ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
МОРДОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМ. Н. П. ОГАРЁВА»**

Институт электроники и светотехники

Кафедра информационной безопасности и сервиса

УТВЕРЖДАЮ

Зав. кафедрой
канд. техн. наук, доц.

_____ С. Н. Ивлиев
(подпись)
«11» 06 2019 г.

БАКАЛАВРСКАЯ РАБОТА

РАЗРАБОТКА БОРТОВОГО КОМПЬЮТЕРА ДЛЯ АВТОМОБИЛЯ НА ОСНОВЕ ARDUINO-NANO

Автор бакалаврской работы (подпись) 11.06.2019 (дата) А.Д. Нуштайкина

Обозначение бакалаврской работы БР–02069964–43.03.01–07–19

Направление 43.03.01 Сервис

Руководитель работы

ст. преп. (подпись) 11.06.2019 (дата) А. В. Волков

Нормоконтролер

канд. физ.-мат. наук, доц. (подпись) 11.06.2019 (дата) Д. А. Салкин

Саранск
2019

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
МОРДОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМ. Н. П. ОГАРЁВА»**

Институт электроники и светотехники

Кафедра информационной безопасности и сервиса

УТВЕРЖДАЮ

Зав. кафедрой
канд. техн. наук, доц.
_____ С.Н. Ивлиев
(подпись)
«16» 11 2018 г.

ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

(в форме бакалаврской работы)

Студент Нуштайкина Анна Дмитриевна

1 Тема: «Разработка бортового компьютера для автомобиля на основе Arduino-Nano»

Утверждена приказом № 9687-с от 16.11.2018

2 Срок представления работы к защите 11.06.2019

3 Исходные данные для выпускной квалификационной работы: техническая литература, нормативная документация, техническая документация, научные статьи по теме микроконтроллера Arduino Nano.

4 Содержание выпускной квалификационной работы:

4.1 Обзор бортовых компьютеров на основе контроллера ELM 327

4.2 Протокол обмена ELM 327 по Bluetooth с микроконтроллером на базе Arduino Nano

4.3 Разработка устройства на основе AtMega328 и модуле HC -05

5 Приложения:

5.1 Схема принципиальная электрическая

5.2 Листинг управляющей программы микроконтроллера

Руководитель работы
ст. преп. _____ 16.11.2018 А.В. Волков
подпись, дата

Задание принял к исполнению _____ 16.11.2018
подпись, дата

РЕФЕРАТ

Бакалаврская работа содержит 92 страниц, 60 рисунков, 2 таблицы, 36 использованных источников, 2 приложения.

БОРТОВОЙ КОМПЬЮТЕР, ДИАГНОСТИКА, МИКРОКОНТРОЛЛЕР, ПЛАТА, СРЕДСТВА ДИАГНОСТИКИ, ПРОТОКОЛ, ШИНА, АВТОСКАНЕР, ИНТЕРФЕЙС, АДПТЕР, МОДУЛЬ.

Объектом разработки является бортовой компьютер для автомобиля на основе Arduino Nano.

Цель работы — исследовать принципы организации и способы передачи и приема информации по шине I2C; получение навыков программирования контроллеров ATMega328 и ELM327; разработка бортового компьютера на основе полученных навыков с использованием контроллера.

В процессе работы проводились исследования разработки бортового компьютера на основе полученных навыков с использованием контроллера.

В результате проведенной работы был разработан бортовой компьютер для автомобиля на основе Arduino Nano.

Степень внедрения — результаты работы частично внедрены на кафедре для обучения студентов.

Область применения — для диагностики автомобилей и обучения студентов программированию микроконтроллеров.

Эффективность — разработка бортового компьютера для автомобиля на основе Arduino Nano значительно увеличит возможности для диагностики автомобиля.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Сегодня даже бюджетные модели автомобилей оснащены датчиками, контролирующими работу его основных узлов и агрегатов - например, системы управления впрыском топлива, зажигания, подачи топлива, обеспечения микроклимата и многих других. А водитель не имеет доступа не только к регулировке этих параметров, но и даже, зачастую, и к их значениям - которые (в купе с кодами неисправностей) записываются в электронный блок управления автомобилем, доступ к которому возможен лишь с помощью специального оборудования в авторизированном центре обслуживания. Однако, в настоящее время рынке представлено большое количество устройств предназначенных автовладельцев, т.е. бортовых компьютеров, оснащенных системами диагностики автомобиля.

Бортовой компьютер представляет собой электронное устройство, регистрирующее происходящие процессы в автомобиле и одновременно помогая водителю контролировать его работу, бортовой компьютер показывает водителю на мультимедийном дисплее текущее состояние автомобиля: скорость движения, расход топлива, давление масла, температуру охлаждающей жидкости и многое другое. Так же следит за правильной эксплуатацией, подскажет о замене свечей зажигания, масла, различных фильтров. В его базе данных даже могут находиться адреса сервисных центров, которые проведут ремонт неисправностей.Бортовой компьютер подключается в штатное гнездо приборной панели и нисколько не нарушает внешний вид салона. При техническом обслуживании помогает определить ошибки работы системы.Работают такие автосканеры следующим образом: бортовой компьютер подключается к блоку управления электропитанием, который считывает информацию с датчиков автомобиля. Разработка собственного бортового компьютера имеет массу преимуществ, например, можно

самостоятельно задать те параметры, которые выводит компьютер, то есть те, которые будут необходимы нам для диагностики.

1 Обзор бортовых компьютеров на основе контроллера ELM 327

1.1 Контроллер ELM 327, устройство и принцип работы

ELM327 представляет собой один из простейших адаптеров с микроконтроллером на борту (рисунок 1.1). ELM327 является программируемым микроконтроллером производстваELM Electronicsдля передачи данных On-Board Diagnostics (OBD) интерфейса, который можно найти в большинстве современных автомобилей.

Оригинальный ELM327 реализуется на микроконтроллере PIC18F2480 от Microchip Technology. ELM327 является одним из семейства OBD преобразователей от ELM Electronics.

Стандартный размер автосканера примерно 5×3 сантиметра, но бывают и более уменьшенные версии.

Рисунок 1.1 — Внешний вид сканера

Подключение может быть:

- Проводным, через COM- или USB-порт;
- Беспроводным, через Bluetooth, либо через WIFI[18].

На рынке представлено 4 интерфейса передачи данных ELM 327:

- **RS232 (или RS серия)**— этот тип интерфейса постепенно исчезает с ПК как и порт COM. Он становится самым дешевым протоколом связи для адаптеров. Но стоит отметить, что некоторый диагностический узкоспециализированый софт поддерживает только COM порты при сопряжении (рисунок 1.2).

Рисунок 1.2 — ELM 327 RS232

- **USB**— дороже, чем RS, но требует установки USB драйвера. Преимущество заключаются в том, что все ПК оснащены USB портом и для диагностики не потребуется современного смартфона (рисунок 1.3).

Рисунок 1.3 — ELM 327USB

- **Bluetooth**— беспроводной стандарт связи «Bluetooth» стоит на многих компьютерах или смартфонах. Настраивает связь ЭБУ автомобиля со смартфоном планшетом или ноутбуком, что делает его универсальным (рисунок 1.4).

Рисунок 1.4 — ELM 327 Bluetooth

- **Wi-Fi**— благодаря беспроводному соединению сканера, его используют с компьютером или смартфоном, расположенным на удалении от самого сканера даже до 50-100 метров. При выборе автосканера стоит учитывать то, что с смартфонами iPhone или iPad связывают только по WiFi (рисунок 1.5)[28].

Рисунок 1.5 — ELM 327 Wi-Fi

Какой выбрать вид подключения не имеет значения, работать он будет одинаково. Однако, автосканеры, работающие посредством Bluetooth обладают

наибольшей популярностью, так как по своему устройству они значительно надежнее, а работа с ними наиболее проста и удобна. Для считывания информации необходимо установить специальное программное обеспечение на компьютер, планшет или телефон.

Микроконтроллер ELM327 содержит внутреннюю программу, которая конвертирует сигналы OBDII (OBD2) стандарта (любого протокола — VPW, PWM, ISO, KWP 2000, а также CAN, протокол определяется автоматически) в стандартные ASCII-коды, позволяя легко передавать коды ошибок и разнообразные рабочие данные без необходимости в дополнительном программном преобразовании[18].

Задачи, которые данное устройство сможет помочь выполнить:

- Мониторинг показателей датчиков установленных на автомобиле;
- Выявление вышедших из строя датчиков и проверка правильности показателей датчиков;
- Мониторинг и считывание показателей кодов ошибок (разъяснение значения каждого кода);
- Возможность самостоятельного сброса ошибок в режиме реального времени;
- Экспорт данных и распечатка.

Версии прошивок варьируются в зависимости от производителя автосканера. В официальной продаже находятся версии сканера 1.3a, 1.4b, 2.2.

В Интернете можно найти ревизию 1.5, которая стала копией v1.2. Они не имеют одинаковый уровень качества, а отображаемая версия программного обеспечения часто является поддельной копией официальных прошивок. В свободной продаже имеются ревизии, не указанные оригинальным производителем.

Список оригинальных и пиратских прошивок:

- Оригинальные: 1.0 1.1 1.2 1.3 1.3a 1.4, 1.4b, 2.2;
- Пиратские: 1.2a, 1.4a, 1.5, 1.5a, 2.1 (и модификации 2.1).

ElmElectronics производит протокольную интегральную схему ELM327 с 2005 года. Компания несколько раз обновляла продукт в соответствии с запросами клиентов, в результате было выпущено много разных прошивок IC. Все версии поддерживают стандартные протоколы OBDII. Цены на устройства с оригинальными чипами ELM327 начинаются от $62.00 [33].

Оригинальный ELM327 доступен в следующих вариантах:

а) v2.2 — самая последняя схема, заключающая в себе улучшенный функционал прошлых моделей. Новая модель стоит $21.00;

б) ELM 327L v2.2 имеет аналогичный функционал, но поддерживает широкий диапазон напряжения от 2,0 В до 5,5 В;

в) v1.3a — это стандартный протокол, имеющий самую большую популярность на рынке;

г) v1.4b отличался поддержкой функций малой мощности, но теперь его не производят.

Таблица 1.1 — Отличия оригинальных прошивок

Параметр	**v1.3a**	**v2.2**	**ELM327L v2.2**
Рабочее напряжение	4.5V to 5.5V	4.2V to 5.5V	2.0V to 5.5V
Режим низкой мощности (сна)	–	✓	✓
Настройки сохраняются при пробуждении	–	✓	✓
RS232 байтов передачи	256	512	2048
AT команды	93	128	128
Проверка частоты CAN во время автоматического поиска протокола	–	✓	✓

Поддержка в ожидании ответа (7F xx 78)	_	✓	✓

Версии 1.5 и 2.1 — это урезанные китайские копии, они популярны на российском рынке. Последняя прошивка - v1.5 является аналогом официальной v1.4b.

Потребители предпочитают покупать диагностический сканер, помеченные как «v.1.4» или «v.1.5», а не «v.2.1», так как эта ревизия содержит много программных «багов»[18].

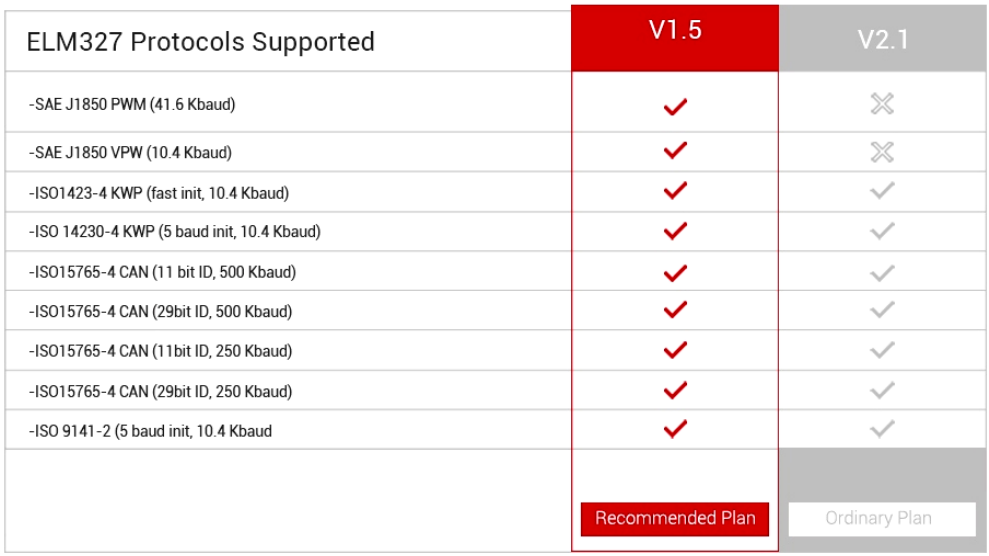


Рисунок 1.6 — Поддерживаемы протоколы v1.5 и v2.1

В Интернете есть много программ, определяющих оригинальность прошивки и её ревизию. Самыми популярными приложениями являются:

- FORScan - это самая популярная программа на рынке, подходящая для большинства иностранных марок. Приложение совместимо с ELM 327 OBD2-RS232-USB.

- ELM327Identifier — идентифицирует номер прошивки. Это программа для проверки автосканера через Bluetooth, Wi-Fi или USB. Программа доступно лишь на мобильных платформах, в чем заключается ее основной минус.

Так же определить подделку можно по MAC-адресу, проверяющийся при подключении, обнаруживается имя или MAC-адрес. Если MAC-адрес автосканера начинается с чисел 66:35:56, то этот адаптер не сможет подключиться к автомобилю. Он также поддерживает не все протоколы и не сможет прочитать информацию.

Работа автосканера ELM327 заключается в связи адаптера с ЭБУ посредством специального протокола OBD-II (On Board Diagnostic). Он необходим, чтобы правильно подключиться к блоку управления двигателем [33].

1.2 Стандарт протокола OBD 2 и отличие его от OBD1

Работа автосканера ELM327 заключается в связи адаптера с ЭБУ посредством специального протокола OBD-II (On Board Diagnostic). Он необходим, чтобы правильно подключиться к блоку управления двигателем.

OBD2— это объединенный стандарт, регламентирующий параметры диагностики автомобиля, обеспечивающий доступ к его системам. Он определяет вид и расположение пинов диагностического разъема, протоколы ошибок и обмена информации, стандарты системы команд.

При этом, необходимо обязательно уточнить совместимость ЭБУ своего автомобиля с данным протоколом. Этот стандарт поддерживается всеми американскими автомобилями, начиная с 1996 года, а европейскими, начиная с 2001 года для бензиновых моторов, и с 2004 для дизельных двигателей[34].

Есть готовый список совместимых с протоколом OBD2 автомобилей.

Таблица 1.2 — Совместимость протоколов с марками а/м

ПРОТОКОЛ ISO 15765-4:	Audi, Saab, Opel, VW, Ford, Jaguar, Renault, Peugeot, Chrysler, Porsche, Volvo, Mazda,

ПРОТОКОЛ ISO 14230-4:	Daewoo, Hyundai, KIA ,Nissan, Toyota,.
ПРОТОКОЛ ISO 9141-2	Honda, Infinity,Lexus, Audi, BMW, Mersedes.
ПРОТОКОЛ J1850 VPW:	Buick, Cadillac, Chevrolet, Chrysler, Dodge, GM, Isuzu.
ПРОТОКОЛ J1850 PWM:	Ford, Lincoln, Mazda.

Чтобы обеспечить правильную работу автосканера для диагностики, необходимо иметь устройство с установленным и настроенным ПО (смартфон, планшет, ноутбук или даже настольный компьютер). Можно пользоваться ПОидущим в комплекте, либо найти подходящую программу в интернете.

Основная функция диагностического разъема (в OBD II он называется диагностическим разъемом связи — Diagnostic Link Connector, DLC) заключается в том, чтобы обеспечить связь диагностического сканера с блоками управления, совместимыми с OBD II. [27].

Он должен находиться в пределах 16 дюймов от рулевого колеса. Производитель может разместить DLC в одном из восьми мест, определённых EPA.

Каждый контакт разъема имеет свое назначение. Функции многих контактов отданы на усмотрение производителям, однако эти контакты не должны использоваться блоками управления, совместимыми с OBD II. Примерами систем, применяющих такие разъемы, являются SRS (дополнительная ограничительная система) и ABS (антиблокировочная система колес).

В рамках диагностического стандарта OBDII существует 5 основных протоколов обмена данными между электронным блоком управления (ЭБУ) и диагностическим сканером. Физически подключение автосканера к ЭБУ производится через разъем DLC (Diagnostic Link Connector), который соответствует стандарту SAE J1962 и имеет 16 контактов (2x8). Ниже представлена схема расположения контактов в разъеме DLC (рисунок 7), а также назначение каждого из них.

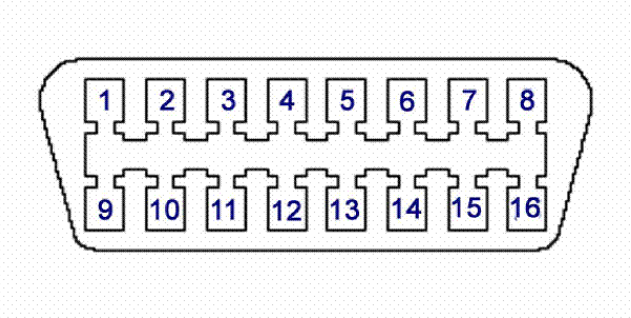


Рисунок 1.7 — Расположение контактов в разъеме DLC (Diagnostic Link Connector)

1—OEM (протокол производителя). Коммутация +12в. при включении зажигания;

2— Шина + (Bus positive Line). SAE-J1850 PWM, SAE-1850 VPW;

3—–;

4—Заземление кузова;

5 —Сигнальное заземление;

6— Линия CAN-High высокоскоростнойшины CAN Highspeed (ISO 15765-4, SAE-J2284);

7— K-Line(ISO 9141-2 и ISO 14230);

8 —–;

9 — Линия CAN-Low, низкоскоростной шины CAN Lowspeed;

10—Шина— (Bus negative Line). SAE-J1850 PWM, SAE −1850 VPW;

11 —.–;

12—–;

13—–;

14—Линия CAN-Low высокоскоростнойшины CAN Highspeed (ISO 15765-4, SAE-J2284);

15— L-Line (ISO 9141-2 и ISO 14230);

16 — Питание +12в от АКБ.

Назначение неопределенных контактов выбирается на усмотрение производителя автомобиля[18].

Иногда разъем OBD-II устанавливается на автомобили, которые в принципе не поддерживают ни один из OBD-II-протоколов. В таких случаях необходимо использовать специальный сканер, рассчитанный на работу с заводскими протоколами конкретной марки автомобиля - например, это касается некоторых автомобилей европейского рынка 1996-1997 гг.

Далее подробно рассмотрен формат и физический уровень каждого протокола связи в рамках стандарта OBDII:

а) **SAE J1850 PWM** —существует два типа протокола J1850. PWM является высокоскоростным и обеспечивает передачу информации со скоростью 41,6 Кбайт/с. Он применяется в автомобилях марок Ford, Jaguar и Mazda. В протоколе PWM сигналы передаются по двум проводам, подсоединенным к 2 и 10 контакту диагностического разъема.

Формат сигнала протокола J1850 включает:

- SOF —Start of Frame (начало кадра, высокий импульс на 200uS);
- Header —заголовок длиной 1 байт;
- EOD —End Of Data (окончание данных,низкий импульс на 200uS) [28].

Если рассматривать формат протокола более подробно по битам, то он примет следующий вид:

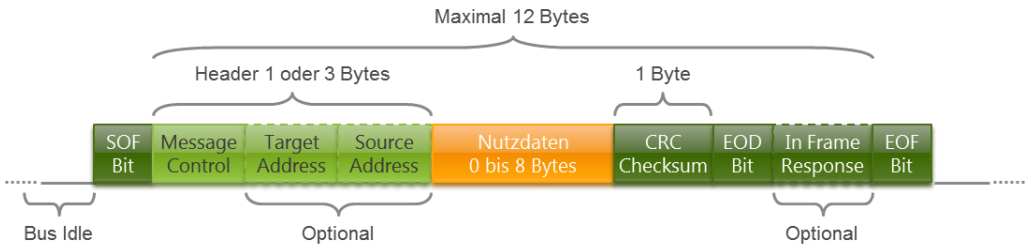


Рисунок 1.8 —Формат протокола по битам

Реальный пример сигнала SAE J1850 выгладит следующим образом:

Рисунок 1.9 —Сигнал SAE J1850

б) **SAE J1850 VPW** —протокол VPW осуществляет передачу данных со скоростью 10,4 Кбайт/с, что существенно медленнее, чем у протокола PWM. Формат данного протокола идентичен SAE J1850 PWM . Данный протокол используется на автомобилях General Motors (GM) и Chrysler. VPW предусматривает обмен данными по одному проводу, подсоединенному ко 2 контакту диагностического разъема. Длина шины может достигать 35 метров.

в) **ISO 9141-2** —данный протокол разработан компанией ISO. Он не такой сложный, как протоколы J1850 и не требует в использовании специальных коммуникационных микропроцессоров, но, с другой стороны, обеспечивает довольно медленную передачу данных со скоростью 10 Кбайт/с. Протоколы ISO 9141 и ISO 14230 схожи по физической реализации обмена информацией, но различаются ее использованием. Поэтому сканер ISO 9141, обычно может работать и с ISO 14230, но не наоборот.

В протоколе ISO 9141-2 сигналы передаются по 7 контакту (К-линия) и опционально по 15 контакту (L-линия). К-линия является двунаправленной (т.е. передает данные в обе стороны), L-линия однонаправленная и используется лишь для соединения ЭБУ и сканера, после чего линия L переходит в состояние логической единицы.

Физический уровень передачи информации в протоколах ISO 9141 и ISO 14230 заключается в одновременной передачи ЭБУ специального 8-битного кода по К- и L-линиям со скоростью 5Б/сек. Если код правильный, то ЭБУ посылает сканеру 8-битный код со скоростью последующего соединения. Затем передается еще два кода с информацией о последующем соединении и

расположении K- и L-линий. Сканер возвращает отражение этих кодов в ЭБУ. На этом процесс распознавания окончен.

В общем виде процесс инициализации сигнала в протоколах ISO 9141 и ISO 14230 выглядит следующим образом:

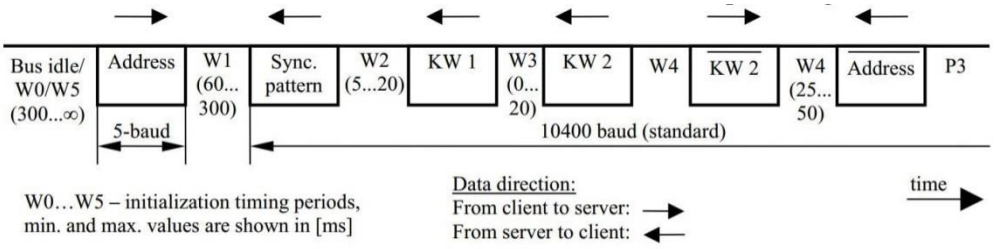


Рисунок 1.10 —Инициализация сигнала

Передача данных в протоколе осуществляется по следующей схеме:

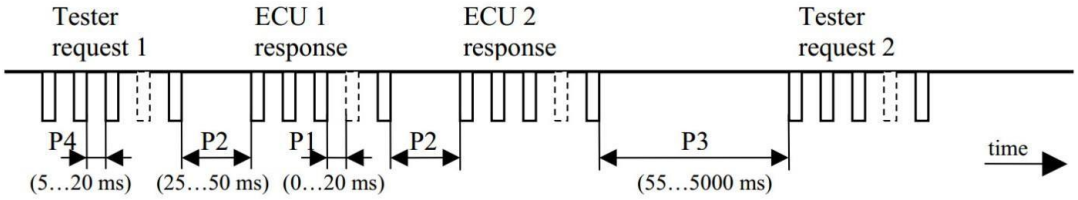


Рисунок 1.11 —Схема передачи данных

г) **ISO 14230-4 (другое название Keyword Protocol 2000)**—на физическом уровне данный протокол идентичен ISO 9141, но является еще более медленным (скорость передачи данных от 1,2 до 10 Кбайт/с в быстрой версии).

д) **ISO 15765 CAN**—CAN-протокол был разработан компанией Bosch для автомобильного и промышленного применения. В рамках стандарта OBD2 протокол использует линии CAN High и CAN Low, т.е. 2 контакта для обмена сигналом: 6 и 14. Он является самым скоростным и совершенным. Сейчас данный протокол используется на большинстве современных автомобилях. Стандарт CAN не регламентирует определенной скорости работы для каждой

шины в автомобиле. С помощью отдельных и встроенных микроконтроллеров есть возможность менять ее от 20 Кбит/с до 1 Мбит/с[18].

Так же необходимо разобрать сходства и различия протоколов OBD1 и OBD2.

Принцип действия обеих систем сходен: блоком управления считываются показания датчиков на разных режимах работы двигателя в процессе эксплуатации автомобиля (запуск, прогрев, холостой ход, разгон и торможение и т.д.). Показания датчиков бывают статическими (дискретными) или динамическими (изменяющимися во времени).

Статические показания датчиков обычно определяются неким пороговым значением - импульсом определенного уровня или "переключателем" (то есть, наличием или отсутствием сигнала), а динамические, как правило, передают изменения параметра и проверяются на допустимые пределы (верхний и/или нижний).В OBD-I предусмотрено определение неисправностей двигателя, подушек безопасности, тормозной системы АБС и АКПП. В OBD-II перечень диагностируемых устройств расширен (к перечисленному добавлена еще и климатическая установка, иммобилайзер и некоторое дополнительное оборудование), а количество диагностических кодов увеличено[27].

Система OBDI позволяет считать коды ошибок "вручную", по миганию лампочкиCheck Engine.

Система OBDII также позволяет считывать коды ошибок с помощью мигания лампочки Check Engine, однако, это происходит в сочетании с иными символами панели приборов, что более информативно и точно отображает коды ошибок ЭБУ автомобиля. Кроме этого имеется возможность подключения диагностического прибора через специальный разъем (в качестве такого прибора может использоватьсяадаптер ELM327к персональному компьютеру, планшету или смартфону и соответствующая программа).

Когда система управления двигателем обнаруживает проблему, например, с составом выхлопных газов, на приборном щитке загорается надпись Check

Engine («Проверьте двигатель»). Этот индикатор называется лампочкой индикации неисправностей (Malfunction Indication Light — MIL). Индикатор обычно выдает следующие надписи: Service Engine Soon («Отрегулируйте двигатель в ближайшее время»), Check Engine («Проверьте двигатель») и Check («Выполните проверку»). Назначение индикатора состоит в информировании водителя о том, что в процессе работы системы управления двигателем возникла проблема.

Индикатор неисправностей MIL загорается при возникновении проблемы в системе управления двигателем, например при неисправности искрового промежутка или загрязнении абсорбера. В принципе, это может быть любая неисправность, приводящая к повышенному выбросу вредных примесей в атмосферу.

Для того чтобы проверить функционирование индикатора OBD II MIL, следует включить зажигание (когда на приборном щитке загораются все индикаторы). При этом загорается и индикатор MIL. Спецификация OBD II требует, чтобы этот индикатор горел некоторое время. При запуске двигателя и отсутствии в нем неисправностей лампочка «Check Engine»должна погаснуть.

Рисунок 1.12—Код ошибки

Лампочка «Check Engine» не обязательно загорается при первом появлении неисправности. Срабатывание этого индикатора зависит от того, насколько серьезна неисправность. Если она считается серьезной лампочка загорается немедленно. Такая неисправность относится к разряду активных (Active). В случае если устранение неисправности может быть отложено, индикатор не горит и неисправности присваивается сохраняемый статус

(Stored). Для того чтобы такая неисправность стала активной, она должна проявиться в течение нескольких драйв-циклов. Обычно драйв-циклом считается процесс, при котором холодный двигатель запускается и работает до достижения нормальной рабочей температуры (при этом температура охлаждающей жидкости должна быть 122 градуса по Фаренгейту) [18].

Каждый символ имеет свое значение.

Альфа-указатель DTC—первый символпринято называть альфа-указателем DTC. Этот символ указывает, в какой части автомобиля обнаружена неисправность. Выбор символа (P, B, C или U) определяется диагностируемым блоком управления. Когда получен ответ от двух блоков, используется буква для блока с более высоким приоритетом.

В первой позиции могут находиться лишь четыре буквы**:**

а) P (двигатель и трансмиссия);

б) B (кузов);

в) C (шасси);

г) U (сетевые коммуникации).

В OBD II неисправность описывается с помощью диагностических кодов неисправностей (Diagnostic Trouble Code — DTC). Коды DTC в соответствии со спецификацией J2012 представляют собой комбинацию одной буквы и четырех цифр. На рисунке1.12показано, что означает каждый символ.

Далее рассмотрим типы кодов.

Второй символ— показывает, что определил код.

- 0 (известный как код P0) —базовый, открытый код неисправности, определенный Ассоциацией автомобильных инженеров (SAE);

- 1 (или код P1) —код неисправности, определяемый производителем автомобиля. Большинство сканеров не могут распознавать описание или текст кодов P1. Однако такой сканер, как, например, Hellion, способен распознать большинство из них. Ассоциация SAE определила исходный перечень диагностических кодов ошибок DTC.

Однако производители стали говорить о том, что у них уже есть собственные системы, при этом ни одна система не похожа на другую. Например, система кодов для автомобилей Mercedes отличается от системы Honda, и они не могут использовать коды друг друга.

Поэтому ассоциация SAE пообещала разделить стандартные коды (P0) и коды производителей (P1).

Следующейидет система, в которой обнаружена неисправность.

Третий символ—обозначает систему, где обнаружена неисправность. Об этом символе информациименьше, но он относится к наиболее полезным. По этому символуможно сразу определить, какая система неисправна, даже не глядя на текст ошибки. Третий символ помогает быстро идентифицировать область, где возникла проблема, не зная точного описания кода ошибки.

И последнийпункт —индивидуальный код ошибки.

Четвертый и пятыйсимволы нужно рассматривать совместно. Они обычно соответствуют старым кодам ошибок OBDI.

Эти коды, как правило, состоят из двух цифр. В системе OBDII также берутся эти две цифры и вставляются в конец кода ошибки — так ошибки легче различать.

Все диагностические системы, в свою очередь, хранят и отображают статические данные —«коды ошибок» (OBDI понимает только их) или динамические характеристики (в OBDII и те, и другие). На дискретные показания датчиков система самодиагностики реагирует обычно только при отсутствии электрического контакта (возвращает сигнал о неисправности датчика), а изменение динамических показателей отслеживается по хранящимся в памяти устройства управления таблицам. Впрочем, один и тот же датчик может проверяться как на электрический контакт, так и на допустимые пределы изменения. И тогда для одного устройства могут быть две ошибки: либо отсутствие сигнала, либо выход за предельные параметры [23].

Необходимо отметить так же, что OBD1 использовался в первые годы автомобильной промышленности, а OBD2 был представлен только в автомобильных моделях, выпущенных в начале 1990-х годов и OBD1 подключен непосредственно к консоли автомобиля, а OBD2 подключен к автомобилю удаленно.

Одной из самых заметных отличительных особенностей является различие в интерфейсах передачи данных OBD1 и OBD2.

1.3 Can – шина (Can High/Can Low)

Чтобы связно и гармонично управлять системами, обеспечить качество и функциональность передачи данных, многие автомобилестроительные компании применяют современную систему, известную как CAN-шина.

Визуально CAN-шина выглядит как асинхронная последовательность. Ее информация передается по двум витым проводникам, радиоканалу или оптоволокну.Управлять шиной способны несколько устройств одновременно. Их количество не ограничено, а скорость обмена информацией запрограммирована до1 Мбит/с.CAN-шина в современных автомобилях регламентируется спецификацией «CAN Sorcjfication version 2,0». Он состоит из двух разделов. Протокол А описываетпередачу информациис применением 11-битной системы передачи данных.Часть В выполняет эти функции при применении 29-битного варианта.

CAN имеет узлы персональных тактовых генераторов. Каждый из них посылает сигналы всем системам одновременно. Получающие устройства, присоединенные к шине, определяют, относится ли сигнал к их компетенции. Каждая система обладает аппаратной фильтрацией адресованных ей посланий [35].

Рассмотрим разновидности и маркировки.

Одной из самых известных на сегодняшний день является разработанная Робертом Бошем CAN-шина. CAN BUS (под таким названием известна система) бывает последовательная, где импульс подается за импульсом. Она называется Serial bus. Если же информация передается по нескольким проводам, то это параллельная шина Parallel bus.

Опираясь на разновидности идентификаторов Can-шин, встречается маркировка двух типов.

В случае, когда узел поддерживает 11-битный формат обмена информацией и не обозначает ошибки на сигналы 29-битного идентификатора, его маркируют «CAN2.0A Active», «CAN2.0B Passive».

Когда таковые генераторы используют оба типа идентификаторов, шина имеет маркировку «CAN2.0B Active».

Встречаются узлы, поддерживающие коммуникации в 11-битном формате, а увидев в системе 29-битный идентификатор, выдают сообщение об ошибке. В современных автомобилях подобные CAN-шины не используются, ведь система должна быть логичной и согласованной.

Система же функционирует при двух типах скоростей передачи сигналов - 125, 250 кбит/с. Первые предназначены для вспомогательных устройств (стеклоподъемники, освещение), а вторые обеспечивают главное управление (коробка-автомат, двигатель, ABS).Физически проводник CAN-шины современного автомобиля выполнен из двух составляющих. *Первый* - черного цвета и называется CAN-High. *Второй* проводник, оранжево-коричневый, именуется CAN-Low. Благодаря представленной структуре коммуникаций из схемы автомобиля удалена масса проводников. При производстве транспортных средств это позволяетуменьшить вес изделия до 50 кг [36].

Общая сетевая нагрузка состоит из разрозненных сопротивлений блоков, которые входят всостав протокола, называемого Can-шина.

Различны и скорости передачи-получения каждой системы. Поэтому обеспечивается обработка разнотипных сообщений. Согласно описанию шины-

CAN, эту функцию выполняет преобразователь сигналов. Он называется межсетевым электронным интерфейсом.

Расположен этот прибор в конструкции управляющего блока, но бывает выполнен в виде обособленного прибора.

Представленный интерфейс применяют также для вывода и ввода сигналов диагностического характера. Для этого предусмотрено наличие унифицированной колодки OBD. Это особый разъем для диагностики системы.

Все управляющие блоки присоединены к CAN-шине трансиверами. Они имеют приемники сообщений, представляющих собой избирательные усилители.

Описание шины CAN оговаривает поступление посланий по проводникам High и Low вусилитель дифференциальный, где он обрабатывается и направляется в блок управления.

Усилитель определяет этот выходной сигнал как разность напряжений проводов High и Low. Такой подход позволяет исключить влияние внешних помех.

Чтобы понять, что собой представляет Can-шина и ее устройство, следует вспомнить ее облик (рисунок 1.13). Это два проводника, скрученные между собой [3].

Рисунок 1.13 —Вид шины

Так как сигнал помехи поступает сразу на оба провода, в процессе обработки значение напряжения Low отнимается от напряжения High.

В состоянии покоя напряжение на проводе CAN High и CAN Low

составляет 2,5 В. Такое состояние называется «рецессивное» и упрощенно соответствует значению бита «0» При переходе в активное «доминантное» состояние (такое состояние может создать любой элемент сети) напряжение на проводе CAN High будет повышаться не меньше чем на 1 В до 3,5 В, а CAN Low понижаться - тоже на 1 В до 1,5В.

Чтобы «понимать» разницу напряжений между CAN High и CAN Low, каждый блок управления подключается к шине CAN через трансивер, где происходит преобразование разности напряжений UCAN Hiи UCAN Lo в итоговое напряжение UDIFF. Разница междуCAN High и CAN Low будет 2В и будет восприниматься принимающими блоками управлениякак значение бита, равное «1».

Так происходит передача сигналов по шине CAN. Сами эти сигналы представляют собой "кадры" (сообщения), которые принимаются всеми элементами сети CAN. Полезная информация в кадре состоит из идентификационного поля (идентификатора) длиной 11 бит (стандартный формат) или 29 бит (расширенный формат, надмножество предыдущего) и поля данных длиной от 0 до 8 байт. Идентификационное поле говрит о содержимом пакета и служит для определения приоритета при попытке одновременной передачи несколькими сетевыми узлами. Также в кадре (сообщении) помимо полезной информации содержится служебная информация. Она представлена полями проверки, полем отзыва и другим полями. В конце кадра содержится "поле конец сообщения"

В шине CAN сообщения от блоков управления должны передаваться в общую шину , то для исключения конфликтов между блоками, каждый узел перед отправкой кадра проверяет сеть на передачу доминантного бита. Устройство передающее доминантный бит считается приоритетным. Таким образом устройство будет дожидаться освобождения линии CAN. С одной стороны такой алгоритм работы повышает быстродействие, но с другой при неправильной работе одного из блоков управления возможна полная «загрузка»

CAN шины и невозможность отправки сообщении другими блоками, элементами сети CAN (линия для них будет всегда занята) [2].

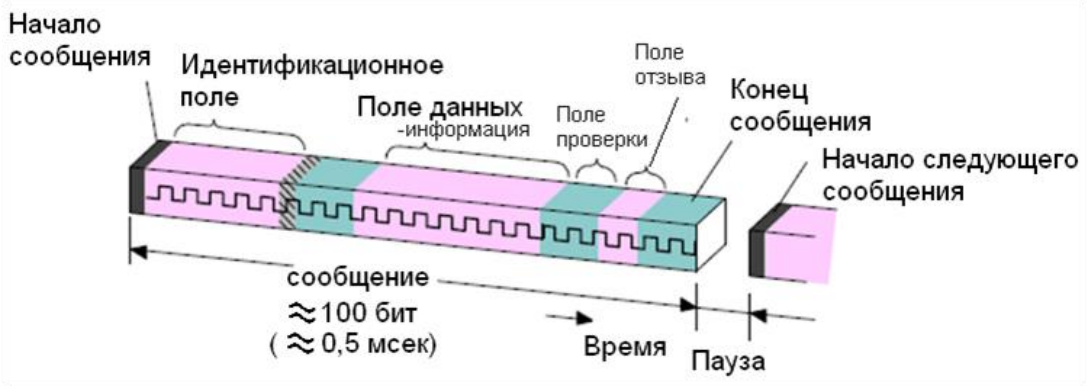


Рисунок 1.14 —Структура сообщения

Благодаря этому CAN-шина считается надежной системой.

Протоколом предусматривается использование при обмене информацией посредством шины CAN четырех типов команд.

Data Frame — такой тип сообщений (фреймов) передает сигналы с определенным идентификатором;

Error Frame —представляет собой сообщение сбоя в процессе обмена. Он предлагает повторить действия сначала;

Overload Frame — послание появляется в момент необходимости перезапустить работу контроллера;

Request Frame Remout Transmission —обозначает запрос данных, где именно находится идентификатор.

В процессе «приема-передачи» информации на проведение одной операции отводится определенное время. Если оно вышло, формируется фрейм ошибки. Error Frame также длится определенное количество времени. Неисправный блок автоматически отключается от шины при накоплении большого количества ошибок.

Устройство шины состоит, помимо кабеля, из нескольких элементов.

Микросхемы приемопередатчика часто встречаются от компании Philips,

а также Siliconix, Bosch, Infineon.

Максимальная длина проводника при скорости 1 Мбит/с достигает 40 м. Шина-CAN (известная еще как CAN-BUS) в конце наделена терминатором.

Для этого на конец проводников устанавливаются резисторы сопротивления по 120 Ом. Это необходимо, дабы устранить отражения сообщения на конце шины и убедиться, что она получает соответствующие уровни тока.

Сам проводник в зависимости от конструкции может быть экранированным или неэкранированным. Концевое сопротивление может отходить от классического и находиться в диапазоне от 108 до 132 Ом [1].

2 Протокол обмена ELM 327 по Bluetooth с микроконтроллером на базе ArduinoNano

2.1 Технические характеристики микроконтроллера ArduinoNano

Основа Arduino Nano —микроконтроллер на базе ATmega328, логическая микросхема (рисунок 2.1) для обработки данных с тактовой частотой 16 МГц, имеющая на борту 8 аналоговых и 14 цифровых контактов общего назначения, а также все необходимые интерфейсы: I2C, SPI и UART. Она является полным аналогом Arduino Uno, но с меньшим форм-фактором. Из-за малых размеров, плата часто используется в проектах где большое значение имеет компактность. На этой плате так же отсутствует вынесенное гнездо внешнего питания, Arduino Nano в данном случае работает через USB (mini- или microUSB). В остальном параметры совпадают с моделью Arduino Uno [6].

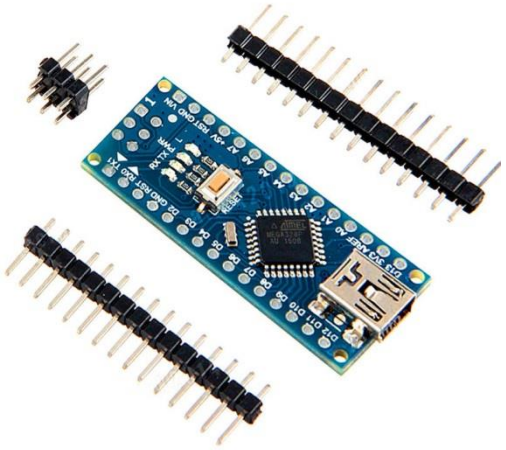


Рисунок 2.1—ArduinoNano

Технические характеристики Arduino Nano:

а) Напряжение питания 5В;

б) Входное питание 7-12В (рекомендованное);

в) Количество цифровых пинов — 14, из них 6 могут использоваться в

качестве выходов ШИМ;

г) 8 аналоговых входов;

д) Максимальный ток цифрового выхода 40 мА;

е) Флэш- память 16 Кб или 32 Кб, в зависимости от чипа;

ж) ОЗУ 1 Кб или 2 Кб, в зависимости от чипа;

и) EEPROM 512 байт или 1 Кб;

к) Частота 16 МГц;

л) Размеры 19 х 42 мм;

м) Вес 7 г.

Питание платы может осуществляться двумя способами:

- Через mini-USB или microUSB при подключении к компьютеру;
- Через внешний источник питания, имеющий напряжение 6-20 В с низким уровнем пульсаций.

Внешний источник стабилизируется на 5В, в соответствии со схемой LM1117IMPX-5.0. Подключение к стабилизатору при подключении через кабель от компьютера происходит через диод Шоттки. Схемы обоих типов питания приведены на рисунке 2.2.

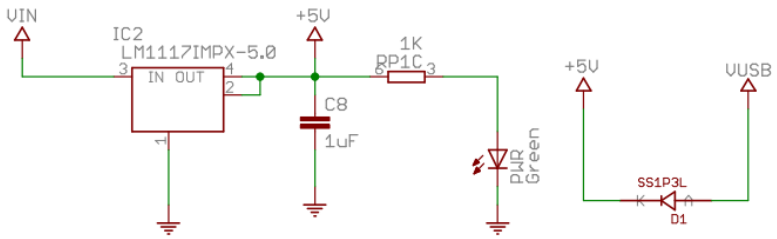


Рисунок 2.2—Схема питания

При подключении, из двух источников напряжения плата всегда выбирает с наибольшим питанием.

Так же существуют ограниченияна входы и выходы платыпо напряжению и току. Все аналоговые и цифровые контакты могут работать только в диапазоне от 0 до 5 В. Если же подается питание выходящее за рамки этих

значений, тонапряжение будет ограничиваться защитными диодами. В данном случае сигнал должен подключиться через резистор, чтобы контроллер не вышелиз строя. Наибольшее значениедля втекающего ивытекающего тока не должно быть более 40 мА, а общий ток контактов должен быть не более 200 мА.

На плате есть 4 светодиода, которые показывают состояние сигнала. Они обозначаются как: TX, RX, PWR и L. В первых двух светодиод светится, когда уровень сигнала низкий, и указывает, что сигнал TX или RX активен. Светодиод PWR загорается при напряжении 5 В и указывает, что питание включено. Последний светодиод - это индикатор общего назначения, когда подается высокий сигнал [7].

На настоящий момент выпускается несколько видов Arduino Nano. Есть версии 2.X, 3.0., которые отличаются только чипом, на котором они работают. В версии 2.X. используется чип ATmega168 с меньшим объемом памяти (флэш, энергонезависимой) и пониженной тактовой частотой, версия 3.0. работает на чипе ATmega328.

Плата Arduino Nano имеет 14 цифровых контактов, которые помечаются буквой D (цифровой, digital) (рисунок 2.3). Контакты используются как входы и выходы, у каждого имеется подтягивающий резистор.

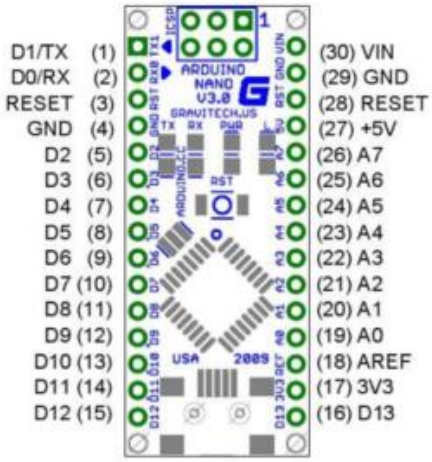


Рисунок 2.4—Контакты платы ArduinoNano

1 пин —TX (передача данных UART), D0;

2 пин —RX (прием данных UART), D1. RX и TX могут использоваться для связи по последовательному интерфейсу или как обычные порты данных;

3,29пины —сброс;

4, 29 пины —земля;

D2 пин —прерывание INT0;

D3пин —прерывание INT1 / ШИМ / AIN0;

A4пин—счетчикT0 / шинаI2CSDA / AIN1.AIN0 и AIN1 – входы для быстродействующего аналогового компаратора;

A5пин—счетчикT1 / шинаI2CSCL / ШИМ;

16 пин —порты D6-D13, из которых D6 (9й), D9 (12й), D10 (13й) и D11 (14й) используются как выходы ШИМ. D13 (16й пин) —светодиод. Также D10 —SS, D11 —MOSI, D12 —MISO, D13 —SCK используются для связи по интерфейсу SPI;

AREFпин —это опорное напряжение для АЦП микроконтроллера;

26 пин —аналоговые входы A— A7. Разрядность АЦП 10 бит. A4 (SDA), A5 (SCL) – используются для связи по шине I2C. Для создания используется специальная библиотека Wire.

Такие контакты можно использовать в качестве выходов ШИМ. Arduino Nano оснащена шестью такими контактами —это пины D3, D5, D6, D9, D10, D11. Чтобы использовать ШИМ, создана специальная функция analogWrite().

Аналоговые выводы обозначены буквой «A» и используются в качестве входов. У них нет подтягивающих резисторов, они измеряют приложенное к ним напряжение и возвращают значение с помощью функции analogRead ().

На некоторых цифровых выводах можноувидеть значок «~». Такие контакты могут использоваться как выходы ШИМ. Arduino Nano оснащен шестью контактами этого типа, это контакты—D3, D5, D6, D9, D10, D11. Чтобы использовать ШИМ, была создана специальная функция analogWrite ().

Микроконтроллеры обладают отличной функциональностью, но у них

есть один недостаток: это ограниченное количество выводов. Поэтому на этапе создания схемыустройства, следует подумать о том, как максимально упростить проект, чтобы уменьшить количество контактов, которые необходимо подключить.

Arduino Nano поддерживает интерфейс I2C для связи с различными устройствами и периферией. Один из часто встречающихся способов применения – это связь с дисплеем через шину I2C,благодаря этой технологии можновыводить наборы символов и данных на дисплей, используя всего лишь 2 пина, в Nanoэто пин D4(SDA) и D5(SCL) [10].

2.2 I2Cшина

Шина I2C является одной из модификаций последовательных протоколов обмена данными.

В стандартном режиме 8-битные последовательные данные передаются со скоростью до 100 кбит / с и до 400 кбит / с в быстром режиме.

Для реализации процесса обмена информацией по шине I2C используются только два сигнала: линия данных SDA и линия синхронизации SCL. Чтобы гарантировать, что шина является двунаправленной, без использования сложных шинных арбитров, выходные каскады устройств, подключенных к шине, имеют открытый сток или открытый коллектор, чтобы обеспечить функцию монтажного «И» [14].

Рассмотрим наглядно схему включения устройств на шинеI2C (рисунок 2.5).

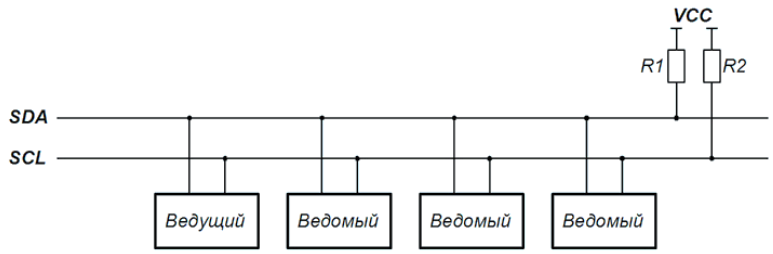


Рисунок 2.5—Схема включения устройств на шине I2C

Простая двухпроводная последовательная шина I2C сводит к минимуму количество соединений между микросхемами, микросхемы имеют меньше контактов и требуется меньше дорожек. В результате печатные платы становятся проще и технологичнее в процессе производства. Интегрированный протокол I2C устраняет необходимость в декодерах адресов и другой внешней логике сопоставления.

Максимально допустимое количество микросхем, подключенных к одной шине, ограничено максимальной емкостью шины 400 пФ.

Алгоритм подавления помех, основанный на аппаратном обеспечении, встроенном в микросхемы, гарантирует целостность данных при наличии значительного шума.

Все I2C-совместимые устройства имеют интерфейс, который позволяет им связываться друг с другом по шине, даже если их напряжение питания значительно отличается. На рисунке 2.6 показан принцип подключения нескольких интегральных схем с различными напряжениями питания к обменной шине.

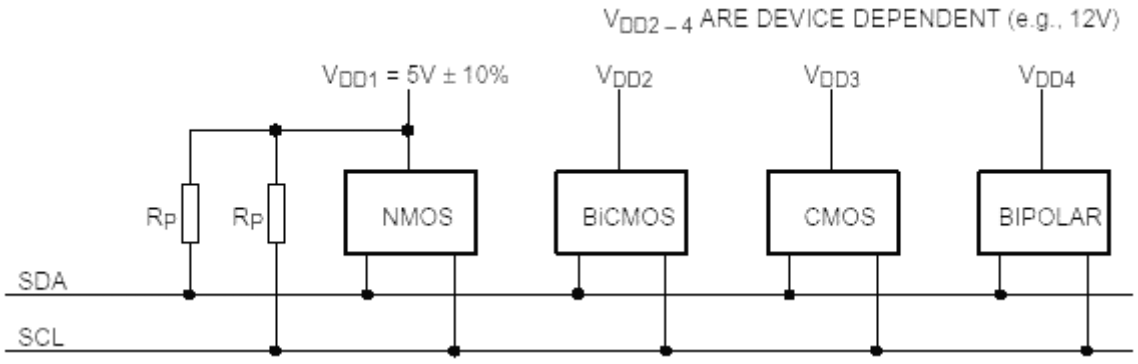


Рисунок 2.6—Принцип подключения нескольких ИМС к одной шине

Каждое из устройств распознается по уникальному адресу и может работать как передатчик или приемник, в зависимости от назначения устройства[25].

Кроме того, во время передачи данных, устройства могут быть

классифицированы как ведущие и ведомые. Ведущий—это устройство, которое инициирует передачу данных и генерирует сигналы синхронизации. В этом случае любое адресуемое устройство считается ведомым по отношению к ведущему.

В соответствии со спецификацией работы шины, в каждый отдельный момент на шине может быть только один ведущий, а именно , устройство обеспечивающее формирование сигнала шины SCL. Ведущий может действовать как ведущий передатчик и ведущий приемник. Однако шина позволяет иметь несколько ведущих, которые накладывают определенные характеристики ее поведения при формировании сигналов управления и при мониторинге состояния шины. Возможность подключения более одного ведущего к шине означает, что более чем один ведущтй могут пытаться начать отправку одновременно. Чтобы устранить «столкновения», которые могут возникнуть в этом случае, была разработана арбитражная процедура: поведение лидера при обнаружении «захвата» шины другим лидером [15].

Процедура синхронизации двух устройств —эта процедура основана на том факте, что все устройства I2C подключены к шине в соответствии с правилами монтажа. В исходном состоянии сигналы SDA и SCL находятся в высоком состоянии.

Состояния «СТАРТ» и «СТОП» —процедура обмена начинается с ведущего, формирующего состояние «СТАРТ» —ведущий генерирует переход сигнала SDA сВЫСОКОГО на НИЗКИЙ при ВЫСОКОМ уровненалинии SCL. Этот переход воспринимается всеми устройствами, подключенными к шине, как сигнал начала процедуры обмена.

На рисунке 2.7проиллюстрирована временнаядиаграмма сигналов шины.

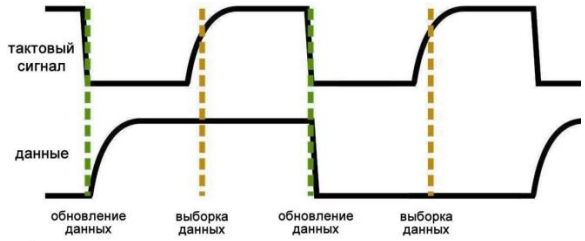


Рисунок 2.7—Временная диаграмма сигналов шины I2C

Генерация синхросигнала —это всегда обязанность ведущего; каждый ведущий генерирует свой собственный сигнал синхронизации при пересылке данных по шине.

Процедура обмена завершается тем, что ведущий формирует состояние «СТОП»—переход состояния линии SDA из низкого состояния в ВЫСОКОЕ при ВЫСОКОМ состоянии линии SCL.

Состояния «СТАРТ» и «СТОП» всегда вырабатываются ведущим. Считается, что шина занята после фиксации состояния «СТАРТ». Шина считается освободившейся через некоторое время после фиксации состояния «СТОП».

При передаче посылок по шине I2C каждый ведущий генерирует свой синхросигнал на линии SCL.

После формирования состояния «СТАРТ», ведущий опускает состояние линии SCL в НИЗКОЕ состояние и выставляет на линию SDA старший бит первого байта сообщения. Количество байт в сообщении не ограничено.

Спецификация шины I2C разрешает изменения на линии SDA только при НИЗКОМ уровне сигнала на линии SCL [26].

Данные действительны и должны оставаться стабильными только во время ВЫСОКОГО состояния синхроимпульса.

Для подтверждения приема байта от ведущего —передатчика ведомым - приемником в спецификации протокола обмена по шине I2C вводится специальный бит подтверждения, выставляемый на шину SDA после приема 8 бита данных.

Далее идет подтверждение. Таким образом передача 8 бит данных от передатчика к приемнику завершаются дополнительным циклом (формированием 9-го тактового импульса линии SCL (рисунок 2.8)), при котором приемник выставляет низкий уровень сигнала на линии SDA, как признак успешного приема байта [15].

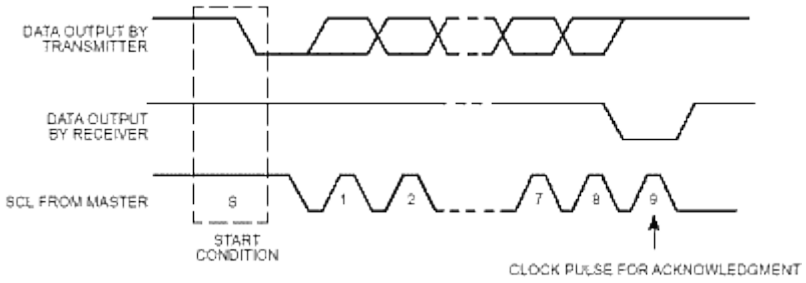


Рисунок 2.8—Формирование 9-го тактового импульса

Подтверждение при передаче данных обязательно. Соответствующий импульс синхронизации генерируется ведущим. Передатчик отпускает (ВЫСОКОЕ) линию SDA на время синхроимпульса подтверждения. Приёмник должен удерживать линию SDA в течение ВЫСОКОГО состояния синхроимпульса подтверждения в стабильном НИЗКОМ состоянии.

В том случае, когда ведомый-приёмник не может подтвердить свой адрес (например, когда он выполняет в данный момент какие-либо функции реального времени), линия данных должна быть оставлена в ВЫСОКОМ состоянии. После этого ведущий может выдать сигнал «СТОП»для прерывания пересылки данных.

Если в пересылке участвует ведущий-приёмник, то он должен сообщить об окончании передачи ведомому-передатчику путем не подтверждения последнего байта. Ведомый-передатчик должен освободить линию данных для того, чтобы позволить ведущему выдать сигнал «СТОП»или повторить сигнал «СТАРТ»[25].

Синхронизация выполняется с использованием подключения к линии SCL по правилу монтажного И.

Это означает, что ведущий не имеет монопольного права на управление переходом линии SCL из НИЗКОГО состояния ВЫСОКОГО. В том случае, когда ведомому необходимо дополнительное время на обработку принятого бита, он имеет возможность удерживать линию SCL в низком состоянии до момента готовности к приему следующего бита. Таким образом, линия SCL будет находиться в НИЗКОМ состоянии на протяжении самого длинного НИЗКОГО периода синхросигналов.

Устройства с более коротким НИЗКИМ периодом будут входить в состояние ожидания на время, пока не кончится длинный период. Когда у всех задействованных устройств кончится НИЗКИЙ период синхросигнала, линия SCL перейдет в ВЫСОКОЕ состояние. Все устройства начнут проходить ВЫСОКИЙ период своих синхросигналов. Первое устройство, у которого кончится этот период, снова установит линию SCL в НИЗКОЕ состояние. Таким образом, НИЗКИЙ период синхролинии SCL определяется наидлиннейшим периодом синхронизации из всех задействованных устройств, а ВЫСОКИЙ период определяется самым коротким периодом синхронизации устройств [22].

Механизм синхронизации может быть использован приемниками как средство управления пересылкой данных на байтовом и битовом уровнях.

На уровне байта, если устройство может принимать байты данных с большой скоростью, но требует определенное время для сохранения принятого байта или подготовки к приему следующего, то оно может удерживать линию SCL в НИЗКОМ состоянии после приема и подтверждения байта, переводя таким образом передатчик в состояние ожидания.

На уровне битов, устройство такое как микроконтроллер без встроенных аппаратных цепей I2C или с ограниченными цепями может замедлить частоту синхроимпульсов путем продления их НИЗКОГО периода. Таким образом скорость передачи любого ведущего адаптируется к скорости медленного устройства [21].

Каждое устройство, подключённое к шине, может быть программно адресовано по уникальному адресу.

Для выбора приемника сообщения ведущий использует уникальный адресную компоненту в формате посылки. При использовании однотипных устройств, ИС часто имеют дополнительный селектор адреса, который может быть реализован как в виде дополнительных цифровых входов селектора адреса, так и в виде аналогового входа. При этом адреса таких однотипных устройств оказываются разнесены в адресном пространстве устройств, подключенных к шине.

В обычном режиме используется 7-битная адресация.

Процедура адресации на шине I2C заключается в том, что первый байт после сигнала СТАРТ определяет, какой ведомый адресуется ведущим для проведения цикла обмена. Исключение составляет адрес "Общего вызова", который адресует все устройства на шине. Когда используется этот адрес, все устройства в теории должны послать сигнал подтверждения. Однако, устройства могут обрабатывать "общий вызов" на практике встречаются редко.

Первые семь битов первого байта образуют адрес ведомого. Восьмой, младший бит, определяет направление пересылки данных. "Ноль" означает, что ведущий будет записывать информацию в выбранного ведомого. "Единица" означает, что ведущий будет считывать информацию из ведомого [13].

После того, как адрес послан, каждое устройство в системе сравнивает первые семь бит после сигнала СТАРТ со своим адресом. При совпадении устройство полагает себя выбранным как ведомый-приёмник или как ведомый-передатчик, в зависимости от бита направления.

Адрес ведомого может состоять из фиксированной и программируемой части.

Часто случается, что в системе будет несколько однотипных устройств (к примеру ИМС памяти, или драйверов LED-индикаторов), поэтому при помощи программируемой части адреса становится возможным подключить к шине

максимально возможное количество таких устройств. Количество программируемых бит в адресе зависит от количества свободных выводов микросхемы. Иногда используется один вывод с аналоговой установкой программируемого диапазона адресов, как это, к примеру, реализовано в ИМС SAA1064. При этом в зависимости от потенциала на этом адресном выводе ИМС, возможно смещение адресного пространства драйвера так, чтобы однотипные ИМС не конфликтовали между собой на общей шине.

Все ИМС, поддерживающие работу в стандарте шины I2C, имеют набор фиксированных адресов, перечень которых указан производителем в описаниях контроллеров.

Комбинация бит 11110XX адреса зарезервирована для 10-битной адресации [15].

В общем виде процесс обмена по шине от момента формирования состояния СТАРТ до состояния СТОП можно проиллюстрировать следующим рисунком :

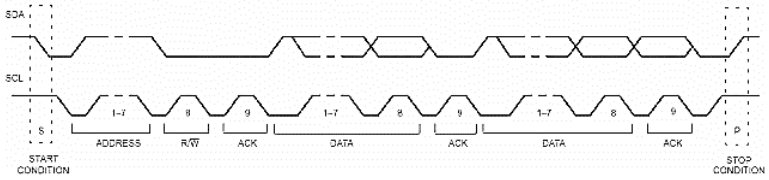


Рисунок 2.9—Момент формирования состояния «СТАРТ/СТОП»

Как следует из спецификации шины, допускаются как простые форматы обмена, так и комбинированные, когда в промежутке от состояния СТАРТ до состояния СТОП ведущий и ведомый могут выступать и как приемник и как передатчик данных. Комбинированные форматы могут быть использованы, например, для управления последовательной памятью.Во время первого байта данных можно передавать адрес в памяти, который записывается во внутренний регистр-защелку. После повторения сигнала СТАРТА и адреса ведомого выдаются данные из памяти. Все решения об авто-инкременте или декременте

адреса, к которому произошел предыдущий доступ, принимаются конструктором конкретного устройства. Поэтому, в любом случае лучший способ избежать неконтролируемой ситуации на шине перед использованием новой (или ранее не используемой) ИМС следует тщательно изучить ее описание (datasheet), получив его с сайта производителя. Более того, производители часто размещают рядом более подробные инструкции по применению.

В любом случае, по спецификации шины все разрабатываемые устройства должны сбрасывать логику шины при получении сигнала СТАРТ или повторный СТАРТ и подготавливаться к приему адреса.

Тем не менее основные проблемы с использованием I2C шины возникают именно из-за того, что разработчики, "начинающие" работать с I2C шиной не учитывают того факта, что ведущий (часто - микропроцессор) не имеет монопольногоправа ни на одну из линий шины [25].

Все новые устройства с I2C интерфейсом работают в быстром режиме. Предпочтительно, они должны уметь принимать и/или передавать данные на скорости 400 кбит/с. Как минимум они должны быть способны входить в синхронизацию в быстром режиме, с тем чтобы снизить скорость передачи (путем удлиннения НИЗКОГО периода SCL) до допустимой величины [26].

2.3 Обмен данных по Bluetooth на основе модуля HC-05

Самыми распространёнными модулями являются модули на основе чипа BC417 – серия называется HC. Так как все они построены на основе одного контроллера, все их различия заключаются в прошивке.

Среди устройств фирмы HC есть модули и адаптеры Bluetooth, к примеру:

а) Модуль последовательного интерфейса Bluetooth:

- Для промышленного применения: HC-03, HC-04(HC-04-M,HC-04-S);

- Для бытового применения: HC-05, HC-06(HC-06-M, HC-06-S),HC-05-D, HC-06- D (с платой, для тестирования и анализа).

б) АдаптерBluetooth: HC-M4;HC-M6.

Модуль предназначен для организации последовательного порта через Bluetooth. У модуля есть два режима работы: режим ведущего (master) и ведомого (slave). Для устройств с чётными номерами в названии данные режимы устанавливаются на заводе и не могут быть изменены. В устройствах же с нечётными номерами в названии пользователь может устанавливать режим работы в режиме ведущего/ведомого с помощью АТ-команд.

Модули HC-03 и HC-05 могут выступать как сервером соединения, так и клиентом, и имеют расширенный набор конфигурационных (АТ-) команд. Остальные модули могут работать только в режиме клиента и имеют урезанный функционал АТ-команд.

При выпуске с завода устройств HC-03 и HC-05 часть параметров предустановлена во время активации устройства. Режим работы не задан, он настраивается пользователем [32].

Основная функция модуля Bluetooth — это организация связи по последовательному интерфейсу там, где ранее для связи применялась кабельная линия, к примеру:

а) Есть две платы с микроконтроллерами, которые должны обмениваться данными. Одна плата соединяется с ведущим устройством Bluetooth, вторая —с ведомым. Между парой устройств устанавливается беспроводное Bluetooth соединение. Данная связь аналогична соединению плат кабелем, при которой данные передаются по линиям RXD и TXD последовательного интерфейса.

б) Когда к устройству подключен модуль Bluetooth, работающий в режиме ведомого. Он позволяет связываться с компьютерами и смартфонами посредством встроенного в них интерфейса Bluetooth.

в) 3.Основная часть устройств Bluetooth на рынке — это устройства в режиме ведомого, например принтеры и модули GPS. Модуль в режиме ведущего может устанавливать с ними связь.

Модули не требуют драйверов, они могут взаимодействовать с другими устройствами Bluetooth. Однако для взаимодействия двух модулей необходимо выполнить следующие условия:

- Связь осуществляется между ведущим и ведомым устройствами;
- Введен верный пароль;
- Однако выполнения этих условий недостаточно. В зависимости от конкретной модели устройства могут присутствовать дополнительные условия.

Выбор модуля —модули с чётными номерами в названии совместимы друг с другом. Также совместимы модули в режиме ведомых. Другими словами, функции HC-04 и HC-06, HC-03 и HC-05 взаимно совместимы. HC-04 и HC-06 являются старыми версиями, в которых пользователь не может установить рабочий режим (ведомое или ведущее)[32].

Доступно также только ограниченное число АТ-команд и функций: смена имени устройства (только в режиме ведомого), смена пароля, установка скорости передачи и проверка версии.

Набор команд для HC-03 и HC-05 предоставляет возможность гибко их использовать. Поэтому, в основном, HC-03/HC-05 наиболее рекомендуемы к применению.

Назначение контактов модулей HC-03, HC-04, HC-05 и HC-06 различно, однако размеры платы одинаковы: 28мм * 15мм * 2.35мм.

На рисунке 2.10представлен модуль HC-06 с основными контактами.

Рисунок 2.10—HC-06

На рисунке 2.11представлен модуль HC-05 с основными контактами.

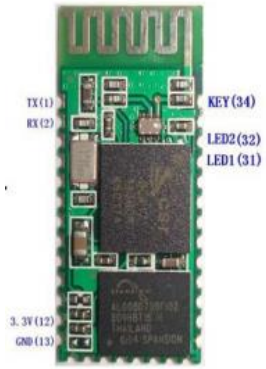
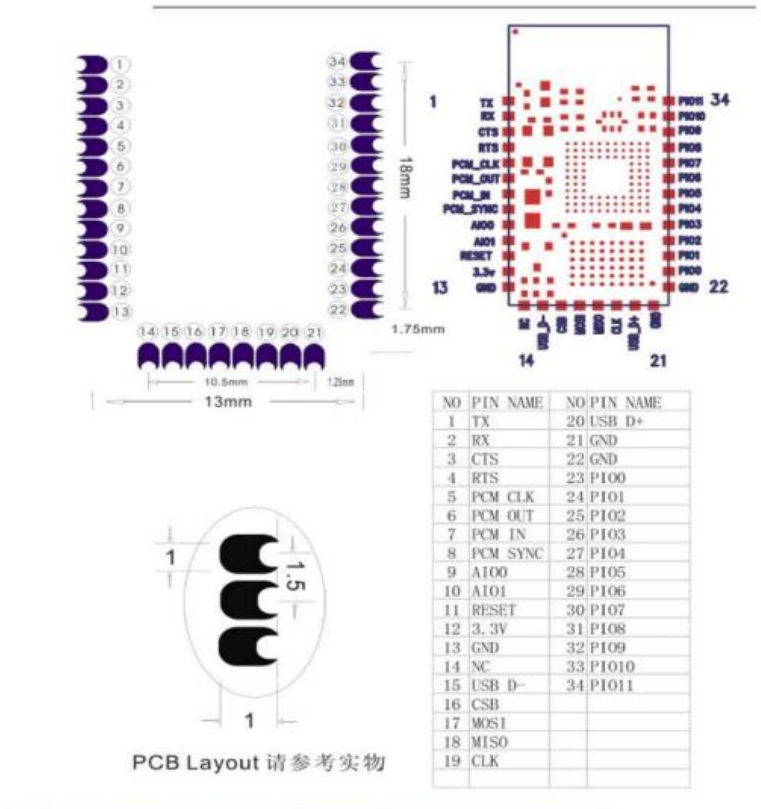


Рисунок 2.11—HC-05

Рисунок 2.12показывает информацию о размере платы.



NO	PIN NAME	NO	PIN NAME
1	TX	20	USB D+
2	RX	21	GND
3	CTS	22	GND
4	RTS	23	PIO0
5	PCM_CLK	24	PIO1
6	PCM_OUT	25	PIO2
7	PCM_IN	26	PIO3
8	PCM_SYNC	27	PIO4
9	AIO0	28	PIO5
10	AIO1	29	PIO6
11	RESET	30	PIO7
12	3.3V	31	PIO8
13	GND	32	PIO9
14	NC	33	PIO10
15	USB D-	34	PIO11
16	CSB		
17	MOSI		
18	MISO		
19	CLK		

Рисунок 2.12—Распиновка и размеры

Далее будет расписано как использовать и тестировать HC-05 при первом запуске и подробно описаны контакты модуля (рисунок 2.13). А так же наглядно показаны прикладные схемы соединения с устройствами 3.3В и 5В (рисунок 2.14-2.15) [29].

PIN1	UART_TXD, контакт передачи последовательного интерфейса Bluetooth, можно соединять с контактом RXD на плате микроконтроллера.
PIN2	UART_RXD, контакт передачи последовательного интерфейса Bluetooth, можно соединять с контактом TXD на плате микроконтроллера. Подтягивающий резистор не установлен.
PIN11	RESET, контакт сброса модуля, подача низкого уровня приводит к сбросу.
PIN12	VCC, электропитание схемы. Стандартный вольтаж: 3,3В, диапазон работы: 3,0-4,2В
PIN13	GND, заземление
PIN31	LED1, светодиод, индикатор рабочего режима. Режима три: Когда на модуль подается электропитание, а на контакт PIN34 — высокий уровень, то на выходе PIN31 присутствует прямоугольный сигнал частотой 1Гц. Светодиод медленно моргает. Это означает работу в режиме AT и скорость передачи 38400. Когда на модуль подается электропитание, а на контакт PIN34 — низкий уровень, то на выходе PIN31 присутствует прямоугольный сигнал частотой 2Гц. Светодиод моргает быстро. Это означает работу в режиме установления связи. При подаче на PIN34 высокого уровня модуль перейдет в режим AT, однако частота на PIN31 будет по-прежнему 2Гц. После установления связи частота сигнала на PIN31 также 2Гц. Примечание: Если высокий уровень на PIN34 подается постоянно, то работают все команды; если высокий уровень не сохраняется, то возможен только ограниченный набор команд. Больше информации об этом представлено в Главе 2
PIN32	Контакт выхода. До установления связи на выходе низкий уровень. После установления связи — высокий.
PIN34	Вход переключения режима. Если подается низкий уровень, то модуль работает в режиме передачи. При подаче высокого уровня модуль переходит в режим AT. Даже при установленной связи модуль может перейти в режим AT. При восстановлении низкого уровня на PIN34 модуль вновь возвращается в режим передачи.

Рисунок 2.13—Описание контактов модуля HC-05

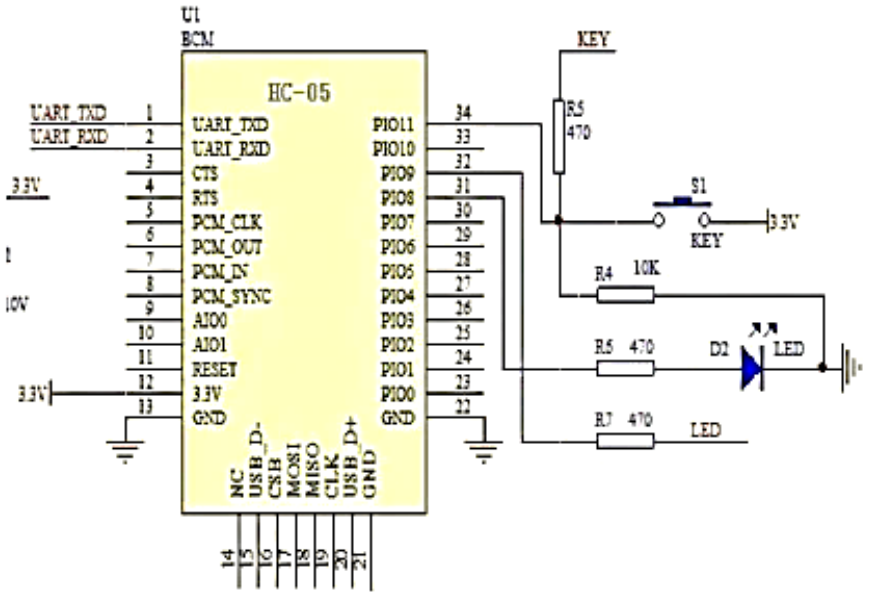


Рисунок 2.14—Прикладная схема 1 (3.3В)

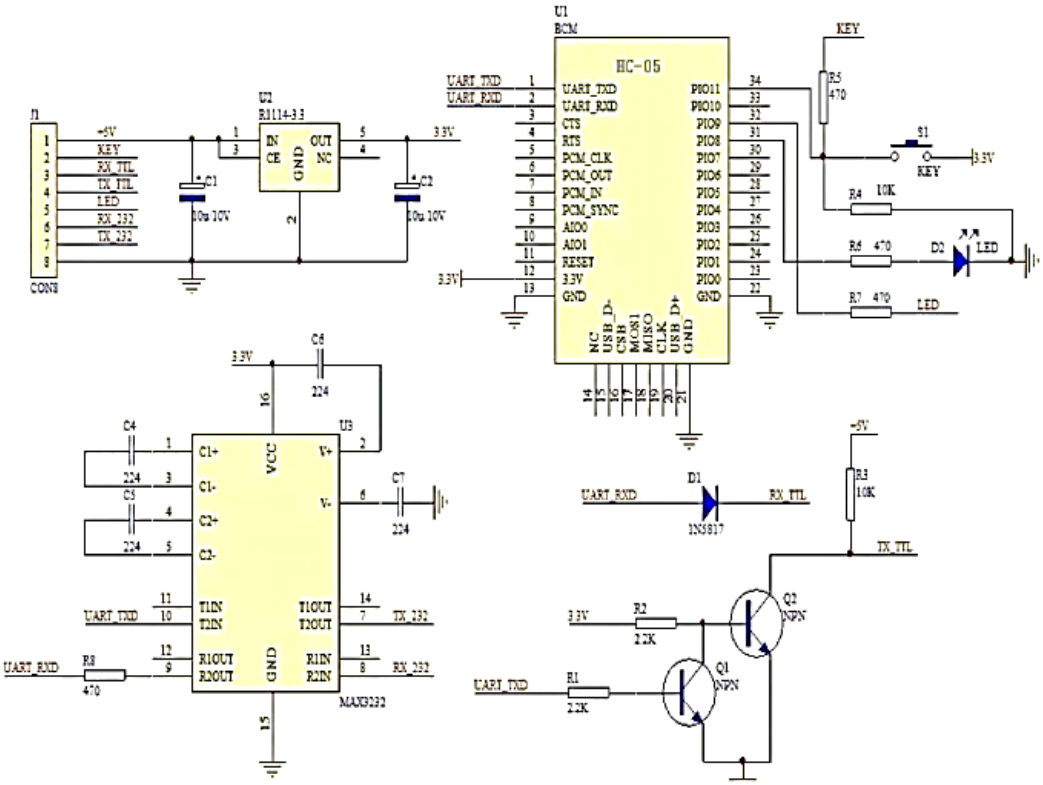


Рисунок 2.15—Прикладная схема 2 (5В)

У модуля HC-05 есть много функций, включая функции модуля HC-06. Однако, HC-05 предоставляет пользователю больше свободы в использовании. Таким образом, HC-05 предпочтительнее модуля HC-06 и рекомендован к применению. Модуль HC-03 схож с HC-05. Информация выше подходит и для него.

Разные прошивки модулей подразумевают разные выводы для индикации.

У модулей HC-03/05 индикация построена следующим образом:

- индикатор рабочего режима —PIO8 (31 пин);
- индикатор статуса соединения —PIO9 (32 пин) [32].

Также имеется специальный пин для ввода модуля в режим AT-команд — PIO11, или пин 34.

У модулей HC-04/06/07 имеется только один выход статуса —PIO11, на который подаётся 1 в случае установления соединения.

Модули «в чистом виде» (без платы-переходника) имеют шаг выводов

1.5мм, что не позволяет припаять к модулю стандартную 0.1гребёнку контактов для макетной платы. Поэтому придётся либо приобретать плату-переходник, либо вывести нужные контакты проводами.

Для стандартного подключения нужно вывести и подключить:

- Контакт 12 модуля 3.3V Arduino;
- Контакт 13 модуля GND Arduino;
- Контакт 2 модуля TX Arduino;
- Контакт 1 модуля RX Arduino.

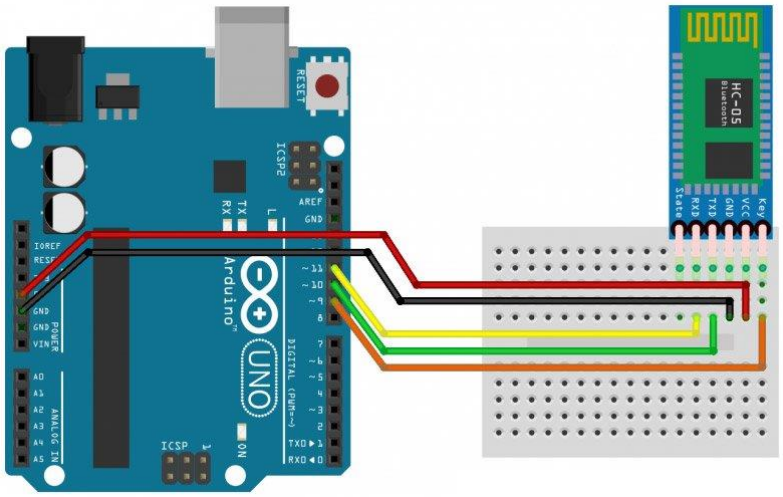


Рисунок 2.16—Пример подключения модуля к Arduino

Модуль Bluetooth HC-05 сможет принимать команды только если правильно настроены следующие параметры последовательного порта:

- **Скорость передачи данных:**В обычном режиме,Bluetooth модуль HC-05 сохраняет последнюю установленную скорость передачи данных, но по умолчанию она равна 38400 бит/сек (редко 9600 бит/сек);
- **Параметры передачи данных:**Модуль сохраняет последние установленные параметры передачи данных. Значения параметров по умолчанию: количество бит в пакете - 8, размер стопового битаь - 1, без проверки чётности;
- **Передаваемый текст:**Нужно установить пункт «добавлять символы

CR &LF(NL)» это символы перевода строки «\r\n» которые Вы не сможете ставить самостоятельно в конце АТ-команд.

При использованииArduino, номер порта указывается во вкладке «Инструменты». Параметры передачи данных используются по умолчанию. Для добавления символов NL&CR воспользуйтесь меню в правом нижнем углу монитора последовательного порта.

После каждого подключения питания или перезагрузки модуля, до того как отправлять команды, нужно кратковременно нажать на кнопку модуля. Если у модуля нет кнопки, то кратковременно подать высокий уровень на вывод K. После чего модуль останется в обычном режиме, но будет воспринимать АТ-команды. АТ-команда это строка начинающаяся с букв «АТ» (от английского **at**tention - «внимание»). Модуль выполняет поступившую команду и отправляет обратно ответ (результат выполнения команды), который также является строкой.

BBluetooth модулях HC-05каждая команда (как и ответ) должна заканчиваться символами перевода строки «\r\n»Помимо обычного режима, модуль может работать в режиме АТ-команд. О том как войти в этот режим и чем он отличается от обычного, рассказано ниже, в разделе примечание [31].

Далее на рисунке 2.17приведеныосновные АТ-команды для модулей HC-05:

Для проверки связи с Bluetooth модулем необходимо отправить тестовую команду**AT**(ввести текст**AT**и нажать Enter). Если связь установлена корректно, то модуль ответит**OK**. После этого можно отправлять остальные АТ-команды [29].

Если отправить команду, которую модуль не знает, не может выполнить, или у команды неправильные аргументы, то модуль вернёт строку «ERROR:(НОМЕР)», где по указанному шестнадцатиричному номеру можно определить, на что «ругается» модуль.

HC-05 (любая AT команда отправляется с символами CR-LF)		
(*) – режим по умолчанию		
Тестирование	AT	Ответ OK
Адрес устройства	00.14.02.12.06.91 (вид адреса на смартфоне)	
	AT+ADDR?	+ ADDR : 14:2:120691 OK
Имя модуля	AT+NAME?	+ NAME : HC-05 (*) OK
	AT+NAME=XXX	Где XXX до 20 символов Ввод: AT+NAME=06-91 Ответ: OK
Пароль	AT+PSWD?	+ PSWD : 1234 (*) OK
	AT+PSWD=XXXX	Где XXXX – новый 4-х разрядный пароль Ввод: AT+PSWD=4321 Ответ: OK
UART	AT+UART?	+ UART : 9600,0,0 (*) OK
	AT+UART=X,Y,Z	Где X → 1200; 2400; 4800; 9600 (*); 19200; 38400; 57600; 115200 … Y → 0 – один стоп бит (*); 1 – два стоп бита Z → 0 – нет проверки (*) 1 – проверка нечетности 2 – проверка четности AT+UART=38400,0,0 Ответ: OK
Режим работы	AT+ROLE?	+ ROLE : 0 OK
	AT+ROLE=X	Где X → 0 – Slave (*) 1 – Master 2 – Slave-loop AT+ROLE=1 Ответ: OK
Режим подключения	AT+CMODE?	+ CMOD : 1 (*) OK
	AT+CMODE=X	Где X → 0 – фиксированный адрес 1 – к любому адресу (*) 2 – Slave-loop AT+CMODE=0 Ответ: OK
Фиксированный адрес Slave	AT+BIND?	+ BIND : 0:0:0 (*) OK
	AT+BIND=NNN	NNN, например, 00.13.12.20.77.18 (так выглядит на смартфоне) Ввод: AT+BIND=13,12,207718 Ответ: OK

Рисунок 2.17—AT команды модуля HC-05

3 Разработка устройства на основе ATmega328 и модуле HC -05

3.1 ATmega328

ATmega328 —это однокристальный микроконтроллер, созданный Atmel в семействе megaAVR (рисунок 3.1). Он имеет модифицированное 8-битное ядро RISC с архитектурой Гарвардской архитектуры.

Рисунок 3.1—Микроконтроллер Atmega328

Память:

- 32 kB Flash (память программ, имеющая возможность самопрограммирования);
- 2 kB ОЗУ;
- 1 kB EEPROM (постоянная память данных).

Периферийные устройства:

- Два 8-битных таймера/счетчика смодулям сравнения и делителями частоты;
- 16-битный таймер/счетчик с модулем сравнения и делителем частоты, а также с режимом записи;
- Счетчик реального времени с отдельным генератором;
- Шесть каналов PWM (аналог ЦАП);
- 6-канальный ЦАП со встроенным датчиком температуры;

- Программируемый последовательный порт USART;
- Последовательный интерфейс SPI;
- Интерфейс I2C;
- Программируемый сторожевой таймер с отдельным внутренним генератором;
- Внутренняя схема сравнения напряжений;
- Блок обработки прерываний и пробуждения при изменении напряжений на выводах микроконтроллера[16].

Ниже представлена схема распиновки микронтроллера:

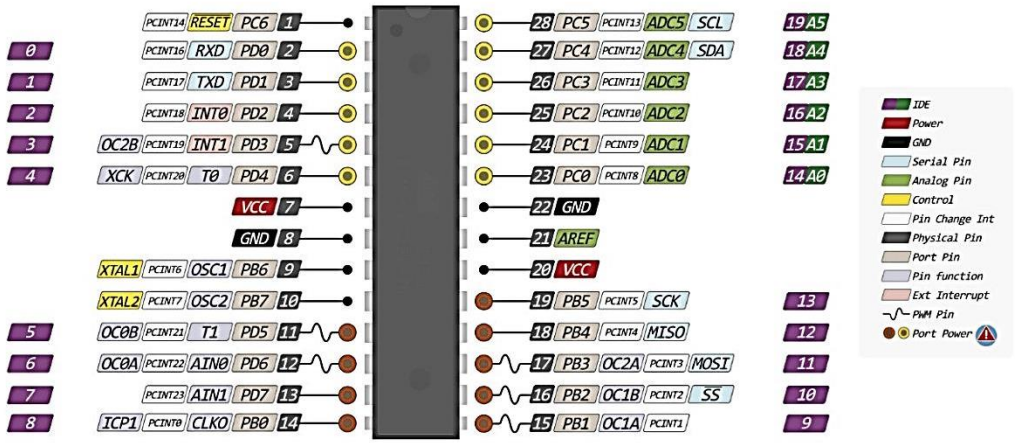


Рисунок 3.2—Распиновка микроконтроллера ATmega328

Специальные функции микроконтроллера ATmega328:

- Сброс при включении питания и программное распознавание снижения напряжения питания;
- Внутренний калибруемый генератор тактовых импульсов;
- Обработка внутренних и внешних прерываний;
- 6 режимов сна (пониженное энергопотребление и снижение шумов для более точного преобразования АЦП).

Напряжения питания и скорость процессора:

- 1.8 — 5.5 В при частоте до 4 МГц;

- 2.7 — 5.5 В при частоте до 10 МГц;
- 4.5 — 5.5 В при частоте до 20 МГц.

Микроконтроллер ATmega328 изготавливается по малопотребляющей КМОП технологии, которая в сочетании с усовершенствованной RISCархитектурой позволяет достичь наилучшего соотношения «быстродействие/энергопотребление».

RISC—архитектура процессора, в котором быстродействие увеличивается за счёт упрощения инструкций, чтобы их декодирование было более простым, а время выполнения — меньшим [17].

Микроконтроллер построен по двухшинной (гарвардской) архитектуре и имеет раздельные шины памяти программ и памяти данных (рисунок 3.3).

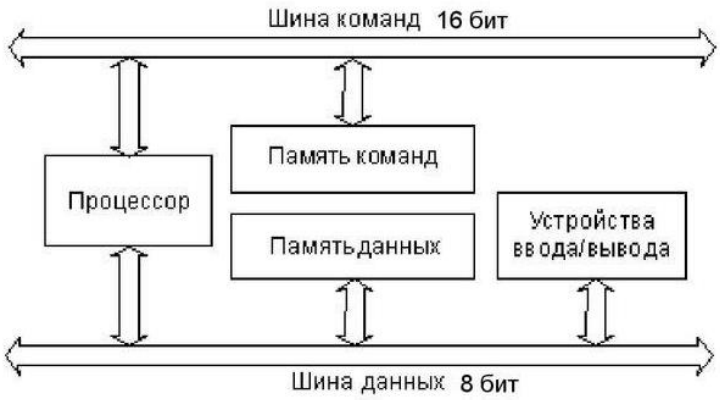


Рисунок 3.3 —Двухшинная Гарвардская архитектура

В соответствии с Гарвардской архитектурой разделены не только адресные пространства памяти программ и памяти данных, но также и шины доступа к ним. Способы адресации и доступа к этим областям памяти так же различны. Такая структура позволяет центральному процессору работать одновременно как с памятью программ, так и с памятью данных, что существенно увеличивает производительность.

Каждая из областей памяти данных (ОЗУ и EEPROM) также расположена в своем адресном пространстве. Схематично представлено на рисунке 3.4.

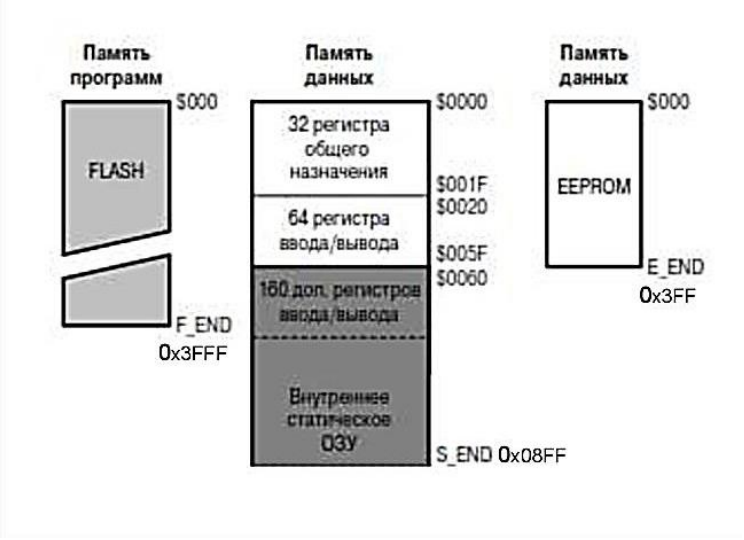


Рисунок 3.4—Схематичное представление Гарвардской архитектуры

В гарвардской архитектуре принципиально невозможно осуществить операцию записи в память программ, что исключает возможность случайного разрушения управляющей программы в случае ошибки программы при работе с данными или атаки третьих лиц. Кроме того, для работы с памятью программ и с памятью данных организуются отдельные шины обмена данными (системные шины).

Эти особенности определили области применения гарвардской архитектуры. Гарвардская архитектура применяется в микроконтролерах и в сигнальных процессорах, где требуется обеспечить высокую надёжность работы аппаратуры. В сигнальных процессорах Гарвардская архитектура дополняется применением трехшинного операционного блока микропроцессора. Трехшинная архитектура операционного блока позволяет совместить операции считывания двух операндов с записью результата выполнения команды в оперативную память микропроцессора. Это значительно увеличивает производительность сигнального микропроцессора без увеличения его тактовой частоты [1].

В Гарвардской архитектуре характеристики устройств памяти программ и памяти данных не всегда выполняются одинаковыми. В памяти данных и команд могут различаться разрядность шины данных и распределение адресов

памяти. Часто адресные пространства памяти программ и памяти данных выполняют различными. Это приводит к различию разрядности шины адреса для этих видов памяти. В микроконтроллерах память программ обычно реализуется в виде постоянного запоминающего устройства, а память данных — в виде ОЗУ. В сигнальных процессорах память программ вынуждены выполнять в виде ОЗУ. Это связано с более высоким быстродействием оперативного запоминающего устройства, однако при этом в процессе работы осуществляется защита от записи в эту область памяти [2].

Применение двух системных шин для обращения к памяти программ и памяти данных в гарвадской архитектуре имеет два недостатка — высокую стоимость и большое количество внешних выводов микропроцессора. При использованиии двух шин для передачи команд и данных, микропроцессор должен иметь почти вдвое больше выводов, так как шина адреса и шина данных составляют основную часть выводов микропроцессора. Для уменьшения количества выводов кристалла микропроцессора фирмы-производители микросхем объединили шины данных и шины адреса для внешней памяти данных и программ, оставив только различные сигналы управления (WR, RD, IRQ) а внутри микропроцессора сохранили классическую гарвардскую архитектуру. Такое решение получило названиемодифицированная гарвардская архитектура.

Модифицированная гарвардская структура применяется в современных микросхемах сигнальных процессоров. Ещё дальше по пути уменьшения стоимости кристалла за счет уменьшения площади, занимаемой системными шинами пошли производители однокристалльных ЭВМ — микроконтроллеров. В этих микросхемах применяется одна системная шина для передачи команд и данных (модифицированная гарвардская архитектура) и внутри кристалла [17].

3.2 Описание дисплея

Символьный дисплей «LCD2004» —жидкокристаллический дисплей (Liquid Crystal Display) экран которого способен отображать одновременно до80 символов (20 столбцов, 04 строки) (рисунок 3.5).

Подключение кArduinoосуществляется по синхронному 8-битному параллельному интерфейсу. Для подключения «LCD2004» к микроконтроллеру имеется интерфейс «I2C».

Рисунок 3.5—Символьный дисплей

Характеристики:

- Тип выводимой информации: символьный;
- Язык в ПЗУ дисплея: латиница, японский;
- Возможность загрузки собственных символов: есть;
- Формат выводимой информации: 20×04 символов;
- Тип дисплея: LCD;
- Технология дисплея: STN;
- Угол обзора: 180°;
- Тип подсветки: LED;
- Цвет подсветки: синий;
- Цвет символов: белый;
- Контроллер: HD44780;
- Интерфейс: синхронный, 8-битный, параллельный;
- Напряжение питания 5 В;
- Рабочая температура: -20 ... +70 °C;

- Температура хранения -30 ... +80 °C;

Дисплей LCD2004 оснащён платой конвертером для преобразования параллельного 8-битного интерфейса дисплея в шину I2C, по которой он и подключается к Arduino по адресу 0x3F или 0x27. Наличие последовательного интерфейса позволяет общаться с контроллером Arduino по средствам 2-х проводной связи, это поможет сэкономить цифровые пины контроллера для подключения дополнительной периферии [27].

Так же на I2C/SPI конвертере установлен потенциометр для регулировки яркости подсветки.

Дисплей LCD-2004B-IIC может одновременно отображать до 80 символов (20 символов, 4 строки).

Дисплей оснащён светодиодной подсветкой синего цвета.

Контроллер дисплея HD44780 имеет ПЗУ в которой хранятся цифры, символы латиницы и некоторые иероглифы японского языка, для их отображения на дисплее. Отсутствующие символы, в т.ч. и символы кириллицы, можно загружать в память ОЗУ контроллера.

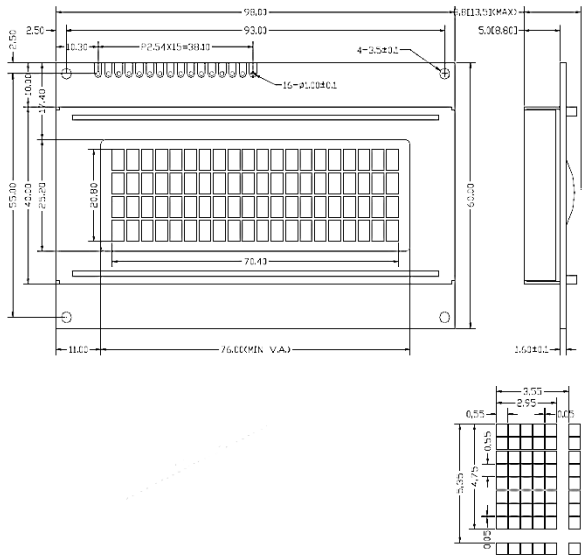


Рисунок 3.6 —Размеры дисплея

Далее рассмотрим как происходит подключение дисплея LCD2004 к

Arduino через I2C:жидкокристаллический монитор LCD2004 с поддержкой I2C подключается к плате Arduino при помощи четырех проводов —два провода для данных, два провода для питания.

- GND —общий;
- VCC — «+5 В»;
- SDA —последовательная линия данных —на Arduino Uno и Nano A4 (SDA), на Arduino Mega – 20 (SDA);
- SCL —последовательная линия синхронизации —на Arduino Uno и Nano A5 (SCL), на Arduino Mega — 21 (SCL)[15].

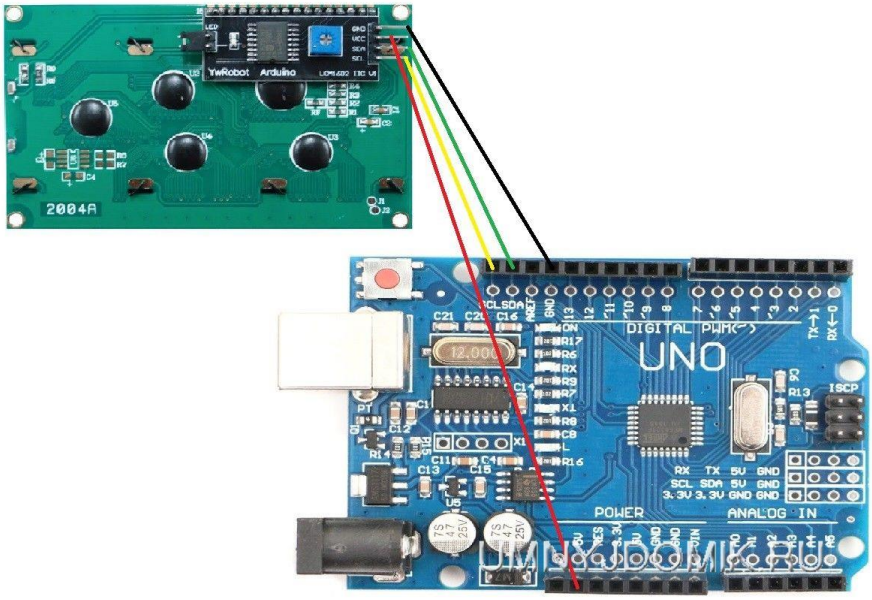


Рисунок 3.7—Схема подключения 2004-LCD к Arduino через I2C

Длявзаимодействие Arduino с LCD 2004 по шине I2C нам потребовалось две библиотеки:

а) Библиотека Wire.h для работы с I2C (уже имеется в стандартной программе Arduino IDE);

б) Библиотека LiquidCrystal_I2C.h, которая включает в себя большое разнообразие команд для управления монитором по шине I2C и позволяет

сделать скетч проще и короче.

После подключения дисплея нужно дополнительно установить библиотеку LiquidCrystal_I2C.h. Существуют различные версии этой библиотеки, актуальная версия NewliquidCrystal_1.3.4.

Далее необходимо:

а) Скачать библиотеку NewliquidCrystal (версия 1.3.4);

б) Разархивировать папку с библиотекой “NewliquidCrystal” в папку, где установлена среда программирования «Arduino IDE» (например: C:Program FilesArduinolibraries);

в) Переименовать папку библиотеки “NewliquidCrystal” на “LiquidCrystal_I2C”;

г) Перезагрузить среду программирования «Arduino IDE»;

д) Загрузить скетч в плату Arduino;

После подключения к скетчу всех необходимых библиотек можно приступать к работе.

3.3Описание схемы устройства

ArduinoNanoэто полный аналог ArduinoUno —он также работает на чипе ATmega328P (хотя все еще можно найти варианты с применением ATmega168), но с меньшим форм-фактором. Плата часто используется в проектах, в которых важна компактность. На плате нет внешнего источника питания, устройство Arduino работает через USB (miniUSB или microUSB). Остальные параметры такие же, как у модели ArduinoUno.

а) рассчитать расход топлива;

б) отображать информацию о температуре антифриза;

в) рассчитать скорость и расстояние поездки;

г) вывести отработавшее топливо на определенный пробег;

д) определить скорость двигателя и т.д.

е) рассчитать расход топлива;

ж) отображать информацию о температуре антифриза;

и) рассчитать скорость и расстояние поездки;

к) вывести отработавшее топливо на определенный пробег;

л)определить скорость двигателя и т.д.

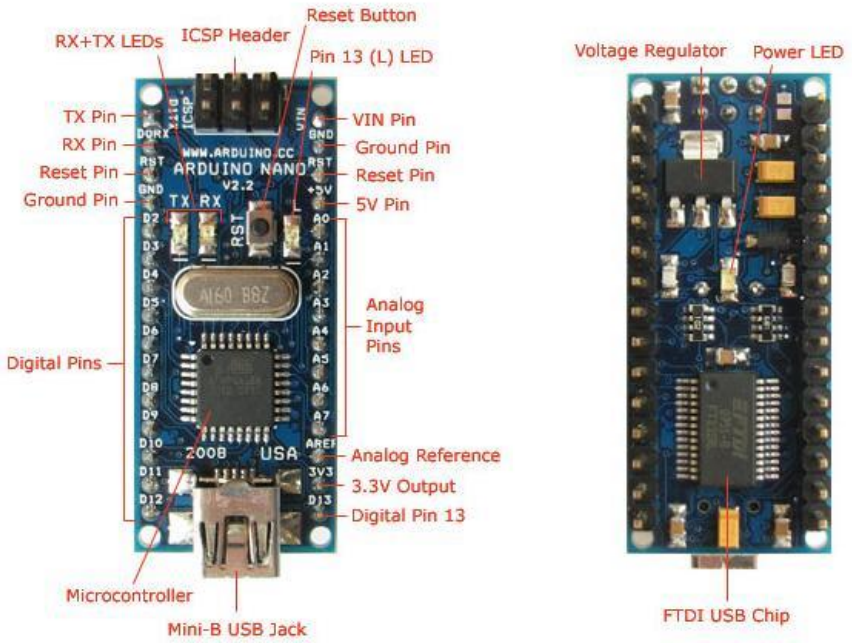


Рисунок 3.8 —Плата ArduinoNano

Используя плату Arduino Nano, можно построить бортовой компьютер автомобиля, который позволит:

В дополнение к устройству DD1 (Arduino-nano) также нужен жидкокристаллический модуль дисплея HG1 (LCD2004), адаптер DD2 (Bluetooth NS-05), а также сканер ELM327 и резистор R1 сопротивлением 10 кОм (используется для настройки яркости дисплея).

Разумеется, необходимо подготовить пьезоэлектрический излучатель PB1 в качестве индикатор выбора меню, который срабатывает каждый раз при нажатии кнопки, которая подключается к разъёму XP1 к контакту 2 (ECU). На элементах DA1,C1-C4 собран стабилизатор напряжения [3] для питания модуля

DD1 напряжение на входе данного блока составляет 9В, светодиод сигнализирует о наличии выходного напряжения со стабилизатора.

Для уменьшения нагрузки на внутренний стабилизатор напряжения модуля DD1, который формирует напряжение 5 В для работы модуля DD2 и индикатора HG1. Резисторы R3, R4 представляют собой делитель напряжения для измерения входного напряжения на бортовой компьютер[11].

Схема устройства приведена на рисунке 3.9.

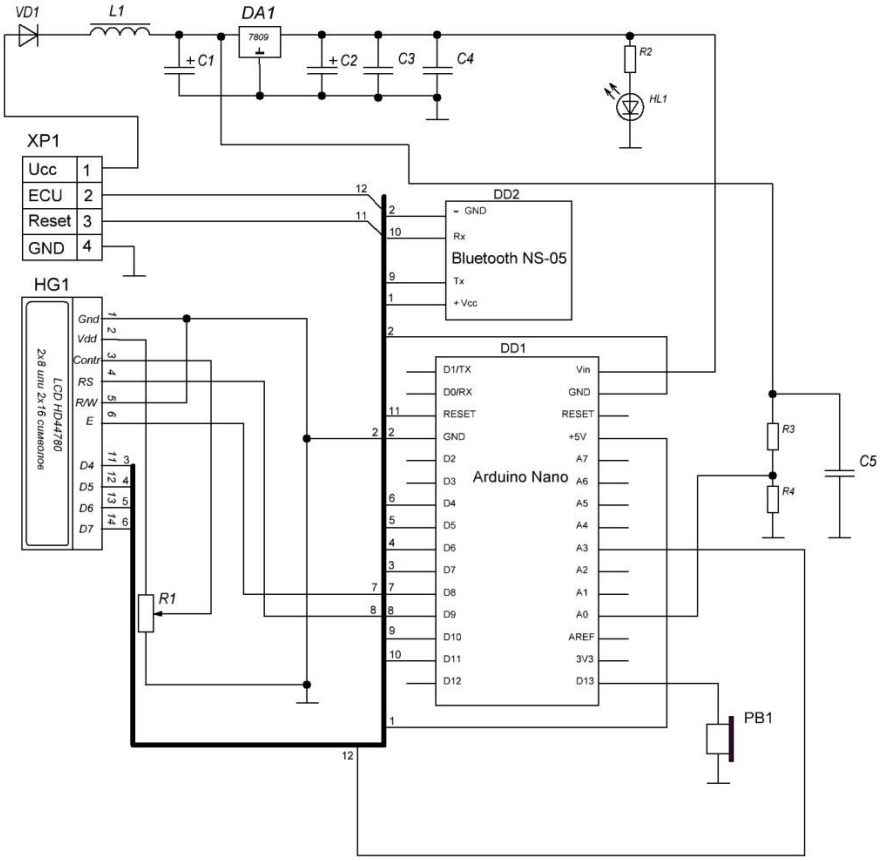


Рисунок 3.9—Схема бортового компьютера на основе Arduino-Nano

Процедура сборки и настройки следующая:

а) Сначала настроимадаптер Bluetooth. Необходимо припаятьпровода к контактам устройства;

б) Сам модуль подключается к плате для конфигурации, для этого нужно

открыть программу Arduino IDE любой версии, после того, как она запустится,залить программу в схему через USB-вход;

в) Когда загрузка будет завершена, нужно перейти в меню Service —Port - Monitor и установить скорость до 9600 бит / с.;

г) Затем схема собирается с платой Arduino-Nano, адаптером и дисплеем, подготовленным заранее. Сначала подключается адаптер Bluetooth (рисунок 3);

- 1 — TX модуля засовываем в 7 Pin (Rx) арудины (именно TX в RX, не так как ранее);
- 2 — RX модуля засовываем в 8 Pin (Tx) арудины;
- 12 — Pin (3,3V) модуля в Pin 3,3V ардуины;
- 13 — Pin (Gnd) в Gnd арудуины;
- 34 — Pin мы никуда не подключаем (заизолируйте или отпаяйте).

Рисунок 3.10—Подключение контактов Bluetooth

д) Кроме того, подключается дополнительная кнопка, которая будет выполнять функцию переключения экранов с информацией. Один контакт от кнопки переходит к элементу GND, второй - к контакту A3. Для подключения бипера положительный контакт подключен к 13 контактам, а отрицательный - к GND;

е) Затем, используя то же программное обеспечение Arduino IDE, необходимо залить прошивку. Теперь просто нужно настроить бортовой компьютер и подключить его к автомобилю [27].

Используя достаточно недорогую плату Arduino-Nano и модуль Bluetooth HC-05 совместно с адаптером ELM327 можно получить возможность читать данные с диагностической шины автомобиля и конфигурировать программное обеспечение таким образом, чтобы выводить на жидкокристаллический индикатор только ту информацию, которая необходима пользователю автомобиля. Предложенная схема выгодно отличается от стандартных компьютеров и приложения для них, где доступность изменения программы и получения дополнительных данных не представляется возможным. Данное

устройство можно усовершенствовать, добавив к нему модуль удаленного управления, описанный в статье.[

Далее описан подробнее диагностическийавтосканерDigimotor Dmscan Elm 327 Obd2 v1.5 (рисунок 3.11).

Рисунок 3.11 —Адаптер Digimotor Dmscan Elm 327 Obd2 v1.5

Вид изнутри, разъем и распиновка адаптера представлены ниже.

Рисунок 3.12 —Внутренний вид адаптера

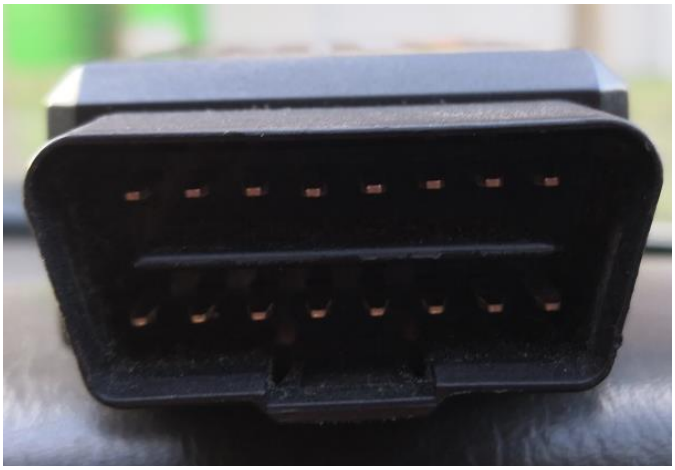


Рисунок 3.13 —Разъем адаптера (OBD2)

Распиновка адаптераDigimotor Dmscan Elm327представлена на рисунке 3.14.

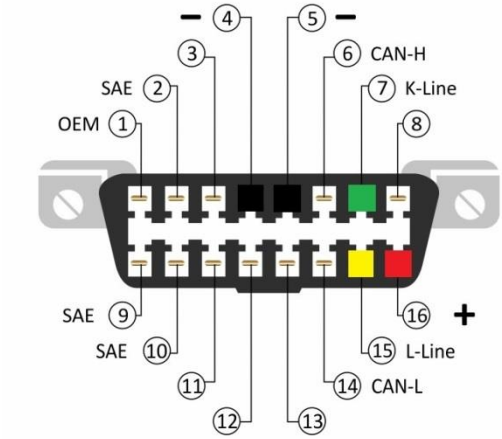


Рисунок 3.14 —Распиновка адаптера

Сканеры OBD-2, построенные на базе микроконтроллера ELM327 хорошо универсальны и надежны. Благодаря этим качествам многие разработчики диагностического программного обеспечения выпускают свои программы для сканеров этого типа. В Интернете размещено множество программ, обладающих различными функциональными возможностями, интерфейсами и языковой поддержкой. Среди них есть как коммерческие продукты, так и бесплатные версии.В нашем случае было использовано приложение TorquePro. Это единственная программа обладающая огромным количеством функций, и к

тому же она имеет полностью настраиваемый русский интерфейс.

Далее поэтапно разберемкак настроить адаптер:

а) Запускаем настройки Android;

б) Включаем Bluetooth. Ждем некоторое время. После того как Android найдет устройствоDigimotor Dmscan Elm 327 Obd2 v1.5 (рисунок 3.15), он попросит ввести пароль. В нашемслучаеподходящим оказалсяпароль: 1234.

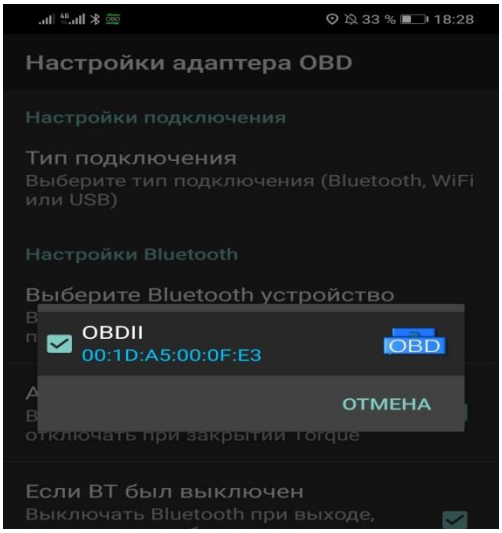


Рисунок 3.15—Выбор Bluetooth-устройства (адаптер OBD2)

в) При включении, на адаптере горит одна лампочка. А когда внешнее устройство Bluetooth подключено и идет передача данных — все лампочки мигают в режиме бегущего огня.

Затем настройки идут в выбраном нами приложении для диагностики (Torque) .

а) Настраиваем профиль нашего автомобиля (рисунок 3.16);

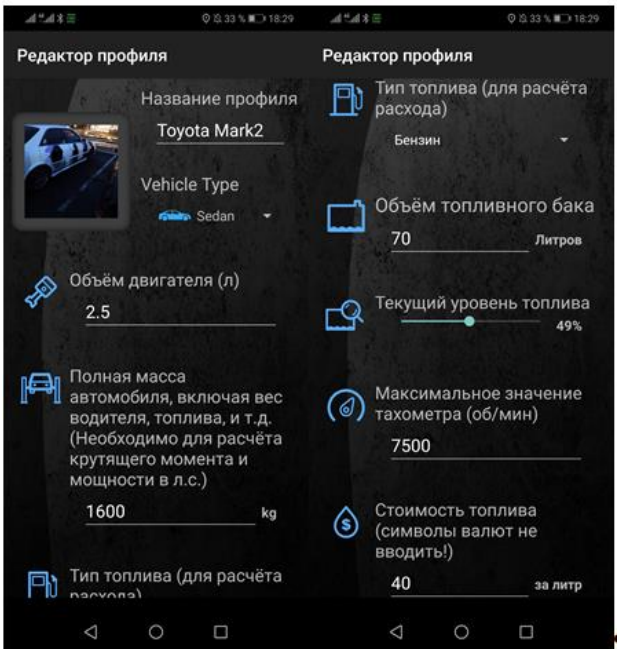


Рисунок 3.16—Настройка профиля автомобиля в приложении Torque

б) Задаем настройки и прописываем альтернативный заголовок для адаптера (рисунок 3.17);

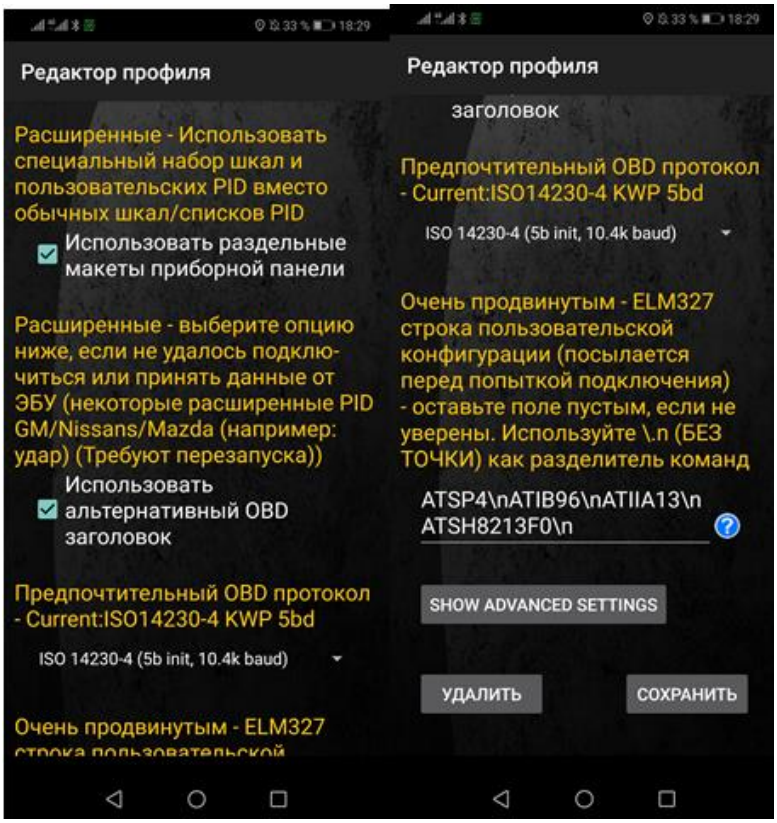


Рисунок 3.17—Настройки адаптера в приложении Torque

Программный листинг управляюйщей программы микроконтроллера представлен в приложении А.

3.4 Сборка устройства

Ниже будет представлен список требуемых деталей для сборки бортового компьютера.

а) Arduino Nano;

Рисунок 3.18 —Плата Arduino Nano

б) ЖК-модуль;

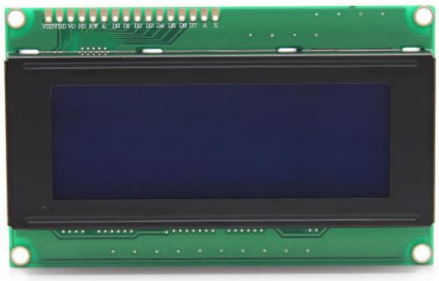


Рисунок 3.19 —LCD2004 экран

в)Модуль Bluetooth HC-05;

Рисунок 3.20 —Модуль Bluetooth

г)DigiMotorOBDELM327 Bluetoothсканер;

Рисунок 3.21 —Адаптер OBDELM327

д)Резистор 10 кОм подстроечный, бипер для звука, 2 кнопки для смены экранов, провода для соединений, корпус;

е)Разъем OBD2 (в салоне и разъем самодиагностики под капотом).

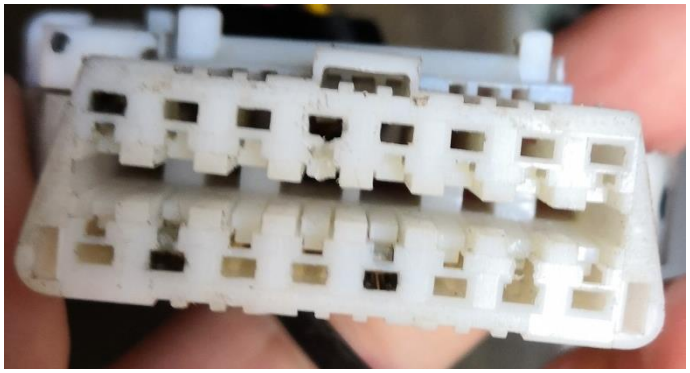


Рисунок 3.22 —Разъем диагностики (в салоне автомобиля)

Стоит отметить, что под капотом имеется разъем самодиагностики (рисунок 3.23), через который так же можно подключиться, однако в нем отсутствуют нужные пины, поэтому вся сборка производилась через разъем в салоне.

Рисунок 3.24 —Разъем самодиагностики (под капотом)

Процесс настройкиBluetooth модуля HC-05:

Подпаиваем провода к пинам Bluetooth:(картинку с выходами можно видеть в описании требуемых деталей).

1 –это TX

2 – этоRX

12 – это 3.3V

13 – это GND

34 – на этот вход тоже 3,3 V (нужен для перевода модуля в режим настройки с помощью AT команд).

Подключаем Bluetoothмодуль к Arduinoдля его настройки.

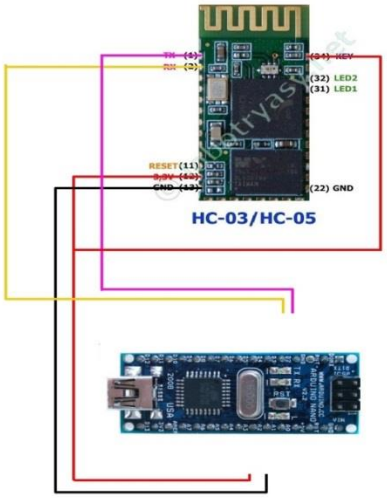


Рисунок 3.25 —Схема подключение модуля Bluetooth к ArduinoNano

1 –TXмодуля в 6 пин Arduino;

2 –RXмодуля в 7 пин Arduino;

12 – и 34 пин к 3,3V Arduino;

13 –GNDArduino.

Далее открываем программу Aduino IDE 1.0.6 и загружаемскетч через USB порт в плату.

```
#include <SoftwareSerial.h>
SoftwareSerial BTSerial(6, 7);
//TX | RX
                                                    voidsetup(){
                                                  Serial.begin(9600);
                                          Serial.println('EnterATcommands:');
                                                  BTSerial.begin(38400);}
void loop()
{
```

```
if (BTSerial.available())
                                        Serial.write(BTSerial.read());
if (Serial.available())
BTSerial.write(Serial.read());
}
```

Послеуспешнойзагрузкискетчаоткрываем следующие настройки:Сервис->Мониторпорта. Далее снизу ставим скорость 9600 бод и NL+CR вместе.

Далее вводим команды по одной и нажимаем «Послать». После каждого ввода должен быть ответ «ok».

AT//

AT+NAME=Car //ПрисваиваемимямодулюCar.

AT+ROLE=1 // Переводим модуль в режим Мастер.

AT+PSWD=1234 // Ставим пароль 1234 как на OBD ELM327.

AT+BIND=AABB,CC,112233 //Прописываем Mac адрес OBD ELM327.

AT+CMODE=1 // Подключение модуля с фиксированным адресом.

AT+UART=9600,0,0 // Скорость работы по UART.

Настройкамодуля Bluetooth закончена.

Далее необходимо собрать схему Arduino + Bluetooth – ЖК экран.

Схема представлена на рисунке 3.26.

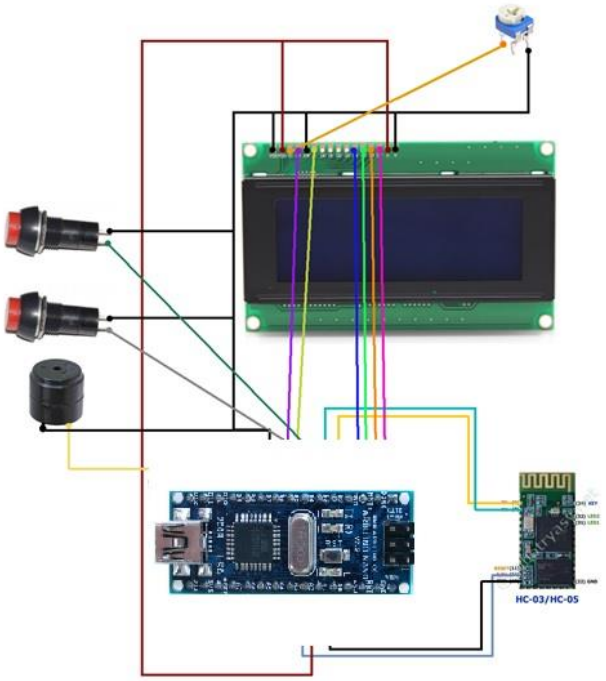


Рисунок 3.26 —схемаArduino + Bluetooth + ЖК экран

Начинаем с подключения HC-05 Bluetooth модуля:

- 1TX модуля засовываем в 7 Pin (Rx) Arduino (именно TX в RX, не так как ранее);
- 2 RX модуля засовываем в 8 Pin (Tx) Arduino;
- 12 Pin (3,3V) модулявPin 3,3VArduino;
- 13 Pin (Gnd) вGndArduino;
- 34 Pin никуда не подключаем.

Подключаем монитор LCD:

- VSS экрана к GND Arduino;
- VDD экрана к 5V Arduino;
- V0 экрана к центральному выходу резистора;
- RS экрана к 12 пину Arduino;
- RW экрана к GND Arduino;
- E экрана к 11 пину Arduino;
- DB4 экрана к 5 пину Arduino;

- DB5 экрана к 4 пину Arduino;
- DB6 экрана к 3 пину Arduino;
- DB7 экрана к 2 пину Arduino;
- A — к 5V Arduino;
- K — GND Arduino.

Одну из оставшихся ног потенциометра пустить на GND Arduino.

Переменный резистор на 10кОм нужен, чтобы управлять контрастностью монитора, отрегулировать контрастность шрифтаможно поворотом резистора.

Подключаем дополнительную кнопку для переключения экранов с данными.«1 кнопка»: один конец от нормально-открытойкнопки подключаем в GND Arduino, а второй конец в пин 10. «2 кнопка»: GND + пин 9.

Бипер для звуковых предупреждений подключается по следующей схеме «+»к пину 13, а минус к GND Arduino.

Загружаем скетч в Arduino с помощью Aduino IDE 1.0.6 . Нужно будет обязательно учесть три переменных:

а) ED=1.998; Например объем двигателя в литрах 1.398;

б) VE_correct=1.0; Корректировка объёмного КПД ДВС по таблице: (если расход реально меньше — то уменьшаем значение в процентном соотношении).

в) Speed_korrect_val=1; Корректировка скорости машины, смотреть по GPS/.

Управление:«Кнопка 1», «Кнопка 2» — листать экран вперед назад.

При включении при надписи «Connecting»… держать «кнопку 1»вход в режим показывания технологических экранов и параметров отдаваемых ЭБУ в 16-чном формате.

«Кнопка 1» + «Кнопка 2»: 4 секунды — Сброс журнала общего пробега и потраченного бензина на втором экране, также это сброс ошибок на экране информации об ошибках.

Далее проверяем работоспособность при подключении к автомобилю.

В программе Torqueможем так же наблюдать отображенные параметры (рисунок 3.27)

Рисунок 3.28—Панель приборов в программе Torque

На рисунке 3.29представлен итоговый вид бортового компьютера.

Рисунок 3.29 —Бортовой компьютер

ЗАКЛЮЧЕНИЕ

В результате выполнения бакалаврской работы рассмотрены общие вопросысвязанные с разработкой бортового компьютерадля автомобиляна основе ArduinoNano, разработан бортовой компьютер на базе кафедры Информационной безопасности и сервиса, Института электроники и светотехники.

В первой главе подробно изучены устройство и принцип работы контроллера ELM327, рассмотрены стандарты протокола OBD2 и принцип работы Can-шины.

Во второй главе описан процессобменаданнымиELM 327 по Bluetooth с микроконтроллером на базе ArduinoNano, где более подробно рассмотрены технические характеристики ArduinoNano и изучена шина I2C.

В третьей главе подробно описан и проиллюстрирован процеес сборки бортового компьютера.

Выпускная квалификационная работа содержит теоретическую и практическую часть. В теоретическойчасти представлены результаты изучения ипоиск информации поустройству и принципу работы контроллера ELM327, стандарту протокола OBD2 , изучен процессобменаданнымипо Bluetooth на основе модуля HC-05.В практическойчасти представлено описание элементов устройства, схема принципиальная электрическая, а так же весь процесс сборки и подключения.

В выпускной квалификационной работеописываются частные и системные проблемы подключения модуляELM327,существующей в автомобилях старых годоввыпуска, не оснащенных штатнымибортовыми компютерами, а так же, имеющими нерабочие разъемы самодиагностики.

В качестве вывода к отчету предлагаетсяготовыйразработанный бортовой компьютер, с приложением в виде листинга программы и схемы электрической принципиальной.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Авдеев В.А. Периферийные устройства: интерфейсы, схемотехника, программирование: учебное пособие / В.А. Авдеев. –М. : ДМК Пресс, 2014. – 848 с.

2 Авдеев В.А. Организация ЭВМ и периферия с демонстрацией имитационных моделей : учебное пособие / В.А. Авдеев.– М. : ДМК Пресс, 2014. – 708 с.

3 Аникин А.Обзор современных технологий беспроводной передачи данных в частотных диапазонах ISM (Bluetooth, ZigBee, Wi-Fi) и 434/868 МГц. /А. Аникин //Беспроводные технологии– 2011.–25.

4 Баранов В.Н. Применение микроконтроллеров AVR: схемы, алгоритмы, программы : учебное пособие / В.Н. Баранов.– М. : ДМК Пресс, 2010. – 288 с.

5 Бейктал Д. Конструируем роботов на Arduino. Первые шаги : руководство / Д. Бейктал; пер. с англ. О.А. Трефиловой.– М. : Издательство «Лаборатория знаний», 2016. – 323 с.

6 Белов А.В. ARDUINO: от азов программирования до создания практических устройств / А.В. Белов. –СПб : Наука и Техника, 2018. – 480 с.

7 Белов А.В. Программирование ARDUINO. Создаем практические устройства / А.В. Белов.–СПб : Наука и Техника, 2018. – 272 с.

8 Бирюков А.А. Информационная безопасность: защита и нападение / А.А. Бирюков. –М.: ДМК Пресс, 2017. – 434 с.

9 Бортовой компьютер своими руками [Электронный ресурс]. –Режим доступа:https://www.instructables.com/id/ArduinoLCD-HC05-Bluetooth/.

10 Боровский А.С. Программирование микроконтроллера Arduino в информационно-упрвляющих системах: учебное пособие / А.С. Боровский, М.Ю. Шрейдер–Оренбург : ОГУ, 2017. – 113 с.

11 Волков А.В. Разработка системы удаленного контроля на GSM модуле SIM900D / А.В. Волков, А.Д. Нуштайкина // В сборнике: XLVI Огарёвские чтения.Материалы научной конференции: В 3-х частях. Ответственный за выпуск П.В. Сенин. – 2018. – С. 268–273.

12 Елисеев Н. Arduino – это очень серьезно. Большие возможности маленьких устройств /Н. Елисеев, И. Шахнович // Электроника: Наука, Технология, Бизнес.– 2016. – 3.

13 Интерфейс I2C и Arduino [Электронный ресурс]. – Режим доступа: https://soltau.ru/i/en/arduino/item/interfejs-i2c-i-Arduino/.

14 Интерфейсная шина IIC (I2C) [Электронный ресурс]. – Режим доступа:http://easyelectronics.ru/interface-bus-iic-i2c.html/.

15 Логан С. Управление несколькими периферийными устройствами по линиям шин SPI/I2C / С. Логан //Компоненты и технологии–2008.–84.

16 Микроконтроллер ATMega328 [Электронный ресурс]. – Режим доступа: https://amperka.ru/product/avr-atmega328/.

17 Микроконтроллер ATmega328 – описание, характеристики [Электронный ресурс]. – Режим доступа: http://robolive.ru/atmega328-opisanie-xarakteristiki/.

18 Обзор OBD 2 ELM327 Bluetooth адаптеров для автодиагностики [Электронный ресурс]. – Режим доступа: https://elm327-obd2.ru/obzory/obd-2-diagnosticheskiy-adapter-elm327-bluetooth.html/.

19 Петин В.А. Практическая энциклопедия Arduino / В.А. Петин, А.А. Биняковский.– М. : ДМК Пресс, 2017. – 152 с.

20 Плата Arduino Nano v 3.0 : распиновка, схемы, драйвера [Электронный ресурс].–Режим доступа: https://arduinomaster.ru/platy-arduino/plata-arduino-nano/.

21 Семенов Б.Ю. Шина I2C в радиотехнических конструкциях / Б.Ю. Семенов.– М. : СОЛОН-Пресс, 2017. – 192 с.

22 Семенов Б.Ю. Шина I2C в радиотехнических конструкциях / Б.Ю. Семенов.– М. : СОЛОН-Пресс, 2015. – 224 с.

23 Спирин Ю.Л. УНИВЕРСАЛЬНЫЙ ЛАБОРАТОРНЫЙ КОМПЛЕКС «ФОРМИРОВАНИЕ ПРАКТИЧЕСКИХ НАВЫКОВ РАЗРАБОТКИ И ПРИМЕНЕНИЯ НАНО-, МИКРО- И ОПТОЭЛЕКТРОННЫХ ТЕХНОЛОГИЙНА ПЛАТФОРМЕ «ARDUINO / Ю.Л. Спирин //Проблемы современной науки и образования – 2015.– 12.

24 Что такое ARDUINO? [Электронный ресурс]. – Режим доступа: http://arduino-nano.ru/#desc/.

25 Шина управления I2C[Электронный ресурс]. – Режим доступа:http://cxem.net/comp/comp67.php/.

26 Шина I2C. Основные понятия [Электронный ресурс]. – Режим доступа:https://radioprog.ru/post/197/.

27 Эксперименты с Arduino. Часть 1. Бортовой компьютер [Электронный ресурс].– Режим доступа: //www.drive2.ru/l/6258151/.

28 Эмуляция ELM327 посредством Arduino [Электронный ресурс]. – Режим доступа:https://www.drive2.com/c/697276/.

29 AT-команды Bluetooth HC-05 [Электронный ресурс]. – Режим доступа: https://wiki.iarduino.ru/page/at-komandy-bluetooth-hc-05/.

30 Arduino Nano: обзор и характеристики миниатюрной платы от Arduino [Электронный ресурс]. – Режим доступа: https://arduinoplus.ru/arduino-nano/.

31 Arduino и Bluetooth-модули HC-05 [Электронный ресурс]. – Режим доступа: https://voltiq.ru/arduino-and-hc-05/.

32 Bluetooth модуль HC-05 [Электронный ресурс]. – Режим доступа: https://3d-diy.ru/wiki/arduino-moduli/bluetooth-modul-hc-05/.

33 ELM327 Bluetooth OBD-II Адаптер для диагностики автомобилей [Электронный ресурс]. – Режим доступа: https://www.drive2.ru/l/1341785/.

34 OBD2 для удаленной диагностики состояния

автомобиля[Электронный ресурс]. – Режим доступа:https://e.lanbook.com/journal/issue/284793//

35 CAN-шина в промышленных сетях [Электронный ресурс]// Компоненты и технологии.– 2018.– 17. – Режим доступа: https: /journal/284793/.

36 Can-шина в автомобильныхсетях [Электронный ресурс]. — Режим доступа:https://www.syl.ru/.

ПРИЛОЖЕНИЕ А

(обязательное)

Схема электрическаяпринципиальная

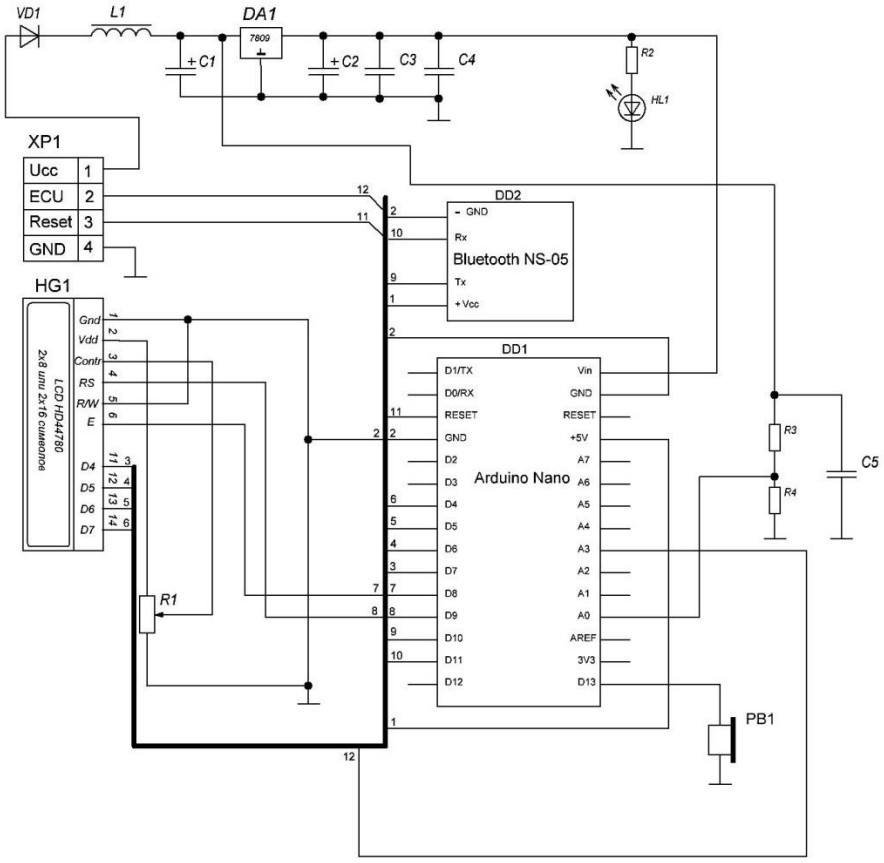


Рисунок А.1 — Схема электрическаяпринципиальная

ПРИЛОЖЕНИЕ Б
(обязательное)

Листинг управляющей программы микроконтроллера

```
  Бортовой компьютер для автомобиля на arduino + bluetooth hc-05 + elm327
*/
#include <LiquidCrystal.h>
#include <EEPROM.h>
#include "OBD.h"
//#define DEBUG Serial //раскоментировать чтобы получить дебаг информацию на Serial port (0) , скорость  для работы терминала 115200 бод
LiquidCrystal lcd(12, 11, 5, 4, 3, 2); // LiquidCrystal(rs, enable, d4, d5, d6, d7)
//----------------------
float voltage = 0;
COBD obd;
//----------------------
// ТРЕБУТКОРРЕКТИРОВКИ----------------------------------------------------------------------------------------------
double tcorrect = 1.0;       // ВНИМАНИЕ!!! Корректировка коэффициента времени arduino uno, у каждого свой.
double speed_korrect_val = 1; // ВНИМАНИЕ!!! Корректировка коэффициента скорости, посмотреть по GPS
double VE_correct = 1.0;       // ВНИМАНИЕ!!! Корректировка Объёмного КПД ДВС:  добиваемся чтобы мгновенный расход на холостых оборотах был в половину объема двигателя
double ED = 1.998;             // ВНИМАНИЕ!!! Объем двигателя в литрах (пример: 1.398)
//----------------------------------------------------------------------------------------------------------------
// НЕ НАДО ИЗМЕНЯТЬ ----------------------------------------------------------------------------------------------
byte count_display = 20; // Количество экранов бк
byte pin = 10, pin2 = 9; // пины кнопок
int engine_on_rpm = 400; // обороты при которых считать двигатель заведенным
double AirFuelRatio = 14.70;  // константа расхода 14,7 воздуха к 1 литра бензина, у дизеля своя, у газа своя
double FuelDensityGramsPerLiter = 750.0;   // константа - грамм бензина в 1 литре бензина
//----------------------------------------------------------------------------------------------------------------
boolean off2 = true;
char v1, v2, v3, v4, v5;
char rxData[20];
char rxIndex = 0;
int selmon, off, value, value2, t1, kol_check_engine_error_val, fss_val;
int speed_error, tmp_error, rpm_error, dvk_error, iat_error, rnd_error, dts_error, uoz_error, maf_error, pdz_error, ut_error, tm_error;
int dvk_var, intake_air_temp_var, davlenie_topliva_var, tmp_masla_var, speed_var, tmp_var, t2, check_engine_km, check_engine_flash;
unsigned long time_new, time_old, time_old_gurnal;
uint8_t time_to_reconnect;
int VE, rpm_var, uoz_var, raschet_nagruzka_dvigatelya_var, maf_var, polozh_dross_zaslon, urov-en_topliva_var, IMAP, MAF  ,
   long_term_val, short_term_val, b1s1_val, b1s2_val;
double LPH, FuelFlowGramsPerSecond, FuelFlowLitersPerSecond, ls_term_val, LP100, benz_add, time, odometr, benz_potracheno, odometr_add, odometr_add_gurnal, benz_add_gurnal;
byte  response[20];
static const unsigned char PROGMEM grad[8] =    // символградуса
{
B01100,
B10010,
```

```
B10010,
B01100,
  B00000,
  B00000,
  B00000,
  B00000,
};
void setup()
{
  time_to_reconnect = 0;
lcd.begin(20, 4);
pinMode(pin, INPUT);
digitalWrite(pin, HIGH);
pinMode(pin2, INPUT);
digitalWrite(pin2, HIGH);
lcd.setCursor(0, 0);
lcd.print("Connecting...");
delay(2000);
value = digitalRead(pin); // проверкакнопкидлявходавrecovery
if (value == LOW) {
off = 222;
   count_display = count_display + 3; // 3 скрытыхтехнологическихэкрана
lcd.clear();
lcd.setCursor(0, 1);
lcd.print("|----[recovery]----|");                    //отключаемвытаскиваниеданныхиз elm327 принажатиинакнопкупринадписи Connecting...
tone(13, 3000, 500);
delay(2000);
value = digitalRead(pin); // проверкакнопкидлявходав admin
if (value == LOW) {
off2 = false; // отключаем функцию getResponse, а то ничего не будет показывать без связи с блютуз
     lcd.clear();
     lcd.setCursor(0, 1);
     lcd.print("|----[all off]----|");   //отключаем вытаскивание данных из elm327
tone(13, 3000, 500);
delay(2000);
   }
  }
lcd.clear();
lcd.print("Connecting...[ok]");
delay(1000);
lcd.clear();
lcd.createChar(1, grad);
  //----------------
lcd.clear();
obd.begin();// созданиеобьекта obd
if (off != 222)
  {
lcd.print("initELM");
for (int8_t i = 9; i >= 0; i--) //небольшая задержка перед инициализацией для исключения ошибок подключения если БК включается от поворота ключа зажигания
{
lcd.setCursor(12, 0);
lcd.print(i);
delay(1000);
   }
   while (!obd.init()); // инициализация связи с компьютером автомобиля
}
```

```
  //------------------------
}

voidloop()
{
time_to_reconnect++;
  rpm_var = 0;
  IMAP = 0;
  MAF = 0;
  FuelFlowGramsPerSecond = 0;
  FuelFlowLitersPerSecond = 0;
  LPH = 0;
  odometr_add = 0;
  benz_add = 0;
ls_term_val = 0;
//-------------------
  if (time_to_reconnect > 10 ) // если в течении 10 циклов не появились обороты или был обрыв связи из-
за перепада напряжения во время работы стартера, реинициализация подключения
{
lcd.clear();
lcd.print("re-init ELM");
obd.end();
delay(100);
obd.begin();
while (!obd.init());
   time_to_reconnect = 0;
}
  //-------------------
  //---------------
  value = digitalRead(pin); // проверка состояния кнопки для переключения экрана
value2 = digitalRead(pin2);
if (value == LOW) {
if (selmon == count_display - 3) {
lcd.clear();
selmon = 0;
   }
else {
lcd.clear();
selmon++;
   }
tone(3, 3000, 100);
delay(150);
  }
if (value2 == LOW) {
if (selmon == 0) {
lcd.clear();
selmon = count_display - 3;
   }
else {
lcd.clear();
selmon--;
   }
delay(150);
tone(3, 3000, 100);
  }
  /*  IMAP = RPM * MAP / IAT
    MAF = (IMAP/120)*(VE/100)*(ED)*(MM)/(R)
    MAP - Manifold Absolute Pressure in kPa
```

```
   IAT - Intake Air Temperature in Kelvin
R  -  Specific Gas Constant (8.314472 J/(mol.K)
   MM -  Average molecular mass of air (28.9644 g/mol)
   VE -  volumetric efficiency measured in percent, let's say 80%
   ED - Engine Displacement in liters   */
if (off2)
  {
   if ((t1 == 60) or (t2 == 10)) { // чтение температуры и вольтажа каждые 100 циклов работы (примерно 30 секунд), первый раз чтение выполниться через примерно 5 секунд со старта
Tmp();
Voltage();
LongTerm();
    t1 = 0;
t2++;
   }
else {
t1++;
t2++;
   }
Rpm();
  }
else {
delay(300);
  }
if (rpm_var > engine_on_rpm) {    // если обороты больше engine_on_rpm то считаем расход и расстояние (машина заведена)
   time_to_reconnect = 0;
   FuelSystemStatus();
   Rnd(); // вытягиваем информацию о нагрузка двигателя, если нету такго параметра в ЭБУ то закомментировать строчку символами //
   if (raschet_nagruzka_dvigatelya_var > 3)  { // Проверка торможения двигателем по нагрузке двигателя, если информации о нагрузке в ЭБУ нету то поменять на ((fss_val==2) or (speed_var>70))
    ShortTerm();
    IntakeAirTemp();
    //-----------------------------------------------------
    //если нет данных от MAF (нету такго параметра в ЭБУ) , то раскоментировать все что ниже до строчки
    //
    //  Maf();
    //
    //-------↓-↓-↓-↓-↓-↓-↓-↓-↓-↓-↓-↓-↓-↓-↓-↓-↓-↓-↓-↓-↓-↓-↓-↓-↓----------------------------
    //     DavlenVpuskKoll();
//     if (rpm_var < 800) {
//       VE = 64;
//     }
    //     else {
    //       if (rpm_var < 1001) {
    //         VE = 66;
    //       }
    //       else {
    //         if (rpm_var < 1251) {
    //           VE = 70;
    //         }
    //         else {
    //           if (rpm_var < 1501) {
//             VE = 70;
//           }
//           else {
```

Продолжение ПРИЛОЖЕНИЯ Б

```
//            if (rpm_var< 1751) {
//              VE = 74;
//            }
    //              else {
    //               if (rpm_var < 2001) {
    //                VE = 74;
    //               }
    //               else {
    //                if (rpm_var < 2126) {
    //                 VE = 76;
    //                }
    //                else {
    //                 if (rpm_var < 2251) {
    //                  VE = 77;
    //                 }
    //                 else {
    //                  if (rpm_var < 2376) {
    //                   VE = 79;
    //                  }
    //                  else {
    //                   if (rpm_var < 2501) {
    //                    VE = 80;
    //                   }
    //                   else {
    //                    if (rpm_var < 2626) {
    //                     VE = 82;
    //                    }
    //                    else {
    //                     if (rpm_var < 2751) {
    //                      VE = 82;
    //                     }
    //                     else {
    //                      if (rpm_var < 2876) {
    //                       VE = 83;
    //                      }
    //                      else {
    //                       if (rpm_var < 3001) {
    //                        VE = 84;
    //                       }
    //                       else {
    //                        if (rpm_var < 3501) {
    //                         VE = 85;
    //                        }
    //                        else {
    //                         if (rpm_var < 4001) {
    //                          VE = 86;
    //                         }
    //                         else {
    //                          if (rpm_var < 4501) {
    //                           VE = 88;
    //                          }
    //                          else {
    //                           if (rpm_var < 5001) {
    //                            VE = 90;
    //                           }
    //                           else {
    //                            if (rpm_var < 5501) {
//                              VE = 93;
```

```
        //                          }
      //                        else {
      //                          if (rpm_var< 6001) {
      //                            VE = 95;
      //                          }
        //                            else {
        //                              if (rpm_var < 6501) {
        //                                VE = 91;
        //                              }
        //                              else {
        //                                if (rpm_var < 7000) {
        //                                  VE = 85;
        //                                }
        //                              }
        //                            }
        //                          }
        //                        }
        //                      }
        //                    }
        //                  }
        //                }
        //              }
        //            }
        //          }
        //        }
        //      }
        //    }
        //  }
        //  IMAP = double(rpm_var * dvk_var) / double(intake_air_temp_var + 273.15);
        //  MAF = double(IMAP / 120.0) * double(double(VE * VE_correct) / 100.0) * ED * 28.9644 / 8.314472;
      //------------------------------------------
        //  если нет данных от MAF (нету такго параметра в ЭБУ) , то закоментировать следущую строчку
        Maf();
        MAF = maf_var;
        if (fss_val == 2) { // если замкнутая обратная связь  - Closed Loop
      ls_term_val = double(100.0 + (long_term_val + short_term_val)) / 100.0; // коэффициенткорректировкирасходапо ShortTerm и LongTerm
        }
      else {
         ls_term_val = double(100.0 + long_term_val) / 100.0; // коэффициенткорректировкирасходапо Long-Term
        }
        FuelFlowGramsPerSecond = double(MAF / AirFuelRatio) * ls_term_val; // Получаемрасходграммбензинавсекундувсоотношении 14,7 воздуха/к 1 литрабензина, корректировка ls_term_val
        FuelFlowLitersPerSecond = FuelFlowGramsPerSecond / FuelDensityGramsPerLiter; // Переводимграммыбензинавлитры
      LPH = FuelFlowLitersPerSecond * 3600.0;      // Ковертированиелитроввчас
      }
       Speed(); // получить скорость
      if (time_old == 0) {
        time_old = millis();  // выполнится один раз при появлении оборотов
       }
```

Продолжение ПРИЛОЖЕНИЯ Б

```
  time_new = millis();   // время со старта программы в мс
   time = (double(time_new - time_old) / 1000.0) * tcorrect; // прошло время с последнего расчета скорости, расхода  - в сек
if (time > 10) {
   time = 0;
  }
  time_old = time_new;  // записать новое время для сравнения в следующем цикле
if (speed_var > 0) {
   odometr_add = double((double(speed_var * 1000.0) / 3600.0) * time) / 1000.0;
odometr = odometr + odometr_add; //общийпробегвкм
  }
  benz_add = FuelFlowLitersPerSecond * time;
  benz_potracheno = benz_potracheno + benz_add; // общийрасходвлитрах
if (((speed_var > 1) and (speed_var < 10) and ((time_new - time_old_gurnal) > 30000)) or ((speed_var == 0) and ((time_new - time_old_gurnal) > 10000))) {
double odometr_eeprom = EEPROM_float_read(111) + odometr_add_gurnal + odometr_add;
double benz_eeprom = EEPROM_float_read(122) + benz_add_gurnal + benz_add;
EEPROM_float_write(111,  (odometr_eeprom)); // записываем в энергонезависимую память журнала расстояние каждые 5 секунд... в памяти занимаются ячейкм 111,112,113,114
EEPROM_float_write(122, (benz_eeprom));  // записываембензин
   odometr_add_gurnal = 0;
   benz_add_gurnal = 0;
   time_old_gurnal = time_new;
  }
else {
   odometr_add_gurnal = odometr_add_gurnal + odometr_add;
   benz_add_gurnal = benz_add_gurnal + benz_add;
  }
 }
if (odometr > 0) {
  LP100 = (benz_potracheno / odometr) * 100.0;                    //расходбензинана 100 км (влитрах)
 }
double full_odometr_gurnal = EEPROM_float_read(111);
double full_benz_gurnal = EEPROM_float_read(122);
switch (selmon) {
case 0:
lcd.clear();
lcd.setCursor(0, 0);
lcd.print(speed_var); // Скорость
lcd.print(" km/h");
lcd.setCursor(12, 0);
lcd.print(int(rpm_var)); //Обороты
lcd.print(" rpm");
lcd.setCursor(0, 1);
lcd.print(LPH); // л/час
lcd.print(" L/h");
lcd.setCursor(12, 1);
if (voltage == 0) {
lcd.print("-"); //Напряжение
lcd.print("-");
lcd.print(".");
lcd.print("-");
   }
else {
lcd.print(voltage); //Напряжение
   }
lcd.print(" V");
lcd.setCursor(0, 2);
```

Продолжение ПРИЛОЖЕНИЯ Б

```
if (odometr > 0.1) { // отображать расход на 100 км только после 100 метров пробега
lcd.print(LP100);
     }
else {
lcd.print("-.--");
     }
lcd.print(" L/100");
lcd.setCursor(12, 2);
lcd.print(tmp_var);  //Температура
lcd.print(" \1");
lcd.print("C");
lcd.setCursor(0, 3);
lcd.print(odometr); // пройденныйпутьсзаводкиавто
lcd.print(" km");
lcd.setCursor(12, 3);
lcd.print(benz_potracheno); // бензинапотраченосзаводкиавто
lcd.print(" L");
break;
case 1:
value = digitalRead(pin); // проверка состояния кнопки для переключения экрана
value2 = digitalRead(pin2);
if ((value == LOW) and (value2 == LOW)) {
float odometr0 = 0.00, benz0 = 0.00;
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("  Ochistka jurnala! ");
lcd.setCursor(0, 1);
lcd.print("--------------------");
lcd.setCursor(0, 2);
lcd.print(" Gdi 3 secundy dlya ");
lcd.setCursor(0, 3);
lcd.print("     ochistki..     ");
       delay(3000);
       lcd.clear();
       value = digitalRead(pin); // проверка состояния кнопки для переключения экрана
value2 = digitalRead(pin2);
if ((value == LOW) and (value2 == LOW)) {
        EEPROM_float_write(111, odometr0);
        EEPROM_float_write(122, benz0);
lcd.setCursor(0, 1);
lcd.print("|----[Clear OK]----|");
delay(2000);
       }
else {
lcd.setCursor(0, 1);
lcd.print("|----[Otmeneno]----|");
delay(2000);
       }
     }
lcd.clear();
lcd.setCursor(0, 0);
lcd.print(full_odometr_gurnal); // Полноерасстояние
lcd.print(" km");
lcd.setCursor(12, 0);
lcd.print(full_benz_gurnal); // Весьпотраченныйбензин
lcd.print(" L");
if (full_odometr_gurnal > 0) {
lcd.setCursor(0, 1);
```

```
lcd.print((full_benz_gurnal / full_odometr_gurnal) * 100.0); // Полныйсреднийрасход
lcd.print(" L/100");
    }
break;
case 2:
lcd.setCursor(0, 0);
lcd.print(selmon);
lcd.print(".");
lcd.print("Nagruzka dvigatela");
lcd.setCursor(0, 3);
lcd.print(raschet_nagruzka_dvigatelya_var);
lcd.print(" % ");
break;
case 3:
Tmp();
lcd.setCursor(0, 0);
lcd.print(selmon);
lcd.print(".");
lcd.print("Temperatura");
lcd.setCursor(0, 1);
lcd.print("ohlazhdaiushchej");
lcd.setCursor(0, 2);
lcd.print("zhidkosti");
lcd.setCursor(0, 3);
lcd.print(tmp_var);
lcd.print(" \1");
lcd.print("C  ");
break;
case 4:
Pdz();
lcd.setCursor(0, 0);
lcd.print(selmon);
lcd.print(".");
lcd.print("Polozhenie");
lcd.setCursor(0, 1);
lcd.print("drosselnoj zaslonki");
lcd.setCursor(0, 3);
lcd.print(polozh_dross_zaslon);
lcd.print(" %  ");
break;
case 5:
if (rpm_var < engine_on_rpm) {
DavlenVpuskKoll();
    }
lcd.setCursor(0, 0);
lcd.print(selmon);
lcd.print(".");
lcd.print("Davlenie");
lcd.setCursor(0, 1);
lcd.print("vpusknogo kollektora");
lcd.setCursor(0, 3);
lcd.print(dvk_var);
lcd.print(" kPa  ");
break;
case 6:
lcd.setCursor(0, 0);
lcd.print(selmon);
lcd.print(".");
```

```
lcd.print("Obototy dvigatelya");
lcd.setCursor(0, 3);
lcd.print(rpm_var);
lcd.print(" rpm   ");
break;
case 7:
lcd.setCursor(0, 0);
lcd.print(selmon);
lcd.print(".");
lcd.print("Skorost avto");
lcd.setCursor(0, 3);
lcd.print(speed_var);
lcd.print(" km/h  ");
break;
case 8:
Uoz();
lcd.setCursor(0, 0);
lcd.print(selmon);
lcd.print(".");
lcd.print("UOZ");
lcd.setCursor(0, 3);
lcd.print(uoz_var);
lcd.print(" \1  ");
break;
case 9:
if (rpm_var < engine_on_rpm) {
IntakeAirTemp();
     }
lcd.setCursor(0, 0);
lcd.print(selmon);
lcd.print(".");
lcd.print("Temperatura");
lcd.setCursor(0, 1);
lcd.print("vsasyvaemogo vozduha");
lcd.setCursor(0, 3);
lcd.print(intake_air_temp_var);
lcd.print(" \1");
lcd.print("C  ");
break;
case 10:
Voltage();
lcd.setCursor(0, 0);
lcd.print(selmon);
lcd.print(".");
lcd.print("Napryazhenie");
lcd.setCursor(0, 3);
if (voltage == 0) {
lcd.print("-"); //Напряжение
lcd.print("-");
lcd.print(".");
lcd.print("-");
     }
else {
lcd.print(voltage); //Напряжение
     }
lcd.print(" V   ");
break;
case 11:
```

Продолжение ПРИЛОЖЕНИЯ Б

```
      //    B1S1();
//    B1S2();
lcd.setCursor(0, 0);
lcd.print(selmon);
lcd.print(".");
lcd.print("Oxygen B1S1:");
lcd.setCursor(0, 1);
lcd.print(b1s1_val);
lcd.print(" V    ");
lcd.setCursor(0, 2);
lcd.print("Oxygen B1S2:");
lcd.setCursor(0, 3);
lcd.print(b1s2_val);
lcd.print(" V    ");
break;
case 12:
lcd.setCursor(0, 0);
lcd.print(selmon);
lcd.print(".");
lcd.print("Short fuel trim");
lcd.setCursor(0, 1);
lcd.print(short_term_val);
lcd.print(" %    ");
lcd.setCursor(0, 2);
lcd.print("Long fuel trim");
lcd.setCursor(0, 3);
lcd.print(long_term_val);
lcd.print(" %    ");
break;
case 13:
Dts();
lcd.setCursor(0, 0);
lcd.print(selmon);
lcd.print(".");
lcd.print("Davlenie topliva");
lcd.setCursor(0, 3);
lcd.print(davlenie_topliva_var);
lcd.print(" kPa  ");
break;
case 14:
Uroventopliva();
lcd.setCursor(0, 0);
lcd.print(selmon);
lcd.print(".");
lcd.print("Uroven topliva");
lcd.setCursor(0, 3);
lcd.print(uroven_topliva_var);
lcd.print(" %  ");
break;
case 15:
Maf();
lcd.setCursor(0, 0);
lcd.print(selmon);
lcd.print(".");
lcd.print("Massovyj rasshod");
lcd.setCursor(0, 1);
lcd.print("vozduha");
lcd.setCursor(0, 3);
```

```
lcd.print(maf_var);
lcd.print(" gramm/sec  ");
break;
case 16:
Tempmasla();
lcd.setCursor(0, 0);
lcd.print(selmon);
lcd.print(".");
lcd.print("Temperatura masla");
lcd.setCursor(0, 1);
lcd.print("dvigatelya");
lcd.setCursor(0, 3);
lcd.print(tmp_masla_var);
lcd.print(" \1");
lcd.print("C  ");
break;
case 17:
if (rpm_var < engine_on_rpm) {
FuelSystemStatus();
     }
lcd.setCursor(0, 0);
lcd.print(selmon);
lcd.print(".");
lcd.print("FuelSystemStatus:");
lcd.setCursor(0, 3);
      lcd.print(fss_val); // если форсунки отключены то 1, если работают то 2
      break;
    case 18:   // экран технологический 1
      lcd.clear();
lcd.setCursor(0, 0);
lcd.print("IMAP:");
lcd.print(IMAP);
lcd.setCursor(12, 0);
lcd.print("VE: ");
lcd.print(VE);
lcd.setCursor(0, 1);
lcd.print("MAF: ");
lcd.print(MAF);
lcd.setCursor(0, 2);
lcd.print("Benzin: ");
lcd.print(String(benz_potracheno, 9));
lcd.setCursor(0, 3);
lcd.print("AddMgn: ");
lcd.print(String(benz_add, 9));
break;
    case 19:  // экран технологический 2
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("time_start: ");
lcd.print(millis() / 1000);
lcd.setCursor(0, 1);
lcd.print("LS_kor: ");
lcd.print(String(ls_term_val, 6));
lcd.setCursor(0, 2);
lcd.print("FSS: ");
lcd.print(fss_val);
lcd.setCursor(8, 2);
lcd.print("Load: ");
```

```
lcd.print(raschet_nagruzka_dvigatelya_var);
lcd.setCursor(0, 3);
lcd.print("time_loop: ");
lcd.print(time);
break;
case 20:
CheckEngineOchistka();
CheckEngineKm();
lcd.clear();
value = digitalRead(pin);
     value2 = digitalRead(pin2);
if ((value == LOW) and (value2 == LOW)) { // проверкасостояниякнопкидляочисткиошибок
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("  Ochistka oshibok! ");
lcd.setCursor(0, 1);
lcd.print("--------------------");
lcd.setCursor(0, 2);
lcd.print(" Gdi 4 secundy dlya ");
lcd.setCursor(0, 3);
lcd.print("     ochistki..     ");
delay(4000);
lcd.clear();
value = digitalRead(pin); // проверкасостояниякнопкидляочисткиошибок
value2 = digitalRead(pin2);
if ((value == LOW) and (value2 == LOW)) {
obd.clearDTC();// очисткаошибок
lcd.setCursor(0, 1);
lcd.print("|----[Clear OK]----|");
delay(2000);
       }
else {
lcd.setCursor(0, 1);
lcd.print("|----[Otmeneno]----|");
delay(2000);
       }
     }
lcd.setCursor(0, 0);
lcd.print("S ochistki oshibok: ");
lcd.setCursor(0, 1);
lcd.print(check_engine_km);
lcd.print(" km    ");
lcd.setCursor(0, 2);
lcd.print("Check Engine gorit: ");
lcd.setCursor(0, 3);
lcd.print(check_engine_flash);
lcd.print(" km    ");
break;
  }
}
void Speed(void) {
obd.readPID(PID_SPEED, speed_var);
}
void Tmp(void) {
obd.readPID(PID_COOLANT_TEMP, tmp_var);
}
void Rpm(void) {
obd.readPID(PID_RPM, rpm_var);
```

```
}
void DavlenVpuskKoll(void) {
obd.readPID(PID_INTAKE_MAP, dvk_var);
}
void IntakeAirTemp(void) {
obd.readPID(PID_INTAKE_TEMP, intake_air_temp_var);
}
void Voltage(void) {
voltage = obd.getVoltage();
}
void Rnd(void) {
obd.readPID(PID_ENGINE_LOAD, raschet_nagruzka_dvigatelya_var);
}
void Dts(void) {
obd.readPID(PID_FUEL_PRESSURE, davlenie_topliva_var);
}
void Uoz(void) {
obd.readPID(PID_TIMING_ADVANCE, uoz_var);
}
void Maf(void) {
obd.readPID(PID_MAF_FLOW, maf_var);
}
void Pdz(void) {
obd.readPID(PID_THROTTLE, polozh_dross_zaslon);
}
void Uroventopliva(void) {
obd.readPID(PID_FUEL_LEVEL, uroven_topliva_var);
}
void Tempmasla(void) {
obd.readPID(PID_ENGINE_OIL_TEMP, tmp_masla_var);
}
void CheckEngineKm(void) {
obd.readPID(PID_DISTANCE, check_engine_km);
}
void CheckEngineOchistka(void) {
obd.readPID(PID_DISTANCE_WITH_MIL, check_engine_flash);
}
void ShortTerm(void) {
obd.readPID(PID_SHORT_TERM_FUEL_TRIM_1, short_term_val);
}
void LongTerm(void) {
obd.readPID(PID_LONG_TERM_FUEL_TRIM_1, long_term_val);
}
//void SearchErrorEngine(void) {
//}
//void B1S1(void) {
//}
//void B1S2(void) {
//}
void FuelSystemStatus(void) {
obd.readPID(PID_FUEL_SYSTEM_STATUS, fss_val);
}
void EEPROM_float_write(int addr, float val) // записьвЕЕПРОМ
{
byte *x = (byte *)&val;
for (byte i = 0; i < 4; i++) EEPROM.write(i + addr, x[i]);
}
float EEPROM_float_read(int addr) // чтениеизЕЕПРОМ
```

ОкончаниеПРИЛОЖЕНИЯБ

```
{
bytex[4];
for (byte i = 0; i < 4; i++) x[i] = EEPROM.read(i + addr);
float *y = (float *)&x;
return y[0];
}
```