**ГБОУ города Москвы «Школа № 444»**

# СОЗДАНИЕ ПРОГРАММНО-АППАРАТНОГО КОМПЛЕКСА ДЛЯ МОНИТОРИНГА НАЛИЧИЯ ВОДЫ В КУЛЕРАХ

**Работу выполнили:**

Карак Юлия Алексеевна,
ученица 11 «И» класса

Рябова Екатерина Николаевна,
ученица 11 «В» класса

**Научный руководитель:**

Вербов Евгений Николаевич,
педагог дополнительного образования

**Москва, 2024**

## Оглавление

# Введение

На территории школы установлено большое количество кулеров с питьевой водой (см. Рисунок 1. Внешний вид кулеров с питьевой водой, установленных на территории школы). Это очень удобно для школьников, так как позволяет не приносить из дома воду в бутылках. Школьников много, кулеры популярны, поэтому иногда складывается ситуация, когда вода в кулерах заканчивается, а сотрудники администрации школы не сразу это замечают. Даже если рядом с кулером находится запасная бутыль с водой, то школьники младших классов самостоятельно не могут заменить пустую бутыль на полную. Подобное несовершенство натолкнуло на идею проекта – нужно создать систему, которая бы отслеживала заполненность бутылей в кулерах и при отсутствии в них воды (или немного заранее) информировала бы об этом сотрудников школы, ответственных за обслуживание кулеров.

Рисунок 1. Внешний вид кулеров с питьевой водой, установленных на территории школы

## Цель проекта

Целью проекта является разработка устройства для мониторинга наличия воды в бутылях, установленных на кулерах на территории школы, разработка программного обеспечения для сбора информации с устройств мониторинга и разработка программного обеспечения для представления собранной информации пользователям.

## Задачи проекта

1. Выбор технологии определения наличия воды в бутыли.
2. Разработка общей архитектуры программно-аппаратного комплекса мониторинга и отображения.
3. Проектирование и разработка устройства мониторинга наличия воды в бутыли.
4. Проектирование и разработка программного обеспечения для сбора и отображения информации с устройств мониторинга.
   4.1. Разработка подсистемы сбора информации с устройств мониторинга.
   4.2. Разработка подсистемы для предоставления информации пользователю.

## Выбор технологии определения наличия воды в бутыли

В качестве возможных вариантов решения задачи определения наличия воды в бутылях рассматривались следующие:

- Использование аппаратного датчика, показывающего вес кулера с установленной бутылью. Вес кулера с пустой бутылью может быть измерен заранее. В процессе мониторинга при достижении этого веса система сможет принять решение об отсутствии воды.
- Использование ёмкостного датчика, измеряющего изменение диэлектрической проницаемости среды рядом с датчиком. Датчик, закреплённый у основания бутыли будет выдавать разные значения при

наличии или отсутствии воды, так как диэлектрические проницаемости этих сред отличаются.

- Использование видеокамеры, направленной на бутыль, и программного алгоритма распознавания границы воздуха и воды. Прозрачности воздуха и воды отличаются, место соприкосновения можно искать с помощью алгоритма выделения контуров на изображении, получаемом с камеры.

Сравнительный анализ перечисленных вариантов показал, что использование решения с ёмкостным датчиком имеет следующие преимущества: простота и надёжность, достаточная точность, доступность и низкая стоимость датчиков. Ниже приведены изображения ёмкостных датчиков разного вида (см. Рисунок 2. Примеры ёмкостных датчиков).

Рисунок 2. Примеры ёмкостных датчиков разных производителей

## Разработка общей архитектуры программно-аппаратного комплекса

Комплекс должен состоять из устройств мониторинга, закреплённых на бутылях с водой, программного обеспечения для получения данных с устройств мониторинга и программного обеспечения для предоставления пользователю информации, полученной с устройств мониторинга. Для простоты внедрения и уменьшения стоимости необходимо использовать готовые программные и аппаратные модули и задействовать технологические решения, которые уже используются на территории школы. Исходя из перечисленных выше требований и задач, которые необходимо решить в рамках работы над проектом,

разработана архитектура программно-аппаратного комплекса, приведённая на схеме ниже (см. Рисунок 3. Структурная схема программно-аппаратного комплекса).

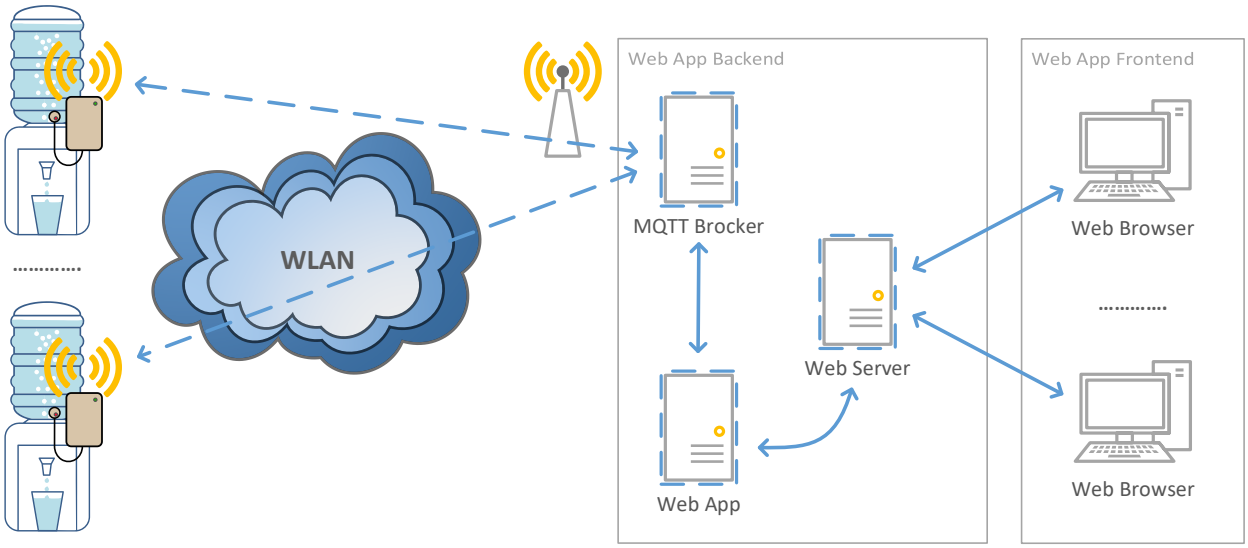


Рисунок 3. Структурная схема программно-аппаратного комплекса

Каждый ёмкостный датчик, закреплённый на бутыли, подключается к отдельному микроконтроллеру на базе ESP32. Семейство ESP32 выбрано из-за наличия в этих микроконтроллерах встроенного блока, отвечающего за работу с Wi-Fi, большой выбор микроконтроллеров в семействе, популярность у специалистов, низкую стоимость и доступность [1].

Микроконтроллеры через WLAN подключаются к сети, в которой находится серверная часть комплекса. Серверная часть комплекса состоит из сервера MQTT, отвечающего за получение и хранение сообщений от микроконтроллера, серверной части веб-приложения, отвечающего за бизнес-логику, и веб-сервера, отвечающего за доступ пользователя к клиентской части веб-приложения. Работа пользователя с комплексом выполняется через веб-браузер, в котором выполняется клиентская часть веб-приложения.

Микроконтроллер в цикле опрашивает датчик и отправляет полученное значение на сервер MQTT (взаимодействие выполняется по протоколу MQTT). Серверная часть веб-приложения подписывается на обновление данных в сервере MQTT (взаимодействие выполняется по протоколу MQTT). Клиентская часть веб-приложения, выполняющаяся в веб-браузере пользователя, подписывается на получение сообщений от серверной части веб-приложения

(взаимодействие выполняется по HTTP на основе Server Side Events). При получении новых данных, серверная часть веб-приложения отправляет их клиентской части.

Серверная часть веб-приложения представляет собой приложение, разработанное на языке программирования Python с использованием фреймворка FastAPI[1]. Клиентская часть веб-приложения представляет собой приложение, разработанное на языке программирования JavaScript с использованием фреймворка Vue.JS[2]. В качестве сервера MQTT используется приложение с открытым исходным кодом Eclipse Mosquitto[3] со свободной лицензией EPL. В качестве веб-сервера используется приложение с открытым исходным кодом NGINX[4] со свободной лицензией на основе BSD.

## Проектирование и разработка устройства для мониторинга наличия воды в бутыли

Устройство состоит из микропроцессорного модуля из семейства ESP32, к которому подключён ёмкостный датчик XKC-Y25 [2]. При наличии жидкости вблизи зоны чувствительности датчика на его сигнальном выходе устанавливается уровень, соответствующий логической единице, при отсутствии – нулю (см. Рисунок 4. Схема подключения датчика XKC-Y25- из документации производителя [2]). Питание на схему подаётся через порт USB, датчик XKC-Y25 рассчитан на работу с напряжением 5В, а ESP32 – с напряжением 3.3В, поэтому в схему дополнительно добавлен преобразователь уровней.

[1] https://fastapi.tiangolo.com/

[2] https://ru.vuejs.org/

[3] https://mosquitto.org/

[4] https://nginx.org/ru/

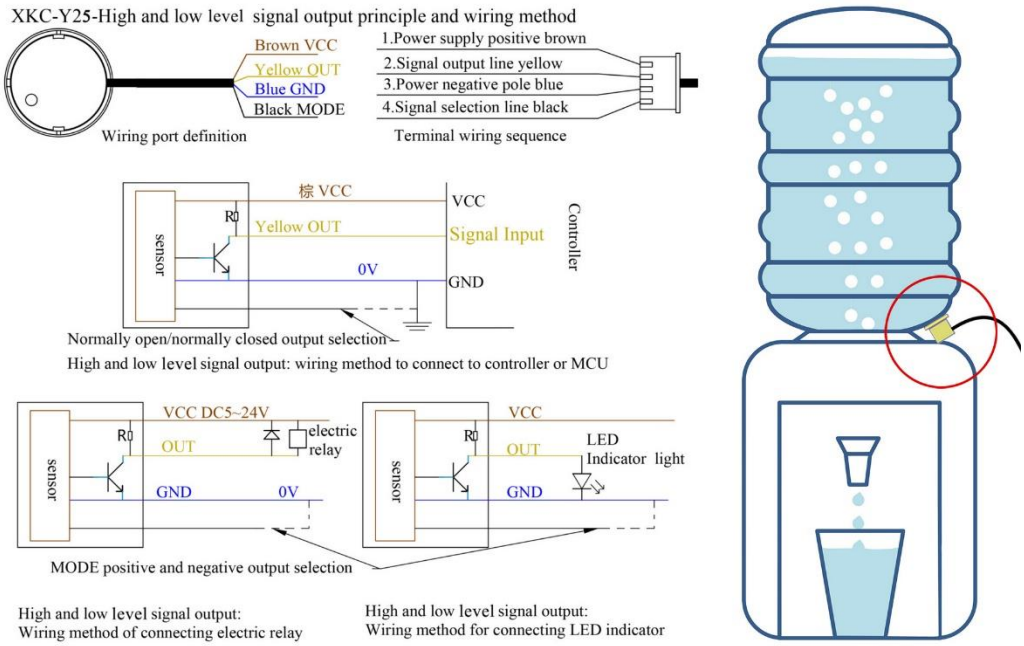


Рисунок 4. Схема подключения датчика XKC-Y25- из документации производителя [2]

На следующей фотографии (см. Рисунок 5) представлены отладочный макет и окончательный вариант устройства.

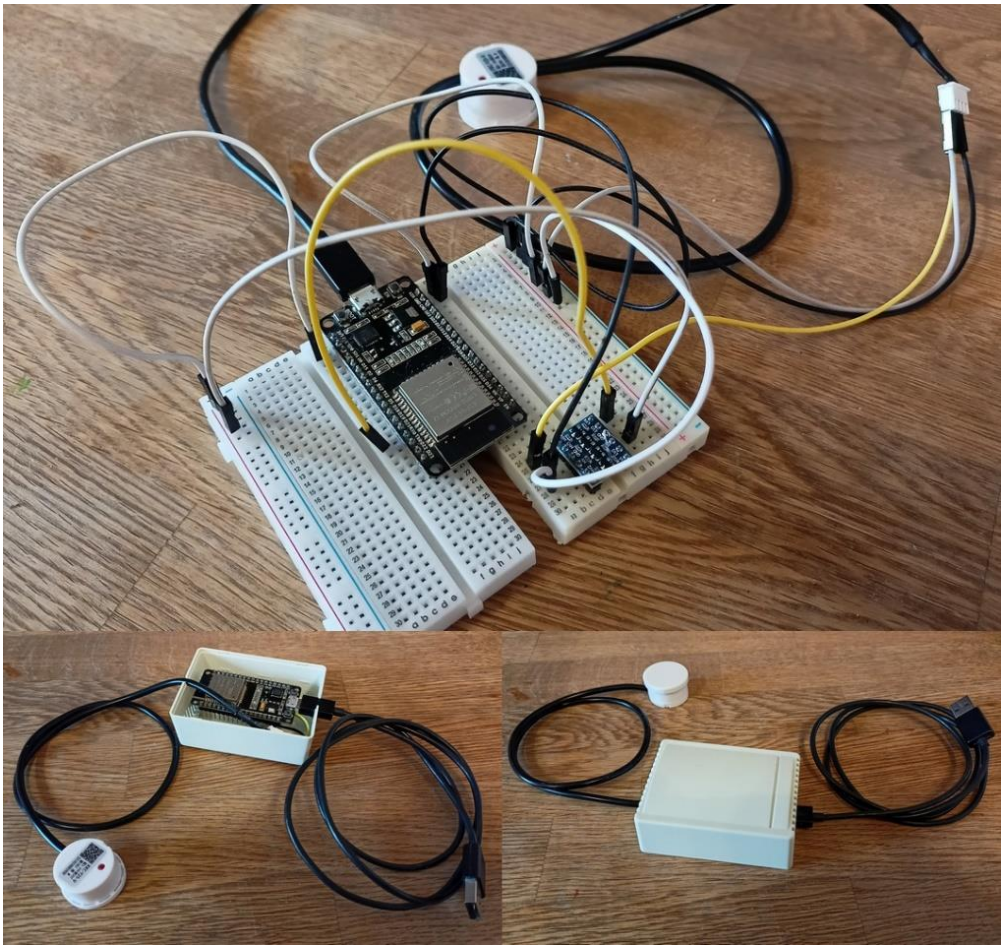

Рисунок 5. Отладочный макет и финальный вариант устройства мониторинга

Для ESP32 на языке MicroPython написано приложение, которое в цикле выполняет алгоритм, состоящий из нескольких последовательных шагов [3]. Для удобства контроля состояния устройства используется индикация текущего шага с помощью изменения режима работы светодиода, расположенного на корпусе устройства. Далее перечислены основные шаги:

1. Подключиться к WLAN. Контрольный светодиод находится в режиме мигания с низкой частотой.
2. Подключиться к серверу MQTT. Контрольный светодиод находится в режиме мигания с высокой частотой.
3. Считать текущее значение с датчика. Контрольный светодиод находится в режиме свечения без мигания.
4. Отправить считанное значение на сервер MQTT.
5. Установить таймер пробуждения на 30 секунд и перейти в режим сна. Контрольный светодиод не светится.

Если на очередном шаге выполнения возникает ошибка (невозможно подключиться к WLAN или MQTT, невозможно считать или отправить значение датчика), то выполняется переход на последний шаг. Далее приведён фрагмент исходного кода основного цикла приложения:

```
async def main():
    initialize()

    onboard_led: AsyncLED = AsyncLED(
        machine.Signal(machine.Pin(ONBOARD_LED_GPIO, machine.Pin.OUT)))
    liquid_sensor: machine.Signal = machine.Signal(
        machine.Pin(LIQUID_SENSOR_GPIO, machine.Pin.IN))

    onboard_led.flash(LED_FLASHING_RATE_CONNECTING_TO_WLAN)
    await connect_to_wlan(config.WLAN_NETWORK, config.WLAN_PASSWORD)

    onboard_led.flash(LED_FLASHING_RATE_CONNECTING_TO_MQTT)
    mqtt: MQTTClient = await connect_to_mqtt(
        config.SENSOR_HOSTNAME,
        config.MQTT_BROKER_HOST,
        config.MQTT_BROKER_PORT,
        config.MQTT_BROKER_USER_NAME,
        config.MQTT_BROKER_USER_PASSWORD,
        config.MQTT_BROKER_KEEPALIVE,
        config.MQTT_TOPIC_ROOT + config.MQTT_TOPIC_STATUS,
```

```
        config.MQTT_TOPIC_STATUS_DOWN,
    )

    result = await sync_to_async(
        mqtt.publish,
        config.MQTT_TOPIC_ROOT + config.MQTT_TOPIC_STATUS,
        config.MQTT_TOPIC_STATUS_UP
    )
    if isinstance(result, Exception):
        raise result

    onboard_led.on()
    sensor_value: int = get_liquid_sensor_value(liquid_sensor)
    await sync_to_async(
        mqtt.publish,
        config.MQTT_TOPIC_ROOT + config.MQTT_TOPIC_LIQUID_SENSOR,
        str(sensor_value).encode(),
    )

    machine.deepsleep(config.SENSOR_SLEEP_AFTER_MEASUREMENT_SEC * 1000)
```

## Проектирование и разработка программного обеспечения для сбора и отображения информации с устройств мониторинга

### Подсистема сбора информации с устройств мониторинга

Подсистема состоит из веб-сервера, сервера MQTT и серверной части веб-приложения. Веб-сервер используется для хостинга клиентской части веб-приложения и является стандартной частью большинства веб-приложений.

Сервер MQTT используется для хранения последних данных, полученных от устройств мониторинга. Устройства различаются по уникальным идентификаторам микроконтроллеров. Для каждого устройства хранится признак его активности (устройство считается активным, если последние данные от него получены не ранее 2 минут назад) и признак наличия жидкости в зоне чувствительности датчика. На следующем скриншоте приведён пример для двух устройств (см. Рисунок 6. Используемые топики в брокере MQTT).

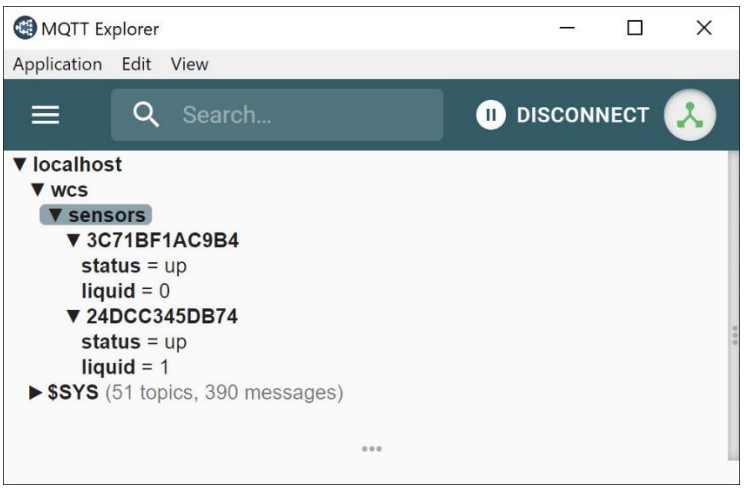


Рисунок 6. Используемые топики в брокере MQTT

Серверная часть веб-приложения выполняет две задачи: в момент загрузки подписывается на получение обновлений от сервера MQTT, полученные сообщения пересылает клиентской части веб-приложения. Для выполнения второй задачи в серверной части веб-приложения реализован следующий RESTful интерфейс:

```
@app.get("/sensors/list")
async def get_sensors_list() -> JSONResponse:
    ...

@app.get("/sensors/map")
async def get_sensors_map() -> FileResponse:
    ...

@app.get("/sse/sensors")
async def sse_sensors_stream(request: Request) -> EventSourceResponse:
    ...
```

Метод `get_sensors_list` возвращает JSON со списком зарегистрированных устройств мониторинга.

```
{
    "3C71BF1AC9B4": {
        "id": "3C71BF1AC9B4",
        "desc": "Левый холл, рядом с кабинетом 1-21",
        "plan": {
            "floor": 1,
            "coords": [134, 172]
        }
    },
    "24DCC345DB74": {
```

```
        "id": "24DCC345DB74",
        "desc": "Правый большой зал, помещение 1-5",
        "plan": {
            "floor": 1,
            "coords": [823, 567]
        }
    }
}
```

Метод `get_sensors_map` возвращает SVG с планом расположения устройств мониторинга. Метод `sse_sensors_stream` позволяет клиентской части веб-приложения подписаться на получение уведомлений об изменении состояния устройств мониторинга.

**Подсистема для предоставления информации пользователю**

В настоящий момент подсистема состоит из единственного интерфейса – клиентской части веб-приложения, работающей в веб-браузере на стороне пользователя.

В процессе работы клиентская часть веб-приложения выполняет следующие действия:

- Запрашивает у серверной части список устройств мониторинга и изображение с планом.
- Подписывается на получение обновлений об изменении состояния устройств мониторинга.

После получения уведомления об изменении состояния устройства мониторинга оно по идентификатору находится на плане, далее включается пиктограмма, соответствующая полученному состоянию.

На рисунке ниже (см. Рисунок 7) приведено окно веб-браузера с загруженным приложением. Левая часть окна приложения содержит план с указанием мест расположения кулеров с питьевой водой, правая – легенду и список кулеров в которых в текущий момент времени отсутствует вода. Красным цветом обозначены кулеры без воды, зелёным – с водой, серым – без устройства мониторинга.

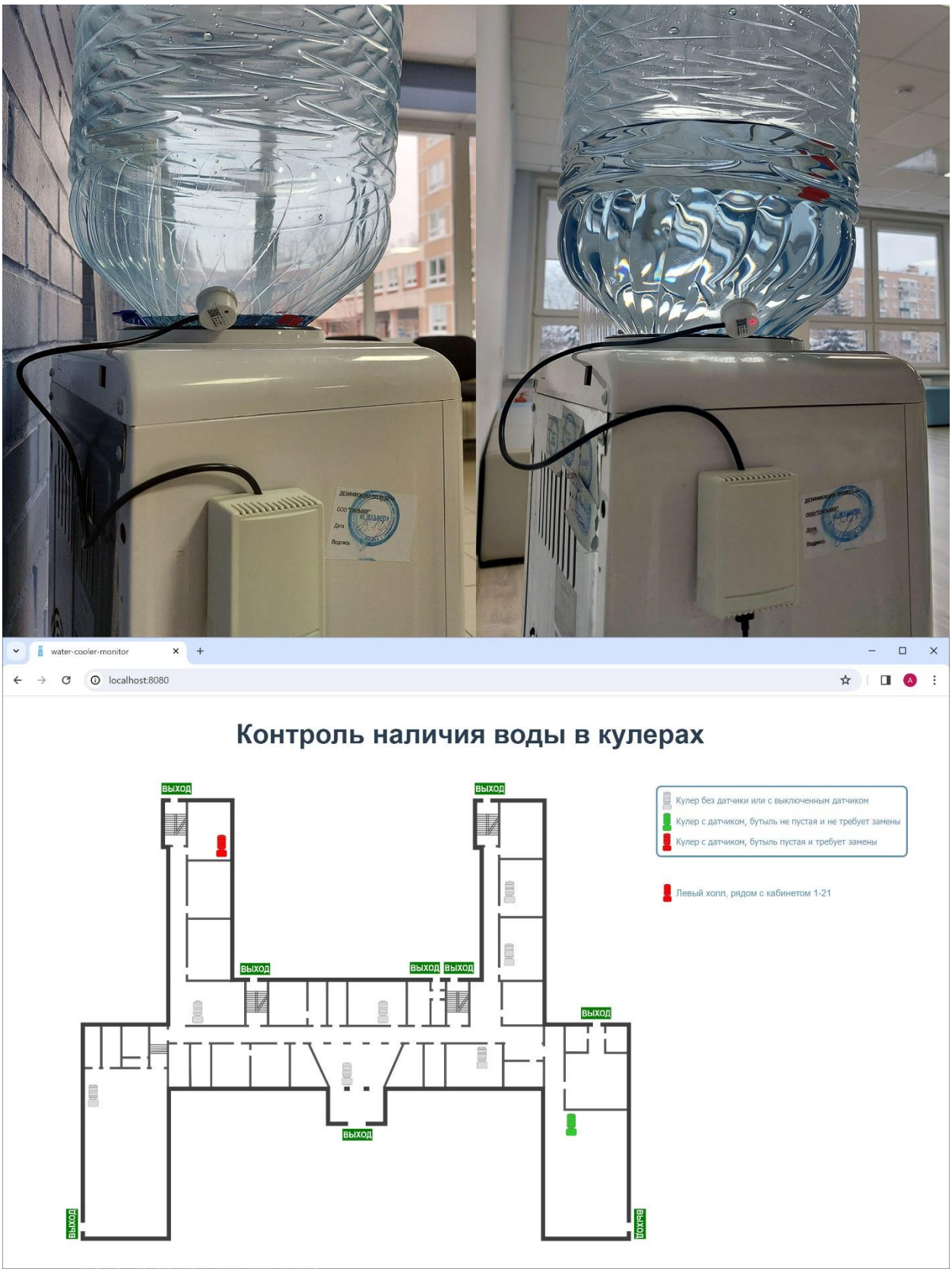


Рисунок 7. Коллаж с фотографиями двух кулеров с устройствами для мониторинга и скриншотом окна веб-браузера с клиентской частью веб-приложения

## Практическая значимость

Разработанный программно-аппаратный комплекс повышает комфорт школьников, не увеличивая нагрузку на сотрудников администрации школы за счёт автоматизации рутинных действий.

## Заключение

В ходе работы над проектом были решены все поставленные задачи и разработан программно-аппаратный комплекс для мониторинга наличия воды в кулерах.

## Перспективы развития

1. Дополнение подсистемы отображения информации подсистемой оповещения пользователей, работающей на основе каналов в мессенджере Telegram.
2. Создание подсистемы предсказания расхода воды в кулерах на основе анализа данных, собираемых подсистемой мониторинга.

## Список литературы


1. Библиотека документации компании Espressif Systems : [сайт]. - China. - URL: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf (дата обращения: 10.12.2023). - Текст : электронный.
2. Библиотека документации компании Shenzhen XingKeChuang Technology Co., Ltd : [сайт]. - Shenzhen,China. - URL: http://pdf.sz-xkc.cn/?pdf-id=80&type=en (дата обращения: 05.01.2024). - Текст : электронный.
3. Библиотека документации проекта MicroPython, компания George Robotics Limited : [сайт]. - URL: https://docs.micropython.org/en/v1.22.0/ (дата обращения: 15.02.2024). - Текст : электронный.