МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт Математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

02.03.03 Математическое обеспечение и администрирование информационных систем
(код и наименование направления подготовки, специальности)

Мобильные и сетевые технологии
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Разработка инструментария для мониторинга и анализа трафика CAN шины»

Студент К.В. Мизгирев
(И.О. Фамилия) (личная подпись)

Руководитель С.В. Митин
(Ученая степень, звание, И.О. Фамилия)

Консультант М.В. Дайнеко
(Ученая степень, звание, И.О. Фамилия)

Тольятти 2021

Аннотация

Тема выпускной квалифицированной работы: «Разработка инструментария для мониторинга и анализа трафика CAN шины».

Целью бакалаврской работы является разработка программно-аппаратного комплекса для анализа трафика.

Объектом бакалаврской работы является изучение методов работы с базой данных DBC, разработка графических интерфейсов, построение графиков качества работы за счёт статических гипотез.

Предметом бакалаврской работы является взаимодействия с CAN-шиной для анализа данных и выявления ошибок в работе датчиков.

Актуальность данной работы заключается в создание инструмента для отладки, тестирования и диагностики устройств, подключённых к CAN сети автомобиля.

В первой главе рассматривается программно-аппаратное содержание устройства. Проводится сравнительный анализ аналогичных устройств, на основе ПО которых ставятся задачи на разработку.

Вторая глава содержит этапы проектирования интерфейса, ПО и технического устройства. Благодаря данному этапу решается ряд проблем, с которыми можно столкнуться в процессе реализации программно-аппаратного комплекса.

В третьей главе описывается процесс разработки данной системы и всех сопутствующих элементов, которые взаимодействуют с ней.

В заключении говорится о заключительных этапах работы с над проектом и говорится о преимуществах и недостатках аппаратно-программного комплекса и рассматривается его будущее развитие.

В итоге было получено средство российской разработки, которое обладает удобным интерфейсом и предоставляет пользователю гибкие возможности для использования и возможность применения как во время разработки и отладки какого-либо оборудования, так и вовремя производства для отслеживания качества продукции.

ABSTRACT

The topic of the present graduation work is *Development of tools for monitoring and analyzing CAN bus traffic.*

The purpose of the research is to develop software and hardware complex to carry out a traffic analysis.

The object of the graduation work is the methods of working with the DBC database, developing the graphic interface and constructing the work quality graphs by means of statistic hypotheses.

The subject of the graduation work is the interaction with the CAN bus to conduct a data analysis and the detection of errors in the sensors' operation.

The relevance of the investigation consists in creating a tool for debugging, testing and diagnosing different devices connected to the car's CAN network.

The first chapter of the research dwells on the software and hardware content of the device. A comparative analysis of the similar devices, on the basis of which development tasks are set, is carried out as well.

The second chapter of the investigation reveals the stages of designing the interface as well as software and a technical device. Thanks to this stage, a number of problems that a person can face with when implementing the software and hardware complex are solved.

The third chapter of the investigation describes the development of this system and all the related elements that interact with it.

In the conclusion, we discuss the advantages and disadvantages of the hardware and software complex and consider its future development.

The result is a Russian development tool that has a user-friendly interface and provides the user with the flexibility to use it both during the development and debugging of a piece of equipment and during its production to monitor and assess the product quality.

Содержание

Введение

Наш проект представляет из себя программно-аппаратный комплекс, реализующий взаимодействие персонального компьютера и электронных систем автомобиля по средствам CAN-шины. Устройство реализовано на базе микроконтроллера STM32F4. Поскольку он обладает реализацией аппаратного CAN интерфейса и USB интерфейсом, которые необходимы для работы нашего устройства.

Данное устройство необходимо для изменения настроек различных систем автомобиля, оценки работы его датчиков, проведение тестов работы различных систем автомобиля, таких как датчики освещённости, системы управления фарами, антиблокировочной системы, управление климат контролем и многое другое.

Для удобства работы с нашим устройством разработано внутреннее API для передачи и получения данных от компьютера в CAN- шину, что дало нам возможность унифицировать запросы к тем или иным устройствам. Далее было разработано программное обеспечение для ПК. Оно даёт возможность удобно и быстро взаимодействовать с нашим устройством. Реализация данного ПО была проведена в QT с использованием их графических библиотек.

Актуальность данной работы заключается в создание инструмента для отладки, тестирования и диагностики устройств, подключённых к CAN сети автомобиля. Импортные аналоги имеют достаточно высокую стоимость ПО и лицензии, например, такие как CANalyzer от компании Vector.

Объектом исследования использование статистических методов для выделения событий на CAN шине.

Предметом исследования являлись отладочная плата STM32F4-discovery, которая обладает аппаратной реализацией CAN – интерфейса.

Разработка ПО верхнему уровню для анализа трафика шины и формирования передачи тестовых кадров в шину.

Для достижения поставленной цели нужно решить следующие задачи:

1. Проанализировать всю научно-методическую литературу, которую необходимо классифицировать.
2. Проанализировать уже существующие варианты реализации подобных устройств.
3. Выбрать инструменты разработки и метод реализации.
4. Разработать программный код для устройства.
5. Протестировать корректность работы устройства.
6. Основываясь на полученных данных сделать вывод об эффективности проделанной работы.

Практическая значимость работы заключается в разработке уникального аппаратного устройства и программ взаимодействия с ним.

1 Теоретическое обоснование задачи для реализации программно-аппаратного устройства

1.1 Описание входных элементов, используемых для разработки

В России такие устройства не производятся, а потребность в них большая, поскольку на данный момент разработчикам и производителям авто приходится закупать иностранные устройства или разрабатывать тесты под каждый новый автомобиль на низком уровне работы с CAN –шиной. Это приводит к влажнею излишних материальных средств и больших затрат времени. Именно поэтому, универсальное устройство будет востребовано отечественными производителями и разработчиками. если сравнивать стоимость нашего продукта и импортных аналогов, то разница цене составит десятки раз. Обучение персонала для работы на нашем оборудовании будет гораздо быстрее чем на импортных аналогах, поскольку наш продукт изначально является русскоязычным.

В данном проекте была реализована работа с CAN-шиной. Рассмотрим её архитектуру и принцип её работы.

CAN-шина представляет собой два параллельных проводника, которые связаны между собой двумя нагрузками так называемыми «нагрузка А» и «нагрузка В». Первый проводник называется CAN_H а второй CAN_L, как показано на рисунке 1.

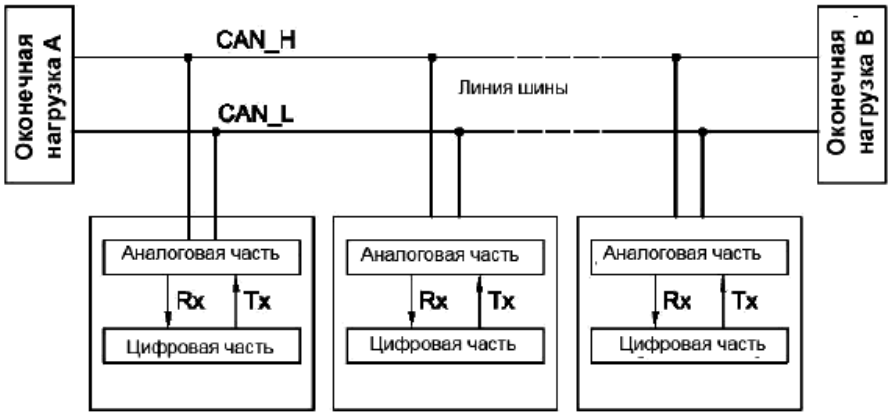
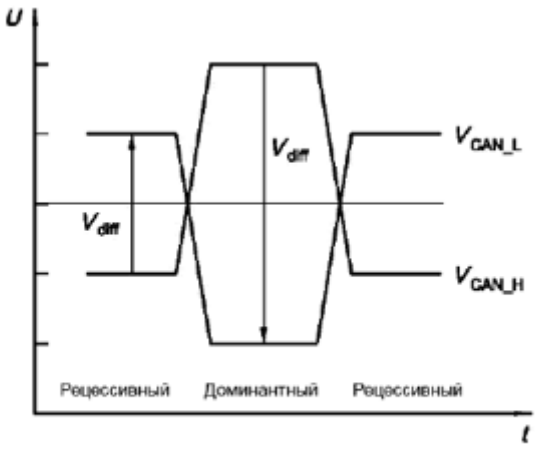


Рисунок 1-Рекомендуемое электрическое подключение

Используется дифференциальный сигнал для передачи данных это означает что разница между CAN_H и CAN_L заключается в том, что CAN_H сигнал идёт за счет повышения напряжения, а в CAN_L он зеркально дублируется, то есть напряжение уменьшается.

Среднее напряжение в обоих проводниках составляет 2,5В, то есть, в проводнике CAN_H в момент передачи «1» напряжение возрастает на 1В, то есть до 3,5В, в этот же момент напряжение в проводнике CAN_L опускается до 1,5В. На рисунке 2 представлено физическое представление битов. Передающее устройство подключается параллельно. Таким образом, всё вышеописанное является физическим описанием модели ISO.



U - уровень напряжения; t - время

Рисунок 2- Физическое представление битов

Рассмотрим канальные протоколы или CAN протоколы. Они используются для различных сред: воздух, проводники, и т.д.), так как он обеспечивает достаточную скорость и имеет низкую искажаемость сигнала. Скорость данного протокола на проводниках, аналогичной классической витой пары, длинной не превышающей 40 метров достигает одного 1 Мб/сек, а на линии длинной до 1км достигает 40 Кб\сек.

Передача данных в CAN-шину происходит покадрово. Каждый кадр состоит из набора определенных битов, которые описаны в таблице 1.

Таблица 1 – Стандартные кадры

Поле	Длина	Описание
Начало кадра (SOF)	1 бит	Начало передачи кадра
Идентификатор (ID)	11 бит	Идентификатор сообщения
Запрос на пересдачу (RTR)	1 бит	Доминантный бит
Бит расширения идентификатора (IDE)	1 бит	Бит определяет длину идентификатора, для базового формата – доминантный бит
Зарезервированный бит	1 бит	Зарезервировано
Длина данных (DLC)	4 бита	Количество байт данных
Данные	0-8 байт	Данные
Контрольная сумма (CRC)	15 бит	Контрольная сумма
Разграничитель контрольной суммы	1 бит	Рецессивный бит
Промежуток подтверждения (ACK)	1 бит	Для приёмника – доминантный бит, для передатчика - рецессивный
Разграничитель подтверждения	1 бит	Рецессивный бит
Конец кадра (EOF)	7 бит	Все биты рецессивные

Так же вариант расширенного кадра, которые влияет на повышенное качество передачи. Структура расширенного кадра представлена во 2 таблице.

Таблица 2 – Расширенные кадры

Поле	Длина	Описание
Начало кадра (SOF)	1 бит	Начало передачи кадра
Идентификатор A (ID A)	11 бит	Первая часть идентификатора
Идентификатор B (ID B)	18 бит	Вторая часть идентификатора
Подмена запроса на передачу (SRR)	1 бит	Рецессивный бит
Запрос на пересдачу (RTR)	1 бит	Доминантный бит
Бит расширения идентификатора (IDE)	1 бит	Бит определяет длину идентификатора, для базового формата – доминантный бит
Зарезервированный бит	1 бит	Зарезервировано
Длина данных (DLC)	4 бита	Количество байт данных
Данные	0-8 байт	Данные
Контрольная сумма (CRC)	15 бит	Контрольная сумма
Разграничитель контрольной суммы	1 бит	Рецессивный бит
Промежуток подтверждения (ACK)	1 бит	Для приёмника – доминантный бит, для передатчика - рецессивный
Разграничитель подтверждения	1 бит	Рецессивный бит
Конец кадра (EOF)	7 бит	Все биты рецессивные

Все элементы данных являются стандартом передачи данных в CAN-шину. Рассмотрим сегменты синхронизации.

Сегмент синхронизации (SyncSeg) - это первый порядковый элемент, который используется для синхронизации узлов на шине. Начальный уровень перепада ожидается именно в этих узлах данного сегмента. Его фиксированная длительность 1 кадр.

Сегмент распространения (ProSeg) – это элемент, который компенсирует физические задержки сигнала между узлами. Длительность данного элемента варьируется от времени распространения сигнала от передающего узла до принимающего и обратно, включая задержки, связанные с драйвером шины. Она может принимать значения от 1 до 8 кадров.

Были рассмотрены свойства и необходимые параметры CAN-шины и CAN-интерфейса для взаимодействия с ним на низком уровне.

1.2 Анализ уже существующих устройств

«Kvaser USBcan Pro 2xHS v2» - это двухканальный интерфейс USB для CAN или CAN FD с возможностью создания сценариев. Благодаря стандартному разъему USB и двум высокоскоростным каналам CAN с совместимыми с ISO 11898-2 приемопередатчиками CAN в двух отдельных 9-контактных разъемах D-SUB CAN, он является высокопроизводительным, но компактным и может использоваться в качестве простого двухканального интерфейса для подключения двух высокоскоростных шин CAN к ПК или мобильному компьютеру, или может быть запрограммирован на большее. На рисунке 3 представлено техническое устройство.

Рисунок 3-Kvaser USBcan Pro 2xHS v2

Данное техническое устройство имеет подобные конфигурации и возможности с разрабатываемым проектом. Является достаточно дорогостоящим, но удобным в процессе эксплуатации.

«USB can 2» - высокопроизводительный интерфейс CAN-USB. устройство объединяет двухканальный интерфейс шины CAN со стандартным интерфейсом USB. Один канал канал CAN предназначен для High-Speed CAN. Второй, опциональный, канал CAN может быть, как High-Speed, так и Low-Speed или Single-Wire CAN. Физическое устройство представлено на рисунке 4.

Рисунок 4– Устройство USB can 2

Данное устройство значительно дешевле своего предшественника, но обладает значительными недостатками, такими как на борту всего 1 CAN-интерфейс, который так же не обладает коррекцией ошибок, что является для него существенными недостатком, а также программное обеспечение данного устройства не является гибким, поскольку устройство не имеет встроенного API.

На основе сравнительного анализа, составлена таблица, включающая преимущества и недостатки вышеописанных устройств, на основе которых будут выстроены требования к новому программно-аппаратному комплексу.

Таблица 3 – Анализ существующих устройств

	Kvaser USBcan Pro 2xHS v2	Устройство USB can 2
Количество интерфейсов CAN	2	1
Выход CAN-F	+	-
Наличие API	+	-
Возможность работы с мобильным устройством	-	-
Стоимость	-	+

На основе проделанного сравнительного анализа можно сделать вывод, что каждое устройство уникально с точки зрения разработки, но обладает свои рядом недостатков, в зависимости от технической реализации. Таким образом, задача данного проекта создать новый уникальный программно-аппаратному комплекс, который бы сочетал в себе комфортную ценовую политику и широкий спектр охватываемых задач.

1.3 Формирование требований к новому программно-аппаратному устройству

Задача, представленная к разработке, это создание новой уникальной программно-аппаратной среды, которая может использоваться любым пользователем со средними навыками использования компьютера.

Благодаря данной среде управления устройств и базы данных будет решен ряд задач:

- комфортный интерфейс;

- скорость внедрения устройства на производство;

- возможность внедрения собственного ПО и изменения уже готового ПО;

- универсальность использования для любой технической среды, где имеется CAN-шина.

Таким образом, перед нами стоит уникальная задача к разработке нового программно-аппаратного комплекса, который является лучшей версией своих зарубежных аналогов.

Программное обеспечение было разработано в среде QtQreator. В данной среде очень удобно разрабатывать графические интерфейсы, также есть уже готовые библиотеки для взаимодействия с com – портом, что очень удобно для нашего проекта. Итак, давайте рассмотрим участки нашего кода. Первое, что нам необходимо рассмотреть, так это взаимодействие нашего устройства с программным обеспечением. Как уже было сказано выше, оно происходит за счет передачи команд или так называемого API нашего физического устройства. Таблица с командами с примерами их использования приведена ниже.

Таблица 4 – Таблица с командами API

Vendor	**Controller**	**Kconfig Option**	**Remarks**
VScom	SJA1000T	CONFIG_CAN_SLCAN	needs slcand
VScom	(unknown)	CONFIG_CAN_SLCAN	needs slcand and socat
TITAN	(unknown)	CONFIG_CAN_SLCAN	needs slcand
Softing	sja1000 or NEC-005(?)	CONFIG_CAN_SOFTING_CS	Supports {CAN,EDIC}card{,SJA,2} PCMCIA cards Needs firmware softing-fw-4.6-binaries.tar.gz
PEAK	sja1000	CONFIG_CAN_PEAK_PCI	supports all PCAN-[mini]PCI[e][104] variants
PEAK	(unknown)	CONFIG_CAN_PEAK_USB	supports PCAN-USB[pro][hub] (no LIN support)
Microchip	PIC18F2680	CONFIG_CAN_MCBA_USB	SocketCAN driver Improved firmware
LAWICEL	(unknown)	CONFIG_CAN_SLCAN	needs slcand
LAWICEL	(unknown)	CONFIG_CAN_SLCAN	needs slcand
Kvaser	sja1000	CONFIG_CAN_KVASER_PCI	supports all PCI-[mini]PCI[e][104] variants
Kvaser	(unknown)	CONFIG_CAN_KVASER_USB	supports USB Leaf / Memorator / Blackbird / R (see detailed USB device IDs in kvaser_usb.c)
Kvaser	(unknown)	CONFIG_CAN_KVASER_USB	supports: • Kvaser USBCan-II HS/LS • Kvaser USBCan-II HS/HS • Kvaser USBcan Rugged ("USBcan Rev B") • Kvaser Memorator HS/LS • Kvaser Memorator HS/HS • Scania VCI2 (if you have the Kvaser logo on top)

Как видно из таблицы, есть множество команд для передачи данных в наш микроконтроллер или из нашего микроконтроллера, а уже сам мк получает их из CAN-шины. За счет происходит взаимодействие С – шина – компьютер и компьютер-CAN-шина.

Также в нашем проекте требуется хранить и интерпретировать данные получаемые входе взаимодействия с CAN-шиной для этого применяется база-данных DBC которая была специально разработана для хранения и обработке кадров с CAN-шины, деваете же рассмотрим структуру этой базы данных, она представлена на Рисунке 5.

```
struct dbc_header
{
  uint32_t magic; // всегда 'WDBC'
  uint32_t record_count; // делает запись за файл
  uint32_t field_count; // область за отчет
  uint32_t record_size; // sum (sizeof (field_type_i)) | 0 <= i < field_count. field_type_i is НЕ определен в файлах.
  uint32_t string_block_size;
};

template<typename record_type>
struct dbc_file
{
  dbc_header header;
  // static_assert (header.record_size == sizeof (record_type));
  record_type records[header.record_count];
  char string_block[header.string_block_size];
};
```

Рисунок 5-структура DBC

Выводы по главе

В этой главе был рассмотрен принцип работы CAN-шины. Её архитектура является одной из самых надежных цифровых линий передачи данных, среди ныне существующих. Данное устройство обладает наибольшей гибкостью и достаточной скоростью большой длине. При этом обладает минимальной потерей данных на больших расстояниях и хорошей помехоустойчивостью сигнала, что позволяет использовать её в автомобилях, средствах противовоздушной обороны (ПВО), и даже применяется на шине данных для ракеты Falcon9.

Вторым этапом исследования был сравнительный анализ уже существующих систем. Были рассмотрены два зарубежным программно-аппаратных комплекса и рассмотрены их функциональные возможности, на основе которых были построены задачи на разработку данного проекта.

Уникальность данного комплекса, заключается в совокупности технических параметров, доступности компонентов и комфортной стоимости будущего устройства.

2 Проектирование системы тестирования электронных устройств на CAN-шине

2.1 Выбор необходимого для реализации проекта микроконтроллера

Для начала проектирования был проведен сравнительный анализ доступных микроконтроллеров на предмет необходимых внутренних аппаратных возможностей.

Для анализа были выбраны микроконтроллеры семейства AVR и STM, так как являются самыми доступными. На рисунке 6 представлен микроконтроллер семейства AVR AT32UC3C.

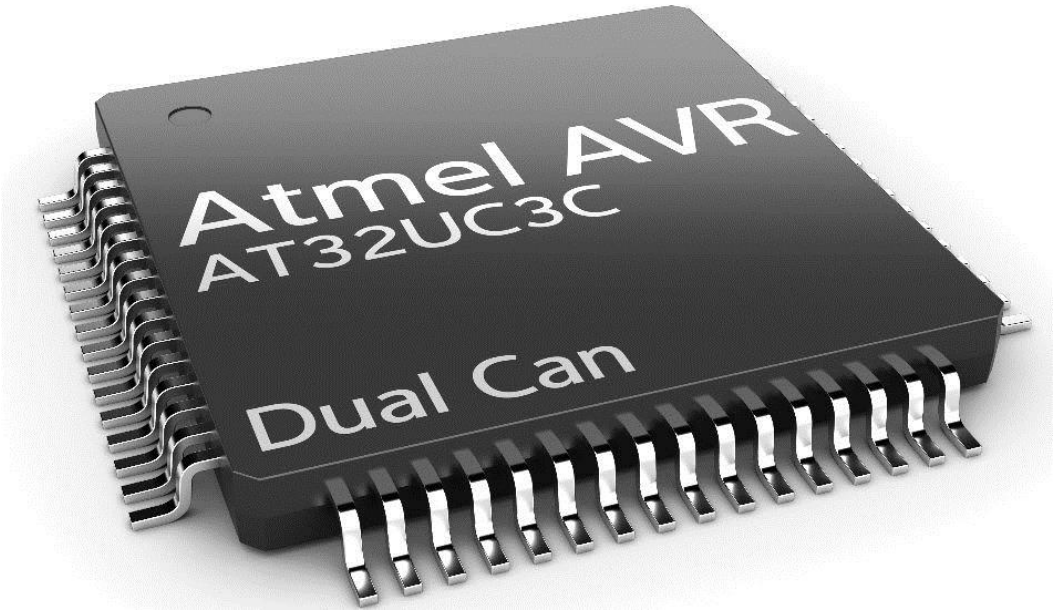


Рисунок 6- Микроконтроллер семейства AVR AT32UC3C

Данный микроконтроллер имеет ряд преимуществ в виде цены, надежности и обладает одним из самых больших запасов прочности по напряжению. Однако же, данное устройство имеет ряд недостатков, таких как низкая частота дискретизации аналогово цифрового преобразователя,

который необходим для чистой работы с CAN-шиной. Он является 8-ми разрядным, а это значит, что данное устройство не позволяет производить высокоточные математические вычисления.

Следующим устройством для анализа был выбран микроконтроллер STM32F407. Данный микроконтроллер представлен на рисунке 7.

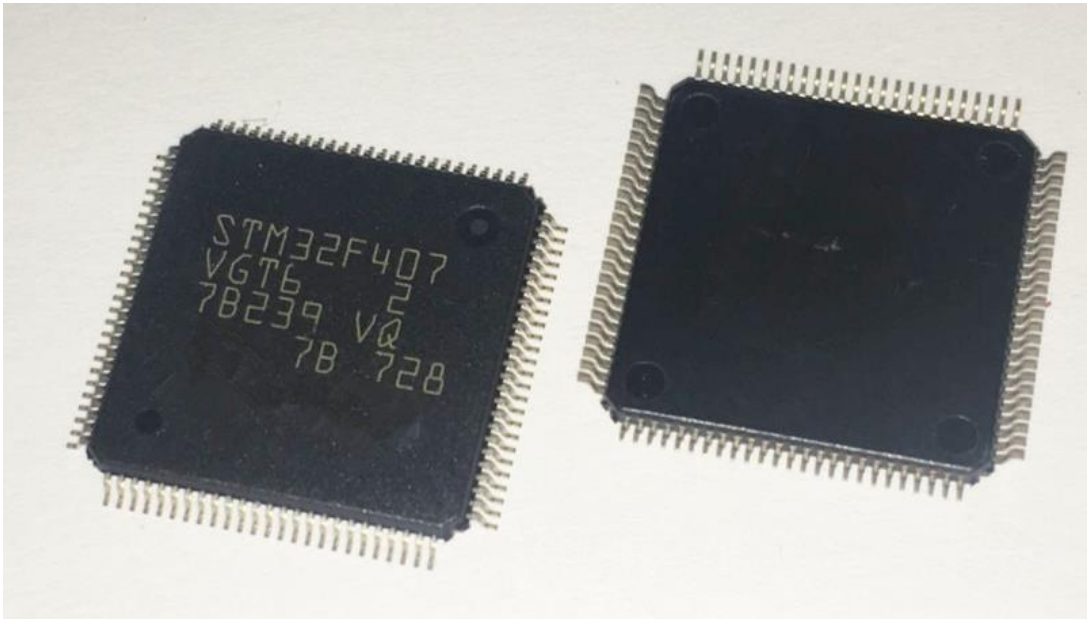


Рисунок 7-Микроконтроллер STM32F407

Данное устройство имеет ряд преимуществ над своим оппонентом, в том, что его ценовая политика значительно ниже, но его вычислительно ядро основано на ARM архитектуре и является 32-ух битным. есть встроенное FP что позволяет более эффективно решать математические задачи. Данная архитектура ядра позволяет быть наиболее эффективным в вычислениях чисел с плавающей запятой, также имеется 16-ти битный АЦП и интерфейс вывода USB 1.1. Микроконтроллер STM32F407 имеет у себя на борту необходимый для нашего проекта CAN-интерфейс, что позволяет напрямую подключаться к CAN-шине без дополнительной обвязки.

Данное устройство не отличается продолжительностью использования, так как имеет достаточно узкий промежуток эффективного рабочего

напряжения. Заливки прошивки проходят не всегда успешно, так как протокол UART не отличается высоким качеством, в связи с чем следует неудобство отладки.

Следующим анализируемым микроконтроллером стал ESP8266. Устройство представлено на рисунке 8.

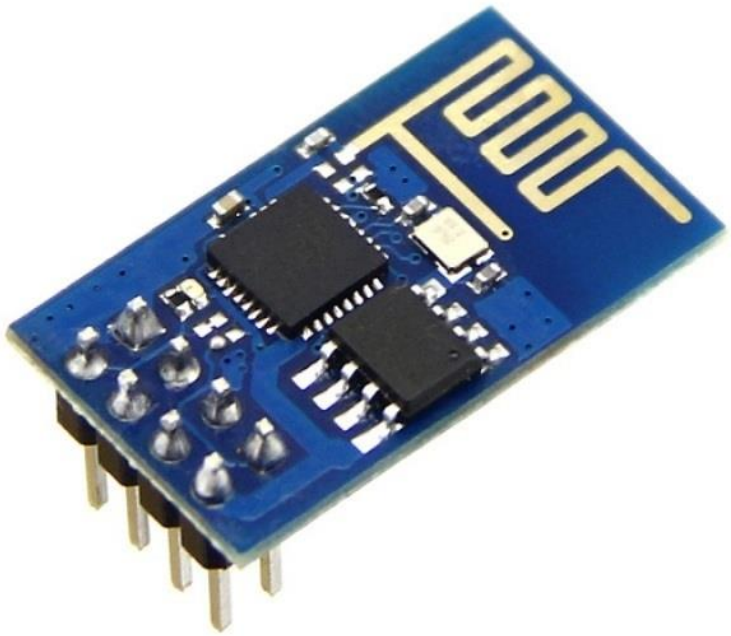


Рисунок 8-Микроконтроллером ESP8266

Данный микроконтроллер интересен тем, что обладает достаточно большим диапазоном питающего напряжения, обладает возможность работать по сети WI-FI, что являлось бы для нашего проекта преимуществом, но у данного контроллера есть несколько неоспоримых недостатков.

Во-первых, необходимо имитировать CAN-шину так как нет аппаратной реализации у данного микроконтроллера. Невозможность её программной реализации, поскольку этот микроконтроллер не обладает достаточной частотой развертки и обладает низкой тактовой частотой.

Во-вторых, нет необходимого для нашего проекта АЦП. Из-за этого на нём не будет возможности скорректировать ошибки при обрыве одной из приводящих линий, что приведёт к искажению сигнала и потере данных.

На данном этапе были проанализированы существующие микроконтроллеры, их архитектура и программная реализация. В следующей главе будут сформулированы задачи на реализацию.

2.2 Проектирование схемы устройства

Для начального моделирования использовалась программа Proteus 8.2, данная программа даёт возможность построения и моделирования электронных схем с микроконтроллерами, так и различных компания что поваляет отладить схему как с программной стороны, так и аппаратной составляющей.

Это позволяет на более качественно и в более зажатые сроки реализовать необходимые схемы или проверить их жизни способность

Интерфейс программы представлен на рисунке 9.

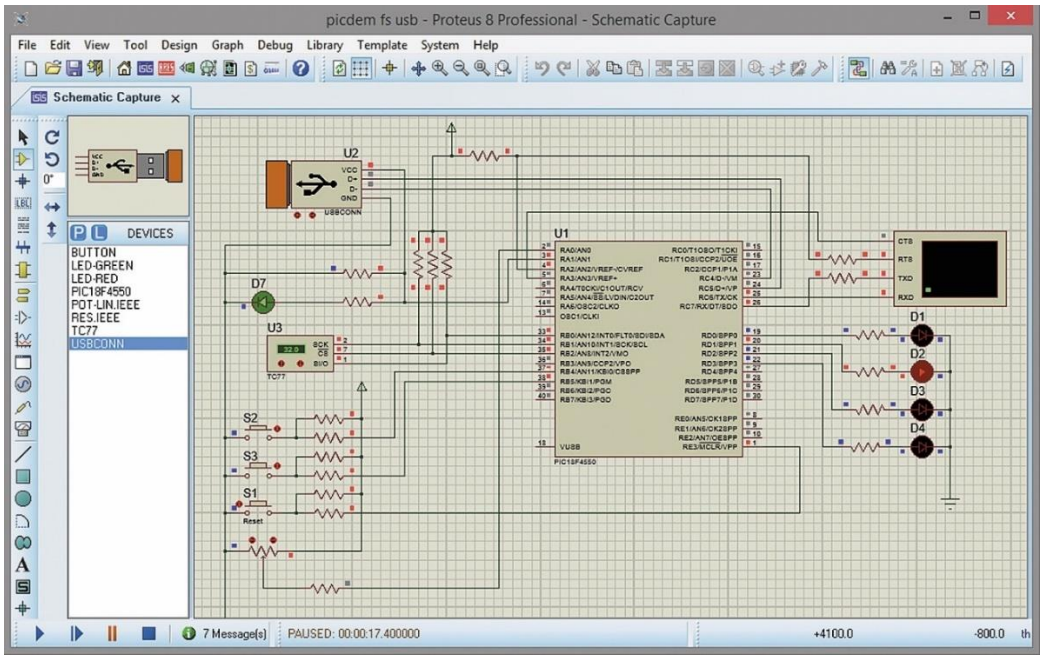


Рисунок 9 – Интерфейс программы Proteus 8.2

2.3 Проектировка макетных плат и разводка проводников в программе SprintLayout 6.0

Разработка макета в программе SprintLayout 6.0 представляет из себя расстановку необходимых компонентов по макету платы и соединении их проводниками на разных слоях, в нашем случае использовалось 2 слоя. Электронные компоненты были использованы в SMD корпусах, поскольку DIP корпуса использовать было бы неэффективно, так как занимаемое место на плате и требование к отверстиям достаточно большие, в то время, когда наше устройство должно быть компактным. Пример SMD и DIP корпусов представлены на рисунках 10,11.

Рисунок 10 – SMD корпус

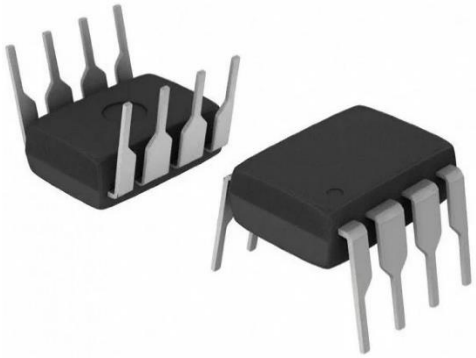


Рисунок 11– DIP корпус

Далее в программе SprintLayout 6.0, разместив все необходимые проводники (дорожки и контакты) на плате, проводим тестирование встроенными средствами программы. Далее необходимо извлечь Gerber файл и передать его в программу, которая преобразует данный код в управляющие команды для ЧПУ-фрезерного станка.

2.4 Исследование и проектирование основных программных модулей

Для использования CAN-интерфейса в STM32 требуется написание драйверов, которые будут последовательно обращаться к CAN-интерфейсу и шине памяти внутри самого устройства. Это необходимо для реализации API, которая будет взаимодействовать с интерфейсом USB и передавать данные с шины памяти на ПК. На рисунке 12 представлен фрагмент кода, опрашивающего CAN-интерфейс.

```
while(1)
{
  while(STM_EVAL_PBGetState(BUTTON_KEY) == KEY_PRESSED)
  {
    if(ubKeyNumber == 0x4)
    {
      ubKeyNumber = 0x00;
    }
    else
    {
      LED_Display(++ubKeyNumber);
      TxMessage.Data[0] = ubKeyNumber;
      CAN_Transmit(CANx, &TxMessage);
      /* Wait until one of the mailboxes is empty */
      while((CAN_GetFlagStatus(CANx, CAN_FLAG_RQCP0) !=RESET) || \
            (CAN_GetFlagStatus(CANx, CAN_FLAG_RQCP1) !=RESET) || \
            (CAN_GetFlagStatus(CANx, CAN_FLAG_RQCP2) !=RESET));

      while(STM_EVAL_PBGetState(BUTTON_KEY) != KEY_NOT_PRESSED)
      {
      }
    }
  }
}
```

Рисунок 12 - Фрагмент кода, опрашивающего CAN-интерфейс

2.5 Разработка программного обеспечения для взаимодействия с данным устройством

Для создания графического интерфейса программы была использована среда разработки qt creator. QT creator был выбран, так как обладает наиболее подходящими библиотеками для реализации проекта.

Для разработки данного ПО использовался язык программирования C++, поскольку обладает наибольшей эффективностью и скоростью работы.

Qt разрабатывается уже много лет, с конца 90-х. За это время его коммерческой версией уже успели наиграться такие компании, как Trolltech и Nokia, а сейчас этим занимается Digia. Основываясь на этих фактах, можно сделать вывод, что проект процветает. Еще несколько лет дизайн разрабатывался на виджетах (C++ классы, все до единого основанные на QWidget), а сегодня его может сделать и маленький ребенок.

Выводы по главе

Первым этапом разработки был проведен сравнительный анализ существующих микроконтроллеров. Были оценены достоинства и недостатки каждого устройства, на основе которых были выстроены задачи на разработку.

Следующим этапом, было проектирование схемы устройства. На данном этапе разработки была спроектирована эффективная архитектура схемы электроустройств и проведено программное тестирование устройства в программе Proteus 8.2

На основе спроектированной схемы была разработана схема для печатной платы, для этого применялась программа SprintLayout 6.0, которая является одной из самых эффективных программ CAD программой для разработки печатных плат.

Заключительным этапом проектирования было изучено использование CAN-интерфейса на микроконтроллере STM32 и начало разработки драйвером, основных компонентов API для передачи данных с CAN-шины на USB интерфейс, а в дальнейшем передачу в нашу программу.

3 Реализация

3.1 Реализация интерфейса программы

Как уже было сказано выше для разработки интерфейсов использовалась среда разработки QT creator рисунке 13, что даёт высокую эргономичность, а на рисунке 14 представлена разработка блока настройки подключаемого CAN интерфейса.

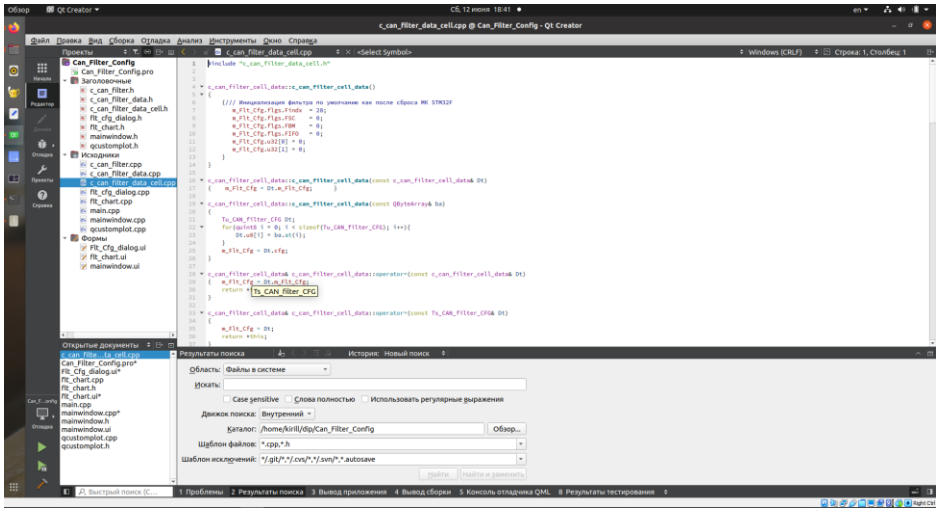


Рисунок 13 - QT creator

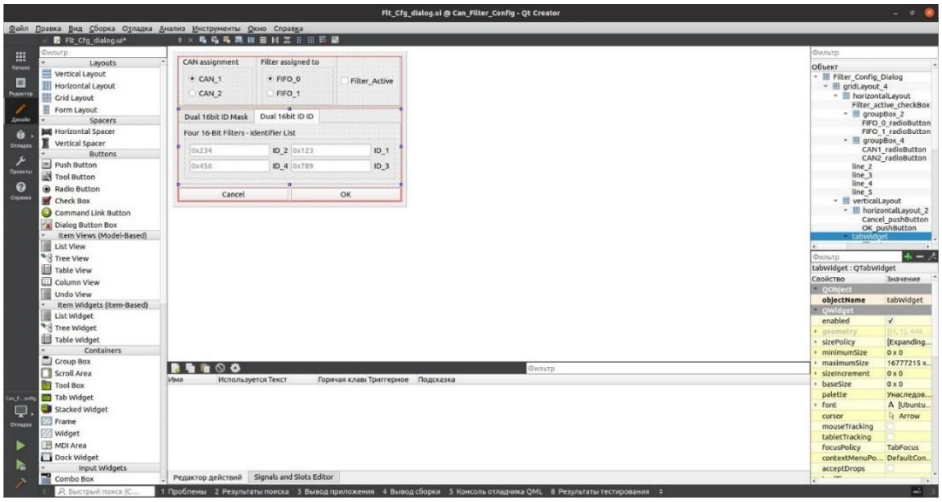


Рисунок 14 разработка интерфейса блок " Добавления CAN"

На рисунке 15 предлагается блок для выбора, уже добавленный интерфейсов для перехода на аналитический график или удаления интерфейса.

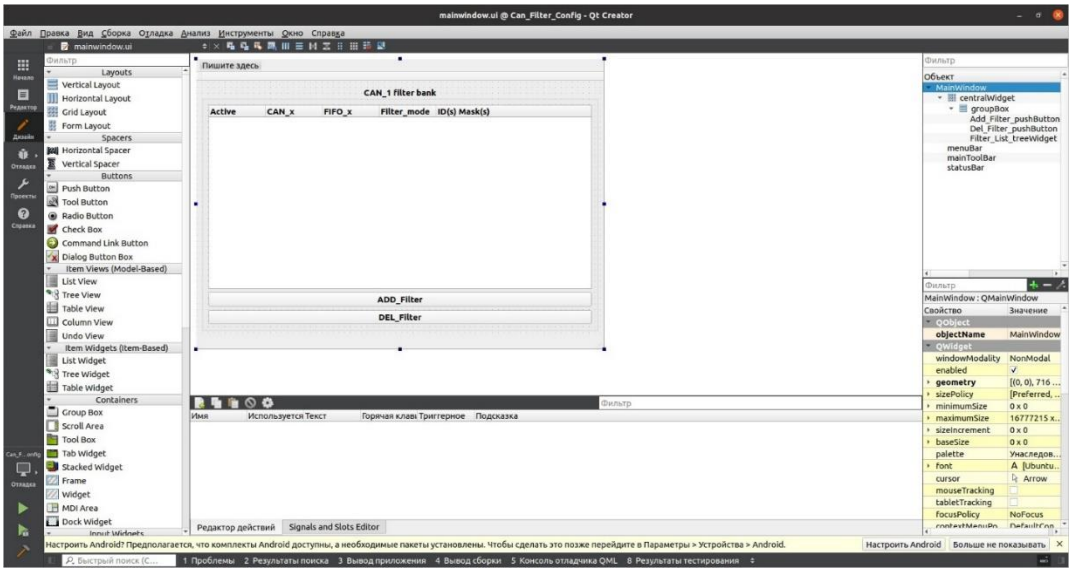


Рисунок 15 - главный блок программы

Для вывода графиков переменялась библиотека QCustomPlot на рисунке 16 представлен один из полученных графиков, а также рисунок 17 показывает работу с библиотекой.

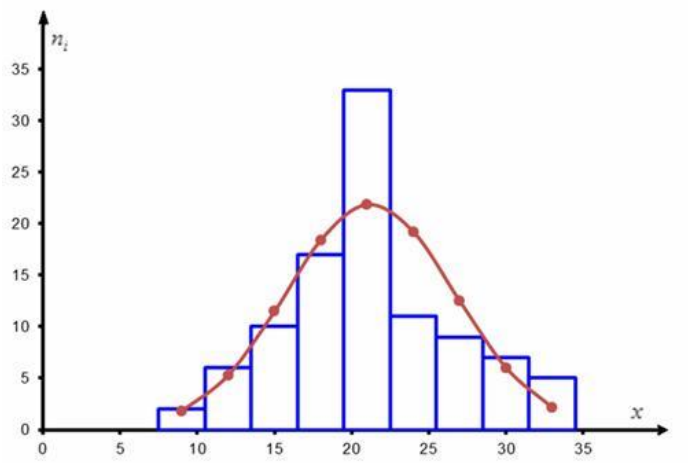


Рисунок 16 - график анализа гипотез

```
void MainWindow::drawfunc(int valT, int xmin, int xmax)
{
    double a = -1;
    double b = 1;
    int N = (b - a) / 0.01 + 2;

    int i = 0;
    for (double x = xmin; x < xmax; x += 0.01)
    {
        x11[i] = x;
        y11[i] = fn(valT, x);
        i++;
    }
    ui->widget->clearGraphs();
    ui->widget->addGraph();
    ui->widget->graph(0)->setData(x11, y11);

    ui->widget->xAxis->setLabel("x");
    ui->widget->yAxis->setLabel("y");

    ui->widget->xAxis->setRange(xmin, xmax);
    double minY = y11[0], maxY = y11[0];
    for (int i = 1; i < N; i++)
    {
        if (y11[i] < minY) minY = y11[i];
        if (y11[i] > maxY) maxY = y11[i];
    }
    ui->widget->yAxis->setRange(minY, maxY);
    ui->widget->replot();
}
```

Рисунок 17 - Листинг с представленной функцией для вывода графика

3.2 Реализация работы базы данных DBC

Взаимодействие с базой данных DBC происходит через подключение к структурированному файлу, который и является базой данных. Открытие базы данных представлено на рисунке 18. Здесь представлен код программы, данный код считывает данные из файла в три одномерных динамических массива для дольнейшего использования и или изменения.

```
void bdb_open(void)
{
    DB *db;
    DBC *dbc;
    DB_ENV *dbenv;

    assert(db_env_create(&dbenv, 0) == 0);
    dbenv->set_errpfx(dbenv, "bdb");
    dbenv->set_errfile(dbenv, stderr);
    assert(dbenv->mutex_set_max(dbenv, 10000) == 0);
    assert(dbenv->set_cachesize(dbenv, 0, 50 * 1024 * 1024, 1) == 0);
    assert(dbenv->open(dbenv, NULL,
        DB_CREATE |
        (g.c_delete_pct == 0 && g.c_insert_pct == 0 && g.c_write_pct == 0 ?
            0 : DB_INIT_LOCK) |
        DB_INIT_MPOOL | DB_PRIVATE, 0) == 0);
    assert(db_create(&db, dbenv, 0) == 0);

    if (g.type == ROW && g.c_reverse)
        assert(db->set_bt_compare(db, bdb_compare_reverse) == 0);

    assert(db->open(
        db, NULL, g.home_bdb, NULL, DB_BTREE, DB_CREATE, 0) == 0);
    g.bdb = db;
    assert(db->cursor(db, NULL, &dbc, 0) == 0);
    g.dbc = dbc;

    key_gen_setup(&keybuf);
}
```

Рисунок 18 - открытие базы данных

На рисунке 19 представлена функция, которая позволяет производить запросы в базу данных на основе ключа и получать данные в виде трёх массивов.

```
void bdb_read(uint64_t keyno, void *valuep, size_t *valuesizep, int *notfoundp)
{
    DBC *dbc = g.dbc;
    size_t size;
    int ret;

    key_gen(keybuf, &size, keyno, 0);
    key.data = keybuf;
    key.size = (uint32_t)size;

    *notfoundp = 0;
    if ((ret = dbc->get(dbc, &key, &value, DB_SET)) != 0) {
        if (ret != DB_NOTFOUND)
            die(ret, "dbc.get: DB_SET: {%.*s}",
            (int)key.size, (char *)key.data);
        *notfoundp = 1;
    }
    else {
        *(void **)valuep = value.data;
        *valuesizep = value.size;
    }
}
```

Рисунок 19 – Чтение данных по ключу

Для внесения изменений в базу данных была использована функция, которая представлена на рисунке 20.

```
bdb_update(const void *arg_key, size_t arg_key_size,
    const void *arg_value, size_t arg_value_size, int *notfoundp)
{
    DBC *dbc = g.dbc;
    int ret;

    key.data = (void *)arg_key;
    key.size = (uint32_t)arg_key_size;
    value.data = (void *)arg_value;
    value.size = (uint32_t)arg_value_size;

    *notfoundp = 0;
    if ((ret = dbc->put(dbc, &key, &value, DB_KEYFIRST)) != 0) {
        if (ret != DB_NOTFOUND) {
            die(ret, "dbc.put: DB_KEYFIRST: {%.*s}{%.*s}",
                (int)key.size, (char *)key.data,
                (int)value.size, (char *)value.data);
        }
        *notfoundp = 1;
    }
}
```

Рисунок 20 - Обновление данных

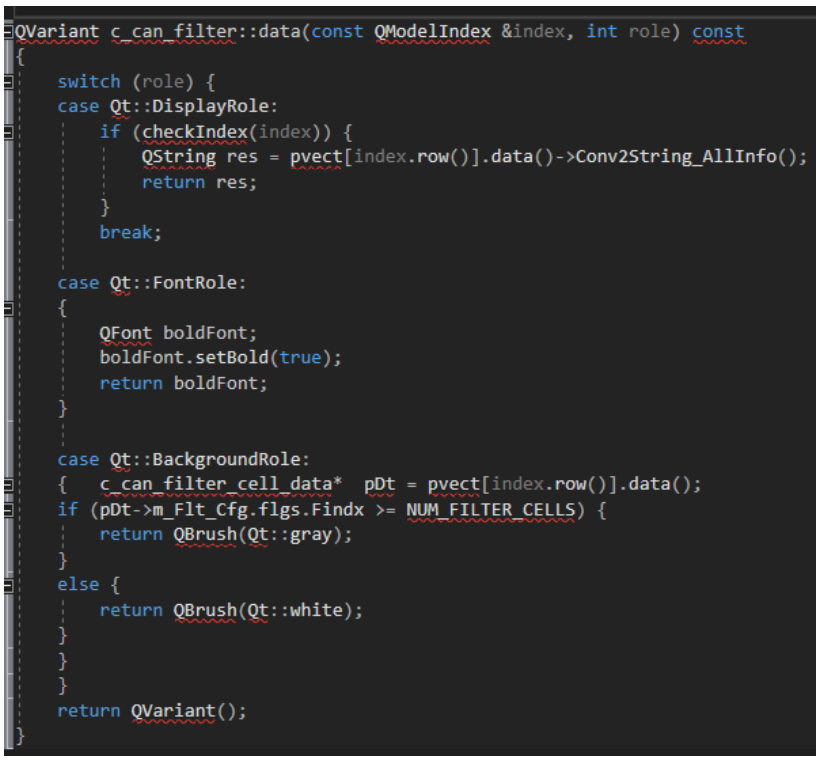
3.3 Разработка программного кода для взаимодействия ПК и устройства

Для получения данных с устройства и записи их в файл, использовался данный класс, который представлен на рисунке 21. В него входят такие методы как: фильтр, предназначенный для обработки получаемых данных с устройства и выбора отдельных элементов данных, это требуется для того, чтобы избавится от ненужных элементов данных, проходящих по шине за определенный участок времени. Метод записи данных применяется для передачи данных в ранее представленные методы по добавлению информации в базу данных DBC.

```
class c_can_filter : public QAbstractListModel
{
    Q_OBJECT

public:
    c_can_filter(QObject *parent = nullptr);

    enum { NUM_FILTER_CELLS = 28 };

    int rowCount(const QModelIndex &parent = QModelIndex()) const;

    QVariant data(const QModelIndex &index, int role = Qt::DisplayRole) const;

private:
    quint8                              Can2Start;
    QVector<c_can_filter_cell_data>     *pvect;
};
```

Рисунок 21 - Класс для получения и передачи данных в шину

Далее будут представлены листинги, отражающие взаимодействие устройства с ПК и обрабатывающие информационные потоки по методам проверки статических гипотез, а именно: независимости выборки, нормально распределенные генеральные совокупности и известные их дисперсии.



```
QVariant c_can_filter::data(const QModelIndex &index, int role) const
{
    switch (role) {
    case Qt::DisplayRole:
        if (checkIndex(index)) {
            QString res = pvect[index.row()].data()->Conv2String_AllInfo();
            return res;
        }
        break;

    case Qt::FontRole:
    {
        QFont boldFont;
        boldFont.setBold(true);
        return boldFont;
    }

    case Qt::BackgroundRole:
    {   c_can_filter_cell_data*  pDt = pvect[index.row()].data();
    if (pDt->m_Flt_Cfg.flgs.Findx >= NUM_FILTER_CELLS) {
        return QBrush(Qt::gray);
    }
    else {
        return QBrush(Qt::white);
    }
    }
    }
    return QVariant();
}
```

Рисунок 22 – Листинг кода методов класса

Для анализа данных полученных с CAN – шины и приведение и к определённому графику диагностики требуется несколько математических преобразований таких как Гипотеза о равенстве генеральных средних двух распределений, а также метод Венгерский представленной ниже в пункте о анализе трафика.

Требуется по выборочным средним при заданном уровне значимости, а проверить нулевую гипотезу, состоящую в том, что генеральные средние (математические ожидания) рассматриваемых совокупностей равны между собой (1):

$$H_0: M(X) = M(Y), (1)$$

Учитывая, что выборочные средние являются несмещенными оценками генеральных средних, т.е. M (X) = M(X) и M(y) = M(Y), нулевую гипотезу можно записать так (2):

$$H_0: M(\overline{X}) = M(Y), (2)$$

```
double round1(double x) { return ((x - floor(x)) >= 0.5) ? ceil(x) : floor(x); }

int RASpret() {

    const int nrolls = 10000;
    const int nstars = 72;
    const int columns = 10;
    static int p[columns];
    default_random_engine generator;
    double numbers[nrolls];


    double mean = 0.;
    double deviation = 1.;
    normal_distribution <double> distribution(mean, deviation);
    cout << "normal_distribution (" << mean << "," << deviation << "):" << endl;


    double min = DBL_MAX, max = DBL_MIN;
    for (int i = 0; i < nrolls; i++) {
        numbers[i] = distribution(generator);
        if (numbers[i] < min) min = numbers[i];
        if (numbers[i] > max) max = numbers[i];
    }
    for (int i = 0; i < nrolls; i++) {
        int interval = round1((numbers[i] - min)*(double)(columns - 1) / (max - min));
        interval = (interval<0 ? 0 : interval>columns - 1 ? columns - 1 : interval);
        p[interval]++;
    }
    for (int i = 0; i < columns; i++) {
        cout << string(round1(p[i] * (double)nstars / (double)(nrolls)), '*') << endl;
    }

    cin.sync();
    cin.get();
    return 0;
}
```

a

Рисунок 23 - Листинг показывающий аналитику данных

3.4 Анализ трафик сети

Для решения задачи с анализом трафика на шине используется Метод Венгерский, далее который будет рассмотрен для упущения рассмотрена тривиальная задача но также этот метод распространён в серив системах.

Венгерский метод является одним из самых распространенных методов решения задач о назначении. Главную идею данного метода впервые высказал венгерский математик Э. Эгервари (отсюда и название метода) задолго до возникновения теории линейного программирования [14, с. 211].

Сам алгоритм был разработан и опубликован Гарольдом Куном в 1955 году. В 1957 году Джеймс Манкрес показал, что этот алгоритм работает за четкое полиномиальное время (то есть, за время порядка полинома от n-ой степени). Таким образом в литературе этот алгоритм стал известнее не только как «венгерский», но и как «алгоритм Куна-Манкреса» или «алгоритм Манкреса» [13, с. 18]. Непосредственно алгоритм решения задачи состоит из

следующих шагов: 1. Вычитаем в матрице С от каждого элемента i-й строки минимальный элемент этой строки, i = 1 ... n. 2. Вычитаем от каждого элемента j-го столбца преобразованной матрицы затрат его минимальный элемент, j = 1 ... n. В результате выполнения двух пунктов каждая строка и каждый столбец матрицы расходов имеют, по крайней мере, один 0. 3. Просматриваем последовательно строки матрицы затрат, начиная с первой. Если строка имеет только один необозначенный 0, обозначаем его символом * и зачеркиваем (с помощью символа ^) все нули в этом же столбце. Ноль считается обозначен, если он обозначен символом *. Повторяем эти действия, пока каждая строка не останется без необозначенных нулей, или будет иметь их по крайней мере 2. 4. Действия пункта 3 повторяем для всех столбцов матрицы затрат. 5. Действия пунктов 3 и 4 повторяем последовательно (если необходимо), пока не получим один из трех возможных случаев: 14 а) каждая строка имеет назначение (с пометок 0 *) б) есть по крайней мере два необозначенных нуля в некоторых строках и некоторых столбцах матрицы затрат; в) нет обозначенных нулей и полное назначение еще не получено (число нулей с пометкой * меньше n). 6. В случае, а) задача об оптимальном назначении решена: *, соответствующие 0 * равны 1. Остальные - 0, конец алгоритма. В случае б) произвольно выбираем один из необозначенных нулей, обозначаем его символом *, зачеркиваем остальные нули в той же строке и в том же столбце и возвращаемся к пункту 3. Если имеет место случай в), то переходим к пункту 7. 7. Обозначаем символом # строки, для которых не получено назначение (в которых нет 0 *). Такие строки считаем обозначенными, остальные – не назначенными. Такую же терминологию будем использовать и для столбцов матрицы затрат. 8. Обозначаем символом # еще необозначенные столбцы, которые имеют зачеркнутый 0 (обозначен символом ^) в обозначенных строках. 9. Обозначаем символом # еще необозначенные строки, которые имеют назначение (то есть 0 *) в обозначенных столбцах. 10. Повторяем действия пунктов 8 и 9 до тех пор, пока больше не сможем обозначать строки

и столбцы матрицы затрат. 11. Зачеркиваем (с помощью отметки &) необозначенные строки и обозначенные столбцы матрицы затрат. 12. Находим минимальный не зачеркнутый элемент матрицы затрат, отнимаем его от элементов каждого из не зачеркнутых строк, добавляем к элементам всех зачеркнутых столбцов и переходим к пункту 3. При этом отметки элементов матрицы затрат (* и ^) теряют свою силу.

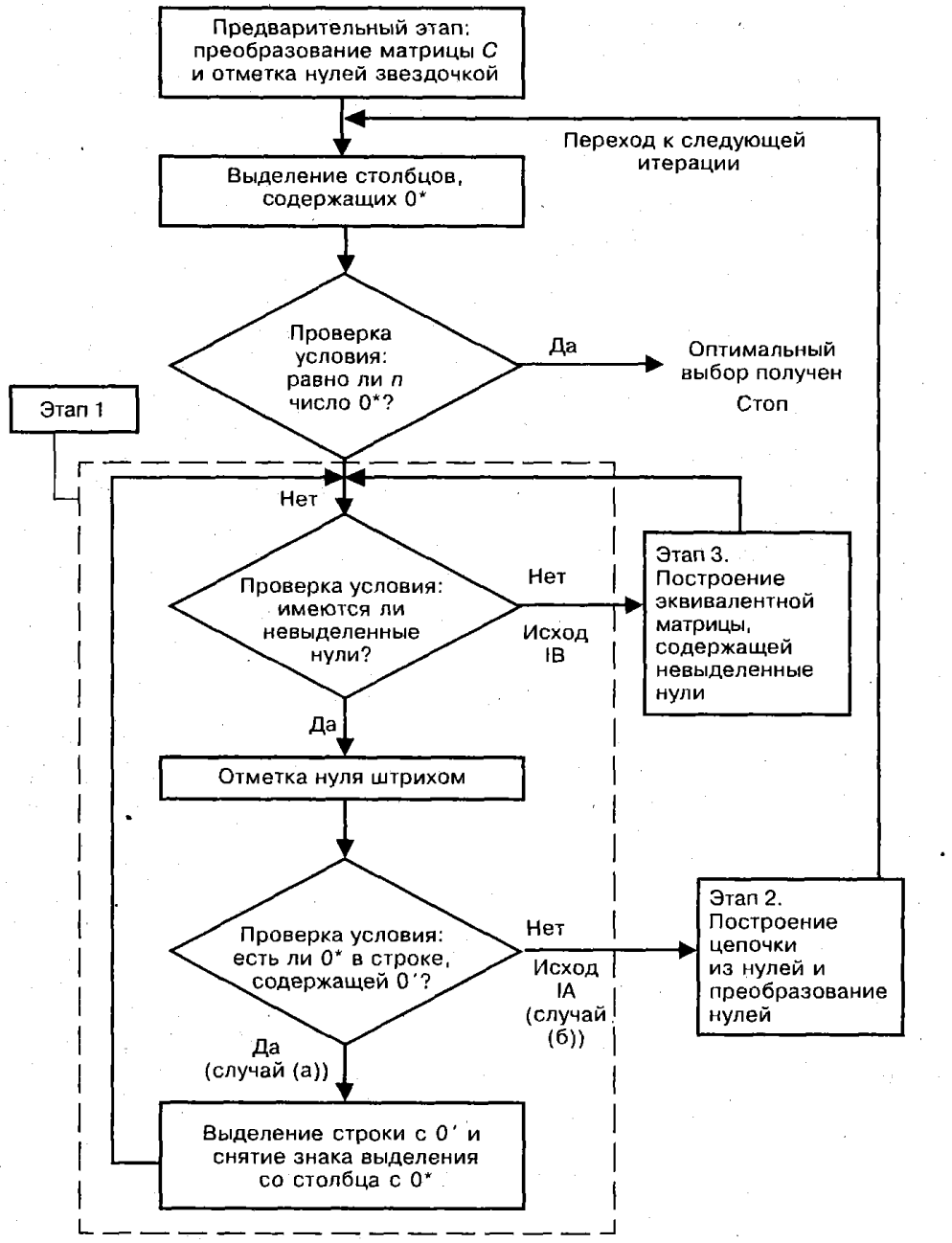


Рисунок 24- На рисунке представлена блок схема показывающая работу Венгерского метода

Выводы по главе

В данной главе была рассмотрена реализация программного обеспечения.

Первым этапом рассматривалась разработка интерфейса программы и построение графиков для удобного представления различной аналитической информации.

Вторым этапом была описана реализация работы с базой данных DBC, в первую очередь для были реализованы методы для считывания и поиска записей в базе, далее также была реализована запись и удаление данных из Базы данных.

Третий этап получение данных с устройства обработка данных, в первую очередь необходимо посмотреть направление данного кадра, а точнее откуда он пришёл, если это кадр подходит, то нам необходимо

И в заключении данной главы была рассмотрена методология метода венгерского, который применяется для анализа сетевого трафика и приводится блок схем к нему.

Заключение

В ходе выполнения выпускной квалифицированной работы был разработан программно-аппаратный комплекс для анализа трафика на CAN-шине.

Возможности данного комплекса достаточно обширные, так как он обладает удобным API на уровне микроконтроллера. Так же, как и динамическим изменением среды взаимодействия и интерфейса программы верхнего уровня, что позволяет пользователю\настроить рабочую среду исключительно под себя и свои задачи.

Данная программа была реализована для замещения импортных аналогов и превосходства над ними за счет более гибкого и эффективного функционала. Гибкость разработки проявляется в том, что структура достаточно модульная и имеется возможность использовать как различные API, т.е. с нашим ПО можно использовать различные устройства, так и зарубежные устройства данного типа. В будущем планируется дорабатывать, как само ПО, так и устройство, и перенос его на несколько типов микроконтроллеров. Например, такие как Меандр. Поскольку ныне используемые STM32 является зарубежным образцом, это накладывает некоторые ограничения на использование в российских разработках.

Поскольку для разработки использовалась среда разработки QTcreator, это дает возможность быстро и функционально вносить различные новые функции и дорабатывать уже существующие на уровне программного кода.

Применение наш аппаратный комплекс найдет во время разработки систем автомобиля, умный дом, в большинстве других разработок с использованием CAN-шина. Для того, чтобы оценить качество и безошибочность работы устройства принимающих и передающих устройствах в CAN-шину. Также можно использовать во время тестирования для поиска неисправностей и быстрого устранения таковых.

Список используемой литературы и используемых источников

1. Диаграмма Ганта [Электронный ресурс]. URL: https://datavizcatalogue.com/RU/metody/diagramma_ganta.html (дата обращения: 25.05.2021).
2. Моделирование сетевого графика проекта [Электронный ресурс]. URL: http://projectimo.ru/planirovanie-proekta/setevoj-grafik.html (дата обращения: 25.05.2021).
3. Онлайн-калькулятор по сетевому планированию [Электронный ресурс]. URL: https://math.semestr.ru/setm/setm_manual.php (дата обращения: 25.05.2021).
4. Платформа Node.js [Электронный ресурс]. URL: https://nodejs.org/en/ (дата обращения: 25.05.2021).
5. Плескунов М. А. Задачи сетевого планирования : учебное пособие. Екатеринбург : Изд-во Урал. ун-та, 2014. 92 с.
6. Программный продукт «1С: Управление проектным офисом» [Электронный ресурс]. URL: https://solutions.1c.ru/catalog/project-office/features (дата обращения: 25.05.2021).
7. Сетевое планирование [Электронный ресурс]. URL: http://upr-proektom.ru/setevoe-planirovanie (дата обращения: 25.05.2021).
8. Сетевое планирование в условиях неопределенности [Электронный ресурс]. URL: https://studme.org/80819/ekonomika/setevoe_planirovanie_usloviyah_neopredelennosti (дата обращения: 25.05.2021).
9. Современный учебник JavaScript [Электронный ресурс]. URL: https://learn.javascript.ru/ (дата обращения: 25.05.2021).
10. Создание десктопного приложения с помощью Electron и веб-технологий [Электронный ресурс]. URL: https://habr.com/ru/post/272075/ (дата обращения: 25.05.2021).

11. Спиридонов Э.С., Телятникова Н. А. Сетевое планирование в строительстве: Учебное пособие, М.: РУТ (МИИТ), 2018. 75 с.

12. Технология Electron [Электронный ресурс]. URL: https://electronjs.org (дата обращения: 25.05.2021).

13. Everything you need to know about PERT charts [Электронный ресурс]. URL: https://medium.com/nulab/everything-you-need-to-know-about-pert-charts-55978c06b5fc (дата обращения: 25.05.2021).

14. Liu Jun-yan “Schedule Uncertainty Control: A Literature review” , Physics Procedia 33 (2012) 1842 – 1848.

15. PERT Analysis [Электронный ресурс]. URL: https://acqnotes.com/acqnote/tasks/pert-analysis (дата обращения: 25.05.2021).

16. PERT chart generator [Электронный ресурс]. URL: https://www.visme.co/pert-chart-generator/ (дата обращения: 25.05.2021).

17. The Chromium Projects [Электронный ресурс]. URL: https://www.chromium.org/ (дата обращения: 25.05.2021).

18. Uncertainty in management [Электронный ресурс]. URL: http://kolibri.teacherinabox.org.au/modules/en-boundless/www.boundless.com/management/definition/uncertainty/index.html (дата обращения: 25.05.2021).

19. What is a Gantt chart? [Электронный ресурс]. URL: https://www.apm.org.uk/resources/find-a-resource/gantt-chart/#:~:text=A%20Gantt%20chart%20is%20a,useful%20for%20simplifying%20complex%20projects.&text=As%20it's%20in%20a%20bar,progress%20with%20a%20quick%20glance (дата обращения: 25.05.2021).

20. What is Project Evaluation Review Technique [Электронный ресурс]. URL: https://corporatefinanceinstitute.com/resources/knowledge/other/project-evaluation-review-technique-pert (дата обращения: 25.05.2021).

21. Just look at the image: viewpoint-specific surface normal prediction for improved multi-view reconstruction [Электронный ресурс] / Режим доступа:

https://www.cvfoundation.org/openaccess/content_cvpr_2016/papers/Galliani_Just_Look_at_CVP R_2016_paper.pdf

22. Introduction To Feature Detection And Matching [Электронный ресурс] / Режим доступа: https://medium.com/data-

23. breach/introduction-tofeature-detection-and-matching-65e27179885d

24. Introduction to SURF [Электронный ресурс] / Режим доступа: https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_surf_intro/py_surf_intro.html 45

25. ORB: an efficient alternative to SIFT or SURF / E. Rublee, V. Rabaud, K. Konolige, G. Bradski // IEEE International Conference on Computer Vision. – 2011. –Vol. 58, Issue 11. – P. 2564-2571.

26. ORB: Oriented FAST and Rotated BRIEF [Электронный ресурс] / Режим доступа: http://www.willowgarage.com/sites/default/files/orb_final.pdf

27. SIFT: Theory and practice [Электронный ресурс] / Режим доступа: https://aishack.in/tutorials/sift-scale-invariant-feature-transform-introduction/

28. SURF: Speeded Up Robust Features [Электронный ресурс] / Режим доступа: http://people.ee.ethz.ch/~surf/eccv06.pdf