СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Микроклимат - один из важнейших показателей комфортности внутренней среды помещения. Главные его компоненты — такие параметры как температура и влажность воздуха, концентрация в нём углекислого и угарного газов. Нарушение допустимых границ температуры и влажности влекут за собой уменьшение эффективности людей, а активное дыхание в условиях высоких концентраций углекислого и тем более угарного газа, несут недомогания, головную боль и даже ущерб здоровью.

Во многих современных физкультурно-оздоровительных комплексах устанавливаются те или иные устройства контроля микроклимата. С помощью данных устройств возможно отслеживание микроклиматических параметров, а в последствии возможна и автоматизация обогрева, принудительной вентиляции помещения, системы увлажнения воздуха.

Одним из таких устройств является микропроцессорное устройство контроля параметров физкультурно-оздоровительного комплекса, которое реализовано в данном курсовом проекте.

Устройство, разрабатываемое в данном проекте, должно выполнять ряд задач:

– измерять температуру, влажность воздуха и выводить эту информацию на дисплей в градусах Цельсия и процентах;

– измерять концентрацию углекислого и угарного газов в воздухе и выводить эти значения на дисплей в миллионных долях (ppm);

– сигнализировать о превышении допустимых значений измеряемых параметров посредством светодиодной индикации;

– иметь органы управления для переключения отображаемой на дисплее информации и включения/отключения подсветки дисплея.

1. ОБЗОР ЛИТЕРАТУРЫ

1.1. Состав устройства

В состав микропроцессорного устройства контроля параметров физкультурно-оздоровительного комплекса входит:

– микроконтроллер;
– датчик температуры;
– датчик влажности;
– датчик углекислого газа;
– датчик угарного газа;
– жидкокристаллический дисплей;
– 4 светодиода;
– 4 резистора;
– 2 кнопки.

1.2. Микроконтроллер

Среди микроконтроллеров доступных на рынке были рассмотрены платы торговых марок Arduino и STM32. Их сравнительные характеристики представлены в таблице 1.1.

Информация для составления сравнительной таблицы была взята с источников [1–3].

Таблица 1.1

Модель	Arduino UNO	Arduino Mega 2560	STM32F401CB
Микроконтроллер	ATmega328	ATmega2560	ARM Cortex-M4
Тактовая частота, МГц	16	16	84
ОЗУ, КБ	2	8	64
Flash-память, КБ	32	256	128
Выходное напряжение, В	3,3; 5	3,3; 5	1,7; 3,6
Количество входов/выходов	20	70	36
Ток потребления, мА	47	50	137

1.3. Датчик температуры и влажности

На рынке распространены устройства, которые совмещают в себе и датчик температуры и датчик влажности. Одним из таких решений являются датчики серии DHT. Сравнение данных датчиков приведено в таблице 1.2.

Информация для составления сравнительной таблицы была взята с источника [4].

Таблица 1.2

Датчик	DHT11	DHT22
Диапазон измерения температуры, °C	0…+50	-40…+125
Точность измерения температуры, °C	±2	±0.5
Диапазон измерения относительной влажности воздуха, %	20…80	0…100
Точность измерения относительной влажности воздуха, %	±5	±2
Частота опроса сенсоров, Гц	1	0,5

1.4. Датчик углекислого газа

Существуют специализированные и неспециализированные датчики углекислого газа. Друг от друга их отличает степень селективности газов, на которые датчик реагирует, точность измерений.

Сравнение датчиков для определения концентрации углекислого газа приведено в таблице 1.3.

Сравнение основных параметров датчиков выполнено на основе источников [5, 6].

Таблица 1.3

Датчик	MH-Z19	Senseair S8	MQ135
Диапазон измерения, ppm	0 – 5000	400 – 2000	0 – 15000
Точность измерения, ppm	± 50	± 40	нет данных
Максимальный потребляемый ток, мА	150	300	150

1.5. Датчик угарного газа

На рынке датчиков измерения различных газов известны сенсоры серии MQ. Сравнение датчиков этой серии, пригодных для измерения концентрации угарного газа приведено в таблице 1.4.

Сравнение основных параметров датчиков выполнено на основе источников [7, 8].

Таблица 1.4

Датчик	MQ7	MQ9
Диапазон измерения, ppm	10 – 10000	10 – 1000
Максимальный потребляемый ток, мА	160	150

1.6. Дисплей

Для реализации вывода информации с датчиков в понятной для человека форме в проекте необходим дисплей. В подобных проектах спросом пользуются знакосинтезирующие жидкокристаллические дисплей. Такие дисплей могут одновременно отображать ограниченное количество символов.

В таблице 1.5 приведены сравнительные характеристики двух таких дисплеев.

Сравнение было произведено на основе источников [9, 10].

Таблица 1.5

Дисплей	LCD1602	LCD2004
Символов в строке	16	20
Количество строк	2	4
Напряжение питания, В	5	5
Максимальный потребляемый ток, мА	120	180

2. РАЗРАБОТКА СТРУКТУРЫ УСТРОЙСТВА

Структурная схема устройства приведена в приложении А.

2.1. Перечень блоков

В данном устройстве можно выделить 7 основных блоков:
1) блок датчика температуры и влажности;
2) блок датчика углекислого газа;
3) блок датчика угарного газа;
4) микроконтроллер;
5) модуль индикации;
6) блок управления;
7) блок отображения информации.

В состав блока индикации входят светодиоды и резисторы для ограничения тока через них.

В состав блока управления входят кнопки для переключения отображаемой на дисплее информации и включения/отключения подсветки дисплея.

2.2. Связи между блоками

Датчики снимают показания и предают эту информацию микроконтроллеру.

Микроконтроллер обрабатывает информацию, принятую от датчиков, и передаёт её блоку отображения информации. Регистрируя отклонение значений, считанных датчиками, от допустимых границ, микроконтроллер посылает сигналы блоку индикации.

Блок управления указывает, информацию от каких датчиков микроконтроллеру следует передавать блоку отображения информации и с какими параметрами должен работать блоку отображения информации.

Блок отображения информации позволяет воспринимать данные, переданные микроконтроллером в понятной для человека форме.

Блок индикации, получая информацию от микроконтроллера, сигнализирует о нарушении допустимых границ определёнными параметрами окружающей среды.

3. ОБОСНОВАНИЕ ВЫБОРА УЗЛОВ, ЭЛЕМЕНТОВ ФУНКЦИОНАЛЬНОЙ СХЕМЫ УСТРОЙСТВА

Функциональная схема устройства приведена в приложении Б.

3.1. Аппаратная платформа

Ключевыми факторами выбора аппаратной платформы были:
– доступность;
– простота загрузки прошивки;
– распространённость платы;
– гнездо для внешнего питания;

Учитывая вышесказанное были отвергнуты варианты плат, требующие внешний программатор для прошивки и платы не торговой марки Arduino, так как платы других производителей имеют меньше руководств, примеров использования и меньшее сообщество, что осложнило бы процесс разработки устройства.

Из плат Arduino были вынесены на рассмотрение 2 модели: Nano и Uno. Рассматривались также их дешёвые аналоги. Однако поскольку Arduino Nano не имеет собственного гнезда для внешнего питания, выбор пал на модель Uno. Сравнивая цену оригинала и аналога под названием Uno R3 SMD CH340G, выбор пал на аналог. Данная плата отличается от оригинала только микроконтроллером (CH340G у аналога против ATmega328 у оригинала). Другой микроконтроллер позволил сделать плату более доступной. В остальном характеристики аналога соответствуют плате Arduino Uno, характеристики которой были приведены в разделе 1.

3.2. Датчик температуры и влажности

В данной категории был выбор между двумя датчиками: DHT11 и DHT22.

Обладая большим диапазоном измерения и большей точностью, несмотря на большую цену, для реализации данного проекта был выбран датчик DHT22 на подложке со встроенным подтягивающим резистором, что облегчает применение датчика.

3.3. Датчик углекислого газа

В качестве датчика углекислого газа был выбран нетривиальный сенсор – MQ135, который не является специализированным датчиком для измерения уровня CO_2. Выбор на неспециализированный датчик пал из-за низкой его стоимости по сравнению со специализированным. Однако за меньшую стоимость приходится платить большей погрешностью измерений.

Датчик MQ135 реагирует не только на углекислый газ, но и на бензол, аммиак, окись азота, спирт и дым. Это значит, что он обладает низкой селективностью и при большой концентрации в воздухе указанных газов, датчик будет ложно сигнализировать о высоком уровне углекислого газа.

Основанием для использования данного датчика именно как средства измерения концентрации углекислого газа, является то, что диоксид углерода, он же CO_2 — четвертый по распространенности газ в атмосфере Земли. Остальные регистрируемые датчиком вещества в газообразном состоянии встречаются несопоставимо реже. Но при этом чувствительность ко всем этим газам у MQ135 почти одинаковая, что, позволяет использовать его в первую очередь как датчик CO_2.

Для того, чтобы сделать возможным измерение абсолютных значений концентрации CO_2 данным датчиком, необходимо провести его калибровку в среде с известной концентрацией углекислого газа и провести суточный прогрев. Данные процедуры были выполнены в полном объёме в ходе работы над проектом.

3.4. Датчик угарного газа

В качестве датчика угарного газа выбран датчик MQ7. Производителем заявлена его большая селективность по сравнению с датчиком MQ9 и больший диапазон измерения.

Однако данный датчик также, как и MQ135, необходимо калибровать в среде с известным уровнем угарного газа. Это является минусом данного датчика.

3.5. Дисплей

Вследствие малого количества выводимой информации был выбран дисплей LCD1602, способный отображать 2 строки по 16 символов каждая. Также данный дисплей имеет меньший ток потребления.

Для подключения дисплея была использована плата PCF8574 I2C для подключения дисплея к микроконтроллеру по шине I2C. Техническая документация к плате приведена в источнике [11].

4. РАЗРАБОТКА ПРИНЦИПИАЛЬНОЙ ЭЛЕКТРИЧЕСКОЙ СХЕМЫ МИКРОПРОЦЕССОРНОГО УСТРОЙСТВА КОНТРОЛЯ ПАРАМЕТРОВ ФИЗКУЛЬТУРНО-ОЗДОРОВИТЕЛЬНОГО КОМПЛЕКСА

Принципиальная схема микропроцессорного устройства контроля параметров физкультурно-оздоровительного комплекса приведена в приложении В.

4.1. Разработка системы питания

Для расчёта характеристик блока питания была составлена таблица 4.1.

Таблица 4.1

Название устройства	Рабочее напряжение, В	Максимальный потребляемый ток, мА	Максимальная потребляемая мощность, мВт
Uno R3 SMD CH340G	5	47	235
MQ135	5	150	750
MQ7	5	160	800
LCD1602	5	120	600
DHT22	5	2,5	12,5
Светодиод (4 шт)	5	80	400
Итого:		559,5	2797,5

Итого максимальный ток потребления устройства составил 559,5 мА, а максимальная потребляемая мощность: 2797,5 мВт. Такое значение тока не превышает максимально допустимый ток с выхода 5V платы, поэтому необходимость во внешнем питании компонентов схемы отсутствует.

Расчёт баланса мощностей производится по формуле:

$$\sum P_{\text{пр}} = \sum P_{\text{ист}}$$

где $P_{\text{ист}}$ – это мощность, отдаваемая источником питания, $P_{\text{ист}}$ – это потребляемая мощность. Необходимо выбрать источник питания, который будет выдавать необходимую для питания устройства мощность. Был выбран блок питания с выходным напряжением 9 В, выходным током 1 А. Для получения стабильного напряжения 5 В, используется встроенный в плату линейный стабилизатор напряжения AMS1117. Документация по данному стабилизатору [12]. Максимальный ток, способный отдать выход 5V платы равняется 800 мА. Итого максимальная мощность источника равняется 4000 мВт, что с запасом превышает потребляемую мощность.

4.2. Расчёт нагрузки светодиодов

В данном устройстве используются 4 светодиода. Для ограничения тока, проходящего через светодиод, используется резистор, номинал которого высчитан по формуле:

$$R = \frac{U_П - U_Д}{I_{ПР}},$$

где $U_П$ – напряжения питания, $U_Д$ – напряжение, падающее на светодиоде, $I_{ПР}$ – прямой ток светодиода.

В устройстве используются светодиоды со следующими параметрами: $U_Д$ = 2 В, $I_{ПР}$ = 20 мА.

Расчёт: R = (5 – 2) / 0,02 = 150 Ом.

За неимением резистора такого номинала в проекте используются резисторы номиналом 220 Ом. Применение большего сопротивления, чем требуется, не влечёт за собой негативных последствий для работоспособности светодиодов и устройства в целом.

4.3. Описание входов и выходов микроконтроллера

Для питания всех компонентов схемы используются выходы микроконтроллера 5V и GND.

К микроконтроллеру подключены 2 кнопки. Назначение первой кнопки: включать и отключать по нажатию подсветку дисплея. Назначение второй: переключать по нажатию отображаемую на дисплее информацию. Эти кнопки заведены на цифровые входы D4 и D5 соответственно. На данных входах активирован встроенный в плату подтягивающий резистор. Вследствие этого, когда кнопка не нажата, на соответствующем входе регистрируется логическая единица, в противном случае – логический ноль.

На цифровой вход D3 подан сигнал DAT с датчика DHT22.

На аналоговые входы A1 и A2 заведены сигналы A0 с датчиков MQ7 и MQ135 соответственно.

С цифровых выходов D8, D9, D10, D11 подаётся питание на светодиоды индикации.

Через аналоговые выходы A4 и A5 по интерфейсу I2C подключен дисплей устройства LCD1602.

5. РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

5.1. Требования к программе

Программа, управляющая микропроцессорным устройством контроля параметров физкультурно-оздоровительного комплекса, должна реализовывать следующий функционал:

– считывание с заданным периодом показаний с датчиков;

– вывод считанной информации на дисплей;

– включение светодиодной индикации при отклонении показаний от предельно допустимых;

– считывание состояния кнопок управления;

– отключение подсветки дисплея по нажатию первой управляющей кнопки;

– смена выводимой на дисплей информации по нажатию второй управляющей кнопки;

Исполняемый файл программы не должен превышать объём флеш-памяти микроконтроллера, равной 32 КБ, и не должен использовать объём оперативной памяти больший, чем 2 КБ.

5.2. Схема программы

Схема программы приведена в приложении Г.

Описание блоков:

1. Начало;
2. Условный оператор: если питание включено: да, иначе: нет
3. Произвести установку режимов работы выходов и входов с подтягивающим резистором;
4. Для функций измерения, индикации и вывода информации завести таймеры. Значения таймера для функции измерения: 300 мс, функции отображения информации: 300 мс, функции индикации красным светодиодом №1: 200 мс, функции индикации желтым светодиодом: 150 мс, функции индикации зеленым светодиодом: 100 мс, функции индикации красным светодиодом №2: 60 мс. Запустить таймеры;
5. Условный оператор: если питание включено: да, иначе: нет;
6. Условный оператор: если зарегистрировано нажатие первой кнопки управления: да, иначе: нет;
7. За состояние подсветки (включена/выключена) отвечает булева переменная. Инвертировать данную переменную;
8. Условный оператор: если таймер функции измерения истёк: да, иначе: нет;
9. Заново завести таймер;
10. Произвести измерения температуры, влажности, концентрации углекислого и угарного газа;

11. Условный оператор: если настало время включить красный светодиод №1: да, иначе: нет;

12. Условный оператор: если Значение температуры меньше 15°C или больше 25°C: да, иначе: нет;

13. Выключить красный светодиод №1;

14. Подать противоположный логический сигнал на выход, с которого питается светодиод;

15. Условный оператор: если настало время включить желтый светодиод: да, иначе: нет;

16. Условный оператор: если Значение влажности меньше 25% или больше 60%: да, иначе: нет;

17. Выключить желтый светодиод;

18. Инвертировать состояние желтого светодиода, установить таймер функции;

19. Условный оператор: если настало время включить зеленый светодиод: да, иначе: нет;

20. Условный оператор: если концентрация CO2 больше 1000 ppm: да, иначе: нет;

21. Выключить зелёный светодиод;

22. Инвертировать состояние зеленого светодиода;

23. Условный оператор: если настало время включить красный светодиод №2: да, иначе: нет;

24. Условный оператор: если концентрация CO больше 35 ppm: да, иначе: нет;

25. Выключить красный светодиод №2;

26. Инвертировать состояние красного светодиода №2, установить таймер функции;

27. Условный оператор: если зарегистрировано нажатие второй кнопки управления: да, иначе: нет;

28. Сменить режим отображения информации на дисплее. Если отображались данные о температуре и влажности, то будут отображаться концентрация углекислого и угарного газов и наоборот;

29. Дисплей может выводить либо информацию о температуре и влажности, либо о концентрации углекислого и угарного газов. Текущий режим отображения хранится в булевой переменной. Инвертировать эту переменную;

30. Условный оператор: если настало время вывести информацию: да, иначе: нет;

31. Вывести информацию на экран согласно булевой переменной, отвечающей за отображаемые данные;

32. Конец.

5.3. Программа управления устройством

Листинг кода программы приведен в приложении Д.

Программа управления микропроцессорного устройства контроля параметров физкультурно-оздоровительного комплекса построена на основе псевдо-потоков. Поскольку микроконтроллер CH340G не поддерживает многопоточность, для написания относительно больших программ используются псевдо-потоки: задачи, с таймером запуска.

Такая реализация позволяет не останавливать работу остальных частей программы, когда определённой функции нужно выждать время. В таком случае заводится таймер на время, которое необходимо выждать и выполнение программы продолжается. Когда на очередной итерации главного цикла программы управление перейдёт к данной функции, она проверит истёк ли её таймер. Если таймер не истёк: функция не выполняется, и программа продолжает выполнение, в противном случае: функция выполняется и заново заводит таймер.

Программа разбита на 2 исходных файла: main.ino, chars.h.

В файле main.ino находится основной код программы, в файле chars.h находятся пользовательские символы, предназначенные для вывода на дисплей.

Код обоих файлов снабжён развернутыми комментариями. Далее будет дано более общее описание исходного кода программ.

5.3.1. Файл main.ino

Описание исходного кода программы:

строки 1 – 6: подключение необходимых библиотек;

строки 9 – 13: инициализация переменных, констант и объекта датчика DHT22;

строки 16 – 18: инициализация переменных, объекта дисплея;

строки 21– 27: инициализация переменных, констант и объекта датчика MQ135;

строки 30 – 33: инициализация переменных, констант и объекта датчика MQ7;

строки 36 – 42: инициализация переменных и констант для поддержки управления с кнопок;

строки 45 – 48: выводы, которые питают светодиоды;

строки 51 – 57: инициализация псевдо-потоков;

строки 61 – 69: начало работы дисплея и инициализация пользовательских символов;

строки 71 – 72: установка встроенных подтягивающих резисторов на входы (блок 3);

строки 74 – 77: объявление выходов для светодиодов (блок 3);

строки 79 – 95: назначение функций псевдо-потокам и установка значений таймеров (блок 4);

строки 98 – 119: главный цикл программы;

строка 99: вызов функции мигания светодиодами LedIndication (блоки 11 – 26);

строка 100: вызов функции управления подсветкой дисплея DisplayBacklightControl (блок 7);

строки 101 – 104: анализ таймера псевдо-потока измерения MeasurementThread и его запуск, если таймер истёк (блоки 8 – 10);

строка 106: чтение состояния первой кнопки (красной) с исключением дребезга, с помощью функции DebounceButtonRead (блок 4);

строки 107 – 112: изменение переменной isFirstScreenDisplaying в случае нажатия кнопки (блок 7);

строки 115 – 118: анализ таймера псевдо-потока отображения инфорации DisplayingDataThread и его запуск, если таймер истёк (блок 30, 31);

строки 122 – 132: функция DisplayData, к которой привязан DisplayingDataThread. Вызывает либо функцию отображения температуры и влажности DisplayTemperatureAndHumidity, либо функцию отображения концентраций углекислого и угарного газов DisplayCO2andCO, в зависимости от переменной isFirstScreenDisplaying (блок 31);

строки 135 – 175: функция DisplayCO2andCO, выводящая на дисплей концентрации углекислого и угарного газов (блок 31);

строки 178 – 239: функция DisplayTemperatureAndHumidity, выводящая на дисплей температуру и влажность (блок 31).

строки 242 – 246: функция TemperatureLedBlink, закреплённая за потоком светодиода нарушения температурного режима. Выполняет мигание светодиодом в случае нарушения (блоки 11 –14);

строки 249 – 253: функция HumidityLedBlink, закреплённая за потоком светодиодом влажности. Выполняет мигание светодиодом в случае нарушения (блоки 15 – 18);

строки 256 – 260: функция CO2LedBlink, закреплённая за потоком светодиода углекислого газа. Выполняет мигание светодиодом в случае нарушения (блоки 19 – 22);

строки 263 – 267: функция COLedBlink, закреплённая за потоком светодиода угарного газа. Выполняет мигание светодиодом в случае нарушения (блоки 23 – 26);

строки 270 – 280: функция контроля подсветки дисплея DisplayBacklightControl (блоки 30, 31);

строки 283 – 304: функция чтения состояния кнопки с задержкой на дребезг DebounceButtonRead (блоки 6, 27);

строки 307 – 356: функция светодиодной индикации при превышении показаний датчиков допустимых значений LedIndication (блоки 11 – 26);

строки 359 – 373: функция снятия показаний с датчиков Measurement (блок 10);

5.3.1. Файл chars.h

строки 1 – 10: код символа «градус Цельсия»;
строки 12 – 21: код цифры уменьшенной цифры 2;
строки 23 – 32: код буквы «У»
строки 33 – 42: код буквы «П»
строки 43 – 52: код буквы «Л»
строки 53 – 62: код буквы «Ж»
строки 63 – 72: код буквы «Ь»

ЗАКЛЮЧЕНИЕ

В ходе работы над проектом было разработано микропроцессорное устройство контроля параметров физкультурно-оздоровительного комплекса. Устройство отслеживает и анализирует текущее значение температуры, влажности, концентрации углекислого и угарного газов. Информация выводится на дисплей. Светодиоды обеспечивают индикацию и оповещают о нарушениях норм. Кнопки управления меняют отображаемую информацию и регулируют подсветку дисплея.

Разработанное устройство обладает рядом достоинств:

1) Относительная дешевизна устройства;

2) Простота реализации и сборки;

3) Большое количество измеряемых параметров.

Весь функционал устройства реализован в полном объёме.

Однако устройство также обладает недостатками, связанных с датчиками серии MQ. Датчики могут реагировать на некоторые другие газы и выдавать ложную информацию. Также их работа зависит от калибровки.

В будущем можно доработать данное устройство и использовать его для автоматизации процессов вентилирования и обогрева помещения.

ЛИТЕРАТУРА

[1]. Arduino Uno Rev3 — Arduino Official Store [Электронный ресурс] – Электронные данные. – Режим доступа: https://store.arduino.cc/products/arduino-uno-rev3/

[2]. Arduino Mega 2560 Rev3 — Arduino Official Store [Электронный ресурс] – Электронные данные. – Режим доступа: https://store.arduino.cc/products/arduino-mega-2560-rev3

[3]. STM32F401CB [Электронный ресурс] – Электронные данные. – Режим доступа: https://www.st.com/en/microcontrollers-microprocessors/stm32f401cb.html

[4]. Сравнение датчиков DHT11, DHT22 и DHT21 [Электронный ресурс] – Электронные данные. – Режим доступа: https://voltiq.ru/dht11-dht22-and-dht21/

[5]. MH-Z14A - датчик углекислого газа, сравнение с конкурентами [Электронный ресурс] – Электронные данные. – Режим доступа: https://mysku.ru/blog/china-stores/72411.html

[6]. Technical data MQ-135 gas sensor [Электронный ресурс] – Электронные данные. – Режим доступа: http://amperkot.ru/static/3236/uploads/datasheets/MQ-135.pdf

[7]. Датчик газа MQ9 (угарный газ, углеводородные газы) [Электронный ресурс] – Электронные данные. – Режим доступа: https://amperka.ru/product/gas-sensor-mq9

[8]. Датчик угарного газа MQ7 [Электронный ресурс] – Электронные данные. – Режим доступа: https://3d-diy.ru/wiki/arduino-datchiki/datchik-ugarnogo-gaza-mq7

[9]. WaveShare LCD1602 [Электронный ресурс] – Электронные данные. – Режим доступа: https://www.waveshare.com/datasheet/LCD_en_PDF/LCD1602.pdf

[10]. WaveShare LCD2004 [Электронный ресурс] – Электронные данные. – Режим доступа: https://www.waveshare.com/datasheet/LCD_en_PDF/LCD2004.pdf

[11]. PCF8574; PCF8574A Remote 8-bit I/O expander for I2C-bus with interrupt [Электронный ресурс] – Электронные данные. – Режим доступа: https://www.nxp.com/docs/en/data-sheet/PCF8574_PCF8574A.pdf

[12]. AMS1117 800mA LOW DROPOUT VOLTAGE REGULATOR [Электронный ресурс] – Электронные данные. – Режим доступа: https://static.chipdip.ru/lib/029/DOC001029248.pdf

ПРИЛОЖЕНИЕ А
(Обязательное)
Структурная схема

ПРИЛОЖЕНИЕ Б
(Обязательное)
Функциональная схема

ПРИЛОЖЕНИЕ В
(Обязательное)
Принципиальная схема

ПРИЛОЖЕНИЕ Г
(Обязательное)
Перечень элементов

ПРИЛОЖЕНИЕ Д
(Обязательное)
Схема программы

ПРИЛОЖЕНИЕ Е
(Обязательное)
Листинг кода

```
// main.ino
001 #include <Thread.h>  // подключение библиотеки ArduinoThread
002 #include <LiquidCrystal_I2C.h> // библиотека для работы с
    дисплеем
003 #include <DHT.h> // библиотека для работы с датчиком
    температуры и важности DHT22
004 #include <MQ135.h> // библиотека для работы с датчиком MQ135
005 #include "MQ7.h" // библиотека для работы с датчиком MQ7
006 #include "chars.h" // пользовательские символы
007
008 // -------DHT22-------
009 #define DHTPIN 3 // пин для сигнала поступающего с датчика
    DHT22
010 #define DHTTYPE DHT22 // модель датчика
011 DHT dht(DHTPIN, DHTTYPE); // инициализация объекта датчика
012 float temperature; // переменная для хранения значений
    температуры, снятых с датчика
013 float humidity; // переменная для хранения значений
    влажности, снятых с датчика
014
015 // -------LCD-------
016 LiquidCrystal_I2C lcd(0x27, 16, 2); // задаем адрес и
    размерность дисплея
017 bool lightOn = true; // переменная подсветки
018 bool isFirstScreenDisplaying = true; // переменная
    отображения экранов
019
020 // -------MQ135-------
021 #define PIN_MQ135 A2 // пин датчика MQ135
022 MQ135 mq135_sensor(PIN_MQ135, 130, 10); // инициализация
    объекта датчика MQ135
023 float rzero;
024 float correctedRZero;
025 float resistance;
026 float CO2ppm; // CO2 в ppm
027 float correctedCO2ppm; // откоректироанное значение CO2 в
    ppm (с поправкой на влажность и температуру)
028
029 // -------MQ7-------
030 #define PIN_MQ7 A1 // Пин к которому подключен MQ7
031 #define VOLTAGE 5 // Напряжение поданное на нагреватель
032 MQ7 mq7(PIN_MQ7, VOLTAGE); // инициализация объекта датчика
    MQ7
033 float COppm; // CO2 в ppm
034
035 // -------Кнопки-------
036 const int whiteButtonPin = 4; // белая кнопка на выводе 4
037 const int redButtonPin = 5; // красная кнопка на выводе 5
```

```
038 int buttonState[2]; // текущее состояние вывода для кнопки
039 bool buttonStateChanged[] = {false, false}; // изменилось ли
    состояние кнопки
040 int lastButtonState[] = {HIGH, HIGH}; // предыдущее
    состояние вывода для кнопки
041 unsigned long lastDebounceTime[] = {0, 0}; // последнее
    время
042 unsigned long debounceDelay = 30; // задержка
043
044 // -------Светодиоды-------
045 const int TemperatureLedPin = 8; // светодиод наружения
    температурного режима
046 const int HumidityLedPin = 9; // ... влажности
047 const int CO2LedPin = 10; // ... углекислого газа
048 const int COLedPin = 11; // ... угарного газа
049
050 // -------Потоки-------
051 Thread TemperatureLedThread = Thread(); // поток управления
    светодиодом нарушения температурного режима
052 Thread HumidityLedThread = Thread(); // поток управления
    светодиодом влажности
053 Thread CO2LedThread = Thread(); // поток управления
    светодиодом концентрации CO2
054 Thread COLedThread = Thread(); // поток управления
    светодиодом концентрации CO
055
056 Thread MeasurementThread = Thread(); // поток для снятия
    показаний с датчиков
057 Thread DisplayingDataThread = Thread(); // поток для
    отображения данных
058
059 void setup() {
060   dht.begin(); // инициализация dht22
061   lcd.begin(); // Инициализация lcd
062   lcd.createChar(5, degree); // Добавление пользовательских
      символов
063   lcd.createChar(6, P); // буква "П"
064   lcd.createChar(2, L); // буква "Л"
065   lcd.createChar(3, J); // буква "Ж"
066   lcd.createChar(7, MZ); // мягкий знак
067   lcd.createChar(8, U); // буква "У"
068   lcd.createChar(4, twoIndex);
069   lcd.backlight(); // включение подсветки
070
071   Serial.begin(9600);
072
073   pinMode(whiteButtonPin, INPUT_PULLUP); // пин белой кнопки
      устанавливаем как вход с подтягивающим резистором
074   pinMode(redButtonPin, INPUT_PULLUP); // пин белой кнопки
      устанавливаем как вход с подтягивающим резистором
075
076   pinMode(TemperatureLedPin, OUTPUT);   // объявляем
```

```
    TemperatureLedPin как выход.
077   pinMode(HumidityLedPin, OUTPUT);   // объявляем
    HumidityLedPin как выход.
078   pinMode(CO2LedPin, OUTPUT);   // объявляем CO2LedPin как
    выход.
079   pinMode(COLedPin, OUTPUT);   // объявляем COLedPin как
    выход.
080
081   TemperatureLedThread.onRun(TemperatureLedBlink);  //
    назначаем потоку задачу
082   TemperatureLedThread.setInterval(200); // задаём интервал
    срабатывания, мсек
083
084   HumidityLedThread.onRun(HumidityLedBlink);  // назначаем
    потоку задачу
085   HumidityLedThread.setInterval(150); // задаём интервал
    срабатывания, мсек
086
087   CO2LedThread.onRun(CO2LedBlink);  // назначаем потоку
    задачу
088   CO2LedThread.setInterval(100); // задаём интервал
    срабатывания, мсек
089
090   COLedThread.onRun(COLedBlink);  // назначаем потоку задачу
091   COLedThread.setInterval(60); // задаём интервал
    срабатывания, мсек
092
093   MeasurementThread.onRun(Measurement);  // назначаем потоку
    задачу
094   MeasurementThread.setInterval(300); // задаём интервал
    срабатывания, мсек
095
096   DisplayingDataThread.onRun(DisplayData);  // назначаем
    потоку задачу
097   DisplayingDataThread.setInterval(300); // задаём интервал
    срабатывания, мсек
098 }
099
100 void loop() {
101   DisplayBacklightControl(); // функция управления
    подстветкой дисплея
102   if (MeasurementThread.shouldRun()) // пора ли снимать
    показания с датчиков
103   {
104     MeasurementThread.run(); // снимаем показания с датчиков
105   }
106
107   LedIndication(); // функия мигания светодиодами
108   DebounceButtonRead(redButtonPin); // чтение состояния
    красной кпоки
109   if (buttonStateChanged[1] && buttonState[1] == LOW) //
    если состояние кнопки изменилась и она нажата
```

```
110   {
111     buttonStateChanged[1] = false; // изменение состояния
      зафиксировано, и с тех пор оно не менялось
112     lcd.clear(); // очистка дисплея
113     isFirstScreenDisplaying = !isFirstScreenDisplaying; //
      инвертируем переменную, привязанную к кнопке
114   }
115
116   // если настало время запускаться потоку отображения
      инфорации
117   if (DisplayingDataThread.shouldRun())
118   {
119     DisplayingDataThread.run(); // запускаем поток
120   }
121 }
122
123 // выбираем какие данные отображеть в зависиости от
    переменной isFirstScreenDisplaying
124 void DisplayData()
125 {
126   if (isFirstScreenDisplaying) // если нужно показывать
      первый экран с температурной и влажностью
127   {
128     DisplayTemperatureAndHumidity(); // отображаем
        темпертуру и влажность
129   }
130   else
131   {
132     DisplayCO2andCO(); // отображаем концентрации
        углекислого и угарного газа
133   }
134 }
135
136 // отображаем концентрации углекислого и угарного газа
137 void DisplayCO2andCO()
138 {
139   int CO2 = (int)correctedCO2ppm; // отбрасываем дробную
      часть числа
140   int digitNumber = (CO2 == 0 ? 1 : int (log10(CO2) + 1));
      // вычисляем количество символов в числе
141   int cursorPosition = 12 - digitNumber; // вычисляем
      позицию курсора для числа
142   lcd.setCursor(0, 0); // устанавливаем курсор в точку (0,
      0)
143   lcd.print("CO"); // пишем постоянный текст
144
145   lcd.setCursor(2, 0); // устанавливаем курсор в точку (2,
      0)
146   lcd.write((byte)4); // 2
147
148   lcd.setCursor(cursorPosition, 0); // устанавливаем курсор
      на высчинанную позицию
```

```
149   lcd.print(CO2); // выводим концентрацию CO2
150
151   // очищаем в цикле неиспользуемые клетки
152   for (int i = 3; i < cursorPosition; i++)
153   {
154       lcd.setCursor(i, 0);
155       lcd.print(" ");
156   }
157
158   lcd.setCursor(12, 0); // устанавливаем курсор в точку (12,
      0)
159   lcd.print(" ppm"); // пишем постоянный текст
160
161   int CO = (int)COppm; // отбрасываем дробную часть числа
162   digitNumber = (CO == 0 ? 1 : int (log10(CO) + 1)); //
      вычисляем количество символов в числе
163   cursorPosition = 12 - digitNumber; // вычисляем позицию
      курсора для числа
164   lcd.setCursor(0, 1); // устанавливаем курсор в точку (0,
      1)
165   lcd.print("CO"); // пишем постоянный текст
166   lcd.setCursor(cursorPosition, 1); // устанавливаем курсор
      на высчинанную позицию
167   lcd.print(CO); // выводим концентрацию CO
168
169   // очищаем в цикле неиспользуемые клетки
170   for (int i = 2; i < cursorPosition; i++)
171   {
172       lcd.setCursor(i, 1);
173       lcd.print(" ");
174   }
175
176   lcd.setCursor(12, 1); // устанавливаем курсор в точку (12,
      1)
177   lcd.print(" ppm"); // пишем постоянный текст
178 }
179
180 void DisplayTemperatureAndHumidity() // выводим показания
      влажности и температуры
181 {
182   // выводим строку "ВЛАЖНОСТЬ"
183   lcd.setCursor(0, 0);
184   lcd.print("В");
185
186   lcd.setCursor(1, 0);
187   lcd.write((byte)2); // буква "Л"
188
189   lcd.setCursor(2, 0);
190   lcd.print("А");
191
192   lcd.setCursor(3, 0);
193   lcd.write((byte)3); // буква "Ж"
```

```
194
195   lcd.setCursor(4, 0);
196   lcd.print("НОСТ");
197
198   lcd.setCursor(8, 0);
199   lcd.write((byte)7); // мягкий знак
200
201   int intHumidity = (int)humidity; // отбрасываем дробную
      часть числа
202   int digitNumber = (intHumidity == 0 ? 1 : int
      (log10(intHumidity) + 1)); // вычисляем количество
      символов в числе
203   int cursorPosition = 15 - digitNumber; // вычисляем
      позицию курсора для числа
204
205   lcd.setCursor(cursorPosition, 0);
206   lcd.print(intHumidity); // выводим показания влажности
207   // очищаем в цикле неиспользуемые клетки
208   for (int i = 13; i < cursorPosition; i++)
209   {
210       lcd.setCursor(i, 0);
211       lcd.print(" ");
212   }
213
214   lcd.setCursor(15, 0);
215   lcd.print("%");
216
217   // выводим строку "ТЕМПЕРАТУРА"
218   lcd.setCursor(0, 1);
219   lcd.print("ТЕМ");
220
221   lcd.setCursor(3, 1);
222   lcd.write((byte)6); // буква "П"
223
224   lcd.setCursor(4, 1);
225   lcd.print("ЕРАТ");
226
227   lcd.setCursor(8, 1);
228   lcd.write((byte)8); // буква "У"
229
230   lcd.setCursor(9, 1);
231   lcd.print("РА");
232
233   lcd.setCursor(12, 1);
234   lcd.print(temperature,0); // выводим показания температуры
235
236   lcd.setCursor(14, 1); // круг, обозначающий градусы
237   lcd.write((byte)5);
238
239   lcd.setCursor(15, 1);
240   lcd.print("C");
241 }
```

```
242
243 // Поток светодиода нарушения температурного режима:
244 void TemperatureLedBlink() {
245   static bool ledStatus = false;    // состояние светодиода
      Вкл/Выкл
246   ledStatus = !ledStatus;           // инвертируем состояние
247   digitalWrite(TemperatureLedPin, ledStatus);  //
      включаем/выключаем светодиод
248 }
249
250 // Поток светодиода влажности:
251 void HumidityLedBlink() {
252   static bool ledStatus = false;    // состояние светодиода
      Вкл/Выкл
253   ledStatus = !ledStatus;           // инвертируем состояние
254   digitalWrite(HumidityLedPin, ledStatus);  //
      включаем/выключаем светодиод
255 }
256
257 // Поток светодиода CO2:
258 void CO2LedBlink() {
259   static bool ledStatus = false;    // состояние светодиода
      Вкл/Выкл
260   ledStatus = !ledStatus;           // инвертируем состояние
261   digitalWrite(CO2LedPin, ledStatus);  // включаем/выключаем
      светодиод
262 }
263
264 // Поток светодиода CO:
265 void COLedBlink() {
266   static bool ledStatus = false;    // состояние светодиода
      Вкл/Выкл
267   ledStatus = !ledStatus;           // инвертируем состояние
268   digitalWrite(COLedPin, ledStatus);  // включаем/выключаем
      светодиод
269 }
270
271 // Функция контроля подсветки дисплея
272 void DisplayBacklightControl()
273 {
274   DebounceButtonRead(whiteButtonPin); // чтение состояния
      белой кпоки
275
276   if (buttonStateChanged[0] && buttonState[0] == LOW) //
      если состояние кнопки изменилось
277   {
278     buttonStateChanged[0] = false;
279     lightOn = !lightOn;           // инвертируем состояние
280     lightOn ? lcd.backlight() : lcd.noBacklight();  //
        включаем/выключаем светодиод
281   }
282 }
```

```
283
284 // Чтение состояния кнопки с задержкой на дребезг
285 void DebounceButtonRead(int buttonPin)
286 {
287   int indexOfButton = buttonPin - whiteButtonPin; // индекс
      текущей кпоки
288   int readState = digitalRead(buttonPin); // считываем
      состояние кнопки
289   if (readState != lastButtonState[indexOfButton]) // если
      состояние изменилось (дребезг или нажатие)
290   {
291     lastDebounceTime[indexOfButton] = millis(); //
        сбрасываем таймер
292   }
293
294   // если нажали кнопку, то ожидаем, чтобы исключить дребезг
295   if ((millis() - lastDebounceTime[indexOfButton]) >
      debounceDelay)
296   {
297     buttonStateChanged[indexOfButton] = readState !=
        buttonState[indexOfButton]; // если состояние кнопки
        изменилось
298     if (buttonStateChanged[indexOfButton])
299     {
300       buttonState[indexOfButton] = readState; // записываем
          новое состояние кнопки
301     }
302   }
303
304   // сохраняем состояние кнопки. В следующий раз в цикле это
      станет значением lastButtonState:
305   lastButtonState[indexOfButton] = readState;
306 }
307
308 // Функция светодиодной индикации при превышении показаний
    датчиков допустимых значений
309 void LedIndication()
310 {
311   if (TemperatureLedThread.shouldRun()) // светодиод
      температуры
312   {
313     if ((int)temperature > 25 || (int)temperature < 15)
314     {
315       TemperatureLedThread.run(); // запускаем поток
316     }
317     else
318     {
319       digitalWrite(TemperatureLedPin, LOW);  // выключаем
          светодиод
320     }
321   }
322
```

```
323   if (HumidityLedThread.shouldRun()) // светодиод влажности
324   {
325     if ((int)humidity > 60 || (int)humidity < 25)
326     {
327       HumidityLedThread.run(); // запускаем поток
328     }
329     else
330     {
331       digitalWrite(HumidityLedPin, LOW);  // выключаем
          светодиод
332     }
333   }
334
335   if (CO2LedThread.shouldRun()) // светодиод CO2
336   {
337     if ((int)correctedCO2ppm > 1000)
338     {
339       CO2LedThread.run(); // запускаем поток
340     }
341     else
342     {
343       digitalWrite(CO2LedPin, LOW);  // выключаем светодиод
344     }
345   }
346
347   if (COLedThread.shouldRun())  // светодиод CO
348   {
349     if ((int)COppm > 35)
350     {
351       COLedThread.run(); // запускаем поток
352     }
353     else
354     {
355       digitalWrite(COLedPin, LOW);  // выключаем светодиод
356     }
357   }
358 }
359
360 // Функция снятия показаний с датчиков
361 void Measurement()
362 {
363   // чтение значений температуры или влажности может
      занимать до 250 мс
364   // и эти данные могут быть не актуальны до 2 секунд. Т.е.
      это очень медленный датчик
365   humidity = dht.readHumidity(); // чтение влажности
366   temperature = dht.readTemperature(); // чтение температуры
367
368   rzero = mq135_sensor.getRZero(); // чтение R0
369   correctedRZero =
      mq135_sensor.getCorrectedRZero(temperature, humidity); //
      его корректировка на температуру и влажность
```

```
370   resistance = mq135_sensor.getResistance(); // расчёт
      сопративления
371   CO2ppm = mq135_sensor.getPPM(); // расчёт концентрации CO2
      в ppm
372   correctedCO2ppm =
      mq135_sensor.getCorrectedPPM(temperature, humidity); //
      корректировка концентрации на температуру и влажность
373
374   COppm = mq7.readPpm(); // расчёт концентрации CO в ppm
375 }

//chars.h
01 byte degree[8] =      // кодируем символ градуса
02 {
03   B00111,
04   B00101,
05   B00111,
06   B00000,
07   B00000,
08   B00000,
09   B00000,
10 };
11
12 byte twoIndex[8] = // уменьшенная цифра 2
13 {
14   B00000,
15   B00000,
16   B01100,
17   B10010,
18   B00100,
19   B01000,
20   B11110,
21 };
22
23 byte U[8] =      // буква У
24 {
25   B10001,
26   B10001,
27   B10001,
28   B10001,
29   B01111,
30   B10001,
31   B01110,
32 };
33 byte P[8] =      // буква П
34 {
35   B11111,
36   B10001,
37   B10001,
38   B10001,
39   B10001,
40   B10001,
```

```
41   B10001,
42 };
43 byte L[8] =      // буква Л
44 {
45   B00111,
46   B01001,
47   B10001,
48   B10001,
49   B10001,
50   B10001,
51   B10001,
52 };
53 byte J[8] =      // буква Ж
54 {
55   B10101,
56   B10101,
57   B10101,
58   B01110,
59   B10101,
60   B10101,
61   B10101,
62 };
63 byte MZ[8] =      // Ь
64 {
65   B10000,
66   B10000,
67   B10000,
68   B11110,
69   B10001,
70   B10001,
71   B11110,
72 };
```

ПРИЛОЖЕНИЕ Ж
(Обязательное)
Ведомость документов