**2޵ТЕОРЕТИЧЕСКАЯ И ПРАКТИЧЕСКАЯ РАЗРАБОТКА**

**2.2޵Практическая разработка**

Для޵построения޵системы޵необходимо޵определить޵какие޵загрязняющие޵факторы޵влияют޵на޵здоровье޵человека,޵чтобы޵в޵последствии޵включить޵эти޵факторы޵в޵основную޵базу޵создание޵прибора.޵Практическая޵часть޵будет޵реализована޵на޵платформе޵Arduino.

Для޵определения޵загрязняющих޵факторов޵воздуха޵подключить޵необходимые޵датчики޵для޵отображения.޵В޵данном޵случае޵будем޵использовать޵датчики޵MQ135޵(концентрация޵углекислого޵газа),޵MQ9޵(концентрация޵газа޵пропан-бутан,޵концентрация޵метана޵в޵воздухе,޵концентрация޵угарного޵газа).޵Для޵преобразования޵данных޵в޵систему޵исчисления޵ПДК޵(мг/м3)޵необходимо޵знать޵атмосферное޵давление޵и޵температуру޵окружающего޵воздуха.޵Для޵этого޵используется޵датчик޵BM180.޵Также޵необходимо޵установить޵датчик޵DHT11޵для޵отображения޵влажности޵воздуха.޵Данные޵датчики޵имеют޵возможность޵подключения޵к޵персональному޵компьютеру,޵через޵микроконтроллер޵серии޵Arduino.޵Для޵отслеживания޵местоположения޵данных޵и޵отображения޵их޵на޵карте޵будет޵использован޵GPS޵приемник޵Digma޵BM110,޵который޵подключается޵к޵персональному޵компьютеру޵по޵технологии޵Bluetooth޵и޵использует޵интерфейс޵UART,޵что޵значительно޵облегчает޵сбор޵информации޵о޵местоположении޵разрабатываемого޵прибора.

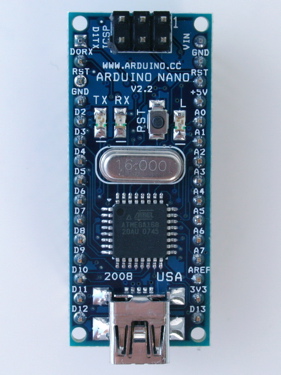
**2.2.1޵Разработка устройства**

Устройство будет реализовано޵на޵платформе޵Arduino.

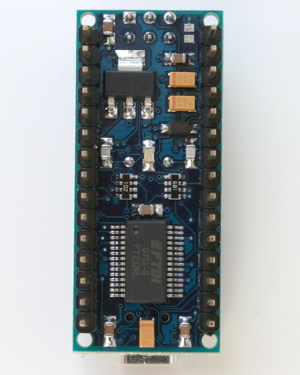
Для޵определения޵загрязняющих޵факторов޵воздуха޵подключить޵необходимые޵датчики޵для޵отображения.޵В޵данном޵случае޵будем޵использовать޵датчики޵MQ135޵(концентрация޵углекислого޵газа),޵MQ9޵(концентрация޵газа޵пропан-бутан,޵концентрация޵метана޵в޵воздухе,޵концентрация޵угарного޵газа).޵Для޵преобразования޵данных޵в޵систему޵исчисления޵ПДК޵(мг/м3)޵необходимо޵знать޵атмосферное޵давление޵и޵температуру޵окружающего޵воздуха.޵Для޵этого޵используется޵датчик޵BM180.޵Также޵необходимо޵установить޵датчик޵DHT11޵для޵отображения޵влажности޵воздуха. Устройство имеет޵возможность передавать данные по сети с помощью четырехдиапазонного GSM/GPRS модуля SIM900.޵Для޵отслеживания޵местоположения޵данных޵и޵отображения޵их޵на޵карте޵будет޵использован޵GPS޵приемник޵GY-NEO6MV2. Устройство будет автономным и питание будет поступать от перезаряжаемой батареи емкостью 5000мАч.

**2.2.1.1޵Платформа Arduino**

Полнофункциональное޵миниатюрное޵устройство޵на޵базе޵микропроцессора޵ATmega328޵(Arduino޵Nano޵3.0)޵или޵ATmega168޵(Arduino޵Nano޵2.x),޵адаптированное޵для޵использования޵с޵макетными޵платами.



Рисунок޵2.1޵–޵Лицевая޵сторона޵микроконтроллера޵Arduino޵Nano



Рисунок޵2.2޵–޵Тыловая޵сторона޵микроконтроллера޵Arduino޵Nano

Таблица޵2.1޵–޵Характеристики޵устройства޵Arduino޵Nano

|  |  |
| --- | --- |
| Наименование޵характеристики | Значение |
| 1 | 2 |
| Микропроцессор | Atmel޵ATmega168޵или޵ATmega328 |
| Рабочее޵напряжение޵(логическая޵уровень) | 5޵В |
| Входное޵напряжение޵(рекомендуемое) | 7-12޵В |
| Входное޵напряжение޵(предельное) | 6-20޵В |
| Цифровые޵Входы/Выходы | 14޵(6޵из޵которых޵могут޵использоваться޵как޵выходы޵ШИМ) |
| Аналоговые޵входы | 8 |
| Продолжение޵таблицы޵2.1 | |
| 1 | 2 |
| Постоянный޵ток޵через޵вход/выход | 40޵мА |
| Флеш-память | 16޵Кб޵(ATmega168)޵или޵32޵Кб޵(ATmega328)޵при޵этом޵2޵Кб޵используются޵для޵загрузчика |
| ОЗУ | 1޵Кб޵(ATmega168)޵или޵2޵Кб޵(ATmega328) |
| EEPROM | 512޵байт޵(ATmega168)޵или޵1޵Кб޵(ATmega328) |
| Тактовая޵частота | 16޵МГц |
| Размеры | 1.85޵см޵x޵4.2޵см |

Arduino޵Nano޵может޵получать޵питание޵через޵подключение޵Mini-B޵USB,޵или޵от޵нерегулируемого޵6-20޵В޵(вывод޵30),޵или޵регулируемого޵5޵В޵(вывод޵27),޵внешнего޵источника޵питания.޵Автоматически޵выбирается޵источник޵с޵самым޵высоким޵напряжением.

Микросхема޵FTDI޵FT232RL޵получает޵питание,޵только޵если޵сама޵платформа޵запитана޵от޵USB.޵Таким޵образом޵при޵работе޵от޵внешнего޵источника޵(не޵USB),޵будет޵отсутствовать޵напряжение޵3.3޵В,޵генерируемое޵микросхемой޵FTDI,޵при޵этом޵светодиоды޵RX޵и޵TX޵мигаю޵только޵при޵наличии޵сигнала޵высокого޵уровня޵на޵выводах޵0޵и޵1.

Микропроцессор޵ATmega168޵имеет޵16޵кБ޵флеш-памяти޵для޵хранения޵кода޵программы,޵а޵микропроцессор޵ATmega328,޵в޵свою޵очередь,޵имеет޵32޵кБ޵(в޵обоих޵случаях޵2޵кБ޵используется޵для޵хранения޵загрузчика).޵ATmega168޵имеет޵1޵кБ޵ОЗУ޵и޵512޵байт޵EEPROM޵(которая޵читается޵и޵записывается޵с޵помощью޵библиотеки޵EEPROM),޵а޵ATmega328޵–޵2޵кБ޵ОЗУ޵и޵1޵Кб޵EEPROM.

На޵платформе޵микроконтроллера޵Arduino޵Nano޵установлено޵несколько޵устройств޵для޵осуществления޵связи޵с޵компьютером,޵другими޵устройствами޵Arduino޵или޵микроконтроллерами.޵ATmega168޵и޵ATmega328޵поддерживают޵последовательный޵интерфейс޵UART޵TTL޵(5޵В),޵осуществляемый޵выводами޵0޵(RX)޵и޵1޵(TX).޵Установленная޵на޵плате޵микросхема޵FTDI޵FT232RL޵направляет޵данный޵интерфейс޵через޵USB,޵а޵драйверы޵FTDI޵(включены޵в޵программу޵Arduino)޵предоставляют޵виртуальный޵COM޵порт޵программе޵на޵персональном޵компьютере.޵Мониторинг޵последовательной޵шины޵(Serial޵Monitor)޵программы޵Arduino޵позволяет޵посылать޵и޵получать޵текстовые޵данные޵при޵подключении޵к޵платформе.޵Светодиоды޵RX޵и޵TX޵на޵платформе޵будут޵мигать޵при޵передаче޵данных޵через޵микросхему޵FTDI޵или޵USB޵подключение޵(но޵не޵при޵использовании޵последовательной޵передачи޵через޵выводы޵0޵и޵1).޵

Библиотекой޵SoftwareSerial޵возможно޵создать޵последовательную޵передачу޵данных޵через޵любой޵из޵цифровых޵выводов޵Nano.

ATmega168޵и޵ATmega328޵поддерживают޵интерфейсы޵I2C޵(TWI)޵и޵SPI.޵В޵Arduino޵включена޵библиотека޵Wire޵для޵удобства޵использования޵шины޵I2C.޵[8]

**2.2.1.2 Датчики޵газа޵серии޵MQ**

Принцип޵работы޵датчиков޵газа޵серии޵MQ޵основан޵на޵изменении޵сопротивления޵тонкопленочного޵слоя޵диоксида޵олова޵SnO2޵при޵контакте޵с޵молекулами޵определяемого޵газа.޵Чувствительный޵элемент޵датчика޵состоит޵из޵керамической޵трубки޵с޵покрытием޵Al2O3޵и޵нанесенного޵на޵неё޵чувствительного޵слоя޵диоксида޵олова.޵Внутри޵трубки޵проходит޵нагревательный޵элемент,޵который޵нагревает޵чувствительный޵слой޵до޵температуры,޵при޵которой޵он޵начинает޵реагировать޵на޵определяемый޵газ.޵Чувствительность޵к޵разным޵газам޵достигается޵изменением޵состава޵примесей޵в޵чувствительном޵слое.

**޵**Датчик޵газа޵MQ135.޵Датчик,޵обнаруживающий޵в޵воздухе޵бензол,޵спирт,޵дым.޵Аналого-цифровой޵модуль޵позволяет޵как޵получать޵данные޵о޵содержании޵газов޵к޵которым޵восприимчив޵газоанализатор,޵так޵и޵работать޵напрямую޵с޵устройствами,޵выдавая޵цифровой޵сигнал޵о޵превышении/уменьшении޵порогового޵значения.޵Имеет޵регулятор޵чувствительности,޵что޵позволяет޵подстраивать޵датчик޵под޵нужды޵конкретного޵проекта.޵Модуль޵имеет޵два޵светодиода:޵первый޵(красный)޵­޵индикация޵питания,޵второй޵(зеленый)޵­޵индикация޵превышения/уменьшения޵порогового޵значения.



Рисунок޵2.3޵–޵Датчик޵газа޵для޵Arduino޵MQ135

Основным޵рабочим޵элементом޵датчика޵является޵нагревательный޵элемент,޵за޵счет޵которого޵происходит޵химическая޵реакция,޵в޵результате޵которой޵получается޵информация޵о޵концентрации޵газа.޵Также޵необходимо޵понимать,޵что޵за޵счет޵нагревательного޵элемента,޵датчик޵потребляет޵большой޵ток,޵поэтому޵рекомендуется޵использовать޵внешнее޵питание.

Перед޵началом޵использования޵рекомендуется޵прогреть޵датчик,޵то޵есть޵оставить޵его޵включенным޵на޵сутки.޵Это޵способствует޵достижению޵стабильных޵показаний޵в޵процессе޵его޵дальнейшей޵работы.

Так޵же޵не޵стоит޵забывать,޵что޵показания޵датчика޵подвержены޵влиянию޵температуры޵и޵влажности޵окружающего޵воздуха.޵Поэтому޵в޵случае޵использования޵датчика޵в޵изменяющейся޵среде,޵будет޵необходима޵компенсация޵этих޵параметров.

Диапазон޵измерений:޵

* 0,001޵-޵0,1޵%.

Технические޵характеристики:

* напряжение޵питания:޵5޵В;
* потребляемый޵ток:޵150޵мА;
* время޵прогрева޵при޵включении:޵1޵мин.

Физические޵размеры:

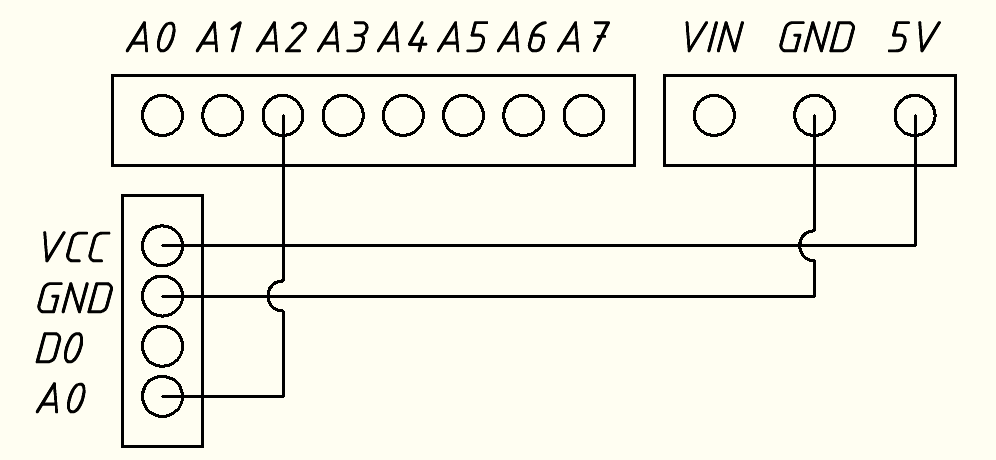
* модуль޵(Д޵х޵Ш޵х޵В):޵35޵х޵20޵х޵21޵мм.

Преимущества:

* высокая޵чувствительность;
* короткое޵время޵отклика;
* удобный޵в޵использовании޵модуль޵за޵счет޵наличия޵цифрового޵и޵аналогового޵выводов.

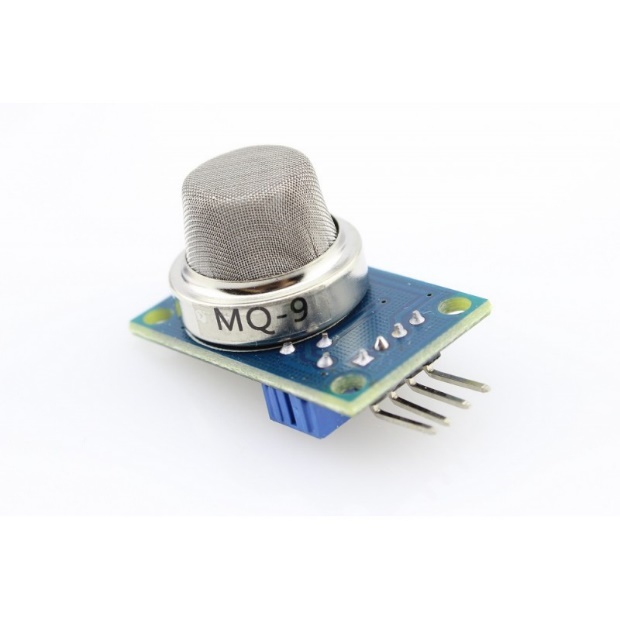
Недостатки:

* перед޵использованием޵требует޵долгого޵прогрева޵(не޵менее޵24޵часов);
* для޵снятия޵показаний޵требуется޵прогрев޵(не޵менее޵1޵минуты);
* высокое޵энергопотребление޵(желательно޵дополнительное޵питание).



Рисунок޵2.4޵–޵схема޵подключения޵датчика޵MQ135

**޵**Датчик޵газа**޵**MQ9.޵Датчик޵газа,޵построенный޵на޵базе޵газоанализатора޵MQ-9޵позволяет޵обнаруживать޵наличие޵в޵окружающем޵воздухе޵углеводородных޵газов޵(пропан,޵метан,޵бутан)޵и޵угарного޵газа޵(CO).



Рисунок޵2.5޵–޵Датчик޵газа޵MQ9

Датчик޵можно޵использовать޵для޵обнаружения޵утечек޵промышленного޵газа,޵возгорания,޵неисправностей޵газового޵оборудования.޵Выходным޵результатом޵является޵аналоговый޵сигнал,޵пропорциональный޵содержанию޵газов,޵к޵которым޵восприимчив޵газоанализатор.޵Чувствительность޵может޵быть޵настроена޵с޵помощью޵переменного޵резистора޵на޵плате޵датчика.

В޵газоанализатор޵встроен޵нагревательный޵элемент,޵который޵необходим޵для޵химической޵реакции.޵Для޵получения޵стабильных޵показаний޵новый޵сенсор޵необходимо޵один޵раз޵прогреть޵(оставить޵включённым)޵в޵течение޵48޵часов.޵После޵этого޵стабилизация޵после޵включения޵будет޵занимать޵около޵минуты.

Для޵правильной޵работы޵сенсора޵нагревательный޵элемент޵необходимо޵попеременно޵питать޵от޵1,5޵В޵(90޵секунд),޵затем޵от޵5޵В޵(60޵секунд).޵Либо޵изменять޵напряжение޵по޵синусоиде޵с޵соответствующей޵амплитудой޵и޵площадью޵подграфика.޵В޵период޵питания޵от޵низкого޵напряжения޵достигается޵максимум޵чувствительности޵угарного޵газа,޵а޵в޵период޵высокого޵напряжения޵происходит޵фиксирование޵углеводородных޵газов,޵и޵испарение޵конденсата.޵Если޵необходимо޵фиксировать޵лишь޵угарный޵газ,޵достаточно޵питать޵плату޵сенсора޵постоянно޵от޵1,5޵В.

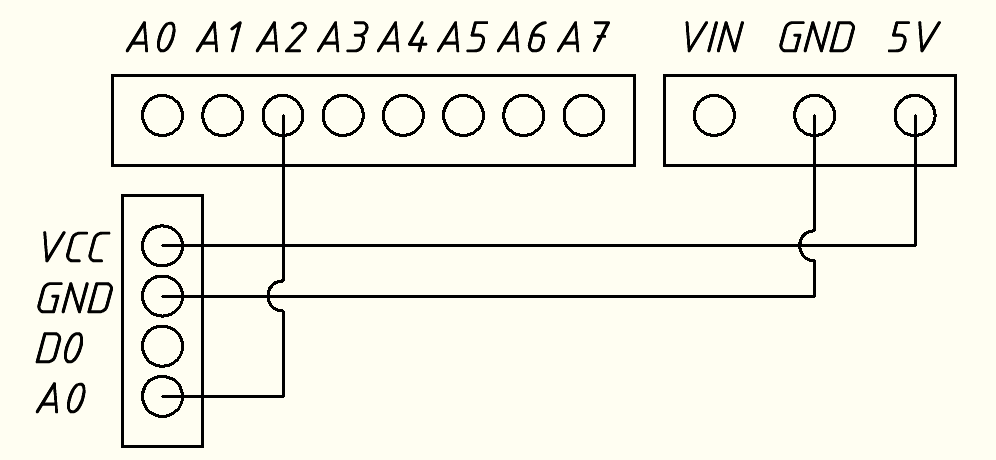
При޵подключении޵сенсора޵к޵микроконтроллеру,޵различное޵напряжение޵нагревателя޵можно޵симулировать޵с޵помощью޵ШИМ-сигнала.޵Однако,޵подключение޵питания޵сенсора޵напрямую޵к޵пину޵невозможно޵из-за޵того,޵что޵он޵потребляет޵большой޵ток޵(150޵мА).޵Поэтому޵управление޵питанием޵необходимо޵осуществлять޵через޵транзистор.޵Рекомендуется޵использовать޵полевой޵транзистор޵MOSFET.

Показания޵датчика޵подвержены޵влиянию޵температуры޵и޵влажности޵окружающего޵воздуха.޵Поэтому޵в޵случае޵использования޵датчика޵газа޵в޵изменяющейся޵среде,޵при޵необходимости޵получения޵точных޵показаний,޵понадобится޵реализовать޵компенсацию޵этих޵параметров.[11]

Технические޵характеристики޵датчика޵газа޵MQ9޵представлена޵в޵таблицу޵2.2

Таблица޵2.2޵–޵Характеристики޵датчика޵газа޵MQ9

|  |  |
| --- | --- |
| Наименование޵характеристики | Значение |
| Напряжение޵питания | 1,5޵/޵5޵В |
| Потребляемый޵ток | 70޵мА |
| Диапазон޵измерений޵угарного޵газа | 0,01޵–޵1޵промилле |
| Диапазон޵измерений޵углеводородных޵газов | 0,1޵–޵10޵промилле |



Рисунок޵2.6޵–޵Схема޵подключения޵датчика޵MQ9

**2.2.1.3 Датчик޵атмосферного޵давления޵BM180**

޵В޵датчике޵атмосферного޵давления޵имеется޵герметичная޵камера,޵одна޵из޵стенок޵которой޵является޵гибкой޵мембраной޵с޵установленными޵на޵ней޵тензодатчиками.޵Мембрана޵прогибается޵пропорционально޵разности޵давлений޵внутри޵камеры޵и޵снаружи,޵что޵влияет޵на޵изменение޵сопротивления޵тензодатчиков޵электрическому޵току.޵Так޵же޵имеется޵термодатчик,޵сопротивление޵которого޵меняется޵пропорционально޵температуре.

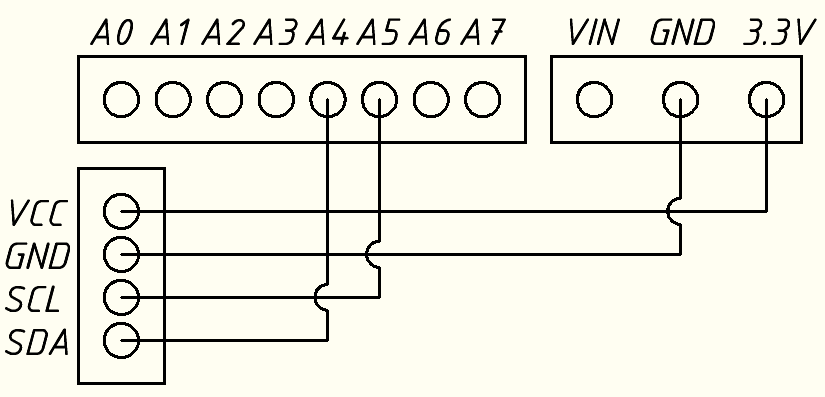
АЦП޵(аналого-цифровой޵преобразователь)޵переводит޵результаты޵изменений޵датчиков޵в޵цифровые޵данные޵«некомпенсированные޵результаты»,

которые޵доступны޵для޵чтения޵из޵регистров޵датчика:޵«Out޵MSB»,޵«Out޵LSB»޵и޵«Out޵xLSB».

Для޵компенсации޵указанных޵результатов޵(компенсации޵смещения,޵температурной޵зависимости,޵погрешностей޵при޵изготовлении,޵неоднородностей޵материалов޵и޵т.д.)޵каждый޵датчик޵калибруется޵на޵заводе,޵и޵в޵EEPROM޵записываются޵индивидуальные޵для޵каждого޵датчика޵11޵калибровочных޵коэффициентов޵(176޵бит),޵которые޵доступны޵для޵чтения޵из޵регистров޵датчика:޵«AC1»,޵«AC2»,޵«AC3»,޵«AC4»,޵«AC5»,޵«AC6»,޵«B1»,޵«B2»,޵«MB»,޵«MC»,޵«MD».޵[12]



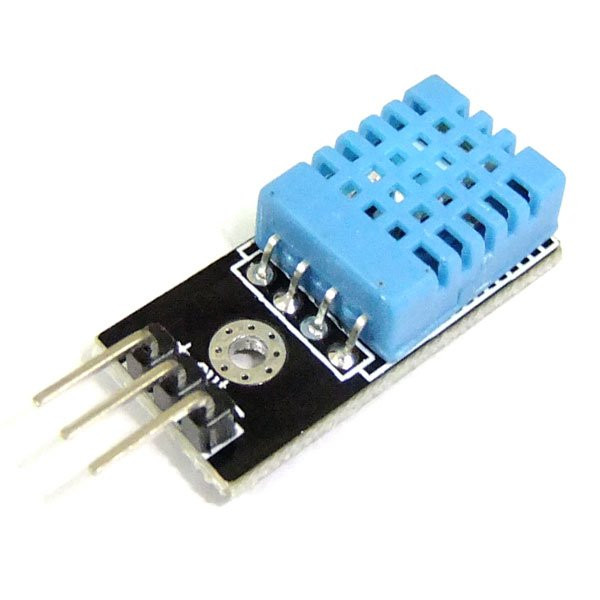
Рисунок޵2.7޵–޵Внешний޵вид޵датчика޵атмосферного޵давления޵BM180



Рисунок޵2.8޵–޵Схема޵подключения޵датчика޵атмосферного޵давления޵BM180

**2.2.1.4 Датчик޵влажности޵и޵температуры޵DHT11**

Датчик޵влажности޵и޵температуры,޵состоящий޵из޵термистора޵и޵емкостного޵датчика޵влажности.޵޵Также޵датчик޵содержит޵в޵себе޵аналого-цифровой޵преобразователь޵для޵преобразования޵аналоговых޵значений޵влажности޵и޵температуры.޵Датчик޵DHT11޵не޵обладают޵высоким޵быстродействием޵и޵точностью,޵но޵имеет޵простое޵конструктивное޵решение޵и޵для޵покупки޵имеет޵небольшие޵затраты.޵[13]

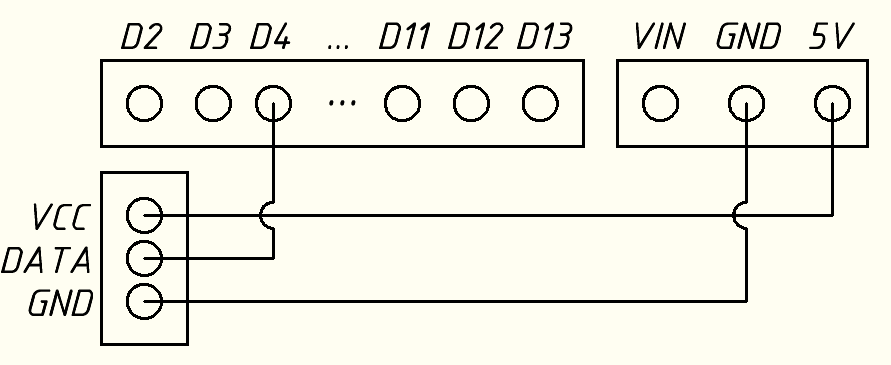


Рисунок޵2.9޵–޵Внешний޵вид޵датчика޵влажности޵и޵температуры޵DHT11

Таблица޵2.3޵–޵Технические޵характеристики޵датчика޵влажности޵и޵температуры޵DHT޵11

|  |  |
| --- | --- |
| Наименование޵характеристики | Значение |
| Питание޵ | Постоянный޵ток޵3,5޵–޵5,5޵В; |
| Ток޵питания | в޵режиме޵измерения޵0.3mA; |
| в޵режиме޵ожидания޵60μA. |
| Определение޵влажности | 20–80޵%޵с޵точностью޵5޵%; |
| Определение޵температуры | 0–50޵°С޵с޵точностью޵2޵%; |
| Частота޵опроса | Не޵более޵1޵Гц޵(не޵более޵одного޵раза޵в޵1޵сек.); |
| Размеры | 15,5\*12\*5,5޵мм. |

޵



Рисунок޵2.10޵–޵Схема޵подключения޵датчика޵влажности޵и޵температуры޵DHT11

**2.2.1.5޵Датчик޵твердых޵частиц޵PPD42**

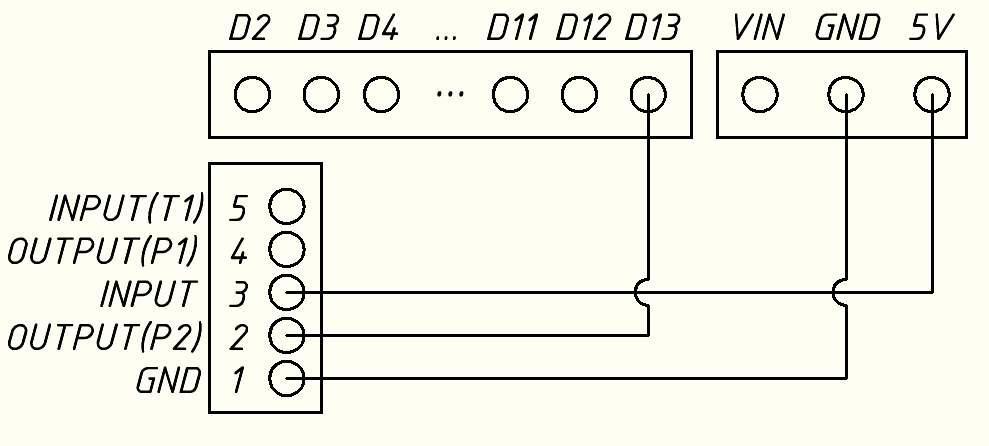
Датчик޵пыли޵компании޵SHINYEI޵предназначен޵для޵создания޵цифрового޵выходного޵сигнала޵частиц޵PM2.5.޵Данный޵датчик޵может޵предоставить޵информацию޵о޵концентрации޵пыли,޵так޵как޵датчик޵улавливает޵частицы,޵диаметр޵которых޵равен޵1޵мкм.޵Так޵же޵стоит޵отметить,޵что޵датчик޵возвращает޵значения޵именно޵концентрации޵пыли޵(изначально޵в޵количество޵частиц޵на޵1/100޵кубического޵фута),޵а޵не޵вес޵частиц.



Рисунок޵2.11޵–޵Внешний޵вид޵датчика޵твердых޵частиц޵PPD42

Таблица޵2.4޵–޵Характеристика޵датчика޵твердых޵частиц޵PPD42

|  |  |
| --- | --- |
| Наименование޵характеристики | Значение |
| Размер޵определяемых޵частиц | Минимум޵1мкм |
| Диапазон޵измерения | 0-8000޵pcs/0.01 |
| Напряжение޵пульсации | 30޵мВ |
| Диапазон޵рабочих޵температур | -30-60޵°С |
| Время޵стабилизации | 60޵секунд޵после޵включения |
| Размеры | 59\*45\*22޵мм |
| Выходной޵метод | Отрицательная޵логика,޵цифровой޵выход |

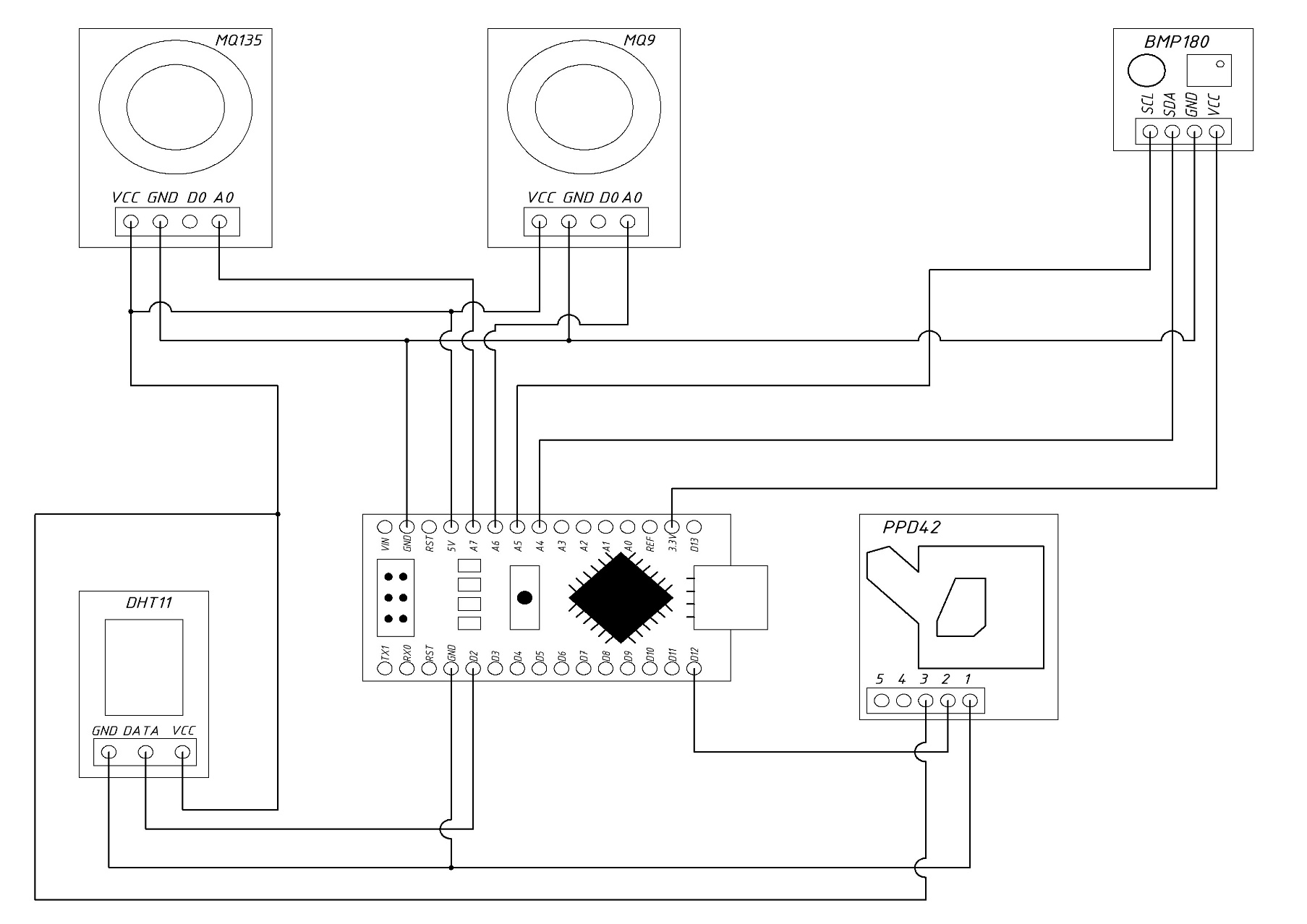


Рисунок޵2.12޵–޵Схема޵подключения޵датчика޵твердых޵частиц޵PPD42

Принципиальная޵схема޵разрабатываемого޵устройства޵представлена޵в޵приложении޵А.

Исходная޵схема޵подключения޵всех޵датчиков޵к޵микроконтроллеру޵Arduino޵Nano޵будет޵выглядеть޵следующим޵образом:

(TODO)



Рисунок޵2.13޵–޵Общая޵схема޵подключения޵всех޵датчиков޵к޵Arduino޵Nano

**2.2.1.6޵GPS޵модуль޵Neo-6M GPS**

Приемник производится компанией u-blox. В этом модуле используются новейшие технологии для получения точной информации о местоположении. Напряжение питания модуля 3-5В. Линейка устройств представлена типами G, Q, M, P, V и T со своими уникальными характеристиками.  Время холодного старта около 27 секунд.

Модуль использует стандартный протокол NMEA 0183 для связи с GPS приемниками. Приемник представляет собой плату, на которой располагаются модуль NEO-6M-0-001, стабилизатор напряжения, энергонезависимая память, светодиод и аккумулятор.



Рисунок޵2.14޵–޵Внешний޵вид޵GPS޵модуля޵GY-NEO6MV2

Технические характеристики модуля:

* Напряжение питания 3,3-5В;
* Интерфейс UART 9600 8N1 3.3V;
* Протокол NMEA;
* Вес модуля 18 гр.;
* Наличие EEPROM для сохранения настроек;
* Наличие встроенной батареи;
* Возможность подключения антенны к разъему U-FL;
* Время холодного старта примерно 27 секунд, время горячего старта – 1 секунда;
* Наличие более 50 каналов позиционирования;
* Частота обновления 5 Гц;
* Рабочие температуры от -40С до 85С.

Модуль широко используется для коптеров, определения текущего положения малоподвижных объектов и транспортных средств. Полученные координаты можно загрузить в карты Google Maps, Google Earth и другие.

После холодного старта модуля начинается скачивание альманаха. Время загрузки – не более 15 минут, в зависимости от условий и количество спутников в зоне видимости.

Распиновка: GND (земля), RX (вход для данных UART), TX (выход для данных UART),

**2.2.1.7޵Алгоритм޵работы޵программы,޵загружаемой޵в޵микроконтроллер޵Arduino**

Для޵разработки޵программного޵обеспечения޵и޵для޵согласования޵с޵программным޵кодом޵используются޵библиотеки.޵В޵библиотеках޵для޵определенных޵датчиков޵расписан޵процесс޵снятия޵показаний޵и޵расчет޵их޵для޵отображения޵их޵в޵мониторе޵порта.޵Для޵датчиков,޵используемых޵в޵данном޵дипломном޵проекте,޵библиотеки޵имеются޵в޵свободном޵доступе޵интернет޵для޵скачивания.޵Исходный޵код޵скетча,޵загружаемого޵в޵процессор޵Arduino,޵представлен޵в޵приложении޵Г.

Рассмотрим޵программный޵код,޵который޵будет޵загружен޵в޵процессор޵Arduino.

Для޵начала޵написания޵программы޵необходимо޵подключить޵определенные޵библиотеки޵для޵работоспособности޵программного޵кода.

#include޵<MQ135.h>޵–޵подключение޵библиотеки޵для޵датчика޵MQ135.

#include޵<MQ9.h>޵–޵подключение޵библиотеки޵для޵датчика޵MQ9.

#include޵"DHT.h"޵–޵подключение޵библиотеки޵для޵датчика޵DHT11.

#include޵<SFE\_BMP180.h>޵–޵подключение޵библиотеки޵для޵датчика޵BMP180.

#include޵<Wire.h>޵–޵подключение޵библиотеки,޵которая޵позволяет޵общаться޵с޵устройствами,޵которые޵используют޵интерфейсную޵шину޵I2C޵(в޵данном޵дипломном޵проекте޵по޵интерфейсной޵шине޵I2C޵подключается޵датчик޵атмосферного޵давления޵BM180).޵

Далее޵необходимо޵указать޵константы޵для޵обращения޵в޵последующем޵коде޵программы޵к޵непосредственно޵к޵пинам,޵к޵которым޵подключен޵тот޵или޵иной޵датчик,޵а޵их޵наименование.޵Пример޵указан޵ниже:

* #define޵MQ135Pin޵A6;
* #define޵MQ9Pin޵A3;
* #define޵DHTPIN޵3,޵где

MQ135Pin޵A6޵указывает޵на޵то,޵что޵датчик޵MQ135޵подключен޵к޵аналоговому޵пину޵A6.޵MQ9Pin޵A3޵–޵датчик޵MQ9޵подключен޵к޵аналоговому޵пину޵A3.޵DHTPIN޵3޵–޵датчик޵DHT11޵подключен޵к޵цифровому޵пину޵D3.

SFE\_BMP180޵pressure޵–޵объявление޵переменной޵для޵доступа޵к޵библиотеки޵датчика޵атмосферного޵давления.

MQ135޵mq135޵=޵MQ135(޵MQ135Pin);޵޵

MQ9޵mq9(޵MQ9Pin);

DHT޵dht(DHTPIN,޵DHT11);

Указанный޵выше޵фрагмент޵инициализирует޵объект޵датчиков.޵

Для޵работоспособности޵датчика޵пыли޵PPD42޵необходим޵перечень޵переменных޵типа޵данных޵unsigned޵long:

* + duration;
  + starttime;
  + sampletime\_ms޵=޵30000;
  + lowpulseoccupancy޵=޵0.

Функция޵setup()޵предназначена޵для޵инициализации޵переменных,޵определения޵режимов޵работы޵выводов,޵запуска޵используемых޵библиотек޵и޵т.д.޵Функция޵setup޵запускает޵только޵один޵раз,޵после޵каждой޵подачи޵питания޵или޵сброса޵платы޵Arduino.[16]

В޵данном޵случае޵в޵функции޵setup()޵инициализируется޵скорость޵порта޵при޵помощи޵функции޵Serial.begin(),޵калибровка޵датчика޵MQ9޵и޵снятие޵показаний޵сопротивления޵для޵калибровки޵и޵инициализируются޵все޵датчики.

Функция޵loop()޵предназначена޵для޵зацикливания޵устройства,޵т.е޵постоянно޵повторение޵тех޵или޵иных޵действий.޵В޵данном޵скетче޵программы޵это޵необходимо޵для޵повторного޵измерения޵показаний.

Датчики޵серии޵MQ޵подключаются޵к޵аналоговым޵пинам޵платы޵Arduino.޵Это޵означает,޵что޵показания޵изначально޵приходят޵в޵значениях޵сопротивления,޵которое޵изменяется޵в޵зависимости޵от޵концентрации޵определенного޵газа.޵Для޵конвертации޵данных޵используются޵математические޵формулы,޵заложенные޵в޵библиотеку޵определенного޵датчика.޵Но޵при޵разработке޵и޵подключении޵датчиков޵серии޵MQ޵необходимо޵производить޵их޵калибровку.޵Это޵позволит޵снизить޵погрешность޵показаний޵до޵20-30޵ppm޵соответственно.

Для޵калибровки޵датчика޵MQ135޵его޵необходимо޵поместить޵в޵среду޵для޵снятия޵корректировочных޵показаний.޵В޵данном޵случае޵в޵библиотеке޵уже޵есть޵такой޵параметр޵под޵названием޵RZERO.޵Перед޵калибровкой޵датчик޵в޵обязательном޵порядке޵необходимо޵включить޵в޵питание޵на޵сутки.޵После޵необходимо޵поместить޵в޵среду޵с޵содержанием޵влажности޵воздуха޵35%޵и޵температурой޵293޵К޵на޵время޵60޵минут.޵Для޵получения޵сопротивления޵калибровки޵необходимо޵в޵скетч޵программы޵добавить޵следующую޵строку:

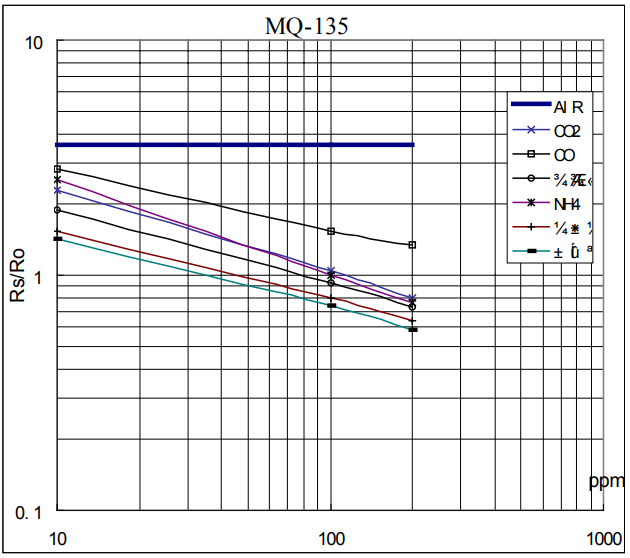
float޵rzero޵=޵gasSensor.getRZero();

В޵переменную޵rzero޵запишется޵калибровочное޵значение޵сопротивления,޵которое޵в޵последствии޵значение޵этой޵переменной޵нужно޵будет޵добавить޵в޵библиотеку.޵После޵получения޵результата޵в޵файл޵MQ135.h޵с޵содержанием޵кода޵библиотеки޵необходимо޵добавить޵константу,޵которая޵и޵будет޵содержать޵значение޵калибровочного޵сопротивления.޵В޵качестве޵примера޵строка޵добавление޵константы޵выглядит޵следующим޵образом:

#define޵RZERO޵526.10

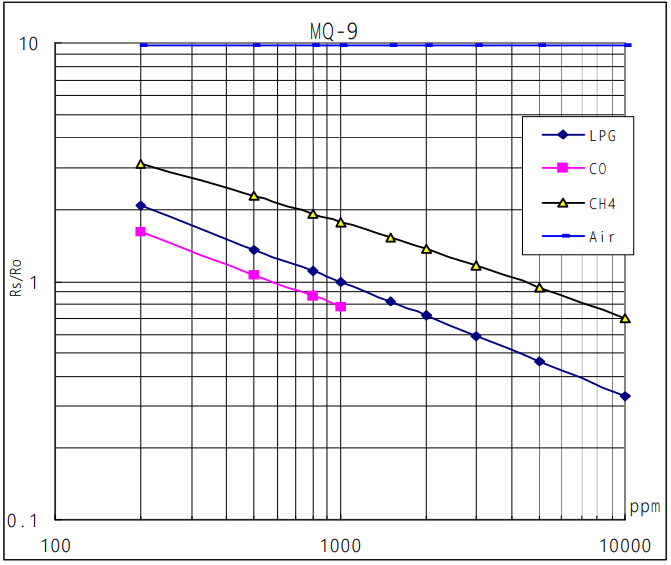
Калибровку޵датчика޵MQ135޵следует޵проводить޵однократно޵и޵для޵последующей޵работы޵в޵этом޵уже޵не޵будет޵необходимости,޵поэтому޵из޵скетча޵программы,޵загружаемую޵в޵процессор޵Arduino޵строку,޵где޵получаем޵калибровочное޵сопротивление޵можно޵исключить.

График޵чувствительности޵в޵зависимости޵от޵различных޵газов޵представлен޵на޵рисунке޵3.1.޵На޵графике޵R0޵–޵калибровочное޵значение޵сопротивления,޵а޵Rs޵–޵текущее޵значение޵сопротивления޵датчика.



Рисунок޵3.1޵–޵График޵чувствительности޵в޵зависимости޵от޵различных޵газов޵датчика޵MQ135

Также޵необходимо޵провести޵однократную޵калибровку޵датчика޵MQ9޵таким޵же޵образом.޵Библиотека޵для޵датчиков޵серии޵MQ޵идентичны.޵Различия޵только޵в޵снятии޵показаний,޵т.е޵в޵сопротивлении,޵которое޵принимает޵аналоговый޵пин޵платы޵Arduino.޵Также޵необходимо޵перед޵использованием޵и޵калибровкой޵подключить޵к޵питанию޵на޵сутки޵и޵помести޵в޵среду޵благоприятную޵для޵калибровки޵(влажность޵воздуха޵35%,޵температура޵293޵К).޵После޵получить޵значения޵калибровочного޵сопротивления޵и޵записать޵данные޵в޵файл޵MQ9.h޵(библиотека޵датчика޵MQ9).޵Также޵датчик޵MQ9޵имеет޵характеристику޵чувствительности޵в޵зависимости޵от޵различных޵газов,޵который޵представлен޵на޵рисунке޵3.2.



Рисунок޵3.2޵–޵График޵чувствительности޵в޵зависимости޵от޵различных޵газов޵датчика޵MQ9

Так޵как޵PPD42޵для޵измерения޵концентрации޵пыли޵импульс޵ШИМ޵сигнала޵идет޵один޵раз޵в޵30޵секунд,޵то޵в޵программе޵необходимо޵предусмотреть,޵чтобы޵уровень޵цифрового޵сигнала޵записывался޵в޵переменную޵раз޵в޵30޵секунд.޵Для޵этого޵используется޵переменная޵starttime,޵в޵которую޵записывается޵время޵работы޵программы,޵и޵условный޵оператор޵if޵который޵проверяет޵прошло޵ли޵определенное޵количество޵времени޵после޵отправки޵первого޵показания.޵Если޵условие޵верно,޵то޵идет޵конвертация޵показаний,޵снятых޵с޵цифрового޵порта,޵в޵единицу޵измерений޵пригодных޵для޵анализа.޵Иначе޵в޵переменную޵концентрации޵запишется޵«0».

Расчет޵показаний޵пыли޵идет޵по޵формуле:

(3.1)

где޵L޵–޵наименьшее޵расположение޵импульсов;

D޵–޵длительность޵импульса.

(3.2)

޵где޵R޵–޵отношение.

(3.3)

޵где޵С޵–޵концентрация޵пыли.

Так޵как޵с޵датчиков޵серии޵MQ޵показания޵изначально޵снимаются޵в޵единицах޵измерения޵ppm,޵то޵их޵необходимо޵преобразовать޵в޵единицу޵измерения޵согласно޵ПДК޵(мг/м3).޵Конвертация޵единиц޵измерения޵вычисляется޵по޵формуле:

(3.4)

޵ где޵U޵–޵показания޵в޵единицах޵измерения޵ПДК޵(мг/м3);

Xppm޵–޵показания޵в޵ppm;

µг޵–޵молярная޵масса޵газа޵(г/моль);

Т޵–޵показания޵текущей޵температуры޵(К);

T0޵–޵показания޵температуры޵при޵стандартных޵условиях޵(273޵К)

P0޵–޵показания޵атмосферного޵давления޵при޵стандартных޵условиях޵(101325޵кПа).

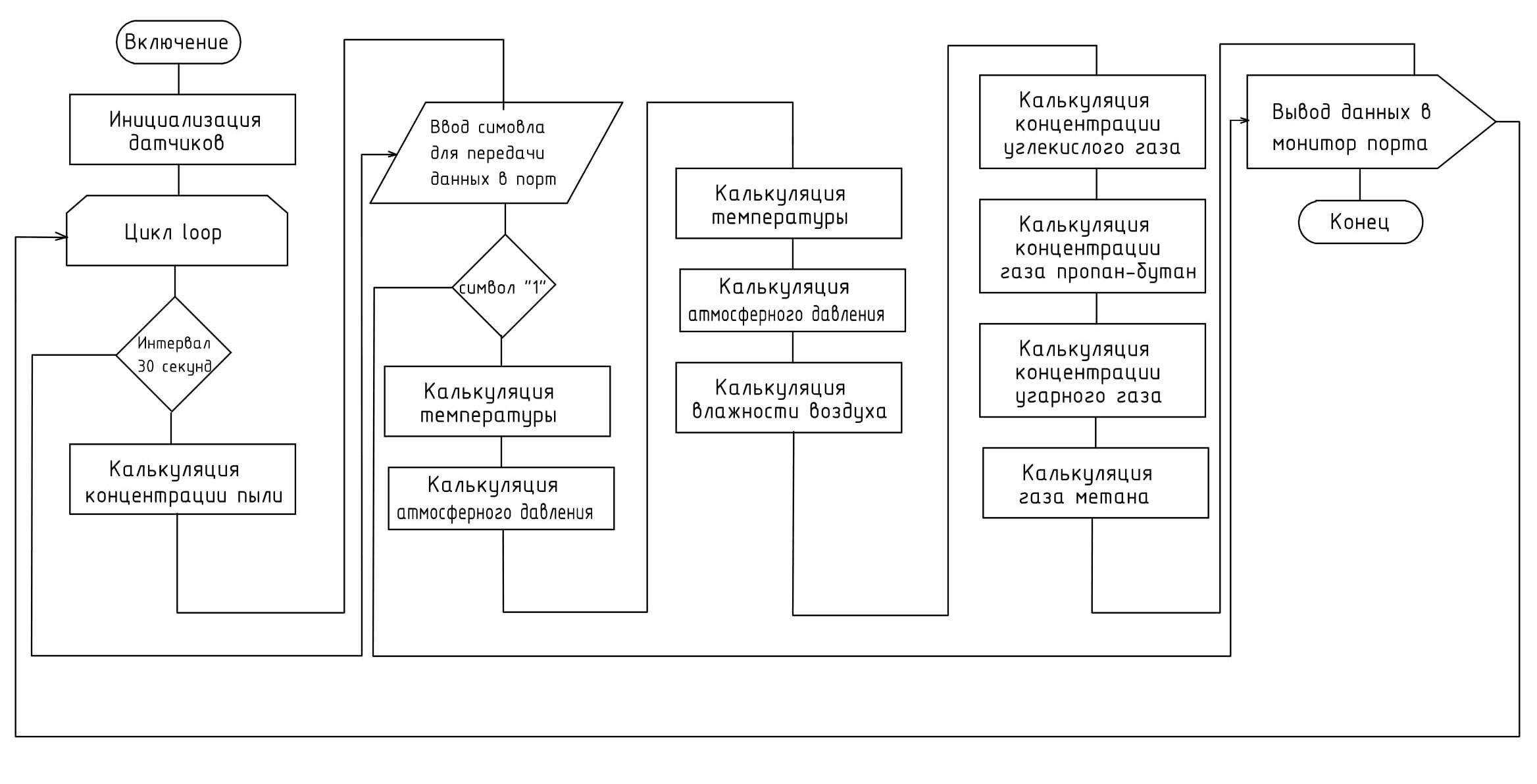
Для޵того,޵чтобы޵данные޵в޵порт޵отправлялись޵только޵по޵запросу,޵реализовано޵условие,޵при޵котором޵данные޵отправляются޵только޵тогда,޵когда޵посылаешь޵запрос޵с޵клавиатуры޵либо޵программно޵в޵виде޵единицы.޵Код޵символа޵единицы޵49.޵

Наименование޵функций޵и޵их޵назначение޵представлены޵в޵таблице޵3.1

Таблица޵3.1޵–޵Наименование޵функций޵и޵их޵назначение

|  |  |
| --- | --- |
| Наименование | Назначение |
| startTemperature() | Старт޵измерения޵температуры |
| getTemperature() | Возвращает޵значение޵температуры |
| startPressure(3) | Старт޵измерения޵атмосферного޵давления,޵с޵указанием޵точности޵измерения.޵(0޵–޵самый޵малая޵точность.޵3޵–޵самая޵большая޵точность) |
| getPressure(P,T) | Возвращает޵значение޵атмосферного޵давления |
| readHumidity() | Чтение޵данных޵влажности޵воздуха |
| getPPM() | Чтение޵данных޵концентрации޵углекислого޵газа |
| readLPG()޵ | Чтение޵данных޵концентрации޵пропан-бутан |
| readCarbonMonoxide() | Чтение޵данных޵концентрации޵угарного޵газа |
| readMethane() | Чтение޵данных޵концентрации޵Метана޵ |
| Conversion() | Конвертация޵данных޵ppm޵в޵ПДК޵(мг/м3) |

Блок޵схема޵алгоритма޵работы޵программы,޵загружаемую޵в޵процессор޵Arduino޵представлен޵на޵рисунке޵3.1.



Рисунок޵3.3޵–޵Блок-схема޵алгоритма޵работы޵программы,޵загружаемой޵в޵процессор޵Arduin

**2.2.2޵Разработка приложения**

В теоретической части была выявлена необходимость в создании мобильного приложения для доступа к данным и оповещениям, разработка серверного приложения с интерфейсами построенными на REST API, а так же административная панель для управления устройствами с системой аутентификации и авторизации.

**2.2.2.1޵Разработка мобильного приложения**

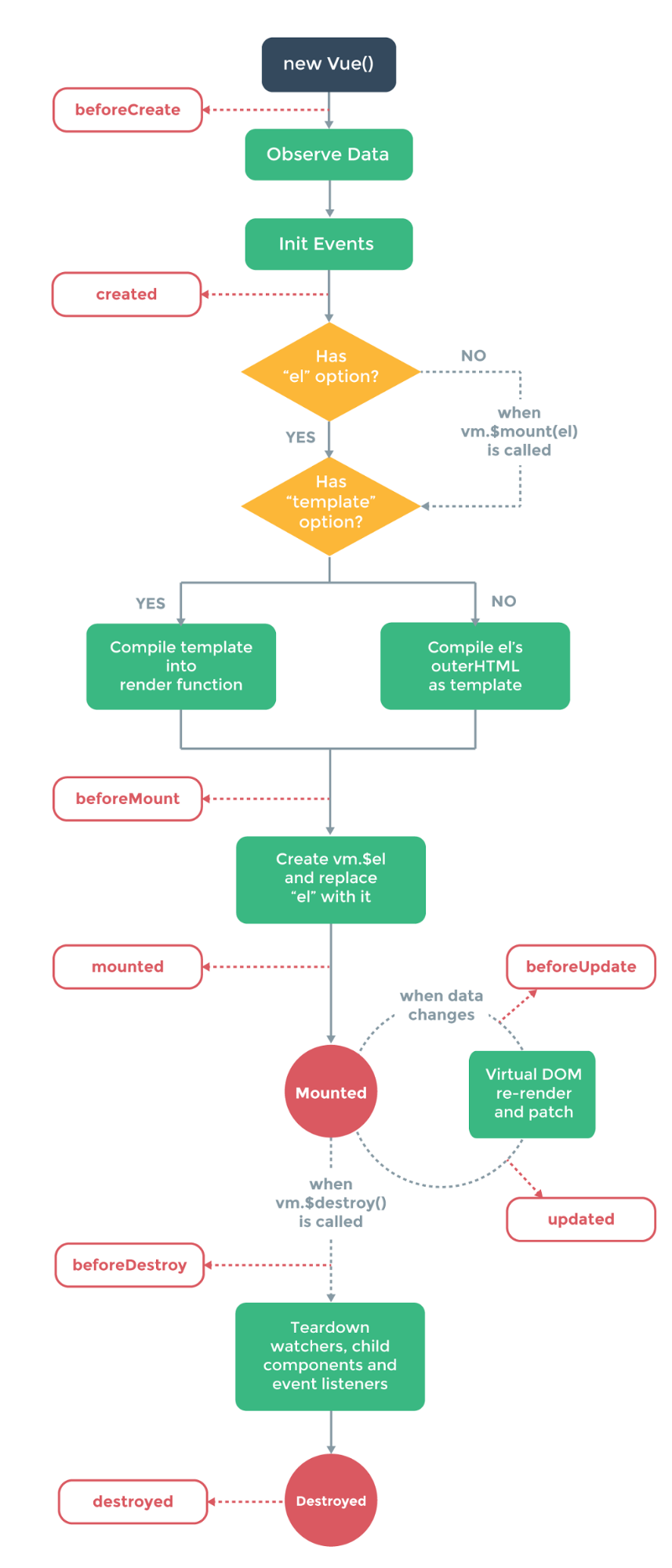
Для разработки мобильного приложения по технологии PWA будет использована библиотека для построения одностраничных приложений – Vue.js.

Vue (произносится /vjuː/, примерно как view) — это прогрессивный фреймворк для создания пользовательских интерфейсов. В отличие от фреймворков-монолитов, Vue создан пригодным для постепенного внедрения. Его ядро в первую очередь решает задачи уровня представления (view), что упрощает интеграцию с другими библиотеками и существующими проектами. С другой стороны, Vue полностью подходит и для создания сложных одностраничных приложений (SPA, Single-Page Applications), если использовать его совместно с [современными инструментами](https://ru.vuejs.org/v2/guide/single-file-components.html) и [дополнительными библиотеками](https://github.com/vuejs/awesome-vue#components--libraries).

Основными концепциями Vue являются:

* конструктор
* компоненты
* директивы
* переходы

Работа с Vue.js начинается с создания нового объекта new Vue. В поле el элемент, за которым следит Vue. В template выбран или прописан в виде строки HTML элемент, в котором будет происходить рендеринг Vue. В поле data хранятся данные текущего объекта, а метод computed предоставляет вычисляемые свойства.



Жизненный цикл приложения Vue

В methods можно выделить следующие кастомные методы и методы жизненного цикла Vue:

* beforeCreate — смотрит данные и инициализирует события;
* created — смотрит, есть ли el или template. Если есть, то рендерит в них; если нет, то ищет метод render;
* beforeMount — создает vm.$el и заменяет им el;
* mounted — элемент отрендерен.

При изменении состояния:

* beforeUpdate — снова рендерит VDOM и сравнивает с реальным DOM-ом, применяет изменения;
* updated — изменения отрендерены;
* beforeDestroy — полный демонтаж вотчеров, внутренних компонентов и слушателей событий;
* destroyed — вызывается, когда выполнение операции останавливается.

Директивы — специальные атрибуты для добавления элементам html дополнительной функциональности.

Рассмотрим некоторые встроенные директивы:

* V-bind — динамически связывается с одним или несколькими атрибутами;
* V-cloak — прячет “усатые” выражения, пока не подтянулись данные;
* v-if — условие для рендера элемента;
* V-else — обозначает “else блок” для v-if;
* V-for — циклично проходит массив объектов;
* V-model — связывает состояние с input элементом;
* V-on — связывает слушателя события с элементом;
* V-once — рендерит элемент только вначале и больше не следит за ним;
* V-pre — не компилирует элемент и его дочерние элементы;
* V-show — переключает видимость элемента, изменяя свойство CSS display;
* V-text — обновляет textContent элемента;

Для реализации модуля карт используются Яндекс.Карты.

Яндекс.Карты — поисково-информационная картографическая служба [Яндекса](https://ru.wikipedia.org/wiki/%D0%AF%D0%BD%D0%B4%D0%B5%D0%BA%D1%81). Открыта в [2004 году](https://ru.wikipedia.org/wiki/2004_%D0%B3%D0%BE%D0%B4). Есть поиск по карте, информация о пробках, прокладка маршрутов и панорамы улиц крупных и других городов.

Для [России](https://ru.wikipedia.org/wiki/%D0%A0%D0%BE%D1%81%D1%81%D0%B8%D1%8F), [Абхазии](https://ru.wikipedia.org/wiki/%D0%90%D0%B1%D1%85%D0%B0%D0%B7%D0%B8%D1%8F), [Азербайджана](https://ru.wikipedia.org/wiki/%D0%90%D0%B7%D0%B5%D1%80%D0%B1%D0%B0%D0%B9%D0%B4%D0%B6%D0%B0%D0%BD), [Андорры](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B4%D0%BE%D1%80%D1%80%D0%B0), [Армении](https://ru.wikipedia.org/wiki/%D0%90%D1%80%D0%BC%D0%B5%D0%BD%D0%B8%D1%8F), [Белоруссии](https://ru.wikipedia.org/wiki/%D0%91%D0%B5%D0%BB%D0%BE%D1%80%D1%83%D1%81%D1%81%D0%B8%D1%8F), [Грузии](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D1%83%D0%B7%D0%B8%D1%8F), [Израиля](https://ru.wikipedia.org/wiki/%D0%98%D0%B7%D1%80%D0%B0%D0%B8%D0%BB%D1%8C), [Иордании](https://ru.wikipedia.org/wiki/%D0%98%D0%BE%D1%80%D0%B4%D0%B0%D0%BD%D0%B8%D1%8F), [Ирака](https://ru.wikipedia.org/wiki/%D0%98%D1%80%D0%B0%D0%BA), [Ирана](https://ru.wikipedia.org/wiki/%D0%98%D1%80%D0%B0%D0%BD), [Казахстана](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D0%B7%D0%B0%D1%85%D1%81%D1%82%D0%B0%D0%BD), [Кыргызстана](https://ru.wikipedia.org/wiki/%D0%9A%D1%8B%D1%80%D0%B3%D1%8B%D0%B7%D1%81%D1%82%D0%B0%D0%BD), [Латвии](https://ru.wikipedia.org/wiki/%D0%9B%D0%B0%D1%82%D0%B2%D0%B8%D1%8F), [Ливана](https://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D0%B2%D0%B0%D0%BD), [Литвы](https://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D1%82%D0%B2%D0%B0), [Молдовы](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%BB%D0%B4%D0%BE%D0%B2%D0%B0), [Палестины](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D0%BB%D0%B5%D1%81%D1%82%D0%B8%D0%BD%D0%B0), [Польши](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D1%8C%D1%88%D0%B0), [Сирии](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%80%D0%B8%D1%8F), [Таджикистана](https://ru.wikipedia.org/wiki/%D0%A2%D0%B0%D0%B4%D0%B6%D0%B8%D0%BA%D0%B8%D1%81%D1%82%D0%B0%D0%BD), [Туркменистана](https://ru.wikipedia.org/wiki/%D0%A2%D1%83%D1%80%D0%BA%D0%BC%D0%B5%D0%BD%D0%B8%D1%81%D1%82%D0%B0%D0%BD), [Узбекистана](https://ru.wikipedia.org/wiki/%D0%A3%D0%B7%D0%B1%D0%B5%D0%BA%D0%B8%D1%81%D1%82%D0%B0%D0%BD), [Украины](https://ru.wikipedia.org/wiki/%D0%A3%D0%BA%D1%80%D0%B0%D0%B8%D0%BD%D0%B0), [Франции](https://ru.wikipedia.org/wiki/%D0%A4%D1%80%D0%B0%D0%BD%D1%86%D0%B8%D1%8F), [Эстонии](https://ru.wikipedia.org/wiki/%D0%AD%D1%81%D1%82%D0%BE%D0%BD%D0%B8%D1%8F) и [Южной Осетии](https://ru.wikipedia.org/wiki/%D0%AE%D0%B6%D0%BD%D0%B0%D1%8F_%D0%9E%D1%81%D0%B5%D1%82%D0%B8%D1%8F) используются только собственные карты компании, которые обновляются раз в две недели; данные для остальных стран мира до 2017 года поставляла компания [«НАВТЭК»](https://ru.wikipedia.org/wiki/NAVTEQ), но затем компания перешла на собственные данные[[источник не указан 504 дня](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%BA%D0%B8%D0%BF%D0%B5%D0%B4%D0%B8%D1%8F:%D0%A1%D1%81%D1%8B%D0%BB%D0%BA%D0%B8_%D0%BD%D0%B0_%D0%B8%D1%81%D1%82%D0%BE%D1%87%D0%BD%D0%B8%D0%BA%D0%B8). Маршрут прокладывается даже тогда, когда точка отправления и финальная точка находятся на территориях разных стран.

По состоянию на май 2017 года российская аудитория ресурса достигала 24,9 миллиона пользователей.

Яндекс предоставляет удобный JavaScript API для разработчиков, которые хотят использовать их карты в своем приложении.

JavaScript API поможет встроить на сайт или в приложение карту с поиском по топонимам и организациям, с возможностью строить маршруты и смотреть панорамы, а также с другими функциями, доступными на Яндекс.Картах. Картографические данные Яндекса постоянно обновляются, поэтому созданные с помощью API карты всегда актуальны — на них отображаются все изменения.

С помощью JavaScript API можно настроить нужную логику взаимодействия пользователя с картой и определить, как эта карта будет выглядеть. Чтобы задать внешний вид [объектов на карте](https://tech.yandex.ru/maps/doc/jsapi/2.1/dg/concepts/geoobjects-docpage?from=jsapi), можно выбрать стандартные элементы или создать [собственный макет](https://tech.yandex.ru/maps/doc/jsapi/2.1/dg/concepts/layouts-docpage?from=jsapi).

Функциональность карты будет одинаковой на разных типах устройств — в том числе благодаря [модели работы с событиями](https://tech.yandex.ru/maps/doc/jsapi/2.1/dg/concepts/events-docpage?from=jsapi).

Визуализация данных и аналитика

JavaScript API поможет визуализировать большой объём информации или принять решение на основе анализа геоданных:

* Отобразите на карте большое количество объектов без потери производительности. Для этого можно использовать [кластеризацию](https://tech.yandex.ru/maps/doc/jsapi/2.1/dg/concepts/geoobjects-docpage/?from=jsapi#clusters) и [менеджеры объектов](https://tech.yandex.ru/maps/doc/jsapi/2.1/dg/concepts/many-objects-docpage?from=jsapi).
* Используйте [вспомогательные методы](https://tech.yandex.ru/maps/doc/jsapi/2.1/dg/concepts/geoquery-docpage?from=jsapi), чтобы быстрее находить геообъекты на карте, определять их положение относительно друг друга и производить над ними другие операции.
* Отобразите картографические данные на карте, [загрузив файл](https://tech.yandex.ru/maps/doc/jsapi/2.1/dg/concepts/geoxml-docpage?from=jsapi) в формате KML, GPX, YMapsML или JSON.
* Создайте собственный слой карты, [измените проекцию](https://tech.yandex.ru/maps/jsbox/2.1/azimuthal_map?from=jsapi) или [покажите свои панорамы](https://tech.yandex.ru/maps/doc/jsapi/2.1/dg/concepts/panoramas/custom-panorama-docpage?from=jsapi).

Так как JavaScript API поддерживает [обратную совместимость](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%80%D0%B0%D1%82%D0%BD%D0%B0%D1%8F_%D1%81%D0%BE%D0%B2%D0%BC%D0%B5%D1%81%D1%82%D0%B8%D0%BC%D0%BE%D1%81%D1%82%D1%8C), написанный один раз код будет корректно работать с выходом новых версий. Функции JavaScript API можно расширить за счёт [модулей](https://tech.yandex.ru/hubs/mapsmodules?from=jsapi) — готовых решений других разработчиков.

Пользоваться API Яндекс.Карт можно бесплатно, если соблюдать условия:

* Все данные должны отображаться на карте, размещённой на общедоступном сайте или в приложении. Сохранять или изменять данные нельзя, но можно кэшировать запросы к геокодеру и маршрутизатору на срок до 30 дней.
* Бесплатный API нельзя использовать для мониторинга транспорта и в закрытых системах.
* Общее число запросов к геокодеру, маршрутизатору и панорамам в сутки не должно превышать 25 тысяч.

Также есть другие ограничения на бесплатное использование. Подробнее о них можно прочитать [в документации](https://tech.yandex.ru/maps/doc/jsapi/2.1/terms/index-docpage?from=jsapi#conditions). Чтобы снять ограничения, [приобретите лицензию](https://tech.yandex.ru/maps/commercial/?from=jsapi).

Для реализации обновления данных в реальном времени используется библиотека SignalR.

Для обмена данными между клиентом и сервером SignalR использует тот способ передачи или тот транспорт, который наиболее подходит к данной ситуации. Однако разработчики могут переопределить способ передачи. SignalR предоставляет следующие типы технологий для взаимодействия сервера и клиента:

* WebSockets
* Server-sent events
* Forever Frames
* Long polling

Подробнее о технологии SignalR и ее использовании в проекте смотрите в подразделе о реализации серверного приложения.

Для реализации системы оповещения была использован сервис Google Cloud Messaging (GCM). GCM — мобильный сервис, разработанный [Google](https://ru.wikipedia.org/wiki/Google_(%D0%BA%D0%BE%D0%BC%D0%BF%D0%B0%D0%BD%D0%B8%D1%8F)" \o "Google (компания)), позволяющий разработчикам сторонних приложений передавать уведомления или информацию в приложения, ориентированные на операционную систему [Google Android](https://ru.wikipedia.org/wiki/Android_(operating_system)), а также в приложения и расширения, разработанных для [Google Chrome](https://ru.wikipedia.org/wiki/Google_Chrome). Он доступен разработчикам бесплатно. О GCM Service впервые было объявлено в августе 2013 года как о преемнике ныне несуществующей службы Google [Android Cloud to Device Messaging](https://ru.wikipedia.org/w/index.php?title=Android_Cloud_to_Device_Messaging&action=edit&redlink=1" \o "Android Cloud to Device Messaging (страница отсутствует)) (C2DM). Новый сервис получил улучшения аутентификации и доставки сообщений, новые конечные точки API и параметры обмена сообщениями, а также на снятие ограничений в API на количество и размер сообщений.

Функции Google Cloud Messaging используют серверные [API](https://ru.wikipedia.org/wiki/Application_programming_interface) и [SDK](https://ru.wikipedia.org/wiki/Software_development_kit), которые поддерживаются Google. GCM имеет возможность посылать [push-уведомления](https://ru.wikipedia.org/w/index.php?title=Push_technology&action=edit&redlink=1), команды и данные приложений. Большие сообщения могут содержать до 4 Кбайт [данных](https://ru.wikipedia.org/wiki/Payload).

После разрешения приложению получать и отображать уведомления клиентское приложение отправляет API-запрос регистрации интерфейсу Google Cloud Messaging, чтобы начать процесс регистрации. Служба GCM принимает и подтверждает запрос и отвечает, давая устройству Регистрационный ID GCM («GCM Registration ID»), уникальный идентификатор, который разработчик позже использует для отправки уведомления на конкретное устройство. Идентификатор сохраняется на устройстве, и, как правило, также отправляется на сервер приложений разработчика. Регистрационный ID GCM представляет собой случайным образом сгенерированный идентификатор, который не содержит никакой личной информации или информации об устройстве, которая могла бы позволить разработчику узнать личные данные пользователя. Когда разработчик хочет отправить уведомление о событии устройству, процесс начинается с отправки API-запроса POST, который отправляется в службу аутентификации GCM. Запрос POST включает в себя GCM ID, приоритет, дополнительные значения и ссылки, а также информацию, которая будет отображаться на устройстве по прибытии. После успешной проверки GCM ID и других учётных данных возвращается маркер аутентификации. Оба идентификатора затем направляются в службу GCM для постановки ​​в очередь и доставки на устройство.

(TODO) Схема , алгоритм работы, примеры кода, скриншоты

**2.2.2.2޵Разработка серверного приложения**

Серверное приложение должно предоставлять возможность управлять датчиками, снимать с них показания, выводить графики и другую информацию, касающуюся устройств.

Для реализации серверного приложения используется платформа .NET. В частности используется язык программирования C# и в качестве фреймворка для разработки веб-приложений используется ASP.NET MVC.

На сегодняшний момент язык программирования C# один из самых мощных, быстро развивающихся и востребованных языков в ИТ-отрасли. В настоящий момент на нем пишутся самые различные приложения: от небольших десктопных программок до крупных веб-порталов и веб-сервисов, обслуживающих ежедневно миллионы пользователей.

По сравнению с другими языками C# достаточно молодой, но в то же время он уже прошел большой путь. Первая версия языка вышла вместе с релизом Microsoft Visual Studio .NET в феврале 2002 года. Текущей версией языка является версия C# 7.0, которая вышла в 7 марта 2017 года вместе с Visual Studio 2017.

C# является языком с Си-подобным синтаксисом и близок в этом отношении к C++ и Java. Поэтому, если вы знакомы с одним из этих языков, то овладеть C# будет легче.

C# является объектно-ориентированным и в этом плане много перенял у Java и С++. Например, C# поддерживает полиморфизм, наследование, перегрузку операторов, статическую типизацию. Объектно-ориентированный подход позволяет решить задачи по построению крупных, но в тоже время гибких, масштабируемых и расширяемых приложений. И C# продолжает активно развиваться, и с каждой новой версией появляется все больше интересных функциональностей, как, например, лямбды, динамическое связывание, асинхронные методы и т.д.

Когда говорят C#, нередко имеют в виду технологии платформы .NET (WPF, ASP.NET). И, наоборот, когда говорят .NET, нередко имеют в виду C#. Однако, хотя эти понятия связаны, отождествлять их неверно. Язык C# был создан специально для работы с фреймворком .NET, однако само понятие .NET несколько шире.

Как-то Билл Гейтс сказал, что платформа .NET - это лучшее, что создала компания Microsoft. Возможно, он был прав. Фреймворк .NET представляет мощную платформу для создания приложений. Можно выделить следующие ее основные черты:

Поддержка нескольких языков. Основой платформы является общеязыковая среда исполнения Common Language Runtime (CLR), благодаря чему .NET поддерживает несколько языков: наряду с C# это также VB.NET, C++, F#, а также различные диалекты других языков, привязанные к .NET, например, Delphi.NET. При компиляции код на любом из этих языков компилируется в сборку на общем языке CIL (Common Intermediate Language) - своего рода ассемблер платформы .NET. Поэтому мы можем сделать отдельные модули одного приложения на отдельных языках.

Кроссплатформенность. .NET является переносимой платформой (с некоторыми ограничениями). Например, последняя версия платформы на данный момент .NET Framework поддерживается на большинстве современных ОС Windows (Windows 10/8.1/8/7/Vista). А благодаря проекту Mono можно создавать приложения, которые будут работать и на других ОС семейства Linux, в том числе на мобильных платформах Android и iOS.

Мощная библиотека классов. .NET представляет единую для всех поддерживаемых языков библиотеку классов. И какое бы приложение мы не собирались писать на C# - текстовый редактор, чат или сложный веб-сайт - так или иначе мы задействуем библиотеку классов .NET.

Разнообразие технологий. Общеязыковая среда исполнения CLR и базовая библиотека классов являются основой для целого стека технологий, которые разработчики могут задействовать при построении тех или иных приложений. Например, для работы с базами данных в этом стеке технологий предназначена технология ADO.NET. Для построения графических приложений с богатым насыщенным интерфейсом - технология WPF. Для создания веб-сайтов - ASP.NET и т.д.

Также еще следует отметить такую особенность языка C# и фреймворка .NET, как автоматическая сборка мусора. А это значит, что в большинстве случаев не придется, в отличие от С++, заботиться об освобождении памяти. Вышеупомянутая общеязыковая среда CLR сама вызовет сборщик мусора и очистит память.

Нередко приложение, созданное на C#, называют управляемым кодом (managed code). Что это значит? А это значит, что данное приложение создано на основе платформы .NET и поэтому управляется общеязыковой средой CLR, которая загружает приложение и при необходимости очищает память. Но есть также приложения, например, созданные на языке С++, которые компилируются не в общий язык CIL, как C# или VB.NET, а в обычный машинный код. В этом случае .NET не управляет приложением.

В то же время платформа .NET предоставляет возможности для взаимодействия с неуправляемым кодом. Наряду со стандартными классами библиотеки .NET можно также использовать сборки COM.

Код на C# компилируется в приложения или сборки с расширениями exe или dll на языке CIL. Далее при запуске на выполнение подобного приложения происходит JIT-компиляция (Just-In-Time) в машинный код, который затем выполняется. При этом, поскольку приложение может быть большим и содержать кучу инструкций, в текущий момент времени будет компилироваться лишь та часть приложения, к которой непосредственно идет обращение. Если происходит обращение к другой части кода, то она будет скомпилирована из CIL в машинный код. При том уже скомпилированная часть приложения сохраняется до завершения работы программы. В итоге это повышает производительность.

Платформа ASP.NET MVC представляет собой фреймворк для создания сайтов и веб-приложений с помощью реализации паттерна MVC.

Концепция паттерна (шаблона) MVC (model - view - controller) предполагает разделение приложения на три компонента:

Контроллер (controller) представляет класс, обеспечивающий связь между пользователем и системой, представлением и хранилищем данных. Он получает вводимые пользователем данные и обрабатывает их. И в зависимости от результатов обработки отправляет пользователю определенный вывод, например, в виде представления.

Представление (view) - это собственно визуальная часть или пользовательский интерфейс приложения. Как правило, html-страница, которую пользователь видит, зайдя на сайт.

Модель (model) представляет класс, описывающий логику используемых данных.

Общую схему взаимодействия этих компонентов можно представить следующим образом:

В этой схеме модель является независимым компонентом - любые изменения контроллера или представления не затрагивают модель. Контроллер и представление являются относительно независимыми компонентами, и нередко их можно изменять независимо друг от друга.

Благодаря этому реализуется концепция разделение ответственности, в связи с чем легче построить работу над отдельными компонентами. Кроме того, вследствие этого приложение обладает лучшей тестируемостью. И если нам, допустим, важна визуальная часть или фронтэнд, то мы можем тестировать представление независимо от контроллера. Либо мы можем сосредоточиться на бэкэнде и тестировать контроллер.

Конкретные реализации и определения данного паттерна могут отличаться, но в силу своей гибкости и простоты он стал очень популярным в последнее время, особенно в сфере веб-разработки.

Свою реализацию паттерна представляет платформа ASP.NET MVC. 2013 год ознаменовался выходом новой версии ASP.NET MVC - MVC 5, а также релизом Visual Studio 2013, которая предоставляет инструментарий для работы с MVC5.

Хотя во многих аспектах MVC 5 не слишком сильно будет отличаться от MVC 4, многое из одной версии вполне применимо к другой, но в то же время есть и существенные отличия:

* В MVC 5 изменилась концепция аутентификации и авторизации. Вместо SimpleMembershipProvider была внедрена система ASP.NET Identity, которая использует компоненты OWIN и Katana.
* Для создания адаптивного и расширяемого интерфейса в MVC 5 используется css-фреймворк Bootstrap
* Добавлены фильтры аутентификации, а также появилась функциональность переопределения фильтров
* В MVC 5 также добавлены атрибуты маршрутизации

Это наиболее важные нововведения в MVC 5. Кроме того, есть еще ряд менее значимых, например, использование по умолчанию Entity Framework 6, некоторые изменения при создании проекта (концепция One ASP.NET), дополнительные компоненты и т.д.

* В качестве базы данных используется MSSQL Server. SQL Server является одной из наиболее популярных систем управления базами данных (СУБД) в мире. Данная СУБД подходит для самых различных проектов: от небольших приложений до больших высоконагруженных проектов.
* SQL Server был создан компанией Microsoft. Первая версия вышла в 1987 году. А текущей версией является версия 16, которая вышла в 2016 году и которая будет использоваться в текущем руководстве.
* SQL Server долгое время был исключительно системой управления базами данных для Windows, однако начиная с версии 16 эта система доступна и на Linux.
* SQL Server характеризуется такими особенностями как:
* Производительность. SQL Server работает очень быстро.
* Надежность и безопасность. SQL Server предоставляет шифрование данных.
* Простота. С данной СУБД относительно легко работать и вести администрирование.
* Центральным аспектом в MS SQL Server, как и в любой СУБД, является база данных. База данных представляет хранилище данных, организованных определенным способом. Нередко физически база данных представляет файл на жестком диске, хотя такое соответствие необязательно. Для хранения и администрирования баз данных применяются системы управления базами данных (database management system) или СУБД (DBMS). И как раз MS SQL Server является одной из такой СУБД.
* Для организации баз данных MS SQL Server использует реляционную модель. Эта модель баз данных была разработана еще в 1970 году Эдгаром Коддом. А на сегодняшний день она фактически является стандартом для организации баз данных.
* Реляционная модель предполагает хранение данных в виде таблиц, каждая из которых состоит из строк и столбцов. Каждая строка хранит отдельный объект, а в столбцах размещаются атрибуты этого объекта.
* Для идентификации каждой строки в рамках таблицы применяется первичный ключ (primary key). В качестве первичного ключа может выступать один или несколько столбцов. Используя первичный ключ, мы можем ссылаться на определенную строку в таблице. Соответственно две строки не могут иметь один и тот же первичный ключ.
* Через ключи одна таблица может быть связана с другой, то есть между двумя таблицами могут быть организованы связи. А сама таблица может быть представлена в виде отношения ("relation").
* Для взаимодействия с базой данных применяется язык SQL (Structured Query Language). Клиент (например, внешняя программа) отправляет запрос на языке SQL посредством специального API. СУБД должным образом интерпретирует и выполняет запрос, а затем посылает клиенту результат выполнения.
* Изначально язык SQL был разработан в компании IBM для системы баз данных, которая называлась System/R. При этом сам язык назывался SEQUEL (Structured English Query Language). Хотя в итоге ни база данных, ни сам язык не были впоследствии официально опубликованы, по традиции сам термин SQL нередко произносят как "сиквел".
* В 1979 году компания Relational Software Inc. разработала первую систему управления баз данных, которая называлась Oracle и которая использовала язык SQL. В связи с успехом данного продукта компания была переименована в Oracle.
* Впоследствии стали появляться другие системы баз данных, которые использовали SQL. В итоге в 1989 году Американский Национальный Институт Стандартов (ANSI) кодифицировал язык и опубликовал его первый стандарт. После этого стандарт периодически обновлялся и дополнялся. Последнее его обновление состоялось в 2011 году. Но несмотря на наличие стандарта нередко производители СУБД используют свои собственные реализации языка SQL, которые немного отличаются друг от друга.
* Выделяются две разновидности языка SQL: PL-SQL и T-SQL. PL-SQL используется в таких СУБД как Oracle и MySQL. T-SQL (Transact-SQL) применяется в SQL Server. Собственно поэтому в рамках текущего руководства будет рассматриваться именно T-SQL.
* В зависимости от задачи, которую выполняет команда T-SQL, он может принадлежать либо к типу DDL, либо к DML, либо к DCL.

DDL (Data Definition Language / Язык определения данных). К этому типу относятся различные команды, которые создают базу данных, таблицы, индексы, хранимые процедуры и т.д. В общем определяют данные.

* В частности, к этому типу мы можем отнести следующие команды:
* CREATE: создает объекты базы данных (саму базу даных, таблицы, индексы и т.д.)
* ALTER: изменяет объекты базы данных
* DROP: удаляет объекты базы данных
* TRUNCATE: удаляет все данные из таблиц

DML (Data Manipulation Language / Язык манипуляции данными). К этому типу относят команды на выбору данных, их обновление, добавление, удаление - в общем все те команды, с помощью которыми мы можем управлять данными.

* К этому типу относятся следующие команды:
* SELECT: извлекает данные из БД
* UPDATE: обновляет данные
* INSERT: добавляет новые данные
* DELETE: удаляет данные

DCL (Data Control Language / Язык управления доступа к данным). К этому типу относят команды, которые управляют правами по доступу к данным. В частности, это следующие команды:

* GRANT: предоставляет права для доступа к данным
* REVOKE: отзывает права на доступ к данным

Entity Framework представляет специальную объектно-ориентированную технологию на базе фреймворка .NET для работы с данными. Если традиционные средства ADO.NET позволяют создавать подключения, команды и прочие объекты для взаимодействия с базами данных, то Entity Framework представляет собой более высокий уровень абстракции, который позволяет абстрагироваться от самой базы данных и работать с данными независимо от типа хранилища. Если на физическом уровне мы оперируем таблицами, индексами, первичными и внешними ключами, но на концептуальном уровне, который нам предлагает Entity Framework, мы уже работает с объектами.

Первая версия Entity Framework - 1.0 вышла еще в 2008 году и представляла очень ограниченную функциональность, базовую поддержку ORM (object-relational mapping - отображения данных на реальные объекты) и один единственный подход к взаимодействию с бд - Database First. С выходом версии 4.0 в 2010 году многое изменилось - с этого времени Entity Framework стал рекомендуемой технологией для доступа к данным, а в сам фреймворк были введены новые возможности взаимодействия с бд - подходы Model First и Code First.

Дополнительные улучшения функционала последовали с выходом версии 5.0 в 2012 году. И наконец, в 2013 году был выпущен Entity Framework 6.0, обладающий возможностью асинхронного доступа к данным.

Центральной концепцией Entity Framework является понятие сущности или entity. Сущность представляет набор данных, ассоциированных с определенным объектом. Поэтому данная технология предполагает работу не с таблицами, а с объектами и их наборами.

Любая сущность, как и любой объект из реального мира, обладает рядом свойств. Например, если сущность описывает человека, то мы можем выделить такие свойства, как имя, фамилия, рост, возраст, вес. Свойства необязательно представляют простые данные типа int, но и могут представлять более комплексные структуры данных. И у каждой сущности может быть одно или несколько свойств, которые будут отличать эту сущность от других и будут уникально определять эту сущность. Подобные свойства называют ключами.

При этом сущности могут быть связаны ассоциативной связью один-ко-многим, один-ко-одному и многие-ко-многим, подобно тому, как в реальной базе данных происходит связь через внешние ключи.

Отличительной чертой Entity Framework является использование запросов LINQ для выборки данных из БД. С помощью LINQ мы можем не только извлекать определенные строки, хранящие объекты, из бд, но и получать объекты, связанные различными ассоциативными связями.

Другим ключевым понятием является Entity Data Model. Эта модель сопоставляет классы сущностей с реальными таблицами в БД.

Entity Data Model состоит из трех уровней: концептуального, уровень хранилища и уровень сопоставления (маппинга).

На концептуальном уровне происходит определение классов сущностей, используемых в приложении.

Уровень хранилища определяет таблицы, столбцы, отношения между таблицами и типы данных, с которыми сопоставляется используемая база данных.

Уровень сопоставления (маппинга) служит посредником между предыдущими двумя, определяя сопоставление между свойствами класса сущности и столбцами таблиц.

Таким образом, мы можем через классы, определенные в приложении, взаимодействовать с таблицами из базы данных.

Entity Framework предполагает три возможных способа взаимодействия с базой данных:

Database first: Entity Framework создает набор классов, которые отражают модель конкретной базы данных

Model first: сначала разработчик создает модель базы данных, по которой затем Entity Framework создает реальную базу данных на сервере.

Code first: разработчик создает класс модели данных, которые будут храниться в бд, а затем Entity Framework по этой модели генерирует базу данных и ее таблицы

Скриншоты админки  
