GeekBrains

КОМПЛЕКС ФИКСАЦИИ, ХРАНЕНИЯ И ВИЗУАЛИЗАЦИИ ПАРАМЕТРОВ ОКРУЖАЮЩЕЙ СРЕДЫ С ДОСТУПОМ ЧЕРЕЗ ИНТЕРНЕТ

Программа:

Инженер умных устройств

Группа: 5471

Ромашин Ростислав Геннадьевич

Москва

2024

СОДЕРЖАНИЕ

Теоретическое описание

Человек является сложным биологическим видом, для жизнедеятельности которого необходимы определенные условия, при несоблюдении которых, будут происходить сбои или невозможность функционирования организма. Условия могут быть внутренними и внешними. Внутренние условия изучает раздел биологии — «анатомия».

Внешние условия окружающей нас среды называются экологическими от греческого слова *экос*, т.е. «местопребывание», а наука, изучающая взаимодействие живых организмов между собой и с внешним миров называется «экология».

Основными экологическими метриками являются:

1. температура окружающей среды
2. относительная влажность
3. атмосферное давление
4. чистота воздуха

При нарушении хотя бы одной из вышеперечисленных метрик, велика вероятность гибели живого организма.

Основные экологические метрики

1. Температура окружающей среды

Рефлексы человека способны с легкостью отличить тепло от холода. В большинстве случаев здоровый индивидуум способен по ощущениям дать и количественную оценку текущей температуры окружающей среды и даже температуру тела другого человека. Мама часто трогают лоб своему малышу, чтобы определить болен он или нет, а ведь разница составляет десятые доли градусов между здоровьем и простудой. Но, вот нездоровые, недееспособные люди или младенцы не способны сказать, что им холодно или тепло. Уставшие или спящие люди, тоже с трудом среагируют на резкие скачки температуры. Много случаев обморожений даже у трезвых людей возникает каждый зимний период. В результате вышеперечисленного, температурные нарушения могут быт причиной:

- гипотермии
- гипертермии

Гипотермия — это состояние, при котором температура тела снижается ниже нормы, что может привести к серьезным последствиям, таким как обморожение и смерть от переохлаждения. При гипотермии организм не способен поддерживать нормальную

температуру, что приводит к замедлению обмена веществ и нарушению работы органов. Важно предпринимать меры для нагревания тела и предотвращения серьезных осложнений.

Гипертермия — это повышение температуры тела, что может привести к тепловому удару, ожогам и другим тепловым повреждениям. При гипертермии организм перегревается, что может вызвать нарушение работы органов, дезориентацию и даже смерть. Необходимо немедленно принимать меры для снижения температуры тела и предотвращения серьезных осложнений.

Для предотвращения таких состояний, а также для контроля температуры в холодильниках, помещении и на улице, человек изобрел *термометр*. Почти в каждой квартире есть по несколько термометров: внешний, внутренний, медицинский. Показания данных приборов являются самой приоритетной биологической метрикой.

2. Относительная влажность

Человеческие рефлексы, также способны отличать влажность. Тактильные ощущения подкрепляются и визуальным восприятием. Так, мы можем сказать влажный климат или сухой. Но, мы не можем дать количественных значений влажности, как с температурой. Тут, наши рецепторы не настолько хороши. Комфортна и полезна средняя степень влажности климата. Некомфортными и даже опасными являются уровни влажности:

- высокий
- низкий

Высокий уровень влажности - способствует развитию плесени, бактерий и вирусов, что увеличивает риск респираторных заболеваний и аллергических реакций. Постоянное воздействие высокой влажности может привести к ухудшению качества воздуха в помещении и ухудшению здоровья.

Низкий уровень влажности может вызвать обезвоживание организма, раздражение слизистых оболочек и проблемы с дыханием. Сухой воздух может способствовать появлению сухости кожи, раздражения глаз и дыхательных проблем, особенно у людей с респираторными заболеваниями.

Приборы позволяющие измерить уровень относительной влажности, называются *гигрометрами*. Они реже встречаются чем термометры в квартирах, но все же пользуются популярностью из-за сложности определения значений влажности с помощью рефлексов.

3. Атмосферное давление

Данную метрику здоровый человек не в состоянии определить почти совсем, если это не экстремальные условия: погружение под воду или восхождение на гору. А вот больные люди, зачастую обладают метеочувствительностью из-за слабых сосудов, они чутко реагируют на колебания атмосферного давления, но и они не способны четко определить количественные характеристики данной метрики.

Изменения в атмосферном давлении могут вызвать баротравму, что проявляется в боли в ушах, давлении в груди и других серьезных проблемах. Баротравма может возникнуть при изменениях высоты, полетах на самолете или на скоростных лифтах.

Изменения атмосферного давления являются предвестниками дождей, бурь или тайфунов.

Из-за сложности определения показаний давления, большой популярностью пользуются *барометры*.

4. Чистота воздуха

Самая сложная для оценки человеческими органами чувств экологическая метрика. В воздухе могут присутствовать газы, не имеющие ни запаха, ни цвета, например угарный - смертельно опасен для человека. Природный газ тоже не имеет запаха, поэтому в системы газоснабжения добавляются специальные отдушки, для определения утечек. Но, все газы снабдить отдушками не получится и из-за их многообразия и большого количества в окружающем мире.

Чистый воздух, необходимый для здоровья человека, должен иметь определенные характеристики и состав.

Состав воздушной смеси необходимый человеку:

- Кислород (O2): 21-40%
- Азот (N2): 57-79%
- Углекислый газ (CO2): менее 0.04%
- Другие газы (: аргон, водяной пар, неон, гелий и другие): сотые доли %

Нарушение пропорций газов, включая перечисленные в воздушной смеси, называют «загазованностью». Например, небольшое увеличение углекислого газа, может вызвать следующие проблемы:

- Дыхательные проблемы: затруднение с дыханием, учащенное дыхание, одышка и удушье.
- Головокружение и головная боль
- Ухудшение концентрации и когнитивные проблемы
- Сердечные проблемы: нагрузка на сердце, учащенное сердцебиение, повышенное давление
- Сонливость и усталость

Помимо газов воздух может содержать токсичные вещества, такие как тяжелые металлы и химические соединения, что может привести к респираторным заболеваниям, раку и другим серьезным заболеваниям.

Данная метрика, самая сложная для оценки из вышеперечисленных. Для диагностики состояния используют дорогие *газоанализаторы* или большое количество датчиков для определения конкретного вещества в воздухе.

Перечисленные метрики являются основными, но не полными. В данной работе не рассматриваются радиационные, электромагнитные и другие метрики, в основном техногенного характера, их влияние на здоровье значительно, и они будут освещены в дальнейших работах в случае продолжения обучения.

Датчик BME680

BME680 — это датчик загазованности, атмосферного давления, температуры и влажности окружающей среды с низким энергопотреблением от компании BOSH (Приложение 1, Рисунок 1).

Модуль датчика помещен в крайне компактный корпус с металлической крышкой LGA, имеющий размеры всего 3,0 × 3,0 мм² и максимальную высоту 1,00 мм (0,93 ± 0,07 мм) (Приложение 1, Рисунок 2). Его небольшие размеры и низкое энергопотребление позволяют интегрировать его в устройства, работающие от батареи или в устройства с частотной связью, такие как мобильные телефоны или носимые устройства.

Из всего многообразия имеющихся на рынке датчиков для встраиваемых систем и IOT, данный датчик является самым дешевым и относительно точным решением для измерения всех основных, ранее описанных, экологических метрик.

Данный датчик 4-в-1 куплен на Али-экспресс за 353 рубля. Что в несколько раз дешевле специализированных на конкретный газ датчиков.

Характеристики BME680

- Измерение температуру: от -40°C до +85°C ±0.5°C
- Измерение влажность от 0% до 100% ±3%
- Измерение давление: от 300 до 1100 ГПа ±1 ГПа
- Измерение загазованности: bVOC, eCO2, IAQ
- I2C, SPI
- Напряжение питания: 1.8 и 3.3V
- Калибровка автоматического нуля
- Компенсация температуры
- Автоматический режим сна для экономии энергии

Загазованность

Понятие загазованности очень объемное. В разрезе BME-680, под загазованностью подразумевается, наличие в воздухе летучих веществ, которые выдыхает человек. В русском языке нет термина, обозначающего спертый воздух или уровень духоты в помещении, поэтому используется термин загазованность.

Датчик загазованности передает внутренние показаний драйверу, который на их основе рассчитывает 3 уникальные метрики:

1. b-VOC
2. eCO2
3. IAQ

1. b-VOC

Эквивалентное значение летучих органических соединений, вырабатываемых дыханием человека (volatile organic compounds in exhaled breath). Измеряются в частях на миллион (ppm).

Данная метрика не является общепринятой, она применяется в лабораториях БОШ, где проводились исследования по содержанию веществ в выдыхаемом воздухе здоровых людей. На основе тестов, была разработана особая смесь веществ эквивалентных дыханию

человека. Данную смесь, не более 6 месяцев хранения, используют при апробации датчиков на производстве в течении нескольких дней их работы.

Вещества и их пропорции в смеси представлены в таблице ниже:

Количество молекул	Вещество	Допустимая концентрация	Погрешность
5 ppm	Этан	20 %	5 %
10 ppm	Изопрен или 2-метил-1,3-бутадиен	20 %	5 %
10 ppm	Этанол	20 %	5 %
50 ppm	Ацетон	20 %	5 %
15 ppm	Моноксид углерода	10 %	2 %

2. eCO2

Эквивалентное значение содержания углекислого газа. Выражается в частях на миллион (ppm).

CO2 не измеряется датчиком на прямую, а вычисляется в зависимости от корреляции показаний bVOC, предположительному содержанию углекислого газа в выдыхаемом воздухе.

Выражается в частях на миллион (ppm).

3. IAQ

Качество воздуха в помещении (Indoor Air Quality) — индекс (метрика), который описывает качество воздуха внутри помещений. Он имеет прямое отношение к уровню загрязнения и чистоты воздуха на основе дыхания людей и их пота. Индекс также чувствителен к запаху от масляных картин, мебели, мусора, готовящейся либо распакованной еде и т.д.

При расчете IAQ учитываются следующие метрики:

- bVOC
- eCO2
- Влажность воздуха
- Температура воздуха
- Атмосферное давление

BME-680 — выдает два вида индекса:

- IAQ — рекомендовано для мобильных устройств, в ней учитывается погрешности при перемещении.

- Static IAQ — как следует из названия, метрика предназначена для помещений.

Все вышеперечисленные метрики загазованности является продвинутыми функциями BME-680, в их расчете участвуют мощности как датчика, так и оконечного оборудования с использованием проприетарного закрытого модуля с запатентованным алгоритмом BOSH.

Принцип работы BME-680

Как было описано ранее (Приложение 1, Рисунок 2) за измерение отвечает элемент размером с булавочную головку — чип с металлической каморой 3x3x1мм.

Камора на крышке сверху имеет маленько отверстие диаметром 0.35мм, через которое происходит забор воздуха. Это все, что видно невооруженным взглядом. Механизм и принцип работы этого устройства не описан в даташитах и бережно хранится в тайне.

Но, благодаря любителям реверс-инжиниринга удалось заглянуть внутрь каморы чипа с помощью электронного микроскопа, разрезав ее, с помощью лазерного скальпеля.

На рисунке (Приложения 2, Рисунок 1) расположены 3 платы с золотыми выводами:

1. 8-контактная комбинированная плата (слева ближе к зрителю)
2. 4-контактная плата загазованности (справа)
3. Плата управления (слева под первой платой)

1. Плата температуры давления и влажности

8-контактная плата комбинирует в себе сразу 3 датчика: температура, влажность и давление. Размеры платы 1,075мм на 1,35 мм (Приложения 2, Рисунок 1).

В центре платы расположена синяя квадратная площадка — это мембрана, реагирующая на атмосферное давления, энергию от деформации она передает на 8 серебряных датчиков в виде группы из 3-х квадратных точек по периметру мембраны.

Внизу платы видны два прямоугольных коричневых датчика влажности.

У правого края платы расположены два сенсора измеряющих температуру. Расположение термосенсоров не случайно, они находятся ближе всего ко второй плате справа.

2. Плата загазованности

Имеет размеры 0,6мм на 0.9мм (Приложение 2, Рисунок 2). В самом центре увеличенной платы загазованности (Приложение 2, Рисунок 3) мы видим круглую каплю оксида железа (Fe_2O_3), именно она, осуществляет большую часть волшебства. К этой капле

подходят 2 пары контактов, расположенные по диагонали против своего напарника. Тонкие контакты отвечают за измерение сопротивления воздуха, которое изменяется от соприкосновения летучих веществ с каплей оксида железа (Приложение 2, Рисунок 4). Толстые контакты отвечают за нагрев змеевика (Приложение 2, Рисунок 5), расположенного под каплей оксида железа имеющего крупные фракции (Приложение 2, Рисунок 6), которые могут накапливать энергию тепла. Змеевик разогревается до 320 градусов по Цельсию. Температуру нагрева можно регулировать программно, что обеспечивается интерфейсом библиотеки BSEC.

320 градусов по Цельсию — это тот порог, который позволяет избежать возгорания воздушной смеси. Например, пары спирта возгораются только при 367 градусах Цельсия. Но, в тоже время, при таком нагреве происходит моментальное испарение всех летучих веществ VOC внутри каморы. Более того, температура плавления припоя, как правило составляет более 320 градусов. Также сохраняется качество изоляционных материалов, таких как текстолит.

Нагрев змеевика длится около 150 миллисекунд, вот поэтому для контроля нагрева термодатчики расположены ближе к плате загазованности.

Далее, реализация механизма замеров загазованности представляет мои домыслы и возможно отличается от оригинальных закрытых технологий.

Расположение всех датчиков в одном корпусе не случайно, как мы поняли это позволяет регулировать температуру нагрева оксида железа. Когда оксид железа нагревается до определенной температуры, он взаимодействует с молекулами веществ в воздухе. Эти реакции изменяют электрические свойства оксида железа, что проявляется как изменение его электрического сопротивления. Молекулы веществ взаимодействуют с поверхностью оксида железа, приводя к обмену электронами или ионами. Например, при взаимодействии с восстановительными газами такие газы "захватывают" электроны с поверхности оксида, уменьшая его сопротивление. При взаимодействии с окислительными газами, они добавляют электроны на поверхность, что повышает сопротивление оксида.

3. Плата управления

Снижение или повышение сопротивления оксида железа фиксируется парой контактов на плате загазованности, описанной выше. Эти изменения преобразуются в цифровые значения, которые передаются на плату управления, где записываются во внутренние регистры для детальнейшей передачи на оконечное устройство. Датчик постоянно подстраивается и калибруется с учетом температуры, влажности и давления для получения

точных показаний. Используя комплексное мультисенсорное решение, датчик может точно различать концентрацию веществ в воздухе.

Но, зачем размещать датчик влажности и атмосферного давления в одной каморе? Про термодатчик мы выяснили выше: он, помогает контролировать нагрев. Размещение большого количества миниатюрных деталей в одном корпусе усложняет процесс «подковки блохи». Выскажу смелое предположение, что для определения загазованности используется не только сопротивление оксида железа, но и давление в каморе в процессе испарения газов. Т.к. при нагреве до 320 градусов происходит вскипание газов и вскипание воздушной смеси, в зависимости от концентрации в ней различных веществ образуется разное давление внутри каморы. Именно поэтому БОШ разместил датчик давления вместе с «испарителем».

Датчик влажности по тем же причинам находится в одной каморе: при резком нагреве летучих органических соединений, таких как спирты, эфиры, кетоны, может происходить их испарение и конденсация паров на более холодных поверхностях. Резкое нагревание некоторых кислот (например, соляной кислоты) или щелочей (например, аммиака) может вызывать их испарение и конденсацию паров на более холодных поверхностях. После резкого нагрева каморы, вскипевшая смесь не улетучится мгновенно через узкое отверстие в крышке, создавая тем самым давление, но постепенно нагретый воздух выйдет полностью наружу и заместиться более холодным из вне. Что, в свою очередь приводит к образованию конденсата, который в свою очередь, также нагревается и удаляется. Т.е. мы имеем дело не с одним циклом замера, а набором таковых: нагрев газа → термоконтроль → повторный нагрев при необходимости→ замер сопротивления → замер давления → выход газа → образование конденсата → замер влажности → и т. д. Как мы увидим далее, все это заложено в ПО, с помощью особых конфигов.

Драйвера BME680

Для BME680 существует три вида ПО:

1. Открытое: BME68x-Sensor-API
2. Проприетарное: BSEC
3. Сторонние решения

1. Открытое ПО

BME68x-Sensor-API поддерживает базовую коммуникацию с датчиком и функции компенсации данных, и доступно как открытый исходный код на GitHub.

Функционал открытого ПО ограничен. Бош бережно хранит алгоритм расчетов показаний загазованности в секрете. Более того, они удалили свой старый открытый репозиторий, который содержал более "простой" для понимания драйвер. В новом репозитории, упомянутом выше, лежит другой драйвер, который требует наличия "тяжелого" COINES SDK.

Ссылка на старый нерабочий репозиторий с открытым драйвером датчика BME680 до сих пор красуется в последнем даташите.

Но, нашел старый открытый драйвер 3.5.9 на стороннем репозитории.

Драйвер содержит открытие API. Для того чтобы начать получать или записывать информацию на датчик, необходимо самостоятельно написать основную программу для конкретной платформы и операционной системы, которая с помощью открытых API будет обращаться через драйвер к датчику. В результате, можно получать показания: температура, влажность, давление и сопротивление воздуха в Омах, последнее никаким интерпретациями не поддается.

В интернете можно встретить много отрицательных отзывов про датчик. Основой причиной недовольства служит именно показания сопротивления воздуха, которое никак не коррелирует с реальностью.

С arduino данный датчик можно использовать только в режиме получения: температура, влажность, давление и сопротивления воздуха, т. к. для расчета и использования метрик загазованности мощностей у atmega328 не хватит.

Встречаются негативные отзывы и по поводу замеров температуры воздуха, причем разница от бытовых приборов может доходить до десятков градусов по Цельсию. Причиной этого служит неправильно составленное самописное ПО и неверное использование режимов работы. Для активации правильного режима необходимо использовать конфиги и массивы стартовых параметров. Все это содержится в закрытом ПО.

2. Закрытое ПО

Чтобы раскрыть полный потенциал BME680 необходимо совместно с открытым ПО использовать закрытую часть BSEC (Bosch Software Environmental Cluster).

BSEC включает интеллектуальные алгоритмы, которые позволяют использовать такие сценарии, как мониторинг качества воздуха по различным метрикам eCO2, bVOC и индексу IAQ, числовые градации которого вместе с их влиянием на здоровье отображены в таблице из даташита (Приложение 3, Рисунок 1)

Программное обеспечение BSEC доступно в виде закрытого двоичного файла, который будет предоставлен через принятие Лицензионного Соглашения на Программное Обеспечение (SLA) на сайте Bosch Sensortec. Для доступа к сайту, по традиции, необходимо пользоваться ВПН.

Лицензия, запрещает выкладывать в открытый доступ проприетарное ПО, что является основной причиной кривых показаний сторонних библиотек, типа Adafruit.

На момент написания данной работы последняя версия BSEC 2.5.0.2 содержит зависимости от COINS SDK. Поэтому в своей работе использую версию BSEC 1.4.8.0 rev.3, способную работать без этих зависимостей, но, которою также необходимо с помощью ВПН и регистрации скачать с сайта BOSH.

В скачанном дистрибутиве, помимо открытых API, содержатся скомпилированные объектные файлы под различные архитектуры, включая ARMv6.

Дистрибутив содержит 8 конфигураций:

- generic_18v_300s_28d
- generic_18v_300s_4d
- generic_18v_3s_28d
- generic_18v_3s_4d
- generic_33v_300s_28d
- generic_33v_300s_4d
- generic_33v_3s_28d
- generic_33v_3s_4d

Датчик может работать с напряжениями: либо 1.8V, либо 3.3V, поэтому и идут разграничения по напряжению, т. к. это важная составляющая для нагрева оксида железа. Следующим параметров в конфигурации являются секунды. Ранее, писал, что принципы работы были связаны с набором замеров, прежде чем данные будут отданы драйверу, необходимо пройти несколько этапов.

Из списка конфигов видно, что есть режимы замеров в 3 секунды и 300 секунд. Почему именно эти магические цифры? Потому, что есть два основных режима работы и два вспомогательных:

1. ULP — ультранизкий режим сбережения энергии
2. LP — низкий режим потребления
3. q-ULP — быстрый ультранизкий
4. CONT — непрерывный режим, только для тестовых целей

ULP потребляет 0.09 mA тока, а LP 0.9 mA. Поэтому LP нагревает оксид железа быстрее и режим снятия показаний через 3 секунды против 300.

Если принудительно без конфигов, задействовать скажем CONT и замерять раз в секунду или раз в минуту. То, датчик не успеет выполнять тот набора замеров и будет возвращать либо неточные данные, либо датчик будет перегреваться, что скажется на температурной метрике.

При каждом новом включении, драйвером считывается файлы с внутренней памяти оконечного устройства: это файл последнего состояния или файл инициализации. В зависимости от выбранного конфига через определенное количество циклов (10000) происходит запись в файл состояний, основных показателей, что позволяет сократить время само-калибровки при каждом включении датчика. Если такого файла нет, то самонастройка происходит около 10 минут. В этот период доступны только метрики: температуры, влажность, давление и сырые данных с датчика загазованности.

3. Сторонние решения

Крупные проекты не обошли вниманием данный датчик, но большинство из них поддерживают BME-680 без вышеописанных индексов: IAQ, bVOC, eCO2, либо используют свои «костыльные» формулы расчета этих показателей, которые далеки от реальности.

Наиболее удачным проектом в плане реализации поддержки является Home Assistant.

Home Assistant

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.

Т.к. мое конечное устройство это одноплатник ODROID-X2, и он поддерживается только современным дистрибутивом ArchLinux, но не HassOS, то данный проект не рассматривался изначально. Более, того основная часть кода написана на python. Моя дипломная работа тесно связана с СИ.

Считаю HA наиболее перспективным проектом в плане IOT. Т.к. то, что реализую сейчас в дипломной работе, включая InfluxDB, NodeRed все это реализовано в их операционной системе HassOS.

В интернете часто можно встретить удачные примеры полноценного запуска BME680 на HA. Но, в официальной документации решение для данного датчика значится как

удаленное. Возможно по лицензионной причине, возможно по другим, но по факту официальной полной поддержки BME680 нет.

BalenaSense

Некое коммерческое подобие Home Assistant, также предлагает решение на базе своей ОС и docker. Предлагают облачное решение, по сути, личный кабинет, с помощью которого можно управлять вашим одноплатником, на который непосредственно и устанавливаются InfluxDB, Grafana и все что мы делаем сейчас. Идею установки докера на одноплатник отмел в сторону сразу, что будет с данными если плата зависла, пропал интернет, отключили свет. А тут еще предлагают своего трояна установить, который нашу телеметрию будет отсылать на их сайт.

Самое оптимальное решение, это одноплатник с датчиком отдельно, а системы БД, визуализации и т. д. на отдельном сервере (виртуальном или настоящем), т. к. такая схема позволяет подключить много одноплатников и не только.

Для тех, кому нужно готовое решение представители balenaSense написали целую статью на хабре, она именно решению с нашим bme680 посвящена. И вроде даже умеет метрику IAQ показывать, и очень подозрительная статья на хабре без кода. По факту они не используют проприетарных драйверов, а используют python библиотеку pimoroni, при просмотре кода которой виден самописный алгоритм расчета индекса. В самом расчете участвуют только влажность и сопротивление газа. Они пытаются учесть 50 предыдущих замеров, привести их к соотношению (25:75, humidity: gas). Но, в конце у них стоит *time.sleep(1).* Что, как я описал выше вообще неприемлемо, нужно либо 3секунды, либо 300!

Adafruit

Аппаратно-программная компания предоставляет драйвер на базе открытого ПО. В репозитории только версия для arduino, которая умеет только измерять сопротивление воздуха и никаких проприетарных индексов.

WaveShare

Еще одна аппаратно-программная компания. Но, они уже честно на странице посвященной BME680 пишут, что их решение основано на базе открытого API, т. е. без вывода индексов загазованности. И вообще ссылки на скачивание их решения нет.

Для регламента дипломной работы конкурентов рассмотрел и не обнаружил ни одного готового программного решения для полноценной работы из самых доступных по цене и наличию датчика загазованности. Будем делать свое, после описания оконечного устройства и сервера.

Оконечное оборудование

Сам по себе датчик бесполезен без устройства, которое будет питать его, управлять и снимать показания. Упоминал выше про одноплатник ODROID-X2, к нему по интерфейсу I2C будет подключен датчик BME680

Основной проблемой Odroid-X2 является его "древность". Снят с поддержки лет 10 назад. Но, по мощности не уступает многим Raspberry даже сегодня. Официальной последней версией была Ubuntu 14.04. Умельцы сделали неофициальный образ на базе Ubuntu 16.04. Для языка СИ это не проблема, но хотелось, что-то посвежее.

Из современных дистрибутивов только ArchLinux имеет образ для odroid-x2 в официальном хранилище. Данная система позиционируется как более профессиональная версия Linux и одной из особенностей являются Rolling обновления. Т.е. как таковых версий у нее нету, периодически накатываются самые последние обновления и все. Таким образом получаем самое свежее ядро, библиотеки и ПО.

Данный одноплатник в состоянии выдержать docker и образы для хранения и визуализации данных. Но, как писал ранее, все это планирую установить на сервер, который в свою очередь будет хранилищем данных не только для одного устройства, но и звеном для доступа к показаниям через интернет.

Сервер

В качестве сервера используется IBM PC совместимый компьютер с установленной на него Ubuntu 24.04. Данный подход позволит хранить информацию централизованно. Мощности позволят подключить множество оконечных устройств и большее количество пользователей данными, которые будут собираться с помощью всевозможных датчиков. Благодаря этому можно будет легко настроить параметры авторизации. Технология docker и, в частности, docker swarm позволят в будущем масштабироваться с помощью кластеров, распределить нагрузку и повысить отказоустойчивость.

Для сбора информации со всех оконечных устройств будет применяться протокол mqtt (брокер mosquitto) с помощью Telegraf все сообщения будут передаваться подписчикам, одним из которых на постоянной основе будет InfluxDB. Визуализация в Grafana.

Практическая реализация

Коммутация BME680

Схема соединения датчика с одноплатником по интерфейсу I2C:

```
BME680  Odroid-X2
+-----+-----------------+
GND         GND (50 pin)
VDD         VDD (48 pin) 1.8V
SDA         SDA (03 pin)
SCL         SCL (05 pin)
SDO         GND (50 pin)
```

По даташиту датчик может запитываться либо от 1.8В, либо 3.3В. Делов в том, что ODROID-X2 в отличие от Raspberry PI не имеет на своем борту питания 3.3 Вольта.

При подключении достаточно использовать 4 провода, но тогда датчик будет занимать вторичный адрес на интерфейсе 0x77, чтобы присвоить первичный адрес 0x76, необходимо контакт SDO датчика заземлить.

На фото (Приложение 4, Рисунок 1) видна схема подключения с помощью монтажной платы breadboard. На верху располагается преобразователь напряжения с диапазона 5-30 Вольт на 3.3 Вольта. Данный преобразователь был куплен по акции за 55 копеек именно из-за отсутствия 3.3В, для BME680 он не нужен, но вот для будущего проекта – датчика радиометра RadSense, без него не обойтись, т.к. его схема поддерживает только вышеупомянутое напряжение.

Внизу платы, подключен сам BME680. У него остается неподключенным только разъем CS, который задействуется при подключении по SPI.

Датчик подключается без подтягивающих резисторов, т.к. имеет собственные.

Одноплатник

На фото (Приложение 4, Рисунок 2) представлен стенд с прикреплёнными к нему одноплатником и макетной платой, типа breadboard. На макетной плате датчик и преобразователь напряжения.

Схема и назначение контактов GPIO ODROID-X2:

```
GND          50 49    ADC_AIN3
VDD_IO       48 47    ADC_AIN2
SYS (5V)     46 45    VD16
VD13         44 43    VD22
VD4          42 41    VD1
VD23         40 39    VD9
VD17         38 37    VD14
VD10         36 35    VD5
VD12         34 33    VD3
GND          32 31    HSYNC
VD20         30 29    VDEN
VD6          28 27    VSYNC
VD11         26 25    VD18
VD7          24 23    VCLK
VD0          22 21    VD15
VD8          20 19    VD2
VD21         18 17    VD19
TXD          16 15    SPI_1_MISO
RXD          14 13    PWM_BRT
VDDQ_LCD     12 11    SPI_1_MOSI
SPI_1_CSN    10 09    XE_INT12
T_SDA        08 07    T_SCL
SPI_1_CLK    06 05    SCL
T_RST        04 03    SDA
BL_EN        02 01    T_INT
```

Установка ArchLinux

Образ системы можно записать либо на SD-карту, либо на проприетарный скоростной eMMC модуль. Т.к. модуль у меня один, а одноплатника два (второй более мощный ODROID-XU) то, буду устанавливать систему на карту памяти.

Для начала карту нужно очистить от разделов и загрузочных секторов:

```
dd if=/dev/zero of=/dev/mmcblk2 bs=1M count=8
```

Важно с помощью fdisk создать первичный раздел, который будет начинаться с сектора 4096, таким образом останется 2 Мегабайта под загрузчик.

С XFS файловой системой у меня образ дружить не захотел. А жаль именно эта система идет по умолчанию при установке RedHat, т.к. она меньше изнашивает твердотельные накопители

и единственная ФС на ядре Linux, которая без проблем позволяет делать дамп системы на горячую. Будем использовать EXT4

```
mkfs.ext4 /dev/mmcblk2p1
```

Далее, монтируем диск в любую директорию, пусть это будет папка root:

```
mkdir root
mount /dev/mmcblk2p1 root
```

Скачиваем и распаковываем образ в нашу папку root

```
wget http://os.archlinuxarm.org/os/ArchLinuxARM-odroid-latest.tar.gz
bsdtar -xpf ArchLinuxARM-odroid-latest.tar.gz -C root
```

Самая важная часть, в первые свободные 2 Мегабайта на нашей карте нужно записать проприетарный загрузчик, без которого одноплатник даже не включится, т.к. он корейский и процессор Samsung требует инициализации с помощью закрытого U-boot:

```
cd root/boot
./sd_fusing.sh /dev/mmcblk2p1
```

Далее можно отмонтировать карту памяти, переключить на одноплатнике джампер в положение загрузки с SD. Вставить карту в ODROID и включить его.
Установленный образ позволяет работать с системой как через серийный порт, так и по SSH. Пароль и логин по умолчанию: alarm

Настройка ArchLinux

Менеджер пакетов по-умолчанию в системе ArchLinux, это pacman, необходимо инициализировать его и активировать систему цифровых подписей, с помощью которой будут проверяться в будущем скачиваемые пакеты:

```
pacman-key --init
pacman-key --populate archlinuxarm
```

Установим утилиты работы с интерфейсом I2C:

```
pacman -S i2c-tools
```

Далее посмотрим определился ли датчик:

```
# i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:                         -- -- -- -- -- -- -- --
10: UU -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
```

```
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- 76 --
```

Отлично, датчик занял первичный для него адрес.

Переходим к тестированию датчика.

Тестирование BMC680

Самым простым способом тестирования или запуска нашего датчика будет с использованием языка python и библиотеки bme680, установим ее в виртуальное окружение, т.к. система безопасности ArchLinux не позволяет устанавливать пакеты пайтона в системное окружение:

```
pip install bme680
```

В репозитории дипломного проекта выложил, набор простых python-скриптов для работы с датчиком, самый простой пример занимает буквально несколько строк кода:

```
import bme680
sensor = bme680.BME680(bme680.I2C_ADDR_PRIMARY)
sensor.set_humidity_oversample(bme680.OS_2X)
sensor.set_pressure_oversample(bme680.OS_4X)
sensor.set_temperature_oversample(bme680.OS_8X)
sensor.set_filter(bme680.FILTER_SIZE_3)
sensor.set_gas_status(bme680.ENABLE_GAS_MEAS)
sensor.set_gas_heater_temperature(320)
sensor.set_gas_heater_duration(150)
sensor.select_gas_heater_profile(0)
while True:
    if sensor.get_sensor_data() and sensor.data.heat_stable:
        print(f'{sensor.data.temperature} {sensor.data.pressure* 0.7501}
{sensor.data.humidity} {sensor.data.gas_resistance}')
```

Пробежимся по коду:

- Импортируем библиотеку **bme680**, которая содержит все необходимые функции для работы с датчиком BME680
- Создаем объект **sensor** класса **BME680**, используя адрес I2C по умолчанию I2C_ADDR_PRIMARY, чтобы датчик знал, какой I2C адрес использовать для связи.
- Устанавливаем коэффициент передискретизации (oversampling) для измерения влажности на 2x. Это увеличивает точность измерений, но также увеличивает время выполнения.

- Устанавливаем коэффициент передискретизации для измерения давления на 4x, что также улучшает точность измерений, но увеличивает время отклика.
- Устанавливаем коэффициент передискретизации для измерения температуры на 8x. Это максимальное значение для температуры для более точных измерений
- Устанавливаем размер фильтрации данных на 3. Фильтр сглаживает колебания в показаниях давления, вызванные быстрыми изменениями окружающей среды, например, при открывании или закрывании дверей или окон
- Включаем измерение сопротивление газа.
- Устанавливаем температуру нагревателя газового сенсора на 320°C
- Определяем длительность нагрева газового сенсора на 150 миллисекунд.
- Выбираем профиль нагревателя с индексом 0. Профиль позволяет датчику использовать ранее установленные параметры нагрева
- Запускаем бесконечный цикл, чтобы постоянно получать данные с датчика.
- Проверяем, удалось ли датчику получить новые данные и является ли измерение газа стабильным.
- Выводим показания температуры, давления (переведенное в миллиметры ртутного столба путем умножения на 0.7501), влажности и газового сопротивления в консоль.

Да, мы не видим самых вкусных метрик загазованности, т.к. не используем проприетарное ПО. Но, это самый быстрый из опробованных мною способов запустить в работу датчик.

Переходим к самому сложному - написание собственного драйвера на СИ с использованием отрытых API и проприетарного ПО.

Т.к. доступ к сайту БОШ закрытый, то я выложил их ПО временно в папку дистрибутивов репозитория диплома.

Как упоминал ранее в теоретической части. Буду использовать старое ПО v1.4.8.0.3.

Для осмысления происходящего вывожу структуру папок из архива дистрибутива в Приложении 5.

Дистрибутив содержит около 400 файлов, большинство из которых это код на СИ.

Но, не так все страшно, как кажется на первый взгляд. Ориентируясь на структуры папок понятно, что тут собрана приличное количество архитектур. Меня интересует процессор ARMv6.

Создаем папку нашего будущего драйвера **bsec** и копируем в нее все содержимое из директории:

```
.\BSEC_1.4.8.0_Generic_Release_updated_v3\algo\normal_version\bin\RaspberryPi\PiThree_ArmV6
```

Это два заголовочных файла с указанием прототипов данных и объявлением интерфейсов взаимодействия с третьим файлом – проприетарным объектным файлом **libalgobsec.a**. Само название модуля намекает, что в нем спрятан секретный алгоритм.

Копируем все файлы из директории с открытыми API

```
.\BSEC_1.4.8.0_Generic_Release_updated_v3\examples\bsec_iot_example
```

И выбираем нужный конфиг, в моем случае для 1.8 Вольта, замер каждый 3 секунды:

```
D:\github\dipiot\distr\BSEC_1.4.8.0_Generic_Release_updated_v3\config\generic_18v_3s_28d
```

28 дней в названии конфига, указывает на то, что каждые 28 дней будут происходить сохранения состояний на карту памяти. Очень разумно, т.к. встраиваемые системы не любят частую запись, что сокращает срок службы носителей информации.

Итого в нашей папке остается 15 файлов из четырёх сотен.

Приступаем к написанию основного файла bsec.c, в котором будем использовать переменные, типы данных и функции, объявленные в остальных файлах с очень подробными комментариями в коде от производителя.

В итоге файл bsec.c содержит 220 строк кода на СИ против дюжины на python. На, самом деле при использовании только открытых API у меня получилось менее 100 строк.

Полученные файлы нужно скомпилировать с правильными ключами и слинковать их с проприетарной библиотекой.

Привожу листинг моего Make-файла:

```
TARGET=bsec
ARG=-lm -lrt -static -std=c99 -pedantic
all: ${TARGET}
${TARGET}: ${TARGET}.o bsec_integration.o bme680.o libalgobsec.a
	gcc -o ${TARGET} ${TARGET}.o bsec_integration.o bme680.o libalgobsec.a ${ARG}
bsec_integration.o:
	gcc -o bsec_integration.o -c bsec_integration.c
bme680.o:
	gcc -o bme680.o -c bme680.c
${TARGET}.o: ${TARGET}.c
	gcc -o ${TARGET}.o -c ${TARGET}.c
```

Убрал из кода выше часть инструкций, отвечающих за установку собранного приложения и конфигов в систему, а также очистку от объектных файлов. Оставил только самое интересное.

При компиляции используем следующие ключи:

```
-lm – математические функции, нужны секретному модулю для расчета газа
-lrt – подключаем библиотеку реального времени
-static – собираем статически все в один исполняемый файл
-std=c99 – соответствие стандратру C99
-pedantic – предупреждать о любых нарушениях синтаксиса
```

После **make install** бинарный файл **bsec** копируется в папку */usr/local/sbin* в которую, по доброй старой традиции, помещают пользовательские системные приложения. Данная директория по умолчанию прописана в переменной окружения PATH, поэтому можно команду **bsec** выполнять из любой директории нашей системы.

Драйвер написан таким образом, что в случае указания ключа *-o*, программа выведет одну строку измерений в консоль и завершится. В случае запуска без ключей **bsec** будет выводить в консоль 13 показаний в бесконечном режиме с интервалом в 3 секунды, пока не будет принудительного завершения.

Список показаний и метрик, выводимых в консоль:

- temperature – компенсированная температура
- raw_temperature – необработанное значение температуры
- humidity – компенсированная влажность
- raw_humidity – необработанная влажность
- pressure - давление
- gas – сопротивление воздуха
- co2_equivalent – эквивалентное значение CO2
- breath_voc_equivalent – эквивалентное значение выдыхаемого человеком воздуха
- iaq – индекс чистоты воздуха, для использования в мобыльных устройствах
- static_iaq - индекс чистоты воздуха для помещений
- iaq_accuracy – точность индекса чистоты воздуха от 0 до 3
- bsec_status – статус работы программы bsec – 0 если нет ошибок
- timestamp – дата и время в формате unixtime, кол-во секунд от сотворения юникса 01.01.1970

Вывод в консоль программы **bsec**:

```
24.34 26.40 55.22 48.86 736.62 51 942.95135498 1.42525113 115.39 94.30 0.00 0 23190005303000
24.33 26.39 55.26 48.87 736.62 51 944.44036865 1.42846274 115.58 94.44 0.00 0 23196004424000
24.32 26.38 55.26 48.85 736.62 51 937.61511230 1.41380048 114.69 93.76 0.00 0 23202003632000
```

```
24.32 26.38 55.31 48.91 736.61 51 930.79541016 1.39930069 113.81 93.08 0.00 0 23208002900000
24.33 26.39 55.26 48.88 736.59 51 922.43688965 1.38173151 112.72 92.24 0.00 0 23214002093000
24.33 26.39 55.19 48.82 736.62 51 919.15991211 1.37490392 112.29 91.92 0.00 0 23220001333000
```

Перечисленные выше параметры вводятся в консоль через пробел. Такой формат позволяет их легко передавать далее по пайплайну в другую программу для обработки.

Основной драйвер готов, ранее мне не удавалось создать рабочий вариант с проприетарным модулем.

Использовал другой, написанной мной драйвер, использующий только открытые API, он выводил не 13 значений, а всего 4.

MQTT клиент

Желанные данные выводятся с датчика в консоль, их необходимо перенаправить на сервер для сохранения. Напишем своего mqtt-клиента на СИ с использованием библиотеки paho. На убунтах и редхатах библиотека paho легко устанавливается через нативные менеджеры пакетов. Гладко было на бумаге, да забыли про овраги. Основной проблемой является то, что пакета с библиотекой paho-mqtt-c нет в офрепозиториях ArchLinux.

Пришлось собирать ее самостоятельно из исходников, скачанных с официального репозитория.

Написал программу для mqtt-клиента. Реализована возможность с помощью ключей командной строки задать следующие параметры:

- -i - ip-адрес и порт mqtt-сервера
- -u - имя пользователя
- -p - пароль

Программа mqtt принимает со стандартного входа 12 параметров, 13-ый параметр, а именно время передавать не будем, т.к. у нас метка времени будет ставится mqtt сервером и базой временных рядов InfluxDB. 13-ый параметр нужен будет, для записи в файл, например.

Свадьба bsec и mqtt

Для того, чтобы данные программы работали сообща, был написан простенький bash-скрипт, использующий особенности систем на базе Linux. Это «именованный канал». С помощью программы **mkfifo** можно создать файл, для того чтобы не портить бесконечным циклом перезаписи нашу карту памяти. Данный файл будет работать по принципу очереди First Input First Output. Программа **bsec** будет построчно класть туда информацию, а **mqtt** построчно, при появлении этой самой строки будет считывать оттуда. Зачем такие

сложности, если можно было бы реализовать передачу информации на mqtt сервер сразу из драйвера **bsec**?

Во-первых, основной принцип Юникс - делать простые программы и затем использовать их через пайп.

Во-вторых, этим виртуальным файлом смогут пользоваться и другие программы, т.е. мы получаем механизм, когда одна и таже информация может получаться разными клиентами. Например, вторым клиентом может быть наш самописный веб-сервер. Тогда не будет конфликтов при вызове драйвера **bsec** два раза. В отличие от системных приложений, драйвер не может быть запущен дважды, т.к. одно устройство нельзя зарезервировать одновременно для двух управляющих им программ, в лучшем случае это приведет к остановке драйвера, а в худшем возможен вариант когда один драйвер включил разогрев оксида железа и через какое-то время, когда змеевик нагрелся до 320 градусов второй драйвер включил его еще раз, не сумев прочитать например показания температуры в итоге датчик выйдет из строя.

Стартовый скрипт для запуска конвейера **bsec** | **mqtt**:

```
#!/bin/bash
set -x
mkfifo /tmp/bsec
/usr/local/sbin/bsec > /tmp/bsec &
sleep 2
cat /tmp/bsec | /usr/local/sbin/mqtt &
```

Выставляем бит исполнения:

```
chmod 777 bsec.sh
```

Копируем в пользовательскую папку системных приложений

```
cp bsec.sh /usr/local/sbin/
```

В ArchLinux файл *rc.local* не является стандартным способом запуска скриптов при загрузке системы через *systemd*.

Однако, это можно исправить::

Создайте файл *rc.local* в каталоге **/etc**:

```
vim /etc/rc.local
```

Добавим в автозагрузку наш скрипт **bsec.sh**, с возможностью журналирования:

```
/usr/local/sbin/bsec.sh > /tmp/bsec.log 2>&1
```

Таким образом перенаправляем наш стандартный вывод и вывод ошибок в файл.

Выставляем бит исполнения:

```
chmod 770 /etc/rc.local
```

Создаем файл юнита *systemd* для *rc.local*:

```
sudo nano /etc/systemd/system/rc-local.service
```

В файл юнита добавляем следующее содержимое:

```
[Unit]
Description=/etc/rc.local Compatibility
ConditionPathExists=/etc/rc.local
[Service]
Type=forking
ExecStart=/etc/rc.local
TimeoutSec=0
StandardOutput=tty
RemainAfterExit=yes
SysVStartPriority=99
[Install]
WantedBy=multi-user.target
```

Включаем и запускаем службу *rc-local*:

```
sudo systemctl enable rc-local.service
sudo systemctl start rc-local.service
```

Теперь файл *rc.local* будет выполняться при загрузке системы на ArchLinux, запуская при этом процесс сбора и отправки информации на сервер.

Сервер

Процесс настройки будет одинаков для ОС Ubuntu начиная с 20.4 LTS.

Текущая установка произведена на версию 24.04 LTS, работоспособность проверялась также на Read Hat Enterprise Linux 9.4.

Максимально старался автоматизировать процесс установки docker и его зависимостей. В папке docker дипломного репозитория располагается скрипт install.sh полностью автоматизирующий процесс развертывания всего комплекса, что называется под-ключ, это и настроенные связи и пароли и токены между компонентами. Можно задать свои параметры:

```
#!/bin/bash
USERNAME=admin
PASSWORD=students
DOMEN=domen.ru
```

```
ORG=IoT
BUCKET=IoT
INFLUXDB_TOKEN="kFhczFje8dRm2SXK1V9Ds7xpcJTr6wVUS881KQoUQWE-QAfcg-S-
6j1FvFiSvWW0wTPlmWHCvXf_JU1hRx5rZg=="
```

Как видно из конфига, по умолчанию на всех службах логин: admin и пароль: students.

В случае отсутствия docker на компьютере, он будет установлен с офсайта.

Процесс развертывания вместе с загрузкой контейнеров займет не более 10 минут.

Первое что мы захотим проверить, это приходят ли данные на сервер с датчика.

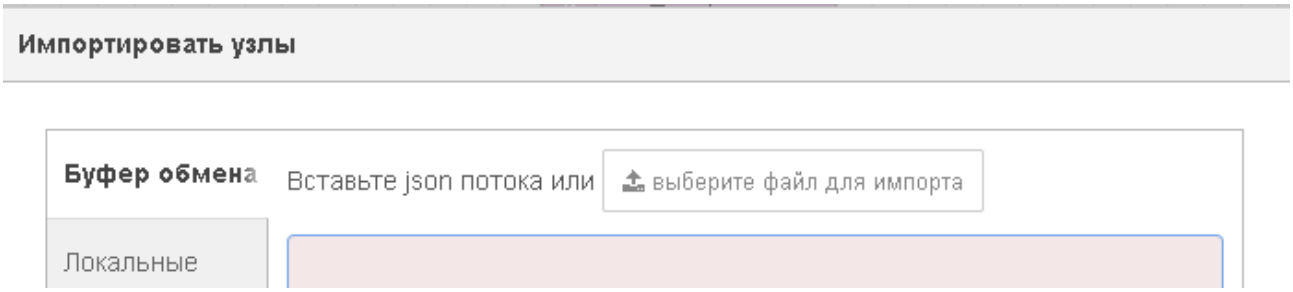
Mosquitto

Mqtt брокер, который принимает все отправляемые на него пакеты от паблишеров и отдает информацию подписчикам. Что-то очень похожее мы реализовали средствами юникса на свадьбе **bsec** и **mqtt** через «именованный канал». Но, тут реализован механизм QoS (quality of service) и если его значение выше 1, тогда все подписчики гарантированно получат пакеты. В случае нуля, если один подписчик забрал пакет, то остальные не получат его. Поспешим, а то все пакеты разберут.

NodeRed

Одним из самых быстрых способов проверки mqtt-трафика является отладка с помощью NodeRed.

Сохранил JSON файл с параметрами потока, настроенными на топики, которые посылаются с датчика. Данный файл достаточно импортировать через меню:



После загрузки json-потока, отобразятся 12 подписчиков, каждый на свою тему и у всех есть вывод в debug (Приложение 6). На этом роль Node-Red в данном проекте заканчивается, хотя его можно и использовать как mqtt-брокер, пересылая пакеты дальше, но это слишком расточительно в плане ресурсов вычислительных мощностей.

Telegraf

Если node-red нам нужен только для отладки. И в дальнейшей схеме может быть исключен, то telegraf является связующим звеном между брокером и базой данных.

Он подписывается на все топики брокера mosquitto? используя заданные в настройках авторотационные данные и, передает их дальше, в нашем случае на InfluxDB

InfluxDB

Если вышеперечисленные службы можно заменить другими каналами или протоколами доставки, например, доставлять показания по почте или через телеграмм, то от этого ценность самой информации не измениться, другое дело с хранилищем данных. Если хранить временные показания в реляционных базах даных, то это будет не эффективно. Поэтому ключевым игроком в нашей связке выступает СУБД времянных рядов Influx-DB, это не просто система управления базой данных, но и визуализация, которая может заменить в простых случаях Grafana.

Немного придется понастроить вручную, возможно в будущем найду способ автоматизировать и этот процесс.

Для начала в веб-интерфейсе InfluxDB необходимо выбрать корзину (bucket) в которую скидывает информация с помощью Telegraf.

В данной корзине можно выбрать те параметры и топики, которые мы хотим отслеживать (Приложение 7)

В результате сформируется запрос на языке Flux:

```
from(bucket: "IoT")
  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
  |> filter(fn: (r) => r["topic"] == "breath_voc_equivalent" or
r["topic"] == "bsec_status" or r["topic"] == "co2_equivalent" or
r["topic"] == "gas" or r["topic"] == "humidity" or r["topic"] == "iaq" or
r["topic"] == "iaq_accuracy" or r["topic"] == "pressure" or r["topic"] ==
"raw_humidity" or r["topic"] == "raw_temperature" or r["topic"] ==
"static_iaq" or r["topic"] == "temperature")
  |> filter(fn: (r) => r["_field"] == "value")
  |> filter(fn: (r) => r["_measurement"] == "mqtt_consumer")
  |> filter(fn: (r) => r["host"] == "cd8fb61b1115")
  |> aggregateWindow(every: v.windowPeriod, fn: mean, createEmpty: false)
  |> yield(name: "mean")
```

На основе данных запросов, прямо внутри веб-интерфейса InfluxDB можно строить визуализацию. На рисунке (Приложение 8) видна визуализация средствами InfluxDB, передаваемых с датчика двенадцатью топиками в виде спидометров.

Дашборд сохранен в формате JSON. Посмотрим, сможет ли визуализация InfluxDB заменить Grafana.

Grafana

Графана - это система визуализации и анализа данных. Она поддерживает огромное количество типов данных и хранилищ, как встроенными средствами, так и с помощью сторонних плагинов.

Необходимо с помощью авторизационных данных (token) подсоединиться к InfluxDB. И в разделе Dashboards создать новый. Можно создавать множество дашбордов на каждый случай жизни, причем можно использовать всевозможные комбинации одних и тех же показателей с разными системами представления.

При создании нового дашборда, отобразится список имеющихся соединений с хранилищами, в нашем случае это InfluxDB

Ранее созданный в InfluxDB запрос можно использовать в дашборде.

На рисунке (Приложение 9) показаны настройки визуализации влажности. В отличие от InfluxDB, настроек больше и они гибче. Задал цветовую индикацию для пороговых значений. В случае с влажностью, если она выше 70%, то спидометр будет синим, 40% и выше зеленым, если ниже 25% то сухой климат будет отображается красным цветом.

Компактнее можно расположить элементы на дашборде, тут элементы делаются меньше, в отличие от InfluxDB, где существует порог сжатия спидометров.

Удобная функция, трансформации данных (Приложение 10), которая чем-то напоминает библиотеку pandas в пайтоне.

Основной дашборд получился очень наглядным (Приложение 11). На нем сверху расположил ряд основных метрик: температура, влажность, давление, IAQ, bVOC, статус программы.

Посредине расположил общую диаграмму, на которой видны графики сразу всех показаний.

В нижнем ряду: необработанная температура, необработанная влажность, эквивалента CO2, статический IAQ, сопротивление воздуха и точность измерение IAQ.

В результате с одного датчика снимается 12 показаний 6 из которых являются метриками, остальные служебными показаниями. И это не придел, изучая программное

обеспечение, обнаружил порядка 20 параметров, которые можно получать с датчика, в основном это служебные данные. В итоге мы за 350 рублей получили датчик с 6 метриками.

Заключение

Дипломная работа будет не полной, если не провести тестирование нашего датчика, немного поменяю местами спидометры, чтобы наверху были интересные данные (Приложение 12).

Обращаем внимание на красные спидометры, это форточка была закрыта, теперь откроем ее и дадим проветрится 5 минут.

Прошло ровно 10 минут (Приложение 13) и все метрики загазованности уменьшились и ушли из красной зоны, а CO2 перешел в зеленую. Это ли не показатель, важности регулярного проветривания помещения.

Продолжаем эксперименты, форточка открыта, показания стабилизировались. Поднесу открытый пузырек со спиртом к датчику и посмотрим на график (Приложение 14). Моментальный зашкал, не успел я поднести маленький пузырек со спиртом к датчику, как он мгновенно отреагировал чумовыми показателями.

Были мысли попробовать с керосином и уайт-спиритом, но это плохая идея, если такая реакция на спирт.

Показания быстро пришли в норму. Пробуем то, подо что датчик заточен – дыхание, вернее всего один небольшой, но с задержкой (Приложение 15). В итоге сделал 3 выдоха, с перерывом в пару минут, т.к. от первого был в шоке и не успел заскринить. На графике отчетливо видно в 10:20 ракетный взлет от спирта. С 10:27 три взлета поменьше. Но, две метрики *iaq* и *static_iaq* перешли красную зону и вышли в сиреневую (или фиолетовую). На всякий случай кидаю еще раз ссылку на таблицу, в соответствии с которой настроил цветовой фон (Приложение 3, Рисунок 1).

Заключительный тест со спичкой (Приложение 16). В 10:40 наблюдается небольшой импульс сравнимый с погрешностью. Следовательно, данный датчик не может служить детектором дыма и уж тем более системой пожаротушения. Лучше докупить дешевых специализированных китайских датчиков дыма. Приобрету себе один с альфа-частицами для теста радиометра и для реальной защиты от дыма.

Данная система будет не совсем полноценной без доступа в интернет, который организован к серверу с помощью nginx. Прикладываю рабочий конфиг (Приложение 17), который был опробован на курсе по распределенным сетям (домен изменен в целях безопасности).

На случай проблем с сервером, через nginx можно будет достучаться до самописного web-сервера на языке СИ.

Ссылки на мой GitHub

В данном разделе дана ссылка на корень дипломного проекта.

И ссылки на составные части внутри проекта с их описаниями.

1. Корень дипломного проекта allseenn/dipiot (github.com)
2. Мое решение на базе BSEC dipiot/bsec at main · allseenn/dipiot (github.com)
3. Закрытые дистрибутивы BOSH dipiot/distr at main · allseenn/dipiot (github.com)
4. Скрипт автоустановки docker dipiot/docker at main · allseenn/dipiot (github.com)
5. Документация BOSH BME680 dipiot/docs at main · allseenn/dipiot (github.com)
6. Мой MQTT-publisher dipiot/mqtt at main · allseenn/dipiot (github.com)
7. Python реализация драйвера dipiot/python at main · allseenn/dipiot (github.com)
8. Мое решение на базе open API dipiot/sense at main · allseenn/dipiot (github.com)
9. Мой веб-сервер на СИ dipiot/web at main · allseenn/dipiot (github.com)

Список литературы

1. BME680 – Datasheet 1.9 02-2024
2. BME680 – Datasheet 1.3 07-2019
3. BSEC Integration Guideline
4. BSEC Binary Size Information
5. BME68x - Shipment packaging details
6. BME680 – Application Note
7. BME680 Integrated Environmental Unit
8. BME680: Handling, Soldering and Mounting Instructions
9. BME680 Shuttle board 3.0 flyer
10. Release Notes 2.5.0.2

Приложение 1

Рисунок 1. Плата датчика BME680

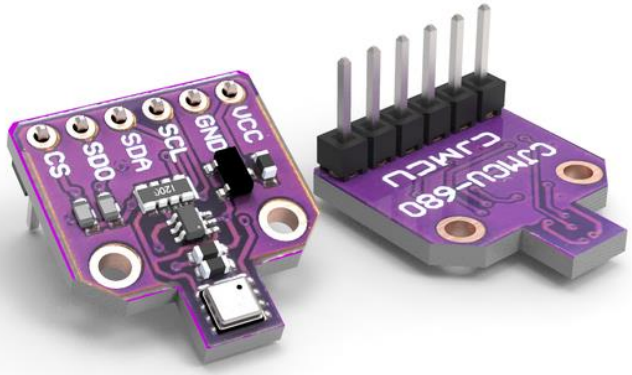


Рисунок 2. Чип-сенсор BME680

Приложение 2

Рисунок 1. Комбинированная плата (слева): сенсор влажности и давления

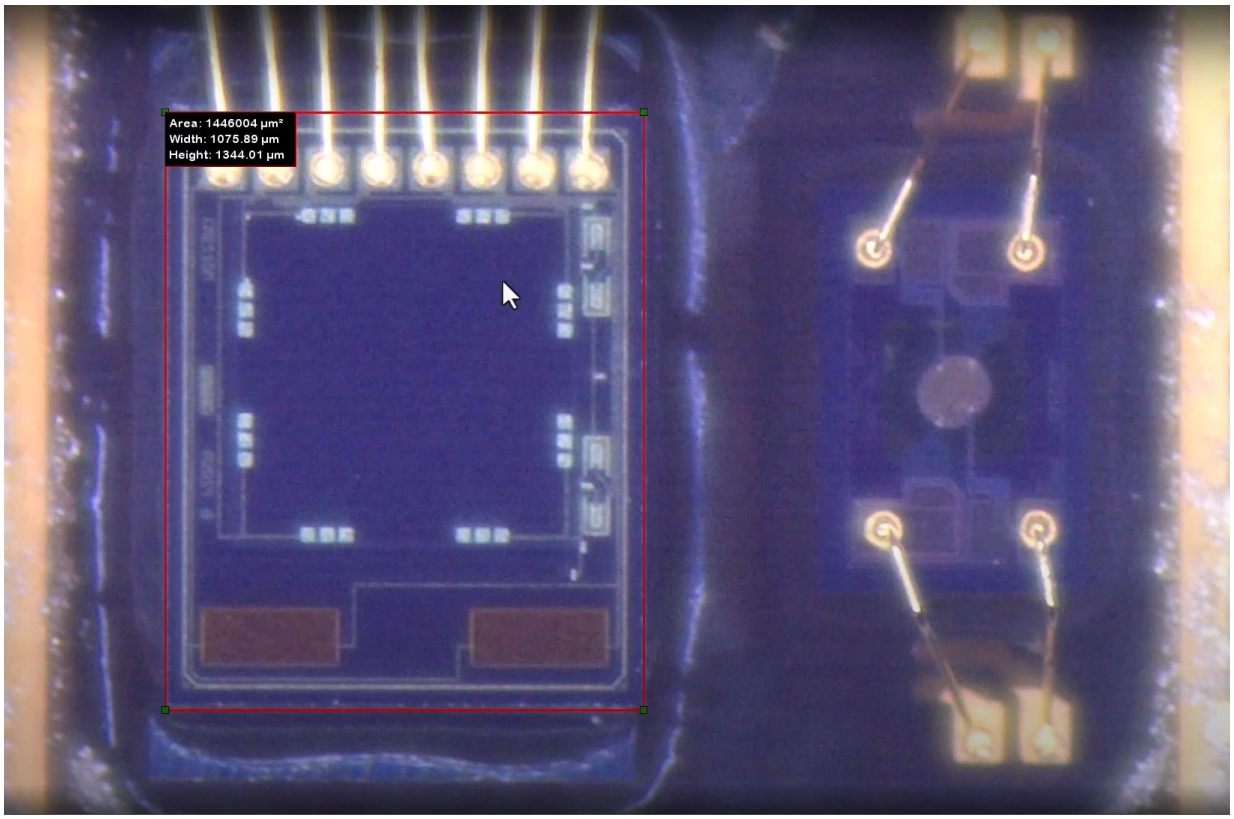


Рисунок 2. Плата загазованности (справа)

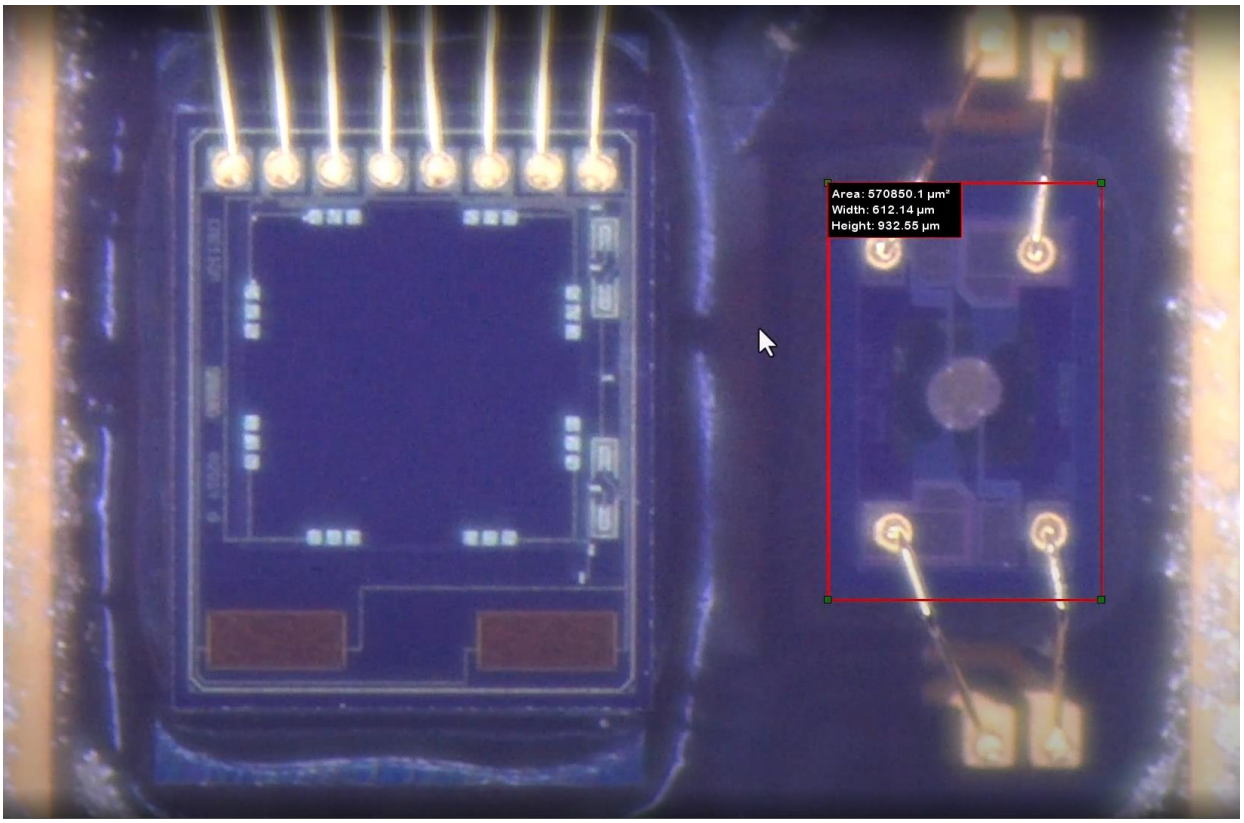


Рисунок 3. Плата загазованности увеличенная

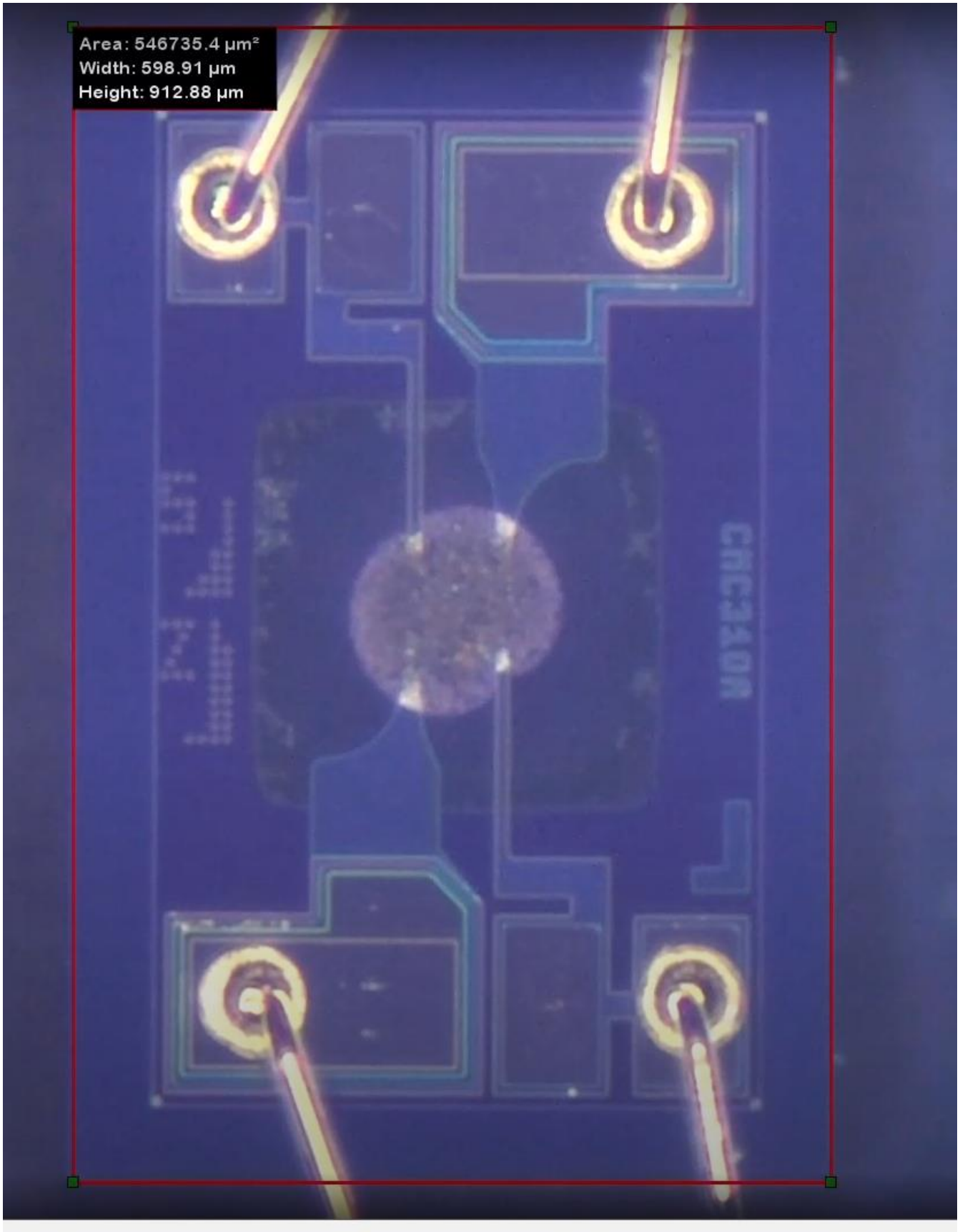


Рисунок 4. Диаметр капли оксида железа 0.17мм

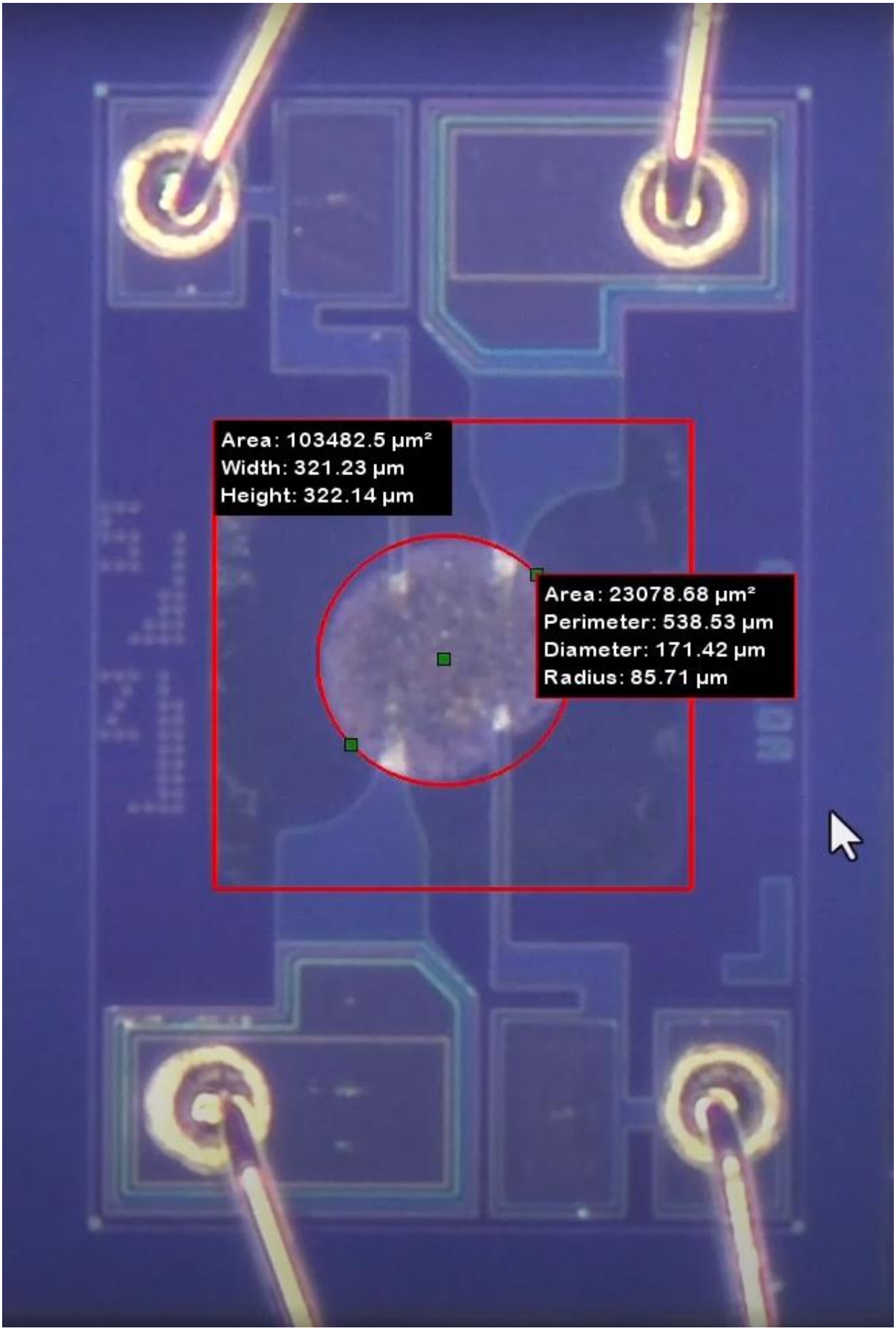


Рисунок 5. Нагревательный элемент под оксидом железа

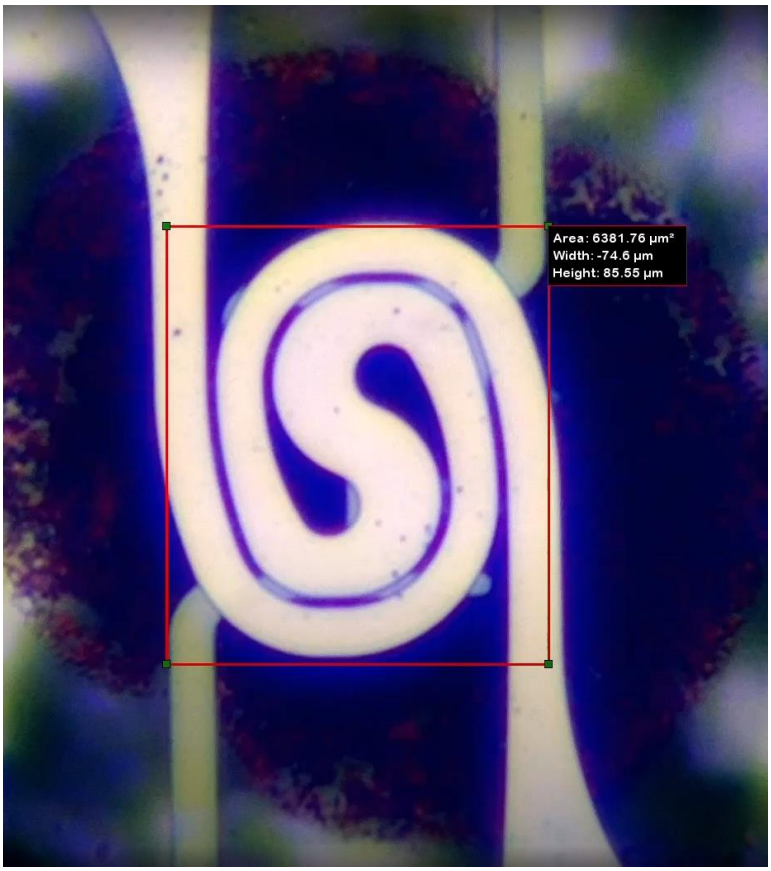
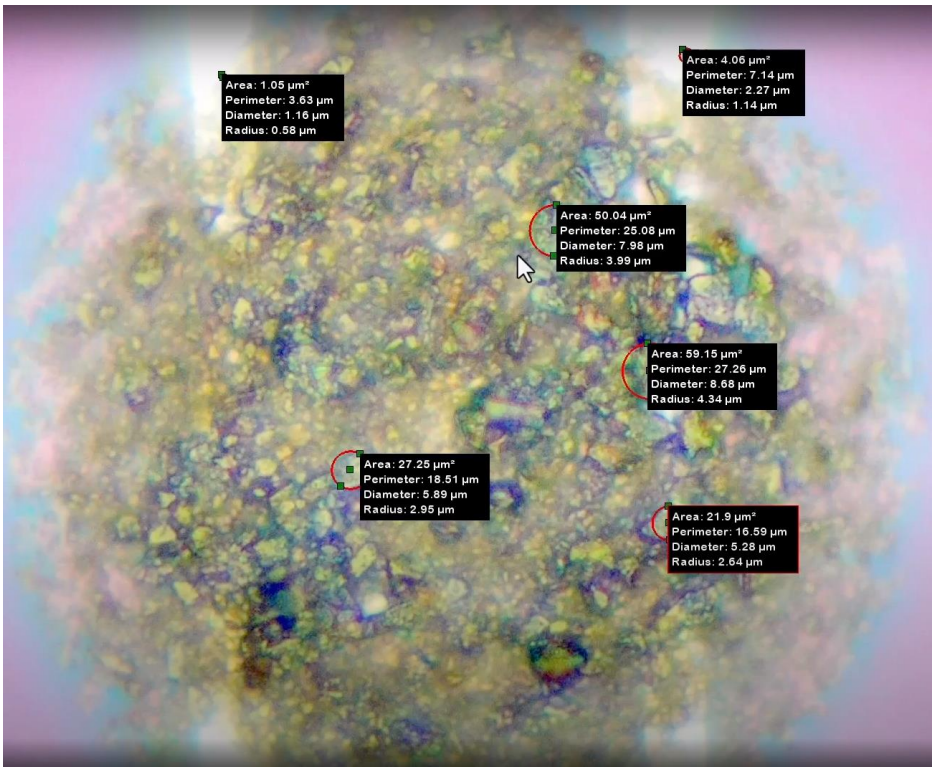


Рисунок 6. Частицы оксида железа диаметром 1-8 микрон



Приложение 3

Рисунок 1. Таблица значений индекса IAQ

IAQ Index	Air Quality	Impact (long-term exposure)	Suggested action
0 – 50	Excellent	Pure air; best for well-being	No measures needed
51 – 100	Good	No irritation or impact on well-being	No measures needed
101 – 150	Lightly polluted	Reduction of well-being possible	Ventilation suggested
151 – 200	Moderately polluted	More significant irritation possible	Increase ventilation with clean air
201 – 250	Heavily polluted	Exposition might lead to effects like headache depending on type of VOCs	optimize ventilation
251 – 350	Severely polluted	More severe health issue possible if harmful VOC present	Contamination should be identified if level is reached even w/o presence of people; maximize ventilation & reduce attendance
> 351	Extremely polluted	Headaches, additional neurotoxic effects possible	Contamination needs to be identified; avoid presence in room and maximize ventilation

Приложение 4

Рисунок 1. Коммутация к GPIO контактам

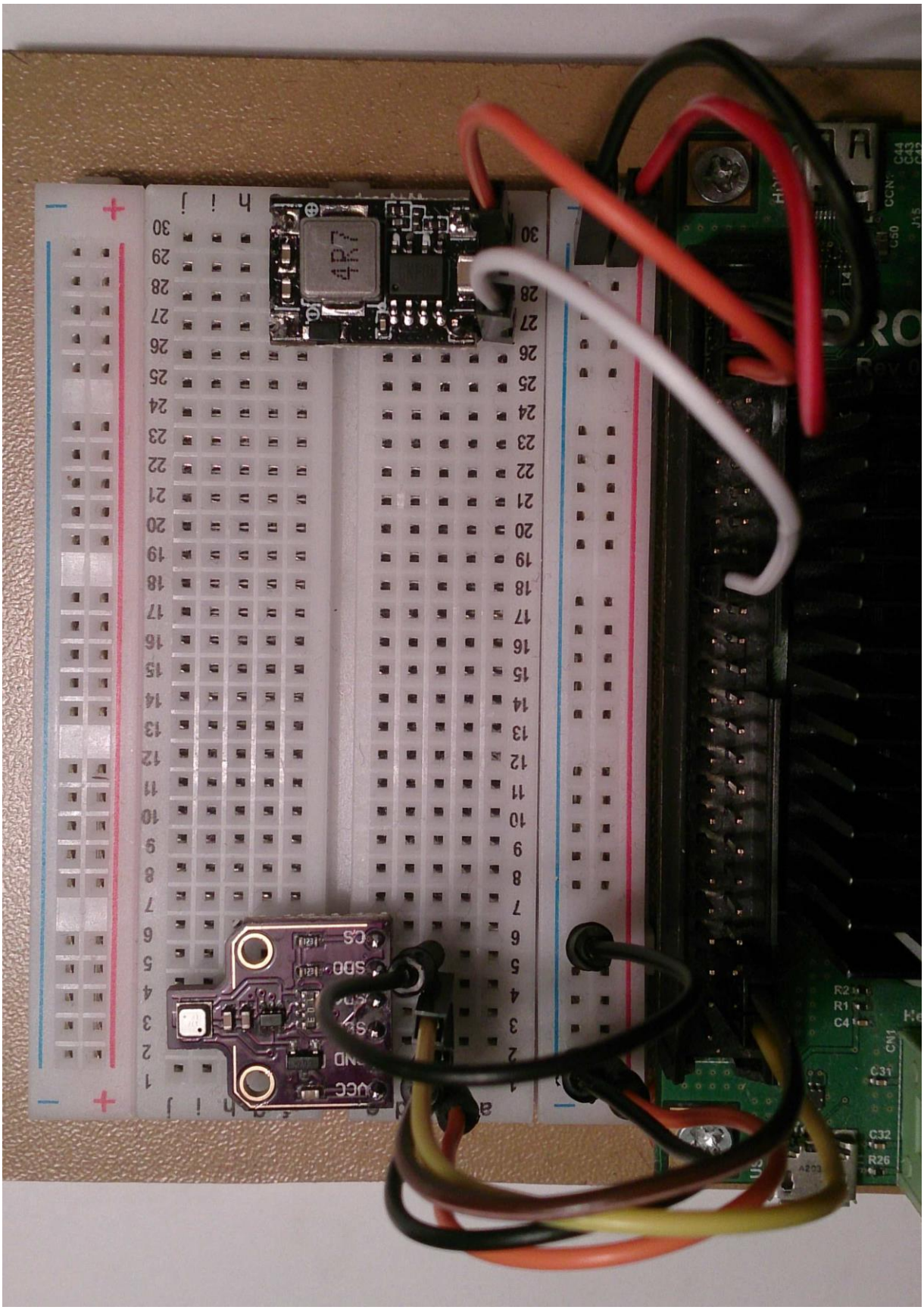
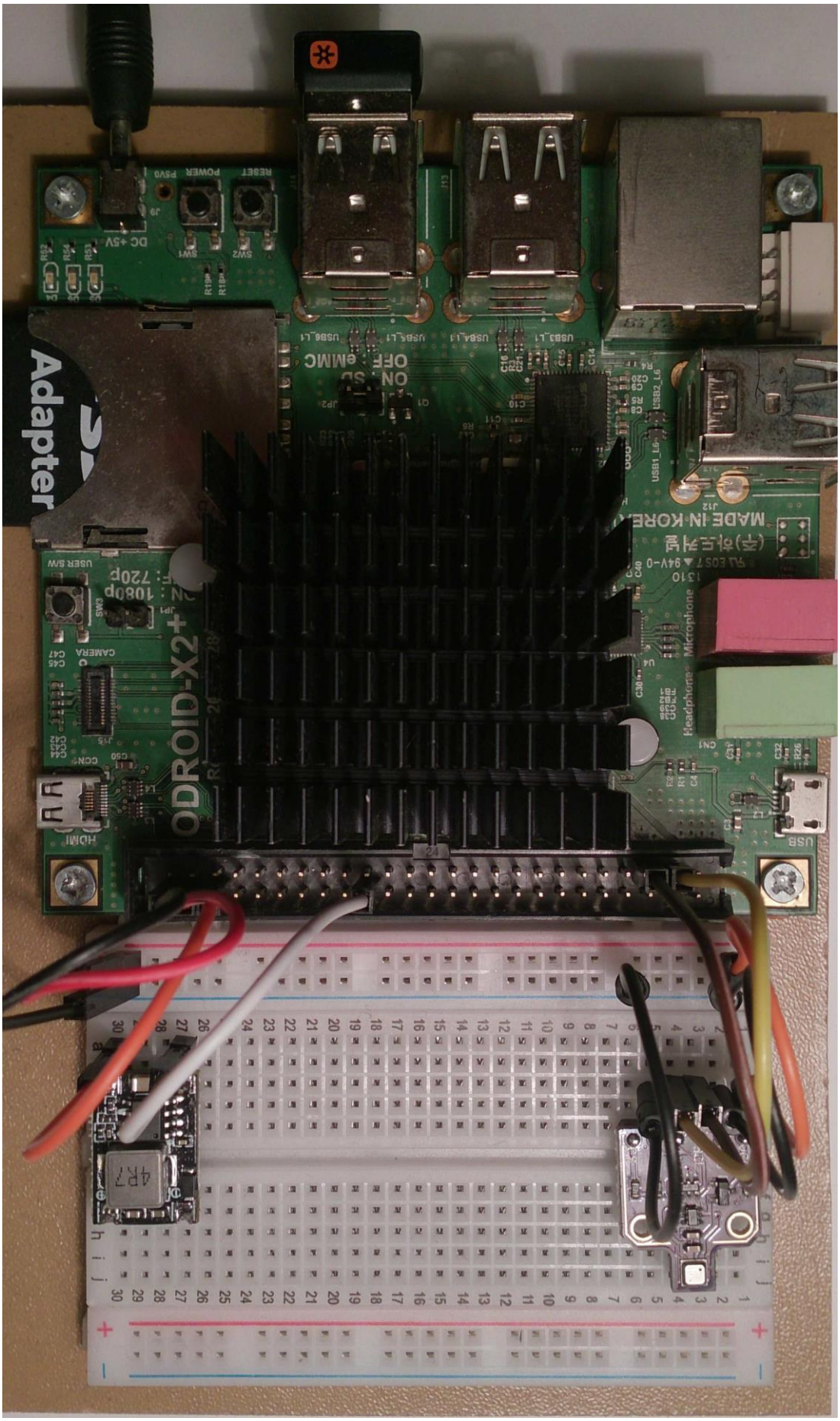
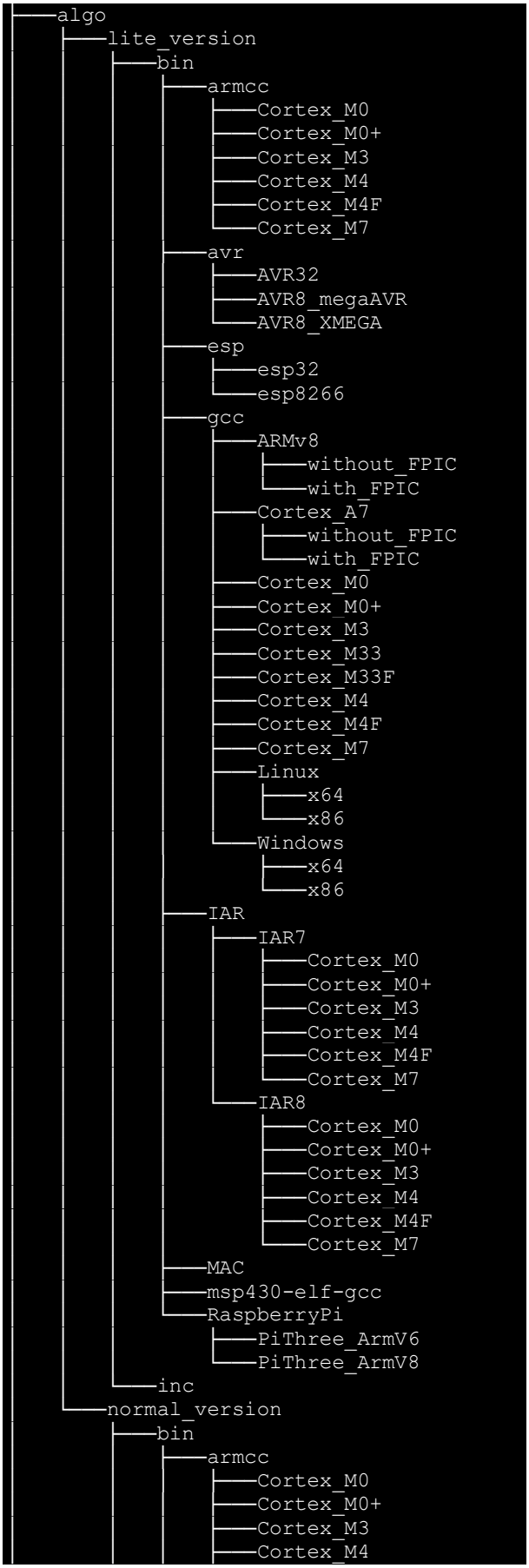


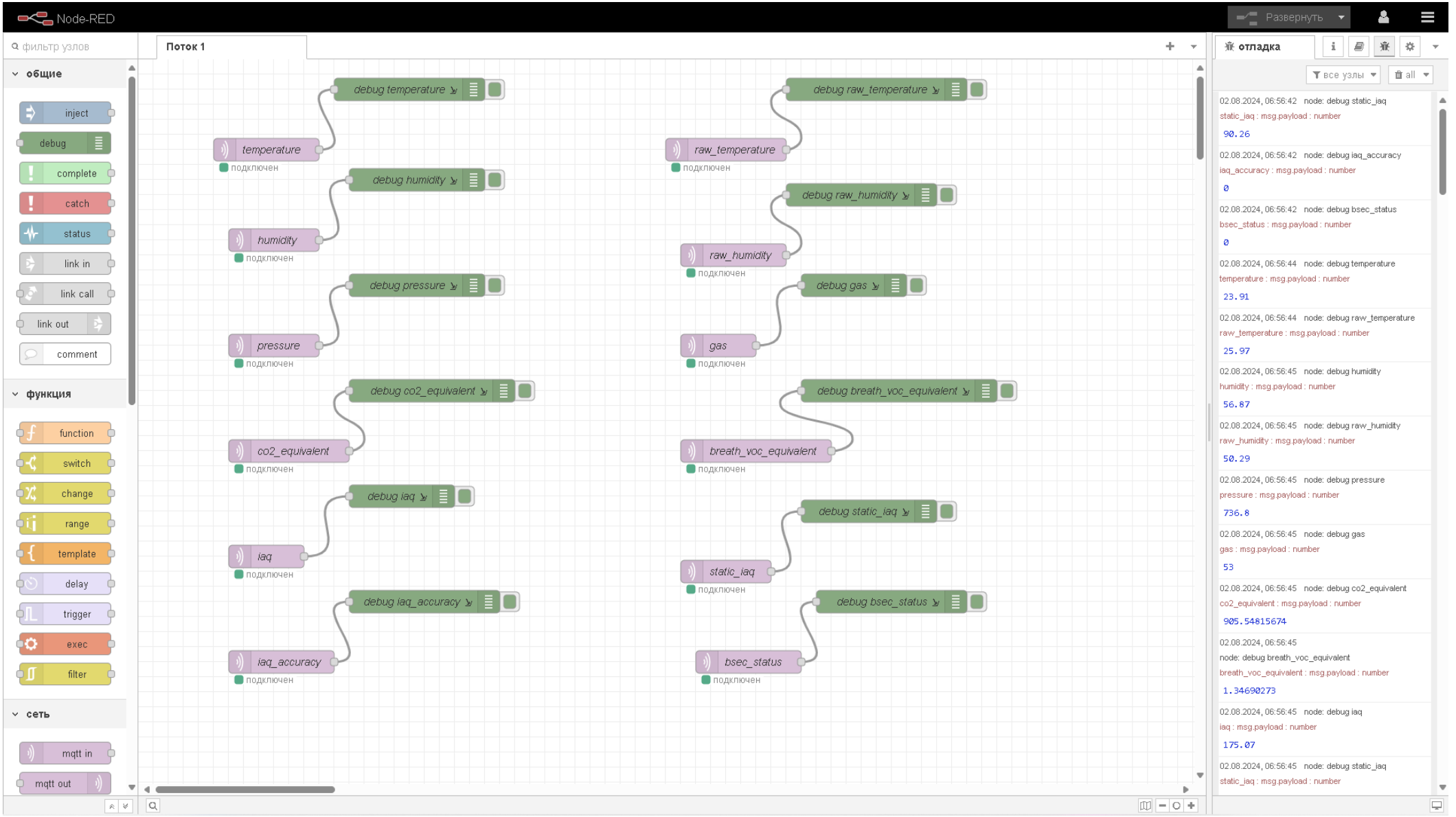
Рисунок 2. ODROID-X2 с конвертером CN3903 и подключенным BME680



Приложение 5

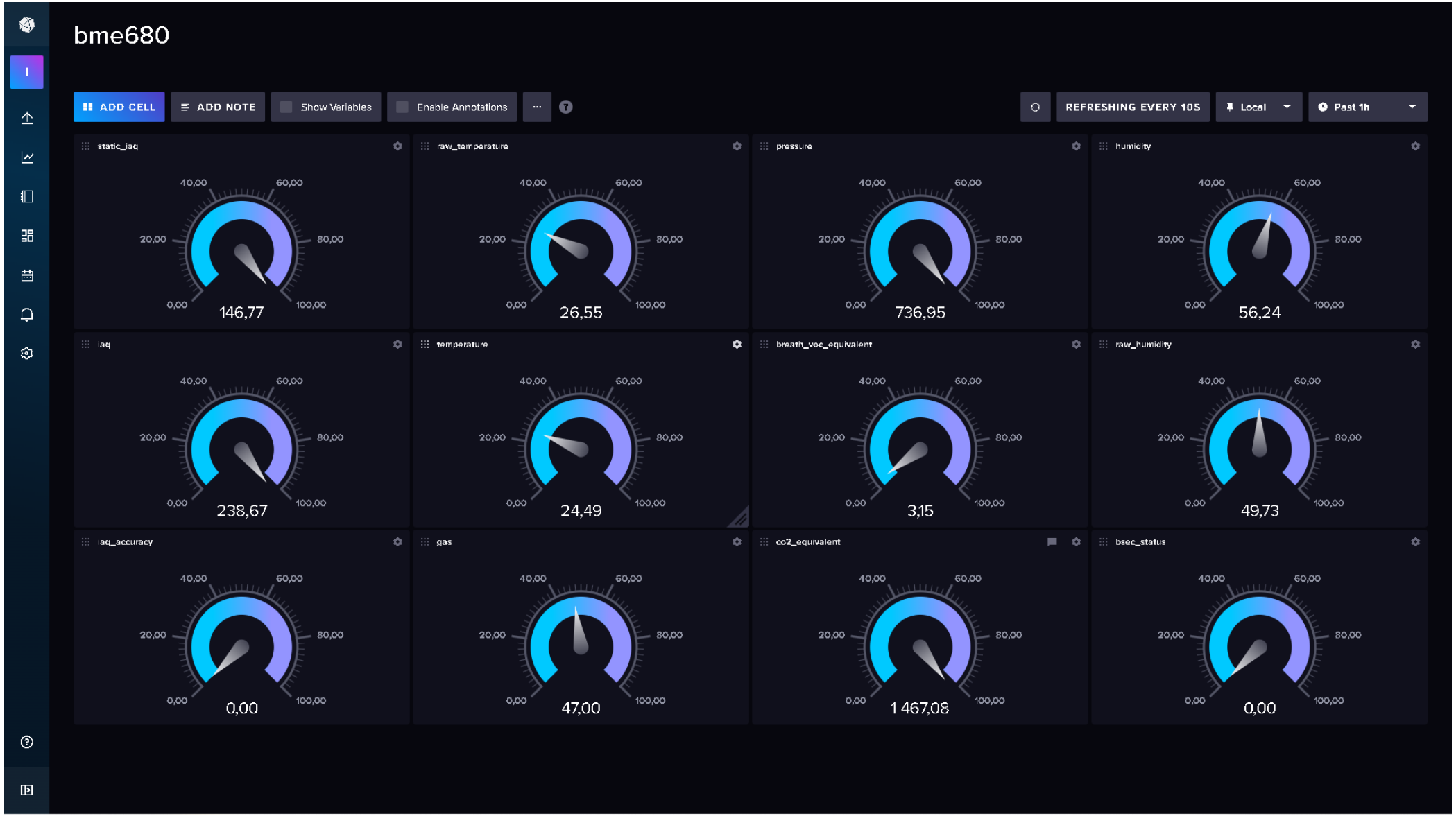


Приложение 6

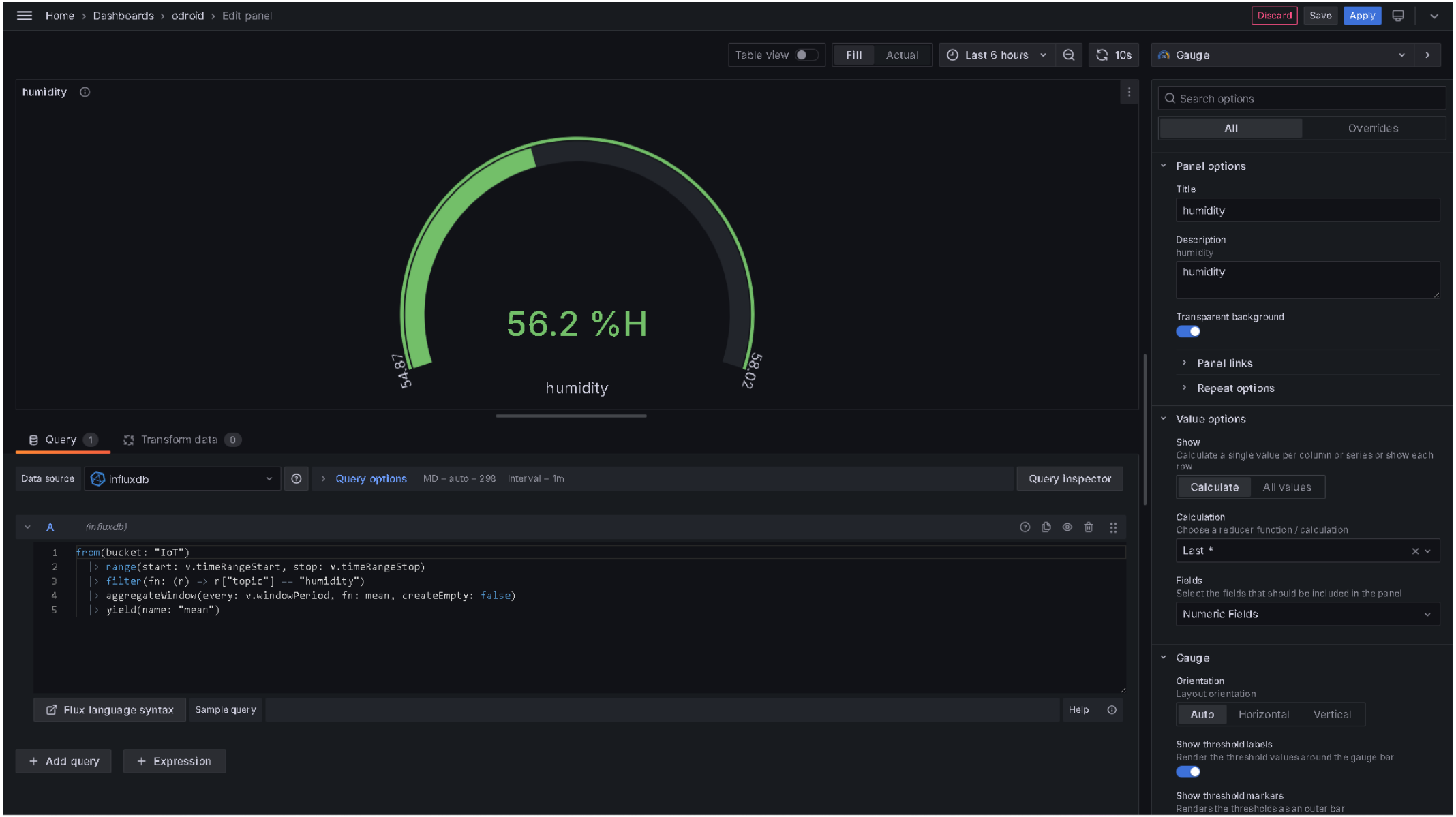


Приложение 7

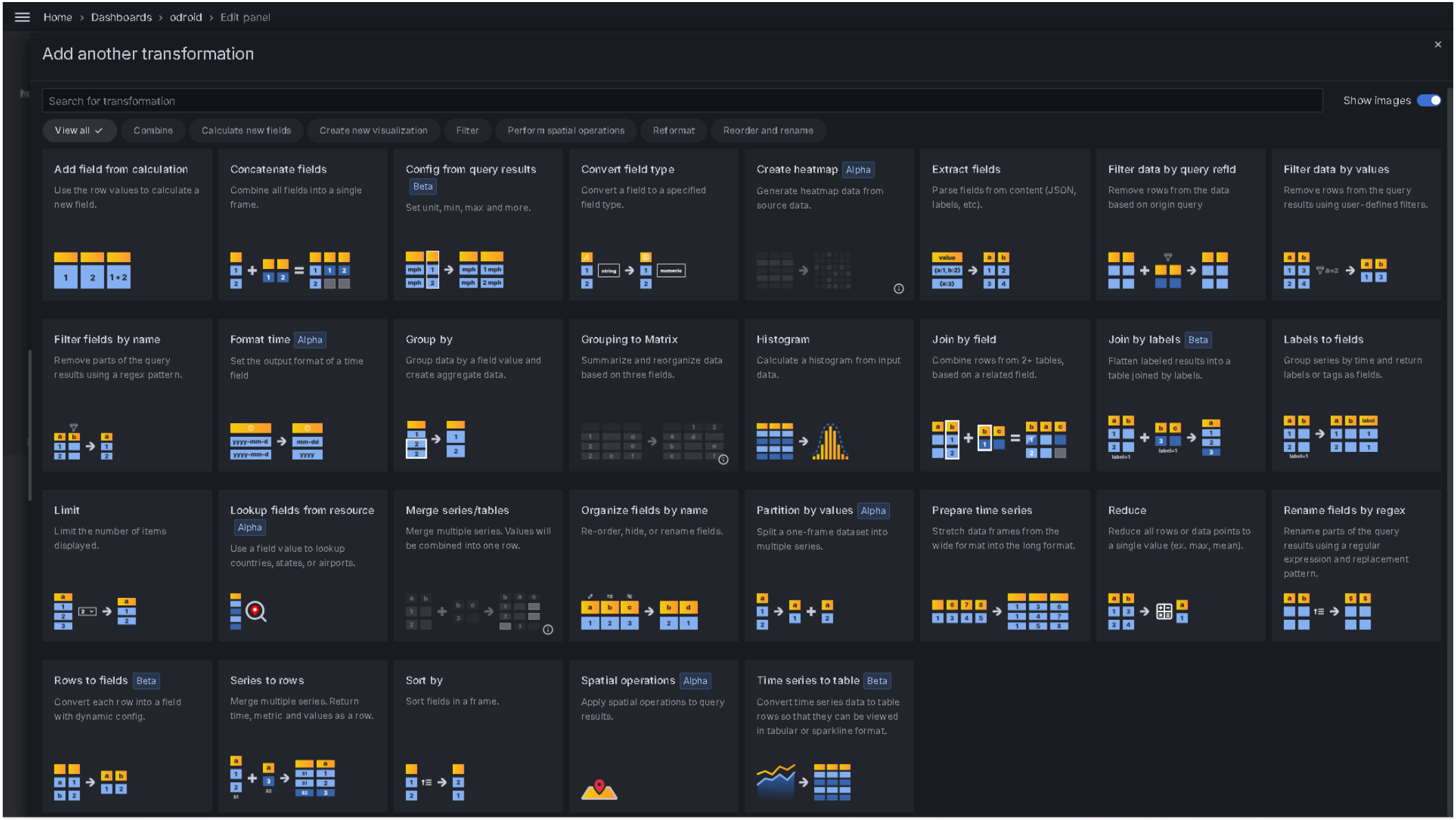
Приложение 8



Приложение 9



Приложение 10



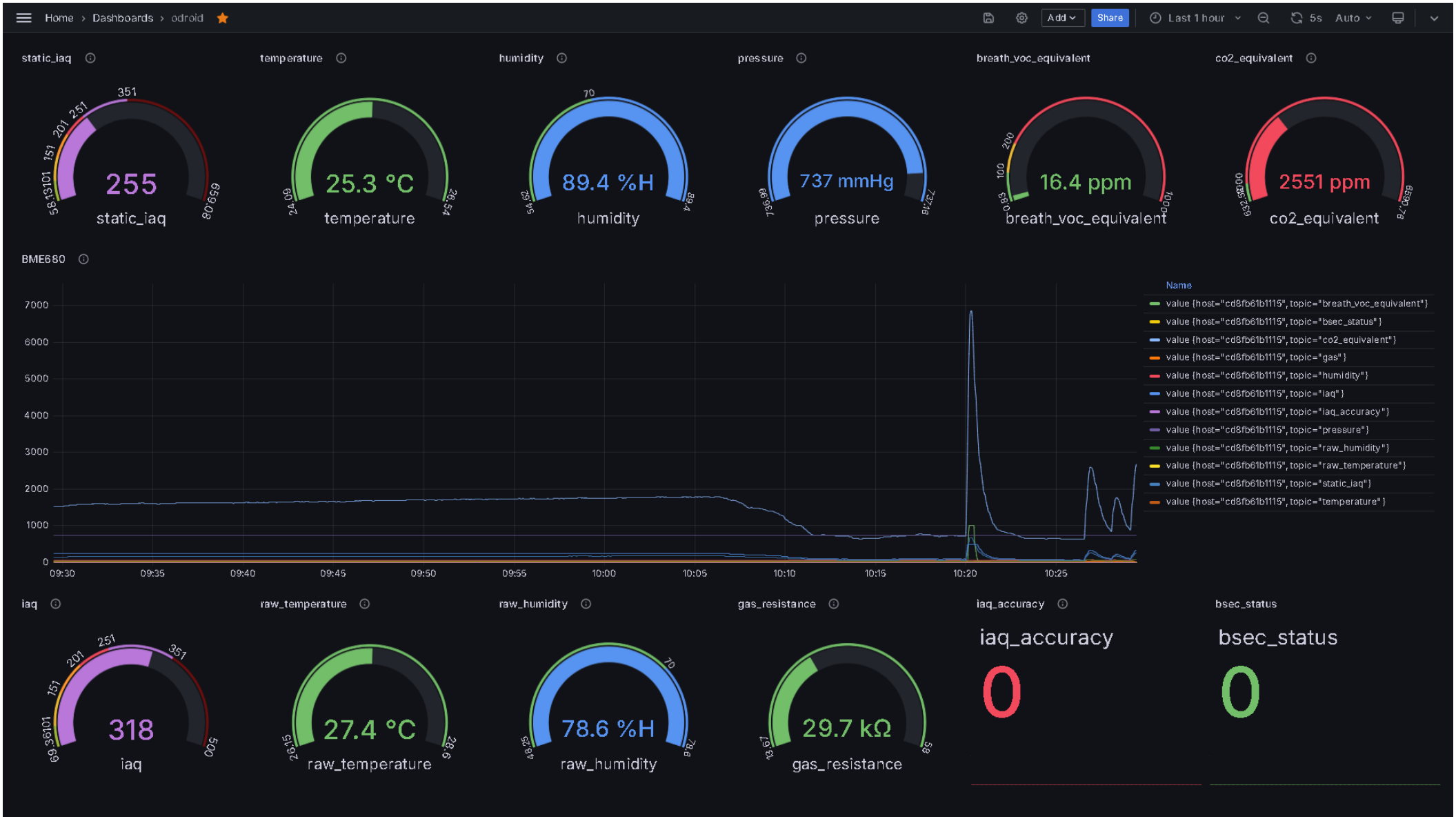
Приложение 11

Приложение 12

Приложение 13

Приложение 14

Приложение 15



Приложение 16

Приложение 17

```
server {
    listen 443 ssl;
    server_name mosquitto-rrg-5471.gb-iot.ru;
    location / {
     proxy_http_version 1.1;
     proxy_pass http://192.168.1.8:8081;
     proxy_set_header Upgrade $http_upgrade;
     proxy_set_header Connection "upgrade";
     proxy_set_header Host $host;
  }

    ssl_certificate /etc/letsencrypt/live/gb-
iot.ru/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/gb-
iot.ru/privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;
}
server {
    listen 443 ssl;
    server_name influxdb-rrg-5471.gb-iot.ru;
    ssl_certificate /etc/letsencrypt/live/gb-
iot.ru/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/gb-
iot.ru/privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;
    location / {
      proxy_set_header X-Forwarded-For $remote_addr;
      proxy_pass http://192.168.1.8:8086;
      proxy_set_header Host $host;
      proxy_set_header X-Real-IP $remote_addr;
    }
}
```

```
server {
    listen 443 ssl;
    server_name grafana-rrg-5471.gb-iot.ru;
    ssl_certificate /etc/letsencrypt/live/gb-
iot.ru/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/gb-
iot.ru/privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;
    location / {
      proxy_set_header X-Forwarded-For $remote_addr;
      proxy_pass http://192.168.1.8:3000;
      proxy_set_header Host $host;
      proxy_set_header X-Real-IP $remote_addr;
    }
}
server {
    listen 443 ssl;
    server_name nodered-rrg-5471.gb-iot.ru;
    ssl_certificate /etc/letsencrypt/live/gb-
iot.ru/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/gb-
iot.ru/privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;
  location / {
    proxy_pass http://192.168.1.8:1880;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
  }
}
```