МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ (ФИЛИАЛ)
ФЕДЕРАЛЬНОГО ГОСУДАРСТВЕННОГО БЮДЖЕТНОГО
ОБРАЗОВАТЕЛЬНОГО УЧРЕЖДЕНИЯ ВЫСШЕГО ОБРАЗОВАНИЯ
«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
В Г. ТАГАНРОГЕ РОСТОВСКОЙ ОБЛАСТИ
ПИ (филиал) ДГТУ в г. Таганроге

ЦМК «ПРИКЛАДНАЯ ИНФОРМАТИКА»

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ РАБОТ

по МДК.03.02 Разработка приложений управления интегрированными системами
профессионального модуля
«Участие в разработке приложений взаимодействия
с интеллектуальными интегрированными системами»

Таганрог

2026

Составитель: И.В Андриян

Практикум по выполнению практической работы по дисциплине «Разработка приложений управления интегрированными системами». ПИ (филиал) ДГТУ в г. Таганроге, 2026 г.

В практикуме кратко изложены теоретические вопросы, необходимые для успешного выполнения практической работы, рабочее задание и контрольные вопросы для самопроверки.

Предназначено для обучающихся по специальности 09.02.08 Интеллектуальные интегрированные системы

Ответственный за выпуск:

Зав. кафедрой (председатель ЦМК) (руководитель структурного подразделения, ответственного за реализацию ОПОП): О.В. Андриян

Ф.И.О.

Издательский центр ДГТУ, 2026 г.

Введение

Данное учебно-методическое пособие предназначено для практикума по курсу "Разработка приложений управления интегрированными системами" и предоставляет необходимую информацию для успешного выполнения практических работ.

В результате освоения материала учебного пособия (дисциплины) обучающийся будет:

Знать:

- основы устройства и функционирования операционных систем;
- классификации и устройства ПО;
- основ теории качества программных систем;
- способы описания алгоритмов.

Уметь:

- устанавливать и удалять прикладное ПО;
- создавать простые программы.

Владеть навыками:

- создания, тестирования и запуска приложений.

Практическая работа № 1.

Выполнение конфигурационных работ с микроконтроллером

Теоретическая часть

Микроконтроллер — это компактный вычислительный модуль, который включает в себя процессор, память и периферийные интерфейсы для управления различными устройствами. Микроконтроллеры широко используются bembedded системах, таких как бытовая электроника, автомобили, медицинское оборудование и многие другие устройства.

Основные компоненты микроконтроллера:

- Центральный процессор (CPU): выполняет инструкции и обрабатывает данные.
- Память: включает в себя оперативную память (RAM) для хранения временных данных и постоянную память (ROM, Flash) для хранения программы.
- Периферийные интерфейсы: позволяют взаимодействовать с внешними устройствами, такими как датчики, двигатели и дисплеи, например, I/O порты, UART, SPI, I2C.

Процесс конфигурации микроконтроллера:

- Определение задачи: выбрать задачу, которую будет решать микроконтроллер (управление светодиодами, считывание данных с датчиков, управление моторами и пр.).
- Подключение периферийных устройств: подключить все необходимые компоненты к микроконтроллеру.
- Написание программного кода: создать программу на языке, поддерживаемом микроконтроллером (как правило, C/C++), которая выполняет требуемые функции.
- Загрузка программы в память микроконтроллера: использовать программное обеспечение и программатор для загрузки кода в микроконтроллер.
- Тестирование и отладка: запустить программу и отладить её при необходимости.

Цель работы:

Изучить процесс конфигурации и программирования микроконтроллеров, освоить основные методы создания и загрузки программного кода, а также наладить взаимодействие микроконтроллера с внешними компонентами.

Рабочее задание

1. Выбор микроконтроллера:

- Выберите микроконтроллер из предложенного списка (например, Arduino, STM32, PIC, ESP32) и подготовьте рабочую среду (IDE для разработки).

2. Определение задачи:

- Определите задачу, которую будет выполнять ваш микроконтроллер (например, управление светодиодом, считывание данных с датчика температуры).

3. Подключение компонентов:

- Соберите схему, используя выбранный микроконтроллер и необходимые компоненты, такие как резисторы, датчики, светодиоды и т.д.

4. Разработка программного кода:

- Напишите код для выполнения заданной задачи. Убедитесь, что вы правильно настроили все порты ввода/вывода.
- Добавьте комментарии к коду, объясняя, что делает каждая часть.

5. Загрузка программы:

- Используйте соответствующий программатор или среду разработки для загрузки написанной программы в память микроконтроллера.

6. Тестирование и отладка: Проверьте, выполняет ли ваш микроконтроллер заданную функцию. Выполните отладку кода при необходимости и внесите корректировки.

7. Документация и отчет:

- Подготовьте отчет о выполненной работе, включая схемы подключения, написанный код, результаты тестирования и описания возникших проблем и их решений.

Контрольные вопросы:

1. Что такое микроконтроллер и каковы его основные компоненты?
2. Каковы основные шаги в процессе конфигурации и программирования микроконтроллера?
3. В чем разница между RAM и ROM в контексте микроконтроллеров?
4. Какие языки программирования используются для программирования микроконтроллеров?
5. Как правильно подключить периферийные устройства к микроконтроллеру?
6. Что такое I/O порты и как они используются в микроконтроллерах?
7. Как загружается программа в память микроконтроллера?
8. Что такое отладка программы, и какие инструменты для этого можно использовать?
9. Каково значение комментариев в коде и как они помогают в процессе программирования?
10. Как микроконтроллер может взаимодействовать с различными датчиками и исполнительными механизмами?

Общие положения

Практические занятия выполняются каждым обучающимся самостоятельно в полном объеме и согласно содержанию методических указаний.

Перед выполнением обучающийся должен отчитаться перед преподавателем за выполнение предыдущего занятия (сдать отчет).

Обучающийся должен на уровне понимания и воспроизведения предварительно усвоить необходимую для выполнения практических занятий теоретическую и информацию.

Обучающийся, получивший положительную оценку и сдавший отчет по предыдущему практическому занятию, допускается к выполнению следующему занятию.

Обучающийся, пропустивший практическое занятие по уважительной либо неуважительной причине, закрывает задолженность в процессе выполнения последующих практических занятий.

Форма отчета:

- титульный лист;
- введение (цель и задачи);
- выполнение
- заключение

Время работы: 6 часов.

Практическая работа № 2.

Первичная настройка проекта под микроконтроллер

Теоретическая часть

Микроконтроллер — это небольшое вычислительное устройство, включающее в себя процессор, память и периферийные устройства для управления различной аппаратурой. Микроконтроллеры используются в широком спектре приложений, от бытовой электроники до промышленных систем управления.

Первичная настройка проекта:

Первичная настройка проекта включает в себя создание основы для разработки программного обеспечения, которое будет работать на микроконтроллере. Этот процесс обычно включает следующие этапы:

- Выбор платформы и IDE: для начала необходимо выбрать платформу (например, Arduino, STM32, PIC) и установить соответствующую среду разработки (IDE) (например, Arduino IDE, STM32CubeIDE, MPLAB X).
- Настройка проекта: Создание нового проекта в IDE, настройка параметров проекта, таких как название, тип микроконтроллера и конфигурация компилятора.
- Подключение библиотек: Включение необходимых библиотек для работы с периферийными устройствами (датчики, двигатели и т.д.), что упрощает работу с оборудованием.
- Создание основной программы: Написание кода для основного цикла (setup и loop в Arduino) и добавление основной логики работы проекта.
- Компиляция и загрузка: Компиляция кода и загрузка его в память микроконтроллера с помощью программатора или USB-адаптера.

Цель работы:

Изучить процесс первоначальной настройки проекта для разработки программного обеспечения под микроконтроллер, освоить создание и конфигурацию проекта, а также научиться работать с библиотеками и примерами кода.

Рабочее задание

1. Установка IDE:
- Выберите подходящую для вашего микроконтроллера среду разработки и установите ее на компьютер. Например, для Arduino — Arduino IDE, для STM32 — STM32CubeIDE.

2. Создание нового проекта:
- Откройте IDE и создайте новый проект.
- Укажите название проекта и выберите модель вашего микроконтроллера.

3. Настройка параметров проекта:

- Проверьте и настройте параметры проекта, такие как частота работы микроконтроллера и выбранный компилятор.

4. Подключение библиотек:

- Найдите и подключите необходимые библиотеки, которые могут понадобиться для работы с периферийными устройствами (например, Adafruit Sensor, Wire для I2C).

5. Написание кода:

- Создайте основной файл программы и напишите код для выполнения базовых операций, например, чтения данных с датчика и их отображения в последовательном мониторе.

6. Компиляция проекта:

- Скомпилируйте проект и убедитесь, что в процессе не возникло ошибок.

7. Загрузка программы:

Загрузите скомпилированный код в память микроконтроллера, используя выбранный программатор или интерфейс UART/USB.

8. Тестирование:

- Проведите тестирование вашего проекта для проверки правильности работы кода и управления периферийными устройствами.

9. Документация:

- Подготовьте отчет о выполненной работе с описанием каждого этапа, кодом, результатами тестов и возникшими проблемами.

Контрольные вопросы:

1. Что такое микроконтроллер, и каковы его основные компоненты?
2. Почему важна первичная настройка проекта при разработке под микроконтроллер?
3. Как выбрать правильную среду разработки (IDE) для работы с микроконтроллером?
4. Что такое библиотеки в контексте микроконтроллеров, и зачем они нужны?
5. Каков процесс компиляции кода для микроконтроллера?
6. Что такое основной цикл (setup и loop) в программе на Arduino?
7. Какие основные параметры необходимо настроить при создании проекта в IDE?
8. Что делать, если при компиляции кода возникают ошибки?
9. Как проводить тестирование работоспособности микроконтроллера после загрузки кода?

10. Какова структура отчета о лабораторной работе, и что в него должно входить?

Общие положения

Практические занятия выполняются каждым обучающимся самостоятельно в полном объеме и согласно содержанию методических указаний.

Перед выполнением обучающийся должен отчитаться перед преподавателем за выполнение предыдущего занятия (сдать отчет).

Обучающийся должен на уровне понимания и воспроизведения предварительно усвоить необходимую для выполнения практических занятий теоретическую и информацию.

Обучающийся, получивший положительную оценку и сдавший отчет по предыдущему практическому занятию, допускается к выполнению следующему занятию.

Обучающийся, пропустивший практическое занятие по уважительной либо неуважительной причине, закрывает задолженность в процессе выполнения последующих практических занятий.

Форма отчета:

- титульный лист;
- введение (цель и задачи);
- выполнение
- заключение

Время работы: 6 часов.

Практическая работа № 3.

Работа с программным кодом (анализ и отслеживание изменение)

Теоретическая часть

Контроль версий — это система, которая отслеживает изменения в файлах, позволяя пользователям возвращаться к ранее сохраненным версиям. Это особенно важно в разработке программного обеспечения, где несколько разработчиков могут работать над одним проектом одновременно.

Основные понятия:

- Репозиторий: Хранилище проекта, где находится весь код и история изменений.
- Коммит: Сохранение изменений в репозитории, сопровождаемое комментарием, объясняющим, почему были внесены изменения.
- Ветка (branch): Отдельная линия разработки, позволяющая работать над различными функциями или исправлениями без влияния на основную ветку (обычно master или main).
- Слияние (merge): Процесс объединения изменений из одной ветки в другую.
- Конфликт: Ситуация, когда изменения в одной и той же части кода противоречат друг другу на разных ветках.

Инструменты контроля версий

Самым популярным инструментом для контроля версий является Git. Git позволяет управлять изменениями в проекте, легко переключаться между версиями и ветками, а также вести совместную разработку.

Цель работы:

Изучить методы анализа и отслеживания изменений в программном коде, познакомиться с инструментами для работы с системами контроля версий, а также понять важность соблюдения контроля версий в процессе разработки программного обеспечения.

Рабочее задание

1. Установка и настройка Git:

- Установите Git на своем компьютере и настройте его, указав имя пользователя и адрес электронной почты.

2. Создание репозитория:

- Создайте новый Git-репозиторий для простого проекта (например, консолевая программа на Python или Java).

3. Работа с коммитами:

- Напишите простой код, затем выполните коммит с соответствующим сообщением. Измените код, добавив новую функциональность, и сделайте еще один коммит.
- Используйте команды `git log` и `git status`, чтобы отследить историю коммитов и состояние репозитория.

4. Создание ветки:

- Создайте новую ветку, внесите несколько изменений и сделайте коммит.
- Вернитесь в основную ветку и выполните слияние изменений из новой ветки.

5. Разрешение конфликтов:

- Попробуйте создать конфликт, изменив одну и ту же строку кода в двух разных ветках. Попытайтесь слить ветки и разрешите конфликт.

6. Документация:

- Подготовьте отчет по выполненной лабораторной работе, включив в него описания всех шагов, использованные команды и экраны терминала.

Контрольные вопросы:

1. Что такое контроль версий и в чем его значимость в разработке программного обеспечения?
2. Описываются ли основные понятия контроля версий, такие как репозиторий, коммит, ветка и слияние?
3. Каковы основные команды Git, которые вы использовали в ходе выполнения лабораторной работы?
4. Что такое `git log` и какую информацию он предоставляет?
5. Как разрешить конфликт, возникший в результате слияния двух веток?
6. Какова процедура создания новой ветки и внесения изменений в ней?
7. Почему важно оставлять полезные комментарии при выполнении коммитов?
8. Что произойдет, если коммит последовательно срабатывать в случаях изменения одной и той же строки после последнего коммита на разных ветках?
9. Как поддерживать актуальность репозитория при работе в команде?
10. Что должно быть включено в отчет о выполненной лабораторной работе?

Общие положения

Практические занятия выполняются каждым обучающимся самостоятельно в полном объеме и согласно содержанию методических указаний.

Перед выполнением обучающийся должен отчитаться перед преподавателем за выполнение предыдущего занятия (сдать отчет).

Обучающийся должен на уровне понимания и воспроизведения предварительно усвоить необходимую для выполнения практических занятий теоретическую и информацию.

Обучающийся, получивший положительную оценку и сдавший отчет по предыдущему практическому занятию, допускается к выполнению следующему занятию.

Обучающийся, пропустивший практическое занятие по уважительной либо неуважительной причине, закрывает задолженность в процессе выполнения последующих практических занятий.

Форма отчета:

- титульный лист;
- введение (цель и задачи);
- выполнение
- заключение

Время работы: 8 часов.

Практическая работа № 4.

Работы по настройке программных инструментов под микроконтроллер

Теоретическая часть

Микроконтроллер — это небольшое компьютерное устройство, которое включает микропроцессор, память и периферийные устройства на одном кристалле. Они используются в самых разных приложениях, включая бытовую электронику, автомобили, промышленные системы и IoT-устройства.

Основные этапы работы с микроконтроллерами:

- Выбор микроконтроллера: Определение требований к производительности, памяти и периферийным устройствам.
- Настройка программной среды: Установка необходимых инструментов, таких как компиляторы, среды разработки (IDE) и программаторы.
- Разработка кода: Писать программы на языках программирования (например, C, C++, Python), используя специализированные библиотеки для работы с периферийными устройствами.
- Компиляция: Преобразование исходного кода в исполняемый файл.
- Загрузка на микроконтроллер: Использование программатора для записи скомпилированного кода в память микроконтроллера.
- Отладка и тестирование: Проверка работы программы, диагностика ошибок и отладка кода.

Инструменты для работы с микроконтроллерами:

- Среды разработки (IDE): Keil, MPLAB X, Arduino IDE, PlatformIO и др.
- Компиляторы: GCC, MPLAB XC8, ARM Keil и др.
- Программаторы: USBasp, J-Link, PICkit, ST-LINK и др.

Цель работы:

Ознакомиться с основами настройки программных инструментов для программирования микроконтроллеров, изучить программные среды разработки, компиляции и отладки, а также понять процесс загрузки кода на микроконтроллер.

Рабочее задание

1. Выбор микроконтроллера:

- Выберите микроконтроллер (например, ATmega328, PIC16F84, STM32) для выполнения лабораторной работы.

2. Установка среды разработки:

- Установите подходящую среду разработки для выбранного микроконтроллера (например, Arduino IDE для ATmega328 или MPLAB X для PIC).

3. Настройка компилятора:

- Убедитесь, что ваша среда разработки правильно настроена для компиляции кода. В случае необходимо установите соответствующий компилятор.

4. Написание программы:

- Создайте простую программу, которая, например, мигает светодиодом, подключенным к определенному пину микроконтроллера. Пример для Arduino:

```
c

void setup() {
       pinMode(LED_BUILTIN, OUTPUT);
   }

   void loop() {
       digitalWrite(LED_BUILTIN,   HIGH);          //   Включить   светодиод
delay(1000);                        // Задержка 1 секунда
       digitalWrite(LED_BUILTIN, LOW);    // Выключить светодиод
       delay(1000);                       // Задержка 1 секунда    }
```

5. Компиляция программы:

- Скомпилируйте ваш код и убедитесь, что ошибок нет.

6. Загрузка на микроконтроллер:

- Подключите микроконтроллер через программатор или USB и загрузите скомпилированный код.

7. Отладка и тестирование:

- Проверьте работоспособность программы. При необходимости используйте средства отладки, чтобы найти и исправить ошибки.

8. Документация:

- Подготовьте отчет, в который должны войти используемые инструменты, процесс написания кода, компиляции и загрузки, а также результаты тестирования.

Контрольные вопросы:

1. Что такое микроконтроллер и какие основные его компоненты?
2. Для чего требуется настройка программной среды при работе с микроконтроллерами?
3. Какова основная роль компилятора при работе с микроконтроллерами?
4. Опишите процесс разработки программы для микроконтроллера от написания кода до его загрузки.
5. Какие среды разработки и компиляторы вы использовали в ходе работы? Какие их основные преимущества?
6. Как можно отладить программу, если она не работает должным образом на микроконтроллере?

7. Почему важно тестирование программы после загрузки её на микроконтроллер?

8. Какое оборудование может понадобиться для работы с микроконтроллерами?

9. Что должно быть включено в отчет о выполненной лабораторной работе?

10. Какие примеры приложений можно создать с использованием микроконтроллеров в реальных задачах?

Общие положения

Практические занятия выполняются каждым обучающимся самостоятельно в полном объеме и согласно содержанию методических указаний.

Перед выполнением обучающийся должен отчитаться перед преподавателем за выполнение предыдущего занятия (сдать отчет).

Обучающийся должен на уровне понимания и воспроизведения предварительно усвоить необходимую для выполнения практических занятий теоретическую и информацию.

Обучающийся, получивший положительную оценку и сдавший отчет по предыдущему практическому занятию, допускается к выполнению следующему занятию.

Обучающийся, пропустивший практическое занятие по уважительной либо неуважительной причине, закрывает задолженность в процессе выполнения последующих практических занятий.

Форма отчета:

- титульный лист;
- введение (цель и задачи);
- выполнение
- заключение

Время работы: 8 часов.

Практическая работа № 5.

Настройка программатора для прошивки микроконтроллера

Теоретическая часть

Программатор — это устройство, используемое для записи кода (прошивки) в память микроконтроллера. Программаторы могут работать через различные интерфейсы, такие как JTAG, SPI, UART и многие другие.

Виды программаторов:

- USB-программаторы: Используют интерфейс USB для подключения к ПК, такие как USBasp для AVR, ST-LINK для STM32 и J-Link для различных микроконтроллеров.
- Параллельные программаторы: Ранее использовались для подключения к компьютеру через параллельный порт, сейчас менее распространены.
- Встраиваемые программаторы: Некоторые микроконтроллеры имеют встроенные возможности для программирования и отладки.

Процесс прошивки микроконтроллера:

- Подготовка: Установка необходимого программного обеспечения (IDE, среда разработки).
- Подключение программатора: Установите соединение между программатором и микроконтроллером.
- Настройка программатора: В настройках программного обеспечения выберите компилятор и программатор.
- Прошивка кода: Загрузите скомпилированный код в памяти микроконтроллера с использованием программного обеспечения.
- Проверка работы: После прошивки выполните тестирование функциональности устройства.

Цель работы:

Ознакомиться с процессом настройки программатора для прошивки микроконтроллеров, изучить основные команды и методы загрузки кода, а также понять взаимодействие между ПК и микроконтроллером.

Рабочее задание

1. Выбор микроконтроллера и программатора:

- Выберите микроконтроллер (например, ATmega328, PIC16F84, STM32) и соответствующий программатор (например, USBasp для AVR, ST-LINK для STM32).

2. Установка программного обеспечения:

- Установите необходимое программное обеспечение для работы с выбранным микроконтроллером (например, Arduino IDE для ATmega, MPLAB X для PIC или STM32CubeIDE для STM32).

3. Подключение программатора:

- Подключите программатор к микроконтроллеру согласно схеме вывода (например, используйте шнур USB для USBasp и соответствующие выводы для подключения к пинам микроконтроллера).

4. Настройка программатора в среде разработки:

- Откройте программное обеспечение и настройте его на использование выбранного программатора, выберите правильный микроконтроллер в настройках.

5. Создание и компиляция программы:

- Напишите простую программу для микроконтроллера, например, мигание светодиодом. Скомпилируйте код, убедитесь, что ошибок нет.

6. Прошивка микроконтроллера:

- С помощью программатора загрузите скомпилированный код в микроконтроллер. Зафиксируйте процесс загрузки и появившиеся сообщения об успехе или ошибках.

7. Проверка функциональности:

- Проверьте работу программы на микроконтроллере. Убедитесь, что все работает должным образом.

8. Документация:

- Подготовьте отчёт о выполненной лабораторной работе, описывая все выполненные действия, настройки программного обеспечения и результаты тестирования.

Контрольные вопросы:

1. Что такое программатор и какова его основная функция в процессе работы с микроконтроллерами?
2. Каковы основные виды программаторов и в каких случаях используются каждый из них?
3. Опишите процесс прошивки микроконтроллера шаг за шагом.
4. Как установить соединение между программатором и микроконтроллером, и почему это важно?
5. Что такое "схема подключения" и как её использовать при работе с программаторами?
6. Почему важно проверять целостность прошивки и тестировать функциональность устройства после прошивки?

7. Как правильно настроить программное обеспечение для работы с выбранным микроконтроллером и программатором?

8. Что делать, если при прошивке возникли ошибки? Какие шаги могут помочь в устранении проблем?

9. Что должно быть включено в отчет о выполненной лабораторной работе?

10. Приведите примеры сценариев использования прошивки микроконтроллера в реальных устройствах.

Общие положения

Практические занятия выполняются каждым обучающимся самостоятельно в полном объеме и согласно содержанию методических указаний.

Перед выполнением обучающийся должен отчитаться перед преподавателем за выполнение предыдущего занятия (сдать отчет).

Обучающийся должен на уровне понимания и воспроизведения предварительно усвоить необходимую для выполнения практических занятий теоретическую и информацию.

Обучающийся, получивший положительную оценку и сдавший отчет по предыдущему практическому занятию, допускается к выполнению следующему занятию.

Обучающийся, пропустивший практическое занятие по уважительной либо неуважительной причине, закрывает задолженность в процессе выполнения последующих практических занятий.

Форма отчета:

- титульный лист;
- введение (цель и задачи);
- выполнение
- заключение

Время работы: 4 часа.

Практическая работа № 6.

Анализ программного кода для микроконтроллера

Теоретическая часть

Анализ программного кода — это процесс изучения и оценивания исходного кода программы с целью выявления ошибок, оптимизации, повышения читаемости и совместимости. В контексте разработки для микроконтроллеров этот процесс особенно важен, поскольку ограниченные ресурсы устройств требуют тщательного подхода к оптимизации.

Основные аспекты анализа кода:

- Стиль кода: Соответствие установленным стандартам кодирования, например, использование понятных имен переменных и функций, правильное форматирование.
- Логика: Проверка на наличие логических ошибок или неточностей в алгоритмах.
- Эффективность: Оценка использования ресурсов (память, скорость выполнения) и возможности повышения производительности.
- Безопасность: Анализ кода на предмет уязвимостей и слабых мест, которые могут привести к сбоям или непредвиденному поведению.
- Документация: Оценка наличия и качества комментариев к коду, что важно для понимания логики работы программы.

Методы анализа кода:

- Ручной анализ: Пересмотр кода разработчиком или группой разработчиков для выявления ошибок, неточностей и возможности улучшения.
- Инструментальный анализ (статический анализ): Использование инструментов для автоматического поиска ошибок, потенциальных уязвимостей и для оценки стиля кодирования.

Цель работы:

Изучить основные принципы анализа программного кода для микроконтроллеров, провести ревью кода с целью оценки его качества, оптимальности и функциональности, а также понять важность документации и комментариев.

Рабочее задание

1. Выбор проекта:

- Выберите проект для микроконтроллера, который будет анализироваться (например, простая программа для управления светодиодом).

2. Изучение исходного кода:

- Ознакомьтесь с исходным кодом. Прочитайте его вслух или совместно с группой, чтобы лучше понять алгоритм работы.

3. Проведение анализа:

Выполните анализ кода по следующим критериям:

- Соответствие стилю кодирования.
- Логическая корректность программы.
- Эффективность использования ресурсов.
- Наличие и качество документации и комментариев.

4. Запись результатов анализа:

- Составьте список найденных недостатков, предложите рекомендации по их исправлению. Обозначьте возможные улучшения.

5. Документация результатов:

- Подготовьте отчет, в который включите результаты анализа, а также оценки по каждому из рассмотренных аспектов (вес, важность, сложность).

6. Обсуждение:

- Проведите групповое обсуждение результатов анализа. Обсуждайте различные мнения и подходы к улучшению кода.

Контрольные вопросы:

1. Почему анализ программного кода важен в разработке для микроконтроллеров?
2, Каковы основные аспекты анализа кода, и зачем их проверять?
3. Что такое стиль кода, и как он влияет на понимание и поддержку программы?
4. Почему важно проводить ревью алгоритмов на предмет логических ошибок?
5. Какие инструменты вы можете использовать для статического анализа кода? Приведите примеры.
6. Каково значение документации и комментариев в исходном коде?
7. Как вы можете оценить эффективность алгоритма, использованного в программе для микроконтроллера?
8. Каковы потенциальные последствия игнорирования анализа программного кода?
9. Как можно улучшить качество кода, помимо исправления ошибок?
10. Какие шаги вы предприняли бы для обучения и внедрения стандартов кодирования в вашу команду разработки?

Общие положения

Практические занятия выполняются каждым обучающимся самостоятельно в полном объеме и согласно содержанию методических указаний.

Перед выполнением обучающийся должен отчитаться перед преподавателем за выполнение предыдущего занятия (сдать отчет).

Обучающийся должен на уровне понимания и воспроизведения предварительно усвоить необходимую для выполнения практических занятий теоретическую и информацию.

Обучающийся, получивший положительную оценку и сдавший отчет по предыдущему практическому занятию, допускается к выполнению следующему занятию.

Обучающийся, пропустивший практическое занятие по уважительной либо неуважительной причине, закрывает задолженность в процессе выполнения последующих практических занятий.

Форма отчета:

- титульный лист;
- введение (цель и задачи);
- выполнение
- заключение

Время работы: 6 часов.

Практическая работа № 7.

Написать приложение для опроса датчиков и отправки значений через протокол MQTT на целевое устройство

Теоретическая часть

MQTT (Message Queuing Telemetry Transport) — это легковесный протокол обмена сообщениями, разработанный для устройств с ограниченными ресурсами и сетями с низкой пропускной способностью. Он работает по принципу "издатель-подписчик", что делает его идеальным для Интернета вещей (IoT).

Основные компоненты MQTT:

- Клиенты: Устройства или приложения, которые отправляют (издатели) или получают (подписчики) сообщения.
- Брокер: Сервер, который управляет передачей сообщений между клиентами. Брокер получает сообщения от отправителей и маршрутизирует их к подписчикам.
- Темы: Именованные каналы, через которые клиенты отправляют и получают сообщения.

Процесс разработки приложения:

- Настройка среды разработки: Установите необходимые библиотеки для работы с MQTT и драйверы для датчиков.
- Подключение датчиков: Выберите и подключите необходимые датчики к вашей платформе (например, Arduino, Raspberry Pi) для сбора данных.
- Разработка кода: Напишите программу, которая будет обрабатывать данные с датчиков и отправлять их через MQTT.
- Подключение к брокеру MQTT: Настройте соединение с брокером для отправки и получения сообщений.
- Тестирование приложения: Проверка работы приложения и корректной отправки сообщений.

Цель работы:

Разработать простое приложение для опроса данных с датчиков и отправки значений на целевое устройство с использованием протокола MQTT, а также ознакомится с основами работы протокола и его применением в IoT.

Рабочее задание

1. Выбор платформы и датчиков:

- Выберите платформу для разработки (например, Arduino, Raspberry Pi) и датчики (например, DHT11 для температуры и влажности, BMP180 для давления).

2. Установка необходимых библиотек:

- Установите библиотеки для работы с вашим оборудованием и MQTT. Для Arduino можно использовать библиотеки `PubSubClient` для MQTT и библиотеки для работы с выбранными датчиками.

3. Подключение датчиков:

- Подключите датчики к платформе и проверьте их работоспособность, используя примеры кода из документации.

4. Разработка приложения:

Напишите программу, которая:

- Подключается к Wi-Fi (если используете ESP8266 или ESP32).
- Подключается к MQTT-брокеру (например, Mosquitto).
- Опросит датчики раз в определённый период времени и отправляет полученные значения на соответствующую тему.

5. Тестирование приложения:

- Запустите приложение и проверьте, что данные отправляются и получаются корректно с помощью MQTT-клиента (например, MQTT.fx или Mosquitto_sub).

6. Документация:

- Подготовьте отчет, описывающий этапы разработки, используемые технологии и библиотеки, а также результаты тестирования.

Контрольные вопросы:

1. Что такое MQTT и в каких случаях он предпочтителен для применения?
2. Каков принцип работы брокера MQTT? Как он осуществляет маршрутизацию сообщений?
3. Что такое темы в MQTT и как они используются в передаче сообщений?
4. Как вы можете обеспечить надежную связь между клиентами и брокером MQTT?
5. Почему важно проверять данные с датчиков перед отправкой их по сети?
6. Как вы можете настроить ваше приложение для обработки ошибок при работе с подключением к брокеру?
7. Что должно быть включено в отчет о выполненной лабораторной работе?
8. Как можно оптимизировать отправку данных для низкоскоростных и ограниченных сетей?
9. Приведите примеры других протоколов, сравните их с MQTT.
10. Какую роль играют библиотеки в упрощении разработки приложений для микроконтроллеров?

Общие положения

Практические занятия выполняются каждым обучающимся самостоятельно в полном объеме и согласно содержанию методических указаний.

Перед выполнением обучающийся должен отчитаться перед преподавателем за выполнение предыдущего занятия (сдать отчет).

Обучающийся должен на уровне понимания и воспроизведения предварительно усвоить необходимую для выполнения практических занятий теоретическую и информацию.

Обучающийся, получивший положительную оценку и сдавший отчет по предыдущему практическому занятию, допускается к выполнению следующему занятию.

Обучающийся, пропустивший практическое занятие по уважительной либо неуважительной причине, закрывает задолженность в процессе выполнения последующих практических занятий.

Форма отчета:

- титульный лист;
- введение (цель и задачи);
- выполнение
- заключение

Время работы: 10 часов.

Список литературы

1. Бердникова А. А., Иванов С. Л., Лямин А. С., Рейн А. Д. Основы алгоритмизации и программирования: Учебное пособие для СПО https://e.lanbook.com/book/434075 "Лань", 2024;
2. Баланов А. Н. Построение микросервисной архитектуры и разработка высоконагруженных приложений https://lanbook.com/catalog/informatika/postroenie-mikroservisnoy-arkhitektury-i-razrabotka-vysokonagruzhennykhprilozheniy73382656/ "Лань", 2025;
3. Андрианова А. А., Исмагилов Л. Н., Мухтарова Т. М. Алгоритмизация и программирование. Практикум: Учебное пособие для СПО https://e.lanbook.com/book/483449 "Лань", 2025.