Geekbrains

Дипломная работа

по специальности:

«Аналитик больших данных»

На тему:

«Автоматизация процессов умного дома на базе ОС Home Assistant»

Выполнил:

Студент факультета

«Аналитики больших данных»

Группы 3330

Аникин Алексей Борисович

Саратов - 2023

Содержание

Введение

Что такое умный дом, интернет вещей и почему они начали появляться.

«Официальной» датой рождения современной интеллектуальной системы автоматизации «Умный дом» (Smart Home) считается 1978 год, когда впервые удалось реализовать идею управления различными приборами и датчиками через электропроводку дома. Инженерам и конструкторам того времени удалось «научить» дома зажигать свет по хлопку, самостоятельно открывать двери перед хозяевами и гостями и т.д. Это удивляло и, нередко, шокировало людей, но, вместе с тем, заинтересовывало, призывая не останавливаться на достигнутом.

Сегодня концепция умного дома — это совокупность технологий, позволяющая создавать различные системы автоматизации жилого пространства, обеспечивающие возможность взаимодействия различных устройств, удаленного управления ими, а также, дружественный графический интерфейс для максимально простого взаимодействия с пользователем. При этом вся система может быть замкнута внутри пространства, и/или взаимодействовать с внешним миром через Internet.

Более широкому и плотному взаимодействию не только через хаб, но и друг с другом обеспечивает концепция «Интернета вещей» (“Internet of Things” - IoT). Интернет вещей (англ. *internet of things, IoT*) — концепция сети передачи данных между физическими объектами (*«вещами»*), оснащёнными встроенными средствами и технологиями для взаимодействия друг с другом или с внешней средой. Предполагается, что организация таких сетей способна перестроить экономические и общественные процессы, исключить из части действий и операций необходимость участия человека.

Концепция сформулирована в 1999 году как осмысление перспектив широкого применения средств радиочастотной идентификации для взаимодействия физических предметов между собой и с внешним окружением. Наполнение концепции многообразным технологическим содержанием и внедрение практических решений для её реализации начиная с 2010-х годов считается устойчивой тенденцией в информационных технологиях, прежде всего, благодаря повсеместному распространению беспроводных сетей, появлению облачных вычислений, развитию технологий межмашинного взаимодействия, началу активного перехода на IPv6 и освоению программно-определяемых сетей. (Ashton, 2014)

Ежегодно усложняемые процессы и нарастающая потребность в их взаимодействии между собой требует не просто более сложных систем, а тех систем, которые будут отвечать при этом параметрам надежности, независимости, стабильности работы и удобству использования. В теплице система автоматического полива должна знать влажность почвы и температуру воздуха, чтобы поливать экономно, но при этом достаточно. Конвейерная линия на производстве должна точно определять местоположение каждой детали и понимать, загружена ли линия и каждая ли деталь снята с конвейера. Отопление загородного дома должно поддерживаться автоматически, в зависимости от температуры в каждой комнате, при этом избегая лишнего перегрева и экономии ресурсов. Во всех системах один процесс имеет информацию о других процессах, тем самым обеспечивая непрерывный контроль и управление.

Человек всегда стремится жить в комфортных условиях, избегать лишней рутины в виде мытья посуды, стирки, готовки, уборки и так далее, освобождая таким образом время для других занятий. Умный дом не только берет на себя часть обыденных действий, то как: включить пылесос, когда никого нет дома, закрыть вечером шторы, передать показания счетчиков в обслуживающую организацию, но и выполняет сценарии по одному только слову, а в будущем будет заказывать еду, которой не хватает в холодильнике и предугадывать действия и пожелания владельца.

Целью работы является создание и объединение на локальном сервере управления компонентами умного дома с частичной автоматизацией процессов. При этом необходимо обеспечить автоматический запуск ПО и непрерывную работу сервера и приложений.

Проект будет решать три основные задачи:

- создание комфортной, легко управляемой среды в помещении;

- обеспечение непрерывного контроля процессов и автоматики без вмешательства человека;

- интегрирование и объединение максимального количества устройств на одной платформе управления умным домом.

Проект позволяет повысить комфортность проживания в условиях увеличивающегося количества интернета вещей и снизить количество рутинных действий. С учетом большого количества разнообразных устройств, которые при этом используют разные протоколы обмена (Wi-Fi, Bluetooth, Z-wave, Zigbee), и часто - собственное приложение, проект позволит использовать одно приложение для управления всеми устройствами.

Для выполнения проекта будет необходимо:

1. Выбрать компоненты для устройства сервера умного дома (УД);

2. Выбрать устройства для подключения к УД;

3. Изучить документацию по установке и настройке устройств;

4. Изучить документацию по установке и настройке ОС Home Assistant;

5. Изучить документацию YAML для доработки конфигурации и создания автоматизаций;

6. Изучить сторонние интеграции для расширения возможностей УД;

7. Протестировать работу УД;

В проекте будут использоваться следующие инструменты:

Git, Python, VirtualBox, YAML, Node-RED, PuTTY, SQL, InfluxDB. Telegram.

Используемые технологии: TCP/IP, Wi-Fi, Telegram API, SSH, RSTP, RS-232, DLNA, FTP.

Состав команды: Аникин А.Б.

Выступал в роли: архитектора системы умного дома, администратора сервера, программист, инженера АСУ.

Глава 1. Выбор компонентов для сервера умного дома (УД)

1.1 Критерии выбора оборудования

Сервером называется компьютер, выделенный из группы персональных компьютеров (или рабочих станций) для выполнения какой-либо сервисной задачи без непосредственного участия человека. Сервер и рабочая станция могут иметь одинаковую аппаратную конфигурацию, так как различаются лишь по участию в своей работе человека за консолью. Специализация серверного оборудования идёт несколькими путями, выбор того, в каком направлении идти, каждый производитель определяет для себя сам. Большинство специализаций удорожают оборудование.

Серверное оборудование зачастую предназначено для обеспечения работы сервисов в режиме 24/7, поэтому часто комплектуется дублирующими элементами, позволяющими обеспечить «пять девяток» (99,999 %; время недоступности сервера или простой системы составляет менее 6 минут в год). Для этого конструкторами при создании серверов создаются специальные решения (Comer & Stevens, 2023).

Сервер должен объединять в себе следующие типы: веб-сервер (для дальнейшего запуска сайта www.smartsar.ru на сервере), файловый сервер (для создания общей папки с медиа-файлами по стандарту DLNA, а так же FTP-сервера для хранения видео с камеры видеонаблюдения); сервер виртуализации (для запуска программ и виртуальной машины под Home Assistant).

Требования к характеристикам сервера: Небольшие габариты (неттоп), ПО Windows 10, DDR не менее 2Гб (с расширением до 8Гб), HDD 120Гб.

Решаемые задачи: запуск виртуальной машины с Home Assistant, развертывание сайта, работа telegram-бота, создание DLNA файлообменника, хранение видеозаписей с камеры видеонаблюдения на FTP-сервере, SQL-сервер.

Для этих целей выбран неттоп «Intel NUC5CPYH» (Рисунок 1), соответствующий всем требованиям. ОС – Windows 10, DDR увеличено до 8Гб. К серверу настроен удаленный доступ для работы с основного ПК.

Рисунок 1. Intel NUC5CPYH

1.2 Обеспечение бесперебойным питанием 220В и сетью WAN

Так как сервер умного дома должен отвечать критерию бесперебойности работы сети 220В и сети Internet, был приобретён источник бесперебойного питания «CyberPower EXP650E» с 6 розетками. В нем имеется 3 розетки с резервным питанием, 3 без резервного питания. ИБП имеет разъемы RJ-11, RJ-45, порт USB для подключения к серверу, программное обеспечение «PowerPanel» для мониторинга состояний, в том числе через облачное решение. Продолжительность под нагрузкой (роутер + неттоп, около 100Вт) более 30 минут (Рисунок 2).

Основное подключение Internet осуществляется через роутер «Keenetic Speedster», подключенного к ИБП, обеспечивающий 4 выхода LAN, в том числе в коммутатор «TP-Link LS1008G» на 8 разъемов RJ-45 для увеличения точек подключения.

Рисунок 2. ИБП CyberPower EXP650E

1.3 Бесшовный Wi-Fi

Некоторые устройства находятся в отдалении от основного роутера, и при слабом сигнале теряют связь. Для обеспечения более стабильной работы устройств и распределения нагрузки на роутеры приобретен дополнительный роутер «Keenetic Air». Роутер настроен как точка доступа, обеспечивает бесшовный Wi-Fi в одной домашней сети. [1] Управление осуществляется через приложение Keenetic (обеспечивающее, в том числе, облачное решение).

Сервер располагается в комнате, рядом с основным роутером, подключен к ИБП. Установлены дополнительные программы:

- PowerPanel Personal (для мониторинга состояния и управления ИБП);
- Python 3.10 (для работы Home Assistant, Telegram-bot);
- ONVIF Device Manager (для подключения видеонаблюдения);
- Oracle VM Virtual Box (для запуска Home Assistant на виртуальной машине);
- FileZilla Client (для создания FTP-сервера);

Создана папка автозагрузки, куда будет помещен ярлык виртуальной машины Home Assistant и другие программы. Установка и настройка Home Assistant будут рассмотрены далее.

Таким образом, выбран, настроен и запущен в работу сервер для Умного дома. Обеспечено бесперебойное питание сервера и роутеров. Настроена Mesh-система бесшовного Wi-Fi.

[1] Настройки бесшовного роуминга https://help.keenetic.com/hc/ru/articles/360000862539-Бесшовный-роуминг-Wi-Fi

Глава 2. Выбор компонентов и устройств УД

2.1 Протоколы связи и ПО умного дома и IoT

На сегодняшний день рынок умного дома развивается быстрыми темпами. Почти любое устройство снабжают Wi-Fi или Bluetooth модулями для подключения к телефону (и управления с помощью приложений).

В настоящий момент используется несколько основных протоколов для обмена информацией с устройствами IoT - Wi-Fi, Bluetooth, Z-wave, Zigbee. По мере развития и усовершенствования разработчики создают новые протоколы связи для нескольких целей – снизить энергопотребление устройств, повысить стабильность работы, перейти на более свободные частоты (вместо 2,4ГГц). При этом устройства, работающие по Wi-Fi, подключаются напрямую к роутеру, а остальные протоколы требуют так называемый «хаб»[2], с которым устройства обмениваются информацией, а сам хаб подключается к роутеру по Wi-Fi.

Многие производители электроприборов создают собственные приложения (Philips, LG, Redmond и др.). При этом некоторые интеграторы умного дома предоставляют пользователям собственное приложение, что может повысить локализацию устройств, но при этом снизить масштабируемость и проекта, и усложнить настройку. И конечно, такие интеграции отражаются на стоимости проекта, сильно удорожая его, так как предлагается некоторое «уникальное решение».

Основными игроками на так называемом рынке «DIY»[3] сейчас выступают платформы «eWelink» (https://ewelink.cc/), «Tuya» (https://www.tuya.com/), Aqara, Xiaomi со своим программным обеспечением («прошивками») и приложениями. Они позволяют практически любое устройство сделать «умным», т.е. подключиться к ПК и телефону, и получить доступ к приложению. Все они имеют облачное решение, сервера в основном располагаются в Китае. Управлять можно как в локальной сети, так и через внешнюю сеть Internet.

Вместе с этим развивается и голосовое управление умным домом. Такие возможности предоставляет «Google Home» (с Google Assistant, от Google), «Alexa» (от Amazon), «Siri» (от Apple), «Алиса» (от Яндекс) и другие. Производители платформ стараются делать интеграции с голосовыми помощниками, что делает их более привлекательными среди конкурентов.

---

[2] Менеджеры умного дома или межплатформенные хабы - это форма управления IoT-устройствами, ориентированная на потребителя, предоставляющая единую платформу для мониторинга и управления потребительскими устройствами, включая термостаты, освещение, системы безопасности и бытовую технику.

[3] DIY – аббревиатура от do it yourself, т.е. самодельное изготовление.

В проекте предполагается использовать устройства с протоколом связи по Wi-Fi, основные платформы «eWelink» и «Tuya», и Google Home для голосового управления.

2.2 Критерии выбора компонентов и устройств для УД

Умный дом проектировался с «нуля», на этапе дизайна квартиры. Такой подход позволил определить возможности и потребности в устройствах, детально обозначить их расположение на проекте, описать критерии и требования к компонентам умного дома.

Система умного дома должна была обеспечивать следующие функции:

- управление светом;
- управление приточной вентиляцией и заслонками (220В местная панель + Wi-Fi);
- управление шаровыми кранами для перекрытия трубопроводов (12В);
- включение/выключение водонагревателя (220В);
- управление шторами (220В, местная панель + Wi-Fi);
- управление теплым полом (220В, местная панель + Wi-Fi);
- управление кондиционерами (220В+ Wi-Fi);
- управление медиаустройствами - ТВ, ресивер (220В + LAN);
- управление робот-пылесосом (Wi-Fi).
- управление климатом (220В + Wi-Fi);
- управление сигнализациями (12В + Wi-Fi);
- управления устройствами по инфракрасному IR сигналу.
- мониторинг ресурсов водоснабжения и электроэнергии (12В + Wi-Fi).

Требования к оборудованию:

- подключение по кабелю RJ-45 или Wi-Fi;
- иметь собственное приложение для управления;
- интеграция с Google Home для голосового управления;
- возможность локального управления внутри домашней сети;
- невысокая стоимость;
- возможность самостоятельной настройки.

Первоначально была задача найти основное устройство, обладающее собственными реле, приложением, креплением на DIN-рейку в распределительный щиток, подключением по кабелю UTP. Выбор был сделан в пользу китайского производителя программируемых устройств KinCony Electronics Co., Ltd (https://www.kincony.com/). Контроллер KC868-H32

(Рисунок 3) представляет собой одноплатый компьютер на базе микропроцессоров ESP32[4], со встроенными 12В реле (на 5А) в количестве 32 шт. с LED-индикаторами включения, портом LAN, разъемом RS-232, приемником RF-433, аналоговыми выходами 6 шт. (для подключения датчиков и сигнализаций). Блок имеет крепление на DIN-рейку, входное питание 12В. Вместе с блоком управления приобретается плата кнопочных включателей для локального ручного управления реле, и плата подключения клавишных выключателей (Рисунок 4) - для управления реле из других мест – в данном случае светом и водогрейкой. Производитель разработал, так же, собственное приложение «Smart Home» для управления контроллером, и обеспечил возможность интеграции устройства к голосовым помощникам и платформам умного дома (в частности, к Google Home для Android устройств, HomeKit для IOS).

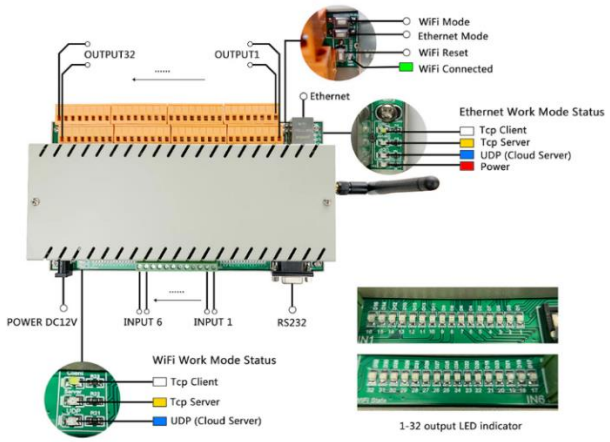

Рисунок 3. KC868-H32B (дополнительно оснащен модулем Wi-Fi).

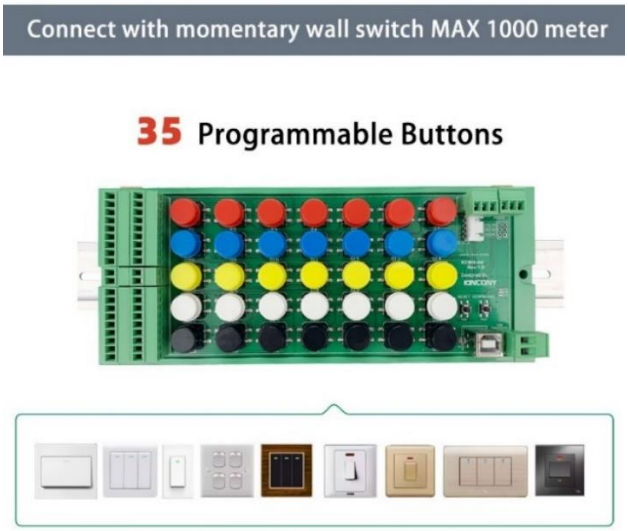

Рисунок 4. Плата с ручным управлением

[4] **ESP32** — серия недорогих микропроцессоров с малым энергопотреблением китайской компании Espressif Systems.

2.3 Задачи управления, контроля и автоматизации

Подключение к контроллеру KC868-H32 и управление реле.

Управление светом. Включение света обеспечивается возвратными одноклавишными выключателями. Подключены к плате управления выключателями, лампы подключены к KC868-H32, реле №1-5.

Управление водонагревателями. К KC868-H32, реле №6 подключен клавишный выключатель, обеспечивающий коммутацию большого промежуточного реле на 40А, которое включает два водонагревателя – проточный и накопительный.

Управление кранами ГВС и ХВС. При включении реле №6 происходит срабатывание (закрытие входных и открытие промежуточных) кранов шаровых горячего и холодного водоснабжения (Рисунок 5). Реле №15 включается автоматически через 2 секунды после реле №6, и выключается через 8 секунд. Это необходимо для кратковременной подачи напряжения на краны при переводе их в положения «открыто-закрыто». Процесс автоматизации описан в главе 4.

Рисунок 5. Кран шаровый с приводом 12В.

Управление заслонками приточной вентиляции. Для управления воздушными клапанами были приобретены ¼ оборотные приводы 220В (Рисунок 6) и подключены к KC868-H32, реле №7-10.

Рисунок 6. Привод для клапана воздушной заслонки.

Реле №11, 16 (KC868-H32) оставлены, как запасные.

Управление датчиками движения, включающими ночную подсветку. К реле №12-14 (KC868-H32) подключены датчики движения в трех комнатах.

Перезагрузка IP-камеры. Осуществляется кратковременным автоматическим включение реле №17. Процесс необходим для перезагрузки камеры при зависании. Процесс автоматизации описан в главе 4.

Устройства, работающие по Wi-Fi.

Управление приточной вентиляцией осуществляется настенной панелью (Рисунок 7). Имеет питание 220В, Wi-Fi, 3 режима вентилятора, 3 режима климата (охлаждение, вентиляций, обогрев. Работает с приложением «MySmartThermostat» или «Tuya». Автоматизация описана в главе 4.

Рисунок 7. Настенная панель приточной вентиляции

Управление теплыми полами осуществляется настенной панелью (Рисунок 8). Имеет питание 220В, Wi-Fi, датчик температуры пола, датчик температуры. Работает с приложением «MySmartThermostat» или «Tuya». Имеет недельное расписание.

Рисунок 8. Настенная панель для теплого пола

Управление шторами осуществляется настенной панелью (Рисунок 9). Имеет питание 220В, Wi-Fi. Работает с приложением «eWelink». Имеет недельное расписание работы.

Рисунок 9. Панель управления шторами

Датчики открытия двери и окон (Рисунок 10) информируют об открытии-закрытии. Имеет питание 3В (работает на батарейках или от блока питания), Wi-Fi. Работает с приложением «eWelink». Датчики необходимы для контроля открытия двери (как элемент охранной системы), и для контроля открытия окон, чтобы избежать работы приточной вентиляции при открытых окнах. При закрытии окон будет включаться вентиляция. Автоматизация описана в главе 4.

Рисунок 10. Датчик открытия.

Умная розетка на 20 Ампер (Рисунок 11). Имеет питание 220В, Wi-Fi. Работает с приложением «eWelink» или «Tuya». Удобно для непосредственного включения устройств и приборов. К одной из розеток подключен 3D принтер для удаленной работы.

Рисунок 11. Умная розетка.

Датчик температуры со встроенным реле на 16 Ампер (Рисунок 12). Имеет питание 220В, Wi-Fi, проводной датчик температуры. Работает с приложением «eWelink». Сам датчик подключен к общедомовому стояку горячего водоснабжения (ГВС). В зависимости от температуры ГВС включает или выключает водонагреватели. Настроены личные и групповые оповещения. Автоматизация описана в главе 4.

Рисунок 12. Датчик температуры со встроенным реле.

Робот-пылесос фирмы ABIR (Рисунок 13). Управление по Wi-Fi, имеет собственное приложение «WeBack», имеет режим влажной уборки, возможность строить карту местности, убираться в заданных местах, виртуальные границы. Возможно настроить автоматизацию уборки, когда никого нет дома. При этом необходимо следить за отсутствием мелких вещей на полу.

Рисунок 13. Робот-пылесос.

В общем коридоре установлена IP-камера (Рисунок 14). Подключение по LAN или Wi-Fi, питание 12В. Имеет собственное приложение, передает сигнал по протоколу RSTP. Подключается к ПО «Onvif Device Manager». Есть возможность записи по событиям и их сохранения на ftp-сервере.

Рисунок 14. IP-camera.

Телевизор LG, подключение по LAN или Wi-Fi. Управляется пультом и из собственного приложения «LG ThinQ».

Кондиционеры LG управляются из собственного приложения «LG ThinQ» по Wi-Fi. Могут управляться от внутренней температуры, погоды, времени и других условий. Автоматизации описаны в главе 4.

Ресивер DANON 1600. Подключение по LAN. Управляется пультом и из собственного приложения «AVR Remote». Доступно онлайн воспроизведение медиатеки из приложения «HEOS».

Для управления устройствами по инфракрасному сигналу IR (в качестве замены физических пультов) и радиосигналу (RF 433), применяется устройства типа «Broadlink» (Рисунок 15), в которое встроен передатчик Wi-Fi для подключения к роутеру, и передатчики IR и RF. Имеет собственное приложение «Broadlink».

Рисунок 15. Broadlink hub.

Показания счетчиков воды и электроэнергии передаются в блок SAURES R5 (Рисунок 15), (https://www.saures.ru/). Имеет питание 11-30В (и независимое от батареек), Wi-Fi, возможность подключения счетчиков электроэнергии по протоколу RS-485, импульсных счетчиков воды, датчиков протечек. Имеет собственное приложение «Saures», где собираются все данные. Можно настраивать оповещения, и автоматическую отправку показаний в ресурсоснабжающие организации (пока только в Москве). Имеет гибкие настройки обновления показаний (для экономии энергии). Позволяет устанавливать счетчики в труднодоступные места, а так же устанавливать непосредственный контроль ответственных служб за показаниями.

Рисунок 16. Контроллер Saures R5.

Таким образом, мы имеем огромное количество устройств, никак между собой не связанных, и более 10 приложений, с индивидуальными настройками. Такой подход усложняет взаимодействие, настройку и работу устройств. При этом нельзя построить логику работы одних устройств от других.

2.4 Современные решения в создании УД

Проблемой объединения множества устройств занимаются крупнейшие разработчики и энтузиасты (с open-source). Их можно разделить на производителей и интеграторов. Производители изготавливаются свои устройства, используя самые популярные платформы, а интеграторы могут не только производить собственные устройства, но главной их задачей является объединения всех устройств под одной платформой.

В таблице №1 показаны основные самые популярные платформы и производители, доступные в РФ.

Таблица №1. Производители и интеграторы умного дома.

Система	**Протокол подключения**	**Голосовые помощники**
Умный дом Яндекс	ZigBee, Wi-Fi	Алиса
Умный дом Sber	Wi-Fi, Bluetooth	Салют
Rubetek	Z-Wave, Wi-Fi, RF, Bluetooth	Алиса, Google Assistant
Polaris IQ Home	Wi-Fi	Алиса, Маруся
Livicom	Ethernet, Wi-Fi, 3G, радиоканал	Алиса, Маруся
Hiper Smart Home	Wi-Fi	Алиса, Маруся, Siri, Google Assistant, Салют
Умный дом Redmond	Wi-Fi, Bluetooth	Алиса, Маруся
Aqara	ZigBee, Wi-Fi	Алиса, Салют, Маруся
Xiaomi Smart Home	ZigBee, Wi-Fi, Bluetooth	Алиса, Google Ассистент, Alexa
Samsung SmartThings	ZigBee, Wi-Fi	Алиса
Philips Hue	ZigBee, Bluetooth	Алиса, Siri, Google Assistant
Google Home	как единый центр	Google Assistant
Node-RED	как единый центр	Алиса, Siri, Google Assistant, Alexa

Google Home позволяет объединять практически все устройства умного дома, работающие по любому протоколу и любого производителя, который поддерживает работы с Google Home. Управляется через приложение «Google Home».

Менее гибкими являются «Яндекс» и «Сбер», которые пытаются догнать и освоить этот рынок. Имеют собственные приложения, но сильно ограничены в интегрируемых устройствах.

Более интересными с точки зрения возможностей объединения, управления, проектирования и автоматизации являются системы с открытым исходным кодом (open-source). Самые популярные из них: Node-RED, openHAB, ioBroker, Domoticz, Home Assistant, MajorDomo.

У каждого из них есть свои сторонники и сообщество, дорабатывающие новые версии, набор оборудования, с которым система может работать сразу. Также, имеются некоторые особенности.

Node-RED представляет собой систему объединения устройств для их автоматизации. Запускается в основном на одноплатных компьютерах, в Docker, или на облачных серверах, благодаря низким требованиям к устройствам. Написана на языке Node.js, управляется через браузер. Является средством визуального программирования (low-code система), а ключевой составляющей выступает парадигма потокового программирования Потоковое программирование — это способ описания поведения приложения в виде сети черных ящиков или «узлов» («node»), как они называются в Node-RED. Каждый узел имеет четкую цель — к нему поступают некоторые данные, он что-то делает с этими данными, а затем передает их на следующий узел. Сеть отвечает за поток данных между узлами. Вся особенность заключается в том, что управление и автоматизации необходимо рисовать. Т.е. строить связи между «нодами». Очень гибкая и мощная система, но только для работы внутренних процессов, без графического интерфейса привычного умного дома. Поддерживает работу с голосовыми ассистентами, есть мало функциональное приложение для телефона.

Majordomo реализовал собственное голосовое управление системой, сделан на PHP, причем, язык PHP используется и для того, чтобы настраивать логику работы системы. Есть уже написанные интеграции к популярным контроллерам. И, конечно, MQTT[5]. Большое русскоязычное сообщество. Поддерживает большое количество сторонних устройств, имеет собственного голосового помощника (приходится в начале команд произносить «Majordomo»), собственное мобильное приложение. Получена информация об успешной интеграции MD с LightHub, Google Assistant.

[5] MQTT – это основанный на стандартах протокол, или набор правил, обмена сообщениями, используемый для взаимодействия между компьютерами. Интеллектуальные датчики, носимые устройства и другие устройства Интернета вещей (IoT) обычно передают и получают данные по сетям с ограниченными ресурсами и пропускной способностью. Эти устройства IoT используют MQTT для передачи данных, поскольку он прост в реализации и может эффективно передавать данные IoT. MQTT поддерживает передачу сообщений от устройств в облако и в обратном направлении.

OpenHab, одна из наиболее старых систем, также, имеет (не-русское) голосовое управление, кроме этого, мобильное приложение, интеграция с HomeKit (управление с устройств Apple без установки каких-либо приложений).

iOBroker. Система написана на языке Node.js. Выглядит достаточно продуманной и универсальной. Легко устанавливается. Имеет визуальный редактор планов помещений VIS, работает с MQTT на основе Node.js. Интегрируется в HomeKit. Имеет собственное мобильное приложение.

Home Assistant (*HA*) — программное обеспечение с открытым исходным кодом для домашней автоматизации, поддерживает устройства разных производителей, обеспечивает создание сложных сценариев автоматизации с возможностью использования голосовых помощников и визуализацией посредством веб-интерфейса, а также приложений для мобильных устройств.

Возможности:

- поддержка основных коммуникационных стандартов, включая Wi-Fi, BlueTooth, Z-Wave, ZigBee;
- решения множества компонентов домашней автоматизации различных производителей;
- организация охранной сигнализации и видеонаблюдения для домашней системы безопасности.

Платформа поддерживает несколько вариантов установки:

- HA Operating System (рекомендован разработчиком): устанавливает программу как операционную для работы в задачах домашней автоматизации, включает ядро, функцию Supervisor для управления ядром, набор предустановленных плагинов для интеграции устройств и обращения к сервисам;
- HA Core: ручная установка ядра с использованием виртуального окружения Python (подходит для опытных пользователей), даёт полный доступ к платформе, требует минимальных знаний об операционной системе и умения работать с окружением Python и командной строкой;
- HA Operating SystemHA Supervised: установка ядра платформы с функцией Supervisor в ручном режиме (подходит для опытных пользователей), предоставляет самый полный контроль над системой, но в этом случае работает только с Linux.
- HA Container: установка ядра в Docker-контейнерах.

По умолчанию для базы данных исторических данных используется SQLite. Интерфейс с базами данных реализован на SQLAlchemy, благодаря чему поддерживается достаточно

широкий набор реляционных СУБД, в частности, MySQL, MariaDB, PostgreSQL, Microsoft SQL Server.

Для конфигураций и настроек используется синтаксис YAML. Хотя большинство интеграций настраиваются через пользовательский интерфейс, для некоторых задач необходимо редактировать основной файл конфигурации (например, указать, в какую базу данных вести запись, или установить обратный прокси). (Schoutsen, https://ru.wikipedia.org/wiki/Home_Assistant, 2023)

Голосовое управление через Google Assistant, Алиса, Alexa и др.

ПО написано на языке Python, имеет непосредственную интеграцию с GitHub[6], и многими облачными системами. Система позволяет написать собственные интеграции для любых устройств, а также собственный интерфейс, сделав ваш умный дом практически уникальным.

Таким образом, дано определение для Умного дома и Интернета вещей, выбрано оборудование и устройства для УД. Все устройства добавлены в сеть Wi-Fi и настроены в своих приложениях. Среди всех современных решений с открытым кодом была выбрана система Home Assistant, как наиболее универсальная, с гибкими настройками и практически неограниченными возможностями управления.

[6] GitHub — крупнейший веб-сервис для хостинга IT-проектов и их совместной разработки. Веб-сервис основан на системе контроля версий Git и разработан на Ruby on Rails и Erlang компанией GitHub, Inc.

Глава 3. Установка и настройка ОС Home Assistant

3.1 Принципы автоматизации.

К основным принципам автоматизации можно отнести следующие: согласованности, интеграции, независимости исполнения и завершенности. Каждое действие в своей совокупности должно образовывать систему. Все они должны быть согласованными не только между собой, но и с входами и выходами процесса. Любое несоответствие может в итоге вылиться в нарушение или сбой, что естественным образом отразиться на результате. Автоматизируемый процесс обязан иметь возможности для интеграции в общую среду организации, так же необходимо гарантировать его взаимодействие с внешней средой. Кроме того, автоматизируемый процесс должен быть абсолютно самостоятельным и независимым, то есть контроль со стороны человека должен стремиться к нулю. (А. В. Брусницын, 2016)

Комплексна интеграция.

Принцип комплексной интеграции означает, что уровень автоматизации зависит от того, как процессы взаимодействуют друг с другом и окружающим миром. Еще одним критерием считается то, насколько быстро интегрируется отдельно взятая технология в общую систему.

Согласованность и гибкость

Принцип согласованности означает, что все операции, совершаемые компьютеризированной системой, должны быть согласованными между собой, а также с ее входами и выходами. Невыполнение этого принципа может привести к сбоям при выполнении работы.

Параметр гибкости систем производства означает, что любые элементы можно оптимизировать или быстро заменить. Такая методика современной автоматизации помогает эффективнее использовать имеющиеся ресурсы и перестраивать их под новые задачи без больших финансовых трат.

Независимое выполнение.

Принцип независимого выполнения заключается в том, что машина должна осуществлять все производственные процессы самостоятельно. Человеческое участие должно быть сведен к минимуму и заключаться лишь в наблюдении за работой системы.

Завершенность.

Правильно выстроенная система автоматизации — это завершенный, полноценный процесс, который не подразумевает перенаправления задач в сторонние подразделения. (ООО «Акрукс», 2020)

3.2 Концепция и терминология Home Assistant

Приоритетами Home Assistant являются локальное управление и конфиденциальность. А главными девелоперами системы являются энтузиасты и умельцы по всему миру.

Концептуально Home Assistant включает в себя:

- дашборы;
- интеграции;
- устройства и сущности;
- автоматизации;
- скрипты;
- сцены;
- дополнения.

Дашборд (dashboards) – адаптируемая страница для отображения связанной и доступной информации в HA.

Интеграции (integrations)– приложения, обеспечивающие возможность привязывать к HA другие приложения и платформы.

Устройства и сущности (devices & entities). Устройства - логически сгруппированные сущности. Представляет собой физическое устройство или сенсоры. Сущности представляют собой некоторые свойства или состояния устройства или объекта.

Автоматизации (automations) – набор повторяющихся действий, которые могут срабатывать автоматически. Состоят из трёх главных компонентов: триггеров (пускателей процесса), условий (срабатывания автоматизации), действий (выполняемых автоматизацией).

Скрипты (scripts) – аналог автоматизации, но без триггеров. Скрипты могут срабатывать в автоматизациях. Полезны при использовании одинаковых действий в разных автоматизациях.

Сцены (scenes) – сцены позволяют задавать предустановленные настройки для устройств. Сценой можно задавать ряд событий, под определенное условие (утренний сценарий, ночной, просмотр кино и т.д.).

Дополнения (add-ons) – это сторонние программы с простой установкой (напрямую или через GitHub), для обеспечения дополнительных функций и подключения устройств. (Концепция и терминология, 2016).

У Home Assistant есть официальный сайт (https://www.home-assistant.io/), документация для разработчиков (https://developers.home-assistant.io/) и сообщество (https://community.home-assistant.io/). Данный подход дает любому пользователю возможность практически неограниченной работы с устройствами.

3.3 Установка и запуск HA на виртуальной машине

HA подразумевает 4 основных установки:

1) Рекомендуемый: Home Assistant Operating System - минимальный набор инструментов с Супервизором (Supervisor) – для управления Ядром и Дополнениями;

2) Установка на виртуальную машину: Home Assistant Container – отдельная контейнерная установка (например, Docker, Virtual Box);

3) Альтернативный метод с Супервизором: Home Assistant Supervised – ручная установка Супервизора;

4) Альтернативный метод с Ядром: Home Assistant Core – ручная установка через виртуальное окружение Python.

Возможна установка на следующие основные платформы:

Home Assistant Yellow, Raspberry Pi, ODROID, ASUS Tinkerboard, Ceneric [86-64 (например Intel NUC), Windows, macOS, Linux. (Schoutsen, Установка (Installation), 2023)

В данном случае была выбрана установка HA на ОС Windows 10, путем установки виртуальной машины (ВМ) VirtualBox (.vdi). Далее в ВМ установка осуществляется на ОС Linux (64-bit). В ВМ записывается Home Assistant. При запуске ВМ загружается ОС Home Assistant (Рисунок 17), который будет доступен в браузере по ссылке «Home Assistant:8123» или http://X.X.X.X:8123 (где X.X.X.X – IP адрес устройства, который ему присваивается при создании ВМ). Ярлык запуска HA добавляется в папку «автоматического запуска» при запуске Windows. Это необходимо при сбоях или перезагрузке Windows, чтобы HA запускался автоматически. Такой вариант загрузки позволит сохранить рабочее пространство Windows и иметь доступ ко всему функционалу HA.

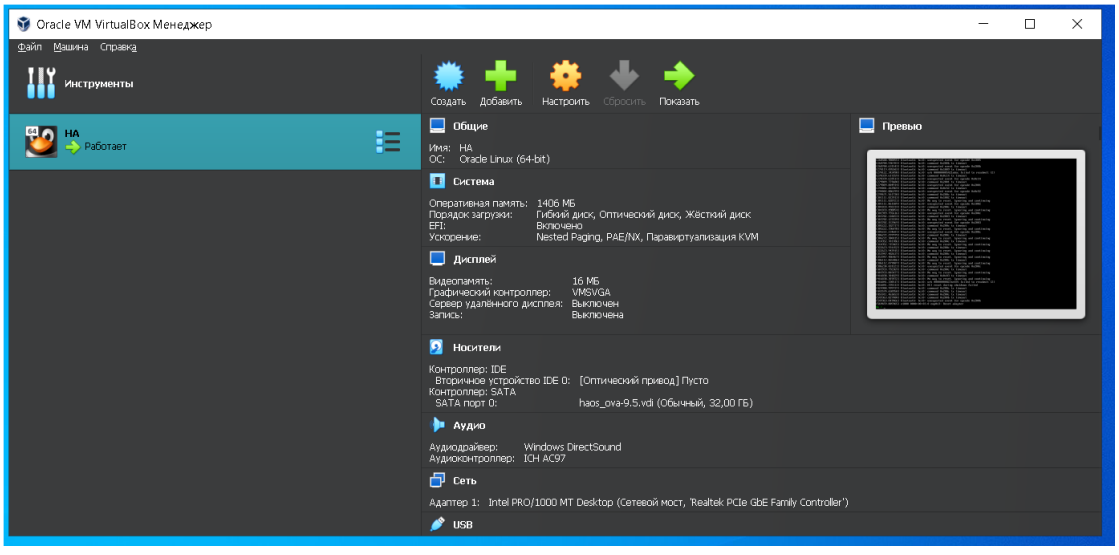

Рисунок 17. Запуск виртуальной машины с Home Assistant.

3.4 Дополнения и интеграции (Add-on and integrations)

Для удобства работы с HA и подключения устройств установим следующие дополнения (Add-ons): Настройки -> Дополнения -> Магазин дополнений (Рисунок 18):

- Terminal & SSH для удаленного управления HA;
- Node-RED для автоматизации и управления блоком Kinckony KC868;
- File editor для получения доступа к папкам и файлам, и кодированию;
- eWeLink Smart Home (официальное дополнение для управления устройствами eWelink);
- InfluxDB для создания дополнительной базы данных.

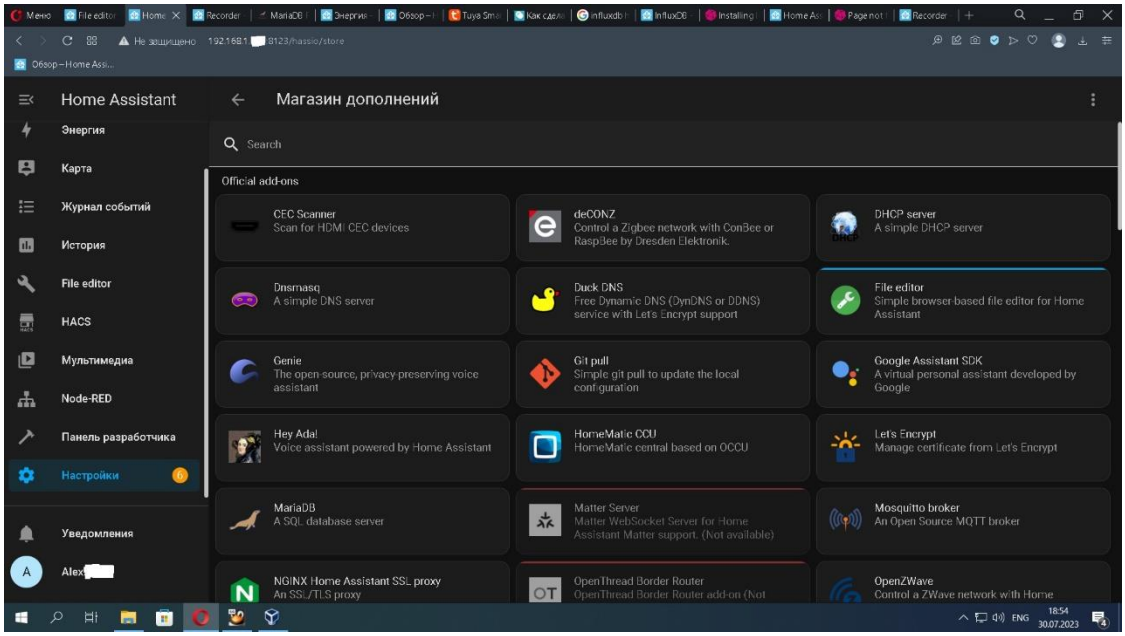

Рисунок 18. Магазин дополнений (add-ons).

Почти все Интеграции доступны в GitHub. Существуют уже готовые встроенные интеграции, которые можно найти через Настройки -> Интеграции и устройства. Для установки дополнительных сторонних интеграций (которые пишут разработчики и энтузиасты) в первую очередь требуется установить Интеграцию HACS (Home Assistant Community Store). Затем через неё устанавливаются сторонние интеграции путем добавления ссылки на репозиторий GitHub.

В нашем случае устанавливаются следующие Интеграции:

- HACS для установки сторонних интеграций;

- Uptime для отслеживания времени работы сервера;

- Local_ip для отображения локального IP-адреса сервера;

- Мобильное приложение для подключения мобильного устройства и работы через мобильное приложение Home Assistant;

- Sun для создания автоматизация по событиям Солнца;

- Android TV для подключения Android приставок и проектора;

- Broadlink для подключения устройств Broadlink;

- Denon AVR-X1600H для управления ресивером;

- Denon HEOS для доступа к медиа и облачным музыкальным сервисам;

- DLNA Digital Meadia Renderer для доступа к DLNA серверу;

- Google Cast для онлайн трансляции с музыкальных серисов;

- Keenetic NDMS2 Router для управления роутером и получения информации об устройствах;

- LG webOS Smart TV для управления телевизором;

- Meteorologisk institutt (Met.no) для доступа информации о погоде и автоматизации по погодным условиям (от норвежского института);

- Node-RED Companion для взаимодействия устройств HA с автоматизациями Node-RED;

- Radio Browser для доступа к онлайн радио по всему миру;

- SmartThinQ LGE Sensors для управления кондиционерами;

- SauresHA для доступа к блоку Saures и показаниям счетчиков;

- Sonoff LAN для управления устройствами с eWelink;

- Tyua Local для управления устройствами с Tuya;

- ONVIF для доступа к IP камере;

- Home Assistant Supervisor устанавливается по умолчанию и дает доступ к информации о сервере.

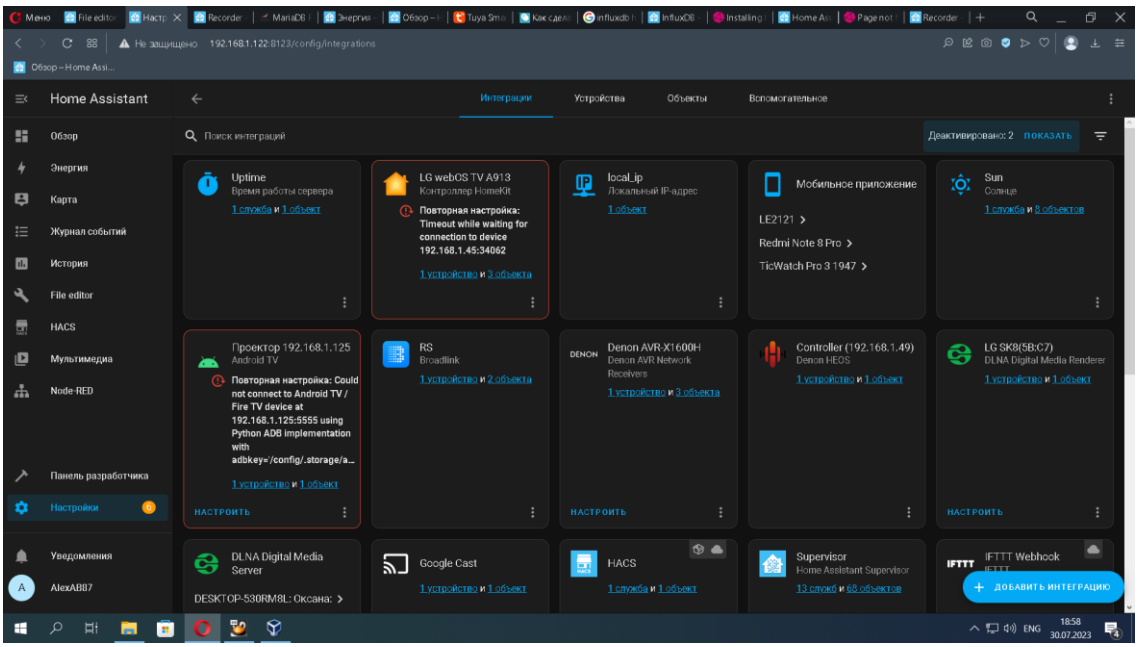

Рисунок 19. Установленные интеграции (integrations).

Часть дополнений и интеграций устанавливаются из «Магазина дополнений». Согласно официальной инструкции HACS – для её установки потребуется подключение к серверу HA (ОС Hassio) через протокол SSH[7]. Инструкцию можно найти на сайте https://hacs.xyz/docs/setup/download/. Для подключения из HA потребуется скачать дополнение «Termina & SSH». После подключения к серверу, установка происходит через командную строку путем ввода команды: «wget -0 – https://get.hacs.xyz | bash – » с дальнейшей разархивацией и инсталляцией.

Все сторонние интеграции находятся в разделе «config/custom_components/.

Пример кода из интеграции SauresHA по добавлению сенсоров (по сути – устройств). Код 1 написан автором интеграции, которая доступна по ссылке https://github.com/volshebniks/sauresha на языке Python. Следует сказать, что HA работает на версии Python 3.10.

Код 1. Пример кода из интеграции SauresHA.

```
"""Provides a sensor for Saures."""
import logging
from Home Assistant.const import CONF_SCAN_INTERVAL
from datetime import timedelta

from Home Assistant.core import Home Assistant
from .const import DOMAIN, COORDINATOR, CONF_ISDEBUG
from .api import SauresHA
from .entity import SauresControllerSensor, SauresSensor

SCAN_INTERVAL = timedelta(minutes=20)

_LOGGER = logging.getLogger(__name__)


async def async_setup_platform(hass, config, async_add_entities, discovery_info=None):
```

[7] SSH — сетевой протокол прикладного уровня, позволяющий производить удалённое управление операционной системой и туннелирование TCP-соединений.

```
        """Setup the sensor platform."""
        _LOGGER.exception(
            "The sauresha platform for the sensor integration does not support YAML platform setup. Please remove it from your config"
        )
        return True


    async def async_setup_entry(hass: Home Assistant, config_entry, async_add_entities):
        """Setup sensor platform."""
        my_sensors: list = []
        is_debug = CONF_ISDEBUG
        scan_interval = config_entry.data.get(CONF_SCAN_INTERVAL)
        controller: SauresHA = hass.data[DOMAIN].get(COORDINATOR)
        for curflat in controller.flats:
            try:
                controllers = await controller.async_get_controllers(curflat)
                for obj in controllers:
                    if len(obj.get("sn")) > 0:
                        my_controller = SauresControllerSensor(
                            hass,
                            controller,
                            curflat,
                            obj.get("sn"),
                            obj.get("name"),
                            is_debug,
                            scan_interval,
                        )
                        my_sensors.append(my_controller)

                        sensors = await controller.async_get_sensors(curflat)
```

```
                    for curSensor in sensors:
                        sensor = SauresSensor(
                            hass,
                            controller,
                            curflat,
                            curSensor.get("meter_id"),
                            curSensor.get("sn"),
                            curSensor.get("meter_name"),
                            is_debug,
                            scan_interval,
                        )
                        my_sensors.append(sensor)
            except Exception:
                _LOGGER.exception(str(Exception))

        if my_sensors:
            async_add_entities(my_sensors, True)
```

Таким образом дополнения и интеграции позволят нам добавлять устройства в Home Assistant и управлять ими. Устройства добавляются автоматически, после привязки облачного аккаунта или вручную через Настройки -> -Устройства и службы -> Вспомогательное. Почти все устройства имеют локальное подключение и управление в домашней сети (кроме LG Sensors - кондиционеров), а значит не привязаны к зарубежным серверам.

Прежде всего добавим в дополнение Node-RED реле блока Kincony KC868-H32. Дополнение будет запускать автоматически и работать непрерывно. Блок KC868-H32 имеет возможность обработки данных по протоколу TCP/IP[8]. Для этого переводим его на работу по этому протоколу. Теперь можно отправлять и получать сообщения на блок через приложение (например PuTTY[9]). В данном случае клиентом для отправки и принятия сообщений у нас будет выступать Node-RED, в котором есть узлы для отправки и получения сообщений (Рисунок 20). Подключение к узлу осуществляется по IP-адресу блока управления и порту «4196». Каждому реле присваивается виртуальный переключатель (toggle, с названием

[8] **TCP/IP** — сетевая модель передачи данных, представленных в цифровом виде. Модель описывает способ передачи данных от источника информации к получателю.

[9] PuTTY — свободно распространяемый клиент для различных протоколов удалённого доступа, включая SSH, Telnet, rlogin. Также имеется возможность работы через последовательный порт.

«input_boolean.relay_name») в Home Assistant. При изменении состояния переключателя создается сообщение типа «RELAY-SET-255,X,1», где «X» - номер реле, «1» - команда включения («0» - команда выключения). В зависимости от текущего состояния реле формируется нужный текст, который далее передается на узел принятия данных по протоколу TCP/IP. Для передачи текущего состояния реле (на блоке управления), при изменении его статуса, порт «прослушивается» и с блока управления приходит текст «RELAY-KEY-255,X,1,OK», которое сообщает, что реле «включено» и через узел «условий» передает информацию о состоянии выключателя в Home Assistant. Таким образом мы получаем актуальный статус переключателя непосредственно в Home Assistant.

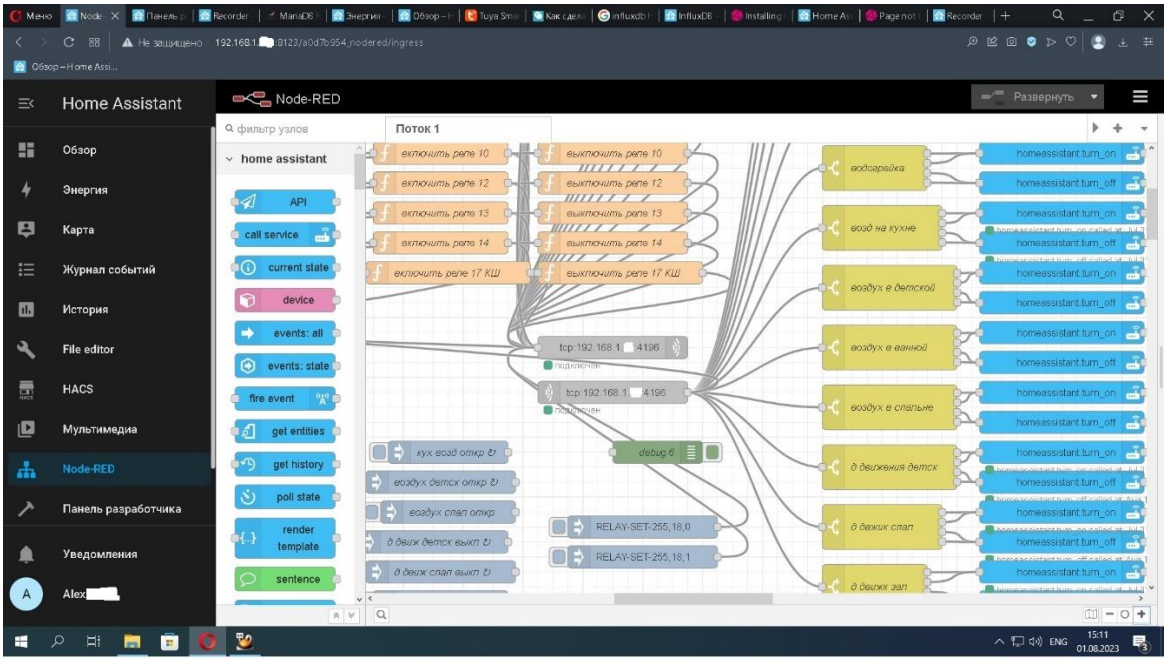

Рисунок 20. Создание связи с блоком управления Kincony по протоколу TCP/IP.

На Рисунке 21 показана вся связка потоков, в том числе с подключением голосового ассистента Google Assistant (который так же передает текстовые команды в узел TCP/IP).

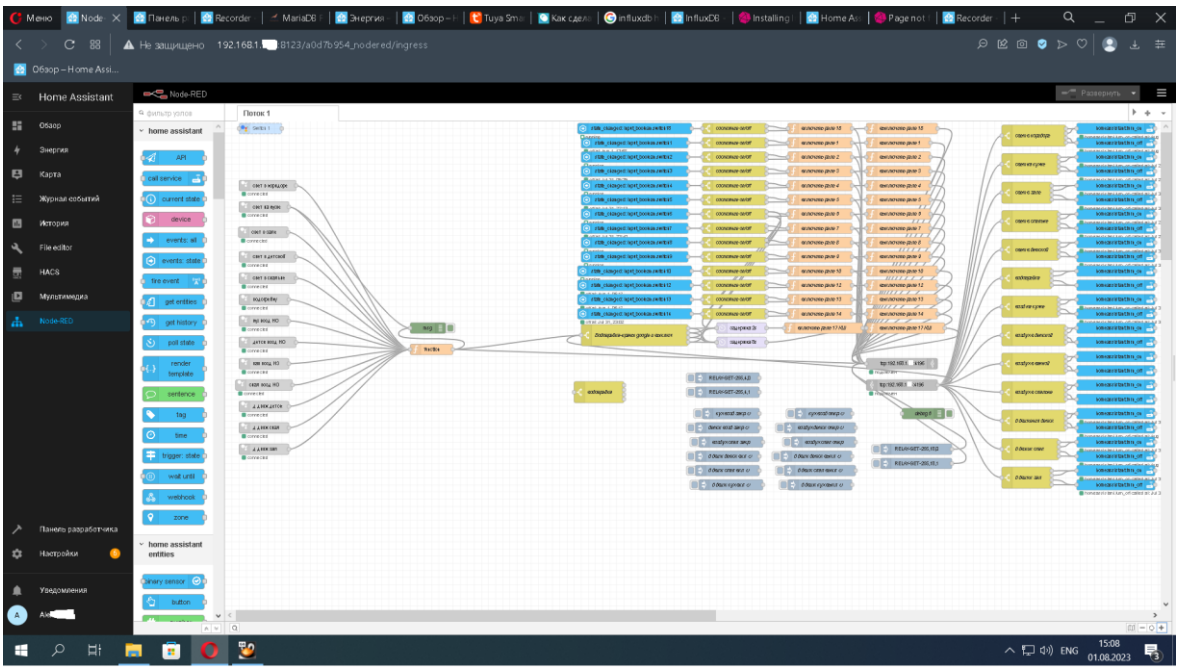

Рисунок 21. Узлы и потоки в Node-RED для управления блоком Kincony KC868-H32.

Все устройства, датчики и параметры можно вывести в так называемый «Обзор» (Рисунок 22), который можно разбить на несколько окон. В данном случае Обзор состоит из 3 окон – «Дом», «Мультимедиа» и «Климат» (Рисунок 23).

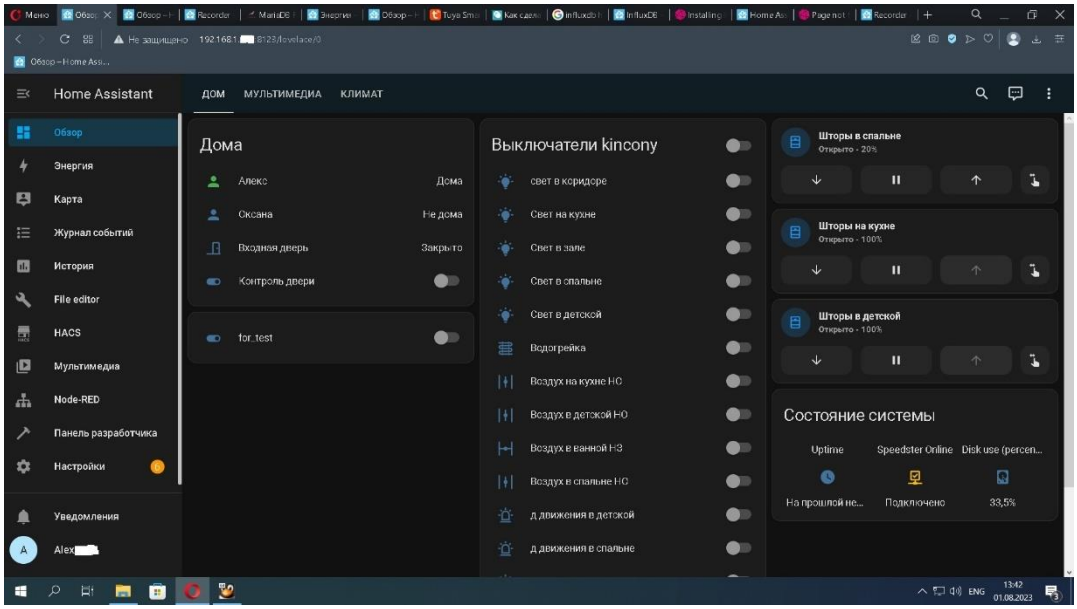

Рисунок 22. Обзор устройств

Рисунок 23. Обзор с Климатом и другими устройствами.

Таким образом мы установили ОС Home Assistant, с соблюдением принципов автоматизации, настроили её автоматический запуск, установили все необходимые дополнение и интеграции, добавили все устройства и вынесли их на главный экран для управления. Следующим этапом будет настройка процессов управления системой и устройствами.

Глава 4. Автоматизация процессов в Home Assistant

4.1 Язык YAML для написания кода

Главным способом конструирования графического интерфейса, дашбордов, карточек устройств и автоматизаций является написания кода на языке YAML. YAML - YAML Ain't Markup Language™, т.е. не язык разметки (хотя есть старые трактовки как «еще один язык разметки»). Является дружественным и удобным для человека языком сериализации[10] данных для всех языков программирования.

Основные элементы YAML:

- потоки YAML используют печатаемые Unicode-символы, как UTF-8, так и UTF-16
- отступы из пробелов (символы табуляции не допускаются) используются для обозначения структуры
- комментарии начинаются с символа «решётки» (#), могут начинаться в любом месте строки и продолжаются до конца строки
- списки обозначаются начальным дефисом (-) с одним членом списка на строку, либо члены списка заключаются в квадратные скобки (**[ ]**) и разделяются запятой и пробелом (**,**)
- ассоциативные массивы представлены двоеточием с пробелом (**:**) в виде ключ: значение, по одной паре *ключ-значение* на строку, либо в виде пар, заключённых в фигурные скобки и разделенных запятой и пробелом (**,**)
    - ключ в ассоциативном массиве может иметь в качестве префикса вопросительный знак (**?**), что позволяет указать сложный ключ, например представленный в виде списка
- строки записываются без кавычек, однако могут быть заключены в одиночные или двойные кавычки
    - внутри двойных кавычек могут быть использованы экранированные символы в С-стиле, начинающиеся с обратной косой (\)
- YAML позволяет задавать подстановки с помощью якорей & и псевдонимов (*). (Эванс, 2023)

Несмотря на то, что YAML конкурирует с XML и JSON, подходы к построению семантики у него совершенно иные. XML для обозначения логических блоков и структур

[10] Сериализация (в программировании) — процесс перевода структуры данных в битовую последовательность.

данных использует парные теги, например, «<message>Текст</message>», а JSON для организации структур данных использует словари («{key: value}) и списки ([item_1, item_2, …]»).

В YAML некоторые элементы синтаксиса хорошо знакомы Python-программистам, например, отступы (пробелы) для указания структуры документа и знак # — однострочный комментарий. Также некоторые структуры YAML можно записать однострочной записью очень похожей на синтаксис Python, например, словарь можно записать так: {ключ: значение, ключ: значение}.

Синтаксис YAML очень прост, достаточно запомнить несколько простых правил:

- файл должен начинаться с 3 дефисов— «---», а заканчиваться 3 точками — «…»;
- первый блок (словарь или список) — задает рабочее количество отступов, которое будет использовано во всём документе;
- после знака «-», обозначающего элемент списка, ставится пробел;
- все именованные наборы данных и строки должны иметь уникальные названия.

Строка — простая примитивная запись данных по типу ключ: значение. В YAML строки можно не брать в кавычки, за исключением случая, когда мы прописываем конвейеры команд в Linux.

В интерпретации Python строка в YAML — это словарь, а значит в качестве ключа или названия строки может быть текст и цифры. Дополним пример выше строкой с цифрой в ключе и выведем это всё на печать через Python:

«{'string': 'This is a string.', 'command': 'sh interface | include Queueing strategy:', 2: 'This is two'}»

Список — это упорядоченная коллекция данных, доступ к которым возможен по их индексам. Вот пример простого списка, где все значения строки. Так же как в Python, YAML-списки могут содержать цифры и булевы значения. Python воспринимает именованный список как словарь, где ключом является название списка, а значением — сам словарь. Вот как это выглядит при печати:

«{'list': [2, 'element2', True]}»

Вложенные именованные списки получаются добавлением двоеточия к нужному элементу, который станет названием вложенного списка и добавлением отступов к другим элементам, которые станут элементами вложенного списка.

Словарь — это набор данных по типу «ключ: значение». Словарь в YAML можно записать блоком. Словари могут иметь вложенность.

Пример кода:

```
---
authors:
  Ivan Katkov:
    age: 35
    speciality: IT
    skill: High
    publications:
      - publication1
      - publication2
      - publication3
  Fedor Pupkin:
    age: 30
    speciality: IT
    skill: Normal
    publications:
      - publication1
      - publication2
      - publication3
...
```

Блочная запись YAML выглядит хорошо и опрятно. Если бы мы писали подобную структуру однострочным кодом, это выглядело бы не слишком понятно:

```
{'authors': {'Ivan Katkov': {'age': 35, 'speciality': 'IT', 'skill': 'Hight', 'publications': ['publication1', 'publication2', 'publication3']}, 'Fedor Pupkin': {'age': 30, 'speciality': 'IT', 'skill': 'Normal', 'publications': ['publication1', 'publication2', 'publication3']}}}
```
(1cloud, 22)

Допускается запись словаря непосредственно в блоке кода YAML.

В данном случае написание автоматизаций будет происходить путем написания словарей.

4.2 Процесс создания автоматизаций (GUI или YAML)

Создание автоматизаций, скриптов и сцен возможно двумя методами – через GUI (графический пользовательский интерфейс): Настройки -> Автоматизации и службы; и непосредственно написанием кода: «File editor -> /config/». В данном разделе будут файлы с расширением «.yaml». Главным файлом конфигурации является файл «configuration.yaml», в котором можно создавать собственный интерфейс и карточки, прописывать доступ к другим файлам, настраивать работу интеграций и устройств, создавать виртуальные сенсоры и объекты. Помимо этого, в разделе «/config/» хранятся все файлы с автоматизациями, скриптами, сценами, таймерами, секретными ключами, базами данных (Рисунок 24). Основное отличие создания автоматизация путем кода от GUI (Рисунок 25) является то, что все графические автоматизации будут храниться в одном общем файле «automations.yaml» (аналогично со скриптами и сценами). Создание автоматизаций кодированием позволяет создать отдельные папки под отдельные задачи, а путь к папкам добавить в файл конфигурации. Данный способ позволит разнести все скрипты по отдельным файлам, что является более удобным для дальнейшей работы и поиску скриптов.

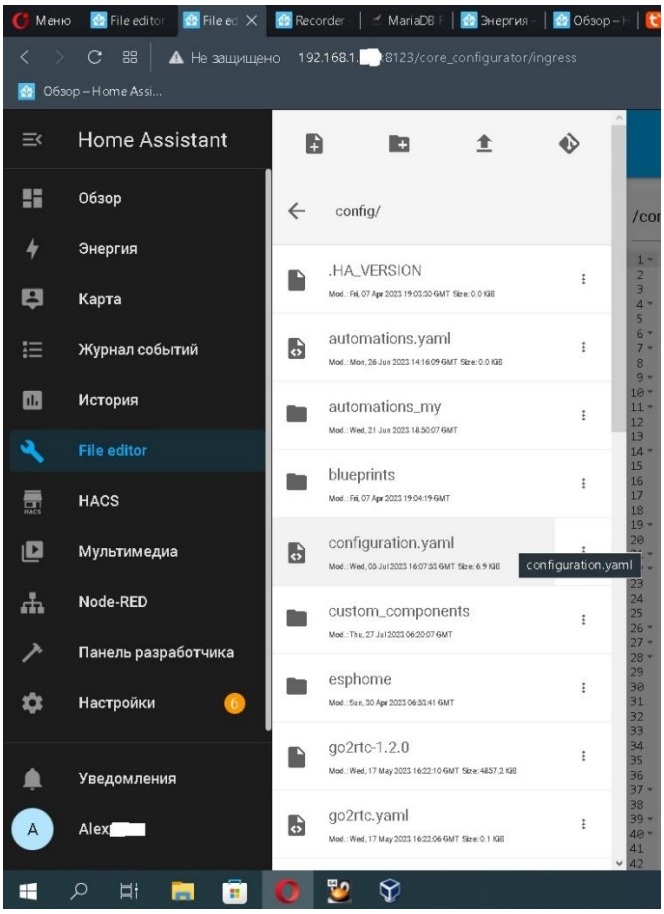

Рисунок 24. Файлы конфигурации.

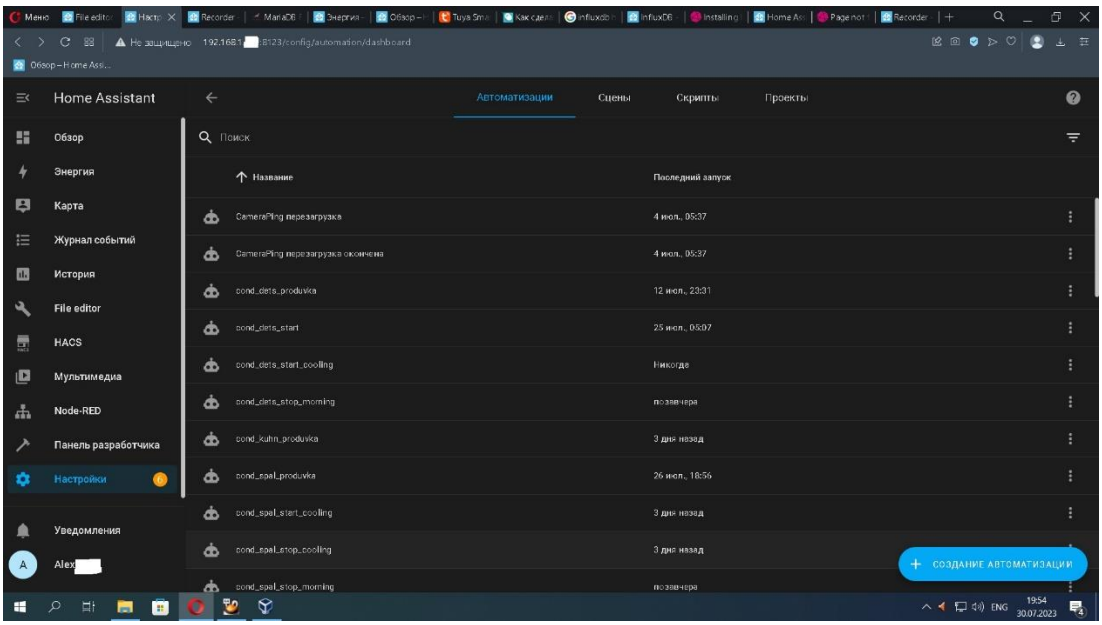

Рисунок 25. Ручное создание автоматизаций.

Таким образом, в директории /config/ создаем папку с названием «automations_my», куда будем вносить все файлы автоматизации. В файле «configuration.yaml» прописываем код (Код 2), чтобы система видела новую папку и брала скрипты из файлов, в ней расположенных.

Код 2. Добавление папки «automations_my» в файл конфигурации.

```
# добавляем папку, для упорядочения файлов, автоматизаций и объектов Home Assistant:
  packages: !include_dir_merge_named automations_my
# стандартные файлы автоматизаций
automation: !include automations.yaml
script: !include scripts.yaml
scene: !include scenes.yaml
timer: !include timers.yaml
```

Таким образом у нас будет одна директория, в которой будут хранится файлы с кодом автоматизации (Рисунок 26).

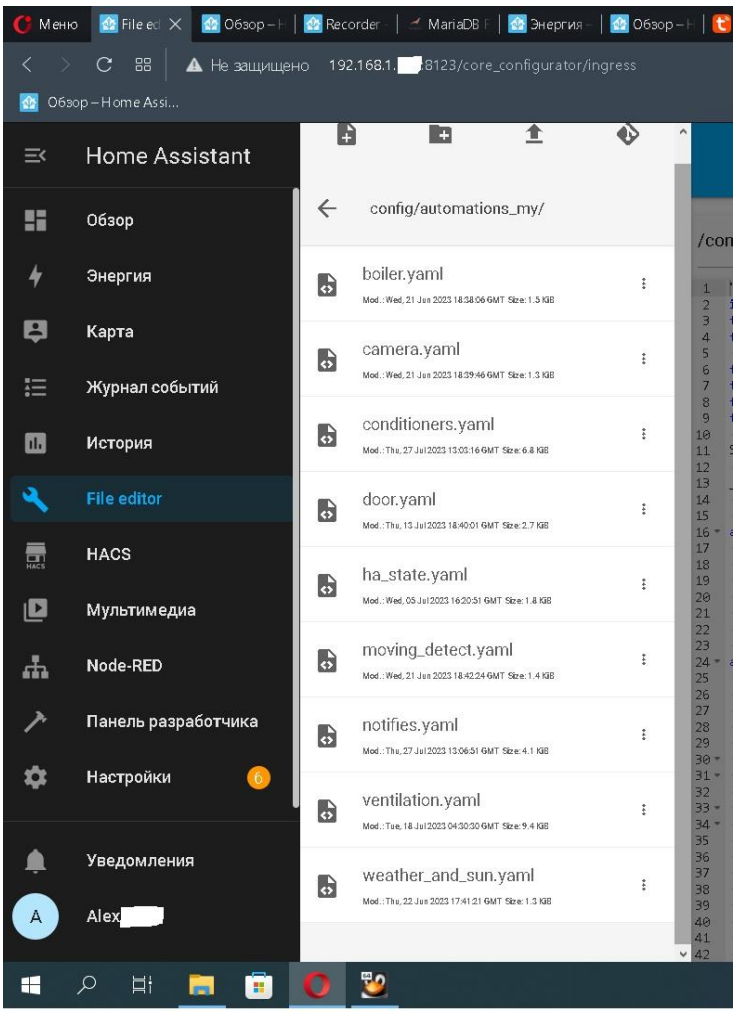

Рисунок 26. Директория с файлами автоматизаций.

Программа ведет запись журнала с любыми происходящими событиями, поэтому встроенная база данных довольно быстро станет занимать много места на жестком диске. Поэтому дополнительно добавим автоматическую очистку базы данных со сроком хранения данных 60 дней (Код 3). Программа ведет за

Код 3. Автоматическая очистка базы данных.

```
# настройка хранения Истории в БД
recorder:
  auto_purge: true # авто отчистка истории
  purge_keep_days: 60 # хранение данных 60 дней
```

Некоторые автоматизации должны отправлять уведомления (личные и групповые), поэтому добавим Телеграм-бота (Telegram-bot) и Уведомления (Notify) в файл конфигурации (Код 4). Уведомления можно группировать и рассылать только на указанные устройства или

указанные id Телеграм-чата. Id чатов, сущностей и IP-адреса скрыты по соображениям конфиденциальности.

Код 4. Добавление уведомлений через Телеграм и на телефоны.

```
telegram_bot:
  - platform: polling
    api_key: !secret telegram_token
    parse_mode: html
    allowed_chat_ids: # Допустимые чаты для рассылки
      - ******** # Личный
      - ******** # Жена
      - ********* # Общедомовой чат в Телеграм
      - ******* # Сантехник

# Уведомления в Телеграм и телефон
notify:
  - name: me # Псевдоним для персонального chat_id
    platform: telegram
    chat_id: *********

  - name: dom44
    platform: telegram
    chat_id: *********

  - name: oksi
    platform: telegram
    chat_id: **********

  - name: oleg
    platform: telegram
    chat_id: **********
#  Групповое уведомление по горячей воде Дом.
  - name: hotwater_all
```

```
platform: group
services:
  - service: dom44
  - service: mobile_app_le2121
```

Таким образом, всё готово для написания основных автоматизаций процессов умного дома.

4.3 Автоматизации процессов

Так как автоматизация работы устройств решает определенные задачи, то будут описываться проблемы и решаемы задачи с указанием кода автоматизации. Обычно автоматизация состоит из трех составляющих: триггер (trigger) - запуск автоматизации по какому-либо явлению; условие (condition) - или условия, при которых возможен запуск автоматизации; действие (action) - описывает, какие процессы будут выполнять при срабатывании триггера. Условие является не обязательным элементом, триггер и действие – обязательным. Конструкция может усложняться в зависимости от процесса. Триггерами выступают: изменение состояния (state), изменение цифрового значения (numeric_state), таймер (time), событие (event) и другие. В качестве условий могут быть эти же элементы. Поскольку все взаимодействие идет непосредственно с устройствами, то следует обозначить, что устройство обладает уникальным идентификационным номером (id), который состоит из домена (domain), к которому относится устройство (sensor, input_boolean, switch, climate, weather и др.) и сущности (датчика, показателя, состояния, и т.д.), которое происходит от названия. Например, «input_boolean.switch5» или «sensor.sonoff_100170f***_temperature». Процессы и примеры написания автоматизаций описаны в официальной документации HA на странице https://www.home-assistant.io/docs/automation/basics/.

Таким образом, каждая задача будет выделена и описана в отдельном файле со своим кодом.

Задача №1: Автоматическое управление водонагревателем и кранами по датчику температуры горячего водоснабжения (ГВС). При срабатывании должно приходить оповещение на телефон о включении или выключении водонагревателя. Создадим файл «boiler.yaml» и напишем код автоматизации (Код 5).

Код 5. Автоматическое управления водонагревателем

```
boiler:
    automation: # Указываем, что код относится к классу automation

    # НАЧАЛО включение водогрейки по датчику Температуры ГВС

    - id: 'Boiler ON' # id автоматизации отображается в Журнале событий
      alias: "Включение водогрейки"
      trigger: # запуск автоматизации
        - platform: numeric_state # цифровое значение
          entity_id: sensor.sonoff_100170f***_temperature # название сущности – сенсор ГВС
          below: 40 # если значение перешагнуло порог в 40 градусов (ниже 40)
          for: '02:00:00' # и находится ниже 40 градусов в течение 2 часов
      condition: # условие срабатывания автоматизации
      - condition: state # состояние сущности
        entity_id: input_boolean.switch6 # выключатель (реле) №6
        state: 'off' # должно быть выключено
      action: # действие, при выполнении всех условий (или последовательность действий)
        - service: input_boolean.turn_on # выполнить функцию для сущности – «включение»
          entity_id: input_boolean.switch6 # выключателя №6
        - service: notify.homers # пришли уведомление на телефоны
          data: # данные для уведомления
            message: "Включаю водогрейку" # текст уведомления
            title: 'Водогрейка' # название уведомления

    - id: 'Boiler OFF' # выключаем водонагреватель при повышении ГВС > 44 градусов.
      alias: "Выключение водогрейки"
```

```
  trigger:
    - platform: numeric_state
      entity_id: sensor.sonoff_100170f***_temperature
      above: 44
      for: '02:00:00'
  condition:
  - condition: state
    entity_id: input_boolean.switch6
    state: 'on'
  action:
    - service: input_boolean.turn_off
      entity_id: input_boolean.switch6
    - service: notify.homers  # уведомление на телефон
      data:
        message: "Выключаю водогрейку"
        title: 'Водогрейка'

# КОНЕЦ включение водогрейки по датчику Температуры ГВС
```

Датчик температуры подключен к стояку ГВС, при падении температуры ГВС ниже 40 градусов в течение 2 часов срабатывает триггер, проверяется условие (реле №6 блока Kincony выключено), и срабатывает автоматизация: включается реле №6, подается питание на промежуточное реле, которое питает водонагреватели, подается питание на краны шаровые (Рисунок 5) для распределения водоснабжения, отправляется уведомление на телефоны домочадцев (Рисунок 27).

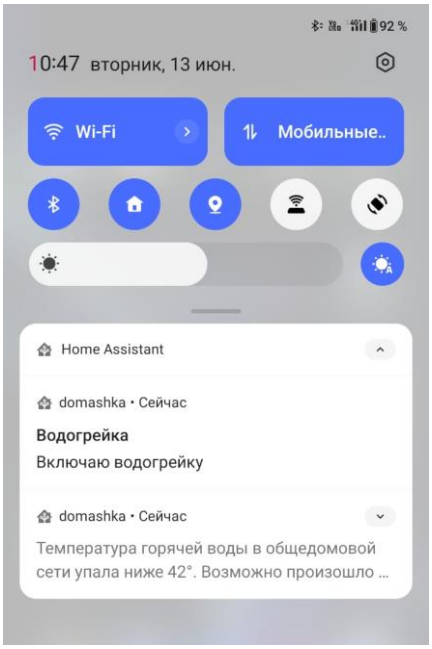

Рисунок 27. Уведомление на телефон от Home Assistant о включении водонагревателя.

Задача №2. Обеспечить временную подачу питания на краны шаровые 12В.

Проблема – существующие концевые выключатели в приводах кранов не до конца срабатывают, что вызывает постоянное дергание привода в попытке открыть/закрыть кран до конца.

Необходимо, чтобы при срабатывании реле №6 (водогрейки) подавалось напряжение на краны в течение 5-7 секунд и далее снималось. Обеспечить этот процесс физическими переключателями представлялось мало возможным. Проблема была решена путем автоматизации этого процесса в Node-RED (проще, чем писать код в YAML).

Питание (минус) на краны будет подаваться с реле №17 блока KC868-H32. Добавляем в Node-RED условие для включения реле №17 (условие – срабатывание реле водогрейки), ставим задержку на включение 2 секунды и на выключение 8 секунд, отправляем текст с командой типа «RELAY-SET-255,17,1» на узел TCP/IP (Рисунок 28). Таким образом при

включении водонагревателя через 2 секунды сработают (переведутся в нужное положение) все краны шаровые, а через 6 секунд отключатся.

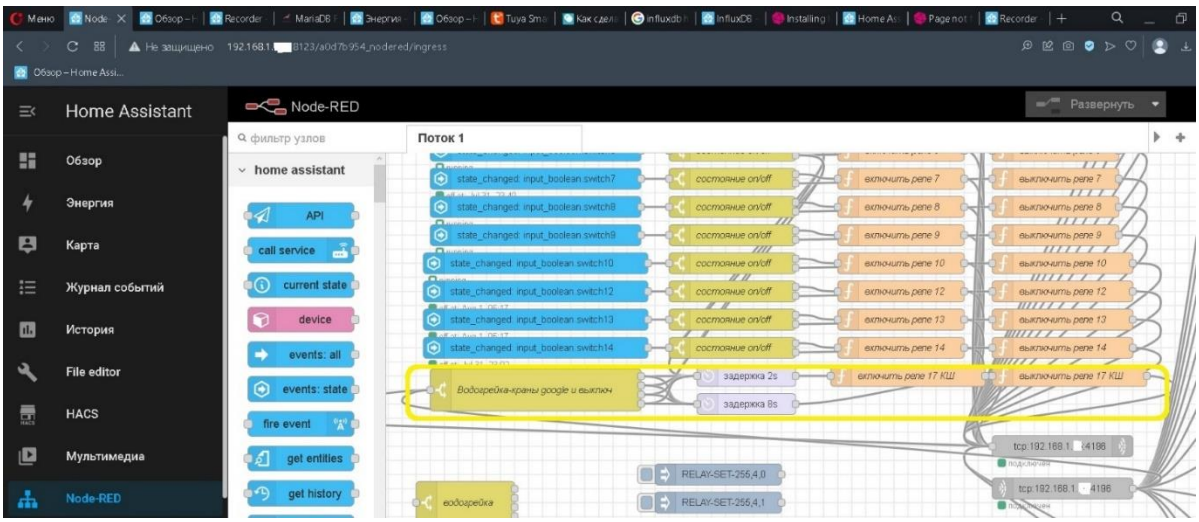

Рисунок 28. Автоматическая подача напряжения на реле №17 блока KC868-H32 (краны шаровые).

Задача №3. Автоматическая перезагрузка IP-камеры.

Проблема – IP-камера иногда зависает и её приходится перезагружать. Решение этой проблемы состоит из трех этапов. Этап 1 – подключаем питание камеры на реле №18 блока KC868-H32. Этап 2 – отслеживание подключения камеры. Будет происходить за счет отслеживания Ping[11] устройства по IP адресу. Для этого в файле конфигурации создадим бинарный сенсор (с двумя состояниями – «on» или «off»), и пропишем Код 6:

Код 6. Создание бинарного сенсора для камеры и ресивера.

```
# Мониторинг работы устройств
binary_sensor:
    # мониторинг работы Ресивера (и электросети 220V)
  - platform: ping # используемая платформа
    host: 192.168.1.** # IP адрес устройства
    name: "ResiverPing" # название для сенсора
    count: 5 # количество пакетов для отправки
```

[11] Ping — утилита для проверки целостности и качества соединений в сетях на основе TCP/IP, а также обиходное наименование самого запроса. Утилита отправляет запросы (ICMP Echo-Request) протокола ICMP указанному узлу сети и фиксирует поступающие ответы (ICMP Echo-Reply). Время между отправкой запроса и получением ответа (RTT, от англ. *Round Trip Time*) позволяет определять двусторонние задержки по маршруту и частоту потери пакетов, то есть косвенно определять загруженность на каналах передачи данных и промежуточных устройствах. (Muuss, 2023)

```
    scan_interval: 300 # интервал сканирования в секундах
    # мониторинг работы камеры
  - platform: ping
    host: 192.168.1.**
    name: "CameraPing"
    count: 5
    scan_interval: 300
device_tracker: # добавим отслеживание нужных устройств
  - platform: ping
    hosts:
      host_reciver: 192.168.1.**
      host_camera: 192.168.1.**
      host_alex: 192.168.1.**
      host_oksi: 192.168.1.**
```

Этап 3 – Создадим файл «camera.yaml» и напишем автоматизацию по перезагрузке камеры (Код 7).

Код 7. Автоматизация перезагрузки камеры.

```
camers:
    automation:

    #Начало мониторинг и перезагрузка камеры в коридоре

    - id: 'CameraPing перезагрузка'
      alias: CameraPing перезагрузка
      trigger:
        platform: state # состояние сенсора
        entity_id: binary_sensor.cameraping # название сущности
        from: 'on' # состояние изменяет с «on» («включено»)
        to: 'off' # на состояние «off» («выключено»)
        for: '00:20:00' # состояние «off» в течение 20 минут
      action:
        - service: input_boolean.turn_on # запускаем команду «включить»
```

```
        entity_id: input_boolean.switch18 # включаем реле №18
    # при изменении состояния на «on» в течение 10 секунд реле №18 выключается:
    - id: 'CameraPing перезагрузка окончена'
      alias: CameraPing перезагрузка окончена
      trigger:
        - platform: state
          entity_id: input_boolean.switch18
          from: 'off'
          to: 'on'
          for: '00:00:10'
      action:
        - service: input_boolean.turn_off
          entity_id: input_boolean.switch18
```

Задача №4. Обеспечить контроль открытия входной двери с настройкой оповещений.

Для этого будем мониторить состояние пользователей в домашней сети (подключение к Wi-Fi). У сущности два состояния – «home» и «not_home». Контроль состояния входной двери («открыто»-«закрыто») будет осуществляться через датчик открытия, который имеет два состояния: «on» и «off». Когда кто-то из пользователей в сети контроль двери выключен, когда дома никого нет – включается контроль состояния двери. При изменении состояния будет приходить оповещение на телефон. Для этого создадим файл «door.yaml», виртуальный переключатель с названием «input_boolean.door_check» и напишем Код 8.

Код 8. Автоматический контроль входной двери.

```
door:
    automation:
      #НАЧАЛО АВТОконтроль по наличию людей в доме

    - id: 'Door_check_on_auto'
      alias: Door_check_on_auto
      trigger:
        - platform: state
          entity_id: device_tracker.host_alex # Алексей
```

```
        from: 'home' # состояние из «дома» меняется
        to: 'not_home' # на состояние «не дома»
      - platform: state
        entity_id: device_tracker.host_oksi # Оксана
        from: 'home'
        to: 'not_home'
    condition:
        condition:  and  #  одновременное  выполнение  нескольких
условий
        conditions:
          - condition: state
            entity_id: device_tracker.host_alex # если Алексей не
дома
            state: 'not_home'
          - condition: state
            entity_id: device_tracker.host_oksi # если Оксана не
дома
            state: ' not_home'
    action:
    - service: input_boolean.turn_on # включение
      entity_id: input_boolean.door_check

  - id: 'Door_check_off_auto'
    alias: Door_check_off_auto
    trigger:
      - platform: state
        entity_id: device_tracker.host_alex # Алексей дома
        from: 'not_home'
        to: 'home'
      - platform: state
        entity_id: device_tracker.host_oksi # Оксана дома
        from: 'not_home'
        to: 'home'
    action:
```

```
    - service: input_boolean.turn_off
      entity_id: input_boolean.door_check

  #КОНЕЦ АВТОконтроль по наличию людей в доме

#НАЧАЛО уведомления от входной двери

  - id: 'Door_check_on'
    alias: Door_check_on
    trigger:
      platform: state
      entity_id: binary_sensor.sonoff_100187**** # входная дверь
      from: 'off'
      to: 'on' # если входная дверь открылась
    condition:
    - condition: state
      entity_id: input_boolean.door_check # если включен контроль
      state: 'on'
    action:
    - service: notify.mobile_app_le2121
      data:
       message: "Входная дверь открылась"
       title: "Входная дверь"

  - id: 'Door_check_off'
    alias: Door_check_off
    trigger:
      platform: state
      entity_id: binary_sensor.sonoff_1001879*** # входная дверь
      from: 'on'
      to: 'off' # если входная дверь закрылась
    condition:
    - condition: state
      entity_id: input_boolean.door_check # если включен контроль
```

```
      state: 'on'
    action:
    - service: notify.mobile_app_le2121
      data:
       message: "Входная дверь закрылась"
       title: "Входная дверь"

#КОНЕЦ контроль входной двери
```

Задача №5. Контролировать работу Home Assistant и питания сети 220В.

Для контроля запуска работы НА сделаем уведомления, приходящие на телефон. Для контроля состояния сети будем использовать сервис Ping и мониторить работу Ресивера. При отключении Ресивера (пропадании 220В), будет приходить уведомление на телефон. При отсутствии питания более 30 минут сервер принудительно выключит работу Home Assistant. Создадим файл «ha_state.yaml» и пропишем туда Код 9.

Код 9. Контроль работы Home Assistant и сети 220В.

```
ha_state:
    automation:
    # уведомление о запуске НА
    - id: 'Notify_HA_Start'
      alias: "Уведомление о запуске НА"
      trigger:
        - platform: Home Assistant
          event: start # триггер – запуск НА
      action:
        - service: notify.mobile_app_le2121 # сообщение на телефон
          data:
            message: "Home Assistant - включился"

    # НАЧАЛО отключение питания = выключается ресивер, приходит оповещение
```

```
    - id: 'Reciver ON'
      alias: Reciver ON
      trigger:
        platform: state
        entity_id:  binary_sensor.resiverping  # ранее созданный
сенсор ping
        from: 'off' # меняет состояние с выключен
        to: 'on' # на включен
      action:
      - service: notify.homers # уведомление домочадцев
        data:
         message: "Электропитание 220В восстановлено"
         title: 'Электричество'

    - id: 'Reciver OFF'
      alias: Reciver OFF
      trigger:
        platform: state
        entity_id: binary_sensor.resiverping
        from: 'on'
        to: 'off'
      action:
      - service: notify.homers
        data:
         message:  "Электропитание  220В  прекращено.  Возможно  нет
электричества"
         title: 'Электричество'

    # КОНЕЦ отключение питания

    # НАЧАЛО выключение НА хоста при потере питания
    - id: 'HA turn OFF'
      alias: HA turn OFF
      trigger:
```

```
    platform: state
    entity_id: binary_sensor.resiverping
    from: 'on'
    to: 'off'
    for: '00:30:00' # если нет питания более 30 минут
  action:
  - service: notify.me # приходит уведомление на телефон
    data:
     message: "HA выключается"
     title: 'Home Assistant'
  - service: hassio.host_shutdown # HA завершает работу
```

Пример оповещений представлен на Рисунке 29.

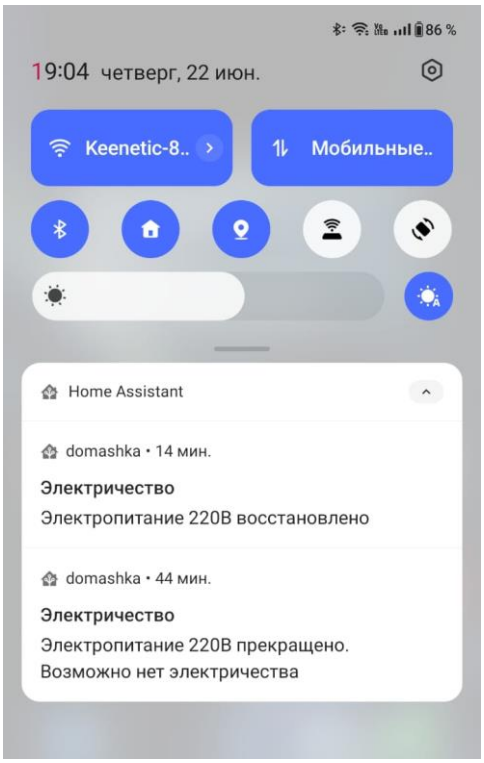

Рисунок 29. Оповещение о подаче э/э 220В.

Задача №6. Автоматическое включение и выключение датчиков движения.

В комнатах расположены датчики движения, которые включают ночную подсветку. Они подключены через реле №12-14 контроллера КС868-Н32. Их работа нужна только в темное время суток. Здесь применим платформу «sun», которая имеет два состояния: «sunrise» и «sunset». Поэтому автоматизация будет срабатывать по Заходу Солнца и Восходу Солнца Создадим файл «moving_detec.yaml» и пропишем туда Код 10.

Код 10. Автоматическое включение датчиков движения.

```
moving_detectors:
    automation:
    #начало включения д движ
        - id: 'Включение ночных д движ по sunset'
      alias: "д движ on"
      trigger:
        - platform: sun # плафторма sun
          event: sunset # событие – «закат»
      action:
        - service: input_boolean.turn_on # включаем объект
          target:
            entity_id: input_boolean.switch12 # выключатель №12
        -  delay:  '00:00:05'  #  делаем  строчку  5  секунд  для
последовательной отправки сообщений по TCP/IP
        - service: input_boolean.turn_on
          target:
            entity_id: input_boolean.switch13
        - delay: '00:00:05'
        - service: input_boolean.turn_on
          target:
            entity_id: input_boolean.switch14

    - id: 'Выключение ночных д движ по sunrise'
      alias: "д движ off"
      trigger:
        - platform: sun
          event: sunrise # восход
      action:
        - service: input_boolean.turn_off
          target:
            entity_id: input_boolean.switch12
        - delay: '00:00:05'
```

```
    - service: input_boolean.turn_off
      target:
        entity_id: input_boolean.switch13
    - delay: '00:00:05'
    - service: input_boolean.turn_off
      target:
        entity_id: input_boolean.switch14

#КОНЕЦ включения д движ
```

Задача №7. Автоматизировать работу приточной вентиляции.

Приточная вентиляция решает несколько задач – обеспечивает приток чистого воздуха в помещении и его подогрев при отрицательных температурах, снижает уровень шума в помещении при закрытых окнах. При этом её работа будет зависеть от нескольких параметров.

Во-первых, приточная вентиляция должна работать только при закрытых окнах. Для контроля закрытия окон будут использованы датчики открытия на двух окнах в разных комнатах. При закрытии окон вентиляция будет автоматически включаться, при открытии - отключаться. Во-вторых, вентиляция может работать по дневному и ночному расписанию при необходимости. В-третьих, добавим возможность снижения скорость вентиляции по времени или в зависимости от температуры «за окном». Создадим файл «ventilation.yaml», два виртуальных переключателя «input_boolean.fan_day_timer» и «input_boolean.fan_night_timer» для работы по расписанию, и напишем туда Код 11.

Код 11. Автоматизация работы приточной вентиляции.

```
ventilation:
    automation:

    #НАЧАЛО мониторинг открытие окна и автоматизация приточной вент.
    - id: 'Window_bedroom_check_open'
      alias: Window_bedroom_check_open
      trigger: # один из триггеров
        - platform: state # при открытии окна в спальне
          entity_id: binary_sensor.sonoff_10018** # датчик открытия1
          to: 'on' # окно открыто
```

```
    - platform: state # при открытии окна в детской
      entity_id: binary_sensor.sonoff_100187* # датчик открытия2
      to: 'on'
  condition:
    condition: not # при условии, что вентиляций НЕ выключена
    conditions:
      - condition: state
        entity_id: climate.fan_home
        state: 'off'
  action:
    - service: climate.turn_off # выключаем
      entity_id: climate.fan_home # вентиляцию

- id: 'Window_bedroom_check_close'
  alias: Window_bedroom_check_close
  trigger:
    platform: state
    entity_id: binary_sensor.sonoff_10018*** # датчик открытия1
    to: 'off'
  condition:
  - condition: state
    entity_id: climate.fan_home # вентилятор выключен
    state: 'off'
  - condition: state
    entity_id: binary_sensor.sonoff_10018*** # датчик открытия2
    state: 'off'
  action:
    - service: climate.turn_on
      entity_id: climate.fan_home

#КОНЕЦ мониторинг открытие окна и автоматизация приточки

#НАЧАЛО расписание приточки
- id: 'fan_start' # включение по таймеру
```

```
    alias: fan_start
    trigger: # триггер – время (одно из множества)
      - platform: time
        at: '21:00:00'
      - platform: time
        at: '22:00:00'
      - platform: time
        at: '23:00:00'
    condition: # при условии, что вентиляция выключена, а окна
закрыты
    - condition: state
      entity_id: climate.fan_home
      state: 'off'
    - condition: state
      entity_id: binary_sensor.sonoff_100187b** # датчик открытия1
      state: 'off'
    - condition: state
      entity_id: binary_sensor.sonoff_100187a** # датчик открытия2
      state: 'off'
    action:
      - service: climate.turn_on # включаем
        entity_id: climate.fan_home # вентиляцию

  - id: 'fan_finish' # включение по таймеру
    alias: fan_finish
    trigger:
      platform: time
      at: '09:00:00'
    condition:
      condition: not
      conditions:
        - condition: state
          entity_id: climate.fan_home
          state: 'off'
```

```
    action:
      - service: climate.turn_off
        entity_id: climate.fan_home
  #КОНЕЦ расписание приточки


# Расписание дневное

  - id: 'fan_day_timer_on' # включение по дневному расписанию
    alias: "fan_day_timer_on"
    trigger:
      - platform: time
        at: "12:00:00"
      - platform: time
        at: "17:00:00"
      - platform: time
        at: "19:00:00"
      - platform: time
        at: "21:00:00"
    condition:
    - condition: state
      entity_id: input_boolean.fan_day_timer # включена работа по
таймеру
      state: 'on'
    - condition: state
      entity_id: climate.fan_home
      state: 'off'
    - condition: state
      entity_id: binary_sensor.sonoff_100187b** # датчик открытия1
      state: 'off'
    - condition: state
      entity_id: binary_sensor.sonoff_100187** # датчик открытия2
      state: 'off'
    action:
```

```
            - service: climate.turn_on
              entity_id: climate.fan_home


        - id: 'fan_day_timer_off' # выключение по дневному расписанию
          alias: "fan_day_timer_off"
          trigger:
            - platform: time
              at: "15:00:00"
            - platform: time
              at: "18:00:00"
            - platform: time
              at: "20:00:00"
          condition:
            condition: not
            conditions:
              - condition: state
                entity_id: input_boolean.fan_day_timer # включена работа
по таймеру
                state: 'off'
              - condition: state
                entity_id: climate.fan_home
                state: 'off'
          action:
            - service: climate.turn_off
              entity_id: climate.fan_home

    # Режимы вентилятора

    # fan_modes:
    #   - auto
    #   - high
    #   - medium
    #   - low
```

```
- id: 'fan_day_timer_mode_medium' # включение средней скорости
  alias: "fan_day_timer_mode_medium"
  trigger: # по таймеру
    - platform: time
      at: "21:05:00"
  action:
    - service: climate.set_fan_mode # установить режим
      target:
        entity_id: climate.fan_home # вентилятора
      data:
        fan_mode: medium # средний

- id: 'fan_day_timer_mode_high' # включение максимальной скорости
  alias: "fan_day_timer_mode_high"
  trigger:
    - platform: time
      at: "08:00:00"
  action:
    - service: climate.set_fan_mode
      target:
        entity_id: climate.fan_home
      data:
        fan_mode: high

- id: 'fan_day_timer_mode_low' # включение слабого режима
  alias: "fan_day_timer_mode_low"
  trigger: # по таймеру
    - platform: time
      at: "23:00:00"
    - platform: time
      at: "03:00:00"
  condition:
    - condition: numeric_state
```

```
            entity_id: sensor.weather_temp # Если температура на улице
            below: 16 # ниже 16 градусов
        action:
          - service: climate.set_fan_mode
            target:
              entity_id: climate.fan_home
            data:
              fan_mode: low # слабый режим


    # Расписание ночное

      - id: 'fan_night_timer_on' # ночное включение вентиляции
        alias: "fan_night_timer_on"
        trigger:
          - platform: time
            at: "02:00:00"
          - platform: time
            at: "04:00:00"
          - platform: time
            at: "06:00:00"
        condition:
        - condition: state
          entity_id: input_boolean.fan_night_timer # включена работа
по таймеру
          state: 'on'
        - condition: state
          entity_id: climate.fan_home
          state: 'off'
          # ВАЖНО - при добавлении новых условий, и состояние будет
"недоступно",
          # может не сработать автоматизация
        - condition: state
          entity_id: binary_sensor.sonoff_100187b** # датчик1
```

```
      state: 'off'
    - condition: state
      entity_id: binary_sensor.sonoff_100187a** # датчик открытия2
      state: 'off'
    action:
      - service: climate.turn_on
        entity_id: climate.fan_home

  - id: 'fan_night_timer_off' # ночное выключение вентиляции
    alias: "fan_night_timer_off"
    trigger:
      - platform: time
        at: "00:30:00"
      - platform: time
        at: "02:30:00"
      - platform: time
        at: "04:30:00"
      - platform: time
        at: "06:30:00"
    condition:
      condition: not
      conditions:
        - condition: state
          entity_id:  input_boolean.fan_night_timer  # включена
работа по таймеру
          state: 'off'
        - condition: state
          entity_id: climate.fan_home
          state: 'off'
    action:
      - service: climate.turn_off
        entity_id: climate.fan_home

#КОНЕЦ Расписание ночное
```

 # НАЧАЛО открыть/закрыть клапан воздуха на кухне для максимальной подачи воздуха в комнаты ночью

```
- id: 'Клапан воздуха на кухне откр'
  alias: "кухня воздух ОТКР"
  trigger:
    - platform: time
      at: "23:00:00"
  condition:
  - condition: state
    entity_id: input_boolean.switch7
    state: 'off'
  action:
    - service: input_boolean.turn_on
      entity_id: input_boolean.switch7

- id: 'Клапан воздуха на кухне закр'
  alias: "кухня воздух ЗАКР"
  trigger:
    - platform: time
      at: "07:00:00"
  condition:
  - condition: state
    entity_id: input_boolean.switch7
    state: 'on'
  action:
    - service: input_boolean.turn_off
      entity_id: input_boolean.switch7
```

 # конец включение клапан воздуха на кухне

Задача №8. Автоматизировать работу кондиционеров.

Задача, помимо автоматического включения, будет решать и одну из главных проблем с кондиционерами – мгновенное завершение работы кондиционера при его выключении, когда он работал в режиме охлаждения. Производитель не заложил возможность проветривания радиатора внутреннего блока кондиционера после охлаждения. При этом, если его не проветрить в режиме «Вентиляция», то на радиаторе будет образовываться сырость и плесень. Запрограммируем кондиционеры на переключение в режим «Вентиляция» при завершении работы после режима «Охлаждение».

Включение кондиционеров будет зависеть от внутренней температуры в комнате, Солнца и погодных условий. В качестве погодных условий будем использовать домен weather с названием «weather.forecast_domashka», который может принимать несколько состояний: «sunny», «rainy», «cloudy», «partly_cloudy». Дополнительно создадим в файле конфигурации сенсоры, которые будут передавать значение Температуры по погоде. Создадим файл «conditioners.yaml» и впишем Код 12.

Код 12. Автоматизация работы кондиционеров

```
conditioners:
    automation:

    #НАЧАЛО включение кондиц в детской по восходу солнца и погоде
# Так как сторона солнечная, то при восходе солнца и солнечную погоду
# температура # в комнате повышается.
    - id: 'cond_dets_start'
      alias: cond_dets_start
      trigger:
        - platform: sun
          event: sunrise # при восходе Солнца
      condition: # выполнение одновременно всех условий
      - condition: state
        entity_id: climate.konditsioner_v_detskoi
        state: 'off' # кондиционер должен быть выключен
      - condition: state
        entity_id: weather.forecast_domashka # погодные условия
```

```
      state: "sunny" # должно быть солнечно
    action:
      - service: climate.set_hvac_mode # режим кондиционера
        target:
          entity_id: climate.konditsioner_v_detskoi
        data:
          hvac_mode: fan_only # включить Вентиляцию
      - delay: '00:00:01' # задержка 1 сек
      - service: climate.set_fan_mode # режим вентилятора
        target:
          entity_id: climate.konditsioner_v_detskoi
        data:
          fan_mode: LOW # на слабой скорости

  - id: 'cond_dets_stop_morning' # выключение кондиционера
    alias: cond_dets_stop_morning
    trigger:
      - platform: time
        at: '08:02:00'
    condition:
        condition: or # одно из условий
        conditions:
        - condition: state
          entity_id: climate.konditsioner_v_detskoi
          state: 'fan_only' # или вентиляция
        - condition: state
          entity_id: climate.konditsioner_v_detskoi
          state: 'cool' # или охлаждение
    action:
      - service: climate.turn_off
        entity_id: climate.konditsioner_v_detskoi

  - id: 'cond_spal_stop_morning'
    alias: cond_spal_stop_morning
```

```
  trigger:
    - platform: time
      at: '08:01:00'
  condition:
      condition: or # одно из условий
      conditions:
      - condition: state
        entity_id: climate.konditsioner_v_spalne
        state: 'fan_only'
      - condition: state
        entity_id: climate.konditsioner_v_spalne
        state: 'cool'
  action:
    - service: climate.turn_off
      entity_id: climate.konditsioner_v_spalne

- id: 'cond_dets_start_cooling' # режим Охлаждения
  alias: cond_dets_start_cooling
  trigger:
    - platform: numeric_state # Если температура в комнате
      entity_id: sensor.konditsioner_v_detskoi
      above: 29 # выше 29 град (датчик завышает Т на неск град)
  condition: # при условии
  - condition: state
    entity_id: climate.konditsioner_v_detskoi # что кондиционер
    state: 'fan_only' # в режиме вентиляции
  action:
    - service: climate.set_hvac_mode # перевести в режим
      target:
        entity_id: climate.konditsioner_v_detskoi
      data:
        hvac_mode: cool # Охлаждения
    - delay: '00:00:02'
    - service: climate.set_temperature # установить температуру
```

```
      target:
        entity_id: climate.konditsioner_v_detskoi
      data:
        temperature: 27 # в 27 градусов

- id: 'cond_spal_start_cooling' # включ кондиц в спальне
  alias: cond_spal_start_cooling
  trigger:
    - platform: time
      at: '15:30:00'
  condition: # при условии, что
    - condition: numeric_state
      entity_id: sensor.cond_v_spal_temp # температура в комнате
      above: 29 # больше 29 градусов
  action:
    - service: climate.turn_on # включить кондиционер
      target:
        entity_id: climate.konditsioner_v_spalne
    - delay: '00:00:05'
    - service: climate.set_hvac_mode # выставить режим
      target:
        entity_id: climate.konditsioner_v_spalne
      data:
        hvac_mode: cool # охлаждения
    - delay: '00:00:05'
    - service: climate.set_temperature # задать температуру
      target:
        entity_id: climate.konditsioner_v_spalne
      data:
        temperature: 25 # 25 градусов
    - delay: '01:00:00' # через 1 час
    - service: climate.set_temperature # задать температуру
      target:
        entity_id: climate.konditsioner_v_spalne
```

```
      data:
        temperature: 26 # 25 градусов

- id: 'cond_spal_stop_cooling' # выключение кондиционера спал
  alias: cond_spal_stop_cooling
  trigger: # по таймеру в 17 часов
    - platform: time
      at: '17:00:00'
  condition: # при соблюдении условий, что
    condition: not
    conditions:
      - condition: state
        entity_id: climate.konditsioner_v_spalne
        state: 'off' # кондиционер НЕ выключен
      - condition: state
        entity_id: device_tracker.host_alex
        state: 'home' # Алексей НЕ дома
  action:
    - service: climate.set_hvac_mode # перевести в режим
      target:
        entity_id: climate.konditsioner_v_spalne
      data:
        hvac_mode: fan_only  # вентиляции
    - delay: '00:05:00' # через 5 минут
    - service: climate.turn_off # выключить кондиционер
      target:
        entity_id: climate.konditsioner_v_spalne


#КОНЕЦ включение кондиц по расписанию

# НАЧАЛО включение временного продува после охлаждения
- id: 'cond_dets_produvka'
  alias: cond_dets_produvka
```

```
  trigger: # триггером будет перевод кондиционера
    - platform: state
      entity_id: climate.konditsioner_v_detskoi
      from: "cool" # из состояния Охлаждения
      to: 'off' # в состояние Выключено
  action: # в таком случае он не выключится
    - service: climate.set_hvac_mode # переведем кондиционер
      target:
        entity_id: climate.konditsioner_v_detskoi
      data:
        hvac_mode: fan_only # в режим Вентиляция
    - delay: '00:05:00' # выключение через 5 минут
    - service: climate.turn_off
      entity_id: climate.konditsioner_v_detskoi

- id: 'cond_spal_produvka'
  alias: cond_spal_produvka
  trigger:
    - platform: state
      entity_id: climate.konditsioner_v_spalne
      from: "cool"
      to: 'off'
  action:
    - service: climate.set_hvac_mode
      target:
        entity_id: climate.konditsioner_v_spalne
      data:
        hvac_mode: fan_only
    - delay: '00:05:00'
    - service: climate.turn_off
      entity_id: climate.konditsioner_v_spalne

- id: 'cond_kuhn_produvka'
  alias: cond_kuhn_produvka
```

```
  trigger:
    - platform: state
      entity_id: climate.konditsioner_na_kukhne
      from: "cool"
      to: 'off'
  action:
    - service: climate.set_hvac_mode
      target:
        entity_id: climate.konditsioner_na_kukhne
      data:
        hvac_mode: fan_only
    - delay: '00:05:00'
    - service: climate.turn_off
      entity_id: climate.konditsioner_na_kukhne

# КОНЕЦ включение временного продува после охлаждения
```

Задача №9. Добавить оповещения о наличии ГВС.

При включении и отключении ГВС необходимо присылать уведомления на телефон или в Телеграм. Уведомления так же приходят в общий чат дома для информирования соседей. Добавлена возможность проверки температуры ГВС для сантехника дома. Для этого будет создан Telegram-bot, который будет рассылать уведомления в общедомовой чат, и через который путем команды можно проверять температуру ГВС. Создадим файл «notifies.yaml» и впишем туда Код 13.

Код 13. Настройка уведомлений о ГВС.

```
hot_water:
    automation:

    #       -----Датчик горячей воды Ewelink ---------

    # НАЧАЛО оповещение о Горячей воде

    - id: 'Hot_water_Yes' # появление ГВС
```

```
  alias: "Уведомление о горячей воде в Telegram"
  trigger:
    - platform: numeric_state # когда цифровое значение
      entity_id: sensor.sonoff_100170f**_temperature # датчика
      above: 41 # находится выше 41 градусов
      for: '02:00:00' # в течение 2 часов
  action:
    - service: notify.hotwater_all # групповая рассылка всем
      data:
        message: "Горячая вода появилась"

- id: 'Hot_water_Yes_low' # подача ГВС
  alias: "Уведомление о горячей воде в Telegram"
  trigger:
    - platform: numeric_state
      entity_id: sensor.sonoff_100170f**_temperature
      above: 40 # когда температура датчика выше 40 градусов
      for: '00:30:00' # в течение 30 минут
  action:
    - service: notify.hotwater_all # оповестить всех
      data:
        message: "Началась подача ГВС"

- id: 'Hot_water_COLD' # отключение ГВС
  alias: "Уведомление о горячей воде в Telegram"
  trigger:
    - platform: numeric_state
      entity_id: sensor.sonoff_100170fb**_temperature
      below: 38 # когда температура датчика ниже 38 градусов
      for: '00:30:00' # в течение 30 минут
  action:
    - service: notify.hotwater_all
      data:
        message: "Температура горячей воды
```

```
                в общедомовой сети упала ниже 38°. ГВС отсутсвует."
                title: 'Горячая вода'

        # Запрос-ответ о ГВС через Телеграм

        - id: "telegram temp chech"
          alias: "telegram temp chech"
          trigger:  # триггером выступает команда в телеграм
          - platform: event
            event_type: telegram_command  # команда в телеграм
            event_data:
              command: '/temp' # выполнить эту команду
          action:
            -  service:  telegram_bot.send_message  #  телеграм  направит
сообщение
              data: # с информацией
                target: '{{ trigger.event.data.user_id }}' # для того,
кто сделал запрос, по user_id чата
                message: 'Температура ГВС
                {{  states("sensor.sonoff_100170fb**_temperature")}}'  #
извлечем атрибут датчика температуры – цифровое значение температуры
      # КОНЕЦ уведомления
```

Работу Телеграм-бота можно увидеть на Рисунке 30.

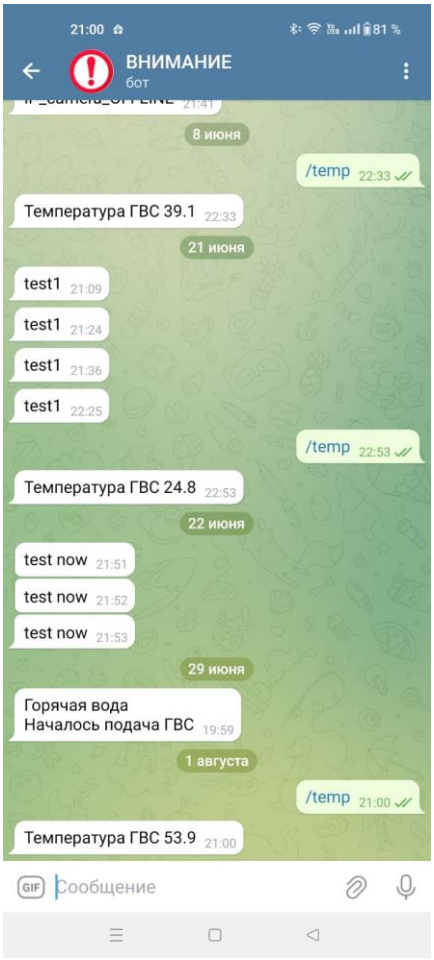

Рисунок 30. Получение показаний температуры ГВС через Телеграм-бота.

В Обзоре показания температуры ГВС можно вывести как карточку со Шкалой, добавив разноцветные уровни (Рисунок 31).

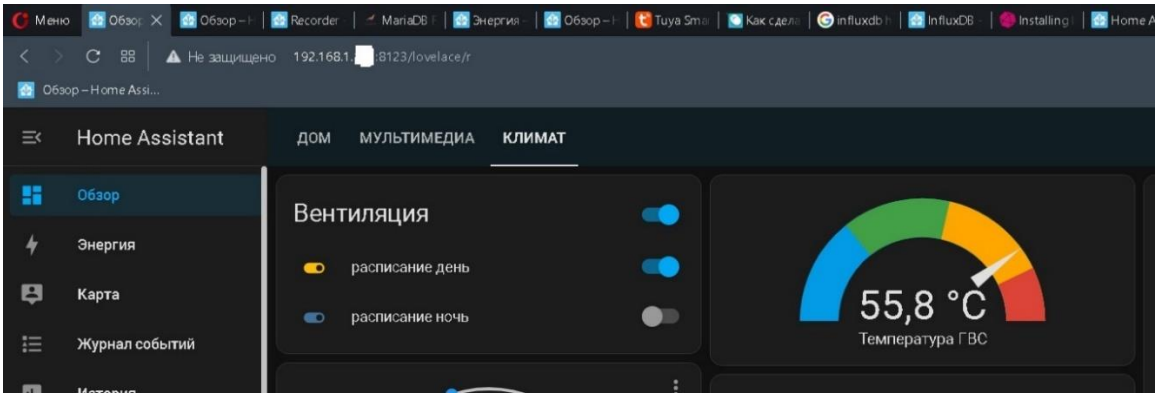

Рисунок 31. Шкала температуры ГВС.

Данные от датчика хранятся во встроенной базе данных SQLite и выводятся стандартными средствами по нажатию на карточку объекта (Рисунок 32).

Рисунок 32. График температуры ГВС за 3 дня.

Задача №10. Не промокнуть под дождем.

Для оповещений о грядущем дожде будем использовать домен погоды и его состояние. Создадим файл «weather.yaml» куда добавим Код 14.

Код 14.

```
weather_sun:
    automation:
      # НАЧАЛО автоматизация погоды
    - id: 'Погода дождь'
      alias: "Погода дождь"
      trigger:
        - platform: state # при состоянии
          entity_id: weather.forecast_domashka # погоды
          to: "rainy" # дождь
      action:
      - service: notify.mobile_app_le2121 # пришли уведомление на тел
        data:
         message: "Будет дождь" # текст уведомления
```

Для активации файлов с расширением «.yaml» требуется их перезагрузка в разделе «Панель разработчика -> YAML -> Проверка и перезапуск». Код проходит валидацию и, если нет ошибок, система вводит в эксплуатацию новый код. Конечно, любая написанная автоматизация нуждается в проверке на работоспособность и ошибки, которые необходимо устранять и заново тестировать работу. Порой это занимает довольно длительное время, так как большинство автоматизаций уникальны (каждый их пишет под себя), а примеры кода на официальном сайте зачастую не раскрывают и половины потенциала, при том, что взаимосвязи устройств и событий могут быть какими угодно.

4.4 Дополнительная база данных InfluxDB

InfluxDB представляется собой стороннюю базу данных, которую можно добавить в HA в качестве интеграции. При этом она не заменяет основную БД, а работает параллельно с ней. Официальная документация описана на сайте https://www.home-assistant.io/integrations/influxdb. По умолчанию Home Assistant ведет запись журнала с регистрацией абсолютно всех событий, поэтому стандартная БД довольно быстро займет много места. Ранее была сделана настройка на автоматическое очищение БД и хранение данных не более 60 дней. Существующая БД не позволяет разбивать сроки хранения данных для разных сущностей, она лишь позволяет указать, какие данные хранить, а какие не хранить. В Коде 15 включим хранение данных только для счетчиков и датчика температуры ГВС.

Код 15. Включаем данные для хранения в БД.

```
recorder: # настройка хранения Истории в БД
  auto_purge: true # авто отчистка истории
  purge_keep_days: 60 # хранение данных 60 дней
  auto_repack: false # для уменьшения БД
  include: # включаем определенные объекты (сенсоры)
    entities:
    - sensor.sonoff_100170fb**_temperature
    - sensor.saures_10670_32388
    - sensor.saures_10670_32389
    - sensor.saures_10670_32390
```

Таким образом, в БД будут храниться данные только от этих сущностей.

Дополнительная БД позволит более гибко настроить хранение нужных данных, а также получать к ним доступ.

5. Заключение

Итак, в ходе работы было сделано следующее: создана архитектура Умного дома (УД), выбраны устройства IoT и сопутствующее оборудование, настроен сервер, на виртуальной машине запущен Home Assistant (HA). Вместе с этим изучена работа виртуальной машины и ее запуска, документация и работа HA, процесс написания кода на языке «YAML», возможности создания автоматизаций в Node-Red и HA, различные интеграции с внешними устройствами.

HA позволил объединить в себе все устройства УД, связать некоторые из них и настроить нужные автоматизации. УД позволил решить все основные задачи, которые ставились изначально: подключение устройств, их объединение под одной локальной оболочкой, управление с телефона и компьютера, автоматизация работы устройств.

Итого, к положительным сторонам УД можно отнести следующие: локализация основной части устройств внутри домашней сети (без зависимости работы от иностранных облачных серверов), их объединение в одной программе, управление всеми заданными устройствами с телефона и компьютера, голосовое управление, настройка любых автоматизаций под свои нужды, интеграция практически любых устройств IoT.

К минусам УД можно отнести следующие: удорожание ремонта и в целом стоимость создания УД, сложность создания системы целиком, необходимы знания в программировании (особенно, если писать интеграции на Python), стабильность системы около 90% (бывают баги системы, отваливаются устройства при обновлениях, требуется наличие высокого уровня сигнала между устройствами). На изучение, настройку HA и написания автоматизаций потребовалось около двух месяцев.

В дальнейшем возможно расширение возможностей системы, за счет добавления новых датчиков температуры и влажности (для работы климатических устройств), датчиками CO2 (для контроля углекислого газа), установки регулирующих клапанов на радиаторы отопления (для более комфортного поддержания температуры воздуха), написания сценариев (например, «дома», «не дома», «кино» и других). Так же планируется изучение документации по написанию интеграций на Python, подключение к БД, создания FTP-сервера, создание более глубоких автоматизаций, не требующих ручного управления.

Данная работа позволит сделать вклад в развитие систем Умного дома в России, создать начальную документация по работе с сервером и УД, упростить работу с HA начинающим пользователям. С учетом стремительного развития экосистем УД, данная статья поможет построить бизнес-модель и сделать расчеты по созданию типового УД.

6. Список литературы и ресурсов

Список литературы и ресурсов

1. 1cloud. (22). *YAML для начинающих*. From www.1cloud.ru: https://1cloud.ru/blog/yaml_for_beginners
2. Ashton, K. (2014). *https://ru.wikipedia.org/wiki*. From https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82_%D0%B2%D0%B5%D1%89%D0%B5%D0%B9
3. Comer, D. E., & Stevens, D. L. (2023). *https://ru.wikipedia.org/wiki/Сервер_(аппаратное_обеспечение)*. From wikipedia.org: https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%B5%D1%80_(%D0%B0%D0%BF%D0%BF%D0%B0%D1%80%D0%B0%D1%82%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5)
4. Muuss, M. (2023). *ping*. From www.wikipedia.org: https://ru.wikipedia.org/wiki/Ping
5. Schoutsen, P. (2023). *https://ru.wikipedia.org/wiki/Home_Assistant*. From https://wikipedia.org/: https://ru.wikipedia.org/wiki/Home_Assistant
6. Schoutsen, P. (2023). *Установка (Installation)*. From www.home-assistant.io: https://www.home-assistant.io/installation/
7. А. В. Брусницын. (2016). *Принципы и уровни автоматизации процессов*. From www.business-gazeta.ru: https://www.business-gazeta.ru/article/298532#:~:text=%D0%9A%20%D0%BE%D1%81%D0%BD%D0%BE%D0%B2%D0%BD%D1%8B%D0%BC%20%D0%BF%D1%80%D0%B8%D0%BD%D1%86%D0%B8%D0%BF%D0%B0%D0%BC%20%D0%B0%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D0%B8%20%D0
8. *Концепция и терминология*. (2016). From www.home-assistant.io: https://www.home-assistant.io/getting-started/concepts-terminology/
9. ООО «Акрукс». (2020). *ОСНОВНЫЕ ПРИНЦИПЫ АВТОМАТИЗАЦИИ ПРОИЗВОДСТВЕННЫХ ПРОЦЕССОВ*. From www.akruks.net: https://www.akruks.net/article/avtomatizatsija/p568-osnovnye-printsipy-avtomatizatsii/

10. Эванс, К. (2023). *YAML*. From www.wikipedia.org: https://ru.wikipedia.org/wiki/YAML