAGER user TAG:LOCK_MANAGER_RESP TEXT: {response:ok, name:front_door,
action:unlock, lockid:0, flag:1, timestamp:1591909664.545926}
LOCK_MANAGER user TAG:LOCK_MANAGER_RESP TEXT: {response:event, name:front_door,
action:lock, lockid:0, flag:0, timestamp:1591909669.559964}
FOO user TEXT: {name:back_door, action:unlock}
LOCK_MANAGER user TAG:LOCK_MANAGER_RESP TEXT: {response:ok, name:back_door,
action:unlock, lockid:1, flag:1, timestamp:1591909736.884624}
LOCK_MANAGER user TAG:LOCK_MANAGER_RESP TEXT: {response:event, name:back_door,
action:lock, lockid:1, flag:0, timestamp:1591909743.902141}
```

3.14. Модуль датчика присутствия (PROXIMITY_SENSOR)

Аппаратное обеспечение Вызывной Панели содержит лазерный дальномер (микросхема ST VL53L0X), который позволяет определить приближение объекта к панели на расстояние ближе, чем 2м. Сам измерительный прибор имеет сложное устройство, описание его инициализации выходит за рамки данного документа. С точки зрения хранения прикладного уровня это простой датчик на шине I2C, который требуется периодически опрашивать (poll-ить).

Для того, чтобы считывать измерения состояния задачи, разработан специальный драйвер vl53l0x.ko, который производит все необходимые действия и экспортирует HWMON интерфейс в Linux систему. В простейшем случае считать текущее значение расстояния до препятствия можно из специализированного файла: /sys/class/hwmon/hwmon2/device/range, например:

```
root@devuan:/home/fabmicro/SIPHomePhone# cat
/sys/class/hwmon/hwmon2/device/range
1976
```

У датчика есть ряд особенностей:

- Датчик возвращает число равное расстоянию в **миллиметрах** до объекта (препятствия).
- Если в поле зрения датчика, т.е. в пределах расстояния 2ух метров, нет объектов, от которых может отразиться лазерный луч, то датчик возвращает нам число более **8100**.
- Если к датчику поднести светоотражающий предмет на расстояние менее 25мм, то он будет выдавать значение менее **20**.
- Любое стекло (в том числе покровное) или прозрачная пленка все равно отражает часть лазерного излучения, по этом появляется эффект **cross talk**, выражается он в том, что датчик начинает выдавать совсем неверные данные. Для того, чтобы этот эффект преодолеть, необходимо провести специальную процедуру калибровки, вычислить компенсационное значение для подавления кросс-толка и в следующий раз при инициализации датчика загрузит ему это значение.

Для унификации и упрощения работы с драйвером, был разработан модуль **PROXIMITY_SENSOR**. Модуль будет непрерывно считывать значения из HWMON, к которому подключен датчик освещенности и отслеживать его состояние. На данный момент имеется пять состояний:

- **unknown** - состояние датчика не известно (происходит при запуске модуля);
- **ranging** - где-то в поле зрения датчика попал объект;
- **approching** - к нам кто-то приближается;
- **dialing** - кто-то стоит прямо возле панели;
- **fingering** - кто-то прикрыл датчик пальцем или рукой;

При запуске модулю указывается имя конфигурационного файла, содержащего следующие параметры:

```
// This is a configuration file example for ProximitySensor module
{
        "ProximitySensor": {
                "proximity_threshold_approaching":1500,
                "proximity_threshold_dialing":700,
                "proximity_threshold_fingering":30,
                "event_interval":100000,
                "hwmon_device":"/sys/class/hwmon/hwmon2/device/range"
        }
}
```

Назначение параметров:

- **hwmon_device** - файл-устройство HWMON, с которого считываются показания датчика, по умолчанию: **/sys/class/hwmon/hwmon2/device/range;**
- **event_interval** - интеревал в микросекундах (1/1000000 секунды) опроса датчика;
- **proximity_threshold_approaching** - порог перехода в состояние "approaching" (кто-то приближается);
- **proximity_threshold_dialing** - порог перехода в состояние "dialing" (кто-то стоит у вызывной панели);
- **proximity_threshold_fingering** - порог перехода в состояние "fingering" (кто-то поднес к датчику палец);

```
rz@devbox:~/SIPHomePhone/ProximitySensor$ build_x86/proximity_sensor
proximity_sensor.json
build_x86/proximity_sensor - an Mrunner module to decode proximity state using
VL53L0X sensor. Copyright (C) 2020, Fabmicro, LLC., Tyumen, Russia.

Loading and parsing config file: proximity_sensor.json
Openning HWMON device /sys/class/hwmon/hwmon2/device/range ...

Thresholds are: proximity_threshold_approaching = 1500,
proximity_threshold_dialing = 700, proximity_threshold_fingering = 30

PROXIMITY_STATE core TEXT:ReadyForJob
PROXIMITY_STATE user TAG:PROXIMITY_STATE_RESP TEXT: {proximity_state:ranging,
range:2125, timestamp:1591134788.780873}
```

```
PROXIMITY_STATE user TAG:PROXIMITY_STATE_RESP TEXT: {proximity_state:ranging,
range:1927, timestamp:1591134792.106122}
PROXIMITY_STATE user TAG:PROXIMITY_STATE_RESP TEXT: {proximity_state:ranging,
range:1887, timestamp:1591134794.181268}
PROXIMITY_STATE user TAG:PROXIMITY_STATE_RESP TEXT: {proximity_state:approching,
range:989, timestamp:1591134796.666460}
PROXIMITY_STATE user TAG:PROXIMITY_STATE_RESP TEXT: {proximity_state:approching,
range:958, timestamp:1591134796.881454}
```

3.15. Модуль считывателя RFID ключей (RFID_READER)

Данный модуль предназначен для общения с аппаратным считывателем электронных ключей (RFID меток). Теоретически, со считывателем возможно двустороннее взаимодействие, но на данный момент модуль не принимает никаких команд, следовательно работает в пассивном режиме - получает от аппаратного считывателя RFID ключ и выдает его в виде сообщений в SunBus.

Для стандарта **Em-Marin**, работающего на частоте 125МГц, ключ (идентификатор метки) это шестнадцатиричное число длиной 5 байт. Em-Marin на данный момент самый распространенный вид RFID ключей в домофонных сетях России. Но имеют хождение и другие виды ключей, например стандарт **MIFARE** и надстройка над ним - **NFC**.

ВАЖНО! Em-Marin, в том виде, в котором он приобрел широкое хождение, не защищен от копирования и подслушивания. В этом смысле стандарт **NFC** на основе **MIFARE** (13.56МГц) более прогрессивен, но мы его пока не поддерживаем, так как там совсем другая аппаратная часть.

Ниже пример вывода модуля RFID_READER при обнаружении метки Em-Marin:

```
root@devuan:/home/fabmicro/SIPHomePhone# ./mRunner_cli
Mrunner command line interface. Copyright (C) 2020, Fabmicro, LLC., Tyumen, Russia.

Connected to UNIX socket: /tmp/Mrunner.socket0, socket_fd: 3
SOCKET: new client idx: 1, fd: 9
SPY: RFID_READER user TAG:RFID_READER_RESP TEXT: {rfid_key:566AC70EC2, type:emmarin,
timestamp:1588885403.447350}
RFID_READER user TAG:RFID_READER_RESP TEXT: {rfid_key:566AC70EC2, type:emmarin,
timestamp:1588885403.447350}
SPY: RFID_READER user TAG:RFID_READER_RESP TEXT: {rfid_key:566AC70EC2, type:emmarin,
timestamp:1588885403.726677}
RFID_READER user TAG:RFID_READER_RESP TEXT: {rfid_key:566AC70EC2, type:emmarin,
timestamp:1588885403.726677}
SPY: RFID_READER user TAG:RFID_READER_RESP TEXT: {rfid_key:566AC70EC2, type:emmarin,
timestamp:1588885422.293753}
RFID_READER user TAG:RFID_READER_RESP TEXT: {rfid_key:566AC70EC2, type:emmarin,
timestamp:1588885422.293753}
SPY: RFID_READER user TAG:RFID_READER_RESP TEXT: {rfid_key:566AC70EC2, type:emmarin,
timestamp:1588885422.434402}
```

RFID_READER user TAG:RFID_READER_RESP TEXT: {rfid_key:566AC70EC2, type:emmarin, timestamp:1588885422.434402}
SPY: RFID_READER user TAG:RFID_READER_RESP TEXT: {rfid_key:566AC70EC2, type:emmarin, timestamp:1588885422.713059}
RFID_READER user TAG:RFID_READER_RESP TEXT: {rfid_key:566AC70EC2, type:emmarin, timestamp:1588885422.713059}

Параметры запуска модуля

Для функционирования модулю требуется сообщить на каком последовательном порту (/dev/tty) находится считыватель и с какой скоростью он выдает данные в этот порт. Все параметры задаются в конфигурационном файле формата JSON, имя которого передается в качестве первого параметра в командной строке.

ВНИМАНИЕ! Данный модуль предназначен для функционирования на изделии SIPHomePhone версии 1.x. Запуск на изделии версии 2.x невозможен по причине отсутствия соответствующего аппаратного обеспечения в его составе. На изделии версии 2.x необходимо использовать аналогичный модель MFRC522Reader предназначенный для работы с колючами и картами NFC/MIFARE.

```
// This is a configuration file example for RFIDReader module

{
    "RFIDReader": {
        "serial": "/dev/ttyS1",
        "baudrate": 115200
    }
}
```

Назначение параметров говорит само за себя.

На плате SIPHomePhone v1.3 аппаратный считыватель привязан к порту **/dev/ttyS1** и общается со скоростью **115200** бод. Собственно, эти данные являются дефолтными настройками для модуля **RFID_READER**.

3.16. Модуль работы с протоколом SIP/2.0 (SIP2)

Модуль sip2 реализует стандартное взаимодействие по протоколу SIP/2.0 с рядом расширений и некоторых отступлений от стандарта.

3.16.1. Опции командной строки

- **--MsgPrefix=** префикс сообщений для обмена по SubBus;
- **--config=** Имя Файла Конфигурации;
- **--CommonSection=** ИмяСекции, в которой необходимо искать шаблоны, должны присутствовать в подсекции "SipPatterns" конфигурационного файла;
- **--AStag=** Имя Тэга с которым посылать команды аудиостримеру;
- **--VStag=** Имя Тэга с которым посылать команды видеостримеру;
- **--SelfTag=** Имя Тэга с которым модуль будет писать свои сообщения;
- **--WithoutVideo** - Не пытаться делать видеозвонок (не создавать видеостример, не предлагать в SDP);
- **--BindPort=НомерПорта** - На какой порт биндить UDP сокет для SIP;
- **--BindAlterPort=НомерПорта** - Номер порта для автодетекта времени пингования NAT (IPv4 only).

3.16.2. Команды и запросы в модуль sip2

Инициировать звонок (INVITE)

prefix: user TAG:SipCallerCmd TEXT: { "cmd":"call", "ReqId":Идентификатор-запроса, "UriFrom":от-какого-имени, "UriTo":куда "proxy":"127.0.0.1", "ProxyPort":"5060", "secret":пароль, "AuthUsername":Логин-авторизации, "pattern":имя-шаблона }

- Если есть **pattern** - недостающие опции берутся из конфига;
- Если **proxy** нет, то он вычисляется из доменной части **UriTo**;
- Если есть **proxy**, но нет **ProxyPort** - то подразумевается 5060;
- Если нет **AuthUsername** - берется левая часть **UriFrom**;

Выполнить регистрацию на SIP прокси (REGISTER)

```
prefix: user TAG:SipCallerCmd TEXT: { "cmd":"RegisterStart",
"ReqId":Идентификатор-запроса, "UriFrom":от-какого-имени,
"RegisterExpire":время_expire, "pattern":имя-шаблона }
```

- Параметры те же что и для INVITE, но:
    - **UriTo** не имеет смысла (регистрируется учетка **UriFrom**)

        - добавляется **RegisterExpire** (по умолчанию - 300 секунд)
- учетка пребывает в двух состояниях - зарегистрирована или нет.
    - При смене этого состояния, т.е.:
        - при первой удачной регистрации;
        - при первом истечении Expire без успешной перерегистрации.
    - Производится отправка информационного сообщения.
- Когда учетка зарегистрирована, то каждые 10 секунд посылается пустой UDP пакет для поддержания пары в **NAT**

Выключить регистрацию (REGISTER END)

```
prefix: user TAG:SipCallerCmd TEXT: { "cmd":"RegisterEnd",
"ReqId":Идентификатор-запроса, "UriFrom":от-какого-имени, "pattern":имя-
шаблона }
```

- Прекращает регистрировать учетку с логином **UriFrom** в течении 10 секунд пытается разрегистрироваться, т.е. посылает **REGISTER** с Expire: 0

Принудительно завершить звонок

```
prefix: user TAG:SipCallerCmd TEXT: { "cmd":"cancel", "ReqId":Идентификатор-
запроса , "CallId":Идентификатор-звонка }
```

Принять входящий звонок

По приходу пакета **INVITE**

```
prefix: user TAG:SipCallerCmd TEXT: { "response":"incoming",
"CallId":Идентификатор-звонка, "From":"vasily@pupkin.org",
"To":"ivan@ivanovich.org", "ip":"8.8.8.8", "port":"5060" "state":"incoming" }
```

- "From" & "To" & "CallId"- поля "From:" & "To:" & "Call-ID:" из пришедшего SIP-пакета;
- "ip", "port" - откуда пришел пакет;

Если нужно игнорировать звонок (т.е. больше про него ничего не сообщать):

prefix: user TAG:SipCallerCmd TEXT: { "cmd":"ignore", "ReqId":Идентификатор-запроса , "CallId":Идентификатор-звонка }

По этой команде полностью игнорируются все SIP пакеты с этим Call-ID, ничего не посылается.

Если нужно отклонить звонок:

prefix: user TAG:SipCallerCmd TEXT: { "cmd":"decline", "ReqId":Идентификатор-запроса , "CallId":Идентификатор-звонка , "code":"601" , "reason":"Declined" }

Посылает ежесекундно ответный пакет, пока не придет **ACK**, или пока не истечет 5 секунд.

Если нужно перейти в предответное состояние (180 Ringing):

prefix: user TAG:SipCallerCmd TEXT: { "cmd":"ringing", "ReqId":Идентификатор-запроса , "CallId":Идентификатор-звонка }

Если этот звонок нужно принять:

prefix: user TAG:SipCallerCmd TEXT: { "cmd":"accept", "ReqId":Идентификатор-запроса , "CallId":Идентификатор-звонка }

Ответ будет такое же как и на команду **call**.

Если нужно провести аутентификацию - то нужно добавить параметр "pattern", и тогда будет проведена процедура WWW-Authentication

Если звонок нет желания принимать - используем вышеописанную команду **cancel**.

Ответ на команды

- Синхронные ответы - краткие, т.к. нужны только как подтверждение того, что команда получена и понята.
- Все подробности о ходе выполнения - идут уже асинхронными сообщениями.
- Признаком того, что сообщение является синхронным ответом на команду является наличие поля **ReqId.**

```
prefix: user TAG:SipCallerCmd TEXT: { "response":"OK", "ReqId":Идентификатор-
запроса , "CallId":Идентификатор-звонка "remark":"OK"}
```

Пример сообщения об отказе от выполнения (закрытие соединения с несуществующим CallId).

prefix: user TAG:SipCallerCmd TEXT: { "response":"fail", "ReqId":Идентификатор-запроса , "CallId":"", "remark":"invalid CallId"}

Асинхронные сообщения

```
prefix: user TAG:SipCallerCmd TEXT: { "response":"SipEvent",
"CallId":Идентификатор-звонка, "code":"100", "remark":"SIP/2.0 100 trying",
"state":"invite" }
```

Содержит следующие поля:

- **responce**
  - **SipEvent** - пришел SIP пакет, поле **remark** содержит его первую строку;
  - **error** - сообщение об ошибке (в **remark**), звонок будет завершен;
  - **timeout** - истек таймаут, звонок будет завершен;
  - **StateChanged** - извещение о смене состояния соединения, в **remark** - timeout для нового состояния;
- **code** - последний полученный код ответа либо команда. К примеру, при исходящем звонке может быть получена команда "BYE". Если еще ни одного пакета не было получено - содержит "0".
- **CallId** - Call-ID sip-диалога.
- **state** - текущее состояние соединения:
  - **init** - идет внутренняя подготовка (запрос DNS, стримеров);
  - **invite-init** - начали посылать пакеты INVITE , но еще не ответили;
  - **invite-ok** - на INVITE есть ответы 1xx;
  - **register** - идет попытка регистрации;
  - **200** - разговор начат или регистрация успешна;
  - **bye** - разговор завершается (BYE & CANCEL посланы но еще не подтверждены);
  - **done** - получено подтверждение окончания разговора;
  - **destroy** - разговор закончен целиком и полностью;
- **remark** - дополнительная информация, к примеру при **StateChanged** - это таймаут нового состояния.
- **audio_stream** - audio netstream_id (пусто если нет).
- **video_stream** - video netstream_id (пусто если нет).

Работа с network стримерами

- Запрос на создание стримеров делается до посылки INVITE — по причине того, что нужно адрес и номер порта в SDP указать;
- после того, как стример стал ненужен (т.е. после завершения звонка либо попытки звонка) - команды закрытия стримеров посылаются с задержкой в 1-2 секунды, т.е. они складываются в очередь на отложенные действия.

3.16.3. Параметры запросов модуля sip2

Пример команды исходящего звонка:

msg TAG:SipCmd TEXT: { cmd:call, ReqIs:ID0001, pattern:MyPattern, ... }

- **cmd** - команда:
    - **accept** - принять входящий звонок (или перейти из предответного состояния в двухсторонний диалог);
    - **call** - новый исходящий звонок;
    - **cancel** - прекращение звонка;
    - **ignore** - делать вид, что мы не видим пакетов с этим Call-ID;
    - **progress** - перейти в предответное состояние;
- **CallId** - Идентификатор SIP диалога, к которому относится команда.
    - Обязательно для команд **accept**, **cancel**, **ignore**, **progress**
- **MaxTalkLen** - Задает максимальное время разговора, если используется - должен быть указан в первой команде **accept** и **progress**.
- **MaxResponceLen** - Задает максимальное время (в секундах) хотя бы какого-то ответа на наш **INVITE**. по умолчанию - 5 секунд.
- **ReqId** - Идентификатор команды, который надо повторить в ответе.
- **pattern** - Имя описания шаблона в конфиге.
    - Шаблон может содержать любые из нижеописанных опций
    - Опции из команды - приоритетней, чем из шаблона
- **display** - Человекочитаемое имя звонящего, т.е. передается в поле **From** в кавычках перед SIP URI.
- **UriFrom** - Адрес кому звоним, например: wasya@pupkin.com
- **UriTo** - От имени какого адреса звоним, см. **UriFrom**.
- **proxy** - Адрес SIP-прокси куда слать SIP пакеты. Если этой опции нет, то используется домен из **UriFrom**. В этом случае модуль пытается найти записи типа **SRV** для доменов типа **_sip._udp.pupkin.com**.
- **ProxyPort** - Номер порта для **proxy** сервера. Если не задан, то подразумевается **5060**.
- **AuthUsername** - Идентификатор пользователя для авторизации. Если не задан, то берется левая часть **UriFrom**.
- **AuthSecret** (он же **secret**) - Пароль для авторизации - только читаемые ASCII символы.

- **AuthRealm** - Значение поля Realm для авторизации входящих звонков. Т.е. указывается в поле **www-authenticate** при ответе с требованием авторизации. Если этой опции не указано - то звонок принимается без авторизации.
- **code** - Опция команды **cancel**, задает с каким кодом отбивать.
- **SendCode** - Опция команды **progress** задает, какой код ответа посылать, например:
    - **100** - Trying
    - **183** - Ringing

Наличие двух опций (**code** & **SendCode**) для одного и того же обусловлено историческими причинами.

- **reason** - Опция команды **cancel**, задает текстовую расшифровку ответа.
- **stun_host** & **stun_port** - Параметра STUN сервера, передаются при создании сетевых стримов.
- **audio_only** -
    - Если **true** - не использовать видео при звонке
    - Если **false** - использовать видео
    - иначе (не указано или левота):
        - при наличии опции командной строки **- - WithoutVideo** - без видео
        - иначе - с видео
- **OnlySending** - Если **true**, то только посылка аудио/видео, принятые пакеты будут сброшены. Используется для видеонаблюдения.
- **RegisterExpire** - Продолжительность действия предлагаемой регистрации в секундах. Наше дело предложить, но что сервер ответит - то и будет использовано.

3.16.4. Опции сообщений модуля sip2

Модуль может посылать ответы на команды, а так же асинхронные сообщения. Ответ всегда включает в себя опцию **ReqId**, взятую из команды.

- **CreatorReqId**
    - То же самое, что и **ReqId**, но не в ответах, а в асинхронных сообщениях.

    - Используется в сообщениях о регистрациях, позволяя узнать - какой же командой она (регистрация) была запущена.
- **CallId**
    - Поле **Call-ID** sip-диалога, к которому относится сообщение.
    - Его может не быть, если это ответ на неудачную команду.
- **responce* (см. remark)

тип сообщения:

- **OK** - команда принята и ошибок не обнаружено;
- **fail** в команде найдены ошибки, подробности в **remark**;
- **SipEvent** - принят SIP пакет диалога, инициированного нами;
- **StartDialog** - принят SIP пакет диалога, инициированного снаружи;
- **IncomingInvite** - принят INVITE нового диалога (и проведены первичные проверки);
- **StreamCreated** - нужные стримеры созданы;
- **StateChanged** - изменилось состояние SIP диалога (см. поле **state**).
- **remark** - Неформальное дополнительное описание, можно использовать для быстрого поиска - в каком месте кода сгенерировано сообщение. Для сообщений типа **SipEvent** - содержит целиком первую строку пришедшего пакета.
- **IsVideoSpy** - Параметр типа **bolean**, указывается как **true**, если входящий звонок - от RtspProxy, на котором домофон зарегистрировался.
- **state** - Состояние SIP диалога, одно из описанных ниже:

для исходящих звонков (см. state, remark):

- **init** - еще ничего не шлем, потому что надо разрезолвить куда слать и получить ответы от стримеров;
- **invite-init** - начали посылать пакеты INVITE;
- **invite-ok** - нам уже пришел хоть какой-то ответ на INVITE;
- **200** - разговор начат;
- **bye** завершаем разговор (ждем, что удаленная сторона подтвердит приём сообщения);
- **done** разговор совсем завершен, как правило сразу ни ним - **destroy**;
- **destroy** - SIP диалог уничтожается (т.е. это точно послденее сообщение с этим **CallId**;

для входящих звонков (см. state, remark):

- **incoming-init** - входящий звонок, еще ничего не проверено, так что реакции не требуется;
- **invite-responce-waiting** - входящий звонок, ждем указания что с ним делать;
- **hide** - не показывать что мы видели INVITE, игнорировать все пакеты (устанавливается командой "ignore");
- **ringing** - предответное состояние (т.е. слать 183 Ringing);
- **answer** - посылать "200 OK" до тех пор, пока не придет ACK;
- **AuthWait** - посылаем 401, ждем когда авторизуются;
- **StreamerWait** - ждем создания стримеров.

После прихода **ACK** в ответ на **200** - переходим в состояние **200**, как при исходящем звонке (потому что дальнейшая логика полностью совпадает).

Для того, чтобы отбить входящий звонок, так же как и при исходящем, посылаем команду **cancel**.

для исходящей регистрации (см. state, remark):

- **unregistered** - резолвим куда слать пакеты;
- **registering started** - начинаем посылку **REGISTER**;
- **registering in progress** - нам уже что-то ответили;
- **registered** - успешно зарегистрировались;
- **re-registering** - время expire наполовину истекло, начинаем повторную регистрацию;
- **unregistering** - отменяем регистрацию (шлем REGISTER с Expire=0);
- **unregistered** - успешно отменили регистрацию;
- **Pause for retry** - выдерживаем паузу до следующей попытки зарегистрироваться.

При **re-registering** имеется два варианта:

- если до истечения времени Expire успешно перерегистрировались - то в **registered**;
- если же нет - то в **unregistered** (т.е. резолвинг сервера осуществляется повторно);
- **code** - Содержит код SIP ответа:
    - "0" если еще ни одного SIP пакета не было получено;

    - три цифры (например "183") - если был ответ на команду;
    - название команды (например "ACK") если это была команда;
- **audio_stream** - Идентификатор аудиостримера, выдается только после того, как удаленная сторона сообщила на какой IP:порт она принимает поток.
- **video_stream** - Идентификатор видеостримера, выдается только после того как удаленная сторона оообщила на какой IP:порт она принимает поток, но и если мы и не собирались посылать видео, то отсутствует.
- **SourceIp** - С какого IP получен SIP пакет.
- **SourcePort** - С какого порта получен SIP пакет.
- **UriFrom** - SIP URI инициатора (отправителя) SIP сообщения.
- **UriTo** - SIP URI получателя SIP сообщения.
- **DisplayFrom** - Человекочитаемое имя отправителя.
- **DisplayTo** - Человекочитаемое имя получателя.

3.16.5. Исходящий звонок

Инициируем вызов:

```
msg: user TAG:SipCallerCmd TEXT: { cmd:call , ReqId:Mid1 ,
UriFrom:"Fabfloor@ezau.ru", secret:"a46cdbsnv3", UriTo:0@ezau.ru,
display:"TheName", pattern:TlmHome , audio_only:false }
```

Отчет о том, что команда принята:

```
msg: user TAG:SipCallerAns TEXT:
{ "CallId":"lknabnpf6961861291123sneadbc5efff92c02.b", "ReqId":"Mid1",
"remark":"OK", "response":"OK" }
```

Сообщается Call-ID звонка!

После того, как поработали с DNS и запустили стримеры:

```
msg: user TAG:SipCallerAns TEXT: { "response":"StateChanged", "code":"0",
"CallId":"lknabnpf6961861291123sneadbc5efff92c02.b", "state":"invite-init",
"remark":"Start sending INVITE" }
```

code равен нулю, потому что мы пока что еще ни одного пакета не получали (да и не посылали еще).

Пришел ответ от сервера (с намеком, что надо авторизоваться):

```
msg: user TAG:SipCallerAns TEXT: { "response":"SipEvent", "code":"401",
"CallId":"lknabnpf6961861291123sneadbc5efff92c02.b", "state":"invite-init",
"remark":"SIP/2.0 401 Unauthorized" }
```

Пришел **Ringing или Progress**:

```
msg: user TAG:SipCallerAns TEXT: { "response":"SipEvent", "code":"183",
"CallId":"lknabnpf6961861291123sneadbc5efff92c02.b", "state":"invite-ok",
"remark":"SIP/2.0 183 Session Progress" }
```

Звонок приняли:

```
msg: user TAG:SipCallerAns TEXT: { "response":"SipEvent", "code":"200",
"CallId":"lknabnpf6961861291123sneadbc5efff92c02.b", "state":"invite-ok",
"audio_stream":"0x1e36a18", "remark":"SIP/2.0 200 OK" }


msg: user TAG:SipCallerAns TEXT: { "response":"StateChanged", "code":"200",
"CallId":"lknabnpf6961861291123sneadbc5efff92c02.b", "state":"200",
"audio_stream":"0x1e36a18", "remark":"Talk started" }
```

С той стороны завершили разговор:

```
msg: user TAG:SipCallerAns TEXT: { "response":"SipEvent", "code":"BYE",
"CallId":"lknabnpf6961861291123sneadbc5efff92c02.b", "state":"200",
"audio_stream":"0x1e36a18", "remark":"BYE sip:Fabfloor@217.116.57.130:5060
SIP/2.0" }


msg: user TAG:SipCallerAns TEXT: { "response":"StateChanged", "code":"BYE",
"CallId":"lknabnpf6961861291123sneadbc5efff92c02.b", "state":"done",
"audio_stream":"0x1e36a18", "remark":"Talk finish" }


msg: user TAG:SipCallerAns TEXT: { "response":"StateChanged", "code":"BYE",
"CallId":"lknabnpf6961861291123sneadbc5efff92c02.b", "state":"destroy",
"audio_stream":"0x1e36a18", "remark":"normal done" }
```

- Первое сообщение - собственно о том, что пришел **BYE**;
- Второе - что мы перешли в фазу завершения разговора;
- Третье - об уничтожении sip диалога. Ну просто так "исторически сложилось".

3.16.6. Входящий звонок

Предварительно регистрируемся, чтобы получать входящие SIP сообщения:

```
msg: user TAG:SipCallerCmd TEXT: { cmd:RegisterStart , ReqId:Mid2 ,
UriFrom:"201@sip.fabmicro.ru", secret:"pass201", UriTo:127@sip.fabmicro.ru ,
audio_only:false }
```

...

Пришел INVITE, тем самым создав новый диалог:

```
msg: user TAG:SipCallerAns TEXT: { "response":"StartDialog", "code":"INVITE",
"CallId":"CCE3F65A-3D5511EB-9A63A550-3A28BFFD@77.242.111.106",
"state":"incoming-init", "remark":"INVITE sip:201@192.168.169.6:5060 SIP/2.0" }
```

Диалог перешел в состояние ожидания нашей команды - что с ним делать:

```
msg: user TAG:SipCallerAns TEXT: { "response":"StateChanged", "code":"INVITE",
"CallId":"CCE3F65A-3D5511EB-9A63A550-3A28BFFD@77.242.111.106", "state":"invite-
responce-waiting", "remark":"incoming invite" }
```

Подробности про входящий звонок (т.е. откуда он, кому и от кого):

```
msg: user TAG:SipCallerAns TEXT: { "response":"IncomingInvite", "code":"INVITE",
"CallId":"3aed43b1-878f-4f6b-8973-f512815d10c6", "state":"invite-responce-
waiting", "remark":"new incoming INVITE", "SourceIp":"178.72.91.128",
"SourcePort":"5010", "DisplayFrom":"0", "DisplayTo":"",
"UriFrom":"Mfloor@172.24.47.73", "UriTo":"Fabfloor@217.116.57.130" }
```

Команда принять звонок (с авторизацией):

```
msg: user TAG:SipCallerCmd TEXT: { cmd:accept, ReqId:Mid2, AuthUsername:TheInc ,
AuthSecret:TheSecret , AuthRealm:TheRealm ,CallId:eafa7c28-307e-4481-87ee-
e6eca9247240 }
```

Переходим к приему, в этот момент будут наконец-то запрошены network streamer'ы:

```
msg: user TAG:SipCallerAns TEXT: { "response":"StateChanged", "code":"INVITE",
"CallId":"CCE3F65A-3D5511EB-9A63A550-3A28BFFD@77.242.111.106", "state":"answer",
"remark":"starting talk" }
```

Все нужные стримеры готовы:

```
msg: user TAG:SipCallerAns TEXT: { "response":"StreamCreated", "code":"INVITE",
"CallId":"CCE3F65A-3D5511EB-9A63A550-3A28BFFD@77.242.111.106", "state":"answer",
"audio_stream":"0x1e36a18", "remark":"all streamers ready" }
```

К нам пришел **ACK** на наши ответы о начале разговора:

```
msg: user TAG:SipCallerAns TEXT: { "response":"StartDialog", "code":"ACK",
"CallId":"CCE3F65A-3D5511EB-9A63A550-3A28BFFD@77.242.111.106", "state":"answer",
"audio_stream":"0x1e36a18", "remark":"ACK
sip:201@192.168.169.6:5060;transport=udp SIP/2.0" }
```

И мы перешли в состояние начавшегося разговора. Т.е. мы не просто начали разговор - мы знаем, что вторая сторона знает об этом и тоже начала:

```
msg: user TAG:SipCallerAns TEXT: { "response":"StateChanged", "code":"ACK",
"CallId":"CCE3F65A-3D5511EB-9A63A550-3A28BFFD@77.242.111.106", "state":"talk",
"audio_stream":"0x1e36a18", "remark":"talk (ACK received)" }
```

3.16.7. Документация по классам и функциям

- **static const unsigned TooBig = 0x10000;**
    - число, большее чем двухбайтовое целое;
- **struct ContactCollection**

    - содержит последние 32 использованных **Contact** (с поиском по полю **Call-ID**);
    - назначение - формировать корректные ответы на SIP-фреймы, пришедшие после уничтожения диалога.
- **struct AboutDomainName**
    - имя, порт (или TooBig), приоритет (или TooBig если это не результат _sip._udp.);
- **struct NameList**
    - Список имен, которые надо разреволвить;
    - Сначала содержит одно имя, но потом добавляются кандидаты в результате резолва CNAME & SRV записей.

3.17. Менеджер сообщений SunBus

Обеспечивает обмен небольшими сообщениями между модулями, с возможностью подглядывать за обменом и руками что-нить для дебага подсовывать.

3.17.1. Основые возможности

- SunBus может запускать программы (называемые **модулями**) с указанными в конфиге аргументами.
- Если один из модуль завершился, то SunBus культурно просит завершиться и все остальные модули, после чего перезапускает все модули заново.
- Есть возможность подключиться по TCP и мониторить все события в системе, в том числе:
    - получать копии всех сообщений от всех модулей;
    - посылать самому сообщения наравне с "настоящими" модулями.
- Все, что модуль выводит в свой стандартный вывод (STDOUT), построчно разбирается, и то, что соответствует формату на сообщения обмена (формат описан ниже) - пересылается другим модулям (согласно описанию заданому в конфиге) на их стандартный вход (STDIN).
- Любой модуль может попросить SunBus сохранить некоторые данные у себя, переслав их модулю в случае повторного старта. Данные хранятся по принципу "ключ-значение".
- Поддерживает аппаратный Watchdog через драйвер */dev/watchdog*.

3.17.2. Модули в системе SunBus

Модуль в системе SunBus - это программа, точнее - исполняемый файл, имя которого указано в разделе конфигурационного файла для описании модуля. Работа с модулем осуществляется следующим образом:

- Все, что выводит программа на свой стандартный вывод (STDOUT) трактуется как сообщения другим модулям.
- Она строка - одно сообщение. Формат строки (сообщения) регламентирован.
- Строки, которые не проходят контроль на формат сообщений считаются сообщениями об ошибках и не транслируются другим модулям, но отображаются в логе.

- Сообщения могут быть адресованы не только другим модулям, но и самому **SunBus**'у (т.е. служебные сообщения).
- Все, что программа (модуль) выводит в стандартный поток ошибок (STDERR), считается сообщениями об ошибках, не транслируется другим модулям, но отображается в логе.
- SunBus может посылать модулю некоторые служебные сообщения, на которые требуется ответ.
- Если SubBus считает, что модулю пора завершить свою работу, то модулю закрывают стандартный вход (STDIN).
- Если модуль не завершился за определенное время - его завершат принудительным способом: kill(SIG_TERM), а через еще некоторое время: kill(SIG_KILL).

3.17.3. Агенты в системе SunBus

В некотором смысле **агент** - это **модуль**, только с особенностями:

- Это не внешний исполняемый файл, а процедура внутри **SunBus**'а.
- Его стандартный вход и выход - являются чем-то другим.
- На стандартный вход агента подается информация обо всем, что происходит в системе: копии сообщений от модулей (и других агентов), их сообщения об ошибках.

Агент - SunBus

Стандартный вход SunBus'а интерпретируется как стандартный выход агента:

- Все, что поступает на вход будет интерпретироваться как сообщения другим модулям.
- Роль стандартного входа агента выполняет стандартный поток вывода SunBus'а.
- Если закрыть стандартный вход, то это будет воспринято как команда завершить работу системы - все модули будут остановлены.

Агент - TCP-соединение

Для удобства отладки **SunBus** может принимать соединения по TCP, каждое такое соединение ведет себя как отдельный агент:

- Стандартный вход - это принятое от удаленной стороны сообщение (строка символов).

- Роль стандартного выхода агента так же выполняет TCP-соединение, т.е. в TCP-соединение посылается все то же самое - копии сообщений, сообщения об ошибках.
- В отличии от SunBus, TCP-соединению требуется пройти простейшую авторизацию - послать стандартное сообщение о готовности работы. Суть в том, что формат сообщения требует, чтобы оно начиналось с префикса, описанного в конфигурации **SunBus**, в опции **SpyLinePrefix**.

3.17.4. Запуск SunBus как демона

При запуске как демона программе закрывается ее стандартный вход, а **SunBus** реагирует на это как на предложение завершиться. Для подавления такого поведения служит опция командной строки:

--RunAsDaemon

Ее побочным эффектом является то, что SunBus перестает читать свой вход - но демону этого и не требуется.

Если требуется, что бы **SunBus** сам ушел в демоны, его следует запускать с опцией:

--daemon

Получив такую опцию в командной строке, SunBus проведет разбор конфига, подавит слежение за stdin, и отправится в фоновое выполнение.

3.17.5. Файл конфигурации

Имя файла конфигурации задается опцией командной строки: **--ConfigList=**

- Если опция не задана, то SunBus смотрит в переменную окружения под названием **PanelCfg**. При её наличии имя конфигурационного файла берется из неё.
- Если не заданы оба указанных выше механизма, то берется фиксированное имя **all.json** из текущего каталога.
- файл конфигурации представляет из себя JSON структуру, парсится библиотекой MiniJsonParser).
- Предполагается, что файл содержит конфигурационные секции не только для SunBus, но и для остальных модулей. Поэтому опцией ** - -CfgSection=** указывать имя секции, в которой лежит конфигурация именно SunBus.

- Если нет этой опции, то берется имя секции **SunBus**.
- Для удобства модулей, **SunBus** устанавливает переменную окружения по имени **JsonConfigFilename**, записывая туда фактическое имя используемого файла конфигурации.

3.17.6. Пример файла конфигурации

```
{
        SunBusCfg:{
                SpyLinePrefix:"msg:",
                ManagmentPort:{ ip:"::", port:"5493" },
                tags:[tag1,tag2, tag3],
                modules:{
                        module1:{
                                disabled:false,
                                exec:"./Test1.pl",
                                args:"msg1",
                                prefix:"msg1:",
                                SendTags:[tag1],
                                SubscribeTags:[tag2]
                        },
                        module2:{
                                disabled:false,
                                exec:"./Test2.pl",
                                args:"msg2 30",
                                prefix:"msg2:",
                                SendTags:[tag2,tag3],
                                SubscribeTags:[tag1]
                        }
                },
                ModuleOrder:[module2, module1] // forced module ordering
        }
}
```

SunBusCfg - название секции, которое надо указать в опции командной строки, т.е. написать типа
SunBus --CfgSection=SunBusCfg

3.17.7. Опции модуля

- **module1** , **module2** - условные имена модулей. Уникальность не контролируется, да и не влияет ни на что.
- **disabled** - если **true**, то модуль не запускается. Если опция не указана - то считается что указали **false** (т.е. по умолчанию - модуль запускать).
- **exec** - имя исполняемого файла, который надо запустить.
- **args** - список аргументов запуска модуля. Может быть задан массивом (и тогда элементы будут переданы как есть), либо строкой. В последнем случае строка разбивается на отдельные опции пробелами, что просто. А

зато если задать массивом - то можно передавать опции, содержащие внутри себя пробелы.

- **prefix** - каждая строка должна начинаться с этого слова (пробелы не допускаются!). Строки без этого префикса - считаются ошибочными. Так же этот префикс стоит спереди у всех строк, которые передаются модулю.
- **SendTags** - массив с перечислением тех тэгов, которые разрешено указывать в своих сообщениях данному модулю.

  *про тэги - ниже, в разделе про формат сообщений
- **SubscribeTags** - сообщения с какими тэгами нужно передавать этому модулю.

3.17.7. Глобальные опции

- **SpyLinePrefix** - аналог опции **prefix**, но для строк не от модулей, а от TCP-агентов и со стандартного входа самого SunBus. Т.е. Вы можете запустить SunBus с командной строки и сами что-нибудь посылать модулям.
- **ManagmentPort** - IP и порт, на котором ждать соединений от TCP-агентов. Для справки - ** : : ** - это аналог 127.0.0.1 , но принимает соединения не только IPv4, но и IPv6.
- **tags** - перечисление всех возможных имен тэгов. Нужно как в педагогических целях, так и для выделения внутренних структур под них.

3.17.8. Формат сообщений

Типичное сообщение:
msg: user TAG:SomeTag TEXT: the line text

Можно считать, что сообщение состоит из из четырех частей:

- префикс (самое первое слово в строке)
  - должен быть таким, как указано в конфиге в описании посылающего сообщение модуля
- флаги
- разделитель флагов и текста сообщения (буквосочетание "** TEXT:**")
- текст сообщения

Префикс служит для отделения осознанно посланных сообщений от чего-то другого - скажем, каких-то служебных посылок из недр библиотек, сообщений об ошибках и т.п.

Флаги - слова, разделенные пробелами:

- Самый первый флаг задает тип сообщения, кои могут быть суть:
  - **core** - сообщение для самого SunBus или наоборот - от него;
  - **user** - сообщение от или для другого модуля;
  - **info** - сообщения со значениями, которые попросили запомнить предыдущие инкарнации модуля;
  - другие типы сообщений интересны только **агентам**.
- Остальные флаги могут идти в любом порядке.
- SunBus знает следующие флаги:
  - **TAG:** - сразу после двоеточия идет тэг. Они используются для маршрутизации сообщений типа **user** между модулями.
  - **hidden** - не показывать это сообщение агентам. Скажем, если их слишком много и это будет мешать отладке.
  - остальные флаги - просто игнорируются

Текст сообщения - все, что идет после двоеточия после **TEXT**

3.17.9. Маршрутизация сообщений между модулями

- если тип сообщения - **user** - то составляется список - какие флаги типа **TAG:** указаны.
- если указать тэг, который для этого модуля НЕ перечислен в конфиге в опции **SendTags** - сообщение считается ошибочным и никуда не пересылается (а только агентам докладывается).
- сообщение пересылается тем модулям, у которых в конфигурации в опции **SubscribeTags** есть хотя бы один из тэгов, указанных во флагах.
- обратно модулю его же сообщение никогда не отправляется. Так, на всякий случай.
- перед отправкой сообщения модулю - его префикс заменяется на **prefix**, указанный в конфигурации модуля-получателя.

3.17.10. Получение сообщений с любым тэгом (специальный тэг "all")

В опции **SubscribeTags** можно указать специальный тэг по имени **all** - он сопоставляется с любым другим тэгом, т.е. модуль будет получать сообщения, кроме тех, у которых вообще ни одного тэга не выставлено.
Как можно догадаться, **all** - заменяется на список всех описанных в конфигурации тэгов.

ПОБОЧНЫЙ ЭФФЕКТ: можно самому послать сообщение с "TAG:all", и его получат все модули, кроме тех, у которых опция **SubscribeTags** совсем пуста.

Сообщения типа core

ModuleName

SunBus посылает модулю сообщение:

msg: core TEXT:ModuleName MyName
где **MyName** - это имя модуля, как оно написано в конфиге **SunBus**
ReadyForJob

- Модуль **обязан** послать такое сообщение как только будет готов к работе.
    - если он не пошлет его в течении 10 секунд - запускается процедура рестарта всех модулей;
    - насчет 10 секунд - см. **DeadlineTimeout**.
- Если модуль уже был ранее запущен и делал сообщения **SetValue** - они будут ему продублированы в ответ на **ReadyForJob**.

DeadlineTimeout

- у модуля есть параметр **таймаут** (измеряется в секундах), который используется на все случаи жизни.
- по умолчанию он равен 10 секундам.
- командой **DeadlineTimeout=DD** его можно менять. **DD** - новое значение (может состоять из одной или двух цифр).

SetValue

- первое слово после "**TEXT:SetValue**" - ключ записи.
- Хранится в записи - вся строка с командой целиком (так, для простоты).
    - После того, как модуль пошлет **ReadyForJob** - ему в ответ все накопленные строки и выдадут, такими, как и посылались.
- записей может быть много, лишь бы ключи были разные.
- команды стирания записи нет, просто старое значение замещается новым.

NeedPing

- после подачи этой команды - SunBus будет каждую секунду посылать модулю строку.

prefix core TEXT:PING

- равномерность - не гарантируется, даже иногда может и единично пропускаться.

WatchPong

- это просьба к SunBus послать через **таймаут** сообщение **ping,** т.е. **ping** будет послан не сразу.

ping

- если модуль хотя бы раз слал **WatchPong** - то ему периодически будут посылать **ping**.
- если модуль в течении **таймаута** (см. **DeadlineTimeout**) не ответит **WatchPong** - запустится процедура рестарта всех модулей.

3.17.11. Порядок запуска модулей

- модули запускаются в порядке их описания в конфиге.
- модули запускаются один за другим, а не параллельно, точнее, следующий модуль запускается только после того, как предыдущий выдаст **ReadyForJob**.
- после того, как запустятся все модули, всем рассылается сообщение

prefix: core TEXT:AllModulesStarted ["module1", "module2"]

В квадратных скобках - перечислены имена реально запущенных модулей.

3.17.12. Порядок завершения модулей

- если один из модулей завершился сам
  - или не выдал в течении **таймаута** команду **ReadyForJob**
  - или SunBus словил сигнал **HUP** или **TERM** или **INT** то SunBus переходит в режим завершения всех модулей
- модули завершаются последовательно, в порядке, обратном описанию в конфиге
  - т.е. пока не завершится один модуль - к завершению другого не приступаем
- завершением считается **EOF** на потоке стандартных ошибок от модуля

`prefix: core TEXT:reload`
посылается если приступаем к завершению всех модулей с последующим перезапуском

`prefix: core TEXT:finalize`

посылается если приступаем к завершению всех модулей с целью окончательного выхода
Для каждого модуля:

- сначала мы посылаем ему команду **shutdown**
- и сразу же закрываем его стандартный вход
    - т.е. в принципе - модуль может просто реагировать на **EOF** на своем входе
- если по истечении **таймаута** модуль не завершился - шлем сигнал **TERM**
    - но если мы знаем, что тот процесс, который мы запускали (fork'ом) уже терминировался - то закрываем стандартный выход модуля, в надежде, что он словит сигнал **PIPE** при попытке что-нить туда написать
- ждем еще **таймаут**
- шлем сигнал **KILL** (если можем)
- если по истечении **таймаута** выход ошибок модуля так и не закрылся - ведем себя так, словно таки он закрылся, но взводим флаг глобальной проблемы - т.е. когда все модули завершаться - SunBus тоже завершится

А если нажать Ctrl-C (или еще как-то послать SunBus сигналы HUP, TERM или KILL) - тоже запустится процедура завершения модулей, причем если это был HUP - то потом заново запустятся, а в остальных случая - SunBus завершится).

ping pong

смотри в "сообщениях типа **core**, команды **ping** & **WatchPong**.

SetValue (хранение данных между рестартами)

смотри в "сообщениях типа **core**" команду **SetValue**.

3.17.13. Агенты TCP (и не только)

- В конфигурации можно указать опцию **ManagmentPort:** и тогда при старте SunBus попробует открыть TCP-сокет на указанном адресе, и туда можно соединиться. Например telnet'ом.
- Каждое такое соединение - с точки зрения SunBus - как будто модуль
    - которому забыли указать **SubscripeTags**;
    - в качестве **prefix** используется глобальная опция **SpyLinePrefix**.
- А стандартный вход и выход самого SunBus'a - тоже образует такой модуль (для удобства дебага).
- Зовутся такие модули - **агентами** (а так же - **TCP-агентами**).

- До тех пор, пока **TCP-агент** не пошлет команду типа "prefix core TEXT:ReadyForJob" - он считается нерабочим и ему ничего не посылается. Т.е. по сути - **SpyLinePreifx** выполняет роль пароля.
- Агент может посылать сообщения типа **user** - и они будут пересланы модулям согласно установленным тэгам.
- Агенту, в отличии от обычных модулей, посылаются служебные сообщения типов:
    - **err:** - копии строк, посланных модулями, но не прошедшими проверку (или посланными в стандартный вывод ошибок);
    - **spy:** - копии сообщений от модулей;
    - **info** - всякое разное, генерируемое в отладочных целях самим SunBus.

3.17.14. Поддержка аппаратного watchdog-а

SunBus может быть запущен с опцией командной строки **--WatchdogTimeput=** которая устанавливает время в секундах до срабатывание аппаратного watchdog-а. Если опция указана и ей знаачение больше нуля, то SunBus проинициализирует аппаратный watchdog и будет регулярно его обновлять (но не чаще чем один раз в секунду). Если по каким ибо причинам обновление таймера watchdog-а не состоится в течении указанного времени, то произойдет аппаратный сброс системы (полная перезагрузка).

Для функционирования этой опции необходима аппаратная поддержка watchdog-а в операционной системе Linux и наличие загруженного драйвера устройства /dev/watchdog. Более подробно об аппаратных watchdog-а в ОС Linux можно прочесть в документации ядра каталоге с исходным кодом ядра: **Documentation/watchdog/**.

3.18. Идентификатор вызывной панели (SID)

По некоторым данным все SoC производства Allwinner содержат небольшой кусочек «перепрошиваемой» памяти называемой e-Fuse (или EEPROM). По умолчанию, производитель SoC зашивает в этот EEPROM различную идентификационную информацию о конкретном чипе, в том числе некий **SecureID*** (или **SID**). Это длинное 128-битное число, которое, по идее, должно быть уникально для всех чипов Allwinner.

Для SoC A13 и A20 область EEPROM располагается в физическом адресном пространстве, начиная с адреса **0x01c23800**, и имеет разную длину. Для A13 это 16 байт, для A20 - 512 байт. Первые 16 байт этой области содержат **SID**. Выглядит это вот так:

0x16 0x25 0x42 0x51 0x50 0x4E 0x54 0x31 0x35 0x30 0x30 0x30 0x09 0x41 0x0A 0xCD

Сам SID тоже имеет структуру: два байта - идентификатор типа SoC (0x1625 это A13), далее следует 10 байт уникальный (если хотите - серийный) номер кристалла, и 4 байта прочего. Если внимательно присмотреться, то можно заметить, что 10-ти байтовый идентификатор это ASCII символы (возможно).

3.18.1. SecureID как задатчик MAC адреса

Так как у нас нет своего блока официально аллоцированных MAC, то приходится использовать т.н. fake-овый MAC, т.е. такой MAC адрес, который генерируется случайным образом при загрузке ядра Linux. В некоторых случаях постоянно изменяющийся MAC-адрес представляет проблему, поэтому не плохо бы его привязать к чему-нибудь уникальному в рамках данной системы. Таким вот уникальным идентификатором и выступает SID.

В ядро системы Linux (3.4.103) был внедрен небольшой патч, который считывает SID из EEPROM, загоняет его в MD5 хэш для большей равномерности и далее использует для вычисления MAC адреса. Ядро с таким патчем обязательно выдает следующую строку при загрузке сетевого драйвера (при создании сетевого интерфейса):

MAC address was generated by SecureID: 0x16 0x25 0x42 0x51 0x50 0x4E 0x54 0x31 0x35 0x30 0x30 0x30 0x09 0x41 0x0A 0xCD

3.18.2. Считывание SecureID из user-space

Чтобы приложения могли считывать SID из EEPROM и использовать его в своих целях, был написан небольшой драйвер (модуль) **sunxi_sid.ko**. Этот драйвер создает специальный файл **/sys/devices/platform/sunxi_sid.0/eeprom** длиной равной размеру EEPROM, содержимое этого файла отображает байтовый массив, содержащийся в EEPROM. Вот пример считывания все того же самого SID из user-space:

```
root@devuan:~# hexdump -C /sys/devices/platform/sunxi_sid.0/eeprom
00000000  16 25 42 51 50 4e 54 31  35 30 30 30 09 41 0a cd  |.%BQPNT15000.A..|
00000010
```

Как видно, длина файла составляет 16 байтов, а первые два байта говорят нам о том, что мы работаем на SoC Allwinner A13.

Драйвер **sunxi_sid.ko** можно взять тут: devbox:/opt/linux-kernels/current/drivers/misc/eeprom/sunxi_sid.ko

Скопировать в каталог **/lib/modules/3.4.103/kernel/drivers/misc/eeprom/sunxi_sid.ko** (если полный путь не существует - его нужно создать).

После чего загрузить драйвер можно командой **modprobe sunxi_sid**, либо поместить его имя в файл **/etc/modules**.

3.19. Исполнение операций по команде от сервера (HTTP API)

(Данный механизм реализуется модулем **SuperGlue**)

HTTP API - это механизм позволяющий выполнить на Вызывной Панели определенную операцию или последовательность операций по команде с Сервера, например, отпереть замок двери (**unlock**), проиграть голосовое сообщение (**play_url**) или установить новый пароль на авторизацию (**change_password**).

Так как Вызывная Панель функционирует в сети строго как клиент, то входящие на неё HTTP запросы обработаны быть не могут. С одной стороны, это нарушило бы принцип "клиент-серверной" архитектуры, с другой - имеется масса технических препятствий на установление TCP соединения от сервера к ВП (как-то NAT и прочие средства защиты сетей). Для того, чтобы сервер мог отправить команду на ВП, в модуле **SuperGlue** предусмотрен механизм, позволяющий устанавливать исходящее от ВП соединение сервером по протоколу HTTP (HTTPS) и отправлять в сервер запрос на получение команды. Если у сервера имеется в распоряжении команда, которую требуется исполнить на данной ВП, то он выдает соответствующий JSON ответ. Если команд для ВП нет, то сервер может удерживать соединение в течении какого-то времени (выполняя HTTP Long Poll), на случай если для данной ВП появится команда на исполнение. По истечению определенного промежутка времени сервер может оборвать соединение, после чего ВП установит новое соединение с сервером для получения команды и т.д.

ВАЖНО: HTTP API запросы выполняются параллельно с запросами модуля **Configurator** на предоставление конфига. Это означает, что возможна ситуация, когда одна и та же Вызывная Панель будет иметь два одновременных соединения с сервером: первое на предоставление конфигурации, второе - запрос на исполнение команд API.

Для включение режима **HTTP API** в конфигурационном файле ВП в секции **Common** необходимо задать значение для следующих параметров:

- **http_api_url** - Указать URL сервера для API запросов. Этот URL может совпадать со значением параметра **config_url** секции **Configurator**.
- **http_api_delay** - Размер временной задержки (в сек) перед каждый HTTP API запросом. По умолчанию, значение данного параметра равно 0 сек, что предполагает наличие на сервере включенной функцией HTTP Long

Poll. Если сервер не поддерживает такую функцию, то постоянные запросы HTTP API могут привести к flood-у (непрерывным повторным TCP соединениям). Чтобы избежать flood-а в данном параметре необходимо задать время задержки в несколько секунд.

- **http_api_enable** - Значение **true** разрешает механизм HTTP API.

3.19.1. Формат запроса от ВП к серверу

Если на ВП разрешен механизм HTTP API, то с интервалом в **http_api_delay** секунд Вызывная Панель будет устанавливать TCP-соединение с сервером и отправлять следующий HTTP запрос:

```
{"command":"request_api_call", "last_serial":"2277",
"auth_cookie":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJjb21tYW5kIjoicmVxdWVzdF9h
dXRoX2Nvb2tpZSIsInJlc3BvbnNlIjoib2siLCJzaWQiOiIxNjI1NDI1MTUwNEU1NDMxMzUzMDMwMzAw
OTQxMEFDRCIsImludGVyY29tX2lkIjoiNyIsImV4cCI6MTY0MjkzMjg0MCwiaWF0IjoxNjI3MzgwODQw
fQ.Rd6NuwNNNT7VmyuNpMW7IgwjEGf88Z_i0noIetB4lGM", "timestamp":"1633204293",
"last_api_response":"OK"}
```

Где,

- **last_serial** - Порядковый номер последнего полученного от сервера запроса. Таким образом ВП сообщает серверу какая команда была выполнена последний раз.
- **last_api_response** - Код ошибки исполнения последней команды от сервера (или OK, если команда исполнена без ошибок).
- **auth_cookie** - Авторизационная кука, точно такая же как и та, что получена при загрузке конфига. Используется сервером для авторизации данного запроса.
- **timestamp** - Временная отметка (UNIX time).

3.19.2. Формат команды в ответе сервера

Ответ сервера должен содержать параметр **response** равный **ok**, что сигнализирут о том, что сервер успешно обработал запрос. Также в ответе сервера должны содержаться следующие параметры:

- **serial** - Содержит серийный номер команды от сервера. ВП сопоставляет значение этого параметра с уже имеющимся значением и если оно меньше или равно текущему, то команда игнорируется. Это позволяет серверу передавать команду в ВП несколько раз подряд с одним и тем же серийным номером, но команда будет исполнения только один раз.

ВП поддерживает следующий набор команд в ответе сервера на HTTP API запрос:

1. Разблокировка дверного замка с идентификатором **front_door_lock**

```
{response:"ok", client_command:"unlock",
lock:"front_door_lock", serial:"1"}
```

2. Проиграть звуковой файл с сообщением:

```
{response:"ok", client_command:"play_url",
url:"http://some.url.com/sound.wav",
aif_name:"FrontSide", serial:"2"}
```

3. Установка нового пароля на авторизацию:

```
{response:"ok", client_command:"change_password",
password_base64:"UGFzc3dvcmRTZXRCeVNlcnZlcg==",
serial:"3"}
```

4. Негативный ответ сервера выглядит следующим образом:

```
{response:"error", command:"request_api_call",
error:"Code not supported"}
```

3.20. Управление внешними входами и выходами

Аппаратное обеспечение Вызывной Панели содержит четыре низковольтных входа (разъемы **XS18-XS21**), предназначенных для подключения внешних кнопок и датчиков, а так же два твердотельных низковольтных реле ***VO14642*** (выходы **XS16, XS17**). Основное назначение реле - коммутировать электромагнитные замки (макс напряжение: 60V, ток: 2A).

Все эти порты управляются через внешний контроллер клавиатуры и GPIO (микросхема **TCA8481**), который сидит на шине **I2C-2**. Поэтому, прежде всего необходимо загрузить драйвер модуль **tca8418_keypad_and_gpio.ko**.

Драйвер экспортирует два системных интерфейса: Input Device (**/dev/input/eventXX**) и GPIO Chip (**/sys/class/gpio/gpiochip100**).

3.20.1. Управление твердотельными реле XS16 и XS17

После загрузки драйвера, управление состоянием реле выглядит так:

```
# Configure XS16 output: GPIO108
echo 108 > /sys/class/gpio/export
echo 0 > /sys/class/gpio/gpio108/value

# Configure XS17 output: GPIO109
echo 109 > /sys/class/gpio/export
echo 0 > /sys/class/gpio/gpio109/value
```

3.20.2. Считывание состояний входных портов XS18, XS19, XS20 и XS21

Так как драйвер экспортирует входы как устройства ввода (читай - клавиатуру), то считывать состояние можно так:

```
root@devuan:~# evtest /dev/input/event4
Input driver version is 1.0.1
Input device ID: bus 0x18 vendor 0x1 product 0x1 version 0x1
Input device name: "tca8418_gpio"
Supported events:
  Event type 0 (EV_SYN)
  Event type 1 (EV_KEY)
  Event type 4 (EV_MSC)
    Event code 4 (MSC_SCAN)
Key repeat handling:
  Repeat type 20 (EV_REP)
    Repeat code 0 (REP_DELAY)
      Value    250
    Repeat code 1 (REP_PERIOD)
      Value     33
Properties:
```

```
Testing ... (interrupt to exit)

Event: time 1590236526.304789, type 4 (EV_MSC), code 4 (MSC_SCAN), value e4
Event: time 1590236526.304796, ------------- SYN_REPORT ------------
Event: time 1590236526.842863, type 4 (EV_MSC), code 4 (MSC_SCAN), value 64
Event: time 1590236526.842868, ------------- SYN_REPORT ------------
Event: time 1590236527.291077, type 4 (EV_MSC), code 4 (MSC_SCAN), value e4
Event: time 1590236527.291081, ------------- SYN_REPORT ------------
Event: time 1590236527.665409, type 4 (EV_MSC), code 4 (MSC_SCAN), value 64
Event: time 1590236527.665412, ------------- SYN_REPORT ------------
Event: time 1590236529.002250, type 4 (EV_MSC), code 4 (MSC_SCAN), value e3
Event: time 1590236529.002254, ------------- SYN_REPORT ------------
Event: time 1590236529.259570, type 4 (EV_MSC), code 4 (MSC_SCAN), value 63
Event: time 1590236529.259574, ------------- SYN_REPORT ------------
Event: time 1590236529.577238, type 4 (EV_MSC), code 4 (MSC_SCAN), value e3
Event: time 1590236529.577242, ------------- SYN_REPORT ------------
Event: time 1590236529.764372, type 4 (EV_MSC), code 4 (MSC_SCAN), value 63
Event: time 1590236529.764376, ------------- SYN_REPORT ------------
```

На каждое изменение состояния порта драйвер формирует отдельное событие - скан-код которого ассоциирован с номером порта. Старший бит в скан-коде указывает на тип события - соединение или разъединение с "землей". В типичной схеме подключение кнопки, нажатие предполагает замыкание на "землю", в этом случае старший бит установлен в "1". При отжатии - соответственно "0".

3.21. Список GPIO для SIPHomePhone V1.3

Аппаратное обеспечение Вызывной Панели содержит несколько устройств, управление которыми осуществляется подачей логической "0" или "1" на соответствующую линию (выход микропроцессора или внешнего контроллера), называется это General Purpose Input Output (**GPIO**). В системе Linux имеется специальная подсистема для этих целей. Драйвера экспортирующие интерфейс в подсистему GPIO появляются как спец файлы в каталоге **/sys/class/gpio**.

В вызывной панели версии 1.2 имеются следующие линии GPIO:

- GPIO1 - port:PC03, Сигнал MUTE для PCM2900 и схемы усилителя линии LN - при подаче значения "1" схема отключена (сигнал MUTE активен). Включать усилитель LN нужно каждый раз при коммутации звонка в аналоговую сеть и выключать при его завершении.
- GPIO2 - port:PC04, Сигнал ENABLE для модуля считывателя меток RFID. Активен при подаче "0" (при подаче "1" считыватель отключен). Включать нужно, скорее всего, один раз при загрузке системы из **/etc/rc.local**.
- GPIO3 - port:PC19, Включение ИК подсветки камеры. Активен при подаче "1". Включать нужно только в ночное время (исходя из показаний Фотодатчика) и только при появлении в зоне видимости камеры объекта (показание проксимити сенсора).
- GPIO4 - port:PC05, Светодиод HEART-BEAT. Активен (светодиод светится) при подаче "1". Этот светодиод нужно включать и выключать с интервалом 1 сек из главного цикла основной программы. Он показывает, что система и главный цикл программного обеспечения функционирует (не завис).
- GPIO108 - col8:TCA8481, управляет твердотельным реле порта XS16. Подача "1" замыкает реле. Данный порт предназначен для подключения основного эл.маг замка двери (основной двери).
- GPIO109 - col9:TCA8481, управляет твердотельным реле порта XS17. Подача "1" замыкает реле. Данный порт предназначен для подключения дополнительного эл.маг замка двери (дополнительной двери).

3.21.1. Изменение значения конкретного GPIO

Драйвера работающие с линиями GPIO экспортируют в систему специальный интерфейс, который действует следующим образом:

Сначала нужно попросить систему экспортировать требуемое GPIO, указав его номер, в файловую систему:

```
# echo 1 > /sys/class/gpio/export
```

После чего в каталоге **/sys/class/gpio/** появится подкататог **gpio1**, в котором будут находится файлы **direction** и **value**.

Чтобы установить GPIO как выходной (output), делаем:

```
# echo out > /sys/class/gpio/gpio1/direction
```

После чего можно записывать "0" или "1" в файл **value**:

```
# echo 0 > /sys/class/gpio/gpio1/value
```

что приводит к снятию сигнала MUTE и включению усилителя линии **«LN»**.

Важно, что GPIO с номерами 100-109, т.е. те, что висят на внешнем контроллере **PCA8481** являются только **output**, в них можно только писать. В связи с чем нет особой надобности устанавливать direction, т.е. они по умолчанию **output** сразу после экспорта.

3.22. Нотификации (оповещения) от вызывной панели

Вызывная панель сообщает о происходящих важных событиях на back-end сервер производя HTTP запросы и передавая в теле запроса JSON структуру определенного формата. В секции **Notify** консолидированного конфига задается сервер для отправки нотификаций (параметр **notify_url**) и разрешаются или запрещаются группы оповещений:

1. **notify_dialing** - сообщения о том, что в зоне видимости вызывной панели кто-то или что-то присутствует.
2. **notify_calling** - группа сообщений и звонках.
3. **notify_ambient** - сообщение о состоянии окружающей среды (на данный момент только интенсивность освещения).
4. **notify_locks** - сообщения об отпирании/запирании замков.
5. **notify_doors** - сообщения о состоянии дверей (датчики на дверях).
6. **notify_buttons** - сообщения о нажатии кнопок отпирания дверей изнутри.

Формат отправляемых JSON сообщений:

1. **{event:«dialing», timestamp:«1599512208.78009», auth_cookie:«1234»}** - Оповещение о том, что кто-то находится вблизи вызывной панели и вероятно набирает номер квартиры. Сообщения этого типа отправляются HTTP запросом типа **multipart/mixed**, а первой части которого содержится JSON с нотификацией, а во второй - файл снимок высокого разрешения с камеры вызывной панели в формате JPEG. Предполагается, что эти снимки будут использованы серверной стороной для системы распознавания лиц.
2. **{event:«calling_direct», apt_num:«100», verbose:«», timestamp:«1599513104.79772», auth_cookie:«1234»}** - Оповещение о том, что произведен вызов аналоговой трубки, подключенной напрямую к вызывной панели.
3. **{event:«calling_direct_connected», apt_num:«100», verbose:«», timestamp:«1599513104.79772», auth_cookie:«1234»}** - Оповещение о том, что вызов с аналоговой трубкой, подключенной напрямую, установлен (трубка поднята).
4. **{event:«calling_switch», apt_num:«104», verbose:«0x20002@1», timestamp:«1599513104.79772», auth_cookie:«1234»}** - Оповещения о том, что произведен вызов абонента (помещения) **apt_num** через аналоговую домофонную сеть.

5. **{event:«calling_switch_connected», apt_num:«104», verbose:«0x20002@1», timestamp:«1599513147.81017», auth_cookie:«1234»}** - Оповещение о том, что соединение с абонентом в аналоговой домофонной сети установлено (поднята трубка домофона).
6. **{event:«calling_switch_proceeded», apt_num:«104», verbose:«0x20002@1», timestamp:«1599513152.40904», auth_cookie:«1234»}** - Оповещение о том, что сейчас идет разговор по аналоговой домофонной сети.
7. **{event:«calling_emergency», apt_num:«911», verbose:«4294967295@1», timestamp:«1599513175.67322», auth_cookie:«1234»}** - Оповещение том, что производится вызов всех абонентов в аналоговой сети (вызов ГОиЧС).
8. **{event:«calling_emergency_connected», apt_num:«911», verbose:«4294967295@1», timestamp:«1599513180.94799», auth_cookie:«1234»}** - Оповещение о том, что вызов ГОиЧС установлен.
9. **{event:«calling_emergency_proceeded», apt_num:«911», verbose:«4294967295@1», timestamp:«1599513186.00437», auth_cookie:«1234»}** - Оповещение о том, что на данный момент ведется проигрывание звукового оповещения ГОиЧС.
10. **{event:«calling_disconnected», apt_num:«104», verbose:«0x20002@1», timestamp:«1599513163.46443», auth_cookie:«1234»}** - Оповещение о том, что вызов (разговор) окончен или не состоялся.
11. **{event:«button_pressed», button_name:«back_door_exit_button», verbose:«», timestamp:«1599517070.87237», auth_cookie:«1234»}** - Оповещает о том, что нажата кнопка открывания двери изнутри.
12. **{event:«lock_unlocked», lock_name:«back_door_lock», verbose:«button:back_door_exit_button», timestamp:«1599517071.23781», auth_cookie:«1234»}** - Оповещает о том, что замок был разблокирован нажатием кнопки.
13. **{event:«lock_unlocked», lock_name:«front_door_lock», verbose:«apt:101,rfid_key:565AC70EC1,rfid_type:emmarin», timestamp:«1599604707.80272», auth_cookie:«1234»}** - Оповещает о том, что замок был разблокирован с помощью RFID ключа.
14. **{event:«door_openned», door_name:«back_door_sensor», verbose:«», timestamp:«1599517247.62742», auth_cookie:«1234»}** - Оповещает о том, что дверь была открыта.

15. **{event:«door_closed», door_name:«back_door_sensor», verbose:«», timestamp:«1599517249.72764», auth_cookie:«1234»}** - Оповещает о том, что дверь теперь закрыта.

Описание параметров:

1. **apt_num** - набранный номер квартиры или идентификатор помещения.
2. **verbose** - дополнение к сообщению, для звонков содержит некий физический адрес устройства или SIP .
3. **auth_cookie** - авторизационная кука, которую домофон получил после выполнения процедуры авторизации на back-end сервере.

3.23. Пользовательские коды (User Code)

Пользовательские коды – это механизм позволяющий выполнить на вызывной панели определенную операцию или последовательность операций по заранее запрограммированному коду, который вводится с панели и отправляется в сервер нажатием кнопки «вызов». Можно провести аналогию с USSD кодами в мобильных GSM сетях. Пользовательский код в вызывной панели всегда начинается с символа '0' и может иметь произвольную длину, но, как правило, ограничен полем ввода графического интерфейса на вызывной панели.

На данный момент интерфейс на вызывной панели позволяет вводить коды длиной четыре знака, таким образом ввод последовательности 0XXX и нажатие кнопки «вызов» рассматривается как «пользовательский код», а не номер помещения.

Когда пользователь вводит на панели последовательность цифр и нажимает кнопку «вызов», модуль **SuperGlue** распознает пользовательский код по наличию префикса '0' и формирует HTTP запрос в сервер с этим кодом. Формат запроса следующий:

```
{command:"user_code", user_code:"%s", timestamp:"%s", auth_cookie:"%s"}
```

Сервер, получив такой запрос, проводит его авторизацию по переданной **auth_cookie**, определяет принадлежность вызывной панели и формирует ответ, исходя из своей логики и значения переданного в поле **user_code**. Ответ сервера может содержать имя команды к исполнению на вызывной панели, а так же ряд параметров для неё.

Ответа сервера:

```
{response:"ok", command:"user_code", client_command:"unlock",
lock:"front_door_lock"}
```

Получив такой ответ, машина состояний внутри модуля SuperGlue исполняет внутренний сценарий с названием **unlock** и передает ему параметр **lock:front_door_lock**. Данный сценарий производит отпирание замка с именем **front_door_lock**.

На данный момент, это единственный поддерживаемый сценарий, но может быть легко расширен.

Негативный ответ сервера выглядит следующим образом:

```
{response:"error", command:"user_code", error:"Code not supported"}
```

Данный механизм может быть использован для реализации услуги «временных кодов для открывания двери».

Адрес сервера для запроса задается в конфиге в параметре **Config → config_url**

3.24. Валидация конфига

Модуль ***Configurator*** перед окончательной установкой нового конфига должен удостоверится в его работоспособности. Для этого он отправляет всем модулям команду типа:

```
MsgPrefix: user TAG:all TEXT:{command:"validate_config",
file:"/full/path/name/all.json"}
```

Приняв это сообщение все модули проверяют конфиг на наличие нужных им для старта секций, и отвечают командами:

```
MsgPrefix: user TAG:all TEXT:{command:"validate_config_ok",
file:"/full/path/name/all.json", module_name:"MODULE_NAME"}
```

либо, если есть ошибки,

```
MsgPrefix: user TAG:all TEXT:{command:"validate_config_error",
file:"/full/path/name/all.json", error:"секция такая-то отсутствует",
module_name:"MODULE_NAME"}
```

где MODULE_NAME - имя модуля в системе SunBus.

ОБЩЕСТВО С ОГРАНИЧЕННОЙ ОТВЕТСТВЕННОСТЬЮ
«ФАБМИКРО»

ВЫЗЫВНАЯ ПАНЕЛЬ

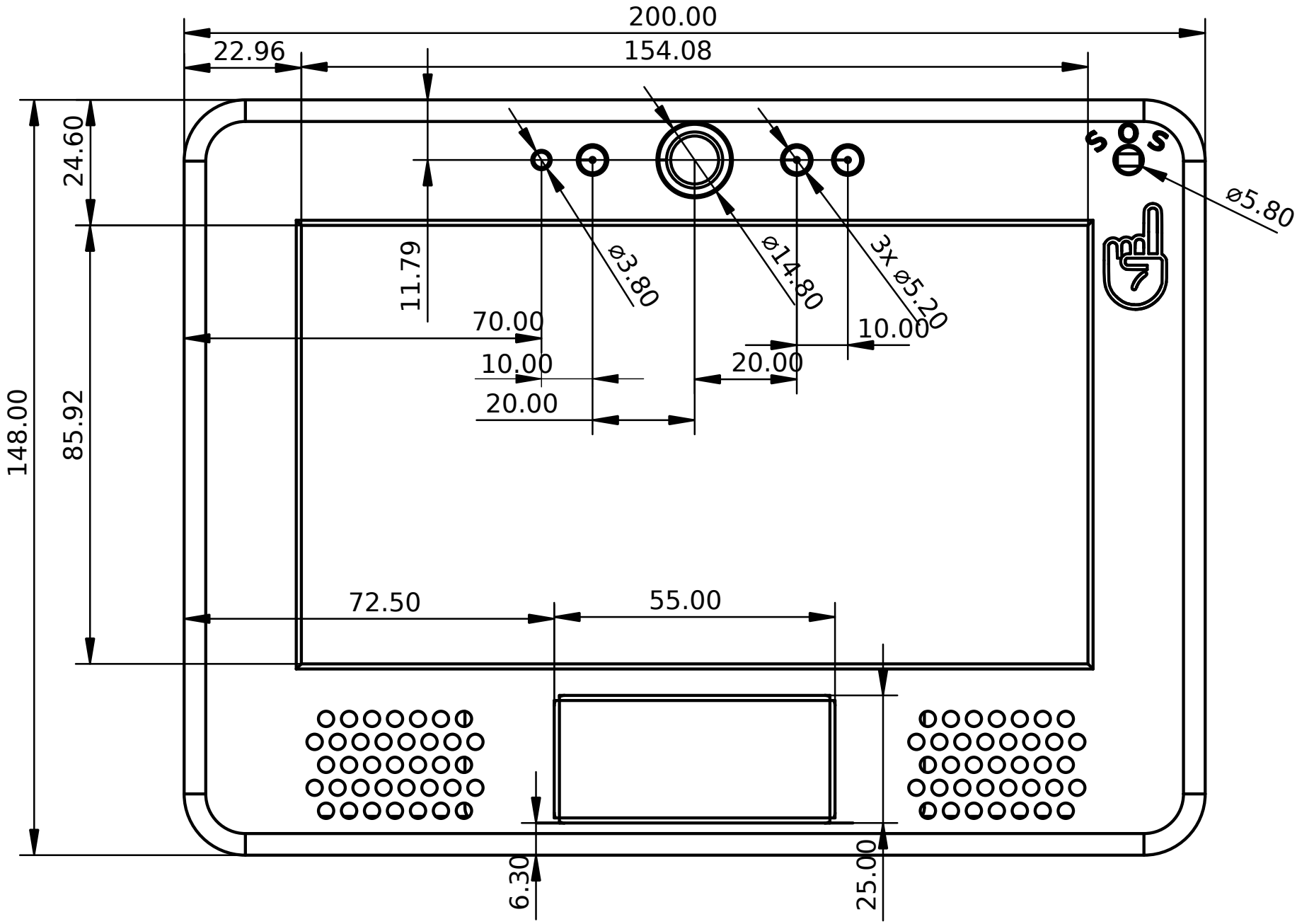
«ЕРМАК»

(SIPHomePhone версия 1.3)

4. Приложения

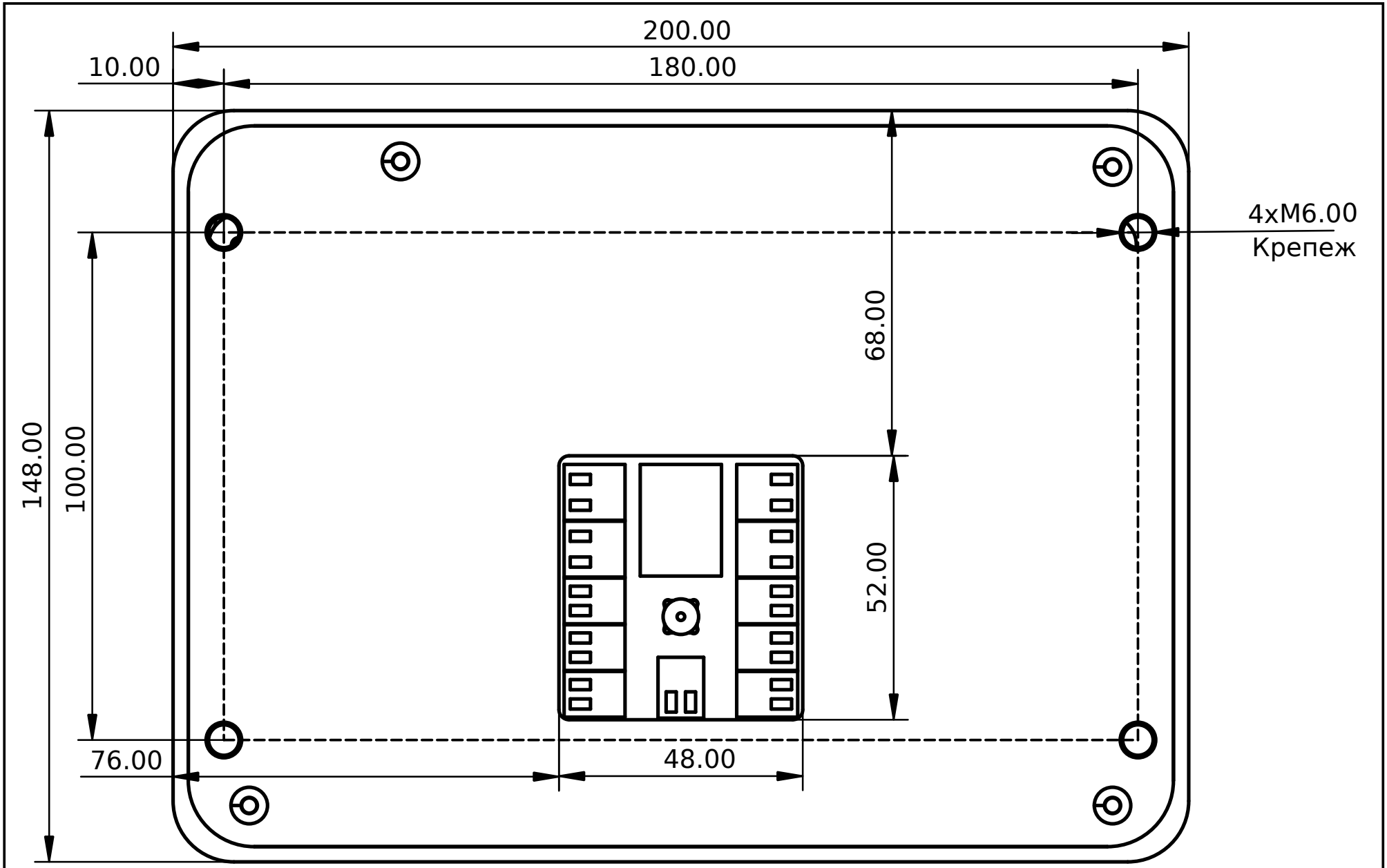
Тюмень 2021

Приложение 1. Габаритный чертеж ВП.



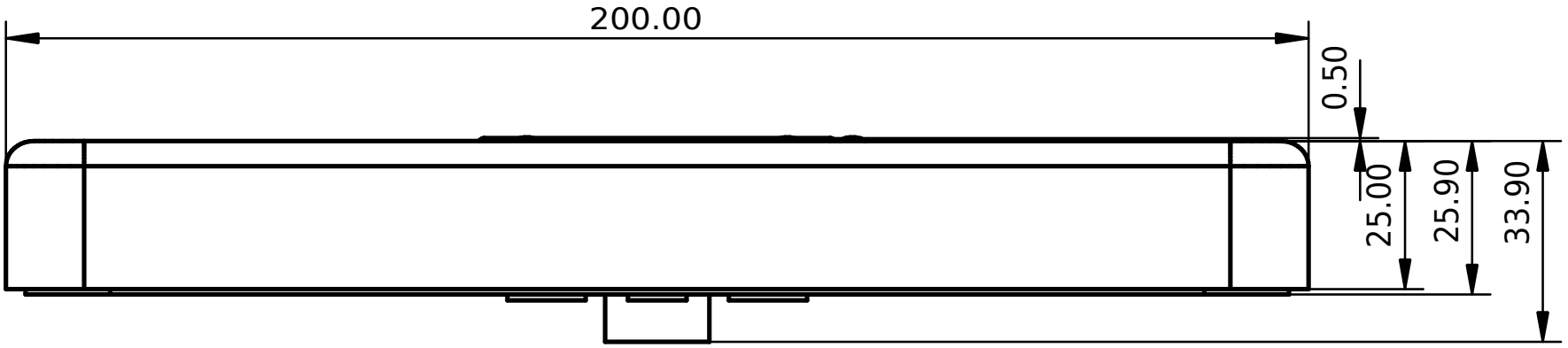
Изм	Лист	№ докум.	Подп.	Дата	Вызывная панель. Вид спереди.	Лист 1

Формат А4



					Вызывная панел. Вид сзади.	Лист
Изм	Лист	№ докум.	Подп.	Дата		1

Формат А4



					Вызывная панель. Вид сверху.	Лист
Изм	Лист	№ докум.	Подп.	Дата		1

Формат А4

Приложение 2. Структура конфигурационного файла

Консолидированный конфигурационный файл представляет собой JSON структуру которая содержит список объектов JSON, каждый из которых представляет секцию конфигурации соответствующего модуля или подсистемы программного обеспечения вызывной панели. Версия 1.3 (SIPHomePhone/1.3) конфигурационного файла содержит следующие секции:

- "Advertizer" — содержит описание страниц с объявлениями.
- "AmbientSensor" — содержит настройки модуля, обслуживающего датчик освещенности.
- "Apartments" — содержит список помещений и их индивидуальные настройки: наименование, таблицу маршрутизации вызовов, RFID ключи.
- "BacklightManager" — содержит настройки модуля, обслуживающего подсветку экрана.
- "Common" — содержит общие настройки вызывной панели.
- "Config" — содержит описание данного конфигурационного файла, источник его происхождения, версию структуры и серийный номер.
- "Configurator" - содержит настройки для модуля Configurator, управляющего конфигурационным спулом и обеспечивающего доставку конфига.
- "DoorsAndButtons" — содержит настройки для модуля DoorAndButtons ответственного за работу с кнопками отпирания замков и датчиками положения двери.
- "GUI" — содержит настройки модуля GUI, обеспечивающего графический интерфейс, в тот числе структуру меню пользователя, шрифты и внешний вид. В этой же секции задается амплуа устройства.
- "HandsetState" — содержит настройки модуля HandsetState, отвечающего за контроль состояния абонентской аналоговой трубки.
- "HTTPRequester" — содержит настройки модуля HTTPRequester, предоставляющего другим модулям интерфейс для работы с HTTP запросами.
- "LockManager" — содержит настройки модуля LockManager, отвечающего за работу с ЭМ замками.
- "Notify" — содержит настройки модуля Notify, обеспечивающего поток нотификационных сообщений о состоянии ВП в сервер.
- "ProximitySensor" — содержит настройки модуля ProximitySensor, обеспечивающего работу с датчиком приближения.

- "RFIDReader" — содержит настройки модуля RFIDReader, обеспечивающего работу с аппаратным считывателем RFID меток.
- "RS485Master" — содержит настройки модуля RS485Master обеспечивающего интерфейс для взаимодействия с домофонным коммутатором «ЕРМАК», а так же с любыми другими устройствами поддерживающими протокол Modbus/RTU.
- "Yard" — содержит конфигурацию «двора», в случае если Вызывная Панель выступает в роли панели на калитке при входе во двор МКД.

JSON схема секции Advertizer

```
{
        "description":"Данные для загрузки рекламных слайдов и объявлений",
        "items":{
                "properties":{
                        "banner_id":{
                                "default":"main_window_banner",
                                "description":"Уникальный идентификатор рекламного объявления",
                                "enum":[
                                        "main_window_banner",
                                        "dial_left_banner"
                                ],
                                "type":"string"
                        },
                        "slides":{
                                "description":"Массив описания рекламных слайдов",
                                "format":"table",
                                "items":{
                                        "properties":{
                                                "descr":{
                                                        "description":"Комментарий к данному
слайду",
                                                        "type":"string"
                                                },
                                                "dwell_time":{
                                                        "default":"5",
                                                        "description":"Время удержания слайда на
экране",
                                                        "type":"integer"
                                                },
                                                "url":{
                                                        "description":"Ссылка на https:// ресурс
содержащий картинку слайда",
                                                        "type":"string"
                                                },
                                                "video_fx":{
                                                        "default":"right1",
                                                        "description":"Видео-эффект для замены
одного слайда на другой",
                                                        "enum":[
                                                                "right1",
                                                                "right2",
                                                                "right3",
                                                                "left1",
                                                                "left2",
                                                                "left3",
                                                                "up1",
                                                                "up2",
                                                                "up3",
                                                                "down1",
                                                                "down2",
                                                                "down3"
                                                        ],
                                                        "type":"string"
                                                }
                                        },
                                        "type":"object"
                                },
                                "type":"array",
                                "uniqueItems":true
                        }
                },
                "type":"object"
        },
        "title":"Advertizer",
        "type":"array",
        "uniqueItems":true
}
```

JSON схема секции Ambient

```
{
        "description":"Параметры датчика освещенности",
```

```
        "properties":{
                "ambient_hysteresis":{
                        "default":"30",
                        "description":"Гистерезис переключения",
                        "maximum":"4096",
                        "minimum":"0",
                        "type":"integer"
                },
                "ambient_threshold":{
                        "default":"700",
                        "description":"Порог переключения день/ночь",
                        "maximum":"4096",
                        "minimum":"0",
                        "type":"integer"
                },
                "event_interval":{
                        "default":"0",
                        "description":"Период (сек) формирования событий",
                        "maximum":"60",
                        "minimum":"0",
                        "type":"integer"
                },
                "hwmon_device_name":{
                        "default":"ads1000",
                        "readOnly":"true",
                        "type":"string"
                },
                "infrared_backlight_disable_value":{
                        "default":"0",
                        "maximum":"255",
                        "minimum":"0",
                        "readOnly":"true",
                        "type":"integer"
                },
                "infrared_backlight_enable_value":{
                        "default":"1",
                        "maximum":"255",
                        "minimum":"0",
                        "readOnly":"true",
                        "type":"integer"
                },
                "infrared_backlight_gpio":{
                        "default":"/sys/class/gpio/gpio3_pc19/value",
                        "readOnly":"true",
                        "type":"string"
                }
        },
        "title":"AmbientSensor",
        "type":"object"
}
```

JSON схема секции Apartments

```
{
        "description":"Описание помещений, персональных настроек, ключей и маршрутизации звонков",
        "items":{
                "properties":{
                        "apt_num":{
                                "default":"1",
                                "description":"Номер помещения",
                                "type":"string"
                        },
                        "calling_image":{
                                "default":"",
                                "description":"https:// адрес персонализированного изображения при
звонке",
                                "type":"string"
                        },
                        "calling_sound":{
                                "default":"",
                                "description":"https:// адрес персонализированного звукового файла
при звонке",
                                "type":"string"
                        },
                        "camera_bitrate":{
                                "default":"1000",
                                "description":"Поток КБит/с",
                                "type":"integer"
                        },
                        "camera_format":{
                                "default":"640x480",
                                "description":"Размер видеокадра для звонка",
                                "type":"string"
                        },
                        "camera_name":{
                                "default":"Internal",
                                "description":"Идентификатор камеры",
                                "type":"string"
                        },
                        "codec_key_interval":{
                                "default":"15",
                                "description":"Интенсивность следования ключевых кадров звонка",
                                "type":"integer"
                        },
                        "descr":{
                                "default":"",
                                "description":"Описание или Ф.И.О. собственника",
                                "type":"string"
                        },
                        "descr_font":{
                                "description":"Шрифт для текста на клавишах",
                                "enum":[
                                        "FONT1",
                                        "FONT2",
                                        "FONT3"
                                ],
                                "type":"string"
                        },
                        "keys":{
                                "description":"Список ключей ассоциированных с данным помещением",
                                "format":"table",
                                "items":{
                                        "properties":{
                                                "descr":{
                                                        "default":"",
                                                        "description":"Комментарий",
```

```
                                        "type":"string"
                                },
                                "key":{
                                        "default":"001122334455",
                                        "description":"Номер ключа",
                                        "type":"string"
                                },
                                "type":{
                                        "description":"Тип RFID ключа",
                                        "enum":[
                                                "emmarine"
                                        ],
                                        "type":"string"
                                }
                        },
                        "type":"object"
                },
                "type":"array",
                "uniqueItems":true
        },
        "lock_open_delay":{
                "default":"0",
                "description":"Персональная задержка (в сек) перед отпиранием
замка",
                "type":"integer"
        },
        "locks":{
                "description":"Список идентификаторов замков доступных к отпиранию",
                "format":"table",
                "items":{
                        "type":"string"
                },
                "type":"array",
                "uniqueItems":true
        },
        "routes":{
                "description":"Список маршрутов для доставки звонка при вызове с
Панели",
                "format":"table",
                "items":{
                        "properties":{
                                "D":{
                                        "default":"0x0001",
                                        "description":"Адрес на коммутаторе:
десятки",
                                        "type":"integer"
                                },
                                "E":{
                                        "default":"0x0001",
                                        "description":"Адрес на коммутаторе:
единицы",
                                        "type":"integer"
                                },
                                "UriFrom":{
                                        "default":"user@domain.com",
                                        "description":"Исходящий SIP URI",
                                        "type":"string"
                                },
                                "UriTo":{
                                        "default":"user@domain.com",
                                        "description":"SIP URI куда отправить
звонок",
                                        "type":"string"
                                },
```

```
                                        "authUsername":{
                                                "default":"",
                                                "description":"Имя пользователя для
авторизации SIP звонка",
                                                "type":"string"
                                        },
                                        "display":{
                                                "default":"Random Stranger",
                                                "description":"Отображаемое имя",
                                                "type":"string"
                                        },
                                        "modbus_address":{
                                                "default":"1",
                                                "description":"Modbus адрес аналогового
коммутатора",
                                                "type":"integer"
                                        },
                                        "pattern":{
                                                "default":"",
                                                "description":"Идентификатор SIP паттерна",
                                                "type":"string"
                                        },
                                        "proxy":{
                                                "default":"192.168.1.1",
                                                "description":"Адрес SIP прокси",
                                                "type":"string"
                                        },
                                        "proxy_port":{
                                                "default":"5060",
                                                "description":"Номер порта SIP прокси",
                                                "type":"string"
                                        },
                                        "secret":{
                                                "default":"",
                                                "description":"Пароль для авторизаии SIP
звонка",
                                                "type":"string"
                                        },
                                        "type":{
                                                "description":"Тип звонка: 'прямой', 'через
аналоговый коммутатор', 'через SIP'",
                                                "enum":[
                                                        "sip",
                                                        "direct",
                                                        "switch",
                                                        "emergency"
                                                ],
                                                "type":"string"
                                        }
                                },
                                "type":"object"
                        },
                        "type":"array",
                        "uniqueItems":true
                },
                "unlocking_image":{
                        "default":"",
                        "description":"https:// адрес персонализированного изображения при
отпирании замка",
                        "type":"string"
                },
                "unlocking_sound":{
                        "default":"",
```

```
                                "description":"https:// адрес персонализированного звукового файла
при отпирании замка",
                                "type":"string"
                        },
                        "unlocking_sound_rfid":{
                                "default":"",
                                "description":"https:// адрес персонализированного звукового файла
при отпирании замка RFID ключом",
                                "type":"string"
                        }
                },
                "type":"object"
        },
        "title":"Apartments",
        "type":"array",
        "uniqueItems":true
}
```

JSON схема секции BacklightManager

```
{
        "description":"Настройки подсветки дисплея",
        "properties":{
                "pwm_file":{
                        "default":"/sys/devices/virtual/pwm-sunxi/pwm0/duty_percent",
                        "readOnly":"true",
                        "type":"string"
                },
                "pwm_is_inverted":{
                        "default":true,
                        "readOnly":"true",
                        "type":"boolean"
                },
                "pwm_max":{
                        "default":"100",
                        "description":"ШИМ скважность максимальной яркости",
                        "maximum":"101",
                        "minimum":"0",
                        "readOnly":"true",
                        "type":"integer"
                },
                "pwm_min":{
                        "default":"1",
                        "description":"ШИМ скважность минимальной яркости",
                        "maximum":"101",
                        "minimum":"0",
                        "readOnly":"true",
                        "type":"integer"
                }
        },
        "title":"BacklightManager",
        "type":"object"
}
```

JSON схема секции Common

```
{
        "description":"Общие настройки системы",
        "properties":{
                "SipPatterns":{
```

```
                        "description":"Массив описания типовых настроек (паттернов) для SIP
серверов",
                        "format":"table",
                        "items":{
                                "properties":{
                                        "AuthSecret":{
                                                "default":"",
                                                "description":"Пароль",
                                                "type":"string"
                                        },
                                        "AuthUsername":{
                                                "default":"",
                                                "description":"Имя пользователя для авторизации",
                                                "type":"string"
                                        },
                                        "UriFrom":{
                                                "default":"domofon@sip.t72.ru",
                                                "description":"Исходящий URI",
                                                "type":"string"
                                        },
                                        "display":{
                                                "default":"Susan Vega",
                                                "description":"Отображаемое имя",
                                                "type":"string"
                                        },
                                        "pattern_id":{
                                                "default":"RussianCompany",
                                                "description":"Уникальный идентификатор",
                                                "type":"string"
                                        },
                                        "proxy":{
                                                "default":"sip.t72.ru:5060",
                                                "description":"Адрес SIP прокси",
                                                "type":"string"
                                        },
                                        "register":{
                                                "default":false,
                                                "description":"Регистрироваться на прокси или нет",
                                                "type":"boolean"
                                        }
                                },
                                "type":"object"
                        },
                        "type":"array",
                        "uniqueItems":true
                },
                "config_url":{
                        "default":"",
                        "description":"Ссылка на ресурс сервера для получения конфигурации",
                        "type":"string"
                },
                "default_call_camera_bitrate":{
                        "default":"1000",
                        "description":"поток КБит/с для звонков",
                        "type":"integer"
                },
                "default_call_camera_format":{
                        "default":"640x480",
                        "description":"Размер видеокадра для звонков",
                        "type":"string"
                },
                "default_call_camera_frame_rate":{
                        "default":"15",
                        "description":"Частота кадров при звонке",
                        "type":"integer"
                },
                "default_call_camera_name":{
                        "default":"Internal",
                        "description":"Камера для звонков",
                        "type":"string"
                },
                "default_call_codec_key_interval":{
                        "default":"15",
```

```
        "description":"Регулярность следования ключевых кадров при звонке",
        "type":"integer"
},
"default_incoming_call_sound":{
        "default":"file:///home/fabmicro/SIPHomePhone/sounds/incoming_call.wav",
        "description":"Звуковой файл 'входящий вызов'",
        "type":"string"
},
"default_incoming_call_volume":{
        "default":"100",
        "description":"Громкость проигрывания звукового файла при входящем вызове",
        "type":"integer"
},
"default_mchs_sound":{
        "default":"file:///home/fabmicro/SIPHomePhone/sounds/default_mchs.wav",
        "description":"Звуковой файл содержащий оповещение ГОиЧС",
        "type":"string"
},
"default_surveillance_camera_bitrate":{
        "default":"1500",
        "description":"поток КБит/с для видеонаблюдения",
        "type":"integer"
},
"default_surveillance_camera_format":{
        "default":"1920x1080",
        "description":"Размер видеокадра для видеонаблюдения",
        "type":"string"
},
"default_surveillance_camera_frame_rate":{
        "default":"15",
        "description":"Частота кадров при видеонаблюдении",
        "type":"integer"
},
"default_surveillance_camera_name":{
        "default":"Internal",
        "description":"Камера для видеонаблюдения",
        "type":"string"
},
"default_surveillance_codec_key_interval":{
        "default":"15",
        "description":"Регулярность следования ключевых кадров при видеонаблюдении",
        "type":"integer"
},
"direct_call_max_alerting_time":{
        "default":"30",
        "description":"Максимальное время нахождения в предответном состоянии",
        "maximum":"65535",
        "minimum":"0",
        "type":"integer"
},
"direct_call_max_emergency_time":{
        "default":"180",
        "description":"Максимальная продолжительность оповещения ГОиЧС",
        "maximum":"65535",
        "minimum":"0",
        "type":"integer"
},
"direct_call_max_talking_time":{
        "default":"60",
        "description":"Максимальная продолжительность звонка",
        "maximum":"65535",
        "minimum":"0",
        "type":"integer"
},
"error_sound":{
        "default":"file:///home/fabmicro/SIPHomePhone/sounds/error.wav",
        "description":"Звуковой файл 'ошибка'",
        "type":"string"
},
"http_api_delay":{
        "default":"5",
        "description":"Задержка в сек между HTTP API запросами",
        "type":"integer"
```

```
                },
                "http_api_enable":{
                        "default":false,
                        "description":"Разрешить HTTP API запросы в сервер",
                        "type":"boolean"
                },
                "http_api_url":{
                        "default":"",
                        "description":"Ссылка на ресурс сервера для API запросов",
                        "type":"string"
                },
                "send_unlock_dtmf":{
                        "default":"1#",
                        "description":"DTMF посылка для отпирания удаленного замка",
                        "type":"string"
                },
                "sip_call_max_talking_time":{
                        "default":"60",
                        "description":"Максимальная продолжительность SIP звонка",
                        "maximum":"65535",
                        "minimum":"0",
                        "type":"integer"
                },
                "sip_drop_call_after_unlock_sent":{
                        "default":false,
                        "description":"Разъединять сразу после посылки команды 'отпереть замок'",
                        "type":"boolean"
                },
                "sip_incoming_call_uri_match":{
                        "default":"^000$",
                        "description":"RegExp для входящих звонков на Панель",
                        "type":"string"
                },
                "sip_incoming_surveillance_uri_match":{
                        "default":"^696969@",
                        "description":"RegExp для входящих звонков видеонаблюдения",
                        "type":"string"
                },
                "stun_host":{
                        "default":"stun.12connect.com",
                        "description":"IP адрес сервера STUN",
                        "type":"string"
                },
                "stun_port":{
                        "default":"3478",
                        "description":"Номер порта STUN",
                        "maximum":"65535",
                        "minimum":"0",
                        "type":"integer"
                },
                "syslog_host":{
                        "default":"217.116.57.130",
                        "description":"IP адрес сервера syslog",
                        "type":"string"
                },
                "syslog_port":{
                        "default":"514",
                        "description":"Номер порта syslog",
                        "maximum":"65535",
                        "minimum":"0",
                        "type":"integer"
                },
                "syslog_progname":{
                        "default":"SIP_HomePhone",
                        "description":"Имя программы в системе syslog",
                        "type":"string"
                }
        },
        "title":"Common",
        "type":"object"
}
```

JSON схема секции Config

```
{
        "description":"Параметры конфигурационного файла",
        "properties":{
                "config_url":{
                        "default":"",
                        "description":"Ссылка URL сервера для загрузки конфига",
                        "readOnly":true,
                        "type":"string"
                },
                "serial":{
                        "default":"0",
                        "description":"Серийный номер конфига",
                        "readOnly":true,
                        "type":"integer"
                },
                "source":{
                        "default":"default_config",
                        "description":"Строка описывающая кто сгенерировал даннный конфиг",
                        "readOnly":true,
                        "type":"string"
                },
                "version":{
                        "default":"SIPHomePhone/1.3",
                        "description":"Номер версии структуры конфига (версия ПО)",
                        "readOnly":true,
                        "type":"string"
                }
        },
        "title":"Config",
        "type":"object"
}
```

JSON схема секции Configurator

```
{
        "description":"Модуль управления конфигами",
        "properties":{
                "auth_cookie_file_name":{
                        "default":"auth_cookie",
                        "description":"Имя файла содержащего авторизационную куку",
                        "readOnly":true,
                        "type":"string"
                },
                "config_delay":{
                        "default":"0",
                        "description":"Время в сек задержки перед запросом конфига с сервера, для
предотвращения flood-a",
                        "readOnly":true,
                        "type":"string"
                },
                "config_file_name":{
                        "default":"/var/configs",
                        "description":"Имя файла текущего конфига",
                        "readOnly":true,
                        "type":"string"
                },
                "config_spool_dir":{
                        "default":"/var/configs",
                        "description":"Каталог содержащий спул конфигов",
                        "readOnly":true,
                        "type":"string"
```

```
                },
                "default_config":{
                        "default":"/home/fabmicro/SIPHomePhone/all.json.default",
                        "description":"Имя файла содержащего конфиг по умолчанию (заводской
конфиг)",
                        "readOnly":true,
                        "type":"string"
                },
                "dhcp_lease_file":{
                        "default":"/var/lib/dhcp/dhclient.eth0.leases",
                        "description":"Имя файл в котором DHCP клиент сохраняет лизы",
                        "readOnly":true,
                        "type":"string"
                },
                "dhcp_option_config":{
                        "default":"siphomephone-config",
                        "description":"Имя опции DHCP переменной содержащей ссылку на URL для
загрузки конфига",
                        "readOnly":true,
                        "type":"string"
                },
                "number_of_configs":{
                        "default":"10",
                        "description":"Максимальное число конфигов удерживаемых в спуле",
                        "readOnly":true,
                        "type":"integer"
                },
                "set_user":{
                        "default":"nobody",
                        "description":"Имя системного пользователя, от которого исполняется процесс
и которому принадлежит содержимое конфигурационного спула",
                        "readOnly":true,
                        "type":"string"
                },
                "sid_eeprom_file":{
                        "default":"/sys/devices/platform/sunxi_sid.0/eeprom",
                        "description":"Имя файла для считывания SID-а устройства",
                        "readOnly":true,
                        "type":"string"
                }
        },
        "title":"Configurator",
        "type":"object"
}
```

JSON схема секции DoorAndButtons

```
{
        "description":"Настройки датчиков состояния дверей и кнопок открывания",
        "properties":{
                "event_interval":{
                        "default":"0",
                        "maximum":"60",
                        "minimum":"0",
                        "readOnly":true,
                        "type":"integer"
                },
                "input_device_name":{
                        "default":"tca8418_gpio",
                        "readOnly":"true",
                        "type":"string"
                },
                "sensors":{
```

```
                        "description":"Список датчиков",
                        "format":"table",
                        "items":{
                                "properties":{
                                        "lock":{
                                                "default":"",
                                                "description":"Идентификатор замка ассоциированный с
кнопкой",
                                                "type":"string"
                                        },
                                        "name":{
                                                "default":"front_door_exit_button",
                                                "description":"Идентификатор датчика",
                                                "type":"string"
                                        },
                                        "polarity":{
                                                "default":"normal",
                                                "description":"Полярность срабатывания",
                                                "enum":[
                                                        "normal",
                                                        "inversed"
                                                ],
                                                "type":"string"
                                        },
                                        "scan_code":{
                                                "default":"1",
                                                "description":"Скан-код",
                                                "type":"integer"
                                        },
                                        "type":{
                                                "default":"button",
                                                "description":"Тип датчика: дверь или кнопка",
                                                "enum":[
                                                        "door",
                                                        "button"
                                                ],
                                                "type":"string"
                                        }
                                },
                                "type":"object"
                        },
                        "type":"array",
                        "uniqueItems":true
                }
        },
        "title":"DoorsAndButtons",
        "type":"object"
}
```

JSON схема секции GUI

```
{
        "description":"Настройки графического интерфейса",
        "properties":{
                "Aliases":{
                        "default":[
                                {
                                        "ani":"arial_ru.ttf",
                                        "color":"0xffe31e24",
                                        "font_size":"16",
                                        "name":"FONT1"
                                },
                                {
                                        "ani":"arial_ru.ttf",
                                        "color":"0xffe31e24",
                                        "font_size":"24",
```

```
                        "name":"FONT2"
                },
                {
                        "ani":"arial_ru.ttf",
                        "color":"0xffe31e24",
                        "font_size":"32",
                        "name":"FONT3"
                },
                {
                        "ani":"arial_ru.ttf",
                        "color":"0xffe31e24",
                        "font_size":"48",
                        "name":"FONT4"
                },
                {
                        "ani":"arial_ru.ttf",
                        "color":"0xffe31e24",
                        "font_size":"64",
                        "name":"FONT5"
                }
        ],
        "description":"Список шрифтов",
        "format":"table",
        "items":{
                "properties":{
                        "ani":{
                                "default":"arial_ru.ttf",
                                "title":"имя файла содержащего шрифт",
                                "type":"string"
                        },
                        "color":{
                                "default":"0xffffffff",
                                "title":"цвет",
                                "type":"string"
                        },
                        "font_size":{
                                "default":"16",
                                "title":"размер шрифта",
                                "type":"integer"
                        },
                        "name":{
                                "default":"FONT1",
                                "title":"имя шрифта",
                                "type":"string"
                        }
                },
                "type":"object"
        },
        "type":"array"
},
"BackgroundColor":{
        "default":"0xffffffff",
        "description":"Цвет фона экрана. 0-черный, 0xffffffff-белый",
        "type":"string"
},
"BaseMenu":{
        "default":"keyboard",
        "description":"второе меню, в которое переходим из стартового",
        "enum":[
                "keyboard",
                "buttons_1",
                "buttons_2",
                "buttons_3",
                "buttons_4",
```

```
                        "yard"
                ],
                "type":"string"
        },
        "BrightnessApproaching":{
                "default":"100",
                "description":"Яркость дисплея в режиме approaching (вдали кто-то есть)",
                "type":"integer"
        },
        "BrightnessDialing":{
                "default":"100",
                "description":"Яркость дисплея в режиме dialing (активный режим)",
                "type":"integer"
        },
        "BrightnessStandBye":{
                "default":"100",
                "description":"Яркость дисплея в режиме stand-bye (никого нет)",
                "type":"integer"
        },
        "CommonMenu":{
                "default":"common",
                "description":"Это файл для объявления общих для всех меню переменных",
                "readOnly":true,
                "type":"string"
        },
        "DelayBeforeStandBye":{
                "default":"20",
                "description":"Время неактивности перед переключением в режим stand-bye(в
секундах)",
                "type":"integer"
        },
        "Dependencies":{
                "default":[
                        "call_5",
                        "income",
                        "picture",
                        "about",
                        "keyboard_building"
                ],
                "description":"Список всех используемых меню",
                "format":"table",
                "items":{
                        "type":"string"
                },
                "type":"array",
                "uniqueItems":true
        },
        "FrameBufferDev":{
                "default":"/dev/fb0",
                "description":"Местоположение драйвера framebuffer",
                "readOnly":true,
                "type":"string"
        },
        "ImagesDir":{
                "default":"images",
                "description":"папка в которой лежат графические ресурсы(картинки, фонты)",
                "readOnly":true,
                "type":"string"
        },
        "InputDev":{
                "default":"",
                "description":"Внутреннее имя тач-скрина. Для резистивного экрана - 'sun4i-
ts', для сенсорного 'ft5x_ts'",
                "readOnly":true,
```

```
                        "type":"string"
                },
                "InputFuzzTime":{
                        "default":"0",
                        "description":"Время, в ms, подавления дребезга тач-скрина. Если между двумя
нажатиями время будет меньше данного, то они отработаются как одно",
                        "type":"integer"
                },
                "MainMenu":{
                        "default":"main",
                        "description":"Стартовое меню. Должно лежать в menu/, иметь
расширение .json",
                        "readOnly":true,
                        "type":"string"
                },
                "ScreenRotate":{
                        "default":"0",
                        "description":"Ориентация панели. 0-нормальная, 1-по часовой,2-вверх
ногами,3-против часовой",
                        "type":"integer"
                },
                "TimeoutAboutMenu":{
                        "default":"120",
                        "description":"Время неактивности, перед возвратом из меню about.json (в
секундах)",
                        "type":"integer"
                },
                "TimeoutButtonsMenu":{
                        "default":"10",
                        "description":"Время неактивности, перед возвратом из меню buttons_N.json (в
секундах)",
                        "type":"integer"
                },
                "TimeoutKeyboardMenu":{
                        "default":"20",
                        "description":"Время неактивности, перед возвратом из меню набора номера,
keyboard.json (в секундах)",
                        "type":"integer"
                },
                "TimeoutShowPicture":{
                        "default":"20",
                        "description":"Время показа картинки show_picture (в секундах)",
                        "type":"integer"
                }
        },
        "title":"GUI",
        "type":"object"
}
```

JSON схема секции HandsetState


```
{
        "description":"Настройки датчика аналоговой трубки",
        "properties":{
                "event_interval":{
                        "default":"0",
                        "maximum":"60",
                        "minimum":"0",
                        "readOnly":true,
                        "type":"integer"
                },
                "handset_button_pressed":{
                        "description":"Порог перехода в состояние 'Трубка снята и нажата кнопка'",
                        "format":"table",
                        "items":{
                                "default":"21",
                                "maximum":"63",
                                "minimum":"0",
                                "type":"number"
                        },
                        "type":"array"
                },
                "handset_offhook":{
                        "description":"Порог перехода в состояние 'Трубка снята'",
                        "format":"table",
                        "items":{
                                "default":"21",
                                "maximum":"63",
                                "minimum":"0",
                                "type":"number"
                        },
                        "type":"array"
                },
                "handset_onhook":{
                        "description":"Порог перехода в состояние 'Трубка висит'",
                        "format":"table",
                        "items":{
                                "default":"21",
                                "maximum":"63",
                                "minimum":"0",
                                "type":"number"
                        },
                        "type":"array"
                },
                "input_device_name":{
                        "default":"sunxi-lradc",
                        "readOnly":true,
                        "type":"string"
                }
        },
        "title":"HandsetState",
        "type":"object"
}
```

JSON схема секции HTTPRequester

```
{
        "properties":{
                "max_file_size":{
                        "default":"5000000",
                        "minimum":"0",
                        "type":"integer"
                },
                "media_dir":{
                        "default":"/var/media",
                        "type":"string"
                },
                "min_disk_space":{
                        "default":"50000000",
                        "minimum":"0",
                        "type":"integer"
                },
                "set_user":{
                        "default":"nobody",
                        "type":"string"
                }
        },
        "title":"HTTPRequester",
        "type":"object"
}
```

JSON схема секции LockManager

```
{
        "description":"Параметры замков",
        "properties":{
                "locks":{
                        "description":"Список замков",
                        "format":"table",
                        "items":{
                                "properties":{
                                        "default_state":{
                                                "default":"locked",
                                                "description":"Обычное состояние",
                                                "enum":[
                                                        "locked",
                                                        "unlocked"
                                                ],
                                                "type":"string"
                                        },
                                        "dtmf":{
                                                "default":"#",
                                                "description":"DTMF последовательность для отпирания
данного замка по SIP",
                                                "type":"string"
                                        },
                                        "gpio_file":{
                                                "default":"/sys/class/gpio/gpio108/value",
                                                "readOnly":"true",
                                                "type":"string"
                                        },
                                        "name":{
                                                "default":"front_door",
                                                "description":"Идентификатор замка",
                                                "type":"string"
                                        },
                                        "polarity":{
```

```
                                                "default":"normal",
                                                "description":"Полярность электромагнита",
                                                "enum":[
                                                        "normal",
                                                        "inversed"
                                                ],
                                                "type":"string"
                                        },
                                        "timeout":{
                                                "default":"5",
                                                "description":"Время (сек) удержания ЭМ при
отпирании",
                                                "maximum":"60",
                                                "minimum":"0",
                                                "type":"integer"
                                        },
                                        "unlock_sound_url":{

"default":"file:///home/fabmicro/SIPHomePhone/sounds/default_unlocking.wav",
                                                "description":"https:// адрес звукового файла
проигрываемого при отпирании",
                                                "type":"string"
                                        },
                                        "unlock_sound_volume":{
                                                "default":"100",
                                                "description":"Громкость проигрывания звукового
файла",
                                                "type":"integer"
                                        }
                                },
                                "type":"object"
                        },
                        "type":"array",
                        "uniqueItems":true
                },
                "primary_lock":{
                        "description":"Идентификатор 'главного' замка для данной Панели",
                        "type":"string"
                }
        },
        "title":"LockManager",
        "type":"object"
}
```


JSON схема секции Notify


```
{
        "description":"Оповещения (нотификации) на сервер о важных событиях",
        "properties":{
                "notify_ambient":{
                        "default":true,
                        "description":"Оповещать о состоянии внешнего освещения",
                        "type":"boolean"
                },
                "notify_calling":{
                        "default":true,
                        "description":"Оповещать об исходящих звонках с Панели",
                        "type":"boolean"
                },
                "notify_dialing":{
                        "default":true,
                        "description":"Оповещать о нахождении человека перед Панелью",
                        "type":"boolean"
```

```
                },
                "notify_doors":{
                        "default":true,
                        "description":"Оповещать об открывании дверей",
                        "type":"boolean"
                },
                "notify_interval":{
                        "default":"5",
                        "description":"Минимальный интервал между оповещениями (сек)",
                        "type":"integer"
                },
                "notify_locks":{
                        "default":true,
                        "description":"Оповещать об отпирании замков",
                        "type":"boolean"
                },
                "notify_url":{
                        "default":"",
                        "description":"https:// адрес сервера для отправки оповещений",
                        "type":"string"
                },
                "surveillance_camera_format":{
                        "default":"1920x1080",
                        "description":"Размер кадра JPEG картинки",
                        "type":"string"
                },
                "surveillance_camera_name":{
                        "default":"Internal",
                        "description":"Идентификатор камеры для JPEG картинки",
                        "type":"string"
                }
        },
        "title":"Notify",
        "type":"object"
}
```

JSON схема секции ProximitySensor

```
{
        "description":"Настройки датчика присутствия",
        "properties":{
                "approaching":{
                        "description":"Состояние 'приближение'",
                        "format":"table",
                        "items":{
                                "default":"2000",
                                "maximum":"8192",
                                "minimum":"0",
                                "type":"integer"
                        },
                        "type":"array"
                },
                "dialing":{
                        "desciption":"Состояние 'набор номера'",
                        "format":"table",
                        "items":{
                                "default":"700",
                                "maximum":"8192",
                                "minimum":"0",
                                "type":"integer"
                        },
```

```
                        "type":"array"
                },
                "event_interval":{
                        "default":"1000000",
                        "maximum":"10000000",
                        "minimum":"250000",
                        "readOnly":true,
                        "type":"integer"
                },
                "fingering":{
                        "description":"Состояние 'Вызов SOS'",
                        "format":"table",
                        "items":{
                                "default":"21",
                                "maximum":"8192",
                                "minimum":"0",
                                "type":"integer"
                        },
                        "type":"array"
                },
                "hwmon_device_name":{
                        "default":"vl53l0x",
                        "readOnly":"true",
                        "type":"string"
                },
                "valid_range":{
                        "description":"Диапазон рабочих значений",
                        "format":"table",
                        "items":{
                                "default":"21",
                                "maximum":"8192",
                                "minimum":"0",
                                "type":"integer"
                        },
                        "type":"array"
                }
        },
        "title":"ProximitySensor",
        "type":"object"
}
```

JSON схема секции RFIDReader

```
{
        "properties":{
                "baudrate":{
                        "default":"115200",
                        "enum":[
                                "9600",
                                "19200",
                                "38400",
                                "57600",
                                "115200",
                                "250000"
                        ],
                        "type":"integer"
                },
                "gpio_enable_file":{
                        "default":"/sys/class/gpio/gpio2_pc4/value",
                        "readOnly":"true",
                        "type":"string"
                },
                "gpio_enable_value":{
```

```
                        "default":"0",
                        "maximum":"1",
                        "minimum":"0",
                        "type":"integer"
                },
                "serial":{
                        "default":"/dev/ttyS1",
                        "readOnly":"true",
                        "type":"string"
                }
        },
        "title":"RFIDReader",
        "type":"object"
}
```

JSON схема секции RS485Master

```
{
        "properties":{
                "baudrate":{
                        "default":"115200",
                        "enum":[
                                "9600",
                                "19200",
                                "38400",
                                "57600",
                                "115200",
                                "250000"
                        ],
                        "type":"integer"
                },
                "serial":{
                        "default":"/dev/ttyUSB1",
                        "readOnly":"true",
                        "type":"string"
                }
        },
        "title":"RS485Master",
        "type":"object"
}
```

JSON схема секции Yard

```
{
        "description":"Настройки вызывной панели для калитки двора",
        "properties":{
                "Buildings":{
                        "description":"Перечень зданий для вызывной панели на калитке",
                        "format":"table",
                        "items":{
                                "properties":{
                                        "address":{
                                                "title":"полный адрес здания",
                                                "type":"string"
                                        },
                                        "aptnum_prefix":{
                                                "title":"префикс подставляемый перед номером
квартиры (желательно оканчиваться на #)",
                                                "type":"string"
                                        },
                                        "max_aptnum":{
                                                "default":"999",
                                                "title":"max квартира",
                                                "type":"integer"
                                        },
                                        "min_aptnum":{
                                                "default":"1",
                                                "title":"min квартира",
                                                "type":"integer"
                                        },
                                        "name":{
                                                "enum":[
                                                        "building1",
                                                        "building2",
                                                        "building3",
                                                        "building4",
                                                        "building5",
                                                        "building6",
                                                        "building7",
                                                        "building8",
                                                        "building9",
                                                        "building0"
                                                ],
                                                "title":"имя здания заданное в yard.json",
                                                "type":"string"
                                        },
                                        "url":{
                                                "title":"url на картинку одного из зданий. Размер
картинки должен совпадать с фоновой. Здание на которое можно нажать должно быть единственным
непрозрачным",
                                                "type":"string"
                                        }
                                },
                                "type":"object"
                        },
                        "type":"array"
                },
                "allbuildings_url":{
                        "description":"url на картинку фона, на калитке",
                        "type":"string"
                }
        },
        "title":"Yard",
        "type":"object"
}
```