**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ 5

[1 ОБЗОР ЛИТЕРАТУРЫ 6](#_Toc184131205)

[1.1 Аналоги устройства](#_Toc184131206) 6

[1.2 Основные компоненты 8](#_Toc184131207)

[1.3 Плата с микроконтроллером 8](#_Toc184131208)

[1.4 Датчик температуры и влажности воздуха 9](#_Toc184131209)

[1.5 Дисплей 9](#_Toc184131210)

[1.6 Датчик звука KY-037 9](#_Toc184131211)

[1.7 Элементы управления 10](#_Toc184131212)

[2 РАЗРАБОТКА СТРУКТУРЫ УСТРОЙСТВА 11](#_Toc184131215)

[2.1 Перечень блоков 11](#_Toc184131216)

[2.2 Взаимодействие блоков 11](#_Toc184131217)

[3 ОБОСНОВАНИЕ ВЫБОРА УЗЛОВ, ЭЛЕМЕНТОВ ФУНКЦИОНАЛЬНОЙ СХЕМЫ УСТРОЙСТВА](#_Toc184131218) 12

[3.1 Аппаратная платформа](#_Toc184131219) 12

[3.2 Датчик температуры и влажности](#_Toc184131220) 12

[3.3 Датчик звука](#_Toc184131221) 12

[3.4 Дисплей](#_Toc184131222) 12

[3.5 Управление и структура меню](#_Toc184131226) 13

[4 РАЗРАБОТКА ПРИНЦИПИАЛЬНОЙ ЭЛЕКТРИЧЕСКОЙ СХЕМЫ УСТРОЙСТВА 1](#_Toc184131227)5

[4.1 Разработка системы питания 1](#_Toc184131228)5

[4.2 Описание входов и выходов микроконтроллера](#_Toc184131229) 15

[5 РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ](#_Toc184131230) 16

[5.1 Требования к программе](#_Toc184131231) 16

[ЗАКЛЮЧЕНИЕ](#_Toc184131235) 17

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ](#_Toc184131236) 18

[ПРИЛОЖЕНИЕ А](#_Toc184131237) 19

[ПРИЛОЖЕНИЕ Б](#_Toc184131238) 20

[ПРИЛОЖЕНИЕ В 21](#_Toc184131239)

[ПРИЛОЖЕНИЕ Г 22](#_Toc184131240)

[ПРИЛОЖЕНИЕ Д 23](#_Toc184131241)

[ПРИЛОЖЕНИЕ Е 30](#_Toc184131242)

[ПРИЛОЖЕНИЕ Ж 3](#_Toc184131243)1

ВВЕДЕНИЕ

Мониторинг окружающей среды является важным элементом для создания комфортных условий как в жилых, так и в рабочих пространствах. Современные устройства для отслеживания параметров, таких как температура, влажность и уровень шума, позволяют поддерживать оптимальный микроклимат, что положительно влияет на здоровье и благополучие людей, а также на работу различных технических систем.

Особенно актуальны подобные решения в условиях изменяющегося климата и роста требований к энергоэффективности зданий и производственных объектов. Устройства, способные измерять и анализировать параметры окружающей среды, имеют множество применений: от обеспечения благоприятных условий в жилых помещениях до контроля за параметрами в производственных цехах и офисах. Они позволяют следить за изменениями температуры, влажности и уровня шума, а также своевременно реагировать на их отклонения от нормальных значений. Это помогает предотвратить перегрев или переохлаждение помещений, контролировать влажность для предотвращения развития плесени и оптимизировать акустические условия для повышения комфорта и продуктивности.

Одной из ключевых задач подобных устройств является предоставление пользователю возможности не только наблюдать за текущими параметрами, но и задавать индивидуальные пороговые значения для каждого из них.

Целью разработки подобных устройств является не только предоставление информации пользователю, но и обеспечение простоты в эксплуатации. Такие устройства обычно оснащены интуитивно понятными интерфейсами и отображают измеренные параметры в реальном времени, что позволяет пользователю легко оценивать текущее состояние окружающей среды. Важной задачей является также создание системы визуальной индикации, которая наглядно демонстрирует изменения параметров, не требуя от пользователя сложных действий.

1 ОБЗОР ЛИТЕРАТУРЫ

1.1 Аналоги устройства

На рынке существует множество устройств для мониторинга параметров окружающей среды, которые измеряют такие показатели, как температура, влажность и уровень шума. Эти устройства варьируются от простых бытовых датчиков до сложных систем для промышленного использования, предоставляющих точные данные в реальном времени.

* + 1. **Xiaomi Mi Temperature and Humidity Monitor 2**

Это компактное устройство, предназначенное для мониторинга температуры и влажности в помещении. Дисплей выполнен на основе электронных чернил, что позволяет экономить заряд батареи и обеспечивает хорошую видимость даже при ярком свете. Устройство подключается к экосистеме Xiaomi через приложение Mi Home для удобного мониторинга и хранения данных.

Достоинства – низкое энергопотребление благодаря технологии электронных чернил. Интеграция с экосистемой умного дома Xiaomi. Удобное приложение с функцией истории данных.

Недостатки – отсутствие встроенного датчика шума. Ограниченные настройки без использования приложения. Не поддерживает внешние датчики.

На рисунке 1.1 изображено устройство Xiaomi Mi Temperature and Humidity Monitor 2.



Рисунок 1.1 – Устройство Xiaomi Mi Temperature and Humidity Monitor 2

**1.1.2 Govee Temperature Humidity Monitor**

Устройство от Govee предлагает мониторинг температуры и влажности с возможностью беспроводного подключения через Bluetooth или Wi-Fi. Оно синхронизируется с приложением Govee Home, которое предоставляет графики и уведомления о состоянии окружающей среды в реальном времени. Имеет LED-дисплей для локального отображения данных.

Достоинства – беспроводное подключение через Bluetooth или Wi-Fi. Удобное приложение для мониторинга в реальном времени. Возможность настройки пороговых значений с уведомлениями.

Недостатки – нет встроенного датчика для мониторинга уровня шума. Требует частой подзарядки в зависимости от режима работы. Зависимость от приложения для расширенных функций.

На рисунке 1.2 изображено устройство Govee Temperature Humidity Monitor.



Рисунок 1.2 – Устройство Govee Temperature Humidity Monitor

**1.1.3 SensorPush Wireless Thermometer/Hygrometer**

SensorPush — это беспроводной термометр-гигрометр, который синхронизируется с мобильным приложением для мониторинга температуры и влажности. Он использует Bluetooth и может хранить данные в облаке с помощью дополнительного Wi-Fi-шлюза. Устройство обладает высокой точностью измерений и рекомендуется для использования в местах, требующих точного контроля микроклимата.

Достоинства – высокая точность измерений. Хранение данных в облаке при наличии Wi-Fi-шлюза. Компактные размеры и простота использования.

Недостатки – не отображает данные на дисплее, только через приложение. Отсутствие датчика шума. Требует дополнительного шлюза для полноценной работы с Wi-Fi.

На рисунке 3 изображено устройство SensorPush Wireless Thermometer/Hygrometer.



Рисунок 1.3 – Устройство SensorPush Wireless Thermometer/Hygrometer

1.2 Основные компоненты

Основные компоненты устройства:

– плата с микроконтроллером;

– датчик температуры и влажности воздуха;

– датчик звука;

– OLED дисплей;

– 3 кнопки.

1.3 Плата с микроконтроллером

Среди микроконтроллеров доступных на рынке была рассмотрена плата Arduino NANO V3.0 CH340. Основные характеристики:

– микроконтроллер ATmega328;

– напряжение питания 5В;

– входное напряжение (рекомендуемое) 7-12В;

– входное напряжение (предельное) 6-20В;

– цифровой ввод-вывод 14 линии;

– аналоговый ввод 8 линий;

– постоянный ток на линиях ввода-вывода 40мА;

– постоянный ток на линии 3.3В 50мА;

– flash-память 32КВ, 2 КВ из них использованы для загрузчика;

– SRAM-память 2КВ;

– EEPROM-память 1КВ;

– тактовая частота 16МГц [1].

1.4 Датчик температуры и влажности воздуха

На рынке распространены устройства, которые совмещают в себе и датчик температуры и датчик влажности. Одним из таких решений являются датчики DHT. Был рассмотрен DHT11. Основные характеристики:

– питание: DC 3,5 – 5,5 В;

– ток питания: в режиме измерения 0.3mA. В режиме ожидания 60μA;

– определение влажности 20–80 % с точностью 5 %;

– определение температуры 0–50 °С c точностью 2 %;

– частота опроса не более 1 Гц [2].

1.5 Дисплей

Для отображения параметров окружающей среды был выбран OLED дисплей размером 0.96 дюйма с разрешением 128x64 пикселей и интерфейсом I2C. Данный дисплей обеспечивает четкое отображение данных даже при низком уровне освещения. Основные характеристики:

– питание: DC 3,3 – 5 В;

– интерфейс: I2C (2 контакта: SCL и SDA);

– тип дисплея: монохромный OLED;

– разрешение: 128x64 пикселей;

– размер экрана: 0.96 дюйма;

– потребляемый ток: менее 20 мА;

– рабочая температура: -40°C – +85°C [3].

1.6 Датчик звука KY-037

Для мониторинга уровня шума был выбран датчик KY-037. Этот датчик позволяет определять звуковую активность в окружающей среде, а также предоставляет возможность регулировать порог чувствительности. Основные характеристики:

– питание: DC 3,3 – 5 В;

– ток потребления: менее 10 мА;

– тип выхода: цифровой и аналоговый сигнал;

– диапазон частот: 50 Гц – 10 кГц;

– регулируемый порог чувствительности: потенциометр;

– время отклика: менее 1 секунды [4].

1.7 Элементы управления

Для управления устройством был выбран способ управления тремя кнопками, с помощью которых пользователь может регулировать граничные значения параметров микроклимата.

2 РАЗРАБОТКА СТРУКТУРЫ УСТРОЙСТВА

Структурная схема устройства приведена в приложении А.

2.1 Перечень блоков

В данном устройстве можно выделить 7 основных блоков:

– блок датчиков;

– микроконтроллер;

– блок управления;

– блок отображения информации.

В состав блока датчиков входят датчики влажности и температуры воздуха, звука.

В состав блока управления входят кнопки, с помощью которых пользователь может регулировать граничные значения параметров микроклимата.

Блок отображения информации состоит из дисплея.

2.2 Взаимодействие блоков

Датчики снимают показания температуры, влажности и уровня звука, передавая эту информацию в микроконтроллер.

Микроконтроллер обрабатывает полученные данные и передает их в блок отображения, где информация визуализируется для пользователя. Если значения, переданные датчиками, выходят за допустимые пределы, микроконтроллер инициирует мигание дисплея.

Блок управления позволяет пользователю настраивать пороговые значения для параметров, таких как температура или уровень шума. Эти параметры передаются микроконтроллеру. В случае бездействия пользователя в течение определённого времени (например, 5 секунд), микроконтроллер автоматически выходит из режима настройки и возвращает устройство в режим обычного мониторинга. Для этого микроконтроллер использует встроенные таймеры, которые следят за временем между нажатиями кнопок.

Блок отображения информации преобразует данные, переданные микроконтроллером, в удобную для пользователя форму, показывая текущие значения параметров окружающей среды — влажность, температуру и уровень шума.

3 ОБОСНОВАНИЕ ВЫБОРА УЗЛОВ, ЭЛЕМЕНТОВ ФУНКЦИОНАЛЬНОЙ СХЕМЫ УСТРОЙСТВА

Функциональная схема устройства приведена в приложении Б.

3.1 Аппаратная платформа

Для курсового проекта была выбрана плата Arduino Nano благодаря своей компактности и удобству в использовании. Она легко интегрируется с различными сенсорами, что позволяет создать эффективную систему мониторинга, адаптированную под конкретные задачи. Благодаря своей архитектуре и большому количеству доступных библиотек, Arduino Nano также проста в программировании и совместима с Arduino IDE, что значительно упрощает процесс разработки.

3.2 Датчик температуры и влажности

Для курсового проекта был выбран датчик DHT11, который отличается надежностью и доступной ценой. Датчик измеряет температуру и влажность воздуха, а также прост в использовании.

Датчик использует однопроводной интерфейс для передачи данных, что упрощает подключение и снижает количество проводов, необходимых для работы системы. DHT11 не является самым точным датчиком на рынке, его точности достаточно для выполнения базового мониторинга параметров окружающей среды. Частота опроса составляет около 1 раза в секунду.

3.3 Датчик звука

Для измерения уровня шума был выбран датчик KY-037, который отличается простотой использования и средней чувствительностью. Датчик работает на основе микрофона, который преобразует звуковые колебания в электрический сигнал. Уровень шума измеряется путем анализа изменения амплитуды сигнала, который поступает с микрофона. Когда уровень звука в окружающей среде повышается, амплитуда сигнала возрастает, что позволяет датчику фиксировать изменение интенсивности шума.

KY-037 легко интегрируется с микроконтроллером, обеспечивая относительно точные данные для мониторинга уровня шума. Это позволяет отслеживать звуковую активность в окружающей.

3.4 Дисплей

Для отображения данных в системе был выбран OLED дисплей 0.96" с разрешением 128x64 пикселей. Этот дисплей обеспечивает высокое качество изображения и чёткую визуализацию данных. Благодаря поддержке интерфейса I2C, дисплей легко подключается к микроконтроллеру, минимизируя количество проводов и упрощая интеграцию в систему. Высокая контрастность и яркость экрана обеспечивают отличную видимость при различных уровнях освещенности, а его компактные размеры делают дисплей удобным для использования в небольшом корпусе. OLED технология позволяет получать яркое и чёткое изображение без подсветки, что повышает энергоэффективность устройства. Благодаря поддержке библиотеки для Arduino, настройка и программирование дисплея быстрая и удобная, что ускоряет процесс разработки интерфейса системы.

3.5 Управление и структура меню

В проекте предусмотрено управление системой через меню, которое позволяет пользователю легко настраивать граничные параметры температуры, влажности и шума. Для навигации по меню используются кнопки. Пользователь может просматривать текущие значения параметров и вносить изменения при необходимости. Структура меню представлена в таблице 3.1.

Таблица 3.1 – Структура меню

|  |  |  |  |
| --- | --- | --- | --- |
| Уровень меню | Кнопка 1 (влево) | Кнопка 2 (выбор) | Кнопка 3 (вправо) |
| Страница 1 – параметры | Переключение на страницу 3 | Меню настроек пороговых значений | Переключение на страницу 2 |
| Страница 2 –время | Переключение на страницу 1 | Меню настроек часов | Переключение на страницу 3 |
| Страница 3 – смайлик | Переключение на страницу 2 | – | Переключение на страницу 1 |
| Меню параметров | Уменьшение значения параметра | Выход из настроек пороговых значений | Увеличение значения параметра |
| Меню часов | Уменьшение значения параметра | Выбор часов, минут, времени, дня, месяца, года | Увеличение значения параметра |

Кнопка 1 (Влево) служит для перемещения по элементам интерфейса влево. Если пользователь находится на крайнем элементе текущего экрана, нажатие кнопки перемещает его на предыдущий экран интерфейса. В меню настройки кнопка используется для выбора предыдущего параметра или уменьшения значения, если параметр редактируется.

Кнопка 2 (Выбор/Выход) отвечает за подтверждение выбора и выход из текущего режима. Если пользователь находится в основном интерфейсе, нажатие этой кнопки активирует режим редактирования выбранного параметра. При повторном нажатии в режиме редактирования кнопка сохраняет изменения и возвращает пользователя к просмотру интерфейса. Если пользователь находится в меню настройки, кнопка завершает редактирование текущего параметра и возвращает в общий режим.

Кнопка 3 (Вправо) предназначена для перемещения по элементам интерфейса вправо. Если пользователь находится на крайнем элементе текущего экрана, нажатие кнопки переводит его на следующий экран интерфейса. В меню настройки кнопка используется для выбора следующего параметра или увеличения значения, если параметр редактируется.

4 РАЗРАБОТКА ПРИНЦИПИАЛЬНОЙ ЭЛЕКТРИЧЕСКОЙ СХЕМЫ УСТРОЙСТВА

Принципиальная схема устройства приведена в приложении В.

4.1 Разработка системы питания

Для расчёта характеристик блока питания была составлена таблица 4.1.

Таблица 4.1 – Расчет характеристик блока питания

|  |  |  |  |
| --- | --- | --- | --- |
| Название устройства | Рабочее напряжение, В | Максимальный потребляемый ток, мА | Максимальная потребляемая мощность, мВт |
| Arduino Nano | 5 | 20 | 100 |
| OLED дисплей (SSD1306) | 5 | 20 | 100 |
| DHT11 | 5 | 0,3 | 1,5 |
| Датчик звука KY-037 | 5 | 10 | 50 |

Максимальный потребляемый ток устройства составил 50,3 мА, максимальная потребляемая мощность – 251,5 мВт. Для устройства, разрабатываемого в проекте, требуется обеспечить уровень входного напряжения 5 В.

При выборе блока питания необходимо учитывать запас мощности в 20% для обеспечения надёжной работы устройства. С учётом этого запаса, характеристики блока питания должны составлять:

– максимальная мощность: 301,8 мВт;

– максимальный ток: 60,36 мА.

Для питания устройства достаточно стандартного источника питания USB на 5 В, поскольку его параметры превышают требуемые характеристики.

4.2 Описание входов и выходов микроконтроллера

Питание микроконтроллера обеспечивается на входах 5V и GND.

К микроконтроллеру подключены 3 кнопки, заведённые на цифровые входы D2, D3 и D4 соответственно. На которых активирован встроенный в плату подтягивающий резистор.

На цифровой вход D7 подаётся сигнал DATA с датчика температуры и влажности DHT11.

На аналоговый вход A0 подаётся сигнал с датчика звука KY-037.

Через аналоговые выходы A4 и A5 по интерфейсу I2C подключён OLED-дисплей SSD1306.

Все компоненты подключены к общей шине питания 5V и общему проводу GND.

5 РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

5.1 Требования к программе

Программа, управляющая устройством, должна реализовывать следующий функционал:

– считывание данных с датчиков температуры, влажности и звука с установленным интервалом;

– обработка считанных данных и отображение их на OLED-дисплее;

– поддержка настройки следующих параметров: пороговые значения температуры и влажности, а также времени и даты;

– меню управления с использованием трёх кнопок (влево, выбор, вправо);

– обработка ошибок;

– минимизация дребезга кнопок при считывании нажатий;

– отображение смайлика на OLED-дисплее, отражающего настроение на основе полученных данных с датчиков.

Схема программы приведена в приложении Г.

Код программы приведен в приложении Д.

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта было разработано устройство для мониторинга параметров окружающей среды, которое успешно решает задачи измерения температуры, влажности и уровня шума. Устройство отображает собранные данные на OLED-дисплее, предоставляя пользователю удобный способ мониторинга параметров состояния среды.

Реализованная система управления позволяет пользователю гибко настраивать пороговые значения для параметров, а также дату и время. Удобный интерфейс и поддержка управления с помощью трёх кнопок обеспечивают простоту использования устройства.

В будущем проект имеет большое количество направлений для расширения и улучшения. Возможности для улучшений включают добавление новых датчиков для измерения других параметров окружающей среды, таких как уровень освещенности, качество воздуха, давление и другие. Также можно интегрировать устройство с мобильным приложением, что позволит пользователю удаленно отслеживать данные и получать уведомления о превышении пороговых значений. Другим возможным улучшением является внедрение функции сохранения данных для последующего анализа, что может быть полезным для исследовательских целей или для долгосрочного мониторинга.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Спецификация Arduino Nano [Электронный ресурс]. – Режим доступа: https://docs.arduino.cc/resources/datasheets/A000005-datasheet.pdf

[2] Спецификация датчика DHT11 [Электронный ресурс]. – Режим доступа: http://robocraft.ru/files/datasheet/DHT11.pdf

[3] Спецификация дисплея OLED SSD1306 [Электронный ресурс]. – Режим https://www.vishay.com/docs/37902/oled128o064dbpp3n00000.pdf

[4] Спецификация датчика звука KY-037 [Электронный ресурс]. – Режим доступа: https://roboeq.ir/files/id/3882/name/KY-037.pdf

ПРИЛОЖЕНИЕ А

**(обязательное)**

**Схема электрическая структурная**

ПРИЛОЖЕНИЕ Б

**(обязательное)**

**Схема электрическая функциональная**

ПРИЛОЖЕНИЕ В

**(обязательное)**

**Схема электрическая принципиальная**

ПРИЛОЖЕНИЕ Г

**(обязательное)**

**Схема программы**

ПРИЛОЖЕНИЕ Д

**(обязательное)**

**Листинг кода**

001 #include <Adafruit\_SSD1306.h>

002 #include <Adafruit\_GFX.h>

003 #include <Wire.h>

004 #include <DHT.h>

005 #include <TimeLib.h>

006 #include <EEPROM.h>

007

008 #define SCREEN\_WIDTH 128

009 #define SCREEN\_HEIGHT 64

010 #define OLED\_RESET -1

011 Adafruit\_SSD1306 display(SCREEN\_WIDTH, SCREEN\_HEIGHT, &Wire, OLED\_RESET);

012

013 #define DHTPIN 7

014 #define DHTTYPE DHT11

015 DHT dht(DHTPIN, DHTTYPE);

016

017 #define SOUND\_SENSOR\_PIN A0

018

019 #define NOISE\_THRESHOLD\_LOW 30

020 #define NOISE\_THRESHOLD\_HIGH 100

021

022 #define BTN\_LEFT 2

023 #define BTN\_SELECT 3

024 #define BTN\_RIGHT 4

025

026 // EEPROM адреса

027 #define EEPROM\_TIME\_ADDR 0

028 #define EEPROM\_TEMP\_MIN\_ADDR 4

029 #define EEPROM\_TEMP\_MAX\_ADDR 8

030 #define EEPROM\_HUM\_MIN\_ADDR 12

031 #define EEPROM\_HUM\_MAX\_ADDR 16

032

033 #define SAMPLE\_WINDOW 25

034 #define DECAY\_FACTOR 0.7

035 #define SETTING\_TIMEOUT 5000 // 5 секунд для выхода из режима настройки

036

037 unsigned long lastSampleTime = 0;

038 int peakToPeak = 0;

039 float decayingPeak = 0;

040 unsigned long lastSettingActivity = 0;

041

042 float tempMin = 20.0;

043 float tempMax = 30.0;

044 float humMin = 30.0;

045 float humMax = 60.0;

046

047 int currentMenuItem = 0;

048 bool isSettingMode = false;

049 bool isAdjusting = false;

050

051 unsigned long lastButtonPress = 0;

052 const unsigned long BUTTON\_COOLDOWN = 300;

053

054 unsigned long lastScreenSwitch = 0;

055 const unsigned long SCREEN\_CHANGE\_INTERVAL = 15000; // 15 секунд

056 bool showMainScreen = true;

057

058 enum Screen {

059 MAIN,

060 TIME,

061 EMOJI

062 };

063 Screen currentScreen = MAIN;

064

065 enum TimeSettingMode { NONE, HOUR, MINUTE, DAY, MONTH, YEAR };

066 TimeSettingMode timeSettingMode = NONE;

067

068 void setup() {

069 Serial.begin(9600);

070

071 dht.begin();

072 delay(2000); // Задержка для инициализации DHT

073

074 if(!display.begin(SSD1306\_SWITCHCAPVCC, 0x3C)) {

075 Serial.println(F("SSD1306 allocation failed"));

076 for(;;);

077 }

078

079 display.clearDisplay();

080 display.display();

081

082 pinMode(BTN\_LEFT, INPUT\_PULLUP);

083 pinMode(BTN\_SELECT, INPUT\_PULLUP);

084 pinMode(BTN\_RIGHT, INPUT\_PULLUP);

085

086 loadSettingsFromEEPROM();

087 loadTimeFromEEPROM();

088 }

089

090 void loadSettingsFromEEPROM() {

091 EEPROM.get(EEPROM\_TEMP\_MIN\_ADDR, tempMin);

092 EEPROM.get(EEPROM\_TEMP\_MAX\_ADDR, tempMax);

093 EEPROM.get(EEPROM\_HUM\_MIN\_ADDR, humMin);

094 EEPROM.get(EEPROM\_HUM\_MAX\_ADDR, humMax);

095

096 // Проверка на корректность значений

097 if (isnan(tempMin) || tempMin < -50 || tempMin > 100) tempMin = 20.0;

098 if (isnan(tempMax) || tempMax < -50 || tempMax > 100) tempMax = 30.0;

099 if (isnan(humMin) || humMin < 0 || humMin > 100) humMin = 30.0;

100 if (isnan(humMax) || humMax < 0 || humMax > 100) humMax = 60.0;

101 }

102

103 void saveSettingsToEEPROM() {

104 EEPROM.put(EEPROM\_TEMP\_MIN\_ADDR, tempMin);

105 EEPROM.put(EEPROM\_TEMP\_MAX\_ADDR, tempMax);

106 EEPROM.put(EEPROM\_HUM\_MIN\_ADDR, humMin);

107 EEPROM.put(EEPROM\_HUM\_MAX\_ADDR, humMax);

108 }

109

110 void loadTimeFromEEPROM() {

111 unsigned long timeData;

112 EEPROM.get(EEPROM\_TIME\_ADDR, timeData);

113 setTime(timeData);

114 }

115

116 void saveTimeToEEPROM() {

117 unsigned long timeData = now();

118 EEPROM.put(EEPROM\_TIME\_ADDR, timeData);

119 }

120

121 void handleButtons() {

122 unsigned long currentTime = millis();

123 static unsigned long lastRepeatTime = 0;

124 const unsigned long REPEAT\_DELAY = 300;

125 const unsigned long REPEAT\_INTERVAL = 100;

126

127 bool leftPressed = digitalRead(BTN\_LEFT) == LOW;

128 bool rightPressed = digitalRead(BTN\_RIGHT) == LOW;

129 bool selectPressed = digitalRead(BTN\_SELECT) == LOW;

130

131 if (leftPressed && currentTime - lastButtonPress > BUTTON\_COOLDOWN) {

132 lastButtonPress = currentTime;

133 if (!isSettingMode && timeSettingMode == NONE) {

134 currentScreen = (Screen)((currentScreen - 1 + 3) % 3);

135 }

136 }

137

138 if (rightPressed && currentTime - lastButtonPress > BUTTON\_COOLDOWN) {

139 lastButtonPress = currentTime;

140 if (!isSettingMode && timeSettingMode == NONE) {

141 currentScreen = (Screen)((currentScreen + 1) % 3);

142 }

143 }

144

145 if (selectPressed && currentTime - lastButtonPress > BUTTON\_COOLDOWN) {

146 lastButtonPress = currentTime;

147 lastSettingActivity = currentTime;

148 if (currentScreen == MAIN) {

149 if (!isSettingMode) {

150 isSettingMode = true;

151 currentMenuItem = 0;

152 } else if (!isAdjusting) {

153 isAdjusting = true;

154 } else {

155 isSettingMode = false;

156 isAdjusting = false;

157 saveSettingsToEEPROM();

158 }

159 } else if (currentScreen == TIME) {

160 if (timeSettingMode == NONE) {

161 timeSettingMode = HOUR;

162 } else {

163 timeSettingMode = (TimeSettingMode)((int)timeSettingMode + 1);

164 if (timeSettingMode > YEAR) {

165 timeSettingMode = NONE;

166 saveTimeToEEPROM();

167 }

168 }

169 }

170 // Сохранение настроек и времени при нажатии второй кнопки

171 saveSettingsToEEPROM();

172 saveTimeToEEPROM();

173 }

174

175 if ((leftPressed || rightPressed) &&

176 (currentTime - lastButtonPress > BUTTON\_COOLDOWN

177 int delta = rightPressed ? 1 : -1;

178 lastButtonPress = currentTime;

179 lastSettingActivity = currentTime;

180 lastRepeatTime = currentTime;

181

182 if (currentScreen == MAIN) {

183 if (isSettingMode) {

184 if (!isAdjusting) {

185 currentMenuItem = (currentMenuItem + delta + 4) % 4;

186 } else {

187 switch (currentMenuItem) {

188 case 0: tempMin += 0.5 \* delta; break;

189 case 1: tempMax += 0.5 \* delta; break;

190 case 2: humMin += 1.0 \* delta; break;

191 case 3: humMax += 1.0 \* delta; break;

192 }

193 }

194 }

195 } else if (currentScreen == TIME) {

196 adjustTimeValue(delta);

197 }

198 }

199 }

200

201 void adjustTimeValue(int delta) {

202 switch (timeSettingMode) {

203 case HOUR:

204 adjustTime(delta \* 3600);

205 break;

206 case MINUTE:

207 adjustTime(delta \* 60);

208 break;

209 case DAY:

210 setTime(hour(), minute(), day() + delta, month(), year());

211 break;

212 case MONTH:

213 setTime(hour(), minute(), day(), month() + delta, year());

214 break;

215 case YEAR:

216 setTime(hour(), minute(), day(), month(), year() + delta);

217 break;

218 default:

219 break;

220 }

221 }

222

223 void displaySettings() {

224 display.clearDisplay();

225 display.setTextSize(1);

226 display.setTextColor(SSD1306\_WHITE);

227

228 display.setCursor(0, 0);

229 display.print("Settings:");

230

231 display.setCursor(0, 16);

232 display.print("TempMin: ");

233 display.print(tempMin, 1);

234 if (currentMenuItem == 0) display.print(isAdjusting ? " <" : " \*");

235

236 display.setCursor(0, 26);

237 display.print("TempMax: ");

238 display.print(tempMax, 1);

239 if (currentMenuItem == 1) display.print(isAdjusting ? " <" : " \*");

240

241 display.setCursor(0, 36);

242 display.print("HumMin: ");

243 display.print(humMin, 1);

244 if (currentMenuItem == 2) display.print(isAdjusting ? " <" : " \*");

245

246 display.setCursor(0, 46);

247 display.print("HumMax: ");

248 display.print(humMax, 1);

249 if (currentMenuItem == 3) display.print(isAdjusting ? " <" : " \*");

250

251 // Удалена строка с настройкой времени

252

253 display.display();

254 }

255

256 void displayMainScreen(float temperature, float humidity) {

257 display.clearDisplay();

258 display.setTextSize(1);

259 display.setTextColor(SSD1306\_WHITE);

260

261 // Температура

262 display.setCursor((SCREEN\_WIDTH - 11 \* 6) / 2, 10); // Центрирование

263 display.print("Temp: ");

264 display.print(temperature, 1);

265 display.print("C");

266

267 // Влажность

268 display.setCursor((SCREEN\_WIDTH - 11 \* 6) / 2, 25); // Центрирование

269 display.print("Hum: ");

270 display.print(humidity, 1);

271 display.print("%");

272

273 // Уровень шума

274 display.setCursor((SCREEN\_WIDTH - (6 + noiseDescription.length()) \* 6);

275 display.print("Noise: ");

276 display.print(noiseDescription);

277

278 display.display();

279 }

280

281 void displayTimeScreen() {

282 display.clearDisplay();

283 display.setTextSize(2);

284 display.setTextColor(SSD1306\_WHITE);

285

286 // Time

287 char timeStr[6];

288 sprintf(timeStr, "%02d:%02d", hour(), minute());

289 int16\_t x1, y1;

290 uint16\_t w, h;

291 display.getTextBounds(timeStr, 0, 0, &x1, &y1, &w, &h);

292 display.setCursor((SCREEN\_WIDTH - w) / 2, 10);

293 if (timeSettingMode == HOUR) display.print(">");

294 display.print(timeStr);

295 if (timeSettingMode == MINUTE) display.print("<");

296

297 // Date

298 display.setTextSize(1);

299 char dateStr[11];

300 sprintf(dateStr, "%02d/%02d/%04d", day(), month(), year());

301 display.getTextBounds(dateStr, 0, 0, &x1, &y1, &w, &h);

302 display.setCursor((SCREEN\_WIDTH - w) / 2, 40);

303 display.print(dateStr);

304

305 // Уменьшенные полоски для выбора даты

306 if (timeSettingMode == DAY) display.drawLine(35, 50, 47, 50);

307 else if (timeSettingMode == MONTH) display.drawLine(53, 50, 65, 50);

308 else if (timeSettingMode == YEAR) display.drawLine(71, 50, 87, 50);

309

310 display.display();

311 }

312

313 void adjustTime(int seconds) {

314 time\_t t = now() + seconds;

315 setTime(t);

316 }

317

318 void displayEmojiScreen(float temperature, float humidity) {

319 display.clearDisplay();

320 display.setTextSize(1);

321 display.setTextColor(SSD1306\_WHITE);

322 display.setCursor(0, 0);

323

324 // Определяем состояние на основе температуры и влажности

325 bool tempOk = (temperature >= tempMin && temperature <= tempMax);

326 bool humOk = (humidity >= humMin && humidity <= humMax);

327

328 // Рисуем основу смайлика (круг и глаза)

329 display.drawCircle(64, 32, 20, SSD1306\_WHITE); // Лицо

330 display.drawPixel(58, 28, SSD1306\_WHITE); // Левый глаз

331 display.drawPixel(70, 28, SSD1306\_WHITE); // Правый глаз

332

333 // Определяем эмоцию смайлика

334 if (tempOk && humOk) {

335 // Веселый смайлик

336 display.drawLine(58, 38, 70, 38, SSD1306\_WHITE);

337 display.drawLine(58, 39, 70, 39, SSD1306\_WHITE);

338 } else if (!tempOk && !humOk) {

339 // Грустный смайлик

340 display.drawLine(58, 44, 70, 38, SSD1306\_WHITE);

341 display.drawLine(58, 45, 70, 39, SSD1306\_WHITE);

342 } else {

343 // Нейтральный смайлик

344 display.drawLine(58, 38, 70, 38, SSD1306\_WHITE);

345 }

346

347 display.display();

348 }

349

350 void loop() {

351 handleButtons();

352

353 float temperature = dht.readTemperature();

354 float humidity = dht.readHumidity();

355

356 if (currentScreen == MAIN) {

357 if (isSettingMode) {

358 displaySettings();

359 // Проверка таймаута настройки

360 if (millis() - lastSettingActivity > SETTING\_TIMEOUT) {

361 isSettingMode = false;

362 isAdjusting = false;

363 saveSettingsToEEPROM();

364 }

365 } else {

366 // Измерение шума

367 unsigned long startMillis = millis();

368 int signalMax = 0;

369 int signalMin = 1024;

370 while (millis() - startMillis < SAMPLE\_WINDOW) {

371 int sample = analogRead(SOUND\_SENSOR\_PIN);

372 if (sample < 1024) {

373 if (sample > signalMax) {

374 signalMax = sample;

375 } else if (sample < signalMin) {

376 signalMin = sample;

377 }

378 }

379 }

380

381 peakToPeak = signalMax - signalMin;

382 decayingPeak = max(peakToPeak, decayingPeak \* DECAY\_FACTOR);

383

384 String noiseDescription = getNoiseLevelDescription(peakToPeak);

385 displayMainScreen(temperature, humidity, noiseDescription);

386 }

387 } else if (currentScreen == TIME) {

388 displayTimeScreen();

389 } else if (currentScreen == EMOJI) {

390 displayEmojiScreen(temperature, humidity);

391 }

392

393 // Переключение если не в режиме настройки и не в режиме регулировки

394 if (!isSettingMode && !isAdjusting && timeSettingMode == NONE) {

395 unsigned long currentTime = millis();

396 if (currentTime - lastScreenSwitch > SCREEN\_CHANGE\_INTERVAL) {

397 currentScreen = (Screen)((currentScreen + 1) % 3);

398 lastScreenSwitch = currentTime;

399 }

400 }

401

402 // Обновляем время каждую секунду

403 static unsigned long lastSecond = 0;

404 if (millis() - lastSecond >= 1000) {

405 lastSecond = millis();

406 adjustTime(1);

407 }

408

409 delay(50);

410 }

411

412 String getNoiseLevelDescription(int peakToPeak, float decayingPeak) {

413 if (peakToPeak > NOISE\_THRESHOLD\_HIGH) {

414 return "Loud";

415 } else if (peakToPeak > NOISE\_THRESHOLD\_LOW) {

416 return "Moderate";

417 } else {

418 return "Quiet";

419 }

420 }

ПРИЛОЖЕНИЕ Е

**(обязательное)**

**Перечень элементов**

ПРИЛОЖЕНИЕ Ж

**(обязательное)**

**Ведомость документов**