**Содержание**

[Введение 3](#__RefHeading___Toc5996_4056234292)

[1 Постановка задачи 4](#__RefHeading___Toc5998_4056234292)

[1.1 Анализ аналогичных решений 5](#__RefHeading___Toc6000_4056234292)

[1.1.1 Аналог «Menu System for ILI9341\_t3 displays (Teensy) with touch or mechanical input» 5](#__RefHeading___Toc2189_2862897311)

[1.1.2 Аналог «GEM» 6](#__RefHeading___Toc6004_4056234292)

[1.2 Вывод 7](#__RefHeading___Toc6006_4056234292)

[2 Проектирование библиотеки 8](#__RefHeading___Toc6008_4056234292)

[2.1 Проектирование классов 8](#__RefHeading___Toc6010_4056234292)

[2.2 Описание средств разработки 10](#__RefHeading___Toc6012_4056234292)

[2.3 Вывод 11](#__RefHeading___Toc6014_4056234292)

[3 Реализация библиотеки 12](#__RefHeading___Toc6016_4056234292)

[3.1 Разработка библиотеки 12](#__RefHeading___Toc6018_4056234292)

[3.2 Вывод 12](#__RefHeading___Toc6020_4056234292)

[4 Тестирование библиотеки 14](#__RefHeading___Toc6022_4056234292)

[4.1 Вывод 16](#__RefHeading___Toc6024_4056234292)

[5 Руководство пользователя 17](#__RefHeading___Toc6026_4056234292)

[Заключение 18](#__RefHeading___Toc6032_4056234292)

[Список использованных источников 19](#__RefHeading___Toc6034_4056234292)

[Приложение А 20](#__RefHeading___Toc6036_4056234292)

# **Введение**

Развитие мобильных систем и устройств значительно изменило наши повседневные задачи и подход к взаимодействию с информацией. С распространением различных типов мобильных устройств, включая кнопочные телефоны, возникает необходимость разработки удобных и эффективных интерфейсов, которые могли бы работать на ограниченных ресурсах этих устройств. Одним из важнейших аспектов таких интерфейсов является система меню, обеспечивающая пользователю простой и интуитивно понятный доступ к функциям и возможностям устройства.

Цель данной курсовой работы заключается в разработке библиотеки классов на языке программирования C++, которая позволяет создавать меню и подменю для кнопочных мобильных устройств с TFT SPI дисплеями. Данная библиотека предназначена для облегчения процесса создания интерактивных графических интерфейсов, позволяющих пользователю комфортно взаимодействовать с устройством.

Для обеспечения высокой производительности и минимального потребления ресурсов процессора, библиотека оптимизирована для работы в условиях ограниченных вычислительных мощностей. Особое внимание уделено предотвращению графических артефактов при обработке изображения, что обеспечивает плавность и эстетичность отображения интерфейса.

Кроме того, библиотека прошла тщательное тестирование для обеспечения надежности и корректности работы. Реализована поддержка сохранения состояния приложения при навигации, что позволяет пользователю продолжить работу с того же места, где она была прервана.

Важной частью разработки является подробное документирование кода, что делает библиотеку удобной для использования и расширения. Комментарии в коде позволяют другим разработчикам легко понимать и модифицировать функционал библиотеки.

Таким образом, разработанная библиотека предоставляет разработчикам инструмент для создания удобных и функциональных меню на кнопочных мобильных устройствах с TFT SPI дисплеями, что способствует улучшению пользовательского опыта и расширению возможностей таких устройств.

# **Постановка задачи**

1. **Создание меню и подменю:**
   * **Создание меню:** Библиотека должна предоставлять возможность создания основного меню, состоящего из нескольких пунктов.
   * **Добавление подменю:** Необходимо реализовать возможность вложения подменю в пункты основного меню для создания иерархической структуры.
2. **Навигация и взаимодействие:**
   * **Переключение между элементами меню:** Пользователь должен иметь возможность легко переключаться между элементами меню с помощью кнопок устройства.
   * **Открытие и закрытие элементов меню:** Библиотека должна обеспечивать открытие выбранных пунктов меню для доступа к их содержимому и закрытие для возврата к предыдущему уровню.
3. **Динамические и графические элементы:**
   * **Создание динамически обновляемых элементов:** В библиотеке должна быть возможность создания элементов меню, которые могут изменяться в реальном времени, отображая актуальную информацию.
   * **Создание графических элементов в подменю:** Поддержка добавления графических элементов, таких как иконки и изображения, для улучшения визуального восприятия и интерфейса.
4. **Интерактивные функции:**
   * **Задание функций для кнопок:** Разработчик должен иметь возможность назначать функции, которые будут выполняться при нажатии на определенные элементы меню.
   * **Добавление интерактивного подменю:** Возможность создания подменю, содержащих интерактивные элементы, такие как переключатели (switches) и кнопки, для более сложного взаимодействия.
5. **Производительность и надежность:**
   * **Минимальное использование ресурсов процессора:** Библиотека должна быть оптимизирована для минимального потребления процессорных ресурсов, чтобы обеспечить плавную работу на устройствах с ограниченными вычислительными мощностями.
   * **Предотвращение графических артефактов:** При обработке графики необходимо избегать появления артефактов, чтобы интерфейс выглядел аккуратно и профессионально.
6. **Тестирование и документация:**
   * **Тестирование библиотеки:** Проведение комплексного тестирования для выявления и устранения ошибок, а также для проверки функциональности всех реализованных возможностей.
   * **Документирование кода:** Обеспечение наличия комментариев в листингах проекта для облегчения понимания и дальнейшего использования библиотеки другими разработчиками.
7. **Сохранение состояния приложения:**
   * **Навигация и сохранение состояния:** Реализация механизма сохранения состояния приложения при навигации по меню, чтобы пользователь мог вернуться к последнему использованному элементу при повторном запуске.

## **Анализ** **аналогичных решений**

Критерии анализа:

* возможность создания меню;
* возможность добавления подменю в меню;
* возможность навигации по меню;

Цели пользователя: удобство интерфейса приложения для создания, навигации и взаимодействия с элементами меню.

### **Аналог «Menu System for ILI9341\_t3 displays (Teensy) with touch or mechanical input»**

Первый аналог «Menu System for ILI9341\_t3 displays (Teensy) with touch or mechanical input» [1] преимуществом данной библиотеки является возможность создания подменю с картинками, что делает его более «дружелюбным» (Рисунок 1.1). Так же есть возможность переключится с механического взаимодействия на сенсорное.

Рисунок 1.1 – Меню с картинками

Один из плюсов приложения — возможность изменения стилевого оформления меню (Рисунок 1.2).

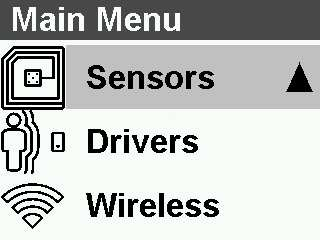


Рисунок 1.2 – Измененное стилевое оформление

Из недостатков данной библиотеки хочется выделить то, что нельзя создавать графические элементы и динамически изменяющиеся элементы.

### **Аналог «GEM»**

Вторым аналогом является «GEM» (a.k.a. Good Enough Menu) [2] данная библиотека использует монохромные дисплеи и выделяется тем что имеет возможность создавать графические элементы (Рисунок 1.3).

Рисунок 1.3 – Анимация при нажатии на кнопку

Ещё одним значимым плюсом этого приложения является отличная документация в которой описано полное руководство по использованию данной библиотеки.

Из минусов только то, что библиотека использует сторонние зависимости которые пользователю также придется устанавливать в свой проект, чтобы библиотека заработала.

## **Вывод**

Анализ аналогичных решений выявил сильные и слабые стороны существующих библиотек для создания меню на мобильных устройствах с TFT SPI дисплеями. «Menu System for ILI9341\_t3 displays (Teensy) with touch or mechanical input» позволяет создавать подменю с картинками и изменять стилевое оформление меню, что улучшает пользовательский интерфейс. Однако, эта библиотека не поддерживает создание графических и динамически изменяющихся элементов, что ограничивает её функциональность.

Библиотека «GEM (Good Enough Menu)» предоставляет возможности для создания графических элементов и обладает отличной документацией, что упрощает её использование. Тем не менее, её основной недостаток – это зависимость от сторонних компонентов, которые необходимо дополнительно устанавливать, что усложняет интеграцию и настройку.

Эти выводы подчеркивают необходимость разработки новой библиотеки, которая объединяет сильные стороны существующих решений, обеспечивая поддержку как графических, так и динамически изменяющихся элементов, минимизируя при этом зависимость от сторонних компонентов и ресурсы процессора. Новая библиотека должна быть гибкой, производительной и легкой в интеграции, что позволит создавать удобные и функциональные меню для мобильных устройств с TFT SPI дисплеями.

# **Проектирование библиотеки**

## **Проектирование классов**

В соответствии с функциональными требованиями курсового проекта были разработаны классы взаимодействие которых представлено в виде UML-диаграммы (Рисунок 2.1)

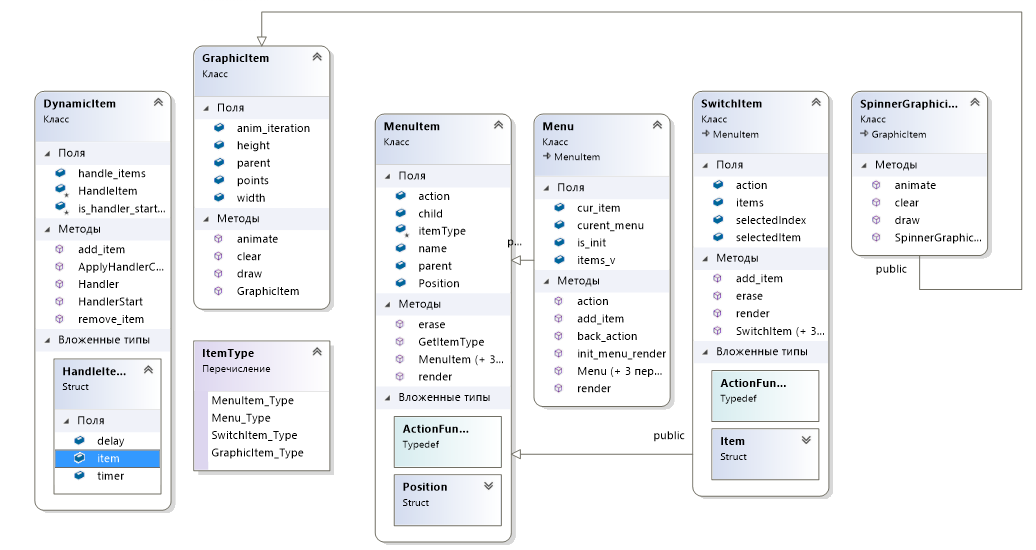


Рисунок 2.1 – Диаграмма классов

Описание структуры классов:

Класс "Menu":

* + static Menu\* current\_menu: текущее активное меню
* + int cur\_item: индекс текущего выбранного пункта меню
* + std::vector<MenuItem\*> items\_v: вектор указателей на элементы меню
* + bool is\_init: флаг инициализации меню
* + void add\_item(MenuItem\* item): метод для добавления элемента в меню
* + void render(): метод для отрисовки текущего меню
* + void action(): метод для выполнения действия при выборе пункта меню
* + void back\_action(): метод для выполнения действия при возврате к предыдущему меню
* + void init\_menu\_render(): метод для инициализации и отрисовки меню
* + Menu(const char\* name): конструктор с именем меню
* + Menu(const char\* name, MenuItem\* parent): конструктор с именем и родительским меню
* + Menu(const char\* name, MenuItem\* parent, std::vector<MenuItem\*> childs): конструктор с именем, родительским меню и вектором дочерних элементов
* + Menu(const char\* name, MenuItem\* parent, void\* action): конструктор с именем, родительским меню и функцией действия

Класс "MenuItem":

* # int itemType: тип элемента меню (MenuItem\_Type, Menu\_Type, SwitchItem\_Type, GraphicItem\_Type)
* + const char\* name: имя элемента меню
* + MenuItem\* parent: указатель на родительский элемент меню
* + MenuItem\* child: указатель на дочерний элемент меню
* + struct Position { int x, y; void Init(int x, int y); } Position: структура для хранения позиции элемента
* + typedef void (\*ActionFunc)(): тип функции действия элемента меню
* + ActionFunc action: указатель на функцию действия элемента меню
* + void render(): метод для отрисовки элемента меню
* + void erase(): метод для стирания элемента меню
* + int GetItemType(): метод для получения типа элемента меню
* + MenuItem(const char\* name): конструктор с именем элемента меню
* + MenuItem(const char\* name, MenuItem\* parent): конструктор с именем и родительским элементом меню
* + MenuItem(const char\* name, MenuItem\* parent, MenuItem\* child): конструктор с именем, родительским и дочерним элементами меню
* + MenuItem(const char\* name, MenuItem\* parent, void\* action): конструктор с именем, родительским элементом и функцией действия

Класс "DynamicItem":

* # static bool is\_handler\_started: статический флаг, указывающий на запущен ли обработчик
* # static struct HandleItemType: структура для хранения информации об элементе для обработки
* + static std::vector<HandleItemType> handle\_items: вектор для хранения элементов для обработки
* + static void add\_item(GraphicItem\* item, unsigned int delay): метод для добавления элемента для обработки
* + static int HandlerStart(): метод для запуска обработчика
* + static void Handler(): метод для обработки элементов
* + static void remove\_item(int id): метод для удаления элемента из обработки
* + static int ApplyHandlerChanges(): метод для применения изменений в обработчике

Класс "GraphicItem":

* + int width, height, anim\_iteration: ширина, высота и количество итераций анимации графического элемента
* + std::vector<short unsigned int> points: вектор точек, определяющих графический элемент
* + Menu\* parent: указатель на родительский элемент меню, к которому принадлежит графический элемент
* + void draw(): метод для отрисовки графического элемента
* + void clear(): метод для очистки графического элемента
* + void animate(): метод для анимации графического элемента
* + GraphicItem(int w, int h): конструктор с заданием ширины и высоты графического элемента

Класс "SwitchItem":

* + struct Item : структура для хранения информации о пунктах переключателя
* + typedef void (\*ActionFunc)(SwitchItem\*): тип функции действия переключателя
* + ActionFunc action: указатель на функцию действия переключателя
* + Item\* selectedItem: указатель на выбранный пункт переключателя
* + int selectedIndex: индекс выбранного пункта переключателя
* + std::vector<Item\*> items: вектор для хранения пунктов переключателя
* + void erase(): метод для стирания переключателя
* + void render(): метод для отрисовки переключателя
* + void add\_item(Item\* item): метод для добавления пункта в переключатель
* + SwitchItem(const char\* name): конструктор с именем переключателя
* + SwitchItem(const char\* name, MenuItem\* parent): конструктор с именем и родительским элементом меню
* + SwitchItem(const char\* name, MenuItem\* parent, MenuItem\* child): конструктор с именем, родительским и дочерним элементами меню
* + SwitchItem(const char\* name, MenuItem\* parent, void\* action): конструктор с именем, родительским элементом меню и функцией действия

Класс "TriangleGraphicItem":

* + int centerX, centerY: координаты центра треугольника
* + int size: размер треугольника
* + float angle: угол поворота треугольника
* + int\* color: указатель на цвет треугольника
* + int x1\_t, y1\_t, x2\_t, y2\_t, x3\_t, y3\_t: координаты вершин треугольника
* + void draw(): метод для отрисовки треугольника
* + void clear(): метод для очистки треугольника
* + void animate(): метод для анимации треугольника
* + TriangleGraphicItem(int x, int y, int s, int w, int h, Menu\* parent, int\* color): конструктор с указанием координат, размера, ширины, высоты, родительского элемента и цвета

## **Описание средств разработки**

Для создания данной библиотеки были использованы следующие инструменты:

1. Arduino IDE: Выбор Arduino IDE обусловлен его простотой в использовании, обширной документацией и широкой поддержкой сообщества. Эта среда предоставляет простой интерфейс для написания, компиляции и загрузки кода на платформу Arduino, что делает ее идеальным выбором для разработчиков всех уровней.
2. C++: Для написания библиотеки был выбран язык программирования C++ из-за его высокой производительности, богатства возможностей и широкой поддержки в различных средах разработки. C++ позволяет создавать эффективные и масштабируемые программы, что особенно важно при работе с микроконтроллерами и встраиваемыми системами.

Использование Arduino IDE и C++ обеспечило эффективное развитие библиотеки для создания меню под кнопочные мобильные устройства, использующие TFT SPI дисплеи. Эти инструменты предоставили удобную среду разработки, обеспечивая легкость создания, тестирования и отладки кода, а также гарантируя высокую производительность и надежность разработанной библиотеки.

## **Вывод**

Процесс разработки библиотеки для создания меню под кнопочные мобильные устройства, использующие TFT SPI дисплеи, был осуществлен с использованием Arduino IDE и языка программирования C++.

Основными средствами разработки были Arduino IDE и язык C++, выбранные в связи с их простотой использования, широкой поддержкой и возможностью создания эффективного и масштабируемого программного обеспечения. Arduino IDE обеспечил удобную среду разработки, а язык C++ — высокую производительность и богатство функциональных возможностей.

Проектирование библиотеки включало в себя определение требований и функциональности, выбор архитектуры и структуры данных, а также разработку методов управления и взаимодействия с элементами меню. Процесс проектирования был тесно связан с анализом требований пользователей и анализом аналогичных решений, что позволило создать библиотеку, отвечающую основным потребностям и ожиданиям.

# **Реализация библиотеки**

Раздел "Реализация библиотеки" посвящен описанию процесса разработки библиотеки для создания меню под кнопочные мобильные устройства, использующие TFT SPI дисплеи. В этом разделе будет представлено описание структуры и основных компонентов библиотеки, а также принципы ее организации и функционирования.

## **Разработка библиотеки**

При разработке библиотеки для создания меню под кнопочные мобильные устройства, использующие TFT SPI дисплеи, была использована структура, ориентированная на эффективное управление и взаимодействие с элементами интерфейса. Каждый компонент библиотеки имеет свою собственную задачу и функцию, что позволяет удобно организовывать работу с меню на устройствах с ограниченными ресурсами.

Структура библиотеки включает в себя следующие основные компоненты:

* **Menu.h:** Основной класс библиотеки, предоставляющий интерфейс для создания и управления меню на TFT SPI дисплеях. Включает методы для добавления элементов меню, управления навигацией и выполнения действий при выборе пунктов меню.
* **MenuItem.h:** Базовый класс для всех элементов меню. Определяет общие атрибуты и методы, такие как отрисовка элемента, получение типа элемента и управление вложенными элементами.
* **DynamicItem.h:** Класс для создания динамически обновляемых элементов меню. Предоставляет функции для добавления элементов с заданным временем обновления и методы для их обработки.
* **GraphicItem.h:** Класс для создания графических элементов в меню. Позволяет определять форму, размеры и анимацию графических объектов.
* **SwitchItem.h:** Класс для создания переключаемых элементов меню. Предоставляет возможность выбора одного из нескольких вариантов и выполнения соответствующих действий.
* **TriangleGraphicItem.h:** Дополнительный класс для создания треугольных графических элементов. Позволяет рисовать и анимировать треугольники на дисплее.

Эта структура обеспечивает гибкость и расширяемость библиотеки, позволяя легко добавлять новые элементы меню и функции в соответствии с требованиями пользователей. Каждый компонент библиотеки выполняет свою специфическую задачу, что делает код модульным и легко поддерживаемым.

Код реализации классов библиотеки представлен в Листинге A.

## **Вывод**

Раздел "Разработка библиотеки" играет ключевую роль в понимании архитектуры и структуры созданной библиотеки. Благодаря детальному описанию каждого компонента библиотеки становится понятно, как они взаимодействуют друг с другом и какую функциональность предоставляют.

Основной принцип, лежащий в основе разработки библиотеки, заключается в обеспечении простоты использования и гибкости настройки. Каждый класс и метод были спроектированы с учетом этих принципов, что позволяет разработчикам легко интегрировать библиотеку в свои проекты и настраивать ее под свои потребности.

Важным аспектом разработки была модульность кода. Каждый компонент библиотеки является независимым модулем, что обеспечивает удобство тестирования и поддержки. Это также делает библиотеку более гибкой и масштабируемой, позволяя легко добавлять новую функциональность и расширять возможности.  
 Архитектура библиотеки была спроектирована с учетом не только текущих потребностей, но и возможных будущих расширений. Это обеспечивает гибкость и масштабируемость решения, что является важным преимуществом при работе над различными проектами.

Принцип модульности и независимости компонентов позволяет разработчикам легко настраивать и расширять функциональность библиотеки, а также обеспечивает удобство ее поддержки и обновлений в долгосрочной перспективе.

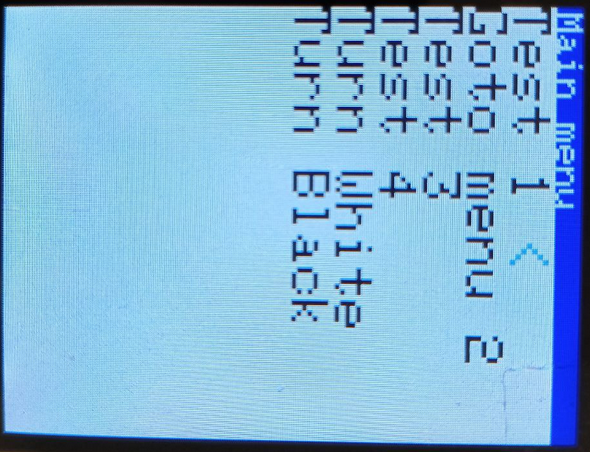
Раздел "Разработка библиотеки" является основой для понимания работы библиотеки и ее применения в практических проектах. Полученные знания об архитектуре и структуре библиотеки помогут разработчикам эффективно использовать ее в своих проектах и создавать удобные и интуитивно понятные пользовательские интерфейсы.

# **Тестирование библиотеки**

Целью раздела "Тестирование библиотеки" является проверка правильности работы каждого компонента библиотеки, а также оценка его производительности и надежности. Тестирование позволяет выявить потенциальные проблемы и улучшить качество библиотеки перед ее публикацией и использованием в реальных проектах.

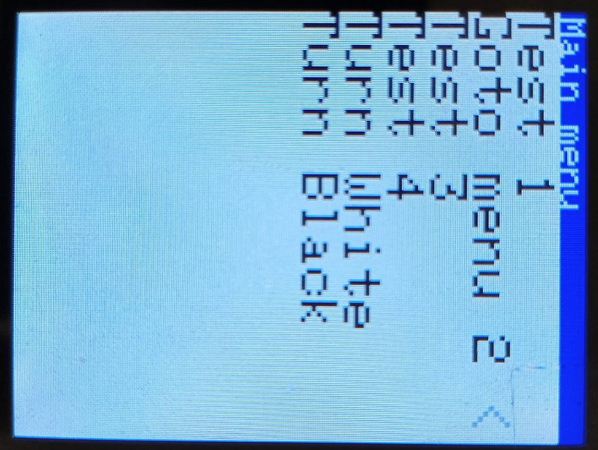
В данном разделе будут представлены результаты тестирования ключевых функциональных возможностей библиотеки, а также описаны методы и средства, использованные для проведения тестов. Это позволит убедиться в надежности и стабильности работы библиотеки перед ее применением в практических задачах.

Рисунок 4.1 ­ Главное меню



При навигации по меню изменяется положение голубого указателя с левой части надписи показано на рисунке 4.2.

Рисунок 4.2 – Тестирование навигации



Так же пользователь может задать индивидуальное действие при нажатии на кнопку, в примере ниже будет действие которое изменяет цветовую палитру на темную (Рисунок 4.3).

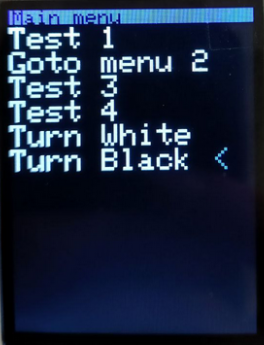


Рисунок 4.3 – Тестирование выполнения действий при нажатии на кнопку

На рисунке 4.4 представлено сразу две функции, переход из меню в другое меню, а также на в низу рисунка представлена одна из реализации графического элемента с динамическим обновлением.

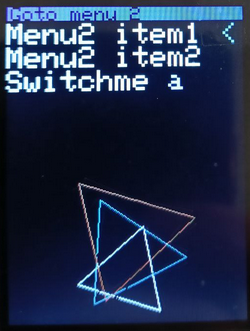


Рисунок 4.4 – подменю с графическими элементами

## **Вывод**

В разделе "Тестирование библиотеки" была проведена проверка правильности работы ключевых компонентов библиотеки и оценена их производительность и надежность. Тестирование позволило выявить потенциальные проблемы и улучшить качество библиотеки перед ее использованием в реальных проектах.

Результаты тестирования показали, что библиотека успешно прошла проверку на соответствие функциональным требованиям и демонстрирует стабильную работу в различных сценариях использования. Были продемонстрированы ключевые возможности библиотеки, такие как навигация по меню, выполнение действий при нажатии на кнопку, а также использование графических элементов с динамическим обновлением.

Таким образом, проведенное тестирование подтвердило надежность и стабильность работы библиотеки, что делает ее готовой к применению в практических задачах разработки мобильных приложений для устройств с TFT SPI дисплеями.

# **Руководство пользователя**

В данном разделе будет описано руководство по созданию и добавлению меню используя библиотеку.

Для создания меню используется класс Menu и его конструкторы, чтобы добавить какой-либо элемент унаследованный от MenuItem в созданное меню нужно воспользоваться метод класса Menu add\_item (Листинг 5.1).

|  |
| --- |
| Menu\* main\_menu = new Menu("Main menu"); MenuItem\* main\_menu\_i1 = new MenuItem("Test 1");  Menu\* main\_menu\_i2 = new Menu("Goto menu 2");  MenuItem\* main\_menu\_i3 = new MenuItem("Test 3");  MenuItem\* main\_menu\_i4 = new MenuItem("Test 4");  MenuItem\* main\_menu\_i5 = new MenuItem("Turn White");  MenuItem\* main\_menu\_i6 = new MenuItem("Turn Black");  main\_menu->add\_item(main\_menu\_i1);  main\_menu->add\_item(main\_menu\_i2);  main\_menu->add\_item(main\_menu\_i3);  main\_menu->add\_item(main\_menu\_i4);  main\_menu->add\_item(main\_menu\_i5);  main\_menu->add\_item(main\_menu\_i6);  main\_menu\_i5->action = TurnWhiteAction;  main\_menu\_i6->action = TurnBlackAction;  Menu::curent\_menu = main\_menu; |

Листинг 5.1 – Создание меню и добавление подменю

Для создания динамического элемента нужно создать экземпляр класса унаследованного от GraphicItem, добавть этот экземпляр в статическую переменную handle\_items из класса DynamicItem и вызвать метод HandlerStart у тогоже класса DynamicItem (Листинг 5.2).

|  |
| --- |
| TriangleGraphicItem\* triangle1 = new TriangleGraphicItem(tft.width()/2, tft.height()-50, 50, 0, 0, main\_menu\_i2, &TRIANGLE\_COLOR\_1);    DynamicItem::add\_item(triangle1, 100);    DynamicItem::HandlerStart(); |

Листинг 5.2 Добавление динамического элемента в обработчик

# **Заключение**

В ходе разработки и реализации библиотеки для управления меню на устройствах с TFT SPI дисплеями были достигнуты следующие результаты. Библиотека предоставляет удобный и гибкий инструмент для создания и управления меню, что позволяет разработчикам легко добавлять интерфейс пользователя в свои проекты и обеспечивать удобное взаимодействие с устройством.

Целью проекта было создание мощной и легко настраиваемой библиотеки, которая позволила бы разработчикам быстро и эффективно создавать меню для своих проектов. Благодаря использованию стандартных библиотек и синтаксиса C++, библиотека обладает высокой гибкостью и расширяемостью, что делает ее привлекательным выбором для различных приложений.

Важным аспектом разработки библиотеки было обеспечение ее надежности и производительности. В процессе тестирования была проверена корректность работы ключевых компонентов библиотеки, а также оценена ее производительность в различных условиях использования. Полученные результаты подтвердили правильность работы библиотеки и ее способность обеспечивать высокую производительность при работе с TFT SPI дисплеями.

В заключение, разработка библиотеки для управления меню на устройствах с TFT SPI дисплеями является важным шагом в обеспечении удобного и интуитивно понятного пользовательского интерфейса для различных проектов. Благодаря ее использованию, разработчики могут сосредоточиться на реализации функциональности своих проектов, не тратя время на создание пользовательского интерфейса с нуля.

# **Список использованных источников**

1. Menu System for ILI9341\_t3 displays (Teensy) with touch or mechanical input [Электронный ресурс]. – Режим доступа: https://github.com/KrisKasprzak/ILI9341\_t3\_Menu – Дата доступа: 01.05.2024
2. TFT\_eSPI [Электронный ресурс]. – Режим доступа: https://github.com/Bodmer/TFT\_eSPI – Дата доступа: 16.04.2024
3. Solutions on Stackoverflow [Электронный ресурс]. – Электронные данные. – Режим доступа: [https://stackoverflow.com.](https://stackoverflow.com./) – Дата доступа: 20.04.2024;
4. Espressif [Электронный ресурс]. – Режим доступа: https://docs.espressif.com/projects/esp-idf/en/stable/esp32s2/get-started/index.html – Дата доступа: 20.03.2024

# **Приложение А**

|  |
| --- |
| #include "TFT\_eSPI\_HEADERS.h" #include "Menu.h"  #include "DynamicItem.h"  #include "TriangleGraphicItem.h"  #include "SwitchItem.h"  #include "CONFIG.h"  TFT\_eSPI tft = TFT\_eSPI(320, 240);  std::mutex tftMutex;  #define BTN\_PIN\_DOWN 22  #define BTN\_PIN\_UP 21  #define BTN\_PIN\_SEL 27  #define BTN\_PIN\_BACK 26  void TurnWhiteAction() {    FONT\_COLOR = 0x000;    SCREEN\_COLOR = 0xffffff;    Menu::curent\_menu->init\_menu\_render();  }  void TurnBlackAction() {    FONT\_COLOR = 0xffffff;    SCREEN\_COLOR = 0x000;    Menu::curent\_menu->init\_menu\_render();  }  void setup() {    digitalWrite(TFT\_CS, LOW);    Serial.begin(115200);    pinMode(BTN\_PIN\_UP, INPUT\_PULLUP);    pinMode(BTN\_PIN\_DOWN, INPUT\_PULLUP);    pinMode(BTN\_PIN\_SEL, INPUT\_PULLUP);    pinMode(BTN\_PIN\_BACK, INPUT\_PULLUP);    tft.begin();    tft.setRotation(3);    Menu\* main\_menu = new Menu("Main menu");    MenuItem\* main\_menu\_i1 = new MenuItem("Test 1");    Menu\* main\_menu\_i2 = new Menu("Goto menu 2");    MenuItem\* main\_menu\_i3 = new MenuItem("Test 3");    MenuItem\* main\_menu\_i4 = new MenuItem("Test 4");    MenuItem\* main\_menu\_i5 = new MenuItem("Turn White");    MenuItem\* main\_menu\_i6 = new MenuItem("Turn Black");    main\_menu->add\_item(main\_menu\_i1);    main\_menu->add\_item(main\_menu\_i2);    main\_menu->add\_item(main\_menu\_i3);    main\_menu->add\_item(main\_menu\_i4);    main\_menu->add\_item(main\_menu\_i5);    main\_menu->add\_item(main\_menu\_i6);    main\_menu\_i5->action = TurnWhiteAction;    main\_menu\_i6->action = TurnBlackAction;    Menu::curent\_menu = main\_menu;      MenuItem\* main\_menu2\_i1 = new MenuItem("Menu2 item1");    MenuItem\* main\_menu2\_i2 = new MenuItem("Menu2 item2");    SwitchItem\* main\_menu2\_i3 = new SwitchItem("Switchme");    main\_menu\_i2->add\_item(main\_menu2\_i1);    main\_menu\_i2->add\_item(main\_menu2\_i2);    main\_menu\_i2->add\_item(main\_menu2\_i3);    SwitchItem::Item\* switch\_item1 =new SwitchItem::Item;    switch\_item1->name = "a";    switch\_item1->value = 0;    main\_menu2\_i3->add\_item(switch\_item1);    Menu::curent\_menu->init\_menu\_render();    TriangleGraphicItem\* triangle1 = new TriangleGraphicItem(tft.width()/2, tft.height()-50, 50, 0, 0, main\_menu\_i2, &TRIANGLE\_COLOR\_1);    TriangleGraphicItem\* triangle2 = new TriangleGraphicItem(tft.width()/2, tft.height()-70, 60, 0, 0, main\_menu\_i2, &TRIANGLE\_COLOR\_2);    TriangleGraphicItem\* triangle3 = new TriangleGraphicItem(tft.width()/2, tft.height()-90, 70, 0, 0, main\_menu\_i2, &TRIANGLE\_COLOR\_3);    DynamicItem::add\_item(triangle1, 100);    DynamicItem::add\_item(triangle2, 80);    DynamicItem::add\_item(triangle3, 70);    DynamicItem::HandlerStart();  }  void loop() {    tftMutex.lock();    if (digitalRead(BTN\_PIN\_DOWN) == LOW) {      Menu::curent\_menu->cur\_item++;      if (Menu::curent\_menu->cur\_item == Menu::curent\_menu->items\_v.size()) Menu::curent\_menu->cur\_item = 0;      tft.setTextColor(FONT\_COLOR);      tft.setCursor(0,17);      tft.setTextSize(3);      for (int i = 0; i < Menu::curent\_menu->items\_v.size(); i++) {        if (i == Menu::curent\_menu->cur\_item) {          if (i > 0) Menu::curent\_menu->items\_v[i-1]->erase();          else if (i == 0) Menu::curent\_menu->items\_v[Menu::curent\_menu->items\_v.size()-1]->erase();          tft.setCursor(Menu::curent\_menu->items\_v[i]->Position.x,Menu::curent\_menu->items\_v[i]->Position.y);          if (Menu::curent\_menu->items\_v[i]->GetItemType() == 2) ((SwitchItem\*)Menu::curent\_menu->items\_v[i])->render(); // if is switch type          else Menu::curent\_menu->items\_v[i]->render();          tft.setTextColor(0x765442);          tft.println(" <");          tft.setTextColor(FONT\_COLOR);        }      }      Serial.println("down\_btn");    }    if (digitalRead(BTN\_PIN\_UP) == LOW) {      Menu::curent\_menu->cur\_item--;      if (Menu::curent\_menu->cur\_item < 0) Menu::curent\_menu->cur\_item = Menu::curent\_menu->items\_v.size()-1;      tft.setTextColor(FONT\_COLOR);      tft.setCursor(0,17);      tft.setTextSize(3);      for (int i = 0; i < Menu::curent\_menu->items\_v.size(); i++) {        if (i == Menu::curent\_menu->cur\_item) {          if (i < Menu::curent\_menu->items\_v.size()-1) Menu::curent\_menu->items\_v[i+1]->erase();          else if (i == Menu::curent\_menu->items\_v.size()-1) Menu::curent\_menu->items\_v[0]->erase();          tft.setCursor(Menu::curent\_menu->items\_v[i]->Position.x,Menu::curent\_menu->items\_v[i]->Position.y);          Menu::curent\_menu->items\_v[i]->render();          tft.setTextColor(0x765442);          tft.println(" <");          tft.setTextColor(FONT\_COLOR);        }      }      Serial.println("up\_btn");    }    if (digitalRead(BTN\_PIN\_SEL) == LOW) {      if (Menu::curent\_menu->items\_v[Menu::curent\_menu->cur\_item]->GetItemType() == 1) ((Menu\*)Menu::curent\_menu->items\_v[Menu::curent\_menu->cur\_item])->action();// menu type      else if (Menu::curent\_menu->items\_v[Menu::curent\_menu->cur\_item]->action != nullptr) Menu::curent\_menu->items\_v[Menu::curent\_menu->cur\_item]->action();      Serial.print("sel\_btn: ");      Serial.println(Menu::curent\_menu->items\_v[Menu::curent\_menu->cur\_item]->name);    }    if (digitalRead(BTN\_PIN\_BACK) == LOW) {      if (Menu::curent\_menu->parent != nullptr) Menu::curent\_menu->back\_action();// menu type      Serial.println("back\_btn");    }    tftMutex.unlock();    delay(100);  } |