

Министерство науки и высшего образования РФ

Пензенский государственный университет

Кафедра «Вычислительная техника»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

по курсу «Программирование»

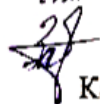
на тему «Разработка программы сложной структуры методом нисходящего
программирования. Программное обеспечение для управления роботом
доставщиком»

Выполнил: ст. гр. 25BVB4

10.06.26
ст.л.

Волосунин Е. В.

Принял:



Кандидат т.н. Токарев А. Н.

Пенза 2026

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет Вычислительной техники

Кафедра "Вычислительная техника"

"УТВЕРЖДАЮ"



Зав. кафедрой ВТ

М.А. Митрохин

«__»__ 20__ г

ЗАДАНИЕ

на курсовое проектирование по курсу

Программирование

Студенту Волошин Егор Вячеславович Группа 25ВВВ4

Тема проекта: *Разработка программы сложной структуры методом нисходящего программирования.*

Исходные данные (технические требования) на проектирование

*Реализация программного обеспечения для
управления роботом-госставщиком.
Требования: реализация ридер-устройства, веб-интерфейс, хранение
карты навигации с возможностью замены.*

Обязательные требования к программе:

1. Многомодульность.
2. Использование сложных типов данных – структур, списков, файлов.
3. Режим работы видеосистемы – текстовый / графический.
4. Устройство ввода информации – клавиатура, мышь.
5. Пользовательский интерфейс должен быть построен на основе меню и панели инструментов.
6. Наличие заставки.
7. Операционная система MS Windows.
8. Язык программирования – Си и Ассемблер.
9. Среда разработки ПО -- Microsoft Visual Studio.

Объем работы по курсу

1. Расчетная часть

Разработка программы.

2. Графическая часть

Схема данных, схема ресурсов системы, схема работы системы, иерархическая структура программы, схема взаимодействия программы.

3. Экспериментальная часть

Отладка программы.

Срок выполнения проекта по разделам

В соответствии с графиком.

Дата выдачи задания “ 16 ” 02 . 26

Дата защиты проекта “ ”

Руководитель _____

Задание получил “ 25 ” 02 2026г.

Студент Воз Волошин Егор Вячеславович

Оглавление

Введение	6
Постановка задачи	8
Выбор решения.....	10
Описание разработки программы.....	12
Отладка и тестирование	14
Описание программы.....	15
Руководство пользователя.....	21
Заключение	25
Список используемых источников.....	26
Приложение В. Результат работы программы.	27
Приложение В.1 - скриншот веб-страницы.	27
Приложение С. Результат работы программы.	28
/spiffs/map.json	28
Приложение D. Результат работы программы	29

Введение

Встраиваемые системы и автономная робототехника являются одними из самых наглядных примеров программ, в которых одновременно используются алгоритмы обработки данных в реальном времени, работа с аппаратными прерываниями, взаимодействие с пользователем по сети и организация кода по модулям. Даже базовая задача движения робота по линии требует продуманной архитектуры: необходимо непрерывно считывать состояние сенсоров, вычислять ошибку отклонения, применять математический аппарат ПД-регулятора, управлять мощностью двигателей и параллельно обслуживать графический веб-интерфейс пользователя.

В данной курсовой работе рассматривается разработка программного обеспечения сложной структуры для робота-доставщика. Целевой аппаратной платформой выступает микроконтроллер ESP32-S3. Робот способен автономно двигаться по размеченной линии, распознавать перекрестки и конечные точки маршрута. В программе реализован функционал дистанционного управления, динамической настройки ПД-регулятора и построения топологической карты маршрута (графа) через встроенный веб-интерфейс.

Особенностью программы является использование многомодульной структуры на языке C и ассемблере Xtensa (ASM). Отдельные файлы отвечают за аппаратную периферию (моторы, сенсоры, АЦП батареи), логику движения, файловую систему и сетевой стек. Такой подход полностью соответствует методу нисходящего программирования: сначала была определена общая логика операционной системы робота, затем она разбита на независимые подзадачи, каждая из которых реализована отдельным компонентом.

Ввиду специфики микроконтроллерной архитектуры, разработка велась не в классической ОС Windows с использованием Microsoft Visual Studio, а в текстовом редакторе VIM с применением официального

фреймворка ESP-IDF и операционной системы реального времени FreeRTOS. Это позволило добиться жесткого контроля над таймингами. Режим работы приложения разделен на два уровня: текстовый (вывод системных журналов ESP_LOG и заставки в терминал через UART) и графический (HTML5 Canvas в браузере). Часть логики обработки показаний АЦП ,батареи реализована с применением вставок ассемблера для аппаратной фильтрации данных.

Разрабатываемая программа предназначена для демонстрации практических навыков проектирования сложной структуры ПО, работы со структурами данных, энергонезависимой памятью (файловая система SPIFFS), сетевыми протоколами (HTTP/JSON) и низкоуровневым управлением оборудованием.

Постановка задачи

Необходимо разработать программное обеспечение для управления автономным роботом-доставщиком. Программа должна базироваться на ОС реального времени (FreeRTOS) и включать в себя алгоритмы управления двигателями, систему технического зрения (массив цифровых датчиков линии), а также графический пользовательский веб-интерфейс.

Основная задача робота заключается в удержании траектории движения с помощью пропорционально-дифференциального (ПД) регулятора и детекции типов перекрестков (повороты, кресты, тупики).

В ходе разработки необходимо определить сложные типы данных: структуры для хранения показаний массива сенсоров, массивы объектов для узлов и ребер топологического графа карты, а также глобальные переменные состояния конечного автомата робота (State Machine).

Программа должна быть разделена на слабо связанные модули. В корневом модуле `main.c` должна происходить инициализация подсистем и запуск главного цикла. Модуль `motion.c` должен отвечать за математику удержания линии. Модули `sensors.c`, `motors.c` и `battery.c` должны инкапсулировать работу с аппаратной частью (GPIO, ШИМ, АЦП). Модуль `webserver.c` должен поднимать точку доступа Wi-Fi и HTTP-сервер для отдачи графического интерфейса.

В качестве устройств ввода-вывода информации выступает беспроводной веб-интерфейс (управление мышью/сенсором на устройстве пользователя). Пользователь должен иметь возможность через браузер визуально строить графовую карту помещения, задавать коэффициенты ПД-регулятора на лету, постоянную скорость и отправлять команды запуска/остановки.

Построенная пользователем карта должна преобразовываться в формат JSON и сохраняться в энергонезависимую память микроконтроллера (файловая система SPIFFS) с возможностью перезаписи.

В результате выполнения задачи должно быть создано отказоустойчивое встроенное ПО с веб-меню, графической панелью инструментов, сохранением данных и модульной структурой исходного кода, написанное методом нисходящего проектирования.

Выбор решения

Для реализации выбран язык программирования C, так как он является стандартом в отрасли встраиваемых систем (Embedded Systems) и позволяет напрямую работать с регистрами микроконтроллера, прерываниями и памятью. В качестве среды программирования применялся консольный редактор VIM, а сборка проекта осуществлялась с помощью системы CMake.

В качестве фундаментального решения был выбран фреймворк ESP-IDF (Espressif IoT Development Framework). В отличие от любительской среды Arduino, ESP-IDF предоставляет полный доступ к ядру FreeRTOS, что гарантирует точное выполнение задач по расписанию (что критически важно для ПИД-регулятора) и эффективную работу с сетевым стеком lwIP без блокировки основного цикла управления.

Аппаратная платформа включает 5 основных датчиков линии. Ошибка отклонения вычисляется путем присвоения каждому датчику весового коэффициента (от -20 до +20). На основе этой ошибки ПД-регулятор корректирует ШИМ-сигнал левого и правого бортов робота.

Для компенсации разряда аккумуляторной батареи (4S Li-Ion) реализован модуль чтения АЦП. Для исключения шумовых всплесков на АЦП, способных привести к некорректному расчету ШИМ, в код на языке C внедрена ассемблерная вставка (архитектура Xtensa), выполняющая аппаратное ограничение (clamping) сырого значения в пределах от 0 до 4095 тактов процессора.

Графический интерфейс решено реализовать с помощью встроенного в ESP32 веб-сервера. Веб-страница содержит элемент HTML5 Canvas, реализующий режим графического редактора. Вместо громоздких библиотек написан легковесный скрипт на языке JavaScript, который обрабатывает клики мыши, рисует узлы графа и связи между ними, а при сохранении формирует JSON-файл.

Сохранение карты реализовано на базе файловой системы SPIFFS (SPI Flash File System). Под нее выделен отдельный логический раздел во Flash-памяти микроконтроллера (2 МБ), что позволяет файлу карты переживать перезагрузки и отключения питания.

Выбранное решение удобно тем, что логика движения, аппаратные абстракции и сетевые интерфейсы полностью разнесены по модулям.

Описание разработки программы

Разработка программы выполнялась методом нисходящего программирования. На верхнем уровне была сформулирована абстрактная модель: инициализация, цикл следования по линии, реакция на перекресток, ожидание команды по сети. Затем абстрактные функции детализировались и выносились в отдельные компоненты (папка components).

Список основных модулей:

- `sensors.c` – отвечает за инициализацию датчиков линии и чтение их состояния. Была создана структура `sensors_data_t`, объединяющая показания всех датчиков.
- `motors.c` – осуществляет настройку и управление драйверами двигателей. Создана структура `motor_ctx_t` для настройки ШИМ-генератора для 2-х драйверов двигателя BTS7960, написана функция `motors_init` для настройки аппаратного таймера и генератора ШИМ, реализованы функции `driver_on`, `driver_of`, `motors_stop` для управления драйверами двигателя, а также функция `motors_set_voltage_scale` для вычисления корректной скорости двигателей исходя из уровня заряда батареи.
- `battery.c` – настройка АЦП, калибровка напряжений, ассемблерная логика очистки данных и вычисление корректирующего множителя для моторов;
- `main.c` – точка входа, инициализация подсистем, текстовая заставка в лог, основной цикл управления роботом;
- `motion.c` – реализация ПИД-регулятора, алгоритмы плавного старта/торможения, логика точного поворота на 90 и 180 градусов с опорой на состояние центрального датчика. Функция `motion_follow_line()` рассчитывает ошибку по весам, применяет коэффициенты K_p и K_d и выдает команды на моторы.

- `webserver.c`: настройка Wi-Fi точки доступа, обработчики HTTP GET/POST запросов, хранение строкового константного массива с кодом HTML/JS графического интерфейса. Поднимает точку доступа DeliveryRobot и веб-страницу управления роботом. Для реализации панели инструментов и меню использованы стандартные элементы HTML (`<button>`, `<select>`), а для отрисовки карты — контекст Canvas 2D. Интегрированы модули работы с файлами. Функция `storage_init()` монтирует SPIFFS, а HTTP-обработчик `save_map_handler()` принимает JSON от браузера, конвертирует его в массив символов и записывает в файл `/spiffs/map.json` с помощью стандартных C-функций `fopen` и `fprintf`.

Разработка программы выполнялась в следующем порядке:

- Реализация точки входа в `main.c`
- Интеграция датчиков в `sensors.c`
- Считывание и обработка напряжения с батареи для корректной работы двигателей в `battery.c`
- Реализация работы драйверов двигателя в `motors.c`
- Написание логики работы пид-регулятора в `motion.c`
- Реализация веб странички для управления роботом и работы с файлами в `webserver.c`

Порядок разработки подтверждает нисходящий подход: от определения базовых структур (узлы, датчики, АЦП батареи, драйверы моторов) к высокоуровневому управлению (ПИД) и, наконец, к интерфейсу пользователя (Веб).

Отладка и тестирование

Отладка программного обеспечения выполнялась непосредственно на реальном оборудовании с применением отладчика GDB и системы текстового журналирования ESP_LOG. В отличие от стандартного printf, макросы ESP_LOGI и ESP_LOGE выводят в терминал метки времени и имена модулей, что позволило реализовать полнофункциональный текстовый режим контроля состояния робота.

Для проверки факта сохранения карты навигации был написан тестовый блок в main.c, который при перезагрузке микроконтроллера открывает файл map.json и посимвольно выводит содержимое JSON-графа в терминал.

В ходе тестирования, проводившегося во время разработки, было выявлено и исправлено огромное количество проблем, включая не только программные, но и аппаратные проблемы, возникавшие на реальном оборудовании (робот), а также пересечение этих проблем.

Результатом тестирования стало подтверждение стабильной работы ПО робота: переполнение памяти не происходит, ПИД-регулятор работает предсказуемо, карта корректно рисуется в графическом режиме и записывается во Flash-память.

Описание программы

Скомпилированная прошивка загружается во Flash-память ESP32-S3. После подачи питания операционная система поднимает сеть Wi-Fi и выводит текстовую заставку инициализации в последовательный порт. Для взаимодействия с программой пользователь использует браузер. В таблице 1 описаны элементы управления панелью инструментов на веб-странице.

Таблица 1 – Описание элементов управления

Действие	Описание
Нажатие кнопки “START”	Запускается цикл опроса сенсоров и вызова ПИД-регулятора. Робот начинает движение.
Нажатие кнопки “STOP”	Робот плавно снижает скорость моторов до нуля и переходит в режим ожидания.
Ввод speed (base и turn), KP, KD	Изменение соответственно скорости (базовое движение по прямой и скорость поворота) и коэффициентов ПИД-регулятора - пропорционального и дифференциального.
Нажатие кнопки “APPLY NEW SETTINGS”	Применение внесенных изменений скорости и коэффициентов ПИД-регулятора на ESP32.
Выбор режима “Add Node”	Включение графического режима добавления узлов графа на холст, последующее нажатие на холст в любую точку добавит туда выбранный узел.
Выбор режима “Connect”	Включение графического режима добавления рёбер между узлами, нажатие на два узла добавит между ними прямой отрезок.
Нажатие кнопки “Clear”	Очищает холст с картой.
Выбор режима “Connect”	Включение графического режима добавления рёбер между узлами, нажатие на два узла добавит между ними прямой отрезок.
Нажатие кнопки “SAVE MAP TO ESP32”	Сохранение карты в памяти ESP32.

Схема взаимодействия программы:

- main.c циклично опрашивает флаги;
- webserver.c асинхронно обрабатывает запросы пользователя и меняет флаги;
- sensors.c считывает показания датчиков линии;
- motion.c и battery.c обеспечивают расчеты скорости, motion.c вычисляет ошибку отклонения от линии и регулирует скорость моторов;
- motors.c применяет рассчитанные параметры к физическим приводам.

На рисунке 1 представлена схема взаимодействия программы.

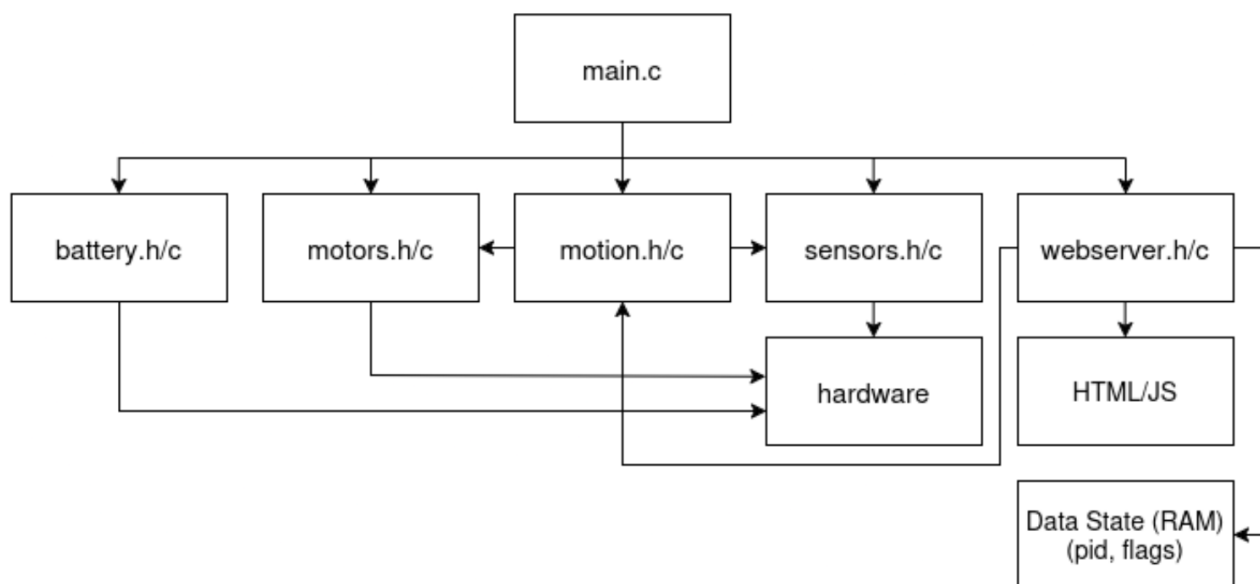


Рисунок 1 – Схема взаимодействия программы

На рисунке 2 представлена схема данных.

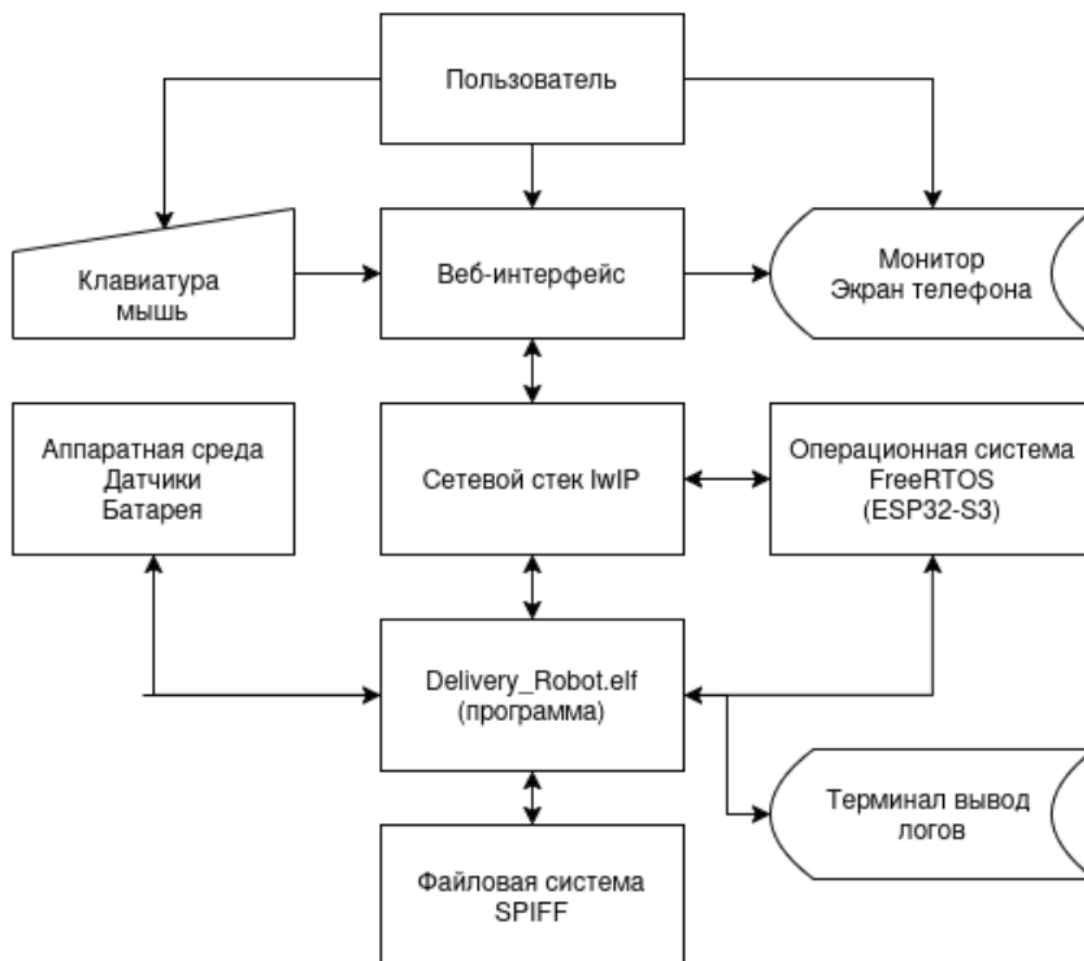


Рисунок 2 – Схема данных

Блок схема main.c представлена на рисунке 3 и рисунке 4.

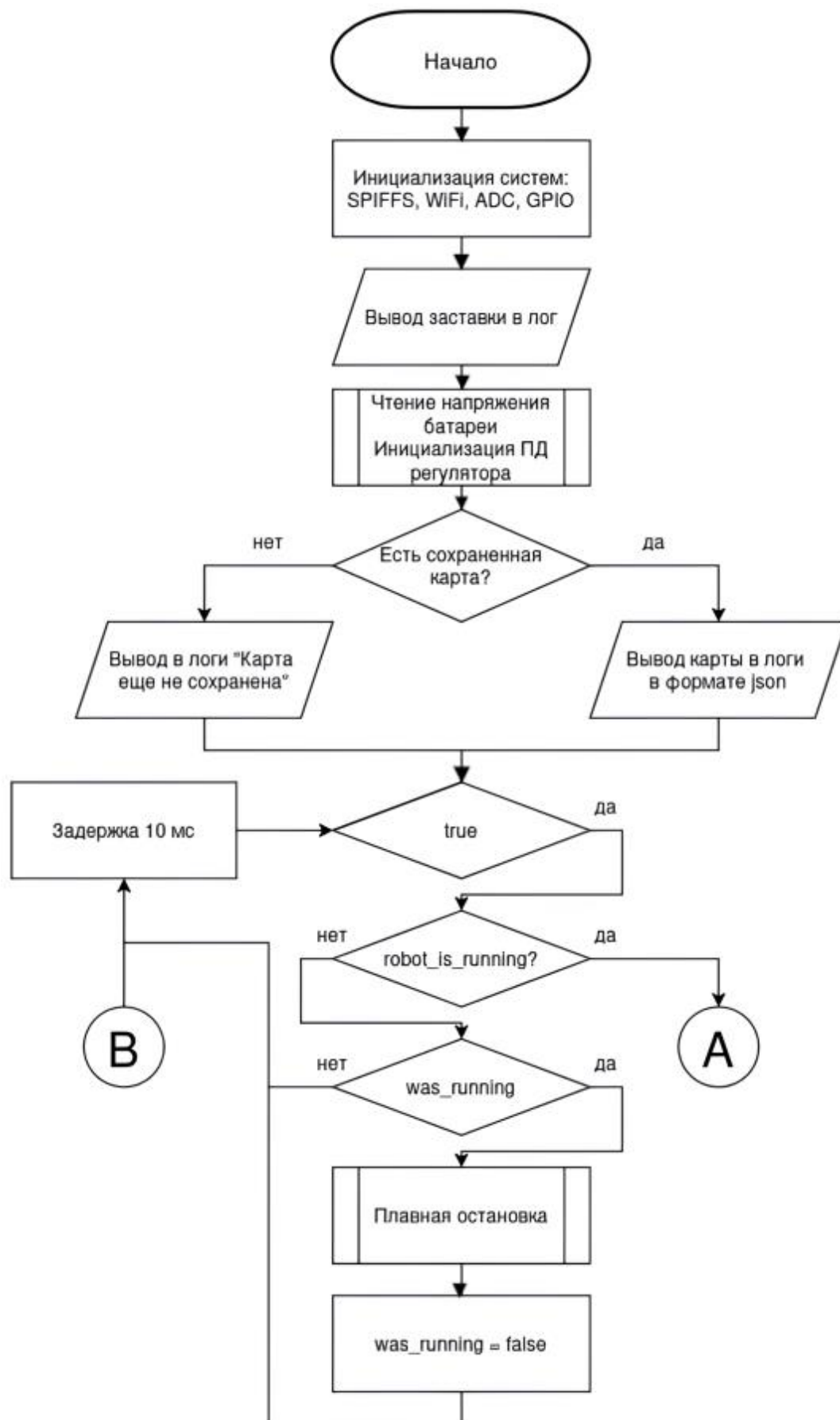


Рисунок 3 – Блок схема main.c

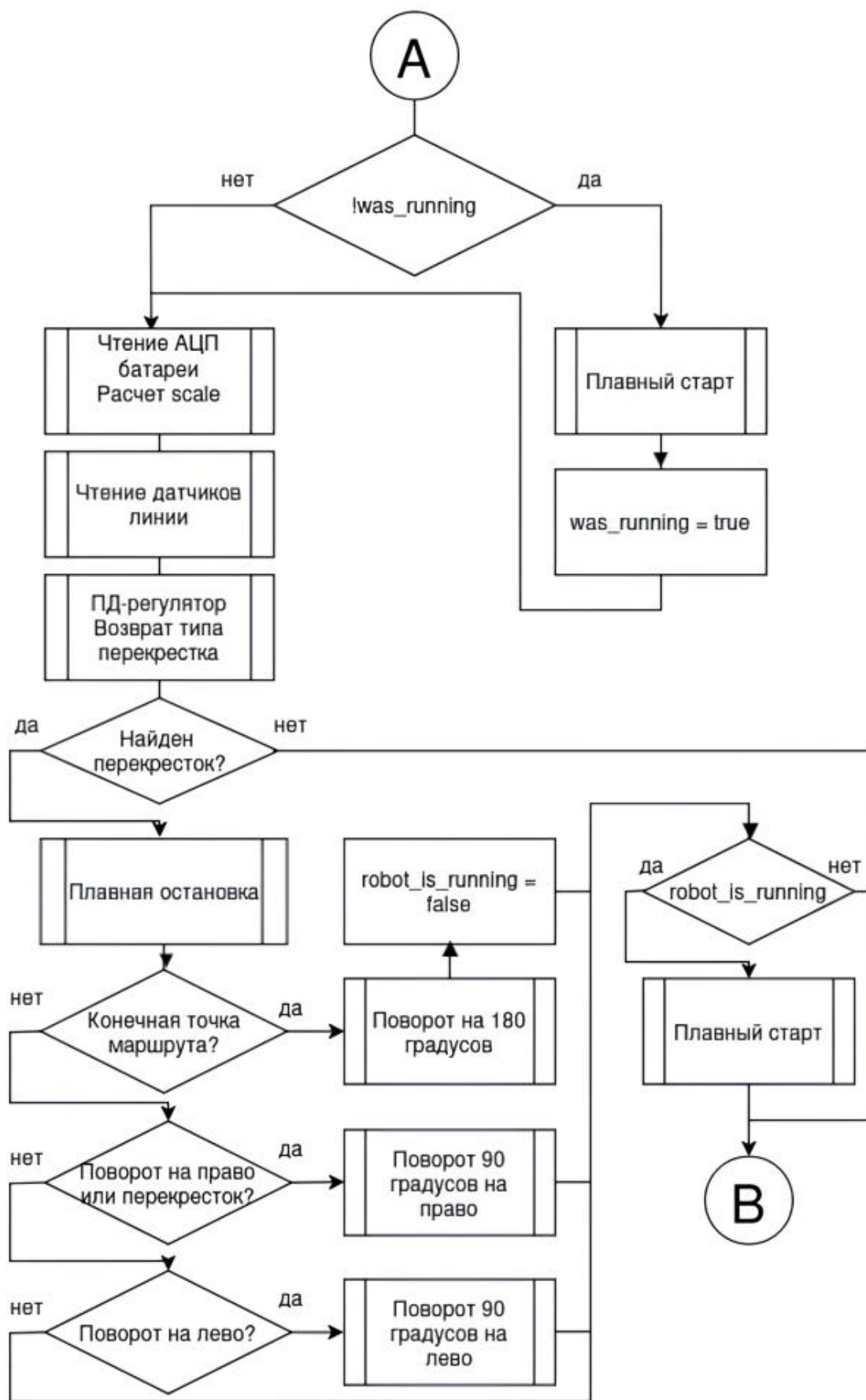


Рисунок 4 – Блок схема `main.c`

На рисунке 5 представлена блок-схема ассемблерской вставки из battery.c.

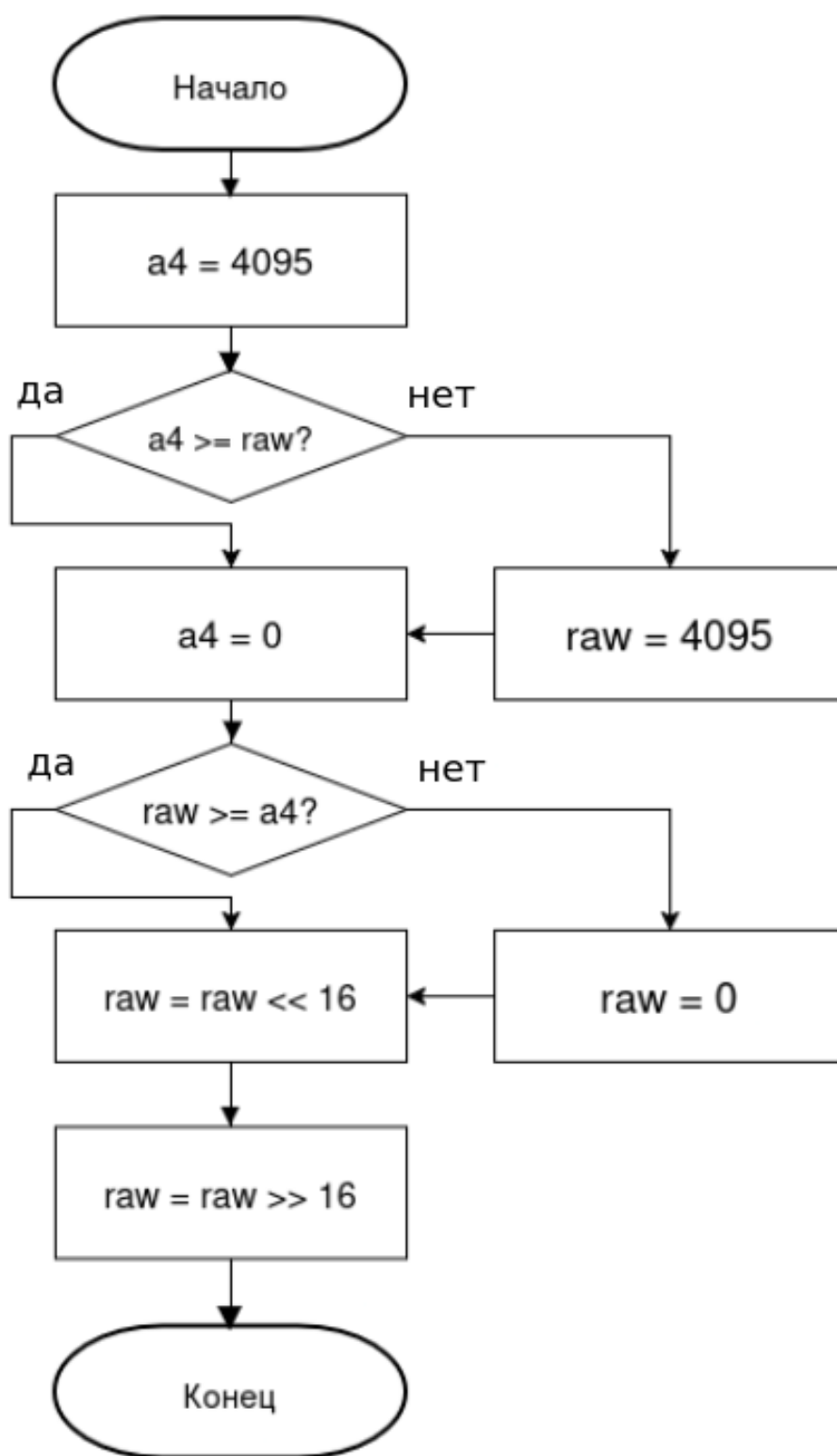


Рисунок 5 – Ассемблерская вставка

Руководство пользователя

Для работы с программным обеспечением робота необходимо подать питание на аппаратную платформу. При наличии подключения по UART и включении логов командой `esp_log` в терминале отобразятся статусы системы (Рисунок 6), текстовая заставка, напряжение батареи и сохраненная карта в json формате (Рисунок 7).

```
[ INIT ] монтирование файловой системы...
I (1319) WEB: SPIFFS mounted successfully
[ OK ] SPIFFS смонтирована успешно.

[ INIT ] запуск беспроводной сети...
I (1339) pp: pp rom version: e7ae62f
I (1339) net80211: net80211 rom version: e7ae62f
I (1349) wifi_init: rx ba win: 6
I (1349) wifi_init: accept mbox: 6
I (1349) wifi_init: tcpip mbox: 32
I (1349) wifi_init: udp mbox: 6
I (1359) wifi_init: tcp mbox: 6
I (1359) wifi_init: tcp tx win: 5760
I (1359) wifi_init: tcp rx win: 5760
I (1369) wifi_init: tcp mss: 1440
I (1369) wifi_init: WiFi IRAM OP enabled
I (1379) wifi_init: WiFi RX IRAM OP enabled
I (1429) esp_netif_lwip: DHCP server started on interface WIFI_AP_DEF with IP: 192.168.4.1
[ OK ] Wi-Fi AP: DeliveryRobot (12345678)
[ OK ] Web-интерфейс доступен по: http://192.168.4.1

[ INIT ] настройка аппаратной части...
I (1439) SENSORS: Sensors initialized successfully.
I (1449) motors: Моторы инициализированы: PWM 25000 Hz, period 400 ticks
I (1459) battery: adc батареи успешно инициализирован
[ OK ] датчики и моторы откалиброваны.
```

Рисунок 6 - Статусы системы

```

  ROBOT
=====
      DELIVERY ROBOT OS v1.0
=====

I (1539) battery: vbat = 4.224 В (0%) -> шим scale = 1000
I (1549) motors: Voltage scale updated to 1000 (макс. 12В на двигателе)

=====
      ЧТЕНИЕ ТОПОЛОГИИ КАРТЫ
=====

Карта найдена! Содержимое графа:
>> {"nodes":[{"id":"A","x":206,"y":199.98863220214844,"type":"CROSS"}, {"id":"B","x":336,"y":185.98863220214844,"type":"CROSS"}, {"id":"C","x":582,"y":207.98863220214844,"type":"CROSS"}, {"id":"D","x":342,"y":59.98863220214844,"type":"CROSS"}, {"id":"E","x":316,"y":335.98863220214844,"type":"CROSS"}], "edges":[{"from":"A", "to":"B"}, {"from":"B", "to":"D"}, {"from":"B", "to":"C"}, {"from":"B", "to":"E"}]}

=====
      СИСТЕМА ГОТОВА К РАБОТЕ. ОЖИДАНИЕ...
=====
```

Рисунок 7 - Текстовая заставка

1. На устройстве пользователя (ПК или смартфон) необходимо выполнить поиск сетей Wi-Fi и подключиться к сети DeliveryRobot (пароль по умолчанию: 12345678).

2. В адресной строке браузера необходимо ввести адрес <http://192.168.4.1>.

3. На экране отобразится графический интерфейс, состоящий из трех логических панелей:

а. **Панель управления** (Рисунок 9): Кнопки старта и остановки робота. Меню инструментов для построения карты.

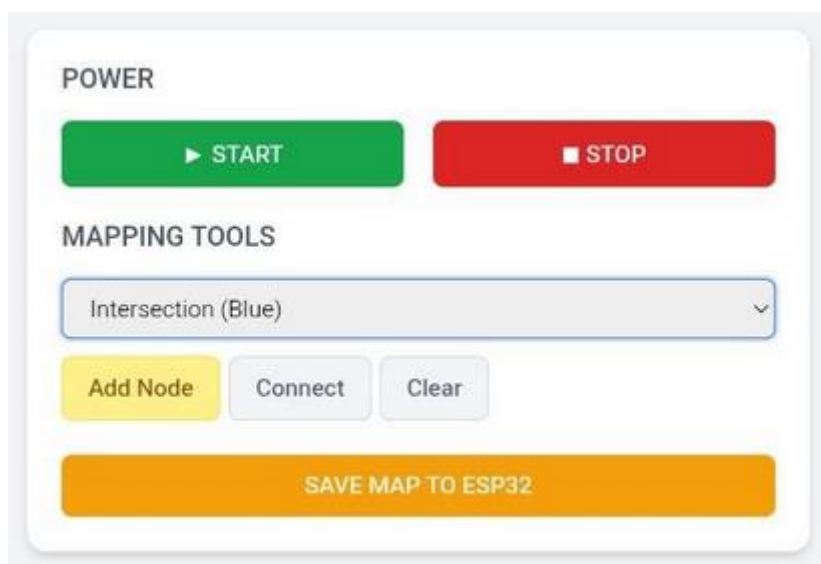
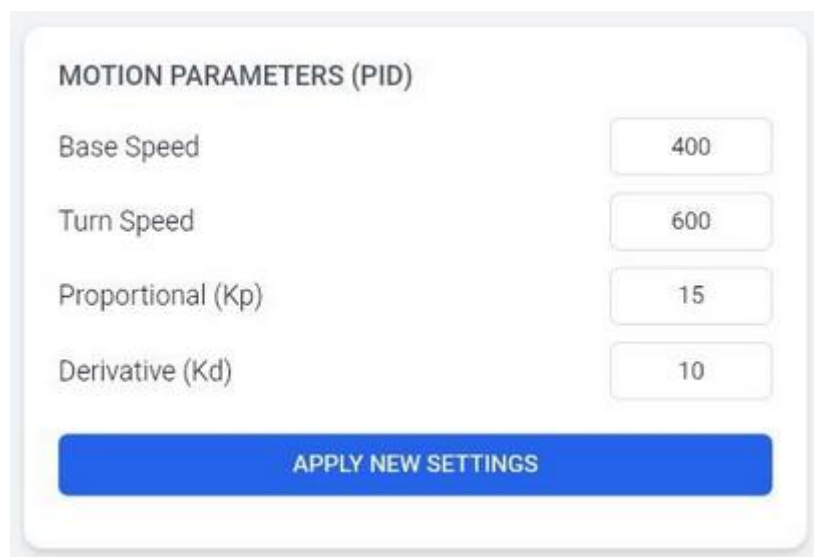


Рисунок 9 - Панель управления

б. **Панель ПИД-регулятора** (Рисунок 10): Текстовые поля для ввода базовой скорости (Base), скорости поворота (Turn) и коэффициентов K_p и K_d .



MOTION PARAMETERS (PID)	
Base Speed	400
Turn Speed	600
Proportional (Kp)	15
Derivative (Kd)	10

APPLY NEW SETTINGS

Рисунок 10 - Панель ПИД-регулятора

с. Графический конструктор карты (Рисунок 11): Холст (Canvas) для создания топологии маршрута.



Рисунок 11 - Холст для создания карты.

4. Для построения карты выберите тип узла из списка - “Start Point (green)”, “Intersection (blue)”, “End Point (red)”, после выберите инструмент “Add Node” и кликните левой кнопкой мыши по пустой области холста, появятся цветной узел с уникальным идентификатором (А, В, С и тд).

5. Для соединения узлов переключитесь на инструмент "Connect" и поочередно кликните на два узла, которые нужно связать дорогой.

6. Для сохранения маршрута в память микроконтроллера нажмите желтую кнопку "SAVE MAP TO ESP32". Система уведомит об успешном сохранении.

7. Для запуска алгоритма удержания линии установите робота на трассу (черная линия) и нажмите "START". При необходимости остановки робота нажмите "STOP".

8. Для изменения параметров ПИД-регулятора введите соответствующие значения в меню, для загрузки настроек в робота нажмите "APPLY NEW SETTINGS". Максимальная скорость ограничена значением в 1000 единиц, минимальная в 0 единиц, но не рекомендуется использовать ниже 300 единиц.

Пример построенной карты на холсте показан на рисунке 12.

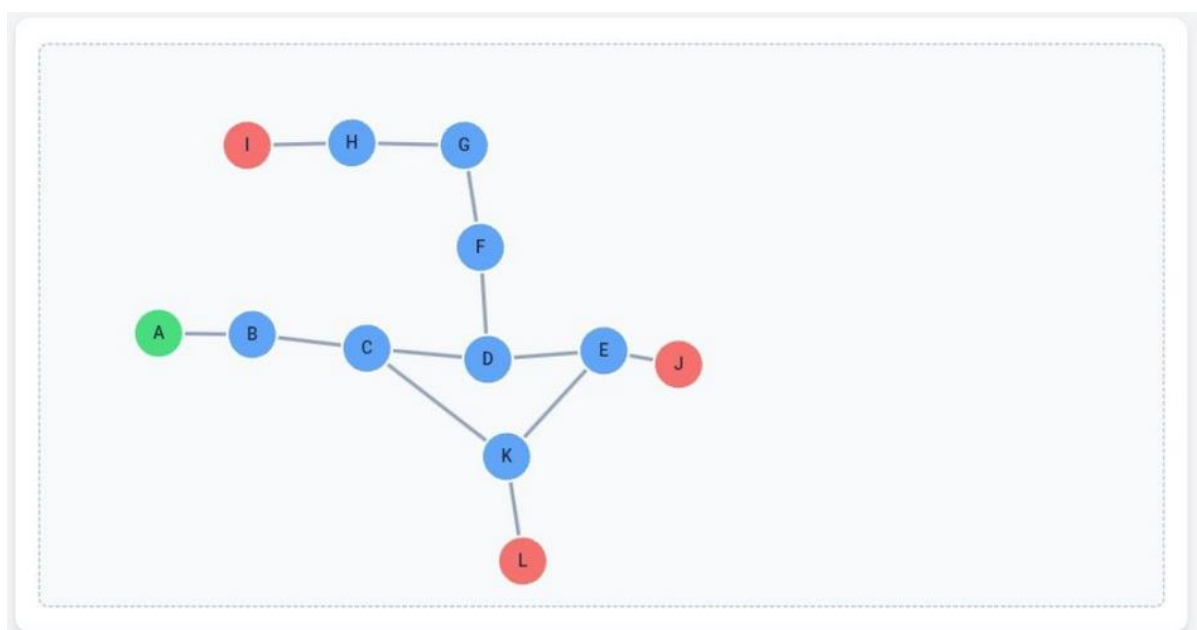


Рисунок 12 - Пример построенной карты

Заключение

В ходе выполнения курсовой работы было разработано программное обеспечение для автономного робота-доставщика на языках C и ASM. Программа реализует сложную многомодульную архитектуру, алгоритм ПИД-регулирования для следования по линии, компенсацию разряда батареи, а также графический веб-интерфейс для рисования и сохранения навигационного графа.

При разработке были закреплены навыки нисходящего проектирования, применения сложных пользовательских структур данных, работы с файловой системой SPIFFS, сетевыми запросами и низкоуровневой аппаратной периферией (АЦП, ШИМ, GPIO). Ассемблерная вставка успешно применена для аппаратного ограничения шумов датчиков на уровне процессора.

Разделение системы на независимые компоненты значительно упростило процессы отладки и интеграции. Графический веб-интерфейс позволил отказаться от жесткого кодирования параметров, предоставив пользователю удобный инструмент для калибровки и маршрутизации.

В дальнейшем программу можно улучшить: внедрить алгоритм поиска кратчайшего пути по графу (алгоритм Дейкстры или A*), добавить поддержку энкодеров для расчета пройденного пути (одометрии) и интегрировать датчики препятствий (LiDAR или датчики расстояния) для объезда динамических преград.

Список используемых источников

1. Керниган Б., Ритчи Д. Язык программирования С. — 2-е изд. — М. : Вильямс, 2021. — 304 с.
2. Таненбаум Э., Остин Т. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 816 с.
3. Yiu, J. The Definitive Guide to ARM Cortex-M0 and Cortex-M0+ Processors (Адаптация общих принципов встраиваемых систем). — Newnes, 2015. — 784 с.
4. Espressif Systems. ESP-IDF Programming Guide [Электронный ресурс]. — URL: <https://docs.espressif.com/projects/esp-idf/> (дата обращения: 20.05.2024).
5. FreeRTOS Real Time Kernel (RTOS) Documentation [Электронный ресурс]. — URL: <https://www.freertos.org/> (дата обращения: 20.05.2024).
6. ГОСТ Р 2.105-2019. Единая система конструкторской документации. Общие требования к текстовым документам. — М.: Стандартинформ, 2019. — 42 с.
7. ГОСТ 19.701-90. Единая система программной документации. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения. — М. : Издательство стандартов, 1990. — 26 с.

Приложение В. Результат работы программы.

Приложение В.1 - скриншот веб-страницы.

POWER

▶ START

■ STOP

MAPPING TOOLS

End Point (Red) ▾

Add Node

Connect

Clear

SAVE MAP TO ESP32

MOTION PARAMETERS (PID)

Base Speed800

Turn Speed700

Proportional (Kp)10

Derivative (Kd)7

APPLY NEW SETTINGS

```
graph LR; A((A)) --- B((B)); B --- C((C)); C --- D((D)); D --- E((E)); D --- G((G)); D --- J((J)); E --- F((F)); F --- K((K)); G --- H((H)); H --- L((L)); I((I)) --- J; I --- M((M)); E --- I
```

Приложение С. Результат работы программы.

/spiffs/map.json

```
{
  "nodes": [
    { "id": "A", "x": 124.81818008422852, "y": 202.4488525390625, "type": "START" },
    { "id": "B", "x": 215.81818008422852, "y": 198.4488525390625, "type": "CROSS" },
    { "id": "C", "x": 309.8181800842285, "y": 209.4488525390625, "type": "CROSS" },
    { "id": "D", "x": 405.8181800842285, "y": 224.4488525390625, "type": "CROSS" },
    { "id": "E", "x": 506.8181800842285, "y": 229.4488525390625, "type": "CROSS" },
    { "id": "F", "x": 600.8181800842285, "y": 232.4488525390625, "type": "CROSS" },
    { "id": "G", "x": 423.8181800842285, "y": 130.4488525390625, "type": "CROSS" },
    { "id": "H", "x": 517.8181800842285, "y": 141.4488525390625, "type": "CROSS" },
    { "id": "I", "x": 410.8181800842285, "y": 307.4488525390625, "type": "CROSS" },
    { "id": "J", "x": 318.8181800842285, "y": 301.4488525390625, "type": "CROSS" },
    { "id": "K", "x": 656.8181800842285, "y": 240.4488525390625, "type": "END" },
    { "id": "L", "x": 607.8181800842285, "y": 148.4488525390625, "type": "END" },
    { "id": "M", "x": 417.8181800842285, "y": 386.4488525390625, "type": "END" }
  ],
  "edges": [
    { "from": "A", "to": "B" },
    { "from": "B", "to": "C" },
    { "from": "C", "to": "D" },
    { "from": "D", "to": "J" },
    { "from": "J", "to": "I" },
    { "from": "I", "to": "M" },
    { "from": "G", "to": "D" },
    { "from": "G", "to": "H" },
    { "from": "H", "to": "L" },
    { "from": "D", "to": "E" },
    { "from": "E", "to": "F" },
    { "from": "F", "to": "K" },
    { "from": "I", "to": "E" }
  ]
}
```

Приложение D. Результат работы программы

Результатом работы программы также является физическое перемещение робота-платформы по черной линии, удержание этой линии с помощью логики ПИД-регулятора. Изменение пропорционального и дифференциального коэффициента заметно отражается на удержании линии, эталонными значениями служат K_p около 5-7 и более высокий K_d в 15-100 единиц, при таких значениях робот стабильно удерживает линию. При увеличении коэффициента K_p , робот начинает вести себе более “резко”, начинает метаться из стороны в сторону в попытке удержания линии. При увеличении коэффициента K_d робот перестает метаться и начинает более плавно удерживать линию, но при слишком высоких значениях K_d робот хуже реагирует на изветвления на трассе. Более низкая скорость, особенно для поворотов (Turn), также улучшает стабильность, но от уменьшения скорости понижается и мощность моторов, значение скорости ниже 300-400 приводит к сильному замедлению или полной остановки робота на сложных участках (повороты, извилки).