Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Архитектура вычислительных систем

*К защите допустить:*

И.О. Заведующего кафедрой информатики

\_\_\_\_\_\_\_\_\_\_\_\_\_С. И. Сиротко

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

**АЛКОТЕСТЕР НА ARDUINO**

БГУИР КП 1-40 04 01 015 ПЗ

Студент В.Ю.Павлович

Руководитель А.А.Калиновская

Нормоконтролер А.А.Калиновская

Минск 2024

**CОДЕРЖАНИЕ**

[Введение 5](#_Toc183359205)

[1 Архитектура вычислительной системы 6](#_Toc183359206)

[1.1 Понятие микроконтроллера 6](#_Toc183359207)

[1.2 Платформа Arduino 6](#_Toc183359208)

[1.3 История развития плат *Arduino* 9](#_Toc183359209)

[1.4 Датчик паров спирта *MQ-3* 13](#_Toc183359210)

[1.5 *OLED*-дисплей 14](#_Toc183359211)

[1.6 Обоснование выбора вычислительной системы 16](#_Toc183359212)

[2 Платформа программного обеспечения 17](#_Toc183359213)

[2.1 Среда разработки (*IDE*) 17](#_Toc183359214)

[2.2 Язык программирования 19](#_Toc183359215)

[2.3 Основные версии *Arduino IDE* 20](#_Toc183359216)

[2.4 Обоснование выбора программной платформы 21](#_Toc183359217)

[3 Теоретическое обоснование разработки программного продукта 23](#_Toc183359218)

[3.1 Углубленный анализ датчика *MQ-3* 23](#_Toc183359219)

[3.2 Калибровка датчиков 24](#_Toc183359220)

[3.3 Пользовательский интерфейс 25](#_Toc183359221)

[4 Проектирование функциональных возможностей программы 27](#_Toc183359222)

[4.1 Обоснование и описание функциональных возможностей 27](#_Toc183359223)

[5 Архитектура разрабатываемой программы 29](#_Toc183359224)

[5.1 Подключение устройств к *Arduino* 30](#_Toc183359225)

[5.2 Общая структура программы 33](#_Toc183359226)

[5.3 Описание функциональной схемы программы 34](#_Toc183359227)

[5.4 Описание блок-схемы алгоритма программы 35](#_Toc183359228)

[5.5 Коды программных ошибок 36](#_Toc183359229)

[Заключение 37](#_Toc183359230)

[Список использованных источников 38](#_Toc183359231)

[Приложение А (Обязательное) Справка о проверке на заимствования 40](#_Toc183359232)

[Приложение Б (Обязательное) Листинг программного кода 41](#_Toc183359235)

[Приложение В (Обязательное) Функциональная схема алгоритма 44](#_Toc183359236)

[Приложение Г (Обязательное) Блок схема алгоритма 45](#_Toc183359237)

[Приложение Д (Обязательное) Графический интерфейс пользователя 46](#_Toc183359238)

[Приложение Е (Обязательное) Ведомость документов 47](#_Toc183359239)

# ВВЕДЕНИЕ

В современном обществе проблема безопасности на дорогах становится все более актуальной. Управление транспортным средством в состоянии алкогольного опьянения является одной из основных причин дорожно-транспортных происшествий. В связи с этим разработка эффективных методов контроля и предупреждения таких ситуаций приобретает особую значимость. Одним из возможных методов решения данной проблемы является использование алкотестеров, которые дают водителю возможность определить свое состояние перед тем, как садиться за руль, что поможет предотвратить возможные трагические последствия на дороге. Алкотестеры являются простыми в использовании устройствами, которые могут измерить уровень алкоголя в выдыхаемом воздухе, позволяя человеку самостоятельно оценить, готов ли он безопасно управлять автомобилем.

Приборы данного типа предназначены для измерения уровня алкоголя в крови человека по содержанию его паров в выдыхаемом воздухе. При этом их использование оправдано не только в случаях, когда человек выпил алкоголь, но и когда он употребил некоторые другие продукты, как квас или кефир, так как при их употреблении так же может немного повыситься уровень алкоголя в крови и проверка может показать, что человек находится в состоянии алкогольного опьянения. Таким образом, использование данного прибора может уберечь водителя от возможных проблем с законом.

Цель данной курсовой работы состоит в разработке простейшего алкотестера на базе Arduino с использованием двух датчиков MQ-3, который будет выводит на дисплей информацию о содержании паров спирта в воздухе. Наличие двух датчиков необходимо для обработки различных ошибок в работе алкотестера, например, выход одного из датчиков из строя.

Задачи курсового проекта:

1 Определить необходимые аппаратные компоненты для разрабатываемого устройства.

2 Разработать программное обеспечение для данного устройства.

Пояснительная записка оформлена в соответствии с СТП 01-2024 [1].

# 1 АРХИТЕКТУРА ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ

## Понятие микроконтроллера

Появление первых микроконтроллеров стало важной вехой в истории развития компьютерной техники. Именно с их появлением началась новая эра, связанная с массовым применением автоматизации в различных сферах управления. Микроконтроллер представляет собой сложное устройство, в котором на одном кристалле объединены центральный процессор, память и периферийные устройства. Эти компоненты позволяют микроконтроллерам выполнять разнообразные задачи по обработке данных и управлению внешними системами. Первый патент на однокристальную микроЭВМ был выдан в 1971 году инженерам Майклу Кокрэну и Гэри Буну, сотрудникам американской *Texas Instruments*. Именно они предложили на одном кристалле разместить не только процессор, но и память с устройствами ввода-вывода.

Одной из проблем работы с микроконтроллерами является необходимость знания схемотехники и устройства конкретного процессора и умения программировать на ассемблере для данного процессора. Также для разработки необходимы программаторы и отладчики, что долгое время не позволяло любителям использовать микроконтроллеры в своих проектах. Однако на сегодняшний день появились устройства, позволяющие работать с микроконтроллерами без наличия серьезной материальной базы и знания многих предметов. Примером такого устройства является проект Arduino, разработанный в Италии в 2005 году [2].

## Платформа Arduino

Устройства из семейства *Arduino* представляют собой наборы, состоящие из электронного блока и программного обеспечения. Электронный блок состоит из печатной платы с установленным микроконтроллером *ATMega* фирмы *Atmel* и набором элементов, необходимых для его работы. По сути электронный блок можно считать аналогом материнской платы современного компьютера. На нем имеются разъемы для подключения внешних устройств, а также разъем для связи с компьютером, по которому и осуществляется программирование микроконтроллера.

С 2008 года в компании-разработчике начался раскол, выразившийся в существовании двух независимых ветвей развития и продаж под одной торговой маркой: одна на сайте *arduino.cc*, другая на *arduino.org*. Докризисные изделия на обоих сайтах продаются под одинаковыми названиями. Набор новых изделий на сайтах различается. Также существует две ветви *Arduino* *IDE*, поддерживающие разный набор плат и библиотек. Одинаковые названия и пересекающиеся номера версий *IDE* вносят путаницу [3].

Под торговой маркой *Arduino* выпускается несколько плат с микроконтроллером и платы расширения (так называемые шилды). Большинство плат с микроконтроллером снабжены минимально необходимым набором обвязки для нормальной работы микроконтроллера (стабилизатор питания, кварцевый резонатор, цепочки сброса и т. п.).

*Arduino* и *Arduino*-совместимые платы спроектированы таким образом, чтобы их можно было при необходимости расширять, добавляя в устройство новые компоненты. Эти платы расширений подключаются к *Arduino* посредством установленных на них штыревых разъёмов. Существует ряд плат с унифицированным конструктивом, допускающим конструктивно жесткое соединение процессорной платы и плат расширения в стопку через штыревые линейки. Кроме того, выпускаются платы уменьшенных габаритов (например, *Nano*, *Lilypad*) и специальных конструктивов для задач робототехники. Независимыми производителями также выпускается большое количество всевозможных датчиков и исполнительных устройств, в той или иной степени совместимых с базовым конструктивом *Arduino*. На рисунке 1.1 представлена плата расширения *Sensor Shield* 4.

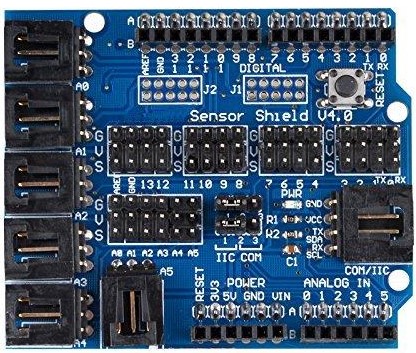


Рисунок 1.1 – Плата расширения *Sensor Shield* 4

Микроконтроллеры для *Arduino* отличаются наличием предварительно прошитого в них загрузчика (*bootloader*). С помощью этого загрузчика пользователь загружает свою программу в микроконтроллер без использования традиционных отдельных аппаратных программаторов. Загрузчик соединяется с компьютером через интерфейс *USB* (если он есть на плате) или с помощью отдельного переходника *UART-USB*. Поддержка загрузчика встроена в *Arduino IDE* и выполняется в один щелчок мыши [4].

На случай затирания загрузчика или покупки микроконтроллера без загрузчика разработчики предоставляют возможность прошить загрузчик в микроконтроллер самостоятельно. Для этого в *Arduino* *IDE* встроена поддержка нескольких популярных дешевых программаторов, а большинство плат *Arduino* имеет штыревой разъем для внутрисхемного программирования (*ICSP* для *AVR*, *JTAG* для *ARM*).

В *Arduino IDE* от компании, базирующейся на сайте *arduino.cc*, встроена возможность создания своих программно-аппаратных платформ. Этой возможностью пользуются сторонние компании, добавляющие в *Arduino* *IDE* свои наборы плат и компиляторов-загрузчиков к ним. Компания на сайте arduino.org не поддерживает такую возможность.

В линейке устройств *Arduino* в основном применяются микроконтроллеры *Atmel AVR*: *ATmega328*, *ATmega168*, *ATmega2560*, *ATmega32U4*, *ATTiny85* с частотой тактирования 16 или 8 МГц. В старых изделиях применялись *ATmega8*, *ATmega1280* и другие. Есть также платы на процессоре *ARM* *Cortex* *M*. Сторонние разработчики портировали в *Arduino* поддержку популярного *Wi-Fi* микроконтроллера *ESP8266*. Теперь компилировать и загружать прошивку для *ESP8266* со своими скетчами и поддержкой *Wi-Fi* можно прямо из *Arduino* *IDE*, получая одноплатную схему с поддержкой сети *Wi-Fi*.

Порты ввода-вывода микроконтроллеров оформлены в виде штыревых линеек. Никакого буферизирования, защиты, конвертации уровней или подтяжек, как правило, нет. Микроконтроллеры питаются от 5В или 3,3В, в зависимости от модели платы. Соответственно порты имеют такой же размах допустимых входных и выходных напряжений. Программисту доступны некоторые специальные возможности портов ввода-вывода микроконтроллеров, например, широтно-импульсная модуляция, аналогово-цифровой преобразователь, интерфейсы *UART*, *SPI*, *I2C*. Количество и возможности портов ввода-вывода определяются конкретным вариантом микропроцессорной платы.

Помимо портов на платах микроконтроллеров иногда устанавливается периферия в виде интерфейсов *USB* или *Ethernet*.

Сторонние производители выпускают широкую гамму датчиков и исполнительных устройств, подключаемых к *Arduino*. Например, гироскопы, компасы, манометры, гигрометры, термометры, релейные модули, индикаторы, клавиатуры и т. п. Всё это превращает *Arduino* в универсальное ядро системы, которое может быть сконфигурировано совершенно разнообразными способами.

Таким образом, можно выделить следующие преимущества платформы Arduino:

1 Программное обеспечение *Arduino* совместимо с операционными системами *Windows*, *Macintosh OSX* и *Linux*, в отличие от большинства аналогичных систем, работающих только в *Windows*.

2 *Arduino* предоставляет возможность как профессионалам, так и новичкам заниматься разработкой микропроцессорных устройств. Благодаря наличию готовых модулей и библиотек программ, создание функционирующих устройств становится гораздо проще даже для тех, кто не имеет опыта в электронике.

3 Семейство платформ *Arduino* включает различные модели, от компактных до более мощных, что позволяет выбрать наиболее подходящий вариант для конкретного проекта.

4 Для Arduino не нужен программатор. Всё сделано так, чтобы программирование *Arduino* для начинающих не составляло труда. Написанный код можно загрузить в микроконтроллер посредством *USB*-кабеля. Достигается это преимущество не каким-то встроенным уже заранее программатором, а специальной прошивкой – бутлоадером. Бутлоадер является специальной программой, которая запускается сразу после подключения и слушает, будут ли какие-то команды, прошивать ли кристалл, есть ли проекты *Arduino* или нет.

## История развития плат *Arduino*

В данном разделе будет рассмотрена история развития официальных плат *Arduino* по годам с перечислением их ключевых изменений.

*Arduino Diecimila* – первая версия платы *Arduino*, выпущенная в 2005 году. Она была разработана как простое и доступное средство для обучения программированию и электронике. Данная плата представлена на рисунке 1.2.



Рисунок 1.2 – Плата *Arduino Diecimila*

В 2006 году была представлена новая версия платы – *Arduino Duemilanove*, которая быстро завоевала популярность среди энтузиастов и разработчиков. Эта плата отличалась улучшенной совместимостью с различными устройствами и поддержкой нескольких типов микроконтроллеров, что значительно расширило её функциональность. Благодаря этим усовершенствованиям, *Duemilanove* стал основой для множества интересных и инновационных проектов в области электроники и программирования. Плата предоставила пользователям возможность легко интегрировать её в различные приложения, от простых до более сложных систем. Данная плата представлена на рисунке 1.3.

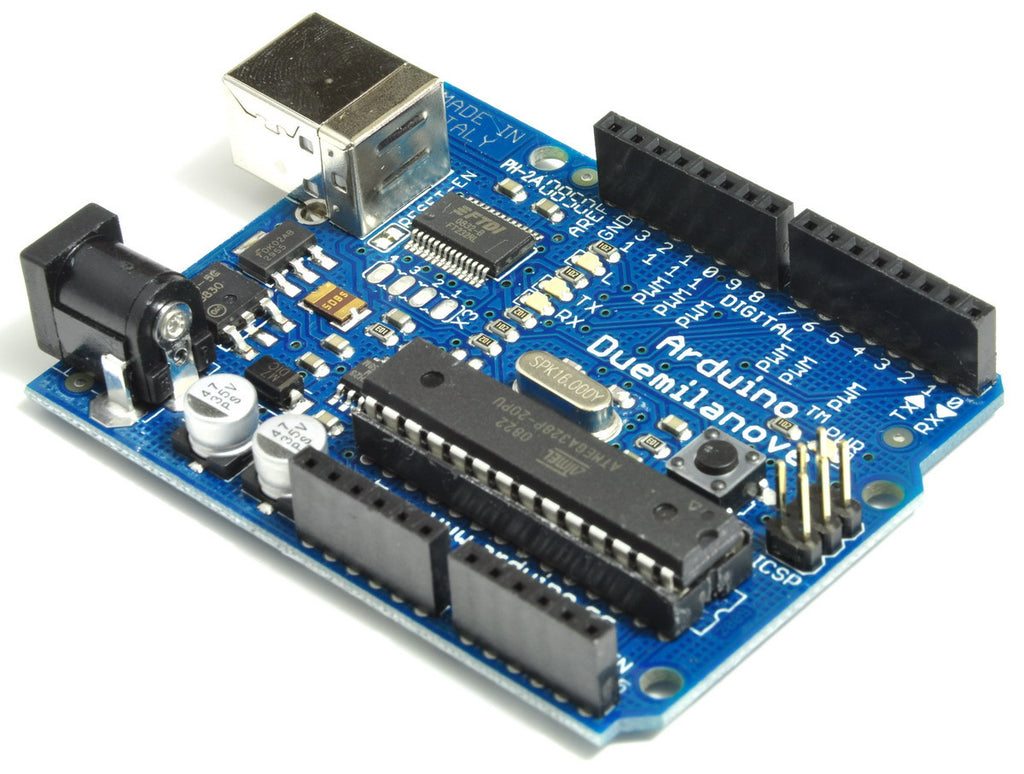


Рисунок 1.3 – Плата *Arduino Duemilanove*

*Arduino Nano* – это компактная плата, представленная в 2008 году, которая идеально подходит для работы в условиях ограниченного пространства. Благодаря своим минимальным размерам, она легко интегрируется в различные проекты, где важно экономить место. *Nano* использует микроконтроллеры *ATmega168* или *ATmega328*, что обеспечивает ей хорошую производительность при малом энергопотреблении. Эта плата стала особенно популярной среди разработчиков, работающих над малыми проектами и прототипами, благодаря своей универсальности и простоте в использовании. Данная плата представлена на рисунке 1.4.



Рисунок 1.4 – Плата *Arduino Nano*

*Arduino Uno* – это плата, которая пришла на смену Duemilanove и была представлена в 2009 году. Она основана на микроконтроллере *ATmega328*, который обеспечивает высокую производительность и широкие возможности для разработчиков. *Uno* быстро стала стандартом для большинства проектов благодаря своей простоте в использовании и обширному сообществу. Эта плата поддерживает множество библиотек и примеров, что делает её идеальным выбором как для новичков, так и для опытных пользователей. Данная плата представлена на рисунке 1.5.

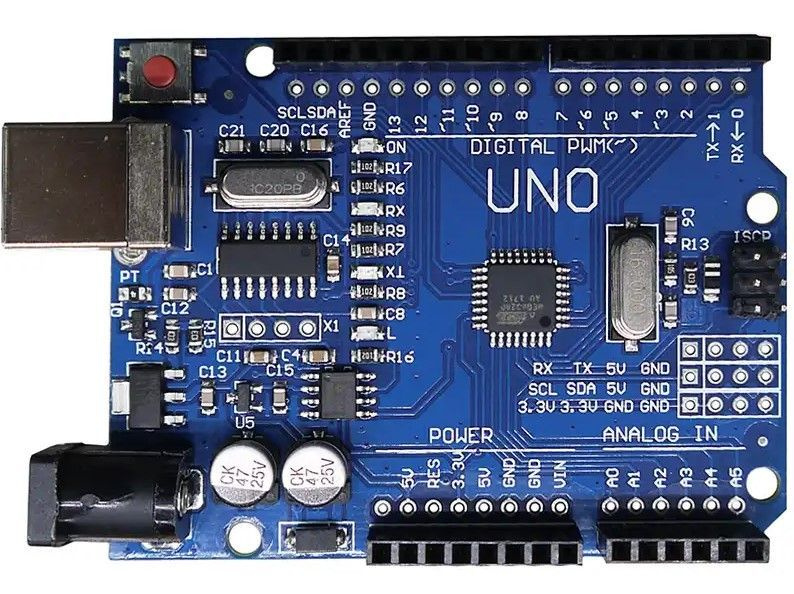


Рисунок 1.5 – Плата *Arduino UNO*

В 2010 году вышла новая плата – *Arduino Mega* с большим количеством входов/выходов (54 цифровых и 16 аналоговых). Она предназначена для более сложных проектов, требующих большей памяти и вычислительной мощности. Данная плата представлена на рисунке 1.6.



Рисунок 1.6 – Плата *Arduino Mega 2560*

*Arduino Leonardo* – это уникальная плата, представленная в 2011 году, которая выделяется своей способностью эмулировать *USB*-устройства, такие как клавиатуры и мыши. Эта функция значительно расширила возможности взаимодействия с компьютером, позволяя разработчикам создавать инновационные проекты, которые могут напрямую взаимодействовать с программным обеспечением на ПК. Благодаря использованию микроконтроллера *ATmega32U4*, *Leonardo* может отправлять данные через *USB*, что делает её отличным выбором для создания различных интерфейсов и управляемых устройств. Например, с помощью этой платы можно разработать проекты, которые автоматически вводят текст, управляют курсором или выполняют другие действия, как будто это делает обычная клавиатура или мышь. Данная плата представлена на рисунке 1.7.

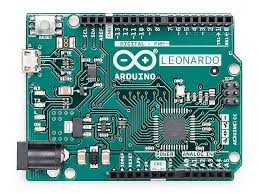


Рисунок 1.7 – Плата *Arduino Leonardo*

В 2013 году была представлена *Arduino Due* – первая плата в линейке *Arduino*, оснащённая 32-битным *ARM Cortex-M3* процессором. Это значительное обновление обеспечило пользователям более высокую производительность и расширенные возможности по сравнению с предыдущими моделями. *Arduino Due* предлагает увеличенную скорость обработки данных и большую ёмкость памяти, что делает её идеальной для сложных проектов, требующих значительных вычислительных ресурсов. Плата поддерживает множество современных интерфейсов, включая *USB*, *I2C* и *SPI*, что позволяет легко интегрировать её в различные системы и устройства. Данная плата представлена на рисунке 1.8.



Рисунок 1.8 – Плата *Arduino Due*

*Arduino MKR* – это серия плат, ориентированных на Интернет вещей (*IoT*), представленная в 2015 году. Эти платы оснащены встроенными модулями для беспроводной связи, такими как *Wi-Fi*, *LoRa* и *GSM*, что делает их идеальными для разработки проектов, связанных с удалённым мониторингом и управлением. Серия *MKR* предлагает компактный и энергоэффективный дизайн, что позволяет легко интегрировать платы в различные устройства и системы. Каждая плата в этой серии поддерживает стандарты подключения, обеспечивая простоту в создании сетевых приложений [5]. Данная плата представлена на рисунке 1.9.

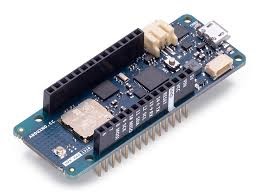


Рисунок 1.9 – Плата *Arduino MKR*

## Датчик паров спирта *MQ-3*

Модуль *MQ-3* – это газовый датчик, который обнаруживает присутствие таких газов, как угарный газ, гексан, сжиженный газ, метан и алкоголь. Он представляет собой датчик типа металл-оксид-полупроводник (*MOS*), который определяет концентрацию алкоголя в воздухе. Внешний вид данного датчика представлен на рисунке 1.10.



Рисунок 1.10 – Датчик паров спирта *MQ-3*

Датчик алкоголя *MQ-*3 рекомендуется использовать в небольшой разводной плате, поскольку он не совместим с макетной платой. Кроме того, он имеет два различных выхода, и его легко использовать. Его функция заключается в обеспечении двоичной индикации наличия алкоголя и его аналоговой концентрации в чистом воздухе.

Аналоговое выходное напряжение датчика (на аналоговом выводе) обычно обеспечивает различную пропорцию аналогового выходного напряжения в зависимости от концентрации алкоголя. Поэтому высокая концентрация алкоголя в чистом воздухе соответствует высокому выходному напряжению и наоборот.

Затем компаратор, например, высокоточный компаратор *LM393*, получает тот же аналоговый сигнал. Затем он подает сигнал на цифровой выход (*DO*).

Принцип работы датчика заключается в том, что при нагреве полупроводникового слоя диоксида олова (*SnO2*) до высоких температур, его поверхность поглощает кислород. Далее (в чистом воздухе) молекулы кислорода будут притягивать электроны из полосы проводимости в диоксиде олова. Следовательно, под поверхностью молекул *SnO2* образуется слой обеднения электронами, который создает потенциальный барьер. Таким образом, пленка *SnO2* препятствует протеканию электрического тока, поскольку обладает высоким сопротивлением. Однако, когда присутствующий спирт вступает в реакцию с адсорбированным кислородом, поверхностная плотность кислорода уменьшается. В результате потенциальный барьер также уменьшается. Затем диоксид олова принимает электроны, тем самым обеспечивая свободное прохождение тока через датчик. Таким образом, большое содержание спирта в воздухе снижает аналоговое сопротивление, тем самым увеличивая показания выходного напряжения [6].

Особенности датчика *MQ-3*:

1 Двоичная индикация – позволяет определить наличие или отсутствие алкоголя в окружающей среде.

2 Аналоговый выход – предоставляет информацию о концентрации алкоголя в виде изменяющегося аналогового напряжения.

3 Легкость использования – простое взаимодействие с Arduino и другими микроконтроллерами.

## *OLED*-дисплей

*OLED*-дисплеи – это технология отображения, использующая органические светодиоды для создания ярких и контрастных изображений. Их гибкость позволяет разрабатывать тонкие и легкие экраны, что особенно подходит для компактных устройств. Благодаря быстрому времени отклика и широкому углу обзора, *OLED*-дисплеи идеально подходят для отображения информации в реальном времени.

Ключевое отличие *OLED*-экранов от более распространенных жидкокристаллических вариантов в том, что *LCD* или *LED* требуют внешней подсветки. Такие дисплеи состоят из множества слоев, в результате чего толщина устройств увеличивается. На рисунке 1.11 представлен пример *OLED*-дисплея.



Рисунок 1.11 – *OLED*-дисплей

Обмен данными между микроконтроллером и дисплеем осуществляется по интерфейсу *I2C* (или *I2C*). *I2C* – последовательная асимметричная шина для связи между интегральными схемами внутри электронных приборов. Использует две двунаправленные линии связи (*SDA* и *SCL*). Шина представляет собой два проводника, а для управления интерфейсом достаточно одного микроконтроллера. Специальный встроенный фильтр способен справляться с всплесками, гарантируя сохранность обрабатываемой информации. Данный интерфейс подключения используется для низкоскоростной (до 100 кбит/с в стандартном режиме и 400 кбит/с в «быстром» режиме) передачи 8-битных данных между микроконтроллерами (или процессорами) и периферийными компонентами. Для передачи информации используется всего две линии: *SDA* — линия данных и *SCL* — линия синхронизации. Каждое устройство определяется как ведущее (*Master*) или ведомое (*Slave*), а также обладает уникальным (в пределах шины) адресом. Максимальное допустимое количество элементов, подсоединённых к одной шине, ограничивается максимальной емкостью шины — 400 пФ. Отношения ведущий-ведомый и приемник-передатчик не постоянны и зависят лишь от направления передачи данных в текущий момент времени. Обычно стандарт *I2C* подразумевает наличие в один момент времени только одного ведущего устройства, однако также допускается и наличие нескольких ведущих устройств в шине без нарушения работы системы. В этом случае для предотвращения возникновения ошибок и хаоса применяется процедура арбитража. Для корректной передачи данных, приемник должен подтверждать прием каждого байта от передатчика. Для этого в спецификации протокола обмена по шине *I2C* вводится специальный бит подтверждения, выставляемый приёмником на шину *SDA* после приема каждого 8-го бита данных. Передача каждых 8-ми бит данных считается успешной, если приемник отправляет бит подтверждения с низкий уровень сигнала по линии *SDA*. Импульс синхронизации для подтверждающего бита генерируется ведущим устройством сразу после переданных 8-ми бит данных [7].

Среди преимуществ *OLED*-дисплеев в проектах можно выделить энергоэффективность, что достигается тем, что каждый пиксель самостоятельно генерирует свет и не требует подсветки, поэтому *OLED*-экраны потребляют меньше энергии, особенно при отображении темных сцен или черного цвета. Также важным преимуществом является их способность отображать важные данные с высокой четкостью. Они также обеспечивают отличную видимость при низком уровне освещения, что особенно полезно в ночное время. Простота программирования на Arduino делает интеграцию *OLED*-дисплеев в устройства более доступной. Наличие большого количества библиотек для работы с данным типом устройств также упрощает разработку.

## Обоснование выбора вычислительной системы

Основные преимущества выбранных для выполнения данной курсовой работы устройств:

1 *Arduino* представляет собой готовый электронный модуль с установленным микроконтроллером и основным набором элементов. Эта платформа разработана с учетом возможности дальнейшего расширения, что дает возможность легко добавлять новые компоненты. Главным преимуществом является то, что разработчикам не нужно тратить время на создание законченных схем и модулей, что упрощает процесс разработки.

2 Датчики обеспечивают возможность обнаружения алкоголя и других газов в воздухе, а использование двух датчиков *MQ-3* позволяет корректно обработать ситуацию с выходом одного из датчиков из строя, что обеспечивает корректную работу устройства.

3 Отличительные характеристики *OLED*-дисплеев включают в себя энергоэффективность и высокую четкость изображения, а также возможность отображения данных в реальном времени и хорошую видимость в условиях низкой освещенности.

Таким образом, платформа Arduino и выбранные дополнительные устройства, такие как датчик паров спирта *MQ-3* и *OLED*-дисплей, являются оптимальными для реализации данного курсового проекта благодаря следующим преимуществам.

# 2 ПЛАТФОРМА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

В данном разделе будет подробно рассмотрена программная платформа, выбранная для выполнения курсового проекта. Будет проанализирована как среда разработки, так и язык программирования, который будет использоваться в процессе реализации проекта. Особое внимание будет уделено характеристикам каждой из выбранных технологий, их возможностям и ограничениям, а также тому, как они соответствуют требованиям проекта. В конце раздела будут приведены обоснования выбора данных средств разработки, включая их преимущества, совместимость и удобство использования в контексте поставленных задач. Это позволит понять, почему именно эти инструменты были выбраны для достижения оптимальных результатов в проекте.

## Среда разработки (*IDE*)

Среда разработки (*IDE*) играет значимую роль в процессе создания проекта. Удобный интерфейс, встроенные инструменты для отладки и тестирования, а также поддержка расширяемости делают работу более продуктивной. Выбор *IDE* должен учитывать особенности проекта и предпочтения команды разработчиков.

В качестве интегрированной среды разработки (*IDE*) программного кода для *Arduino* была выбрана *Arduino IDE*. Интерфейс *Arduino IDE* представлен на рисунке 2.1.

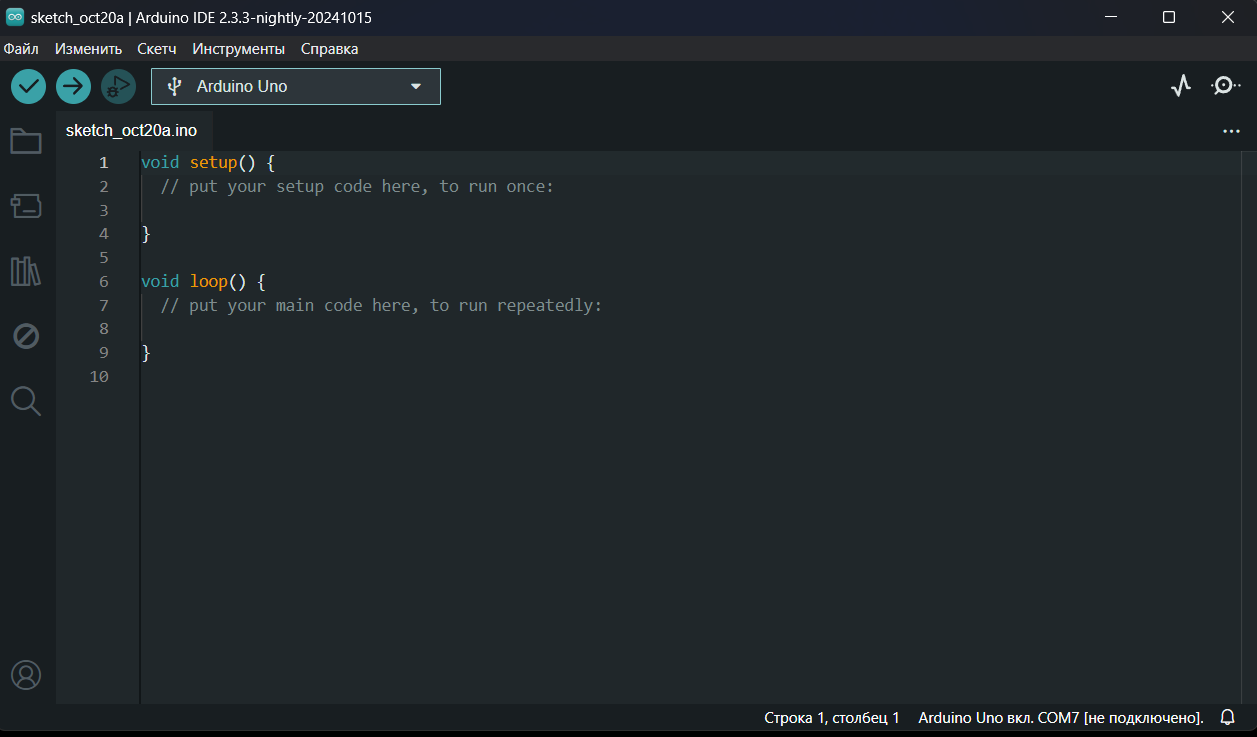


Рисунок 2.1 – Интерфейс *Arduino IDE*

Среда разработки *Arduino* представляет собой мощный инструмент, обеспечивающий разработчиков всеми необходимыми ресурсами для создания проектов на базе платформы *Arduino*. Она включает в себя встроенный текстовый редактор, который позволяет писать программы на языке программирования *Arduino*. Этот редактор обладает функциями подсветки синтаксиса, что значительно упрощает процесс написания и редактирования кода, а также включает дополнительные инструменты для повышения удобства работы.

Важной частью среды является компилятор *AVR-GCC*, который отвечает за преобразование написанного исходного кода в машинный код. Этот код может быть выполнен микроконтроллером, что позволяет реализовывать различные функции и задачи, присущие проектам на *Arduino*.

Среда также предоставляет монитор порта, который играет ключевую роль в отладке и взаимодействии с микроконтроллером. С помощью этого инструмента возможно отправлять и принимать данные через последовательный порт, что позволяет разработчику отслеживать работу программы и осуществлять диагностику.

В дополнение к этим основным компонентам, среда разработки включает панель инструментов, содержащую кнопки для часто используемых команд, а также несколько меню для доступа к различным функциям и настройкам. Эти элементы интерфейса делают процесс разработки более интуитивным и эффективным, позволяя сосредоточиться на реализации идей и проектов. Таким образом, *Arduino* *IDE* становится незаменимым помощником для разработчиков, стремящихся реализовать свои задумки на платформе *Arduino*.

Программирование в среде *Arduino* осуществляется через текстовый редактор, встроенный в *Arduino IDE*. Этот редактор предоставляет удобный интерфейс, который делает процесс написания программ доступным даже для новичков. Язык программирования *Arduino* основан на *C/C++*, что позволяет пользователям использовать знакомые конструкции и библиотеки, упрощая разработку.

После написания кода разработчики могут выполнить несколько важных действий:

Проверка (*Verify*). На этом этапе происходит компиляция написанного кода. *Arduino IDE* анализирует программу на наличие синтаксических и логических ошибок. Если ошибки не обнаружены, *IDE* уведомляет разработчика о том, что код готов к загрузке на микроконтроллер. Этот шаг важен для обеспечения корректности программы и предотвращения сбоев при выполнении.

Загрузка (*Upload*). После успешной проверки код загружается на микроконтроллер *Arduino* с помощью специального загрузчика. Разработчики могут выбрать целевой микроконтроллер и указать последовательный порт, к которому он подключен. Этот процесс позволяет перенести написанный код в память устройства, чтобы оно могло выполнять заданные команды и функции.

Монитор порта (*Serial Monitor*). Для отладки и взаимодействия с программой разработчики могут использовать монитор порта. Этот инструмент позволяет отправлять и получать данные через последовательный порт, что крайне полезно для диагностики работы программы. С его помощью можно отслеживать вывод данных, получать сообщения об ошибках и передавать команды в реальном времени [8].

Кроме того, *Arduino IDE* поддерживает интеграцию с различными библиотеками, которые значительно расширяют функционал проектов. Это позволяет разработчикам использовать готовые решения для работы с датчиками, модулями связи и другими компонентами, ускоряя процесс разработки и упрощая реализацию сложных задач [9].

С помощью *Arduino IDE* и системы разработки на базе *Arduino*, пользователи могут создавать широкий спектр проектов – от простых светодиодных миганий до сложных систем автоматизации и робототехники. Это делает *Arduino* идеальным инструментом как для начинающих, так и для опытных разработчиков.

*Arduino IDE* поддерживает создание и управление проектами, что упрощает организацию кода и ресурсов. Также данная среда разработки предоставляет интерфейс на нескольких языках, что делает ее доступной для разработчиков из разных стран и облегчает совместную работу в многоязычных командах. Это способствует глобальному распространению и популяризации платформы.

## Язык программирования

Выбор языка программирования является критически важным этапом, так как он определяет удобство и эффективность разработки. Например, если выбранный язык обладает хорошей поддержкой библиотек и инструментов для работы с необходимыми технологиями, это ускорит процесс реализации проекта и повысит его качество.

Язык программирования *Arduino* основан на стандартном *C++,* который включает в себя ряд дополнений и библиотек, упрощающих разработку кода для микроконтроллеров *Arduino*. Программы, создаваемые для платформы *Arduino*, называются скетчами и сохраняются в файлах с расширением *.ino*. Эти файлы проходят обработку специальным препроцессором *Arduino*, который выполняет различные задачи, такие как подключение необходимых библиотек и преобразование кода в формат, пригодный для компиляции. Скетчи могут включать в себя функции, специфичные для работы с аппаратными компонентами, такими как датчики, моторы и светодиоды, что позволяет разработчикам легко взаимодействовать с физическим миром.

Основные элементы языка программирования *Arduino*:

1 *setup*(). Это функция, которая вызывается один раз при запуске микроконтроллера. В ней выполняются начальные настройки и инициализация компонентов, таких как порты ввода-вывода, подключенные устройства и параметры работы. Например, здесь можно установить режимы работы пинов (входные или выходные), инициализировать последовательный порт для обмена данными или настроить начальные значения переменных.

2 *loop*(). Эта функция вызывается в бесконечном цикле сразу после завершения выполнения функции *setup*(). В ней размещается основная логика программы, которая выполняется на протяжении всего времени работы микроконтроллера. Здесь можно реализовывать различные действия, такие как считывание данных с датчиков, управление *actuators*, обработка входных сигналов и выполнение условий. Поскольку функция *loop*() непрерывно повторяется, она позволяет реагировать на изменения окружающей среды и поддерживать активное взаимодействие с подключенными компонентами. Это делает программу динамичной и способной выполнять задачи в реальном времени.

3 Библиотеки. *Arduino* предлагает обширный набор библиотек функций для управления различными компонентами, такими как датчики, актуаторы, дисплеи и другие устройства. Эти библиотеки значительно упрощают взаимодействие с аппаратными средствами, позволяя разработчикам сосредоточиться на логике программы, а не на низкоуровневых деталях управления. Кроме стандартных библиотек, предоставляемых *Arduino*, разработчики имеют возможность создавать свои собственные пользовательские библиотеки. Это позволяет не только повторно использовать код в различных проектах, но и делиться им с другими разработчиками. Пользовательские библиотеки помогают организовать код, делают его более понятным и облегчают процесс разработки, особенно при работе над крупными проектами или в командах [10].

Использование уже существующего высокоуровневого языка программирования, такого как *C++*, значительно снижает порог входа для новичков. Это позволяет большему числу людей легко создавать свои проекты на *Arduino*. Высокоуровневый синтаксис и доступные библиотеки упрощают процесс разработки, позволяя сосредоточиться на концепциях и идеях, а не на сложностях низкоуровневого программирования. Кроме того, наличие обширной документации, примеров кода и активного сообщества способствует быстрому обучению и решению возникающих вопросов. Это делает *Arduino* популярным инструментом как для начинающих, так и для опытных разработчиков, желающих реализовать свои идеи в сфере электроники, робототехники и Интернета вещей (*IoT*).

## Основные версии *Arduino IDE*

История *Arduino IDE* начинается в 2005 году, ниже представлены основные версии данной среды разработки и их ключевые изменения:

1 Первая версия (2005 год): В 2005 году была представлена первая версия *Arduino IDE*, основанная на обработке языка *Wiring*. Это дало возможность легко писать код для плат *Arduino*.

2 *Arduino* 0010 (2007 год): Появление новых функций, улучшений в интерфейсе и поддержки различных плат. Это обновление значительно упростило процесс разработки.

3 *Arduino* 0015 (2008 год): Введены новые библиотеки и улучшены инструменты для отладки. Также добавлена поддержка дополнительных плат.

4 *Arduino* 0020 (2010 год): Важные обновления интерфейса и улучшения в поддержке библиотек. Появились новые функции, такие как автоматическое завершение кода.

5 *Arduino* 1.0 (2011 год): Ключевое обновление, которое принесло множество изменений в структуре *IDE*, улучшило производительность и добавило поддержку новых библиотек и плат.

6 *Arduino* 1.5 (2013 год): Введена поддержка для работы с платами *Arduino Due* и улучшены инструменты для работы с библиотеками.

7 *Arduino* 1.6 (2014 год): Значительное обновление, которое включало поддержку дополнительных платформ, улучшенный интерфейс и интеграцию с *GitHub*.

8 *Arduino* 1.6.12 (2016 год): Улучшения в производительности и стабильности, добавление новых функций, таких как поддержка новых библиотек.

9 *Arduino* 1.8 (2017 год): Важное обновление с улучшенной поддержкой для *IoT* и новых плат. Появилась возможность работы с библиотеками с использованием менеджера библиотек.

10 *Arduino* 1.8.10 (2018 год): Обновления безопасности и исправления ошибок, улучшение совместимости с новыми платами.

11 *Arduino* 1.8.13 (2020 год): Новые функции и улучшения, включая поддержку новых библиотек и улучшения в интерфейсе.

12 *Arduino IDE* 2.0 (бета) (2021 год): Появление новой версии *IDE* с переработанным интерфейсом, улучшенной производительностью и новыми инструментами для разработки. Это обновление было направлено на улучшение пользовательского опыта.

13 *Arduino IDE* 2.0 (релиз) (2022 год): Официальный релиз новой версии с множеством новых функций, включая поддержку многопоточности и интеграцию с облачными сервисами [11].

## 2.4 Обоснование выбора программной платформы

Важным аспектом программной платформы *Arduino* являются расширенные возможности микроконтроллеров, которые обеспечивают широкий спектр функциональности для разработчиков. Микроконтроллеры *Arduino* поддерживают различные интерфейсы, такие как *I2C*, *SPI* и *UART*, что позволяет легко интегрироваться с разнообразными периферийными устройствами. Эти интерфейсы обеспечивают высокую скорость передачи данных и позволяют подключать несколько устройств одновременно, что является критически важным при создании сложных встроенных систем.

Используя возможности этих интерфейсов, разработчики могут работать с множеством сенсоров, таких как датчики температуры, влажности, давления и освещения, а также с актуаторами, такими как моторы, реле и светодиоды. Это открывает широкие горизонты для реализации различных проектов – от простых устройств до сложных автоматизированных систем.

Кроме того, *Arduino* предоставляет обширную библиотеку программного обеспечения, которая упрощает взаимодействие с периферийными устройствами. Это позволяет разработчикам сосредоточиться на логике и функциональности своих проектов, а не на низкоуровневом программировании. Таким образом, *Arduino* становится идеальной платформой для обучения и прототипирования, позволяя как новичкам, так и опытным инженерам реализовывать свои идеи быстро и эффективно.

С учетом поддержки различных коммуникационных модулей, таких как *Wi-Fi* и *Bluetooth*, разработчики могут создавать проекты с удаленным управлением и мониторингом. Это делает *Arduino* особенно привлекательным для разработки *IoT*-устройств (интернета вещей), что открывает новые возможности для автоматизации и интеграции в повседневную жизнь.

Таким образом, программная платформа *Arduino* представляет собой мощный инструмент для разработки встроенных систем и приложений. С ее помощью разработчики могут легко создавать программы для микроконтроллеров *Arduino*, используя удобную интегрированную среду разработки, стандартный язык программирования и обширную библиотеку функций. *Arduino* делает процесс программирования доступным и простым, что позволяет как новичкам, так и опытным разработчикам быстро создавать разнообразные встроенные приложения. Также стоит отметить и ее постоянное развитие. *Arduino* остается в центре инноваций в области встраиваемых систем, активно внедряя новые технологии и поддерживая требования современных разработчиков. Благодаря открытой архитектуре и плодотворному взаимодействию с сообществом, платформа продолжает привлекать внимание как опытных инженеров, так и стартапов, исследующих новые горизонты в области встраиваемых технологий.

# 3 ТЕОРЕТИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ ПРОГРАММНОГО ПРОДУКТА

Управление транспортными средствами в состоянии алкогольного опьянения представляет собой серьезную проблему, угрожающую не только жизни водителя, но и безопасности других участников дорожного движения. Статистика показывает, что алкоголь значительно снижает реакцию и способность к адекватному восприятию ситуации на дороге, что часто приводит к авариям с тяжелыми последствиями.

В связи с этим разработка компактного и доступного устройства для оценки уровня опьянения человека является крайне актуальной задачей. Такое устройство может стать важным инструментом в борьбе с пьянством за рулем, способствуя снижению числа аварий и повышению общей безопасности на дорогах.

Применение технологий, таких как датчики уровня алкоголя, позволяет быстро и точно определять состояние водителя. Компактные и недорогие алкотестеры могут быть интегрированы в автомобили или использоваться как портативные устройства, что делает их доступными для широкой аудитории.

Кроме того, подобные устройства могут быть полезны не только водителям, но и работодателям, заботящимся о безопасности своих сотрудников. Регулярная проверка уровня алкоголя может стать частью корпоративной культуры, способствуя формированию ответственного отношения к вождению.

В итоге, создание эффективного и доступного устройства для оценки уровня опьянения человека может существенно снизить риски, связанные с управлением транспортными средствами в состоянии алкогольного опьянения, и повысить безопасность всех участников дорожного движения.

## 3.1 Углубленный анализ датчика *MQ-3*

Датчик *MQ-3* разработан специально для обнаружения паров этилового спирта в окружающем воздухе, что делает его идеальным инструментом для алкотестеров. Этанол, содержащийся в большинстве алкогольных напитков, является основной целью этого датчика. Однако стоит отметить, что, если человек употребил другой вид спирта, например, метанол, MQ-3 не сможет предоставить точные данные, так как он не предназначен для обнаружения этих веществ. Это ограничение следует учитывать при использовании датчика в реальных условиях.

Кроме того, одной из проблем датчика является его чувствительность к шумам и различным внешним факторам. Для повышения точности измерений рекомендуется проводить многократное снятие показаний. Полученные данные следует обрабатывать с использованием соответствующих алгоритмов, и только после этого выводить результаты на дисплей. Такой подход поможет снизить вероятность ошибок и повысить надежность показаний.

Для обеспечения устойчивости к возможным сбоям в работе системы рекомендуется использовать два датчика *MQ-3* одновременно. В случае выхода одного из них из строя или его отключения от платы *Arduino*, обработка данных может основываться на показаниях второго датчика. Это гарантирует, что даже при неисправности одного из устройств, система продолжит функционировать и предоставлять актуальные результаты.

Важно также отметить, что для работы датчиков *MQ-3* требуется предварительный нагрев измерительных элементов. Встроенный нагреватель должен быть активирован, и после запуска алкотестера необходимо подождать около 15 минут, чтобы обеспечить корректную работу устройства. Этот период необходим для стабилизации показаний и достижения оптимальной рабочей температуры датчиков. После полного нагрева их можно использовать без таких ограничений, что значительно упрощает процесс проведения измерений в дальнейшем [12].

## 3.2 Калибровка датчиков

Для повышения точности измерений датчиков алкоголя необходимо провести калибровку на стадии изготовления устройства. Этот процесс включает в себя использование заранее известных концентраций алкоголя, что позволяет установить точную зависимость между показаниями датчиков и реальными значениями.

Калибровка помогает выявить возможные отклонения в измерениях и корректировать их, что обеспечивает более высокую надежность и достоверность результатов. Кроме того, регулярная проверка и калибровка устройства в процессе эксплуатации помогут поддерживать его функциональность и точность на высоком уровне. Таким образом, правильно проведенная калибровка становится ключевым этапом в создании качественного и эффективного алкотестера.

Одним из ключевых инструментов для калибровки является формула Видмарка. Эта формула позволяет установить связь между значениями, полученными от датчиков, и реальными концентрациями алкоголя в крови, выраженными в промилле. Используя эту формулу, можно не только преобразовать значения, полученные от датчиков, в концентрацию алкоголя в крови, но и рассчитать количество алкоголя, необходимое для достижения определённой концентрации. Это особенно полезно для понимания, сколько алкоголя может быть безопасно употреблено.

Кроме того, формула позволяет оценить время, необходимое для полного выведения алкоголя из организма. Зная скорость метаболизма (обычно около 0.1 промилле в час), можно рассчитать временной интервал до момента, когда уровень алкоголя в крови достигнет нуля и наступит состояние «трезвости». Эта информация критически важна для водителей и людей, работающих с механизмами, так как помогает избежать рисков, связанных с управлением в состоянии опьянения. Ниже представлена формула Видмарка:

где:

– концентрация алкоголя в крови в ‰,

– масса выпитого напитка (в пересчете на чистый этанол) в граммах,

– масса тела в килограммах,

– коэффициент распределения Видмарка (0,70 – для мужчин, 0,60 – для женщин)

Формула не учитывает, когда был выпит алкоголь [13].

Сопоставив вручную подсчитанные значения, полученные с использованием формулы, с показателями, измеренными с помощью датчика, можно добиться корректной работы алкотестера. Это позволит обеспечить точную и надежную калибровку устройства. В результате алкотестер будет выводить данные в известных и общепринятых единицах измерения, что сделает результаты более понятными и удобными для пользователей. Такой подход не только повышает доверие к показаниям, но и способствует улучшению общей эффективности устройства в различных условиях эксплуатации.

## 3.3 Пользовательский интерфейс

Создание простого и интуитивно понятного пользовательского интерфейса является ключевым аспектом успешного проекта. Четкое и понятное отображение результатов на *OLED*-дисплее существенно повысит удобство использования устройства, позволяя пользователям легко воспринимать необходимую информацию.

Кроме того, на пользовательский интерфейс можно выводить данные о различных ошибках, возникающих в процессе работы. Это не только упростит процесс диагностики и устранения неполадок, но и позволит пользователям быстрее реагировать на возможные проблемы. В результате они смогут оперативно принимать меры для устранения неисправностей, что значительно повысит надежность устройства. Также стоит отметить, что наличие информативных сообщений об ошибках и состоянии системы может снизить вероятность неправильного использования устройства, обеспечивая тем самым более безопасную эксплуатацию. Поэтому продуманный интерфейс не только улучшает взаимодействие с пользователем, но и способствует повышению общей эффективности и долговечности устройства.

Таким образом, теоретическое обоснование разработки программного продукта для алкотестера на базе датчика *MQ-3* подчеркивает важность данного устройства в обеспечении безопасности дорожного движения. Калибровка устройства и тщательная настройка его рабочих параметров являются ключевыми аспектами, которые делают алкотестер эффективным инструментом не только для контроля за состоянием водителей, но и для предотвращения аварий, связанных с алкогольным опьянением.

Алкотестеры, основанные на технологии *MQ-3*, обладают высокой чувствительностью и точностью, что позволяет им выявлять даже незначительные концентрации алкоголя в выдыхаемом воздухе. Это делает их незаменимыми в ситуациях, когда важно быстро и надежно оценить состояние водителя. Важно отметить, что регулярное использование такого устройства может способствовать формированию ответственного отношения к вождению, повышая осознанность водителей относительно их состояния и влияния алкоголя на способность управлять транспортным средством.

Также, алкотестеры могут стать важным элементом в рамках комплексного подхода к безопасности на дорогах. Их интеграция в программы профилактики ДТП может включать установку алкотестеров на предприятиях, в общественных местах и на мероприятиях, где возможно употребление алкоголя. Это создает дополнительные возможности для повышения общественной безопасности и формирования культуры трезвого вождения.

Кроме того, использование алкотестеров может способствовать снижению стигмы вокруг темы алкогольного опьянения. Когда такие устройства становятся частью повседневной жизни, они помогают создать атмосферу, в которой ответственность за безопасное вождение воспринимается как норма. В конечном итоге, внедрение алкотестеров на базе датчика *MQ-3* имеет значительное значение для общества в целом, так как оно не только снижает количество дорожно-транспортных происшествий, но и способствует изменению общественного сознания в отношении безопасного вождения.

Таким образом, разработка и внедрение алкотестера на базе датчика *MQ-3* представляют собой важный шаг к созданию более безопасной дорожной среды, что является неотъемлемой частью стратегии по снижению числа аварий и повышению общей безопасности на дорогах.

# 4 ПРОЕКТИРОВАНИЕ ФУНКЦИОНАЛЬНЫХ ВОЗМОЖНОСТЕЙ ПРОГРАММЫ

## 4.1 Обоснование и описание функциональных возможностей

**4.1.1** Запуск и инициализация

**Функция:** Запуск и инициализация алкотестера при включении.

**Обоснование:** В момент включения алкотестера требуется выполнить ряд начальных настроек и подготовительных действий, таких как активация датчика *MQ-3*, инициализация дисплея и задание параметров измерений.

**Реализация:** При включении устройства микроконтроллер *Arduino* будет инициализировать все подключенные компоненты, устанавливать начальные значения и отображать приветственное сообщение на экране.

**4.1.2** Измерение уровня алкоголя

**Функция:** Считывание данных с датчика *MQ-3* для измерения уровня алкоголя в выдыхаемом воздухе.

**Обоснование:** Это ключевая функция алкотестера, так как именно здесь происходит сбор данных, необходимых для определения уровня алкоголя. Датчик *MQ-3* осуществляет измерения, и полученные данные передаются в программное обеспечение для дальнейшей обработки.

**Реализация:** Программное обеспечение будет периодически опрашивать датчик *MQ-3*, считывать значения сопротивления и преобразовывать их в концентрацию алкоголя с использованием заранее определенных коэффициентов.

**4.1.3** Анализ результатов

**Функция:** Анализ данных и определение, превышает ли измеренный уровеньалкоголя установленные нормы.

**Обоснование:** Определение, находится ли уровень алкоголя в пределах допустимыхнорм, является критической функцией алкотестера. От этого решения зависитдальнейшее взаимодействие с пользователем и предоставлениерекомендаций.

**Реализация:**

Программное обеспечение будет сравнивать измеренное значение алкоголя с установленными нормами, определенными законодательством. В случае превышения нормы, оно перейдет к следующей функции – отображению результата.

**4.1.4** Отображение результатов

**Функция:** Отображение измеренного уровня алкоголя и рекомендации на экране(например, *OLED*-дисплее).

**Обоснование:** Пользователь должен видеть результаты измерения и рекомендации дляпринятия соответствующих решений. Отображение информации на экранеделает использование алкотестера удобным и информативным.

**Реализация:** При успешном завершении измерения результат отобразится на экранев виде концентрации алкоголя и, возможно, рекомендаций (например,"Допустимо управлять" или "Не допускается управление").

**4.1.5** Отображение ошибок

**Функция:** Вывод на экран возникающих в процессе работы ошибок.

**Обоснование:** Отображение ошибок способно облегчить ремонт аппаратных компонентов устройства, а также предупредить пользователя о том, что данные могут быть некорректными.

**Реализация:** При обнаружении ошибки, ей будет присвоен код, которые будет выведен на экран устройства.

Таким образом, проектирование функциональных возможностей программного обеспечения для алкотестера на базе Arduino с использованием датчиков паров спирта *MQ-3* представляет собой многоэтапный процесс, который охватывает различные аспекты разработки. В первую очередь, необходимо создать и реализовать функции, которые обеспечивают точное измерение и анализ уровня алкоголя в выдыхаемом воздухе. Это включает в себя не только обработку данных, получаемых от датчика, но и их корректную интерпретацию с учетом возможных погрешностей и внешних факторов. Кроме того, проект предполагает внедрение механизмов, которые будут предоставлять пользователю рекомендации на основе полученных результатов. Например, устройство сможет выдавать советы о том, стоит ли садиться за руль, или предлагать альтернативные способы передвижения в случае превышения допустимого уровня алкоголя. Также важной частью функционала станет система оповещения, которая будет информировать пользователя о состоянии его алкоголя в крови. Это может быть реализовано через звуковые сигналы, вибрацию или визуальные уведомления на экране, что делает устройство более интуитивно понятным и доступным для использования в стрессовых ситуациях. В результате, разработанные функции позволят создать полноценное устройство для контроля уровня алкоголя, которое не только обеспечит безопасность на дорогах, но и будет способствовать общей безопасности в обществе, снижая риски, связанные с вождением в нетрезвом виде.

# 5 АРХИТЕКТУРА РАЗРАБАТЫВАЕМОЙ ПРОГРАММЫ

В качестве микроконтроллера для курсового проекта был выбран клон *Arduino UNO* [16], который обеспечивает все те же функциональные возможности, что и оригинальное устройство. Этот выбор обусловлен доступностью и простотой работы с данной платформой, что идеально подходит для реализации поставленных задач. Для сбора данных о концентрации паров спирта в воздухе используются два сенсора *MQ-3* от производителя *Flying-Fish*. Эти сенсоры известны своей чувствительностью и надежностью, что позволяет получить точные измерения в различных условиях. Для отображения полученных данных предусмотрен *OLED*-дисплей *SSD1306* с разрешением 128x64 пикселя. Данный дисплей отличается высоким контрастом и малым потреблением энергии, что делает его идеальным для мобильных решений. Соединение между *Arduino* и остальными компонентами проекта осуществляется с помощью макетной платы. Это позволяет легко настраивать и модифицировать схему, что особенно важно в процессе разработки и тестирования. Такой подход обеспечивает гибкость и упрощает эксперименты с различными конфигурациями устройства.

Макетная плата, или *breadboard* – это беспаечная плата для монтажа. Универсальный инструмент для моделирования прототипов устройств. Альтернатива создания схем без применения паяльника. Монтажную плату применяют для конструирования, отладки и тестирования будущей схемы устройства при разных условиях подключения и эксплуатации [14]. Внешний вид макетной платы представлен на рисунке 5.1.

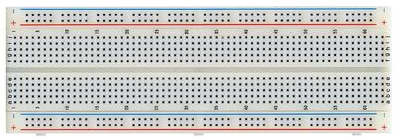


Рисунок 5.1 – Макетная плата

Макетная плата состоит из пластиковой пластины-основания с большим количеством отверстий. Расстояние между отверстиями – 2,54 мм, их диаметр – 0,8 мм. Внутри расположены два вида дорожек: контактные группы пластин и вертикальные дорожки для подачи питания. Ряды металлических пластин-рельс имеют по пять клипс каждая. Отверстия, в которые будут вставляться ножки элементов, соединяют клипсы и таким образом замыкают схему. Проводник, подключенный к отверстию в одном из рядов, одновременно соединяется с остальными контактами этого ряда. На одной рельсе можно подключить до 5 элементов, которые будут связаны между собой. Две рельсы одного ряда изолированы друг от друга. Рельсы питания расположены вертикально по краям. Как правило, синий цвет линии указывает, что нужно подавать «-» напряжения, а красный – «+». Чтобы легче работать с макетной платой, на нее производители наносят обозначения дорожек латинскими буквами и цифрами. Так проще описывать процесс сборки или писать инструкции для подключений [15]. На рисунке 5.2 представлено обозначение дорожек на макетной плате.

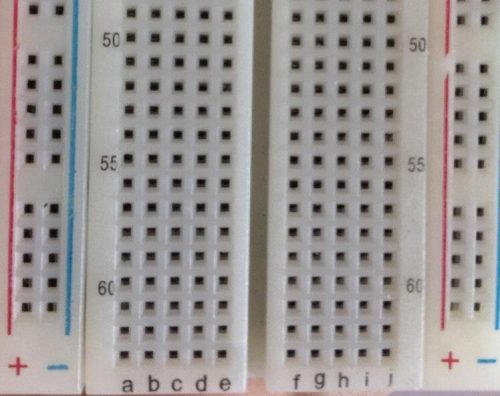


Рисунок 5.2 – Обозначение дорожек на макетной плате

Макетная плата – это современный вариант сборки схем, который позволяет легко изменять варианты схемы, добавлять компоненты и проверять теории и способы подсоединения. Если вы паяете и что-то неправильно соединили или решили что-то изменить в схеме – придется перепаивать. На breadboard процесс реконструкции, отладки и модификации осуществляется мгновенно: снял, подключил по-другому. Однако беспаечный способ соединения элементов схемы не такой надежный, как при сцеплении припоем. Например, при постоянной вибрации устройства контакты будут понемногу ослабляться. Кроме того, внешний вид с висящими проводами требует большего корпуса для проекта.

## 5.1 Подключение устройств к *Arduino*

Так как прямое подключение устройств к *Arduino* невозможно, для этих целей будут использоваться провода "папа-папа", которые относится к типам разъемов, используемым для подключения проводов и периферийных устройств к плате *Arduino* или другим микроконтроллерам. Эти разъемы могут быть представлены в виде "мама" (*female*) и "папа" (*male*), где "мама" представляет собой разъем с отверстиями для вставки, а "папа" – выступающий разъем для вставки в "маму" [17].

В качестве дисплея был выбран *SSD1306* *OLED*-дисплей, который обеспечивает высокое качество отображения информации. Для его подключения используются четыре разъема: два питающих (*GND* и *VDD*) и два для передачи данных по шине *I2C* (*SCK* и *SDA*) [18].

*GND* – разъем для подключения к общему земле, который соединяется с одним из трех разъемов на плате Arduino.

*VDD* – разъем для питания дисплея, который подключается к одному из двух пинов 5V на *Arduino*, обеспечивая необходимое напряжение для работы устройства.

Для передачи данных по шине *I2C* используются следующие разъемы:

*SDA* – разъем для передачи данных между микроконтроллером и дисплеем, который подключается к аналоговому пину A4 на Arduino.

*SCK* – разъем для синхронизации устройств, который подключается к аналоговому пину A5 [22].

Такое подключение обеспечивает надежный обмен данными и позволяет дисплею корректно отображать информацию, получаемую от микроконтроллера. Схема подключения дисплея напрямую к *Arduino* представлена на рисунке 5.3.

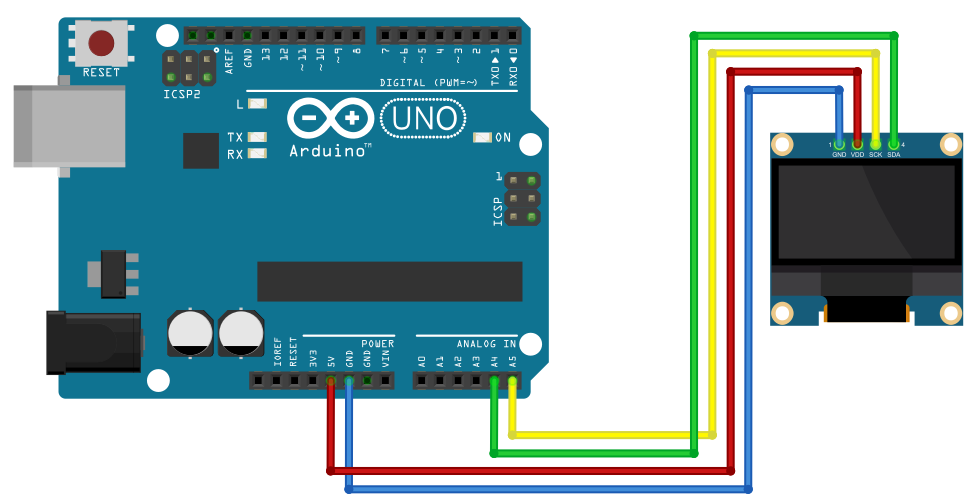


Рисунок 5.3 – Подключение *OLED*-дисплея к *Arduino*

Для подключения датчиков *MQ-3* используется три разъема: два питающих (*GND* и *VCC*) и один для передачи аналоговых данных (*AO*). Хотя на датчике также имеется цифровой выход (разъем *DO*), в данном устройстве он не будет использоваться, поэтому его подключение не требуется. Подключение питающих разъемов осуществляется аналогично подключению дисплея: разъем *VCC* на датчике соединяется с соответствующим пином на Arduino, аналогично *VDD* на дисплее.

Аналоговые выходы датчиков подключаются следующим образом:

Разъем *AO* первого датчика подключается к аналоговому порту A0 на Arduino.

Разъем *AO* второго датчика подключается к аналоговому порту A1 [19].

Такое подключение обеспечивает корректный сбор данных с датчиков, позволяя микроконтроллеру обрабатывать информацию о концентрации паров спирта в воздухе. Схемы подключений для каждого из датчиков напрямую к *Arduino* представлены на рисунках 5.4 и 5.5.

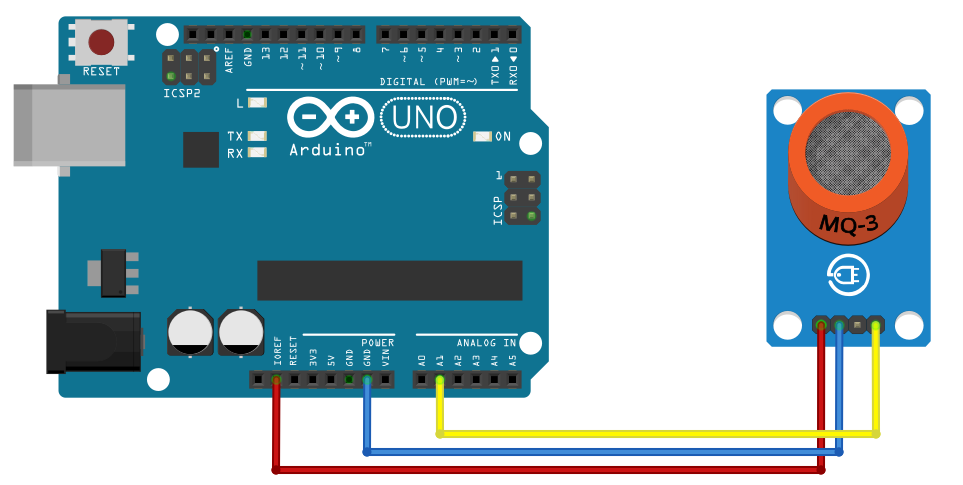


Рисунок 5.4 – Подключение первого датчика *MQ-3* к *Arduino*

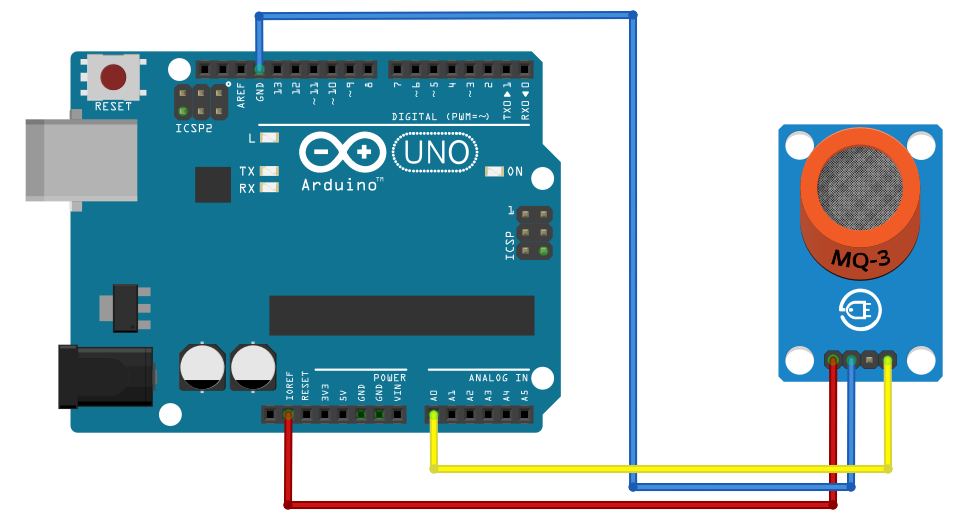


Рисунок 5.5 – Подключение второго датчика *MQ-3* к *Arduino*

На *Arduino UNO* имеется всего два пина *5V*, что может ограничивать количество подключаемых устройств, особенно если требуется питание для нескольких компонентов. В этой ситуации использование макетной платы становится особенно актуальным и удобным. Макетная плата, или прототипная плата, позволяет эффективно организовать подключение множества различных элементов, таких как датчики, светодиоды и модули, обеспечивая при этом простоту и гибкость в настройке схемы. Одним из ключевых преимуществ макетной платы является возможность быстрого монтажа и демонтажа компонентов. Это делает процесс разработки более динамичным, так как позволяет легко вносить изменения в соединения без необходимости пайки, что экономит время и усилия. Например, если нужно протестировать новый датчик или изменить конфигурацию, можно сделать это, просто переставив провода. Кроме того, макетная плата способствует лучшему управлению пространством и уменьшает беспорядок на рабочем месте. Все соединения можно организовать аккуратно и логично, что облегчает отладку и поиск возможных ошибок в схеме. Также такая платформа подходит как для новичков, так и для опытных разработчиков, предоставляя возможность экспериментировать с различными концепциями и проектами без значительных затрат. Схема подключения всех устройств с помощью макетной платы представлена на рисунке 5.6.

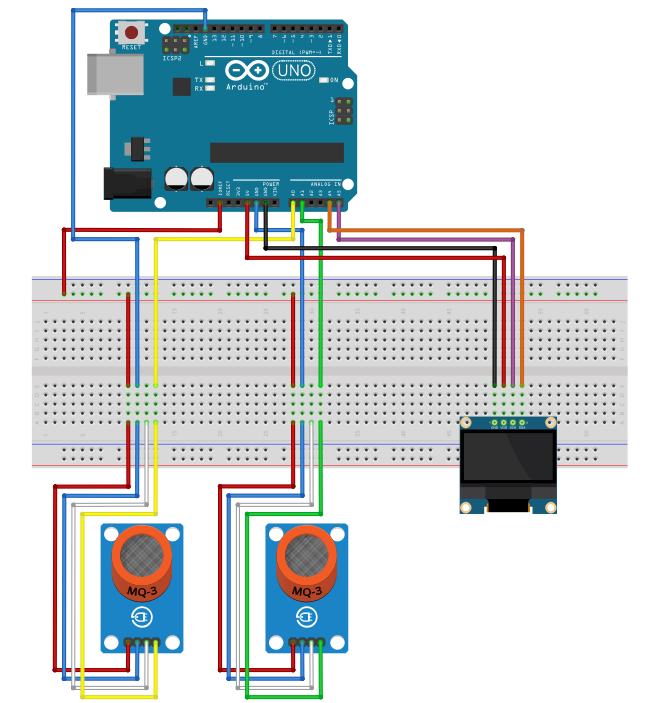


Рисунок 5.6 – Подключение всех устройств к *Arduino* с использованием макетной платы

## 5.2 Общая структура программы

Архитектура программного обеспечения разрабатываемого алкотестера имеет модульную структуру, что обеспечивает организацию программы в виде набора отдельных функциональных блоков, выполняющих определенные задачи и взаимодействующих с другими блоками для обеспечения корректной работы устройства. Ниже описаны основные модули программы:

1 модуль инициализации и настройки отвечает за первоначальную инициализацию платы *Arduino*, настройку портов ввода-вывода и инициализацию подключенный устройств;

2 модуль измерения уровня алкоголя отвечает за считывание данных с конкретного датчика паров спирта *MQ-3*;

3 модуль анализа данных отвечает за обработку данных, полученных от датчиков для дальнейшего вывода этих данных на дисплей;

4 модуль вывода данных отвечает за отображение пользовательского интерфейса и вывод обработанных данных, полученных с датчиков, на экран;

5 модуль обработки ошибок отвечает за обработку данных, поступающих от датчиков, и определение ошибок в работе сенсоров на основе этих данных. В случае обнаружения ошибки, ей присваивается определенный код, который далее выводится на экран;

6 модуль вывода ошибок отвечает за отображение кодов ошибок на дисплее;

7 основной цикл программы отвечает за работу устройства и взаимодействие остальных модулей программы между собой.

Такая организация программы упрощает ее разработку и поддержку, а также упрощает поиск и исправление ошибок в ее работе [20].

## 5.3 Описание функциональной схемы программы

Функциональная схема – это графическая (приложение В) или текстовое представление взаимодействия компонентов программного обеспечения с описанием информационных потоков, состава данных в потоках и указанием используемых файлов и устройств. Она используется для визуализации и анализа функциональных аспектов системы, позволяя легче понять, как система выполняет определенные задачи или функции. Ниже представлено текстовое описание функциональной схемы программы.

**5.3.1** Начало программы:

– запуск программы на микроконтроллере *Arduino*;

– настройка портов микроконтроллера и инициализация подключенных устройств;

– инициализация последовательного порта для взаимодействия с компьютером, подключенным через *USB*.

**5.3.2** Основной цикл:

– программная задержка для уменьшения количества выводимых сообщений;

– проверка подключенных датчиков на наличие ошибок в подключении;

– вывод на экран кода ошибки в случае, если один из датчиков или сразу оба работают некорректно;

– считывание данных с датчиков, для каждого из датчиков считывается по 3 значения, а затем находится их среднее арифметическое. Данное действие необходимо для корректировки значений и получения более точных данных;

– вывод полученных данных на экран прибора;

– возврат к началу основного цикла.

## 5.4 Описание блок-схемы алгоритма программы

Блок-схема – это схематичное представление процесса (приложение Г), системы или компьютерного алгоритма. Блок-схемы часто применяются в разных сферах деятельности, чтобы документировать, изучать, планировать, совершенствовать и объяснять сложные процессы с помощью простых логичных диаграмм. Для построения блок-схем применяются прямоугольники, овалы, ромбы и некоторые другие фигуры (для обозначения конкретных операций), а также соединительные стрелки, которые указывают последовательность шагов или направление процесса. Ниже представлено текстовое описание блок-схемы алгоритма программы.

**5.4.1** Подключение библиотек, инициализация устройств и глобальных переменных программы:

– подключение библиотеки *GyverOLED* [21];

– создание объекта *display*, представляющего *OLED*-дисплей на программном уровне;

– создание глобальной переменной *err\_code*, которая нужна для хранения кода обнаруженной ошибки.

**5.4.2** Установка программы (*setup*):

– инициализация объекта *display* для дальнейшей работы с ним и очистка экрана устройства;

– настройка пинов подключения датчиков для работы в режиме ввода с подтягивающим резистором (*INPUT\_PULLUP*) [23];

– инициализация последовательного порта и установка скорости его работы.

**5.4.3** Обработка ошибок в работе датчиков:

– чтение данных с датчиков;

– проверка наличие питания на датчиках (в случае его отсутствия на аналоговом выходе, к которому подключен датчик, будет значение 0). В случае обнаружения ошибки возврат ее кода в переменную *err\_code*;

– проверка подключения датчиков к аналоговым выходам (т.к. используется подтягивающий резистор, то в случае отсутствия подключения к устройству на пине будет максимальное возможное для него значение, а именно 1024). В случае обнаружения ошибки возврат ее кода в переменную *err\_code*;

– выход из функции обработки ошибок.

**5.4.4** Вывод кода ошибки на экран:

– проверка на наличие кода ошибки в переменной *err\_code*;

– в случае, если ошибок нет, возврат из функции вывода кода ошибки;

– установка размера текста;

– установка курсора в необходимую позицию;

– вывод кода ошибки на экран.

**5.4.5** Чтение данных с датчиков:

– трехкратное считывание данных с каждого датчика;

– нахождение среднего арифметического из считанных значений;

– возврат результата из функции.

**5.4.6** Вывод данных на экран:

– очистка дисплея;

– установка размера текста;

– установка курсора в необходимую позицию;

– вывод данных на экран.

**5.4.7** Программная задержка и возврат в начало главного цикла программы.

## 5.5 Коды программных ошибок

В процессе работы программы могут возникать различные ошибки, каждой из которых поставлен в соответствие определенный числовой код. Коды ошибок представлены в формате *x\_y*, где *x* – идентификатор ошибки, а *y* – идентификатор конкретного датчика. Ниже описаны все коды ошибок с пояснениями по каждому из них:

– 1\_0 – отсутствие питания на датчике, подключенному к выходу A0 на плате *Arduino*;

– 1\_1 – отсутствие питания на датчике, подключенному к выходу A1 на плате *Arduino*;

– 1\_2 – отсутствие питания на обоих датчиках, подключенных к выходам A0 и A1 на плате *Arduino*;

– 2\_0 – отсутствие подключения датчика к выходу A0 на плате *Arduino*;

– 2\_1 – отсутствие подключения датчика к выходу A1 на плате *Arduino*;

– 2\_2 – отсутствие подключения датчиков как к выходу A0, так и к выходу A1 на плате *Arduino*;

– 3 – значения на датчиках слишком сильно различаются.

Таким образом, разработанное устройство алкотестера позволяет измерять уровень алкоголя в крови человека по содержанию паров спирта в выдыхаемом им воздухе. Программа алкотестера обладает модульной структурой, что облегчает ее поддержку. Также данное устройство способно обрабатывать различные ошибки в работе датчиков, что облегчает его починку в случае выхода какого-либо датчика из строя.

# ЗАКЛЮЧЕНИЕ

В процессе выполнения данного курсового проекта было разработано и протестировано устройство для измерения уровня алкоголя в крови человека по концентрации его паров в выдыхаемом воздухе. Данный проект продемонстрировал широкие возможности микроконтроллеров семейства *Arduino* для создания различных систем измерения. В ходе выполнения работы были выполнены следующие задачи:

1 Изучение принципов работы датчиков измерения концентрации паров спирта в воздухе *MQ-3*.

2 Реализация программного обеспечения для *Arduino*, обеспечивающего считывание и обработку поступающих с датчиков данных с последующим выводом результатов на дисплей.

3 Обработка получаемых с датчиков данных для преобразования поступающих числовых значений в промилле.

Важным этапом в разработке устройства были его тестирование в реальных условиях и преобразование получаемых с датчиков значений в стандартные единицы измерения. Поскольку датчики по умолчанию предоставляют значения, которые не могут быть непосредственно интерпретированы человеком, в процессе работы было принято решение воспользоваться формулой Видмарка для преобразования этих величин в промилле.

В заключение, данный курсовой проект подтверждает высокую актуальность использования устройств для проверки уровня алкогольного опьянения человека для обеспечения безопасности дорожного движения и работы на различных предприятиях. Разработанное в ходе курсового проекта устройство представляет собой платформу, которую при необходимости можно расширить добавлением других датчиков для измерения параметров окружающей среды, что сможет повысить точность измерений при влиянии различных внешних факторов.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Стандарт предприятия. Дипломные проекты (работы). Общие требования [Электронный ресурс]. – Режим доступа: https://www.bsuir.by/m/12\_100229\_1\_185586.pdf. – Дата доступа: 23.10.2024.

[2] Аливерти, П Изучаем Arduino. Руководство для начинающих / П Аливерти. – Москва : Эксмо, 2021.– 400 с.

[3] Что такое Arduino – Проекты на микроконтроллерах [Электронный ресурс]. – Режим доступа: <http://micpic.ru/home/proekty-na-arduino/186-chto-takoe-arduino.html>. – Дата доступа: 23.10.2024.

[4] Микроконтроллеры – Arduino – Программирование [Электронный ресурс]. – Режим доступа: <https://radio-hobby.org/modules/instruction/arduino/microcontroller>. – Дата доступа: 23.10.2024.

[5] История создания Arduino [Электронный ресурс]. – Режим доступа: <https://arduino.ua/ru/art2-istoriya-sozdaniya-arduino>. – Дата доступа: 23.10.2024.

[6] Газовый датчик MQ-3: Окончательное руководство по датчику спирта [Электронный ресурс]. – Режим доступа: https://pcbrussia.com/%D0%93%D0%B0%D0%B7%D0%BE%D0%B2%D1%8B%D0%B9-%D0%B4%D0%B0%D1%82%D1%87%D0%B8%D0%BA-Mq-3.html. – Дата доступа: 23.10.2024.

[7] Интерфейс I2C в Arduino [Электронный ресурс]. – Режим доступа: https://voltiq.ru/wiki/i2c-interface-in-arduino/. – Дата доступа: 23.10.2024.

[8] Среда разработки Arduino | Аппаратная платформа [Электронный ресурс]. – Режим доступа: https://arduino.ru/Arduino\_environment. – Дата доступа: 23.10.2024.

[9] Библиотеки Arduino [Электронный ресурс]. – Режим доступа: <https://www.arduino.cc/en/Reference/Libraries>. – Дата доступа: 23.10.2024.

[10] Arduino Programming Language (язык программирования Arduino) [Электронный ресурс]. – Режим доступа: <https://www.arduino.cc/reference/en/>. – Дата доступа: 23.10.2024.

[11] Previous IDE Releases [Электронный ресурс]. – Режим доступа: https://www.arduino.cc/en/software/OldSoftwareReleases. – Дата доступа: 23.10.2024.

[12] Исследование прогрева датчика алкоголя MQ-3 – LabData [Электронный ресурс]. – Режим доступа: https://labdata.ru/article/issledovanie-progreva-datchika-alkogolja-mq-3. – Дата доступа: 23.10.2024.

[13] E. M. P. Widmark, Principles and applications of medicolegal alcohol determination : Databook / R. C. Baselt.– Davis, California, 1932.

[14] Breadboard – макетная (монтажная) беспаечная плата [Электронный ресурс]. – Режим доступа: https://playarduino.ru/project/breadboard-bespaechnaya-plata/. – Дата доступа: 23.10.2024.

[15] Макетная плата (База знаний) [Электронный ресурс]. – Режим доступа: https://ampermarket.kz/base/breadboard/. – Дата доступа: 23.10.2024.

[16] Arduino Uno. Официальная документация [Электронный ресурс]. Режим доступа: <https://www.arduino.cc/en/Guide/ArduinoUno>. – Дата доступа: 23.10.2024.

[17] Что такое папа мама в электрике? [Электронный ресурс]. – Режим доступа : <https://uchet-jkh.ru/i/cto-takoe-papa-mama-v-elektrike/>. – Дата доступа: 23.10.2024.

[18] Роботехника18. Как подключить OLED SSD1306 дисплей к Ардуино [Электронный ресурс]. – Режим доступа: <https://xn--18-6kcdusowgbt1a4b.xn--p1ai/oled-arduino/>. – Дата доступа: 23.10.2024.

[19] Схема подключения датчика алкоголя MQ-3 [Электронный ресурс]. – Режим доступа: <https://prog.life/kb/shema-podklyucheniya-datchika-alkogolya-mq-3/>. – Дата доступа: 23.10.2024.

[20] Модульная структура программных продуктов [Электронный ресурс]. – Режим доступа : <https://studref.com/311804/informatika/modulnaya_struktura_programmnyh_produktov>. – Дата доступа: 23.10.2024.

[21] GyverLibs/GyverOLED: Лёгкая и быстрая библиотека для OLED дисплея [Электронный ресурс]. – Режим доступа: https://github.com/GyverLibs/GyverOLED. – Дата доступа: 23.10.2024.

[22] Arduino и OLED-дисплей [Электронный ресурс]. – Режим доступа: https://kit.alexgyver.ru/tutorials/oled/. – Дата доступа: 23.10.2024.

[23] InputPullupSerial | Arduino Documentation [Электронный ресурс]. – Режим доступа: https://docs.arduino.cc/built-in-examples/digital/InputPullupSerial/. – Дата доступа: 23.10.2024.

# ПРИЛОЖЕНИЕ А

# (Обязательное)

# Справка о проверке на заимствования

# ПРИЛОЖЕНИЕ Б

**(обязательное)**

**Листинг программного кода**

#include <GyverOLED.h>

GyverOLED<SSD1306\_128x64, OLED\_NO\_BUFFER> display;

String err\_code = "";

void setup() {

display.init();

display.clear();

pinMode(A0, INPUT\_PULLUP);

pinMode(A1, INPUT\_PULLUP);

Serial.begin(9600);

}

void loop() {

handleError();

if (err\_code != ""){

printError();

}

int s0 = readAlcohol(0);

int s1 = readAlcohol(1);

printData(s0, s1);

delay(100);

}

int readAlcohol(int pin) {

int val = 0;

int val1;

int val2;

int val3;

val1 = analogRead(pin) - 40;

delay(10);

val2 = analogRead(pin) - 40;

delay(10);

val3 = analogRead(pin) - 40;

val = (val1+val2+val3)/3;

return val;

}

double ConvertAlcohol(int s0\_val, int s1\_val) {

int medium = (s0\_val + s1\_val) / 2;

double promille = (double)medium \* 0.00749805 - 1.06199;

Serial.println(promille);

return promille;

}

void printData(int s0\_val, int s1\_val) {

display.clear();

display.setScale(2);

double alco = ConvertAlcohol(s0\_val, s1\_val);

if(alco < 0){

alco = 0;

}

display.setCursorXY(10, 20);

display.print("Alco: ");

display.print(alco);

String alco\_status = "";

String driving\_status = "Don't drive a car!!!";

if(alco < 0.3){

alco\_status = "You are sober";

driving\_status = "You can drive a car";

}

else if (alco >= 0.3 && alco < 1){

alco\_status = "Light intoxitation";

}

else if (alco >= 1 && alco < 2){

alco\_status = "Medium intoxitation";

}

else if (alco >= 2 && alco < 3){

alco\_status = "Hard intoxitation";

}

else{

alco\_status = "You are dead";

}

display.setScale(1);

display.setCursorXY(10, 40);

display.print(alco\_status);

display.setCursorXY(10, 50);

display.print(driving\_status);

}

void handleError(){

bool s0\_noPower = noPower(0);

bool s1\_noPower = noPower(1);

if(s0\_noPower && s1\_noPower){

err\_code = "1\_2";

return;

}

else if (s0\_noPower){

err\_code = "1\_0";

return;

}

else if (s1\_noPower){

err\_code = "1\_1";

return;

}

bool s0\_noInput = noInput(0);

bool s1\_noInput = noInput(1);

if(s0\_noInput && s1\_noInput){

err\_code = "2\_2";

return;

}

else if (s0\_noInput){

err\_code = "2\_0";

return;

}

else if (s1\_noInput){

err\_code = "2\_1";

return;

}

bool err\_values = errValues();

if(err\_values){

err\_code = "3";

return;

}

err\_code = "";

}

bool noPower(int pin){

int val = analogRead(pin);

return val <= 50 ? true : false;

}

bool noInput(int pin){

int val = analogRead(pin);

return val >= 1020 ? true : false;

}

bool errValues(){

int v0 = analogRead(0);

int v1 = analogRead(1);

int diff = v0 - v1;

return diff < -150 || diff > 150;

}

void printError(){

display.setScale(1);

display.setCursorXY(10, 0);

display.print("Err code: ");

display.print(err\_code);

}

# ПРИЛОЖЕНИЕ В

**(Обязательное)**

**Функциональная схема алгоритма**

# ПРИЛОЖЕНИЕ Г

**(Обязательное)**

**Блок схема алгоритма**

# ПРИЛОЖЕНИЕ Д

**(Обязательное)**

**Графический интерфейс пользователя**

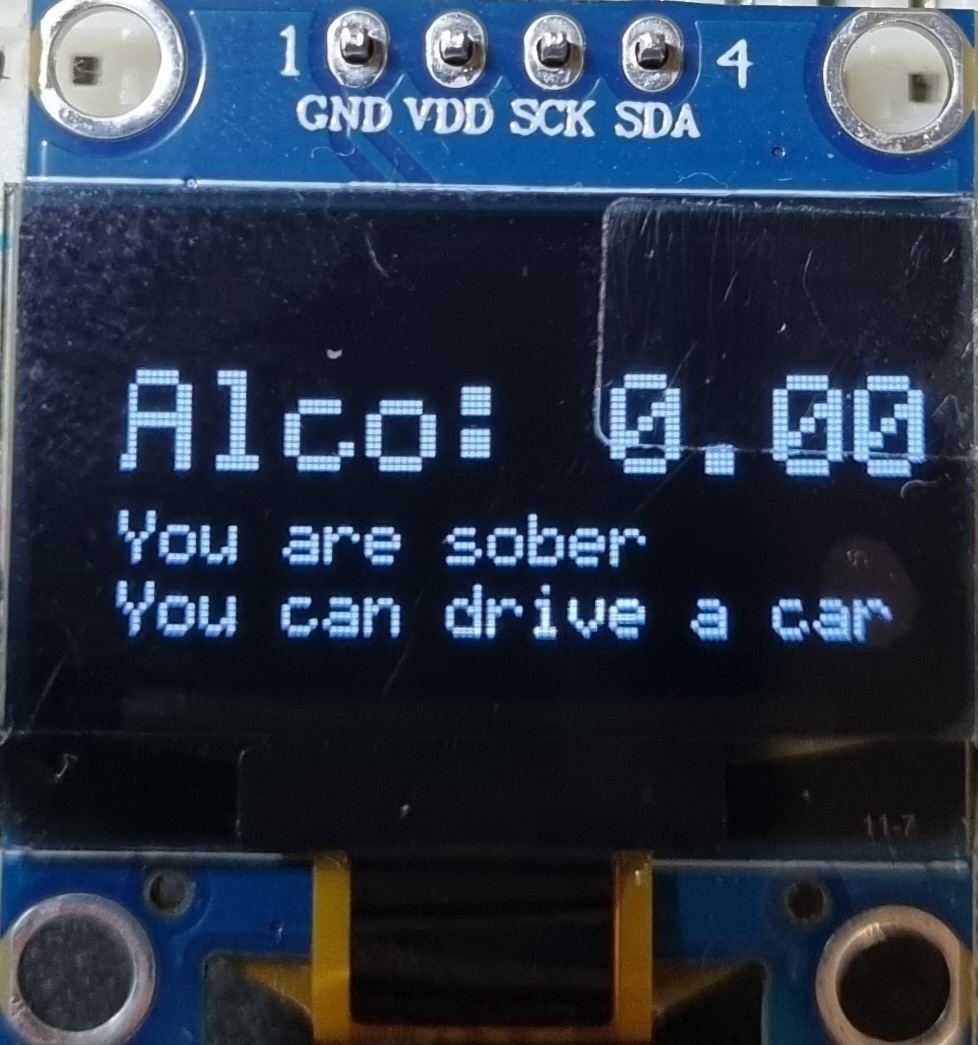


Рисунок Г.1 – Графический интерфейс пользователя

# ПРИЛОЖЕНИЕ Е

**(Обязательное)**

**Ведомость документов**