TOMSK POLYTECHNIC UNIVERSITY ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский Томский политехнический университет» (ТПУ)

Школа Инженерная школа неразрушающего контроля и безопасности
Направление подготовки 11.03.04 Электроника и наноэлектроника
ООП/ОПОП Прикладная электронная инженерия
Специализация Промышленная электроника
Отделение электронной инженерии

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

Тема работы
Разработка блока навигации для системы автоматизации речного судоходства

УДК 004.415:629.056.8:627.713.06(28)

Обучающийся

Группа	ФИО	Подпись	Дата
1А91	Мальцев Михаил Андреевич		

Руководитель ВКР

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОЭИ	Солдатов А. А.	к.т.н., доцент		

Консультант

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОЭИ	Торгаев С.Н.	к.ф.-м.н., доцент		

КОНСУЛЬТАНТЫ ПО РАЗДЕЛАМ:

По разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»

Должность	ФИО	Ученая степень, звание	Подпись	Дата
доцент ОСГН ШБИП ТПУ	Маланина В.А.	к.э.н., доцент		

По разделу «Социальная ответственность»

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Старший преподаватель ООД	Мезенцева И.Л.	_		

Нормоконтроль

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОЭИ	Дикман Е.Ю.	к.т.н.		

ДОПУСТИТЬ К ЗАЩИТЕ:

Руководитель ООП, должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОЭИ	Иванова В.С.	к.т.н.		

Томск – 2023 г.

ПЛАНИРУЕМЫЕ РЕЗУЛЬТАТЫ ОСВОЕНИЯ ООП/ОПОП

Код компетенции	Наименование компетенции
Универсальные компетенции	
УК(У)-1	Способен осуществлять поиск, критический анализ и синтез информации, применять системный подход для решения поставленных задач
УК(У)-2	Способен определять круг задач в рамках поставленной цели и выбирать оптимальные способы их решения, исходя из действующих правовых норм, имеющихся ресурсов и ограничений
УК(У)-3	Способен осуществлять социальное взаимодействие и реализовывать свою роль в команде
УК(У)-4	Способен осуществлять деловую коммуникацию в устной и письменной формах на государственном языке Российской Федерации и иностранном(-ых) языке(-ах)
УК(У)-5	Способен воспринимать межкультурное разнообразие общества в социально-историческом, этическом и философском контекстах
УК(У)-6	Способен управлять своим временем, выстраивать и реализовывать траекторию саморазвития на основе принципов образования в течение всей жизни
УК(У)-7	Способен поддерживать должный уровень физической подготовленности для обеспечения полноценной социальной и профессиональной деятельности
УК(У)-8	Способен создавать и поддерживать безопасные условия жизнедеятельности, в том числе при возникновении чрезвычайных ситуаций
УК(У)-9	Способен проявлять предприимчивость в практической деятельности, в т.ч. в рамках разработки коммерчески перспективного продукта на основе научно-технической идеи
Общепрофессиональные компетенции	
ОПК(У)-1	Способен использовать положения, законы и методы естественных наук и математики для решения задач инженерной деятельности.
ОПК(У)-2	Способен самостоятельно проводить экспериментальные исследования и использовать основные приёмы обработки и представления полученных данных
ОПК(У)-3	Способен применять методы поиска, хранения, обработки, анализа и представления в требуемом формате информации из различных источников и баз данных, соблюдая при этом основные требования информационной безопасности.
ОПК(У)-4	Способен применять современные компьютерные технологии для подготовки текстовой и конструкторско-технологической документации с учетом требований нормативной документации.
Профессиональные компетенции	
ПК(У)-1	Способен строить простейшие физические и математические модели приборов, схем, устройств и установок электроники и наноэлектроники различного функционального назначения, а также использовать стандартные программные средства их компьютерного моделирования
ПК(У)-2	Способен аргументировано выбирать и реализовывать на практике эффективную методику экспериментального исследования параметров и

	характеристик приборов, схем, устройств и установок электроники и наноэлектроники различного функционального назначения
ПК(У)-3	Способен выполнять расчет и проектирование электронных приборов, схем и устройств различного функционального назначения в соответствии с техническим заданием с использованием средств автоматизации проектирования
ПК(У)-4	Способность разрабатывать проектную и техническую документацию, оформлять законченные проектно-конструкторские работы

TOMSK POLYTECHNIC UNIVERSITY ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский Томский политехнический университет» (ТПУ)

Школа	Инженерная школа неразрушающего контроля и безопасности
Направление подготовки	11.03.04 Электроника и наноэлектроника
ООП/ОПОП	Прикладная электронная инженерия
Специализация	Промышленная электроника
Отделение	электронной инженерии

УТВЕРЖДАЮ:
Руководитель ООП
_____ _____ В.С. Иванова
(Подпись) (Дата) (ФИО)

ЗАДАНИЕ
на выполнение выпускной квалификационной работы

Обучающийся:

Группа	ФИО
1А91	Мальцев Михаил Андреевич

Тема работы:

Разработка блока навигации для системы автоматизации речного судоходства	
Утверждена приказом директора (дата, номер)	39-34/с от 08.02.2023

Срок сдачи обучающимся выполненной работы:	

ТЕХНИЧЕСКОЕ ЗАДАНИЕ:

Исходные данные к работе *(наименование объекта исследования или проектирования; производительность или нагрузка; режим работы (непрерывный, периодический, циклический и т. д.); вид сырья или материал изделия; требования к продукту, изделию или процессу; особые требования к функционированию (эксплуатации) объекта или изделия в плане безопасности эксплуатации, влияния на окружающую среду, энергозатратам; экономический анализ и т. д.)*	Функциональные возможности: • возможность передачи пакета данных через GSM-модуль; • определение угла наклона относительно гравитационного поля; • определение азимута; • определение GPS-координат; • определение текущего времени; • возможность принимать команды через GSM-модуль; • реализация возможности режимов световой индикации согласно ГОСТ 26600-98 «Знаки навигационные внутренних судоходных путей»; • реализация на программы на базе FreeRTOS.

TOMSK POLYTECHNIC UNIVERSITY ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский Томский политехнический университет» (ТПУ)

Школа	Инженерная школа неразрушающего контроля и безопасности
Направление подготовки	11.03.04 Электроника и наноэлектроника
ООП/ОПОП	Прикладная электронная инженерия
Специализация	Промышленная электроника
Отделение	электронной инженерии

КАЛЕНДАРНЫЙ РЕЙТИНГ-ПЛАН
выполнения выпускной квалификационной работы

Обучающийся:

Группа	ФИО
1А91	Мальцев Михаил Андреевич

Тема работы:

Разработка блока навигации для системы автоматизации речного судоходства

Срок сдачи обучающимся выполненной работы:	

Дата контроля	Название раздела (модуля) / вид работы (исследования)	Максимальный балл раздела (модуля)
20.02.2023	Обзор литературы	10
10.02.2023	Разработка структурной схемы устройства	10
20.04.2023	Разработка программы для микроконтроллера	50
05.05.2023	Разработка принципиальной схемы	10
10.05.2023	Экспериментальные результаты	10
15.05.2023	Финансовый менеджмент	5
20.05.2023	Социальная ответственность	5

СОСТАВИЛ:
Руководитель ВКР

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОЭИ	Солдатов А. А.	к.т.н		

Консультант (при наличии)

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОЭИ	Торгаев С.Н.	к.ф.-м.н		

СОГЛАСОВАНО:
Руководитель ООП

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОЭИ	Солдатов А. А.	к.т.н		

Обучающийся

Группа	ФИО	Подпись	Дата
1А91	Мальцев Михаил Андреевич		

Перечень разделов пояснительной записки подлежащих исследованию, проектированию и разработке *(аналитический обзор литературных источников с целью выяснения достижений мировой науки техники в рассматриваемой области; постановка задачи исследования, проектирования, конструирования; содержание процедуры исследования, проектирования, конструирования; обсуждение результатов выполненной работы; наименование дополнительных разделов, подлежащих разработке; заключение по работе)*	1. Обзор литературы 2. Разработка структурной схемы устройства 3. Разработка программы для микроконтроллера 4. Разработка принципиальной схемы 5. Экспериментальные результаты
Перечень графического материала *(с точным указанием обязательных чертежей)*	1. Схема принципиальная электрическая 2. Перечень элементов

Консультанты по разделам выпускной квалификационной работы
(с указанием разделов)

Раздел	Консультант
Разработка программы микроконтроллера	доцент ОЭИ ИШНКБ Торгаев С.Н.

Названия разделов, которые должны быть написаны на иностранном языке:

Дата выдачи задания на выполнение выпускной квалификационной работы по линейному графику	

Задание выдал руководитель / консультант (при наличии):

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОЭИ	Солдатов А. А.	к.т.н., доцент		
Доцент ОЭИ	Торгаев С.Н.	к.ф.-м.н., доцент		

Задание принял к исполнению обучающийся:

Группа	ФИО	Подпись	Дата
1А91	Мальцев Михаил Андреевич		

РЕФЕРАТ

Выпускная квалификационная работа 146 с., 40 рис., 22 табл., 29 источников, 3 прил.

Ключевые слова: GSM, навигация, GPS, азимут, программа, микроконтроллер, freeRTOS.

Объектом разработки является блок навигации.

Цель работы – разработка блока навигации для определения координат устройства, азимут и угла наклона относительно гравитационного поля.

В ходе работы проводились: калибровка магнетометра, разработка программы, разработка схем подключения, экспериментальная проверка работоспособности.

В результате разработки: была написана и протестирована программа микроконтроллера.

Основные конструктивные, технологические и технико-эксплуатационные характеристики: использование freeRTOS.

Степень внедрения: проект предлагается к внедрению с целью получения экономического и социального эффекта.

Область применения: навигация внутреннего судоходства, мониторинг водных объектов.

Экономическая эффективность/значимость работы: при оценки экономической эффективности данный проект показал свою финансовую состоятельность.

В будущем планируется оптимизация энергопотребления.

Содержание

ВВЕДЕНИЕ

Для обеспечения сохранности таких объектов, как искусственных сооружений на внутренних водных путях, и безопасного плавания используют навигационное оборудование, которое представляет собой систему береговых и плавучих знаков и огней. Очень важно для обеспечения на водных путях с интенсивным движением круглосуточного судоходства использовать навигационные знаки со светосигнальным оборудованием, создающие огонь определенного цвета и характера горения.

Так же очень важным становится не только навигация, но и отслеживание параметров воды и окружающей среды. Это становится полезным для отслеживания водной среды лабораториями, которые могут проводить исследования о состоянии воды, перемещениях живых организмов.

Разрабатываемое устройство – это инновационная система для навигации, сбора, передачи данных об окружающей и водной среде в режиме реального времени, как для лабораторных исследований, так и для навигации внутреннего судоходства.

Система по определению наклона буя относительно гравитационного поля Земли позволит предотвращать экстренные ситуации (переворот устройства). Система по отслеживанию угла в азимутальной плоскости светосигнальных огней позволит производить отслеживание правильность вектора индикации.

Система передачи данных геолокации, информации об углах даёт возможность перманентно отслеживать состояние устройства. Эти данные будет учитываться при навигации на водных путях, а мониторинговая служба сможет своевременно принимать решение о необходимости ремонтных работ.

Цель работы: разработка блока навигации для определения координат устройства, азимут и угла наклона относительно гравитационного поля.

Задачи:

1. обзор существующих аналогов;
2. разработка структурной схемы устройства;
3. разработка схем подключения модулей;
4. разработка программы для микроконтроллера с использованием freeRTOS;
5. калибровка датчика LSM303DHLC;
6. проверка работоспособности программы.

Объект исследования: блок навигации.

Методы исследования: методы исследования включали в себя вопросы программирования микроконтроллеров, а также экспериментальные методы.

Научная значимость результатов ВКР: результаты, полученные в ходе выполнения ВКР, могут быть использованы при проведении дальнейших исследований в области речной навигации.

Практическая значимость результатов ВКР: данный проект может быть применен в сферах связанных с мониторингом водных объектов и в навигации речного судоходства.

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ, СОКРАЩЕНИЯ, НОРМАТИВНЫЕ ССЫЛКИ

Сокращения:

- АИС – Автоматическая идентификационная система
- GSM – Global System for Mobile Communications
- NMEA – National Marine Electronics Association
- MEMs – Micro Electromechanical system
- EMA – Exponential Moving Average
- UART – Universal Asynchronous Receiver/Transmitter

1 ОБЗОР ЛИТЕРАТУРЫ

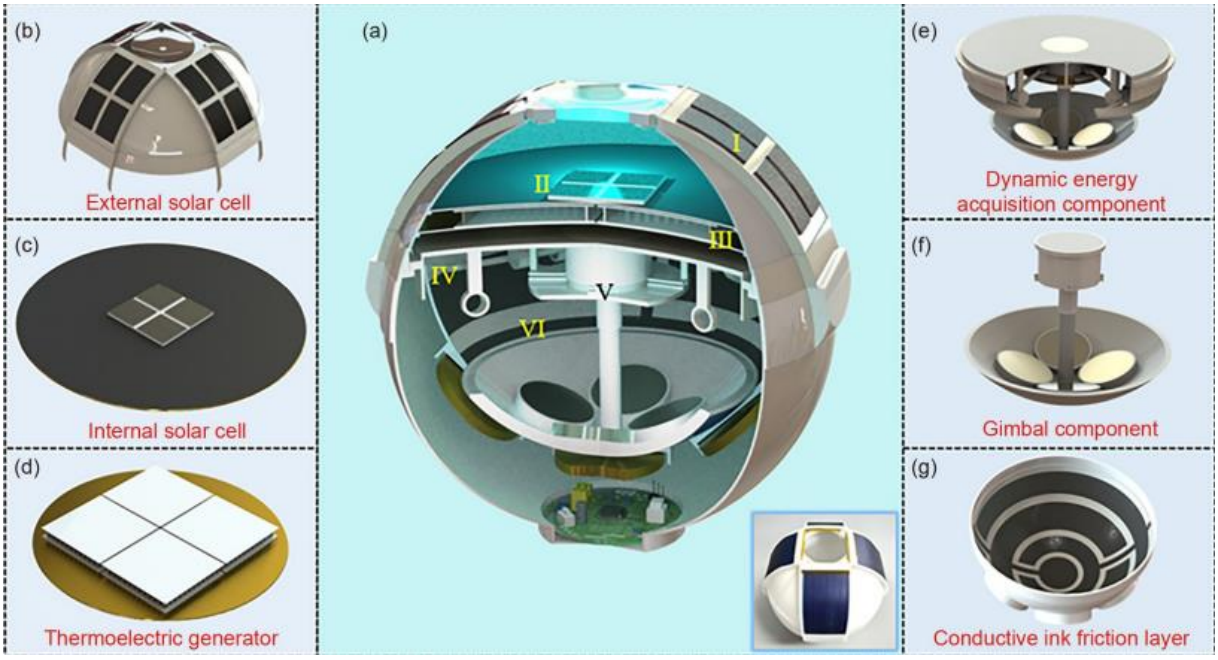
1.1 Обзор аналогов устройства

1.1.1 Статико-динамический сборщик энергии для автономного мониторинга окружающей среды океана.

Данное устройство позволит использовать солнечную энергию для заряда батарей, которые будут применяться для питания буя и его основных блоков. В настоящее время использование химических батарей для океанской беспроводной связи и мониторинга может привести к ряду проблем, таких как ограниченное накопление энергии и загрязнение окружающей среды после утилизации.

Чтобы преодолеть эти недостатки необходимо производить поиск новых экологических и эффективных источников энергии. Океан богат не только ресурсами, но и огромным источником возобновляемой энергии. В океане существует множество форм энергии, таких как солнечная энергия, тепловая энергия океана и кинетическая энергия.

В данном аналоге идет упор на замену обычных литии-ионных батарей на солнечную энергию или же кинетическую энергию волн. Структурная схема представлена на рисунке 1 [1].



a – сфероидальный буй в сборе; b – внешний солнечный элемент; c – внутренний солнечный элемент; d – термоэлектрический генератор; e – компонент получения динамической энергии; f – карданный компонент; g - фрикционный слой проводящих чернил.

Рисунок 1 – Структурная схема сфероидального буя [1]

Четыре гибких солнечных элемента (длина: 70 мм, ширина: 40 мм) соединены последовательно, они располагаются на внешней поверхности сферического устройства.

Выпуклая линза расположена на верхней части сферического устройства для освещения внутренних солнечных элементов (см. рисунок 1 b). Во избежание попадания солнечного света в корпус слой отражающей пленки крепится к внутренней части оболочки.

Данная технология может быть актуальна для оптимизации питания в современных системах навигационного оборудования, в частности в буях.

1.1.2 Плавучие знаки с автоматической идентификационной системой (АИС) от Гидрографического предприятия «Росатом»

Данного типа буи установлены на Морском канале – подходном пути в Обской губе Карского моря. Из 22 буев 6 оснащены станциями АИС. АИС работает в УКВ-диапазоне и предназначена для идентификации судов, определения габаритов, курса и других данных. Оборудование АИС устанавливается на судах, совершающих международные рейсы, пассажирских судах, а также судах с большой валовой вместимостью. Кроме того, оборудование АИС может устанавливаться на стационарных объектах береговых служб. Станции на плавучих предостерегательных знаках передают в эфир сигнал, позволяющий идентифицировать тип буя и его координаты при любых погодных условиях. Это обеспечивает безопасность судоходства. Однако данные буи не оснащены дополнительными датчиками и прибора по отслеживанию состояния окружающей среды и воды [2]. Внешний вид буя представлен на рисунке 2.

Рисунок 2 – Внешний вид буя с автоматической идентификационной системы [2]

1.2 GSM модуль

GSM модуль является одним из наиболее распространенных способов передачи информации в современном мире. Этот модуль использует стандарт связи GSM (Global System for Mobile Communications), который был разработан для передачи голосовой и цифровой информации между мобильными телефонами.

Достоинства GSM модуля заключаются в его широкой распространенности и надежности. Благодаря этому модуль может использоваться в различных приложениях, таких как системы мониторинга и управления, системы безопасности и другие. Кроме того, GSM модуль позволяет передавать информацию на большие расстояния, что делает его идеальным для использования в удаленных местах.

Однако у GSM модуля есть и недостатки. Во-первых, он может быть подвержен помехам, что может привести к потере данных или неправильной интерпретации информации. Во-вторых, GSM модуль требует наличия сигнала мобильной связи, что может быть проблематично в удаленных местах или в зонах с плохим покрытием.

Кроме того, GSM модуль может быть подвержен уязвимостям в отношении безопасности. Некоторые модели GSM модулей могут быть скомпрометированы злоумышленниками, что может привести к утечке конфиденциальной информации или к нарушению работы системы [3].

В заключение GSM модуль является широко используемым и надежным способом передачи информации. Он имеет свои достоинства и недостатки, которые необходимо учитывать при выборе способа передачи информации в конкретном приложении. Структурная схема GSM-модуля представлена на рисунке 3.

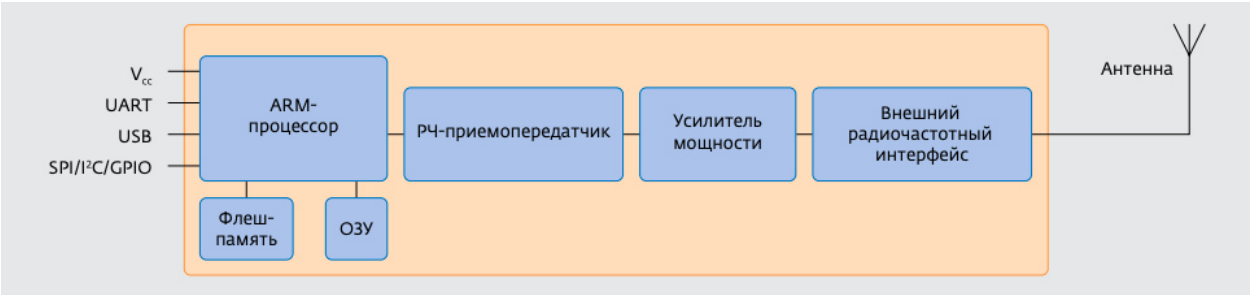


Рисунок 3 – Структура GSM-модуля

1.3 GPS модуль

GPS (Global Positioning System) представляет собой систему спутниковой навигации. Она позволяет определять местонахождение объектов на поверхности Земли, используя сигналы, излучаемые спутниками в низкой земной орбите. GPS-приемник

представляет собой специализированное устройство, которое принимает сигналы, передаваемые спутниками, и вычисляет координаты местоположения.

GPS-приемник работает на основе триангуляционного метода измерения расстояния. Это означает, что приемник принимает сигналы от трех или более спутников, и использует время полета сигналов, чтобы определить расстояние до каждого спутника. После этого, используя техники геометрической триангуляции, приемник вычисляет свое местоположение на поверхности Земли.

Сигналы, излучаемые спутниками GPS, являются сверхширокополосными радиоволнами, которые передают информацию о времени и местоположении спутника на момент излучения сигнала. Эта информация передается на частоте 1575,42 МГц (L1-частота) и 1227,6 МГц (L2-частота).

Чтобы определить свое местоположение, приемник должен знать точное время в момент приема сигналов от спутников. Для этого все спутники GPS синхронизируют свои внутренние часы с базовой системой времени, называемой GPS-временем. GPS-приемник берет сигналы времени от спутников и использует их для определения своего собственного времени, с точностью до нескольких наносекунд.

Как уже упоминалось, для определения координат местоположения приемник использует метод геометрической триангуляции. Для этого он измеряет время полета сигналов от каждого спутника и, зная скорость распространения сигналов (скорость света), вычисляет расстояния до каждого спутника. После этого приемник использует триангуляционный алгоритм, чтобы определить свое местоположение на поверхности Земли [4].

Для вычисления расстояния до спутника и определения местоположения то устройство использует следующую формулу (1.1) [4]:

$$R = c * t, \tag{1.1}$$

где R - расстояние до спутника;

t - время полета сигнала;

c - скорость света.

GPS-приемник — это критически важный инструмент для навигации и ориентирования на поверхности Земли. Он использует спутники GPS для определения координат местоположения с высокой степенью точности и надежности.

1.4 Протокол NMEA 0183

NMEA 0183 (от «National Marine Electronics Association») – стандарт определяющий текстовый протокол связи навигационного оборудования. Стал особенно популярен в связи с распространением GPS-приёмников, использующих этот стандарт для передачи данных по шине UART. На рисунке 4 представлен пример NMEA строки.

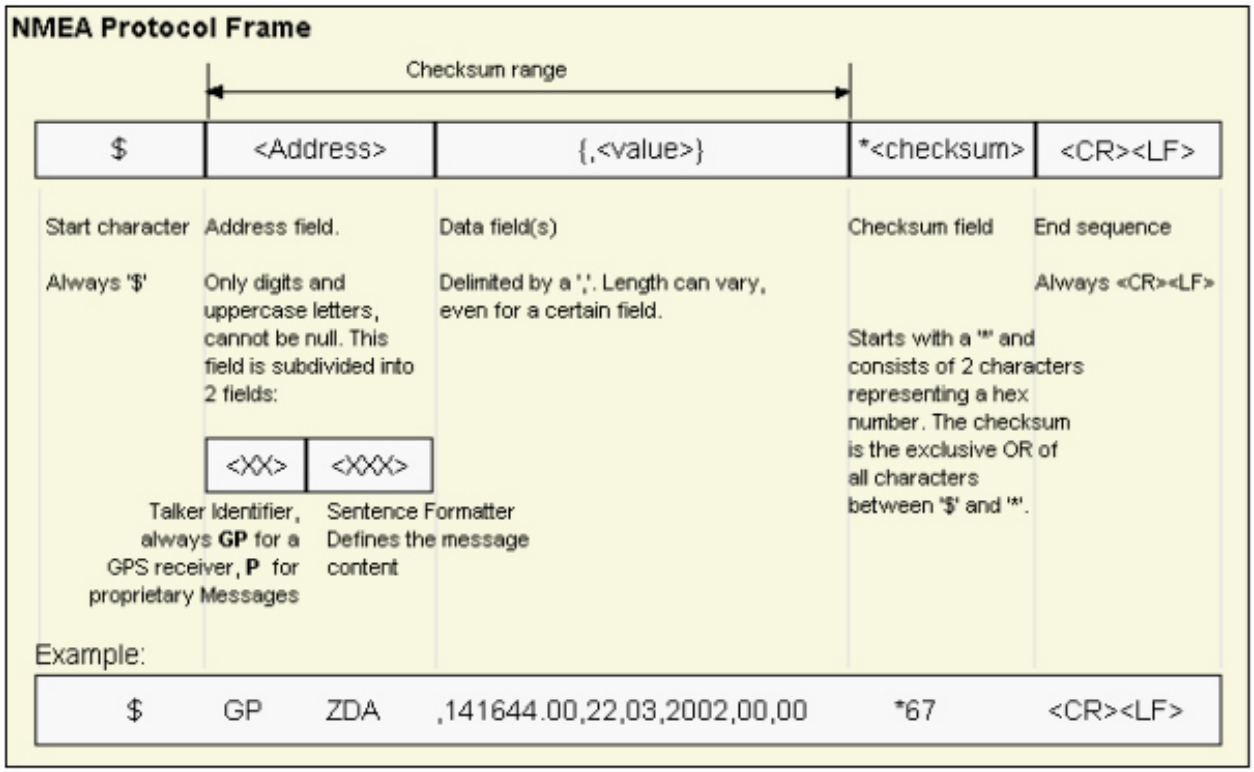


Рисунок 4 – Протокол NMEA [5]

Состав строк:

– каждая строка начинается символом '$';

– далее следуют два символа источника данных и три символа идентификатора строки;

– после идентификатора следуют данные, состоящие из параметров, разделённые запятыми;

– завершают строку символ '*', два символа контрольной суммы и символы "\r\n".

Источник данных представлен двумя символами, которые следуют за символом '$'.

Источники данных:

– GP – информация получена от спутников GPS (США);

– GL – информация получена от спутников Глонасс (Россия);

– GA – информация получена от спутников Galileo (Европа);

– BD – информация получена от спутников Beidou (Китай);

– GQ – информация получена от спутников QZSS (Япония);

– GN – информация получена от спутников разных навигационных систем.

Идентификатор строки представлен тремя символами, следующими за источником данных.

Идентификатор строки:

– GGA – данные о последнем зафиксированном местоположении;

– GLL – географические координаты;

– GSA – информация об активных спутниках (участвующих в позиционировании).

– GSV – информация о всех наблюдаемых спутниках;

– RMC – рекомендуемый минимум навигационных данных;

– VTG – скорость и курс относительно земли;

– ZDA – дата и время;

– DHV – информация о скорости движения GNSS приемника;

– GST – статистика ошибок позиционирования;

– TXT – текстовое сообщение.

Данные состоят из параметров, разделённых запятыми. Тип и состав параметров зависит от идентификатора строки.

Контрольная сумма представлена двумя символами, следующими за символом '*'.

Контрольная сумма представляет из себя шестнадцатеричное представление результата операции XOR с байтами всех символов строки расположенных между '$' и '*', не включительно [5].

Строка с идентификатором RMC содержит рекомендуемый минимум навигационных данных.

Пример строки RMC:

```
$GNRMC,102030.000,A,5546.95900,N,03740.69200,E,0.12,49.75,200220,,,A*FF/r/n
       ^          ^ ^          ^ ^          ^ ^    ^     ^     ^^ ^ ^

       1          2 3          4 5          6 7    8     9     01 2 3
```

Назначение параметров строки RMC:

1. время UTC в формате "ЧЧММСС.СССС";
2. достоверность полученных координат:
   – 'A' - данные достоверны;
   – 'V' - ошибочные данные.
3. широта в формате "ГГММ.ММММММ";
4. направление широты: 'N'-север / 'S'-юг;

5. долгота в формате "ГГГММ.ММММММ";
6. направление долготы: 'E'-восток / 'W'-запад;
7. скорость в узлах;
8. курс на истинный полюс в градусах;
9. дата в формате "ДДММГГ";
10. магнитное склонение в градусах;
11. направление магнитного склонения: 'E'-вычесть / 'W'-прибавить;
12. способ вычисления координат:
    - 'A' – автономный;
    - 'D' – дифференциальный;
    - 'E' – аппроксимация;
    - 'M' – фиксированные данные;
    - 'N' – недостоверные данные.
13. статус навигации.

1.5 MEMS магнитометры

MEMS (Micro-Electro-Mechanical Systems) магнитометр — это небольшой и высокочувствительный инструмент, который способен измерять магнитные поля. MEMS магнитометры получают все большее распространение в различных сферах научных и технических исследований, начиная от промышленности и заканчивая космическими экспедициями.

MEMS магнитометры работают на основе принципа эффекта Холла. Этот эффект возникает при прохождении электрического тока через проводник, располагающийся в магнитном поле. В результате этого образуются два электрических потенциала, которые проявляются вдоль противоположных сторон проводника, перпендикулярных направлению тока и магнитному полю. Если взять проводник и сформировать его в виде плоского элемента, он станет основой MEMS магнитометра.

Когда плоский элемент внутри MEMS магнитометра находится в магнитном поле, возникает разность потенциалов. Этот эффект можно измерить, используя электроды, которые находятся на обоих концах плоского элемента. В результате создания этого эффекта MEMS магнитометры могут измерять магнитные поля на уровне нескольких нанотесл, что позволяет регистрировать изменения магнитного поля при перемещении даже самых маленьких магнитных объектов.

MEMS магнитометры имеют несколько преимуществ перед другими типами магнитометров. Они являются крайне чувствительными и могут работать настолько быстро, что могут измерять даже самые мелкие изменения магнитного поля в области до 10 мкс. Это делает их идеальными для использования в быстро движущихся объектах, где необходимо получать точную информацию о магнитном поле [6].

Одним из основных применений MEMS магнитометров является измерение геомагнитных полей на земле. Они широко используются в аэронавигации, магнитную векторную гравиметрию, геопоисках и других областях.

1.6 MEMS акселерометры

MEMS акселерометры представляют собой микроэлектромеханические системы, способные измерять ускорения и перемещения в трехмерном пространстве. Они широко используются в различных устройствах, включая мобильные телефоны, навигационные системы, спортивные трекеры, медицинское оборудование и многое другое.

MEMS акселерометры работают на основе датчика ускорения, который замеряет изменение емкости конденсатора. Как только акселерометр начинает перемещаться, изменяется гравитационное поле, в результате чего изменяется и расстояние между двумя электродами конденсатора. В результате изменения ёмкости конденсатора возникает электрический сигнал, который обрабатывается внутри акселерометра и выводится на внешние устройства. Одним из главных преимуществ MEMS акселерометров является их маленький размер. Благодаря этому они могут быть легко интегрированы в другие микроэлектромеханические системы, что позволяет создавать компактные устройства, не ограниченные габаритами и весом. Кроме того, MEMS акселерометры обладают высокой точностью измерения и могут работать на очень малых уровнях ускорения. Это делает их идеальными для применения в системах навигации и автоматической стабилизации, где необходимо точно измерять силу и направление ускорения.

Одним из важных применений MEMS акселерометров является контроль движения роботов и автономных транспортных средств. С помощью акселерометров такие устройства могут точно определять свое положение в пространстве, а также корректировать свое движение в реальном времени. В медицинских приложениях MEMs акселерометры используются для мониторинга активности и физического состояния пациента. Они могут измерять частоту сердечных сокращений, частоту дыхания, уровень физической активности и многое другое [7].

Таким образом, MEMS акселерометры являются важной частью современных электронных устройств, обеспечивая высокую точность и надежность измерения

ускорения и перемещения в трехмерном пространстве. Их широкое применение в различных отраслях науки и техники делает их неотъемлемой частью современного технологического прогресса.

1.7 Искажения магнетометра

Калибровка магнетометра – это процесс, позволяющий уменьшить ошибки измерений магнитного поля, вызванные внутренними и внешними факторами.

Основными видами искажения являются Hard Iron и Soft Iron. Hard Iron – аддитивная ошибка, которая суммируется с магнитным полем Земли. Причиной данного искажения является наличие постоянного магнита. Soft Iron – ошибка, которая искажает магнитное поле Земли вокруг датчика. Ошибка Soft Iron вносится предметами, которые искажают окружающее магнитное поле, но не обязательно генерируют свое. Например, данный вид искажений характерен для никеля и железа. Примеры воздействия данных искажений представлены на рисунке 5 [8].

Рисунок 5 – Пример воздействия искажений [8]

На рисунке 6 показаны как искажаются данные при некалиброванном магнитометре.

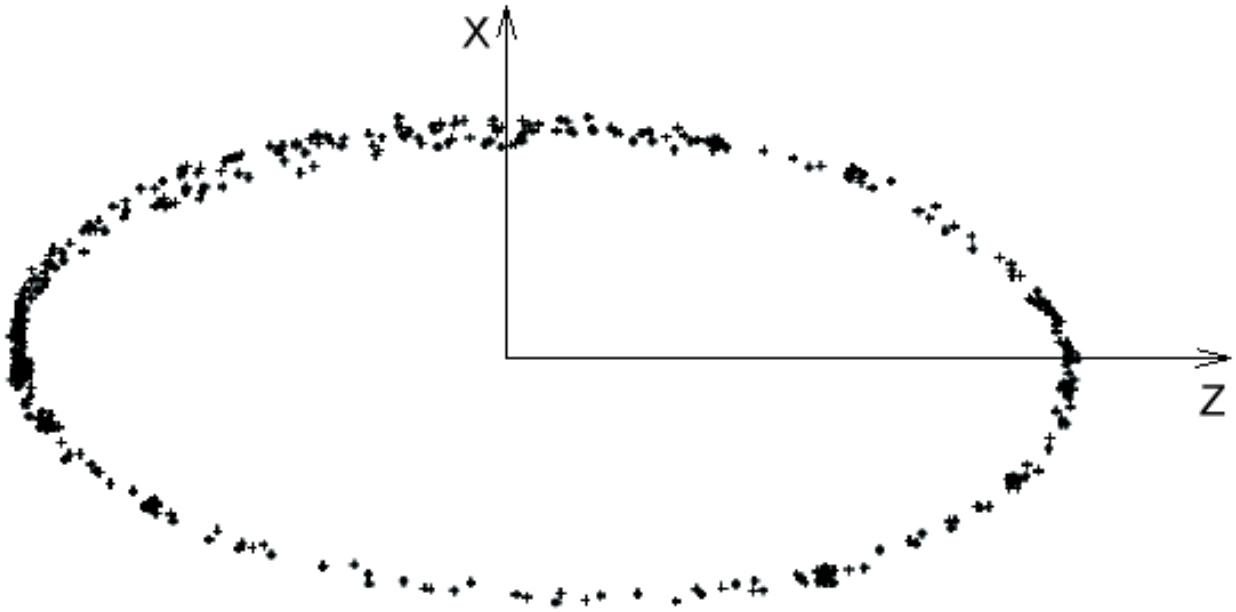


Рисунок 6 – Пример искаженных данных

1.7.1 Калибровка HardIron

Для нивелирования HardIron искажения достаточно увеличить или уменьшить получаемые от прибора значения на величину смещения.

Формула (1.2) для расчёта калибровочного значения представлена ниже [8]:

$$Y_{cal_{hard}} = Y - Ybias, \tag{1.2}$$

где *Ycal_hard* – калиброванное значение;

Y – исходное значение;

Ybias – величина смещения.

Чтобы вычислить Ybias потребуется зафиксировать максимальное и минимальное значение Y, а затем воспользоваться выражением (1.3) [8]:

$$Y_{bias} = \frac{Y_{max} - Y_{min}}{2} - Y_{min}, \tag{1.3}$$

где *Ybias* – искомая величина смещения;

Ymin – минимальное значение оси Y;

Ymax – максимальное значение оси Y.

1.7.2 Калибровка SoftIron

На рисунке 6 видно, что фигура, которую образуют данные, является эллипсом, чтобы нивелировать данные искажения потребуется найти небольшое значения среди всех осей и воспользоваться следящей формулой (1.4) [8]:

$$Y_{cal_{soft}} = Y_{cal_{hard}} * Y_{scale}, \quad (1.4)$$

где *Ycal_hard* – калиброванное значение;

Y – исходное значение;

Yscale – коэффициент масштабирования.

Для расчета коэффициента масштабирования используется формула (1.5) представленная ниже [8].

$$Y_{scale} = \frac{A_{max} - A_{min}}{Y_{max} - Y_{min}}, \quad (1.5)$$

где *Yscale* – искомый коэффициент искажения по оси Y;

Amax – максимальное значение на некоторой оси;

Amin – минимальное значение на некоторой оси;

Ymax – максимальное значение на оси Y;

Ymin – минимальное значение на оси Y.

1.8 Цифровой фильтр EMA

Цифровой фильтр EMA (Exponential Moving Average) — это алгоритм, используемый для обработки цифровых сигналов в рамках цифровой обработки сигналов. Этот фильтр является одним из самых простых цифровых фильтров, который широко используется в обработке сигналов в различных областях, таких как радарная техника, телекоммуникации, медицинская техника и т. д.

Цифровой фильтр EMA использует метод сглаживания экспоненциального, скользящего среднего, который позволяет быстро удалять шумы из сигнала и предоставляет сглаженные значения с высокой точностью. Для реализации этой операции в фильтре EMA используется экспоненциальная функция, которая рассчитывает взвешенное среднее значений входного сигнала, при этом новые значения имеют больший вес.

Если обозначить входной сигнал как x[n], то формула (1.6) EMA будет выглядеть следующим образом [9]:

$$y[n] = (1 - \alpha) * y[n-1] + \alpha * x[n], \quad (1.6)$$

где *y[n]* – выходной сигнал;

y[n-1] – значение выходного сигнала в предыдущий момент времени;

x[n] – входной сигнал;

α – коэффициент сглаживания.

Передаточная функция EMA фильтра H(z) (1.7) [9]:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\alpha}{1 - (1 - \alpha)z^{-1}}, \tag{1.7}$$

АЧХ и ФЧХ фильтра, полученные из передаточной функции (1.7) представлены на рисунке 7.

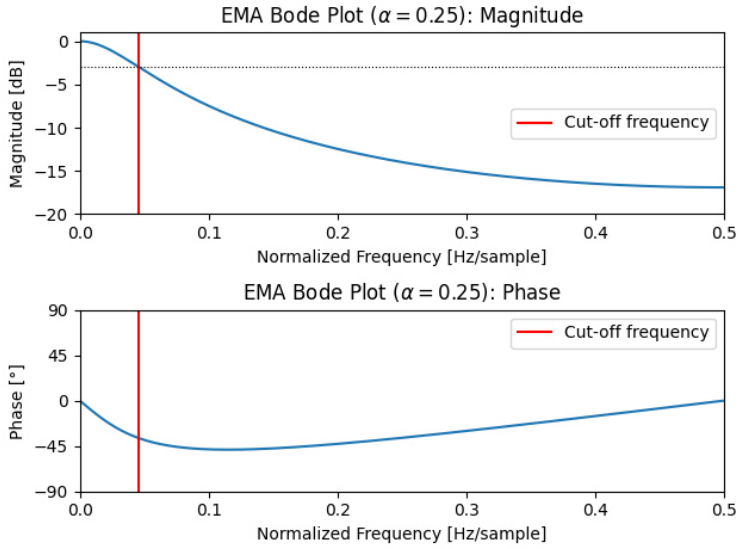


Рисунок 7 – АЧХ и ФЧХ фильтра [9]

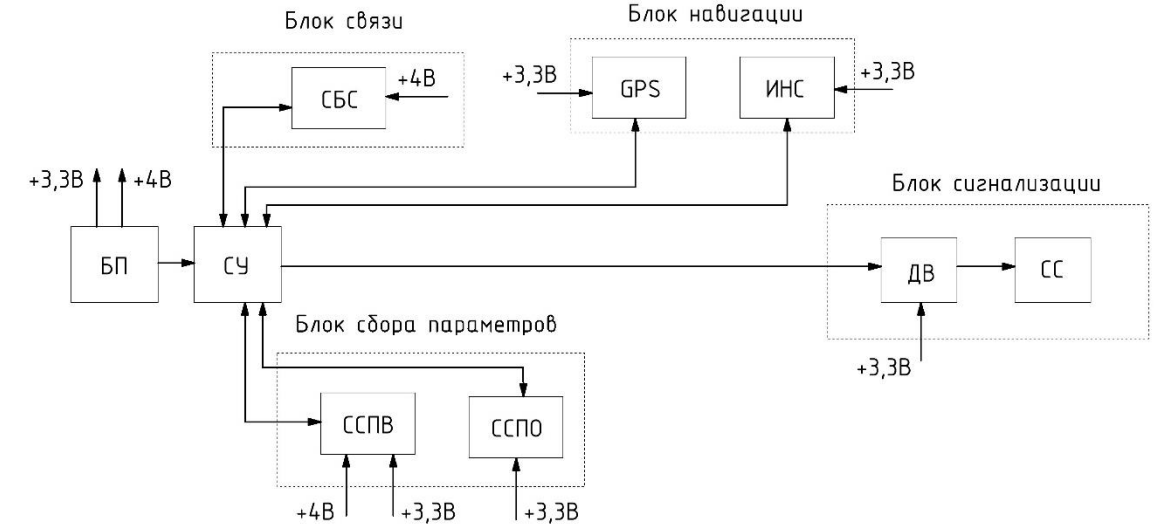
Коэффициент сглаживания, α – это параметр, который задает скорость изменения значения сигнала фильтра EMA. Если значение, а близко к 1, то фильтр EMA будет быстро реагировать на изменения сигнала, но также могут возникнуть высокие значения шумов. Если значение, а близко к 0, то фильтр EMA будет менее чувствителен к изменениям входного сигнала, но это может привести к медленному отклику на изменения сигнала.

Одной из особенностей цифрового фильтра EMA является возможность изменения скорости изменения значения сигнала в зависимости от времени. Для этого используются адаптивные значения коэффициента сглаживания, которые позволяют улучшить качество обработки сигнала в моменты времени, когда изменения сигнала наиболее значимы.

В целом, цифровой фильтр EMA – это простой и эффективный инструмент в обработке цифровых сигналов, который широко используется в различных областях, в том числе в радарной технике, телекоммуникациях и медицинской технике. Использование адаптивных значений коэффициента сглаживания позволяет улучшить качество обработки сигнала и повысить эффективность фильтрации шумов.

2 РАЗРАБОТКА СТРУКТУРНОЙ СХЕМЫ УСТРОЙСТВА

На рисунке 8 представлена структурная схема устройства по автоматизации речного судоходства.



БП – блок питания,
СБС – система беспроводной связи,
GPS – система GPS навигации,
ССПВ – система сбора параметров воды,
ССПО – система сбора параметров окружающей среды

СС – светосигнальная система,
ДВ – драйвер
СУ – система управления,
ИНС – инерциальная навигационная система
СОСВ – система отслеживания степени волнения

Рисунок 8 – Структурная схема

Блок питания (БП) обеспечивает требуемым напряжением каждый элемент схемы. В качестве источника будут выступать Li-ion аккумулятор. Для стабилизации и получения необходимого напряжения используются два преобразователя постоянного напряжения на 4 В и на 3.3 В.

Система управления (СУ) – обеспечивает взаимодействие и управление всех блоков, устройства. Для реализации данной системы будет использоваться МК STM32 F4-серии. Поскольку он оснащен необходимыми интерфейсами: UART, SPI, I2C; достаточным количеством памяти и частоты. Задача СУ будет заключаться в приеме и обработке данных со всем датчиков, последующей отправки этих данных через СБС и управлении СС.

Система беспроводной связи (СБС) – беспроводная связь будет обеспечивать контакт устройства с принимающей стороной (диспетчерами, лабораториями…) для передачи данных или приема команд. Для будет использоваться GSM модуль на базе микросхемы SIM800L, который сможет обеспечить связь на большом расстоянии. Данный блок будет принимать команды об изменении характера индикации и отправлять их в СУ.

Из СУ будут передаваться собранные данные от всех блоков в СБС и отправляться принимающей стороне.

GPS – позволит определять координаты устройства в реальном времени. Для этого подойдёт не дорогой и популярный GPS-модуль NEO 6M. Его характеристик будет достаточно позиционирования на открытой местности. Данный блок будет передавать полученные строки данных в СУ.

Система пространственного позиционирования (СПП) – будет определять угол наклона относительно вектора гравитационного поля земли и направления азимута устройства. Для этих целей будет использоваться магнитометр и акселерометр, входящие в микросхему LSM303DHLC, из-за подходящего разрешения и точности данных. Данный блок будет передавать полученные данные в СУ.

Блок сбора параметров воды включает систему сбора параметров воды, окружающей среды. Параметры воды отслеживаются следующими датчиками: датчик мутности представлен модулем TS-300B, датчик температуры реализован микросхемой DS18B20. Регистрация параметров воздуха осуществляется датчиком температуры DS18B20. Датчики температуры имеют достаточный диапазон температуру и имеют водонепроницаемое исполнение. Данный блок будет передавать полученные данные в СУ.

Блок сигнализации – будет представлять собой набор светоизлучающих элементов, в зависимости от местонахождения устройства и задач, поставленных перед ним, будет иметься возможность менять характер огня. Пользователь сможет изменять характер свечения отсылая команды с помощью SMS-сообщений, блок СБС будет принимать их и отправлять в СУ, которая будет включать нужный режим индикации.

3 РАЗРАБОТКА ПРОГРАММЫ ДЛЯ МИКРОКОНТРОЛЛЕРА

3.1 Разработка библиотеки для получения данных с GPS модуля

Для реализации получения данных с GPS потребуется использовать интерфейс UART для получения массива char. Данный модуль каждую секунду присылает пакет данных с разными GPS данными, исчерпывающая информация для данного проекта находится в строке с идентификатором RMC. Первое, что понадобится это из пакета данных выбрать именно эту строку (RMC) и записать ее в буфер, из которого в дальнейшем будет произведен разбор строки и извлечение из неё данных в более удобном формате. Алгоритм будет реализован в прерывании по выставлению флага RXNE (Received Data Ready to be Read) от UART. Алгоритм представлен на рисунке 9.

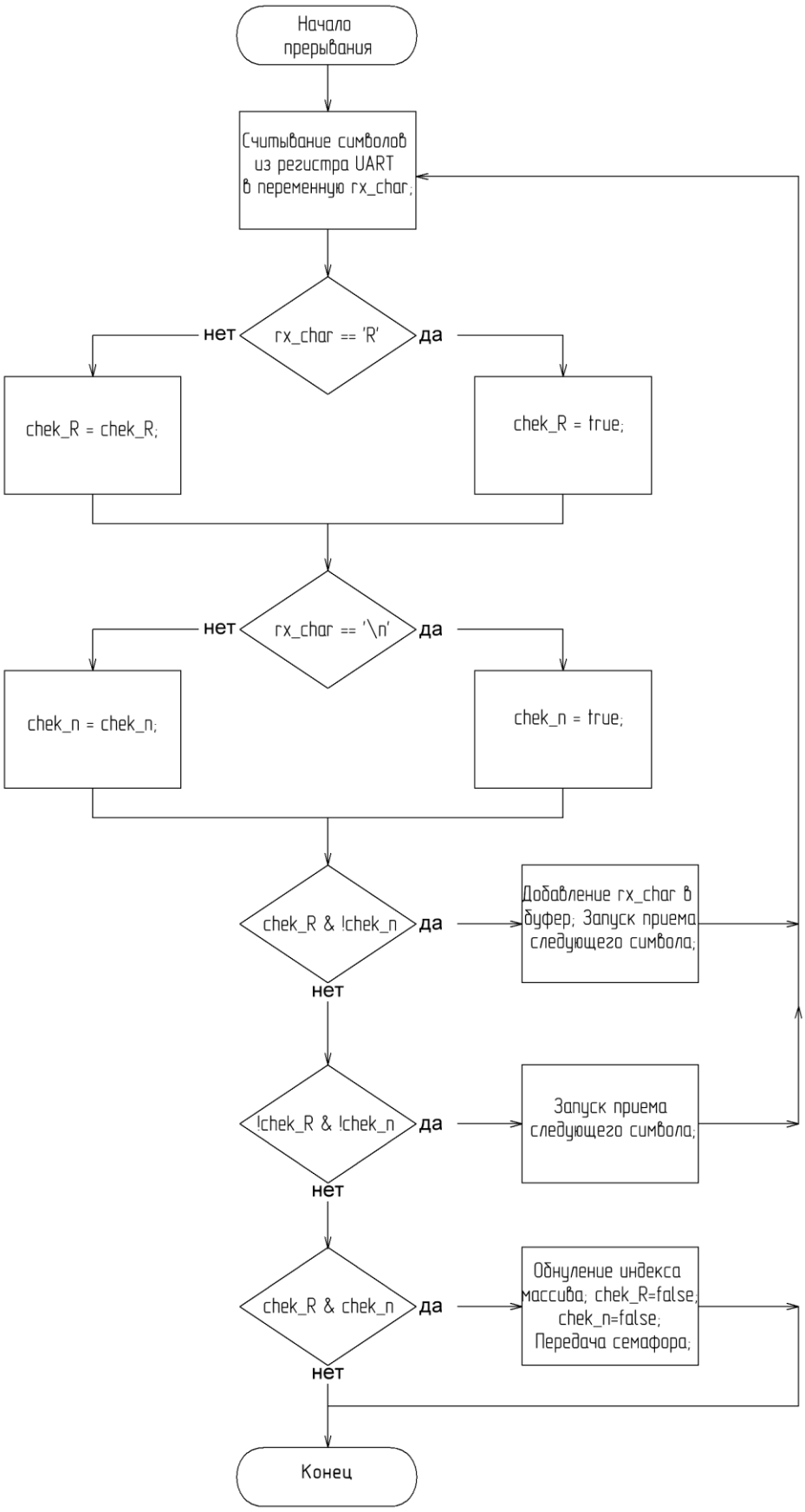


Рисунок 9 – Алгоритм поиска и записи RMC строки в буфер

Далее для была написана задача для операционной системы с использованием средств для синхронизации и безопасной передачи данных между задачами. Алгоритм представлен на рисунке 10. Полная программа на языке С представлена в приложении А.

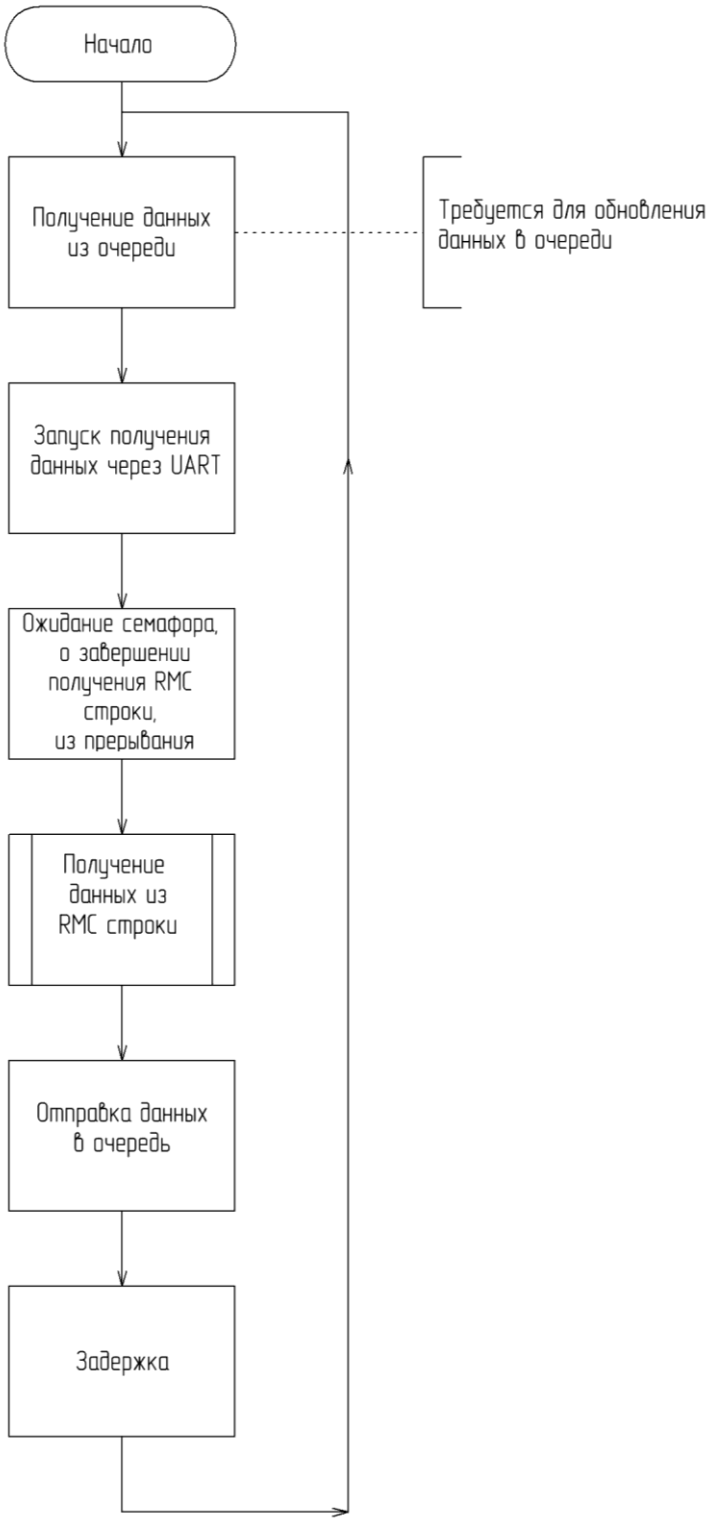


Рисунок 10 – Алгоритм задачи по получения GPS-данных

3.2 Разработка программы для получения угла наклона относительно гравитационного поля Земли и азимута

3.2.1 Настройка MEMS датчика LSM303DLHC

Настройка датчиков заключается в передачи данных в регистры микросхемы LSM303DLHC с помощью интерфейса I2C, для реализации используется библиотека HAL [10]. Для настройки акселерометра требуется установка определённых битов в регистры представленных на рисунке 11, на рисунке 14 регистры настройки магнитометра.

Name	Slave address	Type	Register address		Default	Comment
			Hex	Binary		
Reserved (do not modify)	Table 14		00 - 1F	--	--	Reserved
CTRL_REG1_A	Table 14	rw	20	010 0000	00000111	
CTRL_REG2_A	Table 14	rw	21	010 0001	00000000	
CTRL_REG3_A	Table 14	rw	22	010 0010	00000000	
CTRL_REG4_A	Table 14	rw	23	010 0011	00000000	
CTRL_REG5_A	Table 14	rw	24	010 0100	00000000	
CTRL_REG6_A	Table 14	rw	25	010 0101	00000000	
REFERENCE_A	Table 14	rw	26	010 0110	00000000	
STATUS_REG_A	Table 14	r	27	010 0111	00000000	
OUT_X_L_A	Table 14	r	28	010 1000	output	
OUT_X_H_A	Table 14	r	29	010 1001	output	
OUT_Y_L_A	Table 14	r	2A	010 1010	output	
OUT_Y_H_A	Table 14	r	2B	010 1011	output	
OUT_Z_L_A	Table 14	r	2C	010 1100	output	
OUT_Z_H_A	Table 14	r	2D	010 1101	output	
FIFO_CTRL_REG_A	Table 14	rw	2E	010 1110	00000000	
FIFO_SRC_REG_A	Table 14	r	2F	010 1111		
INT1_CFG_A	Table 14	rw	30	011 0000	00000000	
INT1_SRC_A	Table 14	r	31	011 0001	00000000	
INT1_THS_A	Table 14	rw	32	011 0010	00000000	
INT1_DURATION_A	Table 14	rw	33	011 0011	00000000	
INT2_CFG_A	Table 14	rw	34	011 0100	00000000	
INT2_SRC_A	Table 14	r	35	011 0101	00000000	
INT2_THS_A	Table 14	rw	36	011 0110	00000000	
INT2_DURATION_A	Table 14	rw	37	011 0111	00000000	
CLICK_CFG_A	Table 14	rw	38	011 1000	00000000	
CLICK_SRC_A	Table 14	rw	39	011 1001	00000000	
CLICK_THS_A	Table 14	rw	3A	011 1010	00000000	

Рисунок 11 – Карта регистров акселерометра [11]

Для реализации проекта были использованы следящие регистры акселерометра:

1. *CTRL_REG1_A*. Описание битов данного регистра представлено на рисунке 12.

ODR3	ODR2	ODR1	ODR0	LPen	Zen	Yen	Xen

Table 19. CTRL_REG1_A description

ODR[3:0]	Data rate selection. Default value: 0000 (0000: power-down, others: refer to *Table 20*)
LPen	Low-power mode enable. Default value: 0 (0: normal mode, 1: low-power mode)
Zen	Z-axis enable. Default value: 1 (0: Z-axis disabled, 1: Z-axis enabled)
Yen	Y-axis enable. Default value: 1 (0: Y-axis disabled, 1: Y-axis enabled)
Xen	X-axis enable. Default value: 1 (0: X-axis disabled, 1: X-axis enabled)

Рисунок 12 – Описание *CTRL_REG1_A* [11]

В данном регистре происходила следующая настройка параметров:

– включение X, Y, Z осей;

– настройка скорости обновления данных.

2. *CTRL_REG4_A*. Описание битов данного регистра представлено на рисунке 13.

BDU	BLE	FS1	FS0	HR	0(1)	0(1)	SIM

1. This bit must be set to '0' for correct operation of the device.

Table 27. CTRL_REG4_A description

BDU	Block data update. Default value: 0 (0: continuous update, 1: output registers not updated until MSB and LSB have been read
BLE	Big/little endian data selection. Default value 0. (0: data LSB @ lower address, 1: data MSB @ lower address)
FS[1:0]	Full-scale selection. Default value: 00 (00: ±2 *g*, 01: ±4 *g*, 10: ±8 *g*, 11: ±16 *g*)
HR	High-resolution output mode: Default value: 0 (0: high-resolution disable, 1: high-resolution enable)
SIM	SPI serial interface mode selection. Default value: 0 (0: 4-wire interface, 1: 3-wire interface).

Рисунок 13 – Описание *CTRL_REG4_A* [11]

В данном регистре происходила следующая настройка параметров:

– выбор максимального определяемой приложенной силы к акселерометру;

– режим обновления данных;

– разрешение данных.

Name	Slave address	Type	Register address		Default	Comment
			Hex	Binary		
TIME_LIMIT_A	Table 14	rw	3B	011 1011	00000000	
TIME_LATENCY_A	Table 14	rw	3C	011 1100	00000000	
TIME_WINDOW_A	Table 14	rw	3D	011 1101	00000000	
Reserved (do not modify)	Table 14		3E-3F	--	--	Reserved
CRA_REG_M	Table 16	rw	00	00000000	0001000	
CRB_REG_M	Table 16	rw	01	00000001	0010000	
MR_REG_M	Table 16	rw	02	00000010	00000011	
OUT_X_H_M	Table 16	r	03	00000011	output	
OUT_X_L_M	Table 16	r	04	00000100	output	
OUT_Z_H_M	Table 16	r	05	00000101	output	
OUT_Z_L_M	Table 16	r	06	00000110	output	
OUT_Y_H_M	Table 16	r	07	00000111	output	
OUT_Y_L_M	Table 16	r	08	00001000	output	
SR_REG_M	Table 16	r	09	00001001	00000000	
IRA_REG_M	Table 16	r	0A	00001010	01001000	
IRB_REG_M	Table 16	r	0B	00001011	00110100	
IRC_REG_M	Table 16	r	0C	00001100	00110011	
Reserved (do not modify)	Table 16		0D-30	--	--	Reserved
TEMP_OUT_H_M	Table 16		31	00000000	output	
TEMP_OUT_L_M	Table 16		32	00000000	output	
Reserved (do not modify)	Table 16		33-3A	--	--	Reserved

Рисунок 14 – Карта регистров магнетометра [11]

Для реализации проекта были использованы следящие регистры магнетометра:

1. *CRA_REG_M*. Описание битов данного регистра представлено на рисунке 15.

TEMP_EN	0(1)	0(1)	DO2	DO1	DO0	0(1)	0(1)

1. This bit must be set to '0' for correct operation of the device.

Table 71. CRA_REG_M description

TEMP _EN	Temperature sensor enable. 0: temperature sensor disabled (default), 1: temperature sensor enabled
DO[2:0]	Data output rate bits. These bits set the rate at which data is written to all three data output registers (refer to Table 72). Default value: 100

Table 72. Data rate configurations

DO2	DO1	DO0	Minimum data output rate (Hz)
0	0	0	0.75
0	0	1	1.5
0	1	0	3.0
0	1	1	7.5
1	0	0	15
1	0	1	30
1	1	0	75
1	1	1	220

Рисунок 15 – Описание *CRA_REG_M* [11]

В данном регистре происходила следующая настройка параметров:

– установка скорости обновления данных.

2. *CRB_REG_M*. Описание битов данного регистра представлено на рисунке 16.

Table 73. CRB_REG_M register

GN2	GN1	GN0	0(1)	0(1)	0(1)	0(1)	0(1)

1. This bit must be set to '0' for correct operation of the device.

Table 74. CRB_REG_M description

GN[2:0]	Gain configuration bits. The gain configuration is common for all channels (refer to *Table 75)*

GN2	GN1	GN0	Sensor input field range [Gauss]	Gain X, Y, and Z [LSB/Gauss]	Gain Z [LSB/Gauss]	Output range
0	0	1	±1.3	1100	980	0xF800–0x07FF (-2048 to +2047)
0	1	0	±1.9	855	760	
0	1	1	±2.5	670	600	
1	0	0	±4.0	450	400	
1	0	1	±4.7	400	355	
1	1	0	±5.6	330	295	
1	1	1	±8.1	230	205	

Рисунок 16 – Описание *CRB_REG_M* [11]

В данном регистре происходила следующая настройка параметров:

– установка максимального воспринимаемого значение магнитной индукции.

3. *MR_REG_M*. Описание битов данного регистра представлено на рисунке 17.

MR_REG_M (02h)

Table 76. MR_REG_M register

0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	MD1	MD0

1. This bit must be set to '0' for correct operation of the device.

Table 77. MR_REG_M description

MD[1:0]	Mode select bits. These bits select the operation mode of this device (refer to *Table 78*)

Table 78. Magnetic sensor operating mode

MD1	MD0	Mode
0	0	Continuous-conversion mode
0	1	Single-conversion mode
1	0	Sleep mode. Device is placed in sleep mode
1	1	Sleep mode. Device is placed in sleep mode

Рисунок 17 – Описание *MR_REG_M* [11]

В данном регистре происходила следующая настройка параметров:

– настройка режима приема данных.

3.2.2 Калибровка магнетометра и акселерометра

Программа Magneto

Для описанного в пункте 1.7 способа калибровки требуется специальный стенд, но существует иной способ. Программа Magneto предоставляет из себя автоматический алгоритм калибровки, суть алгоритма в аппроксимации облака полученных точек эллипсоида (подбор параметров эллипсоида таким образом, чтобы он максимально точно совпадал с полученным облаком точек, построенных на основе показаний магнитометра. Из подобранных таким образом параметров получить величину смещения, коэффициенты масштаба и коэффициенты для ортогонализации осей [12]. Интерфейс программы представлен на рисунке 18.

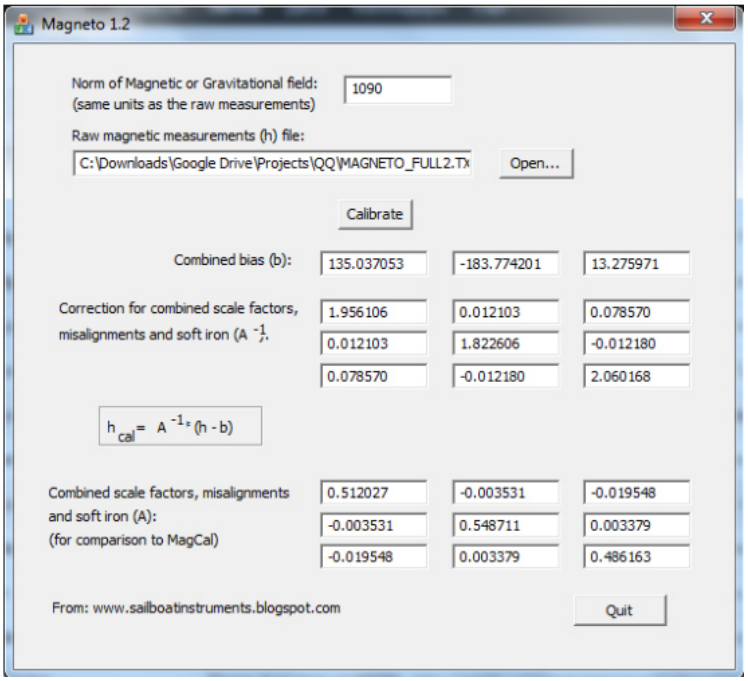


Рисунок 18 – Интерфейс программы

Итоговая формула (4.1) для калибровки магнетометра [12]:

$$h_{cal} = A^{-1} * (h - b), \tag{4.1}$$

где A^{-1}– полученная матрица из программы;

h_{cal} – откалиброванное значение магнетометра;

h – исходные вектор значений датчика;

b – вектор смещения.

Реализация калибровки магнетометра

Первым этапом для калибровки было создание функции по считыванию данных из датчика.

Функция по считыванию:

```
static void LSM303DLHC_MagGetData (LSM303DLHC_HandleTypeDef* LSM303DLHC_Handler,
LSM303DLHC_Data_Type* Data)
{
	uint8_t buffer[6] = {0,};

	HAL_I2C_Mem_Read(LSM303DLHC_Handler->I2C_handler, LSM303DLHC_MAG_ADDRESS,
LSM303DLHC_OUT_X_H_M|LSM303DLHC_MULTIPLE_FLAG, I2C_MEMADD_SIZE_8BIT, buffer, 6,
LSM_MAX_TIMEOUT);

	Data->data_M[0] = (int16_t)((buffer[0] << 8) | buffer[1]);
	Data->data_M[1] = (int16_t)((buffer[4] << 8) | buffer[5]);
	Data->data_M[2] = (int16_t)((buffer[2] << 8) | buffer[3]);
}
```

В данном случае была применена возможность мультибайтовой передачи, для этого использовался флаг:

```
#define LSM303DLHC_MULTIPLE_FLAG	0x80
```

Регистр для скачивания находился по адресу:

```
#define LSM303DLHC_OUT_X_H_M		0x03
```

Функция для считывания и отправки координат по интерфейсу UART:

```
void LSM303DLHC_TransMagData (LSM303DLHC_HandleTypeDef* LSM303DLHC_Handler,
UART_HandleTypeDef* UART_Handle)
{
	LSM303DLHC_Data_Type DataXYZ;
	uint8_t buffer[23];
	LSM303DLHC_MagGetData(LSM303DLHC_Handler, &DataXYZ);
	sprintf(buffer, "%6d;%6d;%6d;\r\n", DataXYZ.data_M[0], DataXYZ.data_M[1],
DataXYZ.data_M[2]);
	HAL_UART_Transmit(UART_Handle, buffer, sizeof(buffer), 0xFF);
	HAL_Delay(10);
}
```

Далее для визуального отображения (оно требуется, чтобы видеть, что было получено достаточно точек и были считаны все полюса) создался код в программе MatLab.

Код в программе MatLab:

```
%%Инициализация порта
s =  serialport('COM3', 115200);

fileID = fopen ('exacell.txt','w');

bbb = 1;
%% Считывание 30000 координат
for n=1:30000

    %%Считывание координат
    C = strsplit (readline(s), ';');
    mx = str2double(C(1));
    my = str2double(C(2));
    mz = str2double(C(3));
    %%Отрисовка координат
    scatter3(mx,my,mz,3);

    hold on;
    axis equal;

    fprintf(fileID, '%d %d %d \r\n', mx, my, mz);
```

```
    pause(0.010);
end

fclose(fileID);
delete(s);
clear s;
```

Полученное облако данных представлен без калибровки на рисунке 19, данные с калибровкой на рисунке 20.

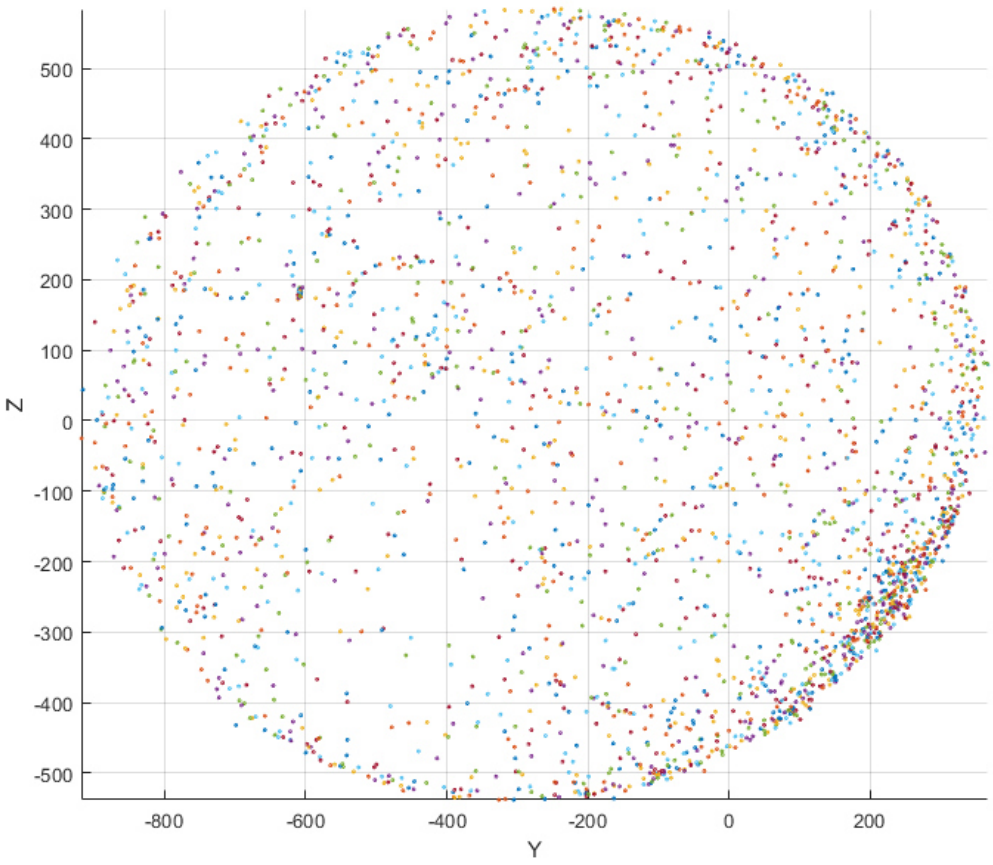


Рисунок 19 – Облако данных без калибровки

Полученные матрица A и вектор смещения h были определены как константы и произвелась повторная калибровка и отрисовка данных.

```
const float CalibMatix_h_M[3] = {-32.335960, -272.726525, 22.747240};

const float CalibMatix_A_M[3][3] =      {{0.887445, -0.028208, 0.014490},
                                         {-0.028208, 0.895270, -0.012892},
                                         {0.014490, -0.012892, 0.998620}};
```

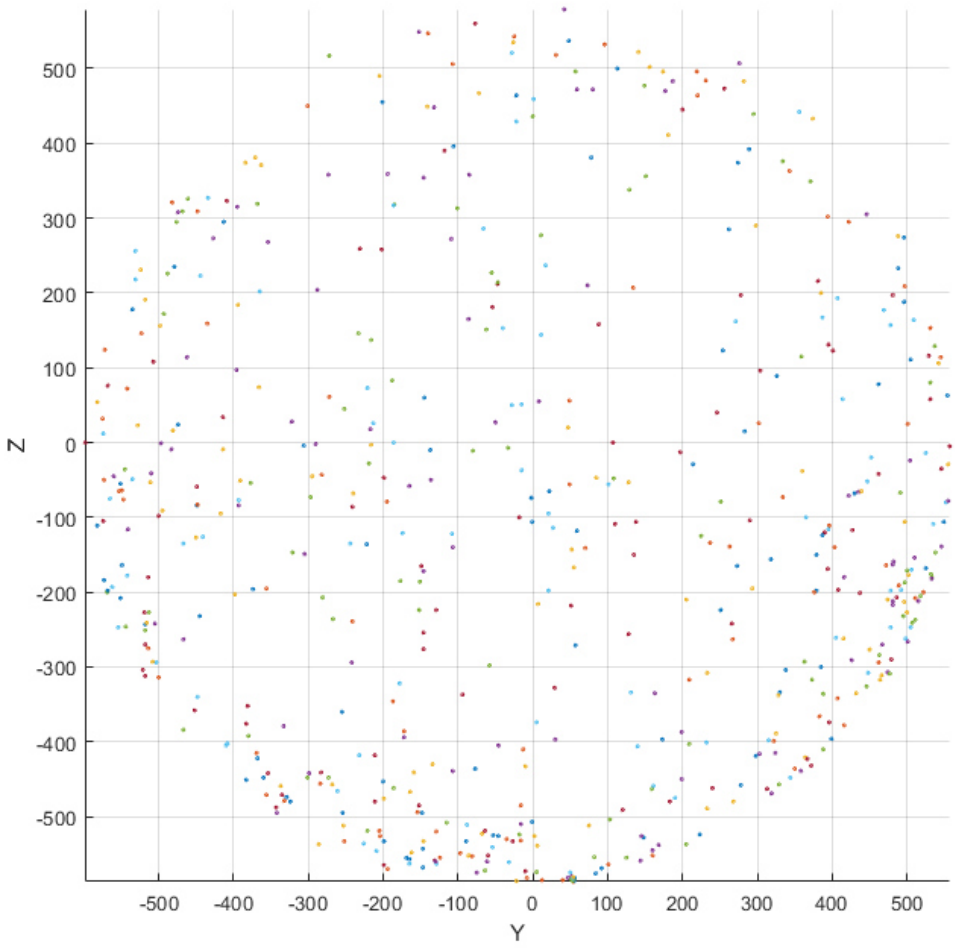


Рисунок 20 – Облако данных после калибровки

Из полученных данных следует, что полученные данные прошли калибровку и образуют сферу, что и требовалось получить.

Калибровка акселерометра

Искажения акселерометра могут возникать из-за различных факторов, таких как дрейф, шум, нелинейность и температурные изменения. Чтобы снизить влияние этих искажений, можно использовать калибровку акселерометра с помощью специальных тестовых процедур. Также можно применять фильтрацию сигнала и компенсацию дрейфа.

Для нивелирования дрейфа воспользуемся тем же методом, что был описан выше с магнетометром, единственное различие, которое стоит отметить это изменившееся функция для считывания данных из акселерометра:

```
void LSM303DLHC_AccelGetData (LSM303DLHC_HandleTypeDef* LSM303DLHC_Handler,
LSM303DLHC_Data_Type* Data)
{
	uint8_t buffer[6] = {0,};
```

```
	HAL_I2C_Mem_Read(LSM303DLHC_Handler->I2C_handler, LSM303DLHC_ACCEL_ADDRESS,
LSM303DLHC_OUT_X_L_A|LSM303DLHC_MULTIPLE_FLAG, I2C_MEMADD_SIZE_8BIT, buffer, 6,
LSM_MAX_TIMEOUT);

	Data->data_A[0] = (int16_t)((buffer[1] << 8) | buffer[0]);
	Data->data_A[1] = (int16_t)((buffer[3] << 8) | buffer[2]);
	Data->data_A[2] = (int16_t)((buffer[5] << 8) | buffer[4]);

}
```

Адрес регистра для считывания:

```
#define LSM303DLHC_OUT_X_L_A              0x28
```

Полученные данные с акселерометра представлены на рисунке 21 и 22.

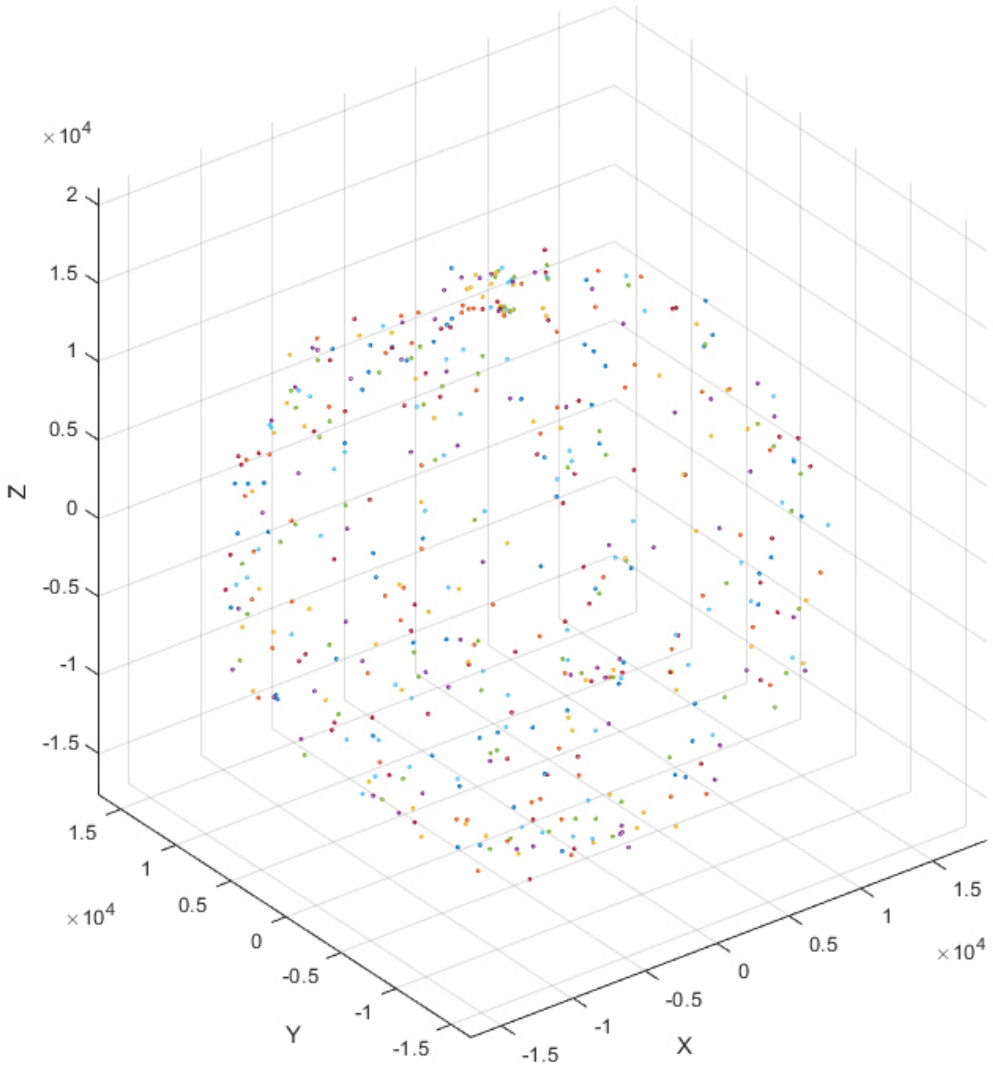


Рисунок 21 – Некалиброванные данные с акселерометра

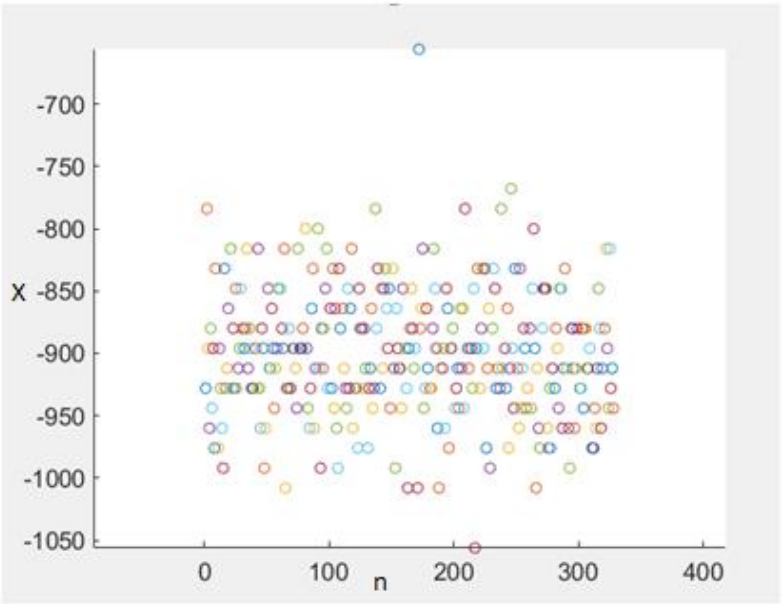


Рисунок 22 – Шумы в статическом положении на одной из осей

Как можно видеть смещения для акселерометра не характерны, но присутствуют значительные шумы, данные шумы будут значительно влиять на дальнейшие измерения. Для нивелирования шума будет использоваться цифровой EMA фильтр.

3.2.3 Алгоритм нахождения азимута и угла склонения

Вектор магнитного поля Земли и вектор гравитационного поля земли образуют 2 неколлинеарных вектора в системе, связанной с устройством. Положение векторов представлено на рисунке 23.

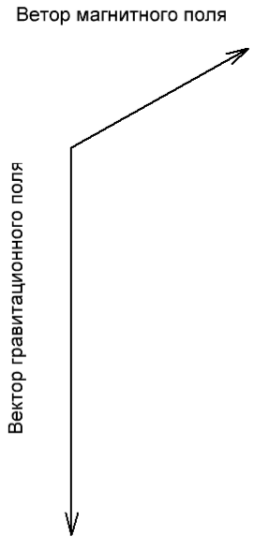


Рисунок 23 – Положение векторов

Для реализации комплексной обработки данных был реализован алгоритм. Ниже были описаны основные функции, которые использовались.

Преобразование в ортогональные векторы

Перпендикулярными векторы становятся на экваторе. Для того, чтобы сделать их ортогональными, нужно получить вектор Rejection, используя изначальный вектор магнитного поля и вектор гравитационного поля. Иллюстрация преобразований представлена на рисунке 24.

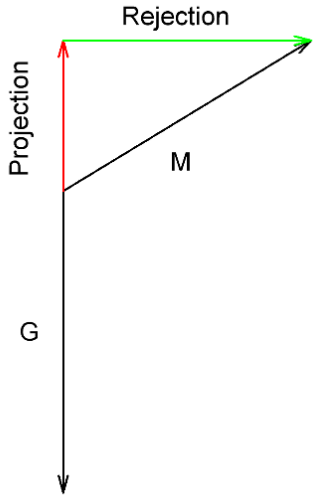


Рисунок 24 – Пример Rejection вектора

Для данного преобразования используется формула (4.2) проекции вектора на вектор (Projection):

$$\text{пр}_{\overline{V2}}\overline{V1} = \frac{\overline{V1} * \overline{V2}}{|\overline{V2}|}, \tag{4.2}$$

где $\overline{V1}$ – первый вектор;

$\overline{V2}$– второй вектор;

$\text{пр}_{\overline{V2}}\overline{V1}$ – проекция $\overline{V1}$ на $\overline{V2}$.

Как видно из рисунка 24 вектор M это сумма векторов M и вектора Projection, следовательно, вектор Rejection будет являться разностью вектора M и вектора Projection.

$$\overline{Rejection} = \bar{M} - \overline{\text{пр}_{\bar{G}}\bar{M}}, \tag{4.3}$$

где $\overline{Rejection}$ – вектор обратный проекции;

$\bar{M}$ – вектор магнитного поля;

$\bar{G}$ – вектор гравитационного поля земли.

Функция для реализации данного преобразования:

```
void MATRIX_Vect_3f_Reject(Vector_3f_t V1, Vector_3f_t V2, Vector_3f_t V_result )
```

```
{
        float temp = 0.f;
        temp = MATRIX_Vect_3f_Dot(V1, V2) / MATRIX_Vect_3f_LengthSqr(V2);

        V_result[0] = V1[0] - V2[0] * temp;
        V_result[1] = V1[1] - V2[1] * temp;
        V_result[2] = V1[2] - V2[2] * temp;
}
```

Нормализация векторов

Для правильной обработки происходит нормализация векторов с использованием функции. Итоговое состояние системы представлено на рисунке 25.

```
void MATRIX_Vect_3f_Notmalize(Vector_3f_t V, Vector_3f_t V_result)
{
        float temp = 0.f;
        temp = 1.f/MATRIX_Vect_3f_Length(V);

        V_result[0] = V[0] * temp;
        V_result[1] = V[1] * temp;
        V_result[2] = V[2] * temp;
}
```

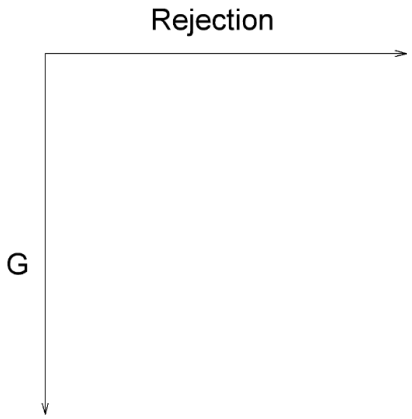


Рисунок 25 – Нормализованные векторы

Получение третьего ортогонального вектора

Векторное произведение — это одна из операций над векторами в трехмерном пространстве. Результатом векторного произведения двух векторов является третий вектор, который перпендикулярен плоскости, образованной исходными векторами. Длина этого вектора равна произведению длин исходных векторов на синус угла между ними.

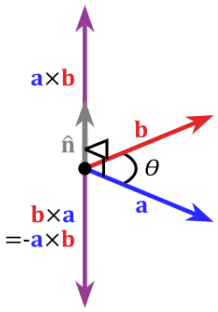


Рисунок 26 – Векторное произведение

Формула (4.4) для вычисления векторного произведения двух векторов a и b в координатной форме выглядит следующим образом:

$$[\bar{a}, \bar{b}] = \{a_2 b_3 - a_3 b_2; a_3 b_1 - a_1 b_3; a_1 b_2 - a_2 b_1\}, \quad (4.4)$$

где (a_1, a_2, a_3) и (b_1, b_2, b_3) - координаты исходных векторов.

Для этого была разработана следующая функция:

```
void MATRIX_Vect_3f_Cross (Vector_3f_t V1, Vector_3f_t V2, Vector_3f_t V_result)
{
	V_result[0] = V1[1]*V2[2] - V1[2]*V2[1];
	V_result[1] = V1[2]*V2[0] - V1[0]*V2[2];
	V_result[2] = V1[0]*V2[1] - V1[1]*V2[0];
}
```

Данные вектора, изображенные на рисунке 27, образуют декартову систему координат, которая определяет положение устройства относительно вектора магнитного поля, гравитационного вектора и вектора векторного умножения.

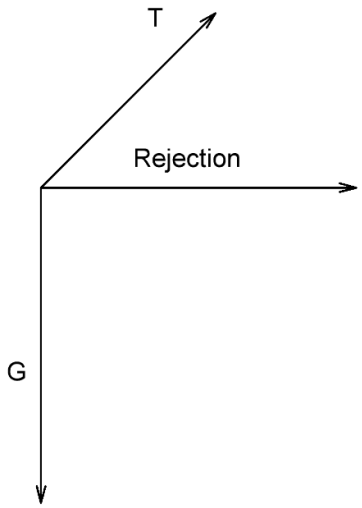


Рисунок 27 – Полученные векторы

Данные вектора образуют декартову систему координат, которая определяет положение устройства относительно вектора магнитного поля, гравитационного вектора и вектора векторного умножения.

Получение азимута и угла относительно гравитационного поля земли

Для определения азимута потребуется найти один из углов Эйлера, а именно угол вращения (yaw).

Для этого 3 полученных вектора запишем в 1 матрицу [3x3] из и нею получим углы Эйлера.

Функция нахождения углов Эйлера:

```
void MATRIX_Matrix_3f_GetEuler (Matrix_3f_t M1, Vector_3f_t Euler)
{
	Euler[0] = atan2(-M1[1][2],M1[2][2]);
	Euler[1] = asin(M1[0][2]);
	Euler[2] = atan2(-M1[0][1],M1[0][0]);
}
```

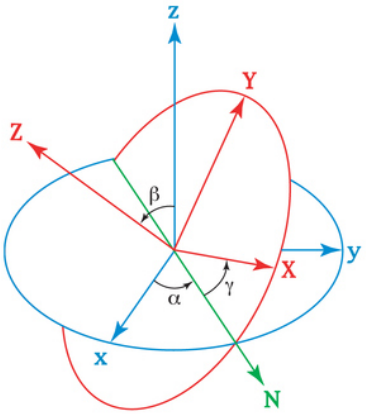


Рисунок 28 – Углы Эйлера

Для нахождения угла между осью Z прибора и вектором гравитационной силы будет рассчитан исходя из формулы (4.2) скалярного умножения векторов.

Функция для расчёта угла между двумя векторами:

```
float MATRIX_Vect_3f_Ang (Vector_3f_t V1, Vector_3f_t V2)
{
	return acos( MATRIX_Vect_3f_Dot(V1, V2)/sqrt(
MATRIX_Vect_3f_LengthSqr(V1)*MATRIX_Vect_3f_LengthSqr(V2) ) );
}
```

Итоговый алгоритм комплексной обработки данных представлен на рисунке 29:

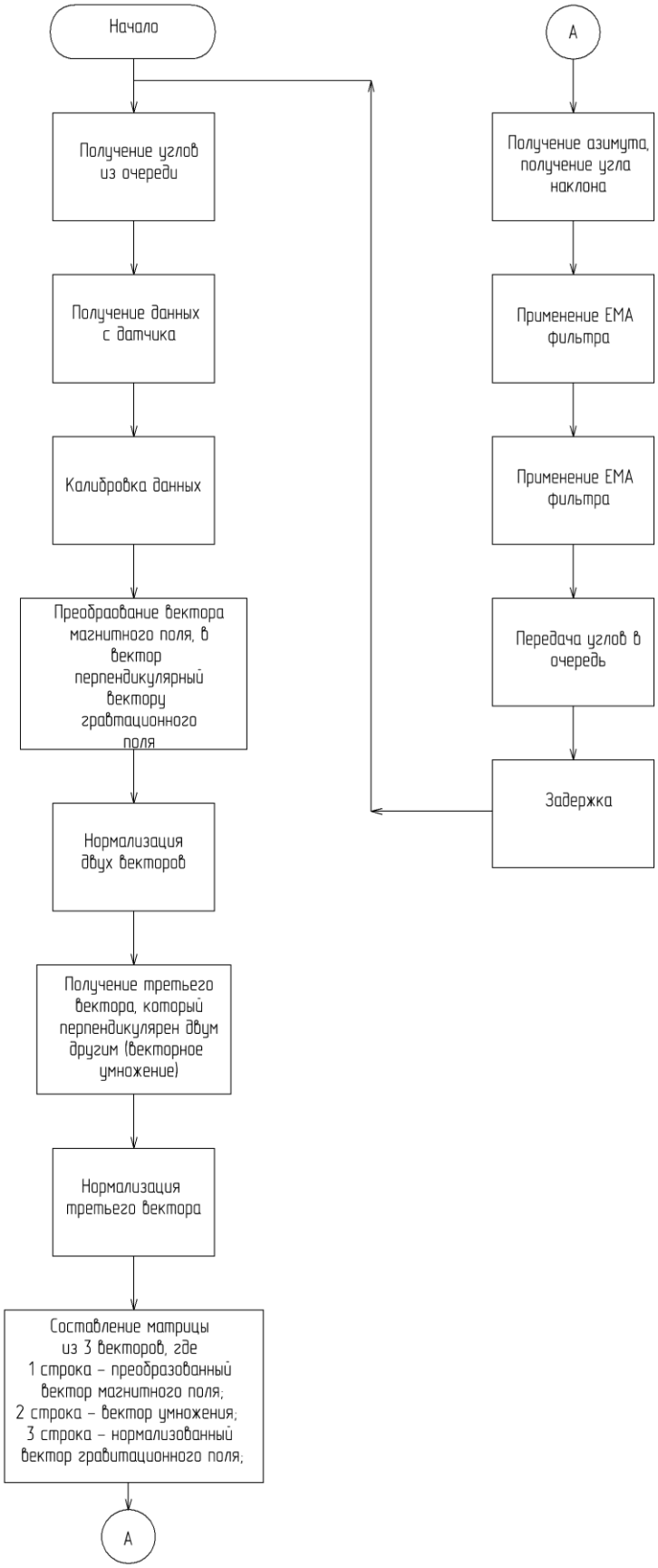


Рисунок 29 – Алгоритм задачи по получению и обработки данных из датчика

3.3 Разработка программы GSM модуля

3.3.1 Первоначальные настройки

Для взаимодействия с GSM-модулем используются AT-команды. Интерфейсом для взаимодействия служит UART. Первоначальные настройки GSM модуля заключались в следующем [13]:

– включение расширенного отчета об ошибках. AT-команда: `"AT+CMEE=2\r\n"`;
– установка текстового режима SMS. AT-команда: `"AT+CMGF=1\r\n"`;
– отключение режима эха. AT-команда: `"ATE0\r\n"`;
– автоматический прием SMS сообщений. AT-команда: `"AT+CNMI=2,2,0,0,0\r\n"`;
– удаление предыдущих сообщений. AT-команда: `"AT+CMGDA=\"DEL ALL\"\r\n"`.

Для приема ответов от GSM модуля из-за разного размера ответов потребуется включить прерывание по концу приема данных. Для этого активируется бит IDLEIE в регистре USART_CR1 [14].

Для передачи данных через UART будет использоваться режим DMA, для уменьшения нагрузки ядра процессора.

3.3.2 Отслеживание ответа от GSM модуля

Требуемые ответы для отслеживания в данной реализации программы будут [13]:

– отслеживание успешной отправки SMS-сообщений (Ключевое слово: `+CMGS`);
– отслеживание получения нового SMS-сообщения (Ключевое слово: `+CMT`);

Для этого после вызова CallBack-функции о завершении получаемых данных (для этого используется семафор `xBinarySemaphore_GSM_recive_data`, далее в задаче `vTask_GSM_Chek_Response` происходит поиск ключевых слов и передаются семафоры `xBinarySemaphore_GSM_send_sms`, `xBinarySemaphore_GSM_recive_sms`, которые дают команду об успешной отправке SMS, либо о пришедшем сообщении.

CallBack-функции:

```
void HAL_UARTEx_RxEventCallback(UART_HandleTypeDef *huart, uint16_t Size)
{
	if(huart == &GSM_CHOSEN_UART)
	{
		xSemaphoreGiveFromISR(xBinarySemaphore_GSM_recive_data, NULL);
		HAL_UARTEx_ReceiveToIdle_IT(&GSM_CHOSEN_UART, recive_buffer,
GSM_RECIVE_BUFFER_SIZE);
	}
}
```

Задача по обработке пришедших строк:

```
for(;;)
{
		xSemaphoreTake(xBinarySemaphore_GSM_recive_data, portMAX_DELAY);
```

```
            if(strstr((char*)recive_buffer ,"+CMGS") != NULL &&
strstr((char*)recive_buffer ,"OK") != NULL) /*Tracking the success of sending a
message.*/
                  xSemaphoreGive(xBinarySemaphore_GSM_send_sms);
            else if(strstr((char*)recive_buffer ,"+CMT") != NULL) /*Tracking the
arrival of a new message.*/
                  xSemaphoreGive(xBinarySemaphore_GSM_recive_sms);
            else
                  memset(recive_buffer, '\0', GSM_RECIVE_BUFFER_SIZE);
}
```

3.3.3 Обработка SMS сообщений

Управление светосигнальными огнями будет осуществляться с помощью приема SMS – сообщения, которое будет содержать команду по включению режима индикации. В таблице 1 представлены режимы индикации согласно ГОСТ 26600-98 «Знаки навигационные внутренних судоходных путей» [15].

Таблица 1 – Режимы работы индикации

SMS-команда	Характер навигационного огня по ГОСТ 26600-98	Временные характеристики	
		Вспышка (проблеск света)	Пауза (затемнение)
Command:0	Однопроблесковый	Один импульс 0,70 с	Одна пауза 2,80 с
Command:1	Затмевающийся	Один импульс 2,80 с	Одна пауза 0,73 с
Command:2	Частопроблесковый	Один импульс 0,59 с	Одна пауза 0,40 с
Command:3	Пульсирующий	Один импульс 0,066 с	Одна пауза 0,055 с
Command:4	Группочастопроблесковый	Пачка (четыре импульса по 0,50 с и три паузы по 0,50 с)	Одна пауза между пачками 2,90 с
Command:5	Прерывистый пульсирующий	Пачка (три импульса по 0,06 с и две паузы по 0,06 с)	Одна пауза между пачками 2,80 с
Command:6	Двухпроблесковый	Пачка (два импульса по 0,60 с и пауза между ними 0,60 с)	Одна пауза между пачками 2,80 с

В задаче по обработке SMS-сообщения происходит извлечение режима и отправка в очередь. Функция по обработке SMS-сообщения.

```
static void SMS_processing(uint8_t* Text)
{
      char* temp = NULL;
      uint8_t  mode = 0;
      temp = strstr((char*)Text,"Mode:");
      temp = strstr((char*)temp,":");
      mode = *(temp+1) - '0';
      xQueueSendToBack(xQueue_GSM_set_mode, &mode, 0);
}
```

Далее в задаче, которая управляет индикацией, происходит выбор полученного режима. Задача по управлению индикацией:

```
static void vTask_Led (void* pvParameter)
{
      TickType_t pxPreviousWakeTime = xTaskGetTickCount ();
```

```
	uint8_t mode = 0;
	for(;;)
	{
		xQueueReceive(xQueue_GSM_set_mode, &mode, 0);
		switch (mode)
		{
			case LED_Mode1:
				Mode_1 (&pxPreviousWakeTime);
				break;
			case LED_Mode2:
				Mode_2 (&pxPreviousWakeTime);
				break;
			case LED_Mode3:
				Mode_3 (&pxPreviousWakeTime);
				break;
			case LED_Mode4:
				Mode_4 (&pxPreviousWakeTime);
				break;
			case LED_Mode5:
				Mode_5 (&pxPreviousWakeTime);
				break;
			case LED_Mode6:
				Mode_6 (&pxPreviousWakeTime);
				break;
			case LED_Mode7:
				Mode_7 (&pxPreviousWakeTime);
				break;
			default:
				Mode_1 (&pxPreviousWakeTime);
				break;
		}
	}
	vTaskDelete(NULL);
}
```

3.3.4 Отправка SMS сообщений

Отправка SMS-сообщения требует нескольких шагов [13]:

1. отправка AT-команды: `AT+CMGS="Номер телефона"\r\n`;
2. отправка текста SMS-сообщения;
3. отправка AT-команды: `\x1A\r\n`.

Для формирования текста сообщения используются функции по получению данных из очереди, далее формируется набор строк с данными. Протокол SMS-сообщений представлен в таблице 2.

Таблица 2 – Протокол сообщения

Состав строки	Отправляемые данные
Azimuth: **X**\r\n	**X** – азимут
Angle of inclination: **X**\r\n	**X** – угол наклона относительно гравитационного поля земли
Data fix: **X**\r\n	**X –** статус полученных координат с GPS (1 – данные подтверждены, 0 – данные не подтверждены)
Date: **X**.**Y**.**Z**\r\n	**X –** день, **Y –** месяц, **X –** год
Time: **X**: **Y**\r\n	**X –** часы, **Y** – минуты
Latitude: **X**\r\n	**X –** широта (градусы)
Latitude direction: **X**\r\n	**X –** направление широты
Longitude: **X**\r\n	**X –** долгота (градусы)
Longitude direction: **X**\r\n	**X –** направление долготы
Temp1: **X**\r\n	**X –** температура с окружающей среды (градусы цельсия) (при ошибке X = 99)
Temp2: **X**\r\n	**X –** температура воды (градусы цельсия) (при ошибке X = 99)
Turbidity percentage: **X**\r\n	**X –** прозрачность воды (проценты)

4 РАЗРАБОТКА ПРИНЦИПИАЛЬНОЙ СХЕМЫ

4.1 Схема подключения GPS-модуля NEO 6M

Схема подключения микросхемы GPS-модуля NEO 6M представлена на рисунке 30.

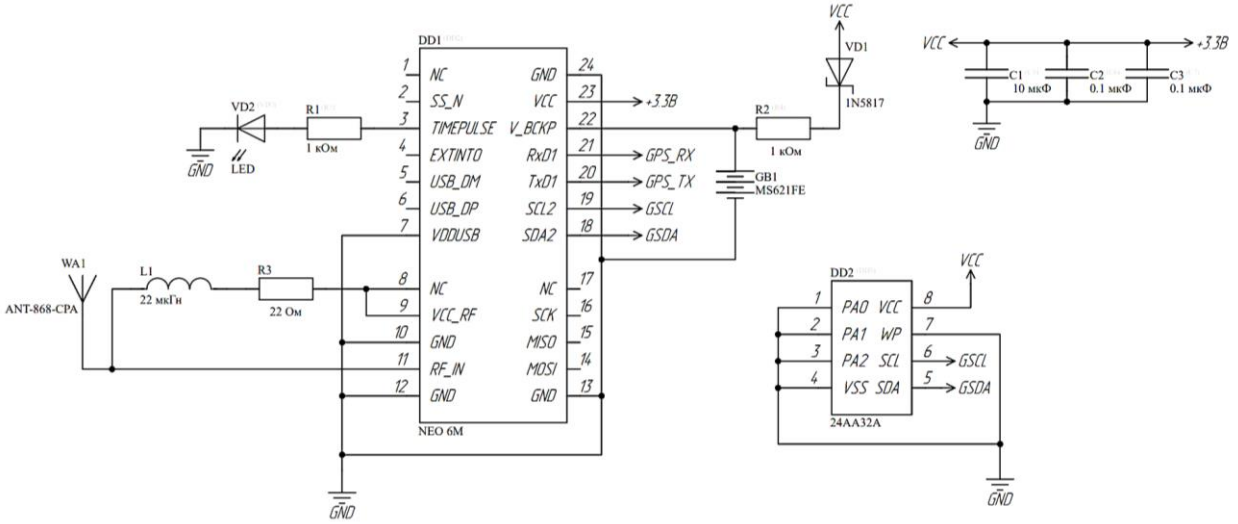


Рисунок 30 – Схема подключения микросхемы NEO 6M

Данная микросхема питается он напряжения 3.3 В, для сглаживания низкочастотных колебаний применяется конденсатор C1, для сглаживания более высокочастотных помех применяются керамические конденсаторы более низкой емкости C2, C3.

Для индикации статуса получаемых данных используется светодиод VD2 с ток ограничительным резистором (R1).

Для сохранения последних принятых данных в память микросхемы используется батарея GB1, для ограничения тока заряда используется резистор R2, так для отсечения обратного используется диод VD1.

Для хранения настроек модуля используется микросхема флэш-памяти 24AA32A с интерфейсом I2C [16].

Выводы GPS_RX и GPS_TX подключаются к микроконтроллеру.

4.2 Схема подключения датчика LSM303DHLC

На рисунке 31 представлена схема подключения MEMS-датчика LSM303DHLC, выбор элементов и подбор номинальных значений для резистор и конденсаторов исходил из рекомендаций разработчика [11].

Выводы LSM_SCL и LSM_SDA подключаются к интерфесу I2C микроконтроллера.

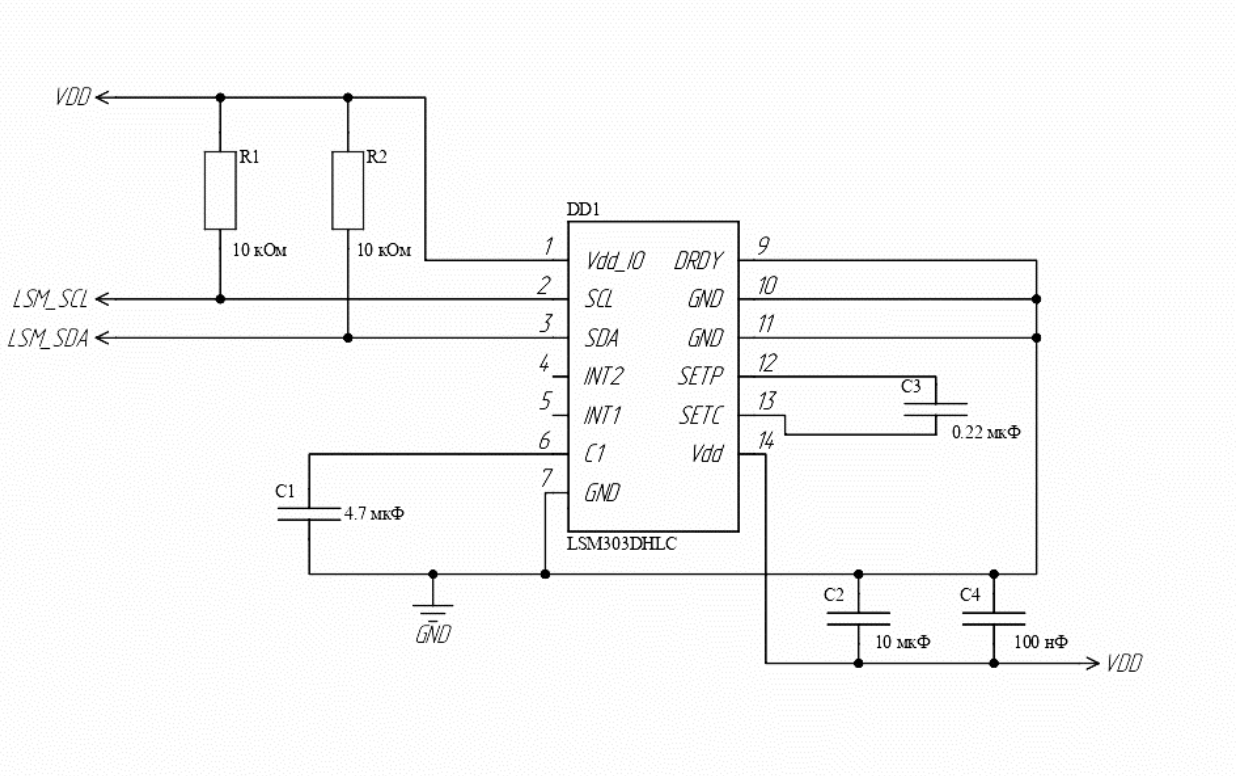


Рисунок 31 – Схема подключения LSM303DHLC

4.3 Схема подключения GSM-модуль ESIM800L

На рисунке 32 представлена схема подключения микросхемы ESIM800L. Для подключения сим-карты используется разъем 78646-3001.

Делитель с резисторами R2, R1 используется для согласования напряжений ножек микроконтроллера (3.3 В) и допустимого напряжения микросхемы (2.8 В) [17]. Принципиальная схема и перечень элементов представлен в приложение Б и В.

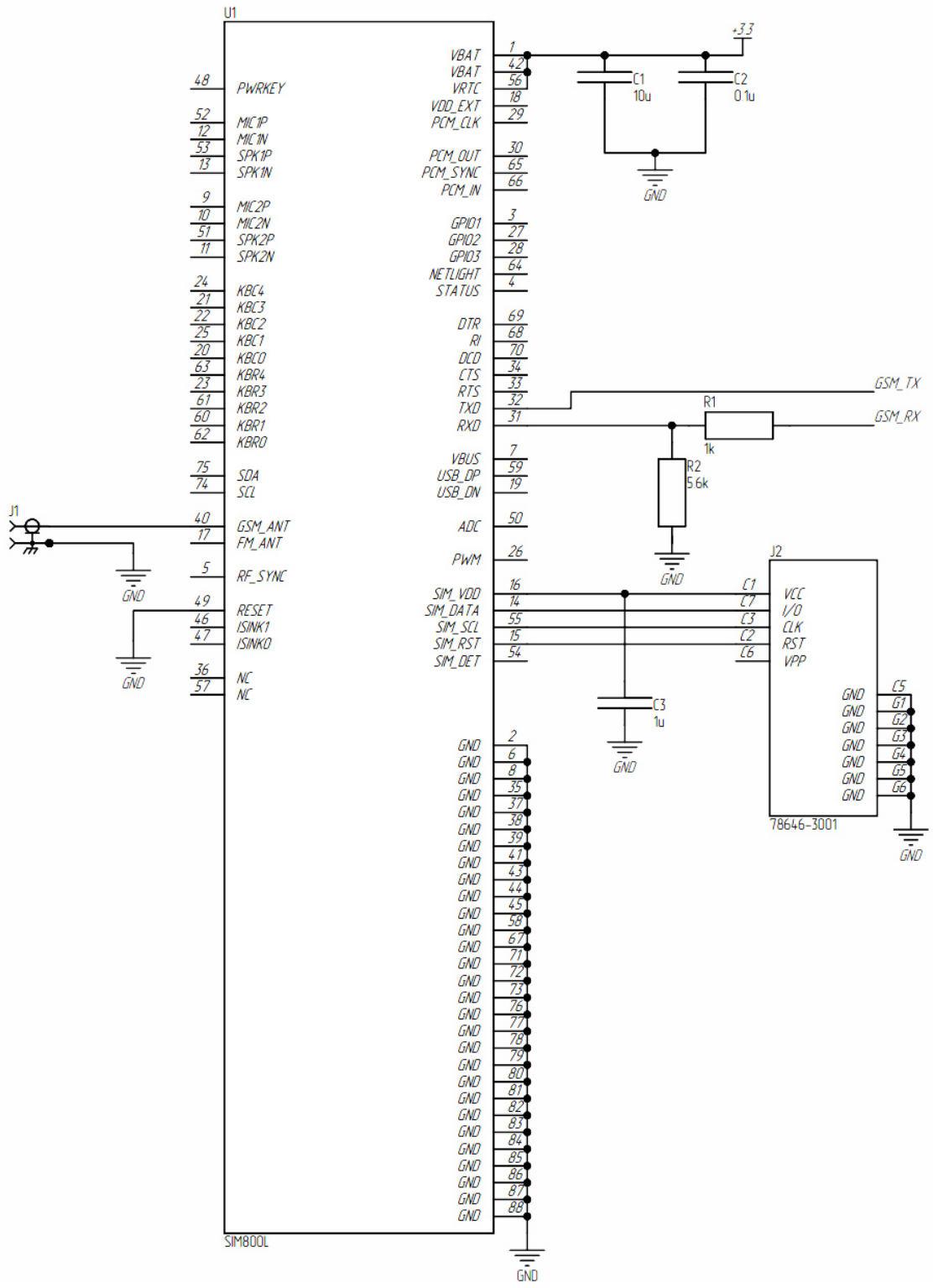


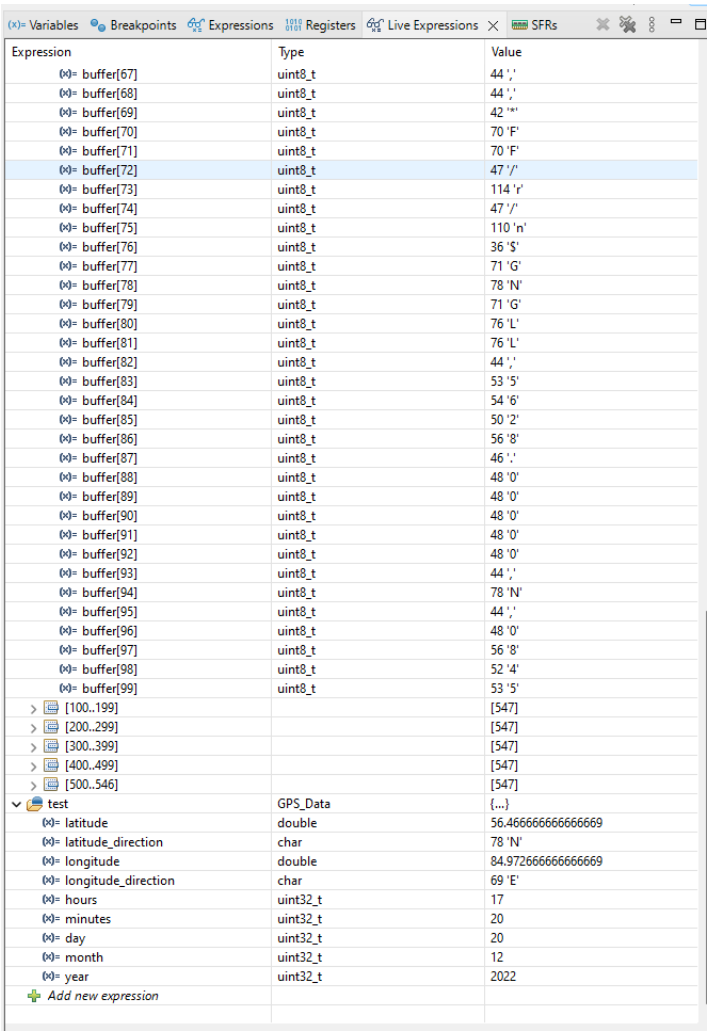
Рисунок 32 – Схема подключения ESIM800L

5 ЭКСПЕРИМЕНТАЛЬНЫЕ РЕЗУЛЬТАТЫ

5.1 Проверка работы GPS

Для проверки результатов были получены данные, которые можно наблюдать на рисунке 33 в массив «buffer» были записаны полученные данные из интерфейса UART. С помощью написанной и подключенной библиотек были распакованы и записаны в удобной форме данные, которые хранятся в структуре “test”.

В дельнейшем для проверки полученных данных, был произведен поиск по полученным координатам в картах “Google Maps”. Результат показан на рисунке 34.



(x)= Variables | Breakpoints | Expressions | Registers | Live Expressions | SFRs

Expression	Type	Value
(x)= buffer[67]	uint8_t	44 ','
(x)= buffer[68]	uint8_t	44 ','
(x)= buffer[69]	uint8_t	42 '*'
(x)= buffer[70]	uint8_t	70 'F'
(x)= buffer[71]	uint8_t	70 'F'
(x)= buffer[72]	uint8_t	47 '/'
(x)= buffer[73]	uint8_t	114 'r'
(x)= buffer[74]	uint8_t	47 '/'
(x)= buffer[75]	uint8_t	110 'n'
(x)= buffer[76]	uint8_t	36 '$'
(x)= buffer[77]	uint8_t	71 'G'
(x)= buffer[78]	uint8_t	78 'N'
(x)= buffer[79]	uint8_t	71 'G'
(x)= buffer[80]	uint8_t	76 'L'
(x)= buffer[81]	uint8_t	76 'L'
(x)= buffer[82]	uint8_t	44 ','
(x)= buffer[83]	uint8_t	53 '5'
(x)= buffer[84]	uint8_t	54 '6'
(x)= buffer[85]	uint8_t	50 '2'
(x)= buffer[86]	uint8_t	56 '8'
(x)= buffer[87]	uint8_t	46 '.'
(x)= buffer[88]	uint8_t	48 '0'
(x)= buffer[89]	uint8_t	48 '0'
(x)= buffer[90]	uint8_t	48 '0'
(x)= buffer[91]	uint8_t	48 '0'
(x)= buffer[92]	uint8_t	48 '0'
(x)= buffer[93]	uint8_t	44 ','
(x)= buffer[94]	uint8_t	78 'N'
(x)= buffer[95]	uint8_t	44 ','
(x)= buffer[96]	uint8_t	48 '0'
(x)= buffer[97]	uint8_t	56 '8'
(x)= buffer[98]	uint8_t	52 '4'
(x)= buffer[99]	uint8_t	53 '5'
[100..199]		[547]
[200..299]		[547]
[300..399]		[547]
[400..499]		[547]
[500..546]		[547]
test	GPS_Data	{...}
(x)= latitude	double	56.466666666666669
(x)= latitude_direction	char	78 'N'
(x)= longitude	double	84.972666666666669
(x)= longitude_direction	char	69 'E'
(x)= hours	uint32_t	17
(x)= minutes	uint32_t	20
(x)= day	uint32_t	20
(x)= month	uint32_t	12
(x)= year	uint32_t	2022
Add new expression		

Рисунок 33 – Полученные данные с GPS модуля

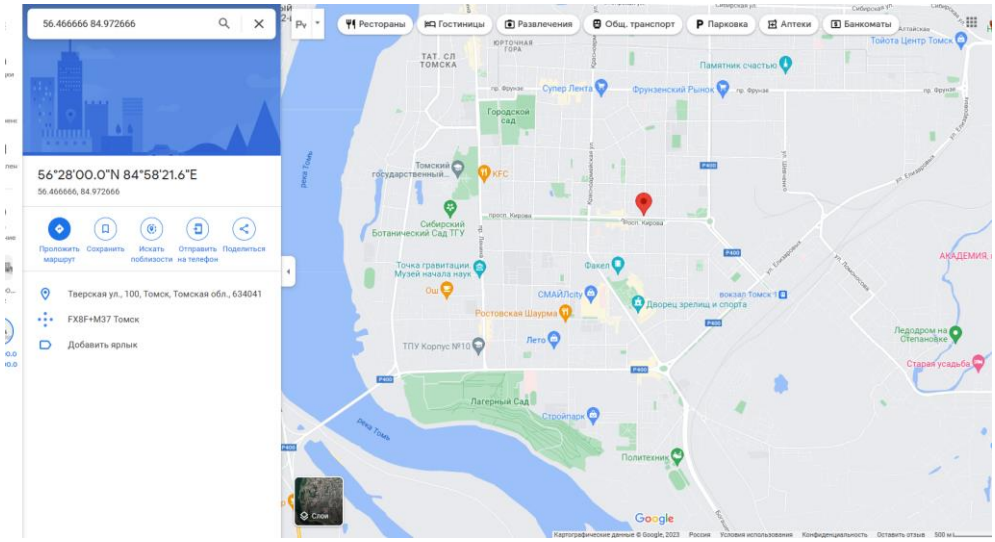


Рисунок 34 – Проверка результата

Данные текущего фактического местоположения совпали, с данными полученными из GPS-модуля

5.2 Результат использования фильтра

На рисунке 35 показаны работа фильтра в 2 ситуациях: в статическом и динамическом состоянии.

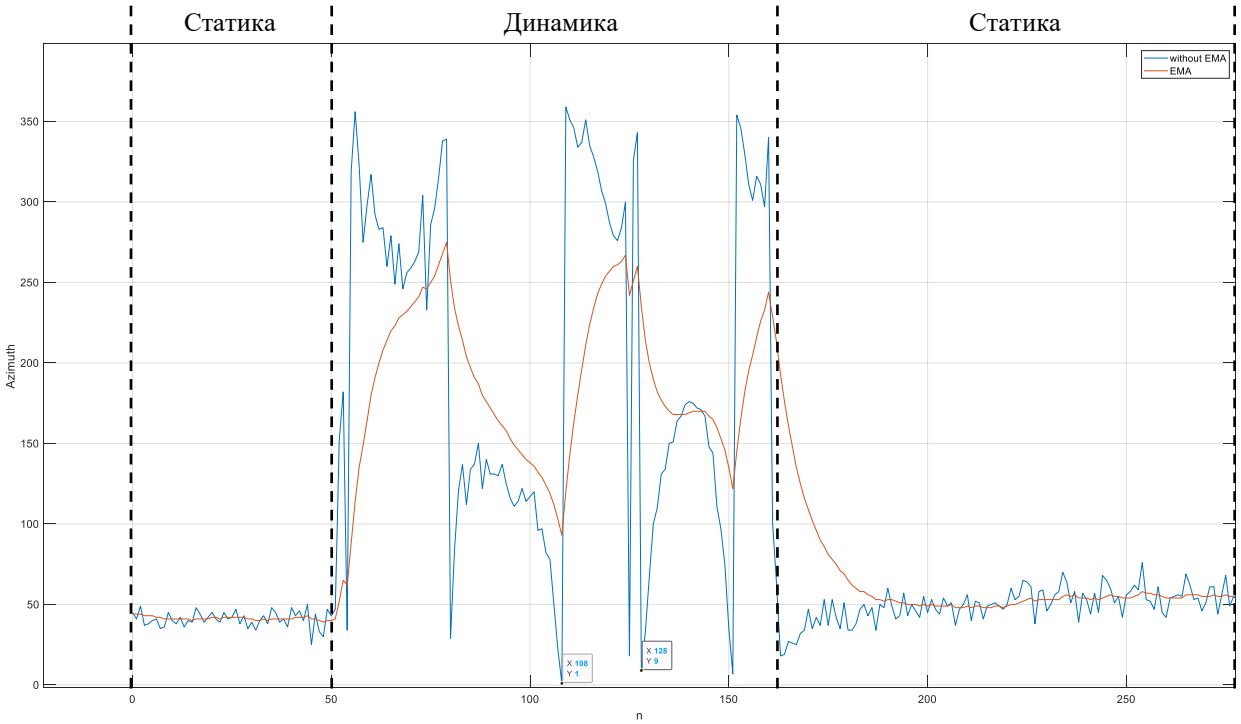


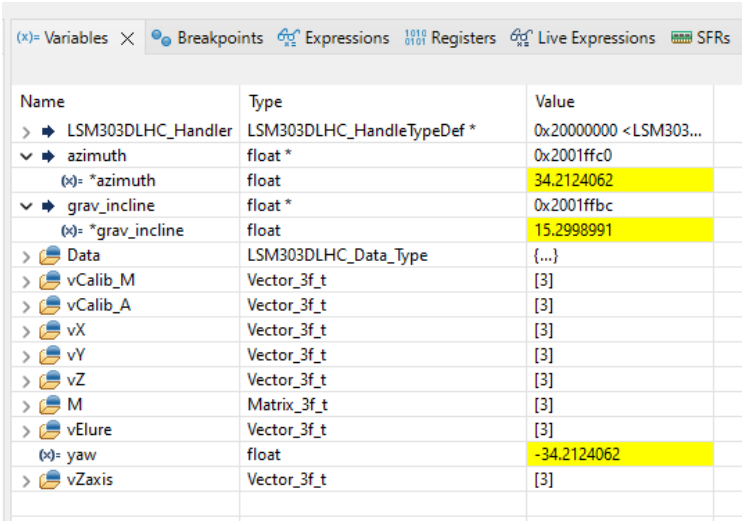
Рисунок 35 – Азимут, полученный после комплексной обработки данных с фильтром и без фильтра (синяя линия – полученный азимут без EMA; красная линия – полученный азимут с использованием EMA)

В статическом состоянии ЦФ значительное уменьшает шумы, но также наблюдается запаздывание данных. Так как итоговое устройство будет практически статичным, запаздывание не будет влиять на детектирование переворота буя.

5.3 Результат работы программы датчика LSM303DHLC

5.3.1 Измерение азимута

На рисунке 36 представлены полученный азимут в ходе выполнения программы, на рисунке 37 показания с встроенного компаса телефона. Датчик и телефон были выставлены параллельно. В таблице 3 показана серия результатов измерения.



Name	Type	Value
LSM303DLHC_Handler	LSM303DLHC_HandleTypeDef *	0x20000000 <LSM303...
azimuth	float *	0x2001ffc0
*azimuth	float	34.2124062
grav_incline	float *	0x2001ffbc
*grav_incline	float	15.2998991
Data	LSM303DLHC_Data_Type	{...}
vCalib_M	Vector_3f_t	[3]
vCalib_A	Vector_3f_t	[3]
vX	Vector_3f_t	[3]
vY	Vector_3f_t	[3]
vZ	Vector_3f_t	[3]
M	Matrix_3f_t	[3]
vElure	Vector_3f_t	[3]
yaw	float	-34.2124062
vZaxis	Vector_3f_t	[3]

Рисунок 36 – Полученные данные в программе

Рисунок 37 – Телефон показывающий азимут(слева), датчик(справа).

Таблица 3 – Результаты измерения

Номер измерения	1	2	3	4	5	6	7	8	9
Действительное значение, град	318	354	30	133	236	213	161	120	282
Полученное значение, град	319	6	35	143	226	219	182	138	298
Δ, град	1	12	5	10	-10	6	21	18	16
δ, %	0,31	3,39	16,67	7,52	4,24	2,82	13,04	15,00	5,67

По полученным результатам видно, что программа работает верно, полученные измерения совпадаю с истинными с учетом незначительной погрешности. Данная погрешность в большей степени связана с необходимостью более точной калибровки магнетометра.

5.3.2 Измерение угла наклона

Проверка измерения угла наклона относительно гравитационного поля земли представлена на рисунке 38.

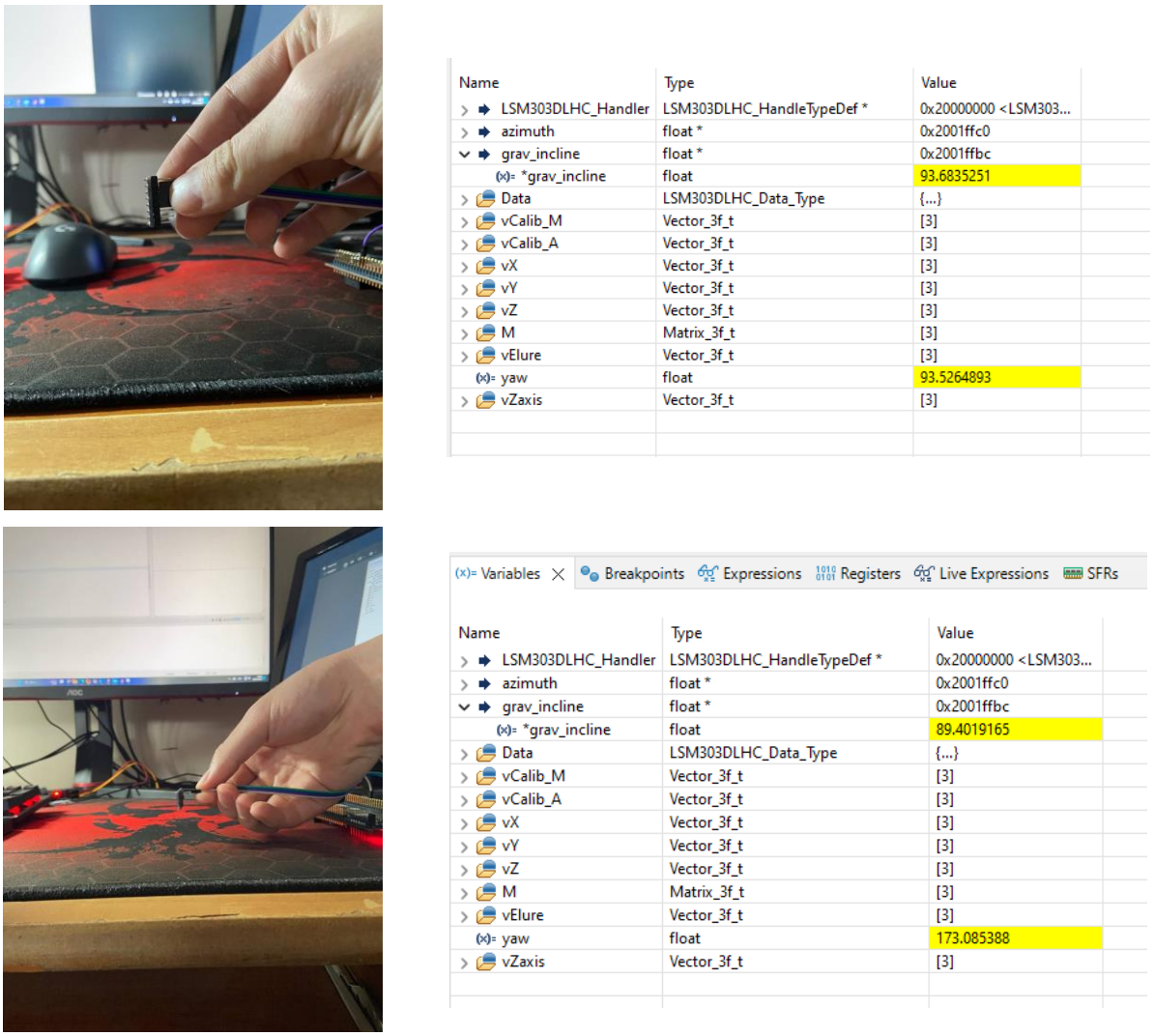


Рисунок 38 – Результат измерения угла наклона относительно гравитационного поля земли

Можно наблюдать, что вне зависимости от того, в какую сторону происходит наклон (вдоль оси x или вдоль оси y) происходит корректное считывание угла. Программа работает верно.

5.4 Результат работы всех датчиков, проверка работы всей программы, проверка отправки и приема сообщений

Для реализации проверки работы всех модулей устройства была собрана макетная версия устройства. Макет устройства представлен на рисунке 39.

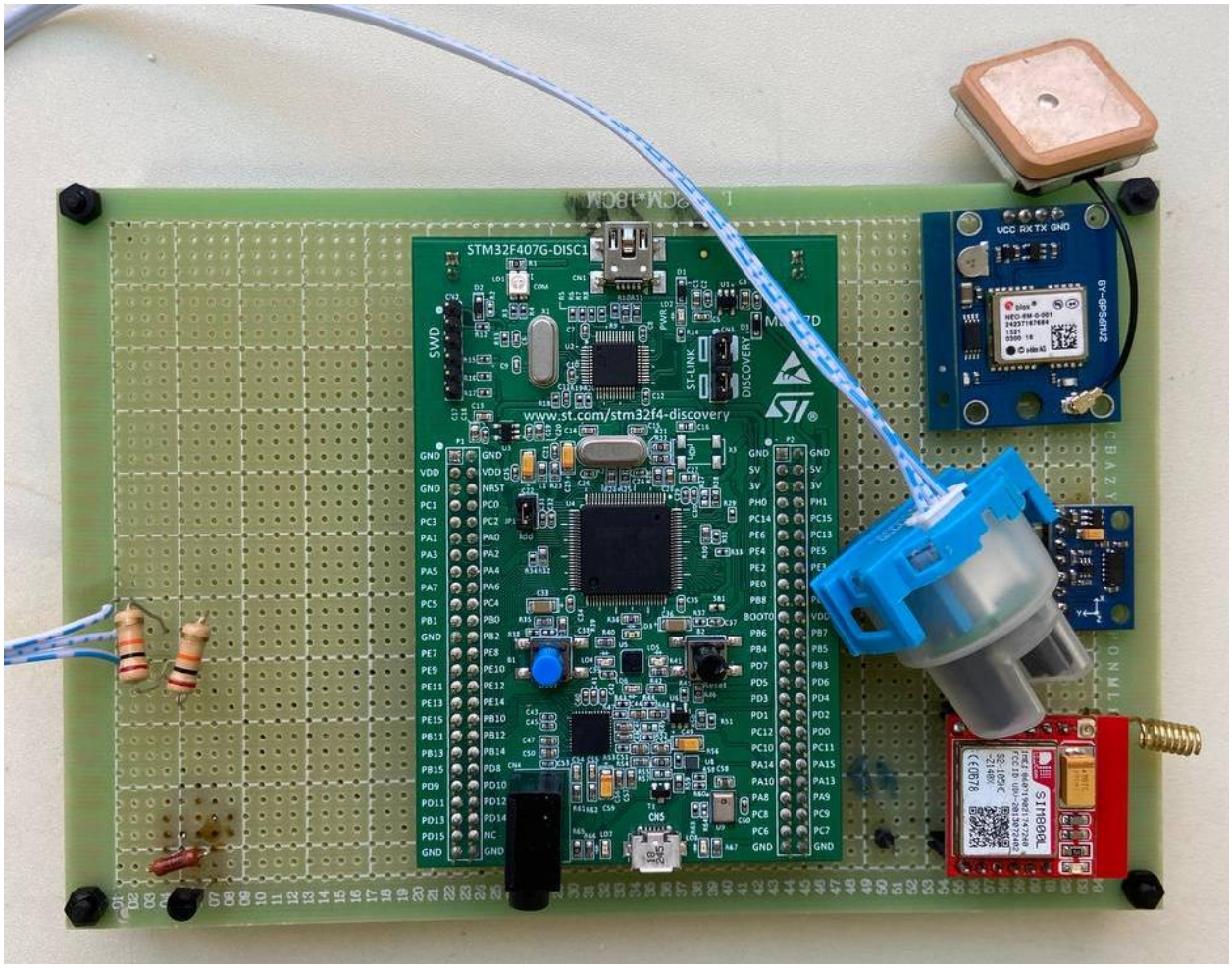


Рисунок 39 – Макет подключения датчиков

В микроконтроллер была загружена программа, представленная в приложении А. Также были подключены все датчики. Далее макет был подключен к источнику напряжения 3.3 В, и 4 В. Результат работы представлен на рисунке 40, здесь можно видеть присылаемые пакеты данных, которые соответствуют протоколу, описанному в таблице 2. Также был проведен эксперимент по отправке команд на смену характера индикации, результат также соответствует написанному в таблице 1.

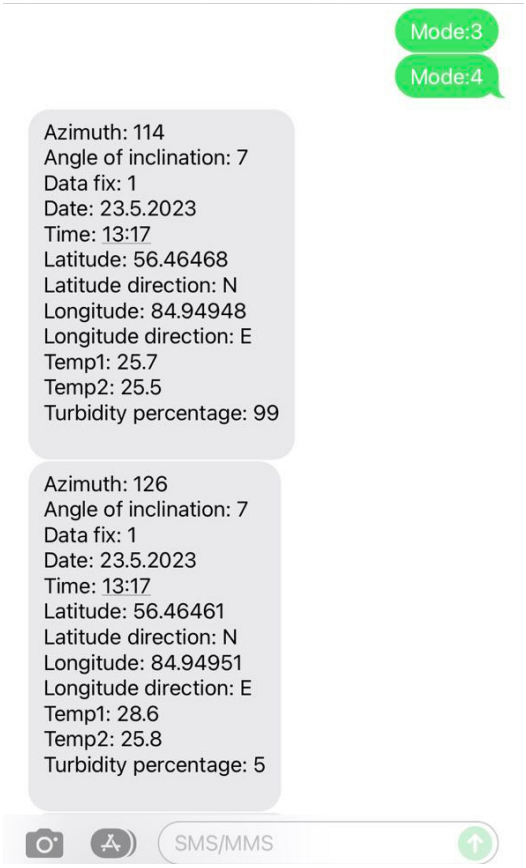


Рисунок 40 – Скриншот отправленных и принятых SMS-сообщений

ЗАДАНИЕ ДЛЯ РАЗДЕЛА

«ФИНАНСОВЫЙ МЕНЕДЖМЕНТ, РЕСУРСОЭФФЕКТИВНОСТЬ И РЕСУРСОСБЕРЕЖЕНИЕ»

Студенту:

Группа	ФИО
1А91	Мальцев Михаил Андреевич

Школа	ИШНКБ	**Отделение школы (НОЦ)**	ОМ
Уровень образования	Бакалавриат	**Направление/специальность**	11.03.04 Электроника и наноэлектроника

Исходные данные к разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»:	
1. *Стоимость ресурсов научного исследования (НИ): материально-технических, энергетических, финансовых, информационных и человеческих*	*Стоимость материальных ресурсов и специального оборудования определены в соответствии с рыночными ценами г. Томска Тарифные ставки исполнителей определены штатным расписанием специального конструкторского бюро «Смена» ТУСУР.*
2. *Нормы и нормативы расходования ресурсов*	*Норма амортизационных отчислений на специальное оборудование.*
3. *Используемая система налогообложения, ставки налогов, отчислений, дисконтирования и кредитования*	*Отчисления во внебюджетные фонды 30 %.*
Перечень вопросов, подлежащих исследованию, проектированию и разработке:	
1. *Оценка коммерческого потенциала, перспективности и альтернатив проведения НИ с позиции ресурсоэффективности и ресурсосбережения*	*Анализ конкурентоспособности, SWOT-анализ*
2. *Планирование и формирование бюджета научных исследований*	*Структура работ. Определение трудоемкости. Разработка графика проведения исследования. Расчет бюджетной стоимости НИ.*
3. *Определение ресурсной (ресурсосберегающей), финансовой, бюджетной, социальной и экономической эффективности исследования*	*Интегральный финансовый показатель. Интегральный показатель ресурсоэффективности. Интегральный показатель эффективности.*
Перечень графического материала *(с точным указанием обязательных чертежей)**:***	
1. *Оценка конкурентоспособности технических решений* 2. *Матрица SWOT* 3. *Альтернативы проведения НИ* 4. *График проведения и бюджет НИ* 5. *Оценка ресурсной, финансовой и экономической эффективности НИ*	

Дата выдачи задания для раздела по линейному графику	1.03.2023

Задание выдал консультант:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
доцент ОСГН ШБИП ТПУ	Маланина В.А.	к.э.н.		

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
1А91	Мальцев М. А.		

6 ФИНАНСОВЫЙ МЕНЕДЖМЕНТ, И РЕСУРСОЭФФЕКТИВНОСТЬ И РЕСУРСОСБЕРЕЖЕНИЕ

Данный раздел, под наименованием «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение», преследует цель провести всестороннюю оценку целесообразности и практичности научно-исследовательской работы, а также определить коммерческую привлекательность конечной продукции, с учетом ее рыночной востребованности.

Для решения данных задач имеется необходимость в следующих действиях: оценить ресурсный, экономический и коммерческий потенциал научной работы, составить календарный план работ и определить стоимость необходимых материально-технических, финансовых и кадровых ресурсов для успешного выполнения проекта.

6.1 Оценка коммерческого потенциала и перспективности проведения исследований с позиции ресурсоэффективности и ресурсосбережения

6.1.1 Потенциальные потребители результатов исследования

Для дальнейшей реализации проекта и вывода его на рынок необходимо определиться с потенциальными потребителями.

Система автоматизации навигации речного судоходства включает в себя систему передачи данных геолокации, информацию о крене, которая даёт возможность перманентно отслеживать состояние устройства. Эти данные будет учитываться при навигации на водных путях, а мониторинговая служба сможет своевременно принимать решение о необходимости ремонтных работ.

Также она позволяет отслеживать параметров воды и окружающей среды. Непрерывное сжигание различных ископаемых видов топлива, ухудшает состояние водной среды на земле, что приводит к экологическим катастрофам.

Потребителями данной системы могут стать различные институты и научные лаборатории, а также Федеральное агентство морского и речного транспорта.

6.1.2 Анализ конкурентных технических решений

В качестве конкурентов были выбраны следующие разработки:

1. система автоматизации навигации речного судоходства;
2. буй AXYS Watchkeeper;

3. плавучие знаки с автоматической идентификационной системой (АИС) от Гидрографического предприятия «Росатом».

В таблице 4 представлена оценочная карта.

Таблица 4 – Оценочная карта

Критерии оценки	Вес критерия	Баллы			Конкурентоспособность		
		Б1	Б2	Б3	К1	К2	К3
Технические критерии оценки ресурсоэффективности							
Система глобального позиционирования	0.1	7	8	8	0.7	0.8	0.8
Система сбора параметров	0.2	8	9	0	1.6	1.8	0
Инерциальная навигационная система	0.1	8	5	5	0.8	0.5	0.5
Возможность передачи данных	0.15	7	8	8	1.05	1.2	1.2
Возможность навигации судов	0.3	9	4	10	2.7	1.2	3
Экономические критерии оценки эффективности							
Конкурентоспособность технологии	0.05	8	9	5	0.4	0.45	0.25
Цена	0.1	10	5	7	1	0.5	0.7
Итог	1	57	48	43	8.25	6.45	6.45

Для сравнительной оценки рассчитывается коэффициент конкурентоспособности (формула 6.1):

$$К_{кс} = \frac{КС_ф}{КС_к} = \frac{8.25}{6.45} = 1.28, \quad (6.1)$$

где в знаменателе этой формулы берется показатель КС с максимальным значением из представленного списка конкурентов.

Если $К_{кс}$ >1, то фирма считается конкурентоспособной;

Если $К_{кс}$ <1, то положение предприятия достаточно слабое и необходимо пересмотреть свою стратегию.

Как видно из таблицы 3 и расчетов исходя из нее, $К_{кс}$ >1, система автоматизации навигации речного судоходства является достаточно конкурентоспособной. На рынке существуют два равнозначных конкурента, основным конкурентным преимуществом является возможность навигации судов в и цена.

6.1.3 SWOT-анализ

SWOT-анализ представляет собой инструмент для оценки сильных и слабых сторон проекта научных исследований. В данном разделе будут описаны преимущества и недостатки проектируемой системы, обозначены возможные угрозы со стороны внутренней и внешней среды, а также раскрыты перспективы для использования данной технологи.

В таблице 5 представлена матрица SWOT-анализа.

Таблица 5 – Матрица SWOT-анализ

	Сильные стороны: 1. Обеспечивает значительное улучшение точности и надежность навигации, что повышает безопасность плавания. 2. Повышает эффективность эксплуатации судов и, как следствие, их доходность. 3. Ускоряет процесс обслуживания пассажиров и грузовых перевозок и значительно снижает время перевозки грузов. 4. Увеличивает эффективность мониторинга водной среды.	Слабые стороны: 1. Высокая стоимость оборудования. 2. Возможны технические проблемы и сбои в работе системы, требующие высококвалифицированных специалистов для их решения.
Возможности: 1. Установка системы автоматизации навигации речного судоходства может привлечь дополнительных клиентов, заинтересованных в быстрой, эффективной и безопасной перевозке грузов. 2. Можно предлагать систему в аренду, что уменьшит начальные вложения и снизит порог входа для новых клиентов. 3. Система оснащена возможностями сбора и анализа данных, которые могут использоваться для улучшения процесса планирования и управления перевозками.	Улучшение точности и надежности навигации позволяет привлекать клиентов, заинтересованных в быстрой, безопасной перевозке грузов. Эффективный мониторинг водной среды позволит улучшить процесс планирования и управления перевозками.	Возможность предлагать систему в аренду, позволит снизить начальные вложения в проект и порог входа новых клиентов. Это нивелирует высокую стоимость и делает продукт более доступным.
Угрозы: 1. Возможные изменения законодательства, требующие установки оборудования стороннего производства. 2. Появление конкурентов, предлагающих аналогичные системы автоматизации навигации речного судоходства.	Широкий функционал системы позволяет долго сохранять конкурентоспособность.	Возможные изменения в законодательстве могут привести к вынужденному усовершенствованию и сертификации продукции, что повлечет увеличение стоимости.

6.2 Планирование научно-исследовательских работ

6.2.1 Структура работ в рамках научного исследования

Планирование комплекса предполагаемых работ осуществляется в следующем порядке:

- определение структуры работ в рамках научного исследования;
- определение участников каждой работы;
- установление продолжительности работ;
- построение графика проведения научных исследований.

Порядок составления этапов и работ, распределение исполнителей по данным видам деятельности приведен в таблицах 6 и 7.

Таблица 6 – Состав рабочей группы

№ п/п	Ф.И.О, место работы, должность	Роль в проекте	Основные обязанности
1	Торгаев С.Н., доцент ОЭИ, к.т.н.	Руководитель проекта	Координация деятельности исполнителя; проверка и анализ результатов проекта
2	Шушарина К.Е., инженер – электроник	Исполнитель проекта	Выполнение блока разработки системы сбора параметров
3	Мальцев М.А., инженер – программист	Исполнитель проекта	Выполнение блока разработки навигационной системы

Таблица 7 – Перечень этапов, работ и распределение исполнителей

Основные этапы	№ работ	Содержание работ	Должность исполнителя
Выбор направления и темы исследований	1	Выбор направления исследований	Руководитель проекта, инженер – электроник, инженер – программист
	2	Выбор темы исследования	Руководитель проекта, инженер – электроник, инженер – программист
	3	Изучение литературы по заданной тематике исследования	Инженер – электроник, инженер – программист
Разработка технического задания	4	Составление и утверждение технического задания	Руководитель проекта, инженер – электроник, инженер – программист
	5	Календарное планирование работ	Руководитель проекта
Проектирование системы	6	Разработка структурной системы устройства	Инженер – электроник, инженер – программист
	7	Выбор элементной базы	Инженер – электроник, инженер – программист
	8	Расчет электронных узлов	Инженер – электроник, инженер – программист
	9	Разработка принципиальной схемы устройства	Инженер – электроник, инженер – программист
	10	Создание печатной узла устройства	Инженер – электроник
Создание программного обеспечения	11	Написание программного кода	Инженер – электроник, инженер – программист
	12	Тестирование и отладка программного кода	Инженер – программист
Проведение тестирования готового устройства	13	Проверка всей системы на работоспособность	Руководитель проекта, инженер – электроник, инженер – программист
	14	Исправление ошибок	Инженер – электроник, инженер – программист
Оформление отчета по работе	15	Составление отчетной документации исследования	Инженер – электроник, инженер – программист
	16	Сдача проекта	Инженер – электроник, инженер – программист

6.2.2 Определение трудоемкости выполнения работ и разработка графика проведения

Для расчета стоимости исследуемого проекта необходимо определить трудоемкость осуществляемых работ, которая находится по формуле (6.2):

$$t_{ожi} = \frac{3 * t_{mini} + 2 * t_{maxi}}{5}, \quad (6.2)$$

где $t_{ожi}$ – ожидаемая трудоемкость выполнения i-ой работы чел.-дн.;

i tmin – минимально возможная трудоемкость выполнения заданной i-ой работы (оптимистическая оценка: в предположении наиболее благоприятного стечения обстоятельств), чел.-дн;

t_{maxi} – максимально возможная трудоемкость выполнения заданной i-ой работы (пессимистическая оценка: в предположении наиболее неблагоприятного стечения обстоятельств), чел.-дн.

Для установления продолжительности работы в рабочих днях используем формулу (6.3):

$$T_{pi} = \frac{t_{ожi}}{Ч_i}, \tag{6.3}$$

где T_{pi} – продолжительность одной работы, раб. дн.;

$t_{ожi}$ – ожидаемая трудоемкость выполнения одной работы, чел.-дн.;

$Ч_i$ – численность исполнителей, выполняющих одновременно одну и ту же работу на данном этапе, чел.

Следующая формула (6.4) позволяет перевести рабочие дни в календарные:

$$T_{ki.инж} = T_{pi} * k_{кал}, \tag{6.4}$$

где T_{ki} – продолжительность выполнения i-й работы в календарных днях;

T_{pi} – продолжительность выполнения i-й работы в рабочих днях;

$k_{кал}$ – календарный коэффициент.

Календарный коэффициент определяется по формуле (6.5):

$$k_{кал.инж} = \frac{T_{кал}}{T_{кал} - T_{вых} - T_{пр}} = \frac{365}{365 - 104 - 14} = 1.48 \tag{6.5}$$

где $T_{кал}$ – общее количество календарных дней в году;

$T_{вых}$ – общее количество выходных дней в году;

$T_{пр}$ – общее количество праздничных дней в году.

В таблице 8 представлены результаты трудоемкости работ руководителя и инженеров, а также длительность в рабочих и календарных днях.

Таблица 8 – Временные показатели проектирования

Название работы	Трудоёмкость работ									Длительность работ в рабочих днях, Tpi			Длительность работ в календарных днях, Tki		
	$t_{min,}$ чел-дни			t_{max}, чел-дни			$t_{ожi}$, чел-дни								
Выбор направления исследований	1	1	1	2	2	2	1,4	1,4	1,4	0,5	0,5	0,5	0,7	0,7	0,7
Выбор темы исследования	1	1	1	2	2	2	1,4	1,4	1,4	0,5	0,5	0,5	0,7	0,7	0,7
Изучение литературы по заданной тематике исследования	0	2	2	0	4	4	0	2,8	2,8	0,0	1,4	1,4	0,0	2,1	2,1
Составление и утверждение технического задания	1	1	1	2	2	2	1,4	1,4	1,4	0,5	0,5	0,5	0,7	0,7	0,7
Календарное планирование работ	1	0	0	2	0	0	1,4	0	0	1,4	0,0	0,0	2,1	0,0	0,0
Разработка структурной системы устройства	0	7	7	0	10	10	0	8,2	8,2	0,0	4,1	4,1	0,0	6,1	6,1
Выбор элементной базы	0	5	5	0	8	8	0	6,2	6,2	0,0	3,1	3,1	0,0	4,6	4,6
Расчет электронных узлов	0	35	10	0	40	12	0	37	10,8	0,0	18,5	5,4	0,0	27,4	8,0
Разработка принципиальной схемы устройства	0	4	4	0	7	7	0	5,2	5,2	0,0	2,6	2,6	0,0	3,8	3,8
Создание печатной узла устройства	0	7	0	0	10	0	0	8,2	0	0,0	8,2	0,0	0,0	12,1	0,0
Написание программного кода	0	10	40	0	13	45	0	11,2	42	0,0	5,6	21,0	0,0	8,3	31,1
Тестирование и отладка программного кода	0	0	10	0	0	13	0	0	11,2	0,0	0,0	11,2	0,0	0,0	16,6
Проверка всей системы на работоспособность	2	10	10	3	13	13	2,4	11,2	11,2	0,8	3,7	3,7	1,2	5,5	5,5
Исправление ошибок	0	8	8	0	12	12	0	9,6	9,6	0,0	4,8	4,8	0,0	7,1	7,1
Составление отчетной документации исследования	0	6	6	0	10	10	0	7,6	7,6	0,0	3,8	3,8	0,0	5,6	5,6
Сдача проекта	0	1	1	0	2	2	0	1,4	1,4	0,0	0,7	0,7	0,0	1,0	1,0
Итог													5,3	85,7	93,6

В таблице 9 представлена диаграмма Ганта, а в таблице 10 сводная таблица.

Таблица 9 – Диаграмма Ганта

№	Вид работы	Исполнители	Tk	Январь		Февраль		Март		Апрель		Май		Июнь	
				1	2	1	2	1	2	1	2	1	2	1	2
1	Выбор направления	Рук	0,7												
		Ин-эл	0,7												
		Ин-прог	0,7												
2	Выбор темы исследов	Рук	0,7												
		Ин-эл	0,7												
		Ин-прог	0,7												
3	Изучение литерату	Рук	0,0												
		Ин-эл	2,1												
		Ин-прог	2,1												
4	Составление и утвержд	Рук	0,7												
		Ин-эл	0,7												
		Ин-прог	0,7												
5	Календарное планиро	Рук	2,1												
		Ин-эл	0,0												
		Ин-прог	0,0												
6	Разработка структур	Рук	0,0												
		Ин-эл	6,1												
		Ин-прог	6,1												
7	Выбор элемент ной базы	Рук	0,0												
		Ин-эл	4,6												
		Ин-прог	4,6												
8	Расчет электронных	Рук	0,0												
		Ин-эл	27,4												
		Ин-прог	8,0												
9	Разработка принцип	Рук	0,0												
		Ин-эл	3,8												
		Ин-прог	3,8												
10	Создание печатно	Рук	0,0												
		Ин-эл	12,1												
		Ин-прог	0,0												
11	Написание програм	Рук	0,0												
		Ин-эл	8,3												
		Ин-прог	31,1												
12	Тестирование и отладка	Рук	0,0												
		Ин-эл	0,0												
		Ин-прог	16,6												
13	Проверка всей системы	Рук	1,2												
		Ин-эл	5,5												
		Ин-прог	5,5												
14	Исправление ошибок	Рук	0,0												
		Ин-эл	7,1												
		Ин-прог	7,1												
15	Составление отчетной докумен	Рук	0,0												
		Ин-эл	5,6												
		Ин-прог	5,6												
16	Сдача проекта	Рук	0,0												
		Ин-эл	1,0												
		Ин-прог	1,0												

Таблица 10 – Сводная таблица по календарным дням

	Количество дней
Общее количество календарных дней для выполнения работы	144
Общее количество календарных дней, в течение которых работал инженер-электроник	86
Общее количество календарных дней, в течение которых работал инженер-программист	94
Общее количество календарных дней, в течение которых работал руководитель	6

6.2.3 Бюджет научно-технического исследования

В этом разделе рассматриваются все виды расходов, связанные с выполнением НИР. Расчет стоимости осуществляли по следующим статьям:

– материальные затраты НТИ;

– затраты на специальное оборудование для научных (экспериментальных) работ;

– основная заработная плата исполнителей темы;

– дополнительная заработная плата исполнителей темы;

– отчисления во внебюджетные фонды (страховые отчисления);

– накладные расходы.

6.2.4 Расчет материальных затрат НТИ

Показывает количество потраченных денежных средств на материалы и оборудование, используемое для исследования. Расчет материальных затрат осуществляется по следующей формуле (6.6) ($k_T = 0.15$):

$$З_м = (1 + k_T) * \sum_{i=0}^{m} Ц_i * N_{расхi}, \quad (6.6)$$

где m – количество видов материальных ресурсов, потребляемых при выполнении научного исследования;

$N_{расхi}$ – количество материальных ресурсов i-го вида, планируемых к использованию при выполнении научного исследования (шт., кг, м, м2 и т. д.);

$Ц_i$ – цена приобретения единицы i-го вида потребляемых материальных ресурсов (руб./шт., руб./кг, руб./м, руб./м2 и т. д.);

k_T – коэффициент, учитывающий транспортно-заготовительные расходы.

В таблице 11 представлены материальные затраты на производство.

Таблица 11 – Материальные затраты

Наименование	Единица измерения	Количество		Цена за ед., руб.		Затраты на материалы, (Зм), руб.	
		Инж-эл	Инж-прог	Инж-эл	Инж-прог	Инж-эл	Инж-прог
GSM-модуль sim800l	шт.	0	2	0	520	0	1196
GPS-модуль neo6m	шт.	0	2	0	480	0	1104
температурный дачик ds18b20	шт.	2	0		0	0	0
Отладочная плата STM32F407VGT6 Discovery	шт.	1	1	4000	4000	4600	4600
Датчик LSM303DHLC	шт.	0	1	0		0	0
Датчик мутности	шт.	1	0	400	0	460	0
Резисторы	шт.	5	5	5	5	28,75	28,75
Аккумулятор Li-Fe-PO4	шт.	3	0	585	0	2018,25	0
Микросхема TPS61030PWPR	шт.	1	0	110	0	126,5	0
Дроссель	шт.	1	0	50	0	57,5	0
Конденсаторы	шт.	3	0	20	0	69	0
Итог для каждого исполнителя						7360	6928,75
Общий итог							14288,75

6.2.5 Специальное оборудование для научных (экспериментальных) работ

Для проведения научно-исследовательской работы требуются следующие виды оборудования: осциллограф АКИП-4122/1, паяльная станция BAKU BK-898D, лабораторный источник питания Element1502D+, компьютер.

Стоимость оборудования, имеющегося в научно-технической организации, учитывается в калькуляции в виде амортизационных отчислений представлено в таблице 12.

Таблица 12 – Расчет затрат на приобретение спецоборудования для научных работ

№ п/п	Наименование оборудования	Кол-во единиц оборудования	Срок эксплуатации, лет	Срок использования в НТИ, кал.дни	Цена единицы оборудования, руб.	Общая стоимость оборудования, руб.
1	Осциллограф АКИП-4122/1	1	10	10	43100	43100
2	Паяльная станция BAKU BK-898D	1	10	10	4415	4415
3	Лабораторный источник питания Element1502D+	1	10	15	3900	3900
4	Компьютер	1	5	50	50000	50000
5	Итог					101415

Расчет амортизации проводится следующим образом:

Норма амортизации рассчитывается по формуле (6.7):

$$\mathrm{H_A} = \frac{1}{n}, \tag{6.7}$$

где n – срок полезного использования в количестве лет.

Амортизация рассчитывается по формуле (6.8):

$$A = \frac{H_A * \text{И}}{12} * \frac{m}{30}, \tag{6.8}$$

где И – итоговая сумма, тыс. руб.;

m – время использования, дни.

Рассчитаем амортизацию для осциллографа АКИП-4122/1, паяльной станции BAKU BK-898D, лабораторного источника питания Element1502D+, с учётом, что срок полезного использования 10 лет, используя формулу (6.7):

$$H_A = \frac{1}{10} = 0.1$$

Рассчитаем амортизацию для компьютера, с учётом, что срок полезного использования 5 лет:

$$H_A = \frac{1}{5} = 0.2$$

Общую сумму амортизационных отчислений находим, используя формулу (6.8):

Осциллограф АКИП-4122/1:

$$\text{А} = \frac{H_A * \text{И}}{12} * \frac{m}{30} = \frac{0.1 * 43100}{12} * \frac{10}{30} = 119{,}72 \text{ руб.}$$

Паяльная станция BAKU BK-898D:

$$\text{А} = \frac{H_A * \text{И}}{12} * \frac{m}{30} = \frac{0.1 * 4415}{12} * \frac{10}{30} = 12{,}26 \text{ руб.}$$

Лабораторный источник питания Element1502D+:

$$\text{А} = \frac{H_A * \text{И}}{12} * \frac{m}{30} = \frac{0.1 * 3900}{12} * \frac{15}{30} = 16{,}25 \text{ руб.}$$

Компьютер:

$$\text{А} = \frac{H_A * \text{И}}{12} * \frac{m}{30} = \frac{0.2 * 50000}{12} * \frac{50}{30} = 1388{,}8 \text{ руб.}$$

Суммарные затраты амортизационных отчислений:

$$A = 1537.03 \text{ руб.}$$

6.2.6 Основная и дополнительная заработная плата исполнителей темы

Заработная плата исследователей состоит из основной ($З_{осн}$) и дополнительной ($З_{доп}$) и рассчитывается по следующей формуле (6.9):

$$З_{зп} = З_{осн} + З_{доп}, \quad (6.9)$$

Расчет дополнительной заработной платы ведется по следующей формуле (6.10):

$$З_{доп} = k_{доп} * З_{осн}, \quad (6.10)$$

где $k_{доп}$ – коэффициент дополнительной заработной платы (на стадии проектирования принимается равным 0,15).

В таблице 13 выполняется расчёт основной заработной платы с использование формул (6.9), (6.10).

Таблица 13 – Расчет основной заработной платы

№	Наименование этапов	Исполнители по категориям	Трудоемкость, чел.-дн.	Заработная плата, приходящаяся на один чел.-дн., тыс. руб.	Дополнительная заработная плата (Здоп)	Всего заработная плата по тарифу (окладам), тыс. руб. (Ззп)
1	1,2,4,5,13	Руководитель	6	1500	1350	10350
2	1,2,3,5,6,7,8,9,10,11, 13,14,15,16	Инж.-элек.	86	650	8385	64285
3	1,2,3,4,6,7,8,9,11,12, 13,14,15,16	Инж.-програм.	94	650	9165	70265

6.2.7 Отчисления во внебюджетные фонды (страховые отчисления)

Отчисления во внебюджетные фонды определяется по формуле (6.11):

$$З_{внеб} = k_{внеб} * (З_{осн} + З_{доп}) \quad (6.11)$$

где $k_{внеб}$ – коэффициент отчислений на уплату во внебюджетные фонды (пенсионный фонд, фонд обязательного медицинского страхования и пр.). $k_{внеб}$ на 2023 год составляет 30%

В таблице 14 выполняется расчёт отчислений во внебюджетные фонды согласно формуле (6.11).

Таблица 14 – Отчисления во внебюджетные фонды

Исполнитель	Основная заработная плата, руб.	Дополнительная заработная плата, руб.	Отчисления во внебюджетные фонды, руб.
Рук.	10350	1350	3510
Инж.-элек.	64285	8385	21801
Инж.-програм.	9165	70265	23829
Коэффициент отчислений во внебюджетные фонды			0,3
Итого:			49140

6.2.8 Накладные расходы

Накладные расходы учитывают затраты организации, не попавшие в предыдущие статьи расходов: печать и ксерокопирование материалов исследования, оплата услуг связи, электроэнергии, почтовые и телеграфные расходы, размножение материалов и т.д. Их величина определяется по следующей формуле (6.12):

$$З_{накл} = (\text{сумма статей } 1 \div 5) * k_{нр}, \quad (6.12)$$

где $k_{нр}$ – коэффициент, учитывающий накладные расходы. Величину коэффициента накладных расходов можно взять в размере 16%.

В таблице 15 произведен расчёт затрат на НИР с использование формулы (6.12).

Таблица 15 – Расчет затрат НИР

Расчет затрат НИР	Сумма, руб.			Примечание
	Рук.	Инж.-эл.	Инж.-прог.	
1. Материальные затраты НИР	0	7360	6928	Пункт 1.3.1.
2. Затраты на специальное оборудование для научных (экспериментальных) работ	0	51476	51476	Пункт 1.3.2.
3. Затраты по основной заработной плате исполнителей темы	9000	55900	61100	Пункт 1.3.3.
4. Затраты по дополнительной заработной плате исполнителей темы	1350	8385	9165	Пункт 1.3.3.
5. Отчисления во внебюджетные фонды	3510	21801	23829	Пункт 1.3.4.
6. Накладные расходы	2218	15074	16287	Пункт 1.3.5.
Бюджет НИР:	16078	159996	168785	344859

6.2.9 Определение ресурсной (ресурсосберегающей), финансовой, бюджетной, социальной и экономической эффективности исследования.

Определение эффективности происходит на основе расчета интегрального показателя эффективности научного исследования. Его нахождение связано с определением двух средневзвешенных величин: финансовой эффективности и ресурсоэффективности. Интегральный финансовый показатель разработки определяется по формуле (6.13) как:

$$I_{\text{финр}}^{\text{исп.}i} = \frac{\Phi_{pi}}{\Phi_{max}}, \tag{6.13}$$

где $I_{\text{финр}}^{\text{исп.}i}$ – интегральный финансовый показатель разработки;

Φ_{pi} – стоимость i-го варианта исполнения;

Φ_{max}–максимальная стоимость исполнения научно-исследовательского проекта (в т. ч. аналоги).

С учетом стоимости разработки аналогов AXYS Watchkeepe = 2000000 руб.; Плавучие знаки с автоматической идентификационной системой (АИС) от Гидрографического предприятия «Росатом» = 1500000 руб. Расчёт производится согласно формуле (6.13).

$$I_{\text{финр}}^{\text{исп.1}} = \frac{344859}{2000000} = 0{,}17$$

$$I_{\text{финр}}^{\text{исп.2}} = \frac{1500000}{2000000} = 0{,}75$$

$$I_{\text{финр}}^{\text{исп.3}} = \frac{2000000}{2000000} = 1$$

1. Система автоматизации навигации речного судоходства.
2. Буй AXYS Watchkeeper.
3. Плавучие знаки с автоматической идентификационной системой (АИС) от Гидрографического предприятия «Росатом».

В таблице 16 приведена сравнительная оценка характеристик.

Таблица 16 – Сравнительная оценка характеристик вариантов исполнения проекта

Критерий\Объект исследования	Весовой коэффициент параметра	К1	К2	К3
1. Система глобального позиционирования	0,15	7	8	8
2. Система сбора параметров	0,23	8	9	0
3. Инерциальная навигационная система	0,15	8	5	5
4. Возможность передачи данных	0,12	7	8	8
5. Возможность навигации судов	0,35	9	4	10
Итог	1,00	8,08	6,38	6,41

Интегральный показатель ресурсоэффективности рассчитывается по формуле (6.14):

$$I_{pi} = \sum a_i * b_i\,, \tag{6.14}$$

где Ipi – интегральный показатель ресурсоэффективности для i-го варианта разработки или аналога;

ai, bi − бальная оценка i-го варианта разработки;

n – число параметров сравнения.

Интегральный показатель эффективности вариантов исполнения разработки (Iисп.1) определяется на основании интегрального показателя ресурсоэффективности и интегрального финансового показателя по формуле (6.15):

$$I_{\text{исп}i} = \frac{I_{p-\text{исп}i}}{I_{\text{финр}}^{\text{исп}.i}} \tag{6.15}$$

Сравнение интегрального показателя эффективности вариантов исполнения разработки позволит определить сравнительную эффективность проекта и выбрать наиболее целесообразный вариант из предложенных. Сравнительная эффективность проекта (Эср) рассчитывается по формуле (6.16):

$$Э_{\text{ср}} = \frac{I_{\text{исп}i}}{I_{\text{исп}max}} \tag{6.16}$$

В таблице 17 представлена сравнительная эффективность разработки.

Таблица 17 – Сравнительная эффективность разработки

Показатели	К1	К2	К3
Интегральный финансовый показатель разработки	0,17	0,75	1,00
Интегральный показатель ресурсоэффективной разработки	8,08	6,38	6,41
Интегральный показатель эффективности	47,53	8,51	6,41
Сравнительная эффективность вариантов исполнения	1,00	0,13	0,10

Выводы по разделу «Финансовый менеджмент»

В результате выполнения первоначально изложенных задач раздела следует сделать следующие выводы.

В результате сравнения текущего проекта с другими альтернативными аналогами системы автоматизации навигации речного судоходства выявлено, что данная НИР является конкурентоспособной, так как отличается меньшей ценой, возможностью навигации судов, системой сбора параметров.

Разработан план-график выполнения этапов работ руководителя, инженера – электроника, инженера – программиста для оценки рабочего времени. Были определены: общее количество календарных дней для выполнения работы –144 дня, общее количество календарных дней, в течение которых работал инженер – электроник – 86, инженер – программист – 94 и общее количество календарных дней, в течение которых работал руководитель – 6.

Рассчитан бюджет научно-исследовательской работы, позволяющий оценить затраты на реализацию проекта. Сумма, необходимая для выполнения работы, составляют 344859 руб.

По факту оценки эффективности НИР можно сделать выводы:

– Значение интегрального финансового показателя составляет 0.17, что является показателем того, что НИР является экономически выгодной для реализации, т.к. понижает стоимость разработки по сравнению с аналогами.

– Значение интегрального показателя ресурсоэффективности научно-исследовательской разработки составляет 8.08 по сравнению со значением аналогов - 6,38 и 6,41.

Данный показатель означает, что разрабатываемый проект позволит достичь максимальных результатов с минимальными затратами ресурсов.

ЗАДАНИЕ ДЛЯ РАЗДЕЛА
«СОЦИАЛЬНАЯ ОТВЕТСТВЕННОСТЬ»

Студенту:

Группа	ФИО		
1А91	Мальцев Михаил Андреевич		
Школа	ИШНКБ	**Отделение (НОЦ)**	ОЭИ
Уровень образования	Бакалавриат	**Направление/ специальность**	11.03.04 Электроника и наноэлектроника

Тема ВКР:

Разработка блока навигации для системы автоматизации речного судоходства	
Исходные данные к разделу «Социальная ответственность»:	
Введение – Характеристика объекта исследования (вещество, материал, прибор, алгоритм, методика) и области его применения. – Описание рабочей зоны (рабочего места) при разработке проектного решения/при эксплуатации	**Объект исследования:** системы автоматизации речного судоходства и сбора показателей ОС. **Область применения:** экологический мониторинг, управление речного транспорта. **Рабочая зона:** офисное помещение **Размеры помещения:** 3*5 м **Количество и наименование оборудования рабочей зоны:** ПК, принимающее устройство. **Рабочие процессы, связанные с объектом исследования, осуществляющиеся в рабочей зоне:** анализ и обработка полученных данных.
Перечень вопросов, подлежащих исследованию, проектированию и разработке:	
1. Правовые и организационные вопросы обеспечения безопасности при разработке проектного решения: – специальные (характерные при эксплуатации объекта исследования, проектируемой рабочей зоны) правовые нормы трудового законодательства; – организационные мероприятия при компоновке рабочей зоны.	– ТК РФ от 30.12.2001 №197-ФЗ (ред. от 09.03.2021). – ГОСТ 12.2.032–78 ССБТ. «Рабочее место при выполнении работ сидя. Общие эргономические требования» – ГОСТ 21889–76 «Система «человек-машина». Кресло человека-оператора. Общие эргономические требования» – ГОСТ Р 50923–96 «Дисплеи. Рабочее место оператора. Общие эргономические требования и требования к производственной среде.
2. Производственная безопасность при разработке проектного решения – Анализ выявленных вредных и опасных производственных факторов	**Вредные факторы:** – производственные факторы, связанные с аномальными микроклиматическими параметрами воздушной среды на местонахождении работающего; – производственные факторы, связанные с отсутствием или недостатком необходимого искусственного освещения; – производственные факторы, обладающие свойствами психофизиологического воздействия на организм человека (нервно-психические перегрузки, связанные с напряженностью трудового процесса; длительность сосредоточенного наблюдения.

	– производственные факторы, связанные с электромагнитными полями, неионизирующими ткани человека (повышенным образованием электростатических зарядов) **Опасные факторы:** – производственные факторы, связанные с электрическим током, вызываемым разницей электрических потенциалов, под действие которого попадает работающий; **Требуемые средства коллективной защиты от выявленных факторов:** заземление электрооборудования, архитектурно-планировочные изменения расположения оборудований, защита расстоянием.

Задание выдал консультант:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Старший преподаватель ООД	Мезенцева Ирина Леонидовна	–		

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
1А91	Мальцев Михаил Андреевич		

7 СОЦИАЛЬНАЯ ОТВЕТСТВЕННОСТЬ

7.1 Введение

Основная задача современных научных технологий – это безопасность для людей и окружающей среды. Социальная ответственность включает в себя, ответственный подход разработчика, полное предотвращение или уменьшение влияния возможных негативных последствий применения научной технологии.

Темой ВКР является разработка блока навигации для системы автоматизации речного судоходства. Область применения: экологический мониторинг, управление речного транспорта. Для обеспечения сохранности таких объектов, как искусственных сооружений на внутренних водных путях, и безопасного плавания используют навигационное оборудование, которое представляет собой систему береговых и плавучих знаков и огней. Очень важно для обеспечения на водных путях с интенсивным движением круглосуточного судоходства использовать навигационные знаки со светосигнальным оборудованием, создающие огонь определенного цвета и характера горения.

В данном разделе будут рассмотрены вредные и опасные производственные факторы, действующие на эколога, который будет принимать данные с устройства, а также управлять ими. Будут рассмотрены такие аспекты как экологическая безопасность, безопасность в чрезвычайных ситуациях, производственная безопасность.

На рабочем месте находится следующее оборудование: ПК, принимающее устройство.

Работа предполагает использование персональной электронной вычислительной машины – персонального компьютера (ПК), расположенного в офисном помещении с размерами 3*5 м. Рабочий процесс эколога базируется на получение данных от буя с помощью GSM канала, обработка полученных данных и вследствие настройки режима работы буя. Правовые и организационные вопросы обеспечения безопасности

7.1.1 Правовые нормы трудового законодательства

В трудовом кодексе Российской Федерации от 30.12.2001 N 197-ФЗ (ред. от 09.03.2021) содержатся основные положения отношений между организацией и сотрудниками. Согласно статье 216 ТК РФ работник имеет право на труд в условиях, отвечающие требованиям охраны труда [18]: работник имеет право на безопасное рабочее место, своевременную выплату заработной платы, выходные и оплачиваемый отпуск один

раз в год. Работодатель обязан предоставить обязательное социальное и пенсионное страхование для работника.

Статья 100 ТК РФ устанавливает режим рабочего времени, включая продолжительность рабочей недели, а также оплату и нормирование труда в соответствии с разделом IV ТК РФ, который гарантирует минимальный размер оплаты труда, установление заработной платы, нормы труда и условия работы [18].

Глава 14 ТК РФ и статья 86 ТК РФ устанавливают требования и ответственность за защиту персональных данных работника от неправомерного использования или утраты. Работодатель обязан обеспечить защиту персональных данных за свой счет в соответствии с законодательством [18].

7.1.2 Эргономические требования к правильному расположению и компоновке рабочей зоны

В соответствии с ГОСТ 12.2.032–78. «Система стандартов безопасности труда». Рабочее место при выполнении работ сидя» рабочий стол может быть любой конструкции, отвечающей современным требованиям эргономики и позволяющей удобно разместить на рабочей поверхности оборудование с учетом его количества, размеров и характера выполняемой работы [19].

Рабочий стул должен обеспечивать рациональную рабочую позу при работе на ПК, позволяя регулировать высоту и углы наклона сиденья и спинки, с независимой, легкой регулировкой и надежной фиксацией. [20].

В соответствии с ГОСТ Р 50923–96 96 «Дисплеи. Рабочее место оператора. Общие эргономические требования и требования к производственной среде. Методы измерения» [21]:

1. рабочее место должно быть оборудовано столом и креслом, соответствующими требованиям удобства и безопасности;
2. клавиатура и мышь должны быть расположены на том же уровне, чтобы предотвратить перенапряжение рук и плеч;
3. экран монитора должен быть на расстоянии от 45 до 70 см от глаз работника, чтобы предотвратить усталость глаз и головную боль;
4. рабочее место должно иметь достаточное освещение;
5. шкафы и полки должны быть доступны на уровне работника, чтобы избежать напряжения;
6. компьютер должен соответствовать техническим требованиям и располагаться безопасно;

7. рабочее место должно быть комфортным, дышащим и обеспечивать защиту от шума и пыли.

7.2 Производственная безопасность

Перечень опасных и вредных факторов, характерных для данного вида работ представлен в таблице 18.

Таблица 18 – Возможные опасные и вредные производственные факторы на рабочем месте эколога [22]

Факторы (по ГОСТ 12.0.003–2015)		Нормативные документы
Вредные факторы	Производственные факторы, связанные с отсутствием или недостатком необходимого искусственного освещения	СП 52.13330.2016 Естественное и искусственное освещение Актуализированная редакция СНиП 23-05-95
	Производственные факторы, связанные с аномальными микроклиматическими параметрами воздушной среды на местонахождении работающего	ГОСТ 12.1.005–88 Система стандартов безопасности труда (ССБТ). Общие санитарногигиенические требования к воздуху рабочей зоны (с Изменением № 1)
	Производственные факторы, обладающие свойствами психофизиологического воздействия на организм человека (нервно-психические перегрузки, связанные с напряженностью трудового процесса; длительность сосредоточенного наблюдения	Р 2.2.2006-05 Гигиена труда. Руководство по гигиенической оценке факторов рабочей среды и трудового процесса. Критерии и классификация условий труда
	Производственные факторы, связанные с электромагнитными полями, неионизирующими ткани человека (повышенным образованием электростатических зарядов)	СанПиН 1.2.3685–21 «Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания»
Опасные факторы	Производственные факторы, связанные с электрическим током, вызываемым разницей электрических потенциалов, под действие которого попадает работающий	ГОСТ 12.1.038–82 ССБТ. «Электробезопасность. Предельно допустимые уровни напряжений прикосновения и токов»

7.2.1 Производственные факторы, связанные с электрическим током, вызываемым разницей электрических потенциалов, под действие которого попадает работающий

Основным оборудованием, с которым будет взаимодействовать сотрудник, является ПК. Данное оборудование подключение в общую сеть и представляет опасность поражения электрическим током.

Проходя через тело человека, электрический ток оказывает сложное воздействие, являющееся совокупностью термического, электролитического и биологического воздействий. Следствием этого может стать первичный паралич сердца, первичный паралич дыхания, электрический шок (паралич мозга), электроожоги, судороги.

Согласно ГОСТ 12.1.038–82 «ССБТ. Электробезопасность. Предельно допустимые уровни напряжений прикосновения и токов». Для переменного тока частотой 50 Гц допустимое значение напряжения прикосновения составляет 2 В, а силы тока – 0,3 мА, для

тока частотой 400 Гц соответственно – 2 В и 0,4 мА; для постоянного тока – 8 В и 1 мА (не более 10 мин в сутки) [23].

В соответствии с правилами устройств электроустановок, основными техническими средствами защиты, являются защитное заземление, автоматическое отключение питания, устройства защитного отключения, изолирующие электрозащитные средства, знаки и плакаты безопасности. Наличие таких средств защиты предусмотрено в рабочей зоне. В целях профилактики периодически проводится инструктаж работников по технике безопасности [24].

7.2.2 Производственные факторы, связанные с электромагнитными полями, неионизирующими ткани человека (повышенным образованием электростатических зарядов)

Мониторы являются источниками интенсивных электромагнитных полей. Имеющиеся внутри монитора многочисленные катушки дают электромагнитное излучение низкой частоты. Повышенный уровень напряженности электростатического поля способны возникнуть, благодаря зарядам, которые накапливают пыль на поверхностях компьютера, клавиатуре и компьютерной мыши при включенном питании.

Длительное воздействие электрического поля низкой частоты вызывает функциональные нарушения центральной нервной и сердечно-сосудистой систем человека, а также некоторые изменения в составе крови, особенно выраженные при высокой напряженности ЭП.

Допустимые нормы приведены в таблицах 19, 20, 21 ниже.

Таблица 19 – Предельно допустимые уровни постоянного магнитного поля на рабочих местах [25]

Время воздействия за рабочий день, мин	Условия воздействия			
	общие		локальные	
	ПДУ напряженности, кА/м	ПДУ магнитной индукции, мТл	ПДУ напряженности, кА/м	ПДУ магнитной индукции, мТл
≤ 10	24	30	40	50
11- 60	16	20	24	30
61- 480	8	10	12	15

Таблица 20 – ПДУ синусоидального магнитного поля частотой 50 Гц [25]

Время пребывания, ч	Допустимые уровни МП, Н[А/м] / В[мкТл] при воздействии	
	общем	локальном
≤ 1	1600 / 2000	6400 / 8000
2	800 / 1000	3200 / 4000
4	400 / 500	1600 / 2000
8	80 / 100	800 / 1000

Таблица 21 – Максимальные ПДУ напряженности и плотности потока энергии ЭМП частот ≥ 30 кГц-300 ГГц [25]

Параметр	Максимально допустимые уровни в диапазонах частот (МГц)			
	≥0,03 – 3	≥3 – 30	≥30 – 50	≥50 – 300
Е, В/м	500	300	80	80
Н, А/м	50	-	3	-

Для защиты от ЭМУ применяются методы:

– выбор рациональных режимов работы оборудования;

– ограничение места и времени нахождения работающих в ЭМП;

– защита расстоянием, т. е. удаление рабочего места от источника электромагнитных излучений;

– рациональное размещение оборудования;

– уменьшение мощности источника излучений.

7.2.3 Производственные факторы, связанные с аномальными микроклиматическими параметрами воздушной среды на местонахождении работающего

При температуре воздуха более чем 30 °С и значительном тепловом излучении от нагретых поверхностей наступает нарушение терморегуляции организма, что может привести к перегреву. В тяжелых случаях наступает тепловой удар, возможна судорожная болезнь. Местное и общее охлаждение организма является причиной таких заболеваний, как миозиты, невриты, радикулиты, простудные заболевания.

Рабочий процесс, связанный с использованием персонального компьютера относиться к категории Iа. Работы с интенсивностью энергозатрат 92 до 120 ккал/ч (до 139 Вт), производимые сидя и сопровождающиеся незначительным физическим напряжением [25].

В таблице 22 представлены допустимые величины показателей микроклимата для работ категории Iа.

Таблица 22 – Допустимые величины показателей микроклимата на рабочих местах производственных помещений [25 пункт 5]

Период года	Категории работ по уровню энергозатрат	Температура воздуха, °С	Температура поверхностей, °С	Относ. влажность воздуха, %	Скорость движения воздуха, м/с
Холодный	Iа	20-25	19-26	15-75	0,1
Теплый	Iа	21-28	20-29	15-75	0,1-0,2

Для поддержания микроклимата предусматриваются приточная и вытяжная вентиляции, нагреватели и кондиционеры. Профилактика перегревания работников

осуществляется организацией режима труда и отдыха, использования средств индивидуальной защиты.

Условия труда по вредному фактору – аномальные микроклиматические параметры на рассматриваемом объекте соответствуют допустимым величинам.

7.2.4 Производственные факторы, связанные с отсутствием или недостатком необходимого искусственного освещения

Неудовлетворительное освещение приводит к напряжению зрения, ослаблению внимания и наступлению преждевременной утомленности. Свет на рабочем месте может создать сильные тени или отблески, а также дезориентировать работающего. Основным документом по требованиям к освещенности является СП 52.13330.2016 "Естественное и искусственное освещение" Актуализированная редакция СНиП 23-05-95 [26].

В офисном помещении имеется естественное и искусственное освещение. Естественное освещение односторонние боковое. Общие освещение складывается из естественного источника света и газоразрядных ламп.

При работе с ПК рабочие столы следует размещать таким образом, чтобы видео дисплейные терминалы были ориентированы боковой стороной к световым проемам, чтобы естественный свет падал преимущественно слева. Освещенность на поверхности стола в зоне размещения рабочего документа должна быть 300–500 лк., не должно создавать бликов на поверхности экрана.

7.2.5 Производственные факторы, обладающие свойствами психофизиологического воздействия на организм человека (нервно-психические перегрузки, связанные с напряженностью трудового процесса; длительность сосредоточенного наблюдения

Психофизиологическим факторам на рабочем месте относится переутомление, стресс, умственное и эмоциональное перенапряжение, монотонность труда. Утомление и снижение работоспособности неизбежны при неправильном положении в работе. Неграмотно организованное рабочее место вызывает мышечные спазмы и усталость.

Длительная работа за экраном дисплея способствует снижению зрения, головной боли, раздражительности, потери внимания, а отсутствие регламентированных перерывов способно вызывать умственное перенапряжение.

Для предотвращения переутомления и напряжения необходимо соблюдать условия труда, когда продолжительность рабочего дня составляет 8–9 часов, а перерывы продолжительностью от 3 до 7% рабочего времени [27].

7.3 Экологическая безопасность

Вышедшая из строя компьютерная техника содержит токсичные вещества и относится к IV классу опасности. Её утилизация должна соответствовать ГОСТ Р 53692-2009 «Ресурсосбережение. Обращение с отходами. Этапы технологического цикла отходов» для минимизации влияния на окружающую среду [28].

Утилизация комплектующих частей ПК, люминесцентных ламп и макулатуры негативно влияет на литосферу. Для снижения вредного воздействия необходим переход на экологически чистые практики, включающие утилизацию и переработку отходов, использование экологически чистых материалов и более эффективных технологий.

Продукты жизнедеятельности персонала неблагоприятно влияют на гидросферу. Мероприятия по устранению вредного воздействия на гидросферу – использование экологичных продуктов для уборки, установку систем очистки сточных вод, проведение кампаний по сбору мусора и разъяснительной работы сотрудникам организации. Воздействие на атмосферу происходит в случае выделения токсических веществ при неправильной утилизации комплектующих ПЭВМ и при горении самого ПК. Мероприятия по устранению вредного воздействия на атмосферу – применение мер по эффективной утилизации отходов, чтобы не загрязнять атмосферу и ограничить количество выбрасываемых вредных веществ.

7.4 Безопасность в чрезвычайных ситуациях

Наиболее распространенный источник чрезвычайной ситуации техногенного характера – пожар. К причинам пожаров в офисных зданиях можно отнести короткие замыкания, поврежденной изоляции, использование неисправного электрооборудования, а также применение обогревательных приборов открытого типа.

К мерам по предупреждению и профилактики пожаров в офисах можно отнести:

1. регулярная проверка электропроводки и оборудования;
2. использование только исправного оборудования;
3. строгое соблюдение правил электробезопасности. Необходимо иметь автопредохранители, не допускать перегрузки и не использовать опасные устройства, такие как нагреватели открытого типа.

В офисном здании обязательно должны быть схемы эвакуации во всех предусмотренных местах, свободные эвакуационные пути, средства самостоятельного пожаротушения.

Ответственные за безопасность должны проинструктировать персонал и следить за выполнением предписаний. Все сотрудники должны пройти инструктаж по технике безопасности и следить за выполнением ее предписаний.

На основании Федерального закона от 22.07.2008 N 123-ФЗ (ред. от 30.04.2021) "Технический регламент о требованиях пожарной безопасности" класс возможного пожара в офисном помещении, как правило, определяется как класс Е, так как в офисах присутствуют горючих вещества, такие как бумага, деревянная мебель, резина и материалы электроустановок, находящиеся под напряжением (ПК).

К первичным средствам тушения пожара относятся: огнетушители (порошковый), пожарные краны (как минимум один на этаже), укомплектованные пожарные щиты или стенды и другие средства пожаротушения [29].

Вывод по разделу СО

В разделе «Социальная ответственность» рассмотрено рабочее место офисного работника эколога и выявлено, что фактические значения потенциально возможных факторов соответствуют нормативным значениям.

Категория помещения по электробезопасности, согласно ПУЭ, соответствует 1 категории (помещение без повышенной опасности). Категория объекта, оказывающего минимальное негативное воздействие на окружающую среду – IV категория. Федерального закона от 22.07.2008 N 123-ФЗ (ред. от 30.04.2021) «Технический регламент о требованиях пожарной безопасности» определено, что офисное помещение относится к В1-В4 категории по пожаро- и взрывобезопасности.

Согласно правилам по охране труда при эксплуатации электроустановок персонал должен обладать I группой допуска по электробезопасности. Присвоение группы I по электробезопасности производится путем проведения инструктажа, который должен завершаться проверкой знаний в форме устного опроса и (при необходимости) проверкой приобретенных навыков безопасных способов работы или оказания первой помощи при поражении электрическим током.

Согласно СанПиН 1.2.3685-21 "Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания" рабочий процесс, связанный с использованием персонального компьютера относиться к категории Iа.

ЗАКЛЮЧЕНИЕ

В ходе выполнения ВКР были получены следующие результаты:

1. Были рассмотрены аналоги устройства статико-динамический сборщик энергии для автономного мониторинга окружающей среды океана показала лучшие возможности по отслеживанию параметров воды, а плавучие знаки с системой АИС в навигации судов, но разрабатываемый проект включает в себя функции двух описанный устройств.
2. Была разработана структурная схема устройства, которая включает в себя блоки:
   - блок навигации;
   - блок питания;
   - блок сбора параметров;
   - блок связи;
   - блок сигнализации.
3. Была разработана схема подключения микросхем: LSM303DHLC, NEO 6M, ESIM800L, к микроконтроллеру с использованием рекомендаций производителей.
4. Была выполнена калибровка магнетометра, которая позволила значительно уменьшить искажения, что позволило достаточно точно определить азимут, угол наклона относительно гравитационного поля.
5. Была разработана программа для МК, которая включает в себя работу с микросхемами:
   - LSM303DHLC, датчик был инициализирован, а полученные данные преобразованы и получены значения углов;
   - NEO 6M, был разработан код для получения строк, отправляемых датчиком, и поиска требуемых данных;
   - ESIM800L, программа принимает и отправляет данные, полученные со всем модулей.

Экспериментальные результаты показали, что программа работает в соответствии с техническим заданием.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. A static-dynamic energy harvester for a self-powered ocean environment monitoring application // SpringerLink URL: https://link.springer.com/article/10.1007/s11431-021-1974-8 (дата обращения: 14.12.2022).
2. Гидрографическое предприятие «Росатома» повышает безопасность навигации на Северном морском пути // Гидрографическое предприятие РОСАТОМ. URL: https://rosatomport.ru/news (дата обращения: 15.12.2022).
3. GSM модули для умного дома // Хабр URL: https://habr.com/ru/companies/intems/articles/513798/ (дата обращения: 13.12.2022).
4. Леонтьев Б. К. GPS: Все, что Вы хотели знать, но боялись спросить. - 1-е изд. - Москва: Литературное агентство «Бук-Пресс», 2006. - 352 с.
5. u-blox 8 / u-blox M8 Receiver description – Manual. Ublox. 2022.
6. Что такое магнитометр и как он работает // DIFITRODE.RU URL: http://digitrode.ru/articles/3515-chto-takoe-magnitometr-i-kak-on-rabotaet.html (дата обращения: 12.02.2023).
7. Павлов Д.В., Лукин К.Г., Петров М.Н Разработка математической модели MEMS-акселерометра // Вестник Новгородского государственного университета им. Ярослава Мудрого. - 2015. - №8. - С. 91.
8. Калибровка магнитометра. Электронный компас // ROBOTCLASS. URL: https://robotclass.ru/articles/magnetometer-and-compass/ (дата обращения: 13.02.2023).
9. Exponential Moving Average (EMA) Filters // mbedded.ninja URL: https://blog.mbedded.ninja/programming/signal-processing/digital-filters/exponential-moving-average-ema-filter/ (дата обращения: 14.02.2023).
10. User Manual 1725. STMicroelectronics. 2021.
11. Datasheet LSM303DLHC. STMicroelectronics. 2013.
12. Geometric interpretation // Sailboatinstruments URL: https://sites.google.com/view/sailboatinstruments1/e-geometric-interpretation?authuser=0 (дата обращения: 26.02.2023).
13. SIM800 Series_SSL_Aplication Note_V1.00. SIMCom. 2013.
14. RM0090 Reference manual. STMicroelectronics. 2021
15. ГОСТ 26600-98 «Знаки навигационные внутренних судоходных путей».
16. NEO-6. u-blox 6 GPS Modules. Data Sheet. 2015.
17. SIM800L_Hardware_Design_V1.00. SIMCom. 2013.

18. Трудовой кодекс Российской Федерации от 30.12.2001 №197-ФЗ (ред. от 27.12.2018).
19. ГОСТ 12.2.032–78 ССБТ. «Рабочее место при выполнении работ сидя. Общие эргономические требования».
20. Согласно ГОСТ 21889–76 «Система «человек-машина». Кресло человека-оператора. Общие эргономические требования».
21. ГОСТ Р 50923–96 «Дисплеи. Рабочее место оператора. Общие эргономические требования и требования к производственной среде. Методы измерения».
22. ГОСТ 12.0.003–2015 ССБТ. «Опасные и вредные производственные факторы. Классификация».
23. ГОСТ 12.1.038–82 «ССБТ. Электробезопасность. Предельно допустимые уровни напряжений прикосновения и токов».
24. ГОСТ 12.1.019–2017 ССБТ. «Электробезопасность. Общие требования и номенклатура видов защиты».
25. СанПиН 1.2.3685–21 «Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания»
26. СП 52.13330.2016 "Естественное и искусственное освещение" Актуализированная редакция СНиП 23-05-95*
27. "Гигиенические критерии оценки и классификация условий труда по показателям вредности и опасности факторов производственной среды, тяжести и напряженности трудового процесса. Руководство р 2.2.755–99» (утв. Главным государственным санитарным врачом РФ 23.04.99).
28. ГОСТ Р 53692–2009 «Ресурсосбережение. Обращение с отходами. Этапы технологического цикла отходов».
29. ГОСТ Р 22.0.02–2016 «Безопасность в чрезвычайных ситуациях Термины и определения».

ПРИЛОЖЕНИЕ А

(обязательное)

Код микроконтроллера

1 Код main.c

```
#include "main.h"
#include "cmsis_os.h"
#include "adc.h"
#include "dma.h"
#include "i2c.h"
#include "tim.h"
#include "usart.h"
#include "gpio.h"

#include "lsm303dlhc.h"
#include "ds18b20.h"
#include "gsm.h"
#include "led.h"
#include "turbidity.h"
#include "gps.h"

LSM303DLHC_HandleTypeDef LSM303DLHC_Handle_main =
{
    .I2C_handler = &hi2c2,
    .Accel_DataRate = LSM303DLHC_ACCEL_DATA_RATE_200HZ,
    .Accel_Enable = LSM303DLHC_ACCEL_ALL_ENABLE,
    .Accel_HR = LSM303DLHC_ACCEL_HR_ON,
    .Accel_Power_Mode = LSM303DLHC_ACCEL_NORMAL_MODE,
    .Accel_Scale = LSM303DLHC_ACCEL_SCALE_2G,
    .Accel_Update_Mode = LSM303DLHC_ACCEL_MODE_CONTINUOUS,
    .Mag_DataRate = LSM303DLHC_MAG_DATA_RATE_220HZ,
    .Mag_Mode = LSM303DLHC_MAG_MODE_CONTINUOUS,
    .Mag_Range = LSM303DLHC_MAG_RANGE_1_3GAUSS,
    .Mag_Temp_Enable = LSM303DLHC_MAG_TEMP_ENABLE_OFF
};

void SystemClock_Config(void);
void MX_FREERTOS_Init(void);

xQueueHandle xQueue_LSM303DLHC_angles;
xQueueHandle xQueue_Ds18b20_temper;
xQueueHandle xQueue_GPS_data;
xQueueHandle xQueue_GSM_set_mode;
xQueueHandle xQueue_turb_percent_turbidity;

int main(void)
{

  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();


  /* Configure the system clock */
  SystemClock_Config();



  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_DMA_Init();
```

```
  MX_ADC1_Init();
  MX_TIM2_Init();
  MX_USART1_UART_Init();
  MX_USART2_UART_Init();
  MX_I2C2_Init();
  LSM303DLHC_Init(&LSM303DLHC_Handle_main);
  GPS_Init();
  Ds18b20_Init(1);
  GSM_Init();
  Led_Init();
  Turb_Init();


  /* Call init function for freertos objects (in freertos.c) */
  MX_FREERTOS_Init();

  /* Start scheduler */
  osKernelStart();
  /* We should never get here as control is now taken by the scheduler */
  /* Infinite loop */
  while (1)
  {

  }
}

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  /** Configure the main internal regulator output voltage
  */
  __HAL_RCC_PWR_CLK_ENABLE();
  __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);

  /** Initializes the RCC Oscillators according to the specified parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
  RCC_OscInitStruct.PLL.PLLM = 8;
  RCC_OscInitStruct.PLL.PLLN = 168;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
  RCC_OscInitStruct.PLL.PLLQ = 4;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }

  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
```

```
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_5) != HAL_OK)
  {
    Error_Handler();
  }
}


/**
  * @brief  Period elapsed callback in non blocking mode
  * @note   This function is called  when TIM1 interrupt took place, inside
  * HAL_TIM_IRQHandler(). It makes a direct call to HAL_IncTick() to increment
  * a global variable "uwTick" used as application time base.
  * @param  htim : TIM handle
  * @retval None
  */
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
  /* USER CODE BEGIN Callback 0 */

  /* USER CODE END Callback 0 */
  if (htim->Instance == TIM1) {
    HAL_IncTick();
  }
  /* USER CODE BEGIN Callback 1 */

  /* USER CODE END Callback 1 */
}
```

2 Библиотека температурного датчика

Файл: onewire.h

```
#ifndef ONEWIRE_H
#define ONEWIRE_H

/* C++ detection */
#ifdef __cplusplus
extern "C" {
#endif
#include "ds18b20Config.h"
#include "gpio.h"
#if (_DS18B20_USE_FREERTOS==1)
#include "cmsis_os.h"
#define     OneWireDelay(x)               osDelay(x)
#else
#define     OneWireDelay(x)               HAL_Delay(x)
#endif


typedef struct {
      GPIO_TypeDef* GPIOx;           /*!< GPIOx port to be used for I/O functions */
      uint16_t GPIO_Pin;             /*!< GPIO Pin to be used for I/O functions */
      uint8_t LastDiscrepancy;       /*!< Search private */
      uint8_t LastFamilyDiscrepancy; /*!< Search private */
      uint8_t LastDeviceFlag;        /*!< Search private */
      uint8_t ROM_NO[8];             /*!< 8-bytes address of last search device */
} OneWire_t;

/* OneWire delay */
void ONEWIRE_DELAY(uint16_t time_us);
```

```
/* Pin settings */
void ONEWIRE_LOW(OneWire_t *gp);
void ONEWIRE_HIGH(OneWire_t *gp);
void ONEWIRE_INPUT(OneWire_t *gp);
void ONEWIRE_OUTPUT(OneWire_t *gp);

/* OneWire commands */
#define ONEWIRE_CMD_RSCRATCHPAD                 0xBE
#define ONEWIRE_CMD_WSCRATCHPAD                 0x4E
#define ONEWIRE_CMD_CPYSCRATCHPAD               0x48
#define ONEWIRE_CMD_RECEEPROM                   0xB8
#define ONEWIRE_CMD_RPWRSUPPLY                  0xB4
#define ONEWIRE_CMD_SEARCHROM                   0xF0
#define ONEWIRE_CMD_READROM                     0x33
#define ONEWIRE_CMD_MATCHROM                    0x55
#define ONEWIRE_CMD_SKIPROM                     0xCC

//################################################################################################
###########
void OneWire_Init(OneWire_t* OneWireStruct, GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin);
uint8_t OneWire_Reset(OneWire_t* OneWireStruct);
uint8_t OneWire_ReadByte(OneWire_t* OneWireStruct);
void OneWire_WriteByte(OneWire_t* OneWireStruct, uint8_t byte);
void OneWire_WriteBit(OneWire_t* OneWireStruct, uint8_t bit);
uint8_t OneWire_ReadBit(OneWire_t* OneWireStruct);
uint8_t OneWire_Search(OneWire_t* OneWireStruct, uint8_t command);
void OneWire_ResetSearch(OneWire_t* OneWireStruct);
uint8_t OneWire_First(OneWire_t* OneWireStruct);
uint8_t OneWire_Next(OneWire_t* OneWireStruct);
void OneWire_GetFullROM(OneWire_t* OneWireStruct, uint8_t *firstIndex);
void OneWire_Select(OneWire_t* OneWireStruct, uint8_t* addr);
void OneWire_SelectWithPointer(OneWire_t* OneWireStruct, uint8_t* ROM);
uint8_t OneWire_CRC8(uint8_t* addr, uint8_t len);
//################################################################################################
###########

/* C++ detection */
#ifdef __cplusplus
}
#endif

#endif
```

Файл: onewire.c

```
#include "onewire.h"
#include "ds18b20Config.h"
#include "tim.h"

void ONEWIRE_DELAY(uint16_t time_us)
{
	_DS18B20_TIMER.Instance->CNT = 0;
	while(_DS18B20_TIMER.Instance->CNT <= time_us);
}
void ONEWIRE_LOW(OneWire_t *gp)
{
	gp->GPIOx->BSRR = gp->GPIO_Pin<<16;
}
void ONEWIRE_HIGH(OneWire_t *gp)
{
	gp->GPIOx->BSRR = gp->GPIO_Pin;
}
void ONEWIRE_INPUT(OneWire_t *gp)
```

```
{
	GPIO_InitTypeDef  gpinit;
	gpinit.Mode = GPIO_MODE_INPUT;
	gpinit.Pull = GPIO_NOPULL;
	gpinit.Speed = GPIO_SPEED_FREQ_HIGH;
	gpinit.Pin = gp->GPIO_Pin;
	HAL_GPIO_Init(gp->GPIOx,&gpinit);
}
void ONEWIRE_OUTPUT(OneWire_t *gp)
{
	GPIO_InitTypeDef  gpinit;
	gpinit.Mode = GPIO_MODE_OUTPUT_OD;
	gpinit.Pull = GPIO_NOPULL;
	gpinit.Speed = GPIO_SPEED_FREQ_HIGH;
	gpinit.Pin = gp->GPIO_Pin;
	HAL_GPIO_Init(gp->GPIOx,&gpinit);

}
void OneWire_Init(OneWire_t* OneWireStruct, GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin)
{
	HAL_TIM_Base_Start(&_DS18B20_TIMER);

	OneWireStruct->GPIOx = GPIOx;
	OneWireStruct->GPIO_Pin = GPIO_Pin;
	ONEWIRE_OUTPUT(OneWireStruct);
	ONEWIRE_HIGH(OneWireStruct);
	OneWireDelay(1000);
	ONEWIRE_LOW(OneWireStruct);
	OneWireDelay(1000);
	ONEWIRE_HIGH(OneWireStruct);
	OneWireDelay(2000);
}

inline uint8_t OneWire_Reset(OneWire_t* OneWireStruct)
{
	uint8_t i;

	/* Line low, and wait 480us */
	ONEWIRE_LOW(OneWireStruct);
	ONEWIRE_OUTPUT(OneWireStruct);
	ONEWIRE_DELAY(480);
	ONEWIRE_DELAY(20);
	/* Release line and wait for 70us */
	ONEWIRE_INPUT(OneWireStruct);
	ONEWIRE_DELAY(70);
	/* Check bit value */
	i = HAL_GPIO_ReadPin(OneWireStruct->GPIOx, OneWireStruct->GPIO_Pin);

	/* Delay for 410 us */
	ONEWIRE_DELAY(410);
	/* Return value of presence pulse, 0 = OK, 1 = ERROR */
	return i;
}

inline void OneWire_WriteBit(OneWire_t* OneWireStruct, uint8_t bit)
{
	if (bit)
	{
		/* Set line low */
		ONEWIRE_LOW(OneWireStruct);
		ONEWIRE_OUTPUT(OneWireStruct);
		ONEWIRE_DELAY(10);

		/* Bit high */
```

```
		ONEWIRE_INPUT(OneWireStruct);

		/* Wait for 55 us and release the line */
		ONEWIRE_DELAY(55);
		ONEWIRE_INPUT(OneWireStruct);
	}
	else
	{
		/* Set line low */
		ONEWIRE_LOW(OneWireStruct);
		ONEWIRE_OUTPUT(OneWireStruct);
		ONEWIRE_DELAY(65);

		/* Bit high */
		ONEWIRE_INPUT(OneWireStruct);

		/* Wait for 5 us and release the line */
		ONEWIRE_DELAY(5);
		ONEWIRE_INPUT(OneWireStruct);
	}

}

inline uint8_t OneWire_ReadBit(OneWire_t* OneWireStruct)
{
	uint8_t bit = 0;

	/* Line low */
	ONEWIRE_LOW(OneWireStruct);
	ONEWIRE_OUTPUT(OneWireStruct);
	ONEWIRE_DELAY(2);

	/* Release line */
	ONEWIRE_INPUT(OneWireStruct);
	ONEWIRE_DELAY(10);

	/* Read line value */
	if (HAL_GPIO_ReadPin(OneWireStruct->GPIOx, OneWireStruct->GPIO_Pin)) {
		/* Bit is HIGH */
		bit = 1;
	}

	/* Wait 50us to complete 60us period */
	ONEWIRE_DELAY(50);

	/* Return bit value */
	return bit;
}

void OneWire_WriteByte(OneWire_t* OneWireStruct, uint8_t byte) {
	uint8_t i = 8;
	/* Write 8 bits */
	while (i--) {
		/* LSB bit is first */
		OneWire_WriteBit(OneWireStruct, byte & 0x01);
		byte >>= 1;
	}
}

uint8_t OneWire_ReadByte(OneWire_t* OneWireStruct) {
	uint8_t i = 8, byte = 0;
	while (i--) {
		byte >>= 1;
		byte |= (OneWire_ReadBit(OneWireStruct) << 7);
```

```
	}

	return byte;
}

uint8_t OneWire_First(OneWire_t* OneWireStruct) {
	/* Reset search values */
	OneWire_ResetSearch(OneWireStruct);

	/* Start with searching */
	return OneWire_Search(OneWireStruct, ONEWIRE_CMD_SEARCHROM);
}

uint8_t OneWire_Next(OneWire_t* OneWireStruct) {
   /* Leave the search state alone */
   return OneWire_Search(OneWireStruct, ONEWIRE_CMD_SEARCHROM);
}

void OneWire_ResetSearch(OneWire_t* OneWireStruct) {
	/* Reset the search state */
	OneWireStruct->LastDiscrepancy = 0;
	OneWireStruct->LastDeviceFlag = 0;
	OneWireStruct->LastFamilyDiscrepancy = 0;
}

uint8_t OneWire_Search(OneWire_t* OneWireStruct, uint8_t command) {
	uint8_t id_bit_number;
	uint8_t last_zero, rom_byte_number, search_result;
	uint8_t id_bit, cmp_id_bit;
	uint8_t rom_byte_mask, search_direction;

	/* Initialize for search */
	id_bit_number = 1;
	last_zero = 0;
	rom_byte_number = 0;
	rom_byte_mask = 1;
	search_result = 0;

	// if the last call was not the last one
	if (!OneWireStruct->LastDeviceFlag)
	{
		// 1-Wire reset
		if (OneWire_Reset(OneWireStruct))
		{
			/* Reset the search */
			OneWireStruct->LastDiscrepancy = 0;
			OneWireStruct->LastDeviceFlag = 0;
			OneWireStruct->LastFamilyDiscrepancy = 0;
			return 0;
		}

		// issue the search command
		OneWire_WriteByte(OneWireStruct, command);

		// loop to do the search
		do {
			// read a bit and its complement
			id_bit = OneWire_ReadBit(OneWireStruct);
			cmp_id_bit = OneWire_ReadBit(OneWireStruct);

			// check for no devices on 1-wire
			if ((id_bit == 1) && (cmp_id_bit == 1)) {
				break;
			} else {
```

```
                        // all devices coupled have 0 or 1
                        if (id_bit != cmp_id_bit) {
                              search_direction = id_bit;  // bit write value for search
                        } else {
                              // if this discrepancy if before the Last Discrepancy
                              // on a previous next then pick the same as last time
                              if (id_bit_number < OneWireStruct->LastDiscrepancy) {
                                    search_direction = ((OneWireStruct->ROM_NO[rom_byte_number]
& rom_byte_mask) > 0);
                              } else {
                                    // if equal to last pick 1, if not then pick 0
                                    search_direction = (id_bit_number == OneWireStruct-
>LastDiscrepancy);
                              }

                              // if 0 was picked then record its position in LastZero
                              if (search_direction == 0) {
                                    last_zero = id_bit_number;

                                    // check for Last discrepancy in family
                                    if (last_zero < 9) {
                                          OneWireStruct->LastFamilyDiscrepancy = last_zero;
                                    }
                              }
                        }

                        // set or clear the bit in the ROM byte rom_byte_number
                        // with mask rom_byte_mask
                        if (search_direction == 1) {
                              OneWireStruct->ROM_NO[rom_byte_number] |= rom_byte_mask;
                        } else {
                              OneWireStruct->ROM_NO[rom_byte_number] &= ~rom_byte_mask;
                        }

                        // serial number search direction write bit
                        OneWire_WriteBit(OneWireStruct, search_direction);

                        // increment the byte counter id_bit_number
                        // and shift the mask rom_byte_mask
                        id_bit_number++;
                        rom_byte_mask <<= 1;

                        // if the mask is 0 then go to new SerialNum byte rom_byte_number and
reset mask
                        if (rom_byte_mask == 0) {
                              //docrc8(ROM_NO[rom_byte_number]);  // accumulate the CRC
                              rom_byte_number++;
                              rom_byte_mask = 1;
                        }
                  }
            } while (rom_byte_number < 8);  // loop until through all ROM bytes 0-7

            // if the search was successful then
            if (!(id_bit_number < 65)) {
                  // search successful so set LastDiscrepancy,LastDeviceFlag,search_result
                  OneWireStruct->LastDiscrepancy = last_zero;

                  // check for last device
                  if (OneWireStruct->LastDiscrepancy == 0) {
                        OneWireStruct->LastDeviceFlag = 1;
                  }

                  search_result = 1;
            }
```

```
	}

	// if no device found then reset counters so next 'search' will be like a first
	if (!search_result || !OneWireStruct->ROM_NO[0]) {
		OneWireStruct->LastDiscrepancy = 0;
		OneWireStruct->LastDeviceFlag = 0;
		OneWireStruct->LastFamilyDiscrepancy = 0;
		search_result = 0;
	}

	return search_result;
}

int OneWire_Verify(OneWire_t* OneWireStruct) {
	unsigned char rom_backup[8];
	int i,rslt,ld_backup,ldf_backup,lfd_backup;

	// keep a backup copy of the current state
	for (i = 0; i < 8; i++)
	rom_backup[i] = OneWireStruct->ROM_NO[i];
	ld_backup = OneWireStruct->LastDiscrepancy;
	ldf_backup = OneWireStruct->LastDeviceFlag;
	lfd_backup = OneWireStruct->LastFamilyDiscrepancy;

	// set search to find the same device
	OneWireStruct->LastDiscrepancy = 64;
	OneWireStruct->LastDeviceFlag = 0;

	if (OneWire_Search(OneWireStruct, ONEWIRE_CMD_SEARCHROM)) {
		// check if same device found
		rslt = 1;
		for (i = 0; i < 8; i++) {
			if (rom_backup[i] != OneWireStruct->ROM_NO[i]) {
				rslt = 1;
				break;
			}
		}
	} else {
		rslt = 0;
	}

	// restore the search state
	for (i = 0; i < 8; i++) {
		OneWireStruct->ROM_NO[i] = rom_backup[i];
	}
	OneWireStruct->LastDiscrepancy = ld_backup;
	OneWireStruct->LastDeviceFlag = ldf_backup;
	OneWireStruct->LastFamilyDiscrepancy = lfd_backup;

	// return the result of the verify
	return rslt;
}

void OneWire_TargetSetup(OneWire_t* OneWireStruct, uint8_t family_code) {
   uint8_t i;

	// set the search state to find SearchFamily type devices
	OneWireStruct->ROM_NO[0] = family_code;
	for (i = 1; i < 8; i++) {
		OneWireStruct->ROM_NO[i] = 0;
	}

	OneWireStruct->LastDiscrepancy = 64;
	OneWireStruct->LastFamilyDiscrepancy = 0;
```

```
	OneWireStruct->LastDeviceFlag = 0;
}

void OneWire_FamilySkipSetup(OneWire_t* OneWireStruct) {
	// set the Last discrepancy to last family discrepancy
	OneWireStruct->LastDiscrepancy = OneWireStruct->LastFamilyDiscrepancy;
	OneWireStruct->LastFamilyDiscrepancy = 0;

	// check for end of list
	if (OneWireStruct->LastDiscrepancy == 0) {
		OneWireStruct->LastDeviceFlag = 1;
	}
}

uint8_t OneWire_GetROM(OneWire_t* OneWireStruct, uint8_t index) {
	return OneWireStruct->ROM_NO[index];
}

void OneWire_Select(OneWire_t* OneWireStruct, uint8_t* addr) {
	uint8_t i;
	OneWire_WriteByte(OneWireStruct, ONEWIRE_CMD_MATCHROM);

	for (i = 0; i < 8; i++) {
		OneWire_WriteByte(OneWireStruct, *(addr + i));
	}
}

void OneWire_SelectWithPointer(OneWire_t* OneWireStruct, uint8_t *ROM) {
	uint8_t i;
	OneWire_WriteByte(OneWireStruct, ONEWIRE_CMD_MATCHROM);

	for (i = 0; i < 8; i++) {
		OneWire_WriteByte(OneWireStruct, *(ROM + i));
	}
}

void OneWire_GetFullROM(OneWire_t* OneWireStruct, uint8_t *firstIndex) {
	uint8_t i;
	for (i = 0; i < 8; i++) {
		*(firstIndex + i) = OneWireStruct->ROM_NO[i];
	}
}

uint8_t OneWire_CRC8(uint8_t *addr, uint8_t len) {
	uint8_t crc = 0, inbyte, i, mix;

	while (len--) {
		inbyte = *addr++;
		for (i = 8; i; i--) {
			mix = (crc ^ inbyte) & 0x01;
			crc >>= 1;
			if (mix) {
				crc ^= 0x8C;
			}
			inbyte >>= 1;
		}
	}

	/* Return calculated CRC */
	return crc;
}
```

Файл: ds18b20Config.h

```
#ifndef     _DS18B20CONFIG_H
#define     _DS18B20CONFIG_H


//    Init timer on cube    1us per tick                    example 72 MHz cpu >>>
Prescaler=(72-1)      counter period=Max
//#########################################################################################
#define     _DS18B20_USE_FREERTOS                                   1
#define _DS18B20_MAX_SENSORS                                2
#define     _DS18B20_GPIO
      SENSOR_GPIO_Port
#define     _DS18B20_PIN
      SENSOR_Pin

#define     _DS18B20_CONVERT_TIMEOUT_MS                             5000
#if (_DS18B20_USE_FREERTOS==1)
#define     _DS18B20_UPDATE_INTERVAL_MS                             10000
      //  (((     if==0  >> Ds18b20_ManualConvert()  )))     ((( if>0  >>>> Auto refresh )))
#endif


#define     _DS18B20_TIMER                                                  htim2
//#########################################################################################

#endif
```

Файл: ds18b20.h

```
#ifndef     _DS18B20_H
#define     _DS18B20_H
//    2018/09/08


#include "onewire.h"
#include "ds18b20Config.h"
#include <stdbool.h>

#if (_DS18B20_USE_FREERTOS==1)
#include "cmsis_os.h"
#define     Ds18b20Delay(x)               osDelay(x)
#else
#define     Ds18b20Delay(x)               HAL_Delay(x)
#endif

//#########################################################################################
typedef struct
{
      uint8_t     Address[8];
      float             Temperature;
      bool        DataIsValid;

}Ds18b20Sensor_t;
typedef struct
{
      float             temp_sensor1;
      float             temp_sensor2;
}Ds18b20SensorQu_t;
//#########################################################################################

extern Ds18b20Sensor_t  ds18b20[_DS18B20_MAX_SENSORS];
extern xQueueHandle xQueue_Ds18b20_temper;
//#########################################################################################
```

```
/* Every onewire chip has different ROM code, but all the same chips has same family code */
/* in case of DS18B20 this is 0x28 and this is first byte of ROM address */
#define DS18B20_FAMILY_CODE                             0x28
#define DS18B20_CMD_ALARMSEARCH                         0xEC

/* DS18B20 read temperature command */
#define DS18B20_CMD_CONVERTTEMP                         0x44  /* Convert temperature */
#define DS18B20_DECIMAL_STEPS_12BIT       0.0625
#define DS18B20_DECIMAL_STEPS_11BIT       0.125
#define DS18B20_DECIMAL_STEPS_10BIT       0.25
#define DS18B20_DECIMAL_STEPS_9BIT        0.5

/* Bits locations for resolution */
#define DS18B20_RESOLUTION_R1                           6
#define DS18B20_RESOLUTION_R0                           5

/* CRC enabled */
#ifdef DS18B20_USE_CRC
#define DS18B20_DATA_LEN                                        9
#else
#define DS18B20_DATA_LEN                                        2
#endif

//!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
typedef enum {
      DS18B20_Resolution_9bits = 9,   /*!< DS18B20 9 bits resolution */
      DS18B20_Resolution_10bits = 10, /*!< DS18B20 10 bits resolution */
      DS18B20_Resolution_11bits = 11, /*!< DS18B20 11 bits resolution */
      DS18B20_Resolution_12bits = 12  /*!< DS18B20 12 bits resolution */
} DS18B20_Resolution_t;

//!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
#if (_DS18B20_USE_FREERTOS==1)
void              Ds18b20_Init(osPriority Priority);
#else
bool              Ds18b20_Init(void);
#endif
bool              Ds18b20_ManualConvert(void);
//!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
uint8_t     DS18B20_Start(OneWire_t* OneWireStruct, uint8_t* ROM);
void              DS18B20_StartAll(OneWire_t* OneWireStruct);
bool              DS18B20_Read(OneWire_t* OneWireStruct, uint8_t* ROM, float* destination);
uint8_t     DS18B20_GetResolution(OneWire_t* OneWireStruct, uint8_t* ROM);
uint8_t     DS18B20_SetResolution(OneWire_t* OneWireStruct, uint8_t* ROM, DS18B20_Resolution_t
resolution);
uint8_t     DS18B20_Is(uint8_t* ROM);
uint8_t     DS18B20_SetAlarmHighTemperature(OneWire_t* OneWireStruct, uint8_t* ROM, int8_t
temp);
uint8_t     DS18B20_SetAlarmLowTemperature(OneWire_t* OneWireStruct, uint8_t* ROM, int8_t
temp);
uint8_t     DS18B20_DisableAlarmTemperature(OneWire_t* OneWireStruct, uint8_t* ROM);
uint8_t     DS18B20_AlarmSearch(OneWire_t* OneWireStruct);
uint8_t     DS18B20_AllDone(OneWire_t* OneWireStruct);
//!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!


#endif
```

Файл: ds18b20.c

```
#include "ds18b20.h"
```

```
/*#########################################################################################*/
Ds18b20Sensor_t   ds18b20[_DS18B20_MAX_SENSORS];

OneWire_t OneWire;
uint8_t       OneWireDevices;
uint8_t     TempSensorCount=0;
uint8_t           Ds18b20StartConvert=0;
uint16_t    Ds18b20Timeout=0;
#if (_DS18B20_USE_FREERTOS==1)
osThreadId  Ds18b20Handle;
void Task_Ds18b20(void const * argument);
#endif

/*#########################################################################################*
/
#if (_DS18B20_USE_FREERTOS==1)
void  Ds18b20_Init(osPriority Priority)
{
      osThreadDef(myTask_Ds18b20, Task_Ds18b20, Priority, 0, 128);
  Ds18b20Handle = osThreadCreate(osThread(myTask_Ds18b20), NULL);
  xQueue_Ds18b20_temper = xQueueCreate(1,sizeof(Ds18b20SensorQu_t));
}
#else
bool  Ds18b20_Init(void)
{
      uint8_t     Ds18b20TryToFind=5;
      do
      {
            OneWire_Init(&OneWire,_DS18B20_GPIO ,_DS18B20_PIN);
            TempSensorCount = 0;
            while(HAL_GetTick() < 3000)
                  Ds18b20Delay(100);
            OneWireDevices = OneWire_First(&OneWire);
            while (OneWireDevices)
            {
                  Ds18b20Delay(100);
                  TempSensorCount++;
                  OneWire_GetFullROM(&OneWire, ds18b20[TempSensorCount-1].Address);
                  OneWireDevices = OneWire_Next(&OneWire);
            }
            if(TempSensorCount>0)
                  break;
            Ds18b20TryToFind--;
      }while(Ds18b20TryToFind>0);
      if(Ds18b20TryToFind==0)
            return false;
      for (uint8_t i = 0; i < TempSensorCount; i++)
      {
            Ds18b20Delay(50);
    DS18B20_SetResolution(&OneWire, ds18b20[i].Address, DS18B20_Resolution_12bits);
            Ds18b20Delay(50);
    DS18B20_DisableAlarmTemperature(&OneWire,  ds18b20[i].Address);
  }
      return true;
}
#endif
/*#########################################################################################*
/
bool  Ds18b20_ManualConvert(void)
{
      #if (_DS18B20_USE_FREERTOS==1)
      Ds18b20StartConvert=1;
      while(Ds18b20StartConvert==1)
            Ds18b20Delay(10);
```

```
	if(Ds18b20Timeout==0)
		return false;
	else
		return true;
	#else
	Ds18b20Timeout=_DS18B20_CONVERT_TIMEOUT_MS/10;
	DS18B20_StartAll(&OneWire);
	Ds18b20Delay(100);
	while (!DS18B20_AllDone(&OneWire))
	{
		Ds18b20Delay(10);
		Ds18b20Timeout-=1;
		if(Ds18b20Timeout==0)
			break;
	}
	if(Ds18b20Timeout>0)
	{
		for (uint8_t i = 0; i < TempSensorCount; i++)
		{
			Ds18b20Delay(100);
			ds18b20[i].DataIsValid = DS18B20_Read(&OneWire, ds18b20[i].Address,
&ds18b20[i].Temperature);
		}
	}
	else
	{
		for (uint8_t i = 0; i < TempSensorCount; i++)
			ds18b20[i].DataIsValid = false;
	}
	if(Ds18b20Timeout==0)
		return false;
	else
		return true;
	#endif
}
/*##########################################################################################*
/
#if (_DS18B20_USE_FREERTOS==1)
void Task_Ds18b20(void const * argument)
{
	uint8_t	Ds18b20TryToFind=5;
	Ds18b20SensorQu_t temperature = {0,};
	portTickType xLastWakeTime = xTaskGetTickCount();
	do
	{
		OneWire_Init(&OneWire,_DS18B20_GPIO ,_DS18B20_PIN);
		TempSensorCount = 0;
		while(HAL_GetTick() < 3000)
			Ds18b20Delay(100);
		OneWireDevices = OneWire_First(&OneWire);
		while (OneWireDevices)
		{
			Ds18b20Delay(100);
			TempSensorCount++;
			OneWire_GetFullROM(&OneWire, ds18b20[TempSensorCount-1].Address);
			OneWireDevices = OneWire_Next(&OneWire);
		}
		if(TempSensorCount>0)
			break;
		Ds18b20TryToFind--;
	}while(Ds18b20TryToFind>0);
	if(Ds18b20TryToFind==0)
		vTaskDelete(Ds18b20Handle);
	for (uint8_t i = 0; i < TempSensorCount; i++)
```

```
      {
            Ds18b20Delay(50);
    DS18B20_SetResolution(&OneWire, ds18b20[i].Address, DS18B20_Resolution_9bits);
            Ds18b20Delay(50);
    DS18B20_DisableAlarmTemperature(&OneWire,  ds18b20[i].Address);
  }
      for(;;)
      {
            xQueueReceive(xQueue_Ds18b20_temper, &temperature, 0);
            while(_DS18B20_UPDATE_INTERVAL_MS==0)
            {
                  if(Ds18b20StartConvert==1)
                        break;
                  Ds18b20Delay(10);
            }
            Ds18b20Timeout=_DS18B20_CONVERT_TIMEOUT_MS/10;
            DS18B20_StartAll(&OneWire);
            osDelay(100);
    while (!DS18B20_AllDone(&OneWire))
            {
                  osDelay(10);
                  Ds18b20Timeout-=1;
                  if(Ds18b20Timeout==0)
                        break;
            }
            if(Ds18b20Timeout>0)
            {
                  for (uint8_t i = 0; i < TempSensorCount; i++)
                  {
                        Ds18b20Delay(1000);
                        ds18b20[i].DataIsValid = DS18B20_Read(&OneWire, ds18b20[i].Address,
&ds18b20[i].Temperature);
                  }
            }
            else
            {
                  for (uint8_t i = 0; i < TempSensorCount; i++)
                        ds18b20[i].DataIsValid = false;
            }
            Ds18b20StartConvert=0;

            if(ds18b20[0].DataIsValid)
                  temperature.temp_sensor1 = ds18b20[0].Temperature;
            else
                  temperature.temp_sensor1 = 99.f;

            if(ds18b20[1].DataIsValid)
                  temperature.temp_sensor2 = ds18b20[1].Temperature;
            else
                  temperature.temp_sensor2 = 99.f;

            xQueueSendToBack(xQueue_Ds18b20_temper, &temperature, 0);

            vTaskDelayUntil(&xLastWakeTime, _DS18B20_UPDATE_INTERVAL_MS/portTICK_RATE_MS);
      }
}
#endif
/*#########################################################################################*
/
uint8_t DS18B20_Start(OneWire_t* OneWire, uint8_t *ROM)
{
      /* Check if device is DS18B20 */
      if (!DS18B20_Is(ROM)) {
            return 0;
```

```
	}

	/* Reset line */
	OneWire_Reset(OneWire);
	/* Select ROM number */
	OneWire_SelectWithPointer(OneWire, ROM);
	/* Start temperature conversion */
	OneWire_WriteByte(OneWire, DS18B20_CMD_CONVERTTEMP);

	return 1;
}

void DS18B20_StartAll(OneWire_t* OneWire)
{
	/* Reset pulse */
	OneWire_Reset(OneWire);
	/* Skip rom */
	OneWire_WriteByte(OneWire, ONEWIRE_CMD_SKIPROM);
	/* Start conversion on all connected devices */
	OneWire_WriteByte(OneWire, DS18B20_CMD_CONVERTTEMP);
}

bool DS18B20_Read(OneWire_t* OneWire, uint8_t *ROM, float *destination)
{
	uint16_t temperature;
	uint8_t resolution;
	int8_t digit, minus = 0;
	float decimal;
	uint8_t i = 0;
	uint8_t data[9];
	uint8_t crc;

	/* Check if device is DS18B20 */
	if (!DS18B20_Is(ROM)) {
		return false;
	}

	/* Check if line is released, if it is, then conversion is complete */
	if (!OneWire_ReadBit(OneWire))
	{
		/* Conversion is not finished yet */
		return false;
	}

	/* Reset line */
	OneWire_Reset(OneWire);
	/* Select ROM number */
	OneWire_SelectWithPointer(OneWire, ROM);
	/* Read scratchpad command by onewire protocol */
	OneWire_WriteByte(OneWire, ONEWIRE_CMD_RSCRATCHPAD);

	/* Get data */
	for (i = 0; i < 9; i++)
	{
		/* Read byte by byte */
		data[i] = OneWire_ReadByte(OneWire);
	}

	/* Calculate CRC */
	crc = OneWire_CRC8(data, 8);

	/* Check if CRC is ok */
	if (crc != data[8])
		/* CRC invalid */
```

```
		return 0;


	/* First two bytes of scratchpad are temperature values */
	temperature = data[0] | (data[1] << 8);

	/* Reset line */
	OneWire_Reset(OneWire);

	/* Check if temperature is negative */
	if (temperature & 0x8000)
	{
		/* Two's complement, temperature is negative */
		temperature = ~temperature + 1;
		minus = 1;
	}


	/* Get sensor resolution */
	resolution = ((data[4] & 0x60) >> 5) + 9;


	/* Store temperature integer digits and decimal digits */
	digit = temperature >> 4;
	digit |= ((temperature >> 8) & 0x7) << 4;

	/* Store decimal digits */
	switch (resolution)
	{
		case 9:
			decimal = (temperature >> 3) & 0x01;
			decimal *= (float)DS18B20_DECIMAL_STEPS_9BIT;
		break;
		case 10:
			decimal = (temperature >> 2) & 0x03;
			decimal *= (float)DS18B20_DECIMAL_STEPS_10BIT;
		 break;
		case 11:
			decimal = (temperature >> 1) & 0x07;
			decimal *= (float)DS18B20_DECIMAL_STEPS_11BIT;
		break;
		case 12:
			decimal = temperature & 0x0F;
			decimal *= (float)DS18B20_DECIMAL_STEPS_12BIT;
		 break;
		default:
			decimal = 0xFF;
			digit = 0;
	}

	/* Check for negative part */
	decimal = digit + decimal;
	if (minus)
		decimal = 0 - decimal;


	/* Set to pointer */
	*destination = decimal;

	/* Return 1, temperature valid */
	return true;
}

uint8_t DS18B20_GetResolution(OneWire_t* OneWire, uint8_t *ROM)
```

```
{
	uint8_t conf;

	if (!DS18B20_Is(ROM))
		return 0;

	/* Reset line */
	OneWire_Reset(OneWire);
	/* Select ROM number */
	OneWire_SelectWithPointer(OneWire, ROM);
	/* Read scratchpad command by onewire protocol */
	OneWire_WriteByte(OneWire, ONEWIRE_CMD_RSCRATCHPAD);

	/* Ignore first 4 bytes */
	OneWire_ReadByte(OneWire);
	OneWire_ReadByte(OneWire);
	OneWire_ReadByte(OneWire);
	OneWire_ReadByte(OneWire);

	/* 5th byte of scratchpad is configuration register */
	conf = OneWire_ReadByte(OneWire);

	/* Return 9 - 12 value according to number of bits */
	return ((conf & 0x60) >> 5) + 9;
}

uint8_t DS18B20_SetResolution(OneWire_t* OneWire, uint8_t *ROM, DS18B20_Resolution_t
resolution)
{
	uint8_t th, tl, conf;
	if (!DS18B20_Is(ROM))
		return 0;


	/* Reset line */
	OneWire_Reset(OneWire);
	/* Select ROM number */
	OneWire_SelectWithPointer(OneWire, ROM);
	/* Read scratchpad command by onewire protocol */
	OneWire_WriteByte(OneWire, ONEWIRE_CMD_RSCRATCHPAD);

	/* Ignore first 2 bytes */
	OneWire_ReadByte(OneWire);
	OneWire_ReadByte(OneWire);

	th = OneWire_ReadByte(OneWire);
	tl = OneWire_ReadByte(OneWire);
	conf = OneWire_ReadByte(OneWire);

	if (resolution == DS18B20_Resolution_9bits)
	{
		conf &= ~(1 << DS18B20_RESOLUTION_R1);
		conf &= ~(1 << DS18B20_RESOLUTION_R0);
	}
	else if (resolution == DS18B20_Resolution_10bits)
	{
		conf &= ~(1 << DS18B20_RESOLUTION_R1);
		conf |= 1 << DS18B20_RESOLUTION_R0;
	}
	else if (resolution == DS18B20_Resolution_11bits)
	{
		conf |= 1 << DS18B20_RESOLUTION_R1;
		conf &= ~(1 << DS18B20_RESOLUTION_R0);
	}
```

```
	else if (resolution == DS18B20_Resolution_12bits)
	{
		conf |= 1 << DS18B20_RESOLUTION_R1;
		conf |= 1 << DS18B20_RESOLUTION_R0;
	}

	/* Reset line */
	OneWire_Reset(OneWire);
	/* Select ROM number */
	OneWire_SelectWithPointer(OneWire, ROM);
	/* Write scratchpad command by onewire protocol, only th, tl and conf register can be
written */
	OneWire_WriteByte(OneWire, ONEWIRE_CMD_WSCRATCHPAD);

	/* Write bytes */
	OneWire_WriteByte(OneWire, th);
	OneWire_WriteByte(OneWire, tl);
	OneWire_WriteByte(OneWire, conf);

	/* Reset line */
	OneWire_Reset(OneWire);
	/* Select ROM number */
	OneWire_SelectWithPointer(OneWire, ROM);
	/* Copy scratchpad to EEPROM of DS18B20 */
	OneWire_WriteByte(OneWire, ONEWIRE_CMD_CPYSCRATCHPAD);

	return 1;
}

uint8_t DS18B20_Is(uint8_t *ROM)
{
	/* Checks if first byte is equal to DS18B20's family code */
	if (*ROM == DS18B20_FAMILY_CODE)
		return 1;

	return 0;
}

uint8_t DS18B20_SetAlarmLowTemperature(OneWire_t* OneWire, uint8_t *ROM, int8_t temp)
{
	uint8_t tl, th, conf;
	if (!DS18B20_Is(ROM))
		return 0;

	if (temp > 125)
		temp = 125;

	if (temp < -55)
		temp = -55;

	/* Reset line */
	OneWire_Reset(OneWire);
	/* Select ROM number */
	OneWire_SelectWithPointer(OneWire, ROM);
	/* Read scratchpad command by onewire protocol */
	OneWire_WriteByte(OneWire, ONEWIRE_CMD_RSCRATCHPAD);

	/* Ignore first 2 bytes */
	OneWire_ReadByte(OneWire);
	OneWire_ReadByte(OneWire);

	th = OneWire_ReadByte(OneWire);
	tl = OneWire_ReadByte(OneWire);
	conf = OneWire_ReadByte(OneWire);
```

```
	tl = (uint8_t)temp;

	/* Reset line */
	OneWire_Reset(OneWire);
	/* Select ROM number */
	OneWire_SelectWithPointer(OneWire, ROM);
	/* Write scratchpad command by onewire protocol, only th, tl and conf register can be
written */
	OneWire_WriteByte(OneWire, ONEWIRE_CMD_WSCRATCHPAD);

	/* Write bytes */
	OneWire_WriteByte(OneWire, th);
	OneWire_WriteByte(OneWire, tl);
	OneWire_WriteByte(OneWire, conf);

	/* Reset line */
	OneWire_Reset(OneWire);
	/* Select ROM number */
	OneWire_SelectWithPointer(OneWire, ROM);
	/* Copy scratchpad to EEPROM of DS18B20 */
	OneWire_WriteByte(OneWire, ONEWIRE_CMD_CPYSCRATCHPAD);

	return 1;
}

uint8_t DS18B20_SetAlarmHighTemperature(OneWire_t* OneWire, uint8_t *ROM, int8_t temp)
{
	uint8_t tl, th, conf;
	if (!DS18B20_Is(ROM))
		return 0;

	if (temp > 125)
		temp = 125;

	if (temp < -55)
		temp = -55;

	/* Reset line */
	OneWire_Reset(OneWire);
	/* Select ROM number */
	OneWire_SelectWithPointer(OneWire, ROM);
	/* Read scratchpad command by onewire protocol */
	OneWire_WriteByte(OneWire, ONEWIRE_CMD_RSCRATCHPAD);

	/* Ignore first 2 bytes */
	OneWire_ReadByte(OneWire);
	OneWire_ReadByte(OneWire);

	th = OneWire_ReadByte(OneWire);
	tl = OneWire_ReadByte(OneWire);
	conf = OneWire_ReadByte(OneWire);

	th = (uint8_t)temp;

	/* Reset line */
	OneWire_Reset(OneWire);
	/* Select ROM number */
	OneWire_SelectWithPointer(OneWire, ROM);
	/* Write scratchpad command by onewire protocol, only th, tl and conf register can be
written */
	OneWire_WriteByte(OneWire, ONEWIRE_CMD_WSCRATCHPAD);

	/* Write bytes */
```

```
	OneWire_WriteByte(OneWire, th);
	OneWire_WriteByte(OneWire, tl);
	OneWire_WriteByte(OneWire, conf);

	/* Reset line */
	OneWire_Reset(OneWire);
	/* Select ROM number */
	OneWire_SelectWithPointer(OneWire, ROM);
	/* Copy scratchpad to EEPROM of DS18B20 */
	OneWire_WriteByte(OneWire, ONEWIRE_CMD_CPYSCRATCHPAD);

	return 1;
}

uint8_t DS18B20_DisableAlarmTemperature(OneWire_t* OneWire, uint8_t *ROM)
{
	uint8_t tl, th, conf;
	if (!DS18B20_Is(ROM))
		return 0;

	/* Reset line */
	OneWire_Reset(OneWire);
	/* Select ROM number */
	OneWire_SelectWithPointer(OneWire, ROM);
	/* Read scratchpad command by onewire protocol */
	OneWire_WriteByte(OneWire, ONEWIRE_CMD_RSCRATCHPAD);

	/* Ignore first 2 bytes */
	OneWire_ReadByte(OneWire);
	OneWire_ReadByte(OneWire);

	th = OneWire_ReadByte(OneWire);
	tl = OneWire_ReadByte(OneWire);
	conf = OneWire_ReadByte(OneWire);

	th = 125;
	tl = (uint8_t)-55;

	/* Reset line */
	OneWire_Reset(OneWire);
	/* Select ROM number */
	OneWire_SelectWithPointer(OneWire, ROM);
	/* Write scratchpad command by onewire protocol, only th, tl and conf register can be
written */
	OneWire_WriteByte(OneWire, ONEWIRE_CMD_WSCRATCHPAD);

	/* Write bytes */
	OneWire_WriteByte(OneWire, th);
	OneWire_WriteByte(OneWire, tl);
	OneWire_WriteByte(OneWire, conf);

	/* Reset line */
	OneWire_Reset(OneWire);
	/* Select ROM number */
	OneWire_SelectWithPointer(OneWire, ROM);
	/* Copy scratchpad to EEPROM of DS18B20 */
	OneWire_WriteByte(OneWire, ONEWIRE_CMD_CPYSCRATCHPAD);

	return 1;
}

uint8_t DS18B20_AlarmSearch(OneWire_t* OneWire)
{
	/* Start alarm search */
```

```
	return OneWire_Search(OneWire, DS18B20_CMD_ALARMSEARCH);
}

uint8_t DS18B20_AllDone(OneWire_t* OneWire)
{
	/* If read bit is low, then device is not finished yet with calculation temperature */
	return OneWire_ReadBit(OneWire);
}
```

3 Библиотека GPS-модуля

Файл: gps.h

```
/*
 * gps.h
 *
 *  Created on: Apr 25, 2023
 *      Author: Mihail M.
 */
#ifndef INC_GPS_H_
#define INC_GPS_H_

#include "cmsis_os.h"
#include "main.h"

/*GPS Config*/
#define GPS_CHOSEN_UART		huart2
#define GPS_BUFFER_SIZE		70
#define GPS_TASK_PRIORITY	3
#define GPS_TASK_PERIOD		40000


typedef struct
{
	uint8_t data_fix;

	float latitude;
	char latitude_direction;
	float longitude;
	char longitude_direction;

	uint8_t hours;
	uint8_t minutes;

	uint8_t day;
	uint8_t month;
	uint32_t year;

}GPS_Data_t ;

extern xQueueHandle xQueue_GPS_data;
extern UART_HandleTypeDef GPS_CHOSEN_UART;

HAL_StatusTypeDef GPS_Init (void);
#endif /* INC_GPS_H_ */
```

Файл: gps.c

```
/*
 * gps.c
 *
```

```
 *  Created on: Apr 28, 2023
 *      Author: Mihail M.
 */

#include <string.h>
#include <stdbool.h>
#include "gps.h"

xSemaphoreHandle xBinarySemaphore_GPS_ReciveData;

uint8_t GPS_buffer[GPS_BUFFER_SIZE] = {0,};
volatile uint8_t GPS_buffer_lenth = 0;
uint8_t temp_RX;

bool chek_R = false;
bool chek_n = false;

static HAL_StatusTypeDef GPS_GetData (uint8_t* nmea_buffer, uint32_t buffer_len, GPS_Data_t*
data);
static void vTask_GPS_GetData (void* pvParameter);

/**
  * @brief        Getting latitude, longitude, etc. from the data buffer.
  * @param[in]    nmea_buffer EMA data buffer
  * @param[in]    azimuth
  * @param[out]   K
  * @return       HAL_StatusTypeDef
  * @note         Data is parsed from a string RMC
  */
static HAL_StatusTypeDef GPS_GetData (uint8_t* nmea_buffer, uint32_t buffer_len, GPS_Data_t*
data)
{
      //$GNRMC,102030.000,A,5546.95900,N,03740.69200,E,0.12,49.75,200220,,,A*FF/r/n
      uint32_t i = 0x00;
      /*search for the required string*/
      while (!(nmea_buffer[i] == 'R'))
      {
            i++;
            if(i == buffer_len - 1)
                  return HAL_ERROR;
      }

      while (nmea_buffer[i] != ',')
            i++;
      i += 1;
      /*pointer to first byte of time*/
      uint32_t start_pos = i;

      while(nmea_buffer[i] != ',')
            i++;

      /*pointer to the second comma*/
      /*verification of received data*/
      if(nmea_buffer[i+1] == 'V')
            data->data_fix = 0;
      else
            data->data_fix = 1;

      i = start_pos;
      /*pointer to first byte of time*/

      if(nmea_buffer[i] != ',')
      {
            data->hours = (nmea_buffer[i] - '0') * 10 + (nmea_buffer[i+1] - '0');
```

```
            data->minutes = (nmea_buffer[i+2] - '0') * 10 + (nmea_buffer[i+3] - '0');
      }
      else
      {
            data->hours = 0;
            data->minutes = 0;
      }

      while (nmea_buffer[i] != ',')
            i++;
      i += 1;

      while (nmea_buffer[i] != ',')
            i++;
      i += 1;

      if(data->data_fix)
            data->latitude = (nmea_buffer[i] - '0') * 10 + (nmea_buffer[i+1] - '0') +
                                    ((nmea_buffer[i+2] - '0') * 10.0 + (nmea_buffer[i+3] - '0')
+
                                    ((nmea_buffer[i+5] - '0') * 10000 + (nmea_buffer[i+6] - '0')
* 1000 +
                                     (nmea_buffer[i+7] - '0') * 100 + (nmea_buffer[i+8] - '0') *
10 + nmea_buffer[i+9] - '0') / 100000.0) /60.0;
      else
            data->latitude = 0;

      while (nmea_buffer[i]!=',')
            i++;
      i += 1;
      /*pointer to the direction of latitude*/

      if(data->data_fix)
            data->latitude_direction = nmea_buffer[i];
      else
            data->latitude_direction = '0';

      while (nmea_buffer[i]!=',')                                  /*pointer to the longitude*/
            i++;
      i += 1;

      if(data->data_fix)
            data->longitude = (nmea_buffer[i] - '0') * 100 + (nmea_buffer[i+1] - '0') * 10 +
(nmea_buffer[i+2] - '0') +
                                    ((nmea_buffer[i+3] - '0') * 10.0 + (nmea_buffer[i+4] -
'0') + ((nmea_buffer[i+6] - '0') * 10000 +
                                     (nmea_buffer[i+7] - '0') * 1000 + (nmea_buffer[i+8] -
'0') * 100 +
                                     (nmea_buffer[i+9] - '0') * 10 + nmea_buffer[i+10] - '0')
/ 100000.0) / 60.0;
      else
            data->longitude = 0;

      while (nmea_buffer[i] != ',')
            i++;
      i += 1;
      /* pointer to the direction of longitude*/

      if(data->data_fix)
            data->longitude_direction = nmea_buffer[i];
      else
            data->longitude_direction = '0';

      while (nmea_buffer[i]!=',')
```

```
            i++;
      i += 1;
      while (nmea_buffer[i]!=',')
            i++;
      i += 1;
      while (nmea_buffer[i]!=',')
            i++;
      i += 1;
      /*pointer to the date(day)*/

      if(nmea_buffer[i] != ',')
      {
            data->day = (nmea_buffer[i] - '0') * 10 + (nmea_buffer[i+1] - '0');
            i += 2;
      /*pointer to the date(month)*/
            data->month = (nmea_buffer[i] - '0') * 10 + (nmea_buffer[i+1] - '0');
            i += 2;
      /*pointer to the date(year)*/
            data->year = (nmea_buffer[i]-'0')*10 + (nmea_buffer[i+1]-'0') + 2000;
      }
      else
      {
            data->day = 0;
            i += 2;
      /*pointer to the date(month)*/
            data->month = 0;
            i += 2;
      /*pointer to the date(year)*/
            data->year = 0;
      }

      if (data->data_fix)
            return HAL_OK;
      return HAL_ERROR;
}

/*
 * Creating a queue, semaphore, and task.
 */
HAL_StatusTypeDef GPS_Init (void)
{
      if(xTaskCreate(vTask_GPS_GetData, NULL, configMINIMAL_STACK_SIZE, NULL,
GPS_TASK_PRIORITY, NULL) == errCOULD_NOT_ALLOCATE_REQUIRED_MEMORY)
            return HAL_ERROR;

      xQueue_GPS_data = xQueueCreate(1,sizeof(GPS_Data_t));
      if(xQueue_GPS_data == NULL)
            return HAL_ERROR;

      vSemaphoreCreateBinary(xBinarySemaphore_GPS_ReciveData);
      if(xBinarySemaphore_GPS_ReciveData == NULL)
            return HAL_ERROR;

      xSemaphoreTake(xBinarySemaphore_GPS_ReciveData, portMAX_DELAY);

      return HAL_OK;
}

/*
 *
 */
static void vTask_GPS_GetData (void* pvParameter)
{
      portTickType xLastWakeTime = xTaskGetTickCount();
```

```
	GPS_Data_t Data = {0, .latitude_direction = '0', .longitude_direction = '0'};


	for(;;)
	{
		xQueueReceive(xQueue_GPS_data, &Data, 0);
		HAL_UART_Receive_IT(&GPS_CHOSEN_UART, &temp_RX, 1);
		xSemaphoreTake(xBinarySemaphore_GPS_ReciveData, portMAX_DELAY);
		GPS_GetData(GPS_buffer, GPS_BUFFER_SIZE, &Data);
		xQueueSendToBack(xQueue_GPS_data, &Data, 0);
		memset(GPS_buffer,'\0',GPS_BUFFER_SIZE);
		vTaskDelayUntil(&xLastWakeTime, GPS_TASK_PERIOD / portTICK_RATE_MS);
	}
	vTaskDelete(NULL);
}

/*
 * The UART interrupt callback function that fills the NMEA buffer.
 */

void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
	if(huart == &GPS_CHOSEN_UART)
	{

		uint8_t rx_char = (uint8_t) (GPS_CHOSEN_UART.Instance->DR & 0x00FF);
		/*
		 * String RMC search.
		 */
		chek_R = (rx_char == 'R')? true : chek_R;
		chek_n      = ((rx_char == '\n') && chek_R) ? true : chek_n;
		if (chek_R && !chek_n)
		{
			GPS_buffer[GPS_buffer_lenth++] = rx_char;
			HAL_UART_Receive_IT(&GPS_CHOSEN_UART, &temp_RX, 1);
		}
		else if (chek_R && chek_n)
		{
			GPS_buffer_lenth = 0;
			chek_R = false;
			chek_n = false;
			xSemaphoreGiveFromISR(xBinarySemaphore_GPS_ReciveData, NULL);
		}
		else if (!(chek_R && !chek_n))
		{
			HAL_UART_Receive_IT(&GPS_CHOSEN_UART, &temp_RX, 1);
		}
	}
}
```

4 Библиотека модуля индикации

Файл: led.h

```
/*
 * led.h
 *
 *  Created on: Apr 28, 2023
 *      Author: Mihail M.
 */

#ifndef INC_LED_H_
#define INC_LED_H_
```

```
#include "main.h"
#include "cmsis_os.h"

/*
 * Config
 */
#define     LED_GPIO                LED_GPIO_Port
#define     LED_PIN                 LED_Pin
#define     LED_TASK_PRIORITY       1

typedef enum
{
      LED_Mode1, //Однопроблесковый
      LED_Mode2, //Затмевающийся
      LED_Mode3, //Частопроблесковый
      LED_Mode4, //Пульсирующий
      LED_Mode5, //Группочастопроблесковый
      LED_Mode6, //Прерывистый пульсирующий
      LED_Mode7, //Двухпроблесковый
}LED_Mode_t;

extern xQueueHandle xQueue_GSM_set_mode;

HAL_StatusTypeDef Led_Init(void);
#endif /* INC_LED_H_ */
```

Файл: led.c

```
/*
 * led.c
 *
 *  Created on: Apr 28, 2023
 *      Author: Mihail M.
 */
#include "led.h"

static void vTask_Led (void* pvParameter);
static void Mode_1 (TickType_t* pxPreviousWakeTime);
static void Mode_2 (TickType_t* pxPreviousWakeTime);
static void Mode_3 (TickType_t* pxPreviousWakeTime);
static void Mode_4 (TickType_t* pxPreviousWakeTime);
static void Mode_5 (TickType_t* pxPreviousWakeTime);
static void Mode_6 (TickType_t* pxPreviousWakeTime);
static void Mode_7 (TickType_t* pxPreviousWakeTime);
/*
 * Creation task.
 */
HAL_StatusTypeDef Led_Init(void)
{
      if(xTaskCreate(vTask_Led, NULL, configMINIMAL_STACK_SIZE, NULL, LED_TASK_PRIORITY, NULL)
== errCOULD_NOT_ALLOCATE_REQUIRED_MEMORY)
                  return HAL_ERROR;
      return HAL_OK;
}
/*
 * The task of processing and selecting the flashing mode.
 */
static void vTask_Led (void* pvParameter)
{
      TickType_t pxPreviousWakeTime = xTaskGetTickCount ();
      uint8_t mode = 0;
      for(;;)
```

```
	{
		xQueueReceive(xQueue_GSM_set_mode, &mode, 0);
		switch (mode)
		{
			case LED_Mode1:
				Mode_1 (&pxPreviousWakeTime);
				break;
			case LED_Mode2:
				Mode_2 (&pxPreviousWakeTime);
				break;
			case LED_Mode3:
				Mode_3 (&pxPreviousWakeTime);
				break;
			case LED_Mode4:
				Mode_4 (&pxPreviousWakeTime);
				break;
			case LED_Mode5:
				Mode_5 (&pxPreviousWakeTime);
				break;
			case LED_Mode6:
				Mode_6 (&pxPreviousWakeTime);
				break;
			case LED_Mode7:
				Mode_7 (&pxPreviousWakeTime);
				break;
			default:
				Mode_1 (&pxPreviousWakeTime);
				break;
		}
	}
	vTaskDelete(NULL);
}
/*
 * Single flash mode.
 */
static void Mode_1 (TickType_t* pxPreviousWakeTime)
{
	HAL_GPIO_WritePin(LED_GPIO, LED_PIN, GPIO_PIN_SET);
	vTaskDelayUntil(pxPreviousWakeTime, 700 / portTICK_RATE_MS);
	HAL_GPIO_WritePin(LED_GPIO, LED_PIN, GPIO_PIN_RESET);
	vTaskDelayUntil(pxPreviousWakeTime, 2800 / portTICK_RATE_MS);
}
/*
 * Eclipse Mode.
 */
static void Mode_2 (TickType_t* pxPreviousWakeTime)
{
	HAL_GPIO_WritePin(LED_GPIO, LED_PIN, GPIO_PIN_SET);
	vTaskDelayUntil(pxPreviousWakeTime, 2800 / portTICK_RATE_MS);
	HAL_GPIO_WritePin(LED_GPIO, LED_PIN, GPIO_PIN_RESET);
	vTaskDelayUntil(pxPreviousWakeTime, 730 / portTICK_RATE_MS);
}
/*
 * Frequent flash mode.
 */
static void Mode_3 (TickType_t* pxPreviousWakeTime)
{
	HAL_GPIO_WritePin(LED_GPIO, LED_PIN, GPIO_PIN_SET);
	vTaskDelayUntil(pxPreviousWakeTime, 590 / portTICK_RATE_MS);
	HAL_GPIO_WritePin(LED_GPIO, LED_PIN, GPIO_PIN_RESET);
	vTaskDelayUntil(pxPreviousWakeTime, 400 / portTICK_RATE_MS);
}
/*
 * Pulsing mode.
```

```
 */
static void Mode_4 (TickType_t* pxPreviousWakeTime)
{
	HAL_GPIO_WritePin(LED_GPIO, LED_PIN, GPIO_PIN_SET);
	vTaskDelayUntil(pxPreviousWakeTime, 66 / portTICK_RATE_MS);
	HAL_GPIO_WritePin(LED_GPIO, LED_PIN, GPIO_PIN_RESET);
	vTaskDelayUntil(pxPreviousWakeTime, 55 / portTICK_RATE_MS);
}
/*
 * Group flash mode.
 */
static void Mode_5 (TickType_t* pxPreviousWakeTime)
{
	for (int var = 0; var < 3; ++var)
	{
		HAL_GPIO_WritePin(LED_GPIO, LED_PIN, GPIO_PIN_SET);
		vTaskDelayUntil(pxPreviousWakeTime, 500 / portTICK_RATE_MS);
		HAL_GPIO_WritePin(LED_GPIO, LED_PIN, GPIO_PIN_RESET);
		vTaskDelayUntil(pxPreviousWakeTime, 500 / portTICK_RATE_MS);
	}
	HAL_GPIO_WritePin(LED_GPIO, LED_PIN, GPIO_PIN_SET);
	vTaskDelayUntil(pxPreviousWakeTime, 500 / portTICK_RATE_MS);
	HAL_GPIO_WritePin(LED_GPIO, LED_PIN, GPIO_PIN_RESET);
	vTaskDelayUntil(pxPreviousWakeTime, 2900 / portTICK_RATE_MS);
}
/*
 * Intermittent pulsing.
 */
static void Mode_6 (TickType_t* pxPreviousWakeTime)
{
	for (int var = 0; var < 2; ++var)
	{
		HAL_GPIO_WritePin(LED_GPIO, LED_PIN, GPIO_PIN_SET);
		vTaskDelayUntil(pxPreviousWakeTime, 60 / portTICK_RATE_MS);
		HAL_GPIO_WritePin(LED_GPIO, LED_PIN, GPIO_PIN_RESET);
		vTaskDelayUntil(pxPreviousWakeTime, 60 / portTICK_RATE_MS);
	}
	HAL_GPIO_WritePin(LED_GPIO, LED_PIN, GPIO_PIN_SET);
	vTaskDelayUntil(pxPreviousWakeTime, 60 / portTICK_RATE_MS);
	HAL_GPIO_WritePin(LED_GPIO, LED_PIN, GPIO_PIN_RESET);
	vTaskDelayUntil(pxPreviousWakeTime, 2800 / portTICK_RATE_MS);
}
/*
 * Double flash mode.
 */
static void Mode_7 (TickType_t* pxPreviousWakeTime)
{
	HAL_GPIO_WritePin(LED_GPIO, LED_PIN, GPIO_PIN_SET);
	vTaskDelayUntil(pxPreviousWakeTime, 600 / portTICK_RATE_MS);
	HAL_GPIO_WritePin(LED_GPIO, LED_PIN, GPIO_PIN_RESET);
	vTaskDelayUntil(pxPreviousWakeTime, 600 / portTICK_RATE_MS);
	HAL_GPIO_WritePin(LED_GPIO, LED_PIN, GPIO_PIN_SET);
	vTaskDelayUntil(pxPreviousWakeTime, 600 / portTICK_RATE_MS);
	HAL_GPIO_WritePin(LED_GPIO, LED_PIN, GPIO_PIN_RESET);
	vTaskDelayUntil(pxPreviousWakeTime, 2800 / portTICK_RATE_MS);
}
```

5 Библиотека датчика LSM303DHLC

Файл: matrix.h

```
/*
 * Matrix.h
 *
 *  Created on: Mar 14, 2023
 *      Author: Mihail M.
 */

#ifndef INC_MATRIX_H_
#define INC_MATRIX_H_

#define M_PI           3.14159265358979323846
#define CON_DEGREE  (180.f/M_PI)

typedef float Vector_3f_t[3];
typedef float Matrix_3f_t[3][3];

void MATRIX_Vect_3f_Transf(Vector_3f_t V, const Matrix_3f_t M, Vector_3f_t V_result);
void MATRIX_Vect_3f_Dif (Vector_3f_t V1, const Vector_3f_t V2, Vector_3f_t V_result);
void MATRIX_Matrix_3f_Mult(Matrix_3f_t M1, Matrix_3f_t M2, Matrix_3f_t M_result);
void MATRIX_Vect_3f_EMA(Vector_3f_t V1, Vector_3f_t V2, float K);
void MATRIX_Matrix_3f_SetRow (Vector_3f_t V, Matrix_3f_t M, int row);
float MATRIX_Vect_3f_Dot (Vector_3f_t V1, Vector_3f_t V2);
float MATRIX_Vect_3f_Length (Vector_3f_t V);
float MATRIX_Vect_3f_LengthSqr (Vector_3f_t V);
float MATRIX_Vect_3f_Cos (Vector_3f_t V1, Vector_3f_t V2);
float MATRIX_Vect_3f_Ang (Vector_3f_t V1, Vector_3f_t V2);
void MATRIX_Vect_3f_Cross (Vector_3f_t V1, Vector_3f_t V2, Vector_3f_t V_result);
void MATRIX_Vect_3f_Notmalize(Vector_3f_t V, Vector_3f_t V_result);
void MATRIX_Vect_3f_NotmalizeSelf(Vector_3f_t V);
void MATRIX_Vect_3f_Project(Vector_3f_t V1, Vector_3f_t V2, Vector_3f_t V_result);
void MATRIX_Vect_3f_Reject(Vector_3f_t V1, Vector_3f_t V2, Vector_3f_t V_result );
void MATRIX_Matrix_3f_GetEuler (Matrix_3f_t M1, Vector_3f_t Euler);
#endif /* INC_MATRIX_H_ */
```

Файл: matrix.c

```
/*
 * matrix.c
 *
 *  Created on: 7 мар. 2023 г.
 *      Author: Mihail M.
 */
#include <math.h>
#include "matrix.h"
/**
  * @brief        Matrix multiplication.
  * @param[in]    A Pointer to vector A
  * @param[in]    B Pointer to matrix B
  * @param[out] C Pointer to matrix C
  * @return       none
  * @note         A[3] * B [3 x 3] = C [3]
  */
void MATRIX_Vect_3f_Transf(Vector_3f_t V, const Matrix_3f_t M, Vector_3f_t V_result)
{

      V_result[0] = V[0]*M[0][0] + V[1]*M[1][0] + V[2]*M[2][0];
      V_result[1] = V[0]*M[0][1] + V[1]*M[1][1] + V[2]*M[2][1];
      V_result[2] = V[0]*M[0][2] + V[1]*M[1][2] + V[2]*M[2][2];
}

/**
  * @brief        Matrix subtraction.
  * @param[in]    A Pointer to vector A
```

```
  * @param[in]    B Pointer to vector B
  * @param[out] C Pointer to vector C
  * @return       none
  * @note         V1[i]-V2[i]=V_result[i]
  */
void MATRIX_Vect_3f_Dif (Vector_3f_t V1, const Vector_3f_t V2, Vector_3f_t V_result)
{
      V_result[0] = V1[0] - V2[0];
      V_result[1] = V1[1] - V2[1];
      V_result[2] = V1[2] - V2[2];
}

/**
  * @brief        Matrix multiplication.
  * @param[in]    M1 Pointer to matrix M1
  * @param[in]    M2 Pointer to matrix M2
  * @param[out] M_result Pointer to matrix M_result
  * @return       none
  * @note         M1[3 x 3] * M2 [3 x 3] = M_result [3 x 3]
  */
void MATRIX_Matrix_3f_Mult(Matrix_3f_t M1, Matrix_3f_t M2, Matrix_3f_t M_result)
{
      M_result[0][0] = M1[0][0]*M2[0][0] + M1[0][1]*M2[1][0] + M1[0][2]*M2[2][0];
      M_result[0][1] = M1[0][0]*M2[0][1] + M1[0][1]*M2[1][1] + M1[0][2]*M2[2][1];
      M_result[0][2] = M1[0][0]*M2[0][2] + M1[0][1]*M2[1][2] + M1[0][2]*M2[2][2];

      M_result[1][0] = M1[1][0]*M2[0][0] + M1[1][1]*M2[1][0] + M1[1][2]*M2[2][0];
      M_result[1][1] = M1[1][0]*M2[0][1] + M1[1][1]*M2[1][1] + M1[1][2]*M2[2][1];
      M_result[1][2] = M1[1][0]*M2[0][2] + M1[1][1]*M2[1][2] + M1[1][2]*M2[2][2];

      M_result[2][0] = M1[2][0]*M2[0][0] + M1[2][1]*M2[1][0] + M1[2][2]*M2[2][0];
      M_result[2][1] = M1[2][0]*M2[0][1] + M1[2][1]*M2[1][1] + M1[2][2]*M2[2][1];
      M_result[2][2] = M1[2][0]*M2[0][2] + M1[2][1]*M2[1][2] + M1[2][2]*M2[2][2];

}

void MATRIX_Vect_3f_EMA(Vector_3f_t V1, Vector_3f_t V2, float K)
{
      V1[0] = V1[0]*(1-K) + V2[0]*K;
      V1[1] = V1[1]*(1-K) + V2[1]*K;
      V1[2] = V1[2]*(1-K) + V2[2]*K;
}

/**
  * @brief        Setting a row in a matrix.
  * @param[in]    V Pointer to vector V.
  * @param[in]    M Pointer to matrix M.
  * @param[in]  row Selected row.
  * @return       none
  * @note         none
  */
void MATRIX_Matrix_3f_SetRow (Vector_3f_t V, Matrix_3f_t M, int row)
{
      M[row][0] = V[0];
      M[row][1] = V[1];
      M[row][2] = V[2];
}

/**
  * @brief        Get dot vector product.
  * @param[in]    V1 Pointer to vector V1.
  * @param[in]    V2 Pointer to vector V2.
  * @return       Vector product.
  * @note         (V1,V2)
```

```
  */
float MATRIX_Vect_3f_Dot (Vector_3f_t V1, Vector_3f_t V2)
{
      return V1[0]*V2[0] + V1[1]*V2[1] + V1[2]*V2[2];
}

/**
  * @brief        Get vector length.
  * @param[in]    V Pointer to vector V.
  * @return       Length.
  * @note         none
  */
float MATRIX_Vect_3f_Length (Vector_3f_t V)
{
      return sqrt(V[0]*V[0] + V[1]*V[1] + V[2]*V[2]);
}

/**
  * @brief        Get vector length squared.
  * @param[in]    V Pointer to vector V.
  * @return       Length squared.
  * @note         none
  */
float MATRIX_Vect_3f_LengthSqr (Vector_3f_t V)
{
      return V[0]*V[0] + V[1]*V[1] + V[2]*V[2];
}

/**
  * @brief        Get cosine between vectors.
  * @param[in]    V1 Pointer to vector V1.
  * @param[in]    V2 Pointer to vector V2.
  * @return       Cosine between vectors.
  * @note         cos(V1,V2)
  */
float MATRIX_Vect_3f_Cos (Vector_3f_t V1, Vector_3f_t V2)
{
      return MATRIX_Vect_3f_Dot(V1,
V2)/sqrt(MATRIX_Vect_3f_LengthSqr(V1)*MATRIX_Vect_3f_LengthSqr(V2));
}

/**
  * @brief        Get angle between vectors.
  * @param[in]    V1 Pointer to vector V1.
  * @param[in]    V2 Pointer to vector V2.
  * @return       Angle between vectors.
  * @note         ang(V1,V2)
  */
float MATRIX_Vect_3f_Ang (Vector_3f_t V1, Vector_3f_t V2)
{
      return acos( MATRIX_Vect_3f_Dot(V1, V2)/sqrt(
MATRIX_Vect_3f_LengthSqr(V1)*MATRIX_Vect_3f_LengthSqr(V2) ) );
}

/**
  * @brief        Cross vector product.
  * @param[in]    V1 Pointer to vector V1.
  * @param[in]    V2 Pointer to vector V2.
  * @param[out] V_result Pointer to vector V_result
  * @return       nope
  * @note         [V1[3],V2[3]] = V_result[3]
  */
void MATRIX_Vect_3f_Cross (Vector_3f_t V1, Vector_3f_t V2, Vector_3f_t V_result)
{
```

```
	V_result[0] = V1[1]*V2[2] - V1[2]*V2[1];
	V_result[1] = V1[2]*V2[0] - V1[0]*V2[2];
	V_result[2] = V1[0]*V2[1] - V1[1]*V2[0];
}

/**
  * @brief        Normalize vector.
  * @param[in]    V Pointer to vector V.
  * @param[out] V_result Pointer to vector V_result
  * @return       nope
  * @note         nope
  */
void MATRIX_Vect_3f_Notmalize(Vector_3f_t V, Vector_3f_t V_result)
{
	float temp = 0.f;
	temp = 1.f/MATRIX_Vect_3f_Length(V);

	V_result[0] = V[0] * temp;
	V_result[1] = V[1] * temp;
	V_result[2] = V[2] * temp;
}

/**
  * @brief        Normalize vector.
  * @param[in/out]      V Pointer to vector V.
  * @return       nope
  * @note         nope
  */
void MATRIX_Vect_3f_NotmalizeSelf(Vector_3f_t V)
{
	float temp = 0.f;
	temp = 1.f/MATRIX_Vect_3f_Length(V);

	V[0] = V[0] * temp;
	V[1] = V[1] * temp;
	V[2] = V[2] * temp;
}

/**
  * @brief        Get projection vector.
  * @param[in]    V1 Pointer to vector V1.
  * @param[in]    V2 Pointer to vector V2.
  * @param[out] V_result Pointer to vector V_result.
  * @return       nope
  * @note         proreject(V1,V2)
  */
void MATRIX_Vect_3f_Project(Vector_3f_t V1, Vector_3f_t V2, Vector_3f_t V_result)
{
	float temp = 0.f;
	temp = MATRIX_Vect_3f_Dot(V1, V2) / MATRIX_Vect_3f_LengthSqr(V2);

	V_result[0] = V2[0] * temp;
	V_result[1] = V2[1] * temp;
	V_result[2] = V2[2] * temp;
}

/**
  * @brief        Get rejection vector.
  * @param[in]    V1 Pointer to vector V1.
  * @param[in]    V2 Pointer to vector V2.
  * @param[out] V_result Pointer to vector V_result.
  * @return       nope
  * @note         reject(V1,V2)
  */
```

```
void MATRIX_Vect_3f_Reject(Vector_3f_t V1, Vector_3f_t V2, Vector_3f_t V_result )
{
	float temp = 0.f;
	temp = MATRIX_Vect_3f_Dot(V1, V2) / MATRIX_Vect_3f_LengthSqr(V2);

	V_result[0] = V1[0] - V2[0] * temp;
	V_result[1] = V1[1] - V2[1] * temp;
	V_result[2] = V1[2] - V2[2] * temp;
}

/**
  * @brief        Get Euler angle.
  * @param[in]    M1 Pointer to matrix M1.
  * @param[out] Euler Pointer to vector Euler.
  * @return       nope
  * @note         Euler[3] = {roll,pitch,yaw};
  */
void MATRIX_Matrix_3f_GetEuler (Matrix_3f_t M1, Vector_3f_t Euler)
{
	Euler[0] = atan2(-M1[1][2],M1[2][2]);
	Euler[1] = asin(M1[0][2]);
	Euler[2] = atan2(-M1[0][1],M1[0][0]);
}
```

Файл: lsm303dhlc.h

```
/*
 * lsm303dlhc.h
 *
 *  Created on: 7 мар. 2023 г.
 *      Author: Mihail M.
 */

#ifndef INC_LSM303DLHC_H_
#define INC_LSM303DLHC_H_

#include "main.h"
#include "cmsis_os.h"


#define LSM303DLHC_MULTIPLE_FLAG                0x80

#define LSM_MAX_TIMEOUT                         0x0001

#define LSM303_DEBUG                            0
#define LSM303_FREERTOS                         1

#define MAX(a, b) (a >= b ? a : b)
#define MIN(a, b) (a <= b ? a : b)

#define LSM303DLHC_ACCEL_ADDRESS                0x32
#define LSM303DLHC_MAG_ADDRESS                  0x3C
/************************FREERTOS*****************************************/
#define LSM303DLHC_TASK_PERIOD                  30
#define LSM303DLHC_TASK_PRIORITY                3

/************************Register map*************************************/
// Accel
#define LSM303DLHC_WHO_AM_I_A                   0x0F
#define LSM303DLHC_CTRL_REG1_A                  0x20  // Data rate, Low-power mode, Z,Y,X-axis
enable.
#define LSM303DLHC_CTRL_REG2_A                  0x21  // High-pass filter.
#define LSM303DLHC_CTRL_REG3_A                  0x22  // Interrupts INT1.
```

```
#define LSM303DLHC_CTRL_REG4_A                  0x23  // Update, Scale, Resolution, etc.
#define LSM303DLHC_CTRL_REG5_A                  0x24  // Reboot memory, FIFO, etc.
#define LSM303DLHC_CTRL_REG6_A                  0x25  // Interrupts PAD2
#define LSM303DLHC_REFERENCE_A                  0x26
#define LSM303DLHC_STATUS_REG_A                 0x27
#define LSM303DLHC_OUT_X_L_A                    0x28
#define LSM303DLHC_OUT_X_H_A                    0x29
#define LSM303DLHC_OUT_Y_L_A                    0x2A
#define LSM303DLHC_OUT_Y_H_A                    0x2B
#define LSM303DLHC_OUT_Z_L_A                    0x2C
#define LSM303DLHC_OUT_Z_H_A                    0x2D

// Mag
#define LSM303DLHC_CRA_REG_M                    0x00
#define LSM303DLHC_CRB_REG_M                    0x01
#define LSM303DLHC_MR_REG_M                     0x02
#define LSM303DLHC_OUT_X_H_M                    0x03
#define LSM303DLHC_OUT_X_L_M                    0x04
#define LSM303DLHC_OUT_Z_H_M                    0x05
#define LSM303DLHC_OUT_Z_L_M                    0x06
#define LSM303DLHC_OUT_Y_H_M                    0x07
#define LSM303DLHC_OUT_Y_L_M                    0x08
#define LSM303DLHC_SR_REG_M                     0x09
#define LSM303DLHC_IRA_REG_M                    0x0A
#define LSM303DLHC_IRB_REG_M                    0x0B
#define LSM303DLHC_IRC_REG_M                    0x0C
#define LSM303DLHC_TEMP_OUT_H_M                 0x31
#define LSM303DLHC_TEMP_OUT_L_M                 0x32

/* Address of device */
#define LSM303DLHC_I_AM_A                       0x33

/*********************Bit mask***********************/

/*********************Accel*************************/
/*********************CTRL_REG1_A*******************/
typedef enum
{
      LSM303DLHC_ACCEL_DATA_RATE_OFF      = (0<<4),
      LSM303DLHC_ACCEL_DATA_RATE_1HZ      = (1<<4),
      LSM303DLHC_ACCEL_DATA_RATE_10HZ     = (2<<4),
      LSM303DLHC_ACCEL_DATA_RATE_25HZ     = (3<<4),
      LSM303DLHC_ACCEL_DATA_RATE_50HZ     = (4<<4),
      LSM303DLHC_ACCEL_DATA_RATE_100HZ    = (5<<4),
      LSM303DLHC_ACCEL_DATA_RATE_200HZ    = (6<<4),
      LSM303DLHC_ACCEL_DATA_RATE_400HZ    = (7<<4),
      LSM303DLHC_ACCEL_DATA_RATE_1_620kHZ = (8<<4),   // Low-power mode
      LSM303DLHC_ACCEL_DATA_RATE_1_344kHZ = (9<<4)    // Low-power mode (5.376 kHz)
} LSM303DLHC_Accel_DataRate_t;

typedef enum
{
      LSM303DLHC_ACCEL_MODE_CONTINUOUS    = (0<<7),
      LSM303DLHC_ACCEL_MODE_SINGLE              = (1<<7)
} LSM303DLHC_Accel_Update_Mode_t;

typedef enum
{
      LSM303DLHC_ACCEL_NORMAL_MODE        = (0<<3),
      LSM303DLHC_ACCEL_LOW_POWER_MODE     = (1<<3)
} LSM303DLHC_Accel_Power_Mode_t;

typedef enum
{
```

```
      LSM303DLHC_ACCEL_X_ENABLE                 = (1<<0),
      LSM303DLHC_ACCEL_Y_ENABLE                 = (1<<1),
      LSM303DLHC_ACCEL_Z_ENABLE                 = (1<<2),
      LSM303DLHC_ACCEL_ALL_ENABLE               = 0x07

} LSM303DLHC_Accel_Enable_t;

/*********************CTRL_REG4_A******************/
typedef enum
{
      LSM303DLHC_ACCEL_SCALE_2G           = (0<<4),
      LSM303DLHC_ACCEL_SCALE_4G           = (1<<4),
      LSM303DLHC_ACCEL_SCALE_8G           = (2<<4),
      LSM303DLHC_ACCEL_SCALE_16G          = (3<<4)
} LSM303DLHC_Accel_Scale_t;

typedef enum
{
      LSM303DLHC_ACCEL_HR_OFF                   = (0<<3),
      LSM303DLHC_ACCEL_HR_ON                    = (1<<3)
} LSM303DLHC_Accel_HR_t;

/*********************Mag**************************/
/*********************CRA_REG_M********************/
typedef enum
{
      LSM303DLHC_MAG_DATA_RATE_0_75HZ     = (0<<2),
      LSM303DLHC_MAG_DATA_RATE_1_5HZ      = (1<<2),
      LSM303DLHC_MAG_DATA_RATE_3_0HZ      = (2<<2),
      LSM303DLHC_MAG_DATA_RATE_7_5HZ      = (3<<2),
      LSM303DLHC_MAG_DATA_RATE_15HZ       = (4<<2),
      LSM303DLHC_MAG_DATA_RATE_30HZ       = (5<<2),
      LSM303DLHC_MAG_DATA_RATE_75HZ       = (6<<2),
      LSM303DLHC_MAG_DATA_RATE_220HZ      = (7<<2)
} LSM303DLHC_Mag_DataRate_t;

typedef enum
{
      LSM303DLHC_MAG_TEMP_ENABLE_OFF            = (0<<7),
      LSM303DLHC_MAG_TEMP_ENABLE_ON       = (1<<7),

} LSM303DLHC_Mag_Temp_Enable_t;

/*********************CRB_REG_M********************/
typedef enum
{
      LSM303DLHC_MAG_RANGE_1_3GAUSS       = (1<<5),
      LSM303DLHC_MAG_RANGE_1_9GAUSS       = (2<<5),
      LSM303DLHC_MAG_RANGE_2_5GAUSS       = (3<<5),
      LSM303DLHC_MAG_RANGE_4_0GAUSS       = (4<<5),
      LSM303DLHC_MAG_RANGE_4_7GAUSS       = (5<<5),
      LSM303DLHC_MAG_RANGE_5_6GAUSS       = (6<<5),
      LSM303DLHC_MAG_RANGE_8_1GAUSS       = (7<<5)
} LSM303DLHC_Mag_Range_t;

/*********************CRM_REG_M********************/
typedef enum
{
      LSM303DLHC_MAG_MODE_CONTINUOUS      = 0x00,
      LSM303DLHC_MAG_MODE_SINGLE          = 0x01,
      LSM303DLHC_MAG_MODE_SLEEP           = 0x02
} LSM303DLHC_Mag_Mode_t;

/*********************MagErrorData*****************/
```

```
typedef enum
{
	LSM303DLHC_Mag_Data_Error_NO_overflow	= 0x00,
	LSM303DLHC_Mag_Data_Error_YES_overflow	= 0x01,
} LSM303DLHC_Mag_Data_Error_t;


/*********************InitStruct******************/
typedef struct
{
	I2C_HandleTypeDef*				I2C_handler;

	LSM303DLHC_Accel_DataRate_t		Accel_DataRate;
	LSM303DLHC_Accel_Power_Mode_t		Accel_Power_Mode;
	LSM303DLHC_Accel_Enable_t		Accel_Enable;

	LSM303DLHC_Accel_HR_t			Accel_HR;
	LSM303DLHC_Accel_Scale_t		Accel_Scale;
	LSM303DLHC_Accel_Update_Mode_t		Accel_Update_Mode;

	LSM303DLHC_Mag_DataRate_t		Mag_DataRate;
	LSM303DLHC_Mag_Temp_Enable_t		Mag_Temp_Enable;

	LSM303DLHC_Mag_Range_t			Mag_Range;

	LSM303DLHC_Mag_Mode_t			Mag_Mode;

} LSM303DLHC_HandleTypeDef;

/*********************DataStruct******************/
typedef struct
{

	float data_A[3];
	float data_M[3];

} LSM303DLHC_Data_t;

typedef float Vector_3f_t[3];
typedef float Matrix_3f_t[3][3];

#if (LSM303_FREERTOS == 1)
typedef struct
{
	float azimuth;
	float grav_incline;
}LSM303DLHC_Azimuth_GarvInkl_t;

extern LSM303DLHC_HandleTypeDef LSM303DLHC_Handle_main;
extern xQueueHandle xQueue_LSM303DLHC_angles;
#endif

#ifdef MAG_CALIBRATION

void LSM303DLHC_TransMagData (LSM303DLHC_HandleTypeDef* LSM303DLHC_Handler,
UART_HandleTypeDef* UART_Handle);
void LSM303DLHC_TransAcellData (LSM303DLHC_HandleTypeDef* LSM303DLHC_Handler,
UART_HandleTypeDef* UART_Handle);

void LSM303DLHC_TransMagCalibData (LSM303DLHC_HandleTypeDef* LSM303DLHC_Handler,
UART_HandleTypeDef* UART_Handle);

void LSM303DLHC_TransAzimuth (LSM303DLHC_HandleTypeDef* LSM303DLHC_Handler,
UART_HandleTypeDef* UART_Handle);
```

```
#endif

HAL_StatusTypeDef LSM303DLHC_Init(LSM303DLHC_HandleTypeDef* LSM303DLHC_Handler);

void LSM303DLHC_AccelGetData (LSM303DLHC_HandleTypeDef* LSM303DLHC_Handler, LSM303DLHC_Data_t*
Data);
void LSM303DLHC_MagGetData (LSM303DLHC_HandleTypeDef* LSM303DLHC_Handler, LSM303DLHC_Data_t*
Data);

void LSM303DLHC_GetAzimuth_GravIvclin (LSM303DLHC_HandleTypeDef* LSM303DLHC_Handler, float*
azimuth, float* grav_incline);
#endif /* INC_LSM303DLHC_H_ */
```

Файл: lsm303dhlc.c

```
/*
 * lsm303dhlc.c
 *
 *  Created on: 7 мар. 2023 г.
 *      Author: Mihail M.
 */
#include <stdio.h>
#include "lsm303dlhc.h"
#include "matrix.h"

const float CalibMatix_h_M[3] = {12.499429, -120.765112, 32.770673};

const float CalibMatix_A_M[3][3] = {{0.803805, 0.001796, 0.014684},
                                    {0.001796, 0.769718, 0.000536},
                                    {0.014684, 0.000536, 0.874538}};

const float CalibMatix_h_A[3] = {108.896153, 430.867198, -79.326651};

const float CalibMatix_A_A[3][3] = {{0.989742,  0.001388, -0.022085},
                                    {0.001388,  0.972571, 0.007682},
                                    {-0.022085, 0.007682, 1.017349}};

static void LSM303DLHC_AccelWrite (LSM303DLHC_HandleTypeDef* LSM303DLHC_Handler, uint8_t
RegisterAddr, uint8_t value);
static uint8_t LSM303DLHC_AccelRead (LSM303DLHC_HandleTypeDef* LSM303DLHC_Handler, uint8_t
RegisterAddr);
static void LSM303DLHC_MagWrite (LSM303DLHC_HandleTypeDef* LSM303DLHC_Handler, uint8_t
RegisterAddr, uint8_t value);
static uint8_t LSM303DLHC_MagRead (LSM303DLHC_HandleTypeDef* LSM303DLHC_Handler, uint8_t
RegisterAddr);
static void LSM303DLHC_Filt_EMA (float* grav_incline, float* azimuth, float K);
static void vTask_LSM303DLHC_GetAzimuth_GravIvclin (void* pvParameter);

/**
  * @brief        LSM303DLHC Initialization Function
  * @param[in]    LSM303DLHC_Handler Points to an LSM303DLHC_HandleTypeDef structure that
contains
  *                          the configuration information for the specified LSM303DLHC.
  * @retval       HAL_StatusTypeDef
  */
HAL_StatusTypeDef LSM303DLHC_Init(LSM303DLHC_HandleTypeDef* LSM303DLHC_Handler)
{

      if(LSM303DLHC_AccelRead(LSM303DLHC_Handler, LSM303DLHC_WHO_AM_I_A) != LSM303DLHC_I_AM_A)
            return HAL_ERROR;

      uint8_t temp = 0x00;
```

```
	/************************************************************************************/
	//Set CTRL_REG1_A
	temp = LSM303DLHC_Handler->Accel_DataRate | LSM303DLHC_Handler->Accel_Power_Mode |
LSM303DLHC_Handler->Accel_Enable;
	LSM303DLHC_AccelWrite(LSM303DLHC_Handler, LSM303DLHC_CTRL_REG1_A, temp);
	if (LSM303DLHC_AccelRead(LSM303DLHC_Handler, LSM303DLHC_CTRL_REG1_A) != temp)
		return HAL_ERROR;


	/************************************************************************************/
	//Set CTRL_REG4_A
	temp = 0x00;
	temp = LSM303DLHC_Handler->Accel_Scale | LSM303DLHC_Handler->Accel_HR |
LSM303DLHC_Handler->Accel_Update_Mode;;
	LSM303DLHC_AccelWrite(LSM303DLHC_Handler, LSM303DLHC_CTRL_REG4_A, temp);
	if (LSM303DLHC_AccelRead(LSM303DLHC_Handler, LSM303DLHC_CTRL_REG4_A) != temp)
			return HAL_ERROR;


	/************************************************************************************/
	//Set CRA_REG_M
	temp = 0x00;
	temp = LSM303DLHC_Handler->Mag_DataRate | LSM303DLHC_Handler->Mag_Temp_Enable;
	LSM303DLHC_MagWrite(LSM303DLHC_Handler, LSM303DLHC_CRA_REG_M, temp);
	if (LSM303DLHC_MagRead(LSM303DLHC_Handler, LSM303DLHC_CRA_REG_M) != temp)
			return HAL_ERROR;

	/************************************************************************************/
	//Set CRB_REG_M
	temp = 0x00;
	temp = LSM303DLHC_Handler->Mag_Range;
	LSM303DLHC_MagWrite(LSM303DLHC_Handler, LSM303DLHC_CRB_REG_M, temp);
	if (LSM303DLHC_MagRead(LSM303DLHC_Handler, LSM303DLHC_CRB_REG_M) != temp)
			return HAL_ERROR;

	/************************************************************************************/
	//Set MR_REG_M
	temp = 0x00;
	temp = LSM303DLHC_Handler->Mag_Mode;
	LSM303DLHC_MagWrite(LSM303DLHC_Handler, LSM303DLHC_MR_REG_M, temp);
	if (LSM303DLHC_MagRead(LSM303DLHC_Handler, LSM303DLHC_MR_REG_M) != temp)
			return HAL_ERROR;

#if (LSM303_FREERTOS == 1)
	if(xTaskCreate(vTask_LSM303DLHC_GetAzimuth_GravIvclin, NULL, configMINIMAL_STACK_SIZE,
NULL, LSM303DLHC_TASK_PRIORITY, NULL) == errCOULD_NOT_ALLOCATE_REQUIRED_MEMORY)
		return HAL_ERROR;

	xQueue_LSM303DLHC_angles = xQueueCreate(1,sizeof(LSM303DLHC_Azimuth_GarvInkl_t));
	if(xQueue_LSM303DLHC_angles == NULL)
		return HAL_ERROR;
#endif

	return HAL_OK;
}

/**
  * @brief     Register data writing to Accel.
  * @param[in] LSM303DLHC_Handler Points to an LSM303DLHC_HandleTypeDef structure that
contains
  *                     the configuration information for the specified LSM303DLHC.
  * @param[in] RegisterAddr Register address.
  * @param[in] value Transferred value.
  * @return    none
```

```
  */
 static void LSM303DLHC_AccelWrite (LSM303DLHC_HandleTypeDef* LSM303DLHC_Handler, uint8_t
RegisterAddr, uint8_t value)
{
      HAL_I2C_Mem_Write(LSM303DLHC_Handler->I2C_handler, LSM303DLHC_ACCEL_ADDRESS,
(uint16_t)RegisterAddr, I2C_MEMADD_SIZE_8BIT, &value, 1, LSM_MAX_TIMEOUT);
}


/**
  * @brief     Read register from Accel.
  * @param[in] LSM303DLHC_Handler Points to an LSM303DLHC_HandleTypeDef structure that
contains
  *                    the configuration information for the specified LSM303DLHC.
  * @param[in] RegisterAddr Register address.
  * @return    Value of register.
  */
 static uint8_t LSM303DLHC_AccelRead (LSM303DLHC_HandleTypeDef* LSM303DLHC_Handler, uint8_t
RegisterAddr)
{
      uint8_t value=0;
      HAL_I2C_Mem_Read(LSM303DLHC_Handler->I2C_handler, LSM303DLHC_ACCEL_ADDRESS,
(uint16_t)RegisterAddr, I2C_MEMADD_SIZE_8BIT, &value, 1, LSM_MAX_TIMEOUT);
      return value;
}

/**
  * @brief     Get data accel.
  * @param[in] LSM303DLHC_Handler Points to an LSM303DLHC_HandleTypeDef structure that
contains
  *                    the configuration information for the specified LSM303DLHC.
  * @param[in] Data Points to an LSM303DLHC_Data_t structure that contains
  *                    the data from accel LSM303DLHC.
  * @return    none
  * @note      none
  */
void LSM303DLHC_AccelGetData (LSM303DLHC_HandleTypeDef* LSM303DLHC_Handler, LSM303DLHC_Data_t*
Data)
{
      uint8_t buffer[6] = {0,};
#if (LSM303_FREERTOS == 1)

      vTaskSuspendAll();
      HAL_I2C_Mem_Read(LSM303DLHC_Handler->I2C_handler, LSM303DLHC_ACCEL_ADDRESS,
LSM303DLHC_OUT_X_L_A|LSM303DLHC_MULTIPLE_FLAG, I2C_MEMADD_SIZE_8BIT, buffer, 6,
LSM_MAX_TIMEOUT);
      xTaskResumeAll();
#else
      HAL_I2C_Mem_Read(LSM303DLHC_Handler->I2C_handler, LSM303DLHC_ACCEL_ADDRESS,
LSM303DLHC_OUT_X_L_A|LSM303DLHC_MULTIPLE_FLAG, I2C_MEMADD_SIZE_8BIT, buffer, 6,
LSM_MAX_TIMEOUT);
#endif

      Data->data_A[0] = (int16_t)((buffer[1] << 8) | buffer[0]);
      Data->data_A[1] = (int16_t)((buffer[3] << 8) | buffer[2]);
      Data->data_A[2] = (int16_t)((buffer[5] << 8) | buffer[4]);

}

/**
  * @brief     Register data writing to Mag.
  * @param[in] LSM303DLHC_Handler Points to an LSM303DLHC_HandleTypeDef structure that
contains
  *                    the configuration information for the specified LSM303DLHC.
```

```
  * @param[in] RegisterAddr Register address.
  * @param[in] value Transferred value.
  * @return    none
  */
static void LSM303DLHC_MagWrite (LSM303DLHC_HandleTypeDef* LSM303DLHC_Handler, uint8_t
RegisterAddr, uint8_t value)
{
      HAL_I2C_Mem_Write(LSM303DLHC_Handler->I2C_handler, LSM303DLHC_MAG_ADDRESS,
(uint16_t)RegisterAddr, I2C_MEMADD_SIZE_8BIT, &value, 1, LSM_MAX_TIMEOUT);
}

/**
  * @brief     Read register from Mag.
  * @param[in] LSM303DLHC_Handler Points to an LSM303DLHC_HandleTypeDef structure that
contains
  *                     the configuration information for the specified LSM303DLHC.
  * @param[in] RegisterAddr Register address.
  * @return    Value of register.
  */
static uint8_t LSM303DLHC_MagRead (LSM303DLHC_HandleTypeDef* LSM303DLHC_Handler, uint8_t
RegisterAddr)
{
      uint8_t value=0;
      HAL_I2C_Mem_Read(LSM303DLHC_Handler->I2C_handler, LSM303DLHC_MAG_ADDRESS,
(uint16_t)RegisterAddr, I2C_MEMADD_SIZE_8BIT, &value, 1, LSM_MAX_TIMEOUT);
      return value;
}

/**
  * @brief     Get data mag.
  * @param[in] LSM303DLHC_Handler Points to an LSM303DLHC_HandleTypeDef structure that
contains
  *                     the configuration information for the specified LSM303DLHC.
  * @param[in] Data Points to an LSM303DLHC_Data_t structure that contains
  *                     the data from mag LSM303DLHC.
  * @return    none
  * @note      none
  */
void LSM303DLHC_MagGetData (LSM303DLHC_HandleTypeDef* LSM303DLHC_Handler, LSM303DLHC_Data_t*
Data)
{
      uint8_t buffer[6] = {0,};
#if (LSM303_FREERTOS == 1)
      vTaskSuspendAll();
      HAL_I2C_Mem_Read(LSM303DLHC_Handler->I2C_handler, LSM303DLHC_MAG_ADDRESS,
LSM303DLHC_OUT_X_H_M|LSM303DLHC_MULTIPLE_FLAG, I2C_MEMADD_SIZE_8BIT, buffer, 6,
LSM_MAX_TIMEOUT);
      xTaskResumeAll();

#else

      HAL_I2C_Mem_Read(LSM303DLHC_Handler->I2C_handler, LSM303DLHC_MAG_ADDRESS,
LSM303DLHC_OUT_X_H_M|LSM303DLHC_MULTIPLE_FLAG, I2C_MEMADD_SIZE_8BIT, buffer, 6,
LSM_MAX_TIMEOUT);

#endif
      Data->data_M[0] = (int16_t)((buffer[0] << 8) | buffer[1]);
      Data->data_M[1] = (int16_t)((buffer[4] << 8) | buffer[5]);
      Data->data_M[2] = (int16_t)((buffer[2] << 8) | buffer[3]);
}

/**
  * @brief     Get data accel.
```

```
  * @param[in] LSM303DLHC_Handler Points to an LSM303DLHC_HandleTypeDef structure that
contains
  *                    the configuration information for the specified LSM303DLHC.
  * @param[out] azimuth Points to an data azimuth
  * @param[out] grav_inclineh Points to an data grav_incline
  * @return    none
  * @note      none
  */
void LSM303DLHC_GetAzimuth_GravIvclin (LSM303DLHC_HandleTypeDef* LSM303DLHC_Handler, float*
azimuth, float* grav_incline)
{
      LSM303DLHC_Data_t Data;
      LSM303DLHC_MagGetData(LSM303DLHC_Handler, &Data);
      LSM303DLHC_AccelGetData(LSM303DLHC_Handler, &Data);

      Vector_3f_t vCalib_M;
      Vector_3f_t vCalib_A;
      MATRIX_Vect_3f_Dif(Data.data_A, CalibMatix_h_A, Data.data_A);
      MATRIX_Vect_3f_Dif(Data.data_M, CalibMatix_h_M, Data.data_M);
      MATRIX_Vect_3f_Transf(Data.data_A, CalibMatix_A_A, vCalib_A);
      MATRIX_Vect_3f_Transf(Data.data_M, CalibMatix_A_M, vCalib_M);

      Vector_3f_t vX, vY, vZ;
      MATRIX_Vect_3f_Reject(vCalib_M, vCalib_A, vX);
      MATRIX_Vect_3f_NotmalizeSelf(vX);
      MATRIX_Vect_3f_Notmalize(vCalib_A, vZ);

      MATRIX_Vect_3f_Cross(vX, vZ, vY);

      Matrix_3f_t M;
      MATRIX_Matrix_3f_SetRow(vX, M, 0);
      MATRIX_Matrix_3f_SetRow(vY, M, 1);
      MATRIX_Matrix_3f_SetRow(vZ, M, 2);

      Vector_3f_t vElure;
      MATRIX_Matrix_3f_GetEuler(M, vElure);

      float yaw = vElure[2] * CON_DEGREE;
      *azimuth = (yaw > 0) ? 360-yaw : -yaw;
      /*Calib*/
      if(*azimuth > 360)
      {
            *azimuth -= 360;
      }

      Vector_3f_t vZaxis = {0,0,1};
      *grav_incline = MATRIX_Vect_3f_Ang(vZ, vZaxis)*CON_DEGREE;

      LSM303DLHC_Filt_EMA(grav_incline, azimuth, 0.1);
}


/**
  * @brief     Implementation of the EMA filter
  * @param[in] grav_incline
  * @param[in] azimuth
  * @param[in] K
  * @return    none
  * @note      none
  */
static void LSM303DLHC_Filt_EMA (float* grav_incline, float* azimuth, float K)
{
      static float filt_grav_incline = 0;
      static float filt_azimuth = 0;
```

```
	filt_grav_incline += (*grav_incline - filt_grav_incline) * K;
	filt_azimuth += (*azimuth - filt_azimuth) * K;

	*grav_incline = filt_grav_incline;
	*azimuth = filt_azimuth;
}
#if (LSM303_FREERTOS == 1)
/**
  * @brief Task for the implementation of the sensor.
  */
static void vTask_LSM303DLHC_GetAzimuth_GravIvclin (void* pvParameter)
{
	LSM303DLHC_Data_t Data = {0,};
	Vector_3f_t vCalib_M = {0,};
	Vector_3f_t vCalib_A = {0,};
	Vector_3f_t vX = {0,};
	Vector_3f_t vY = {0,};
	Vector_3f_t vZ = {0,};
	Matrix_3f_t M = {0,};
	Vector_3f_t vElure = {0,};
	float yaw = 0;
	Vector_3f_t vZaxis = {0,0,1};
	LSM303DLHC_Azimuth_GarvInkl_t DataAngle = {0,};

	portTickType xLastWakeTime = xTaskGetTickCount();
	for(;;)
	{
		/*
		 * Refresh data.
		 */
		xQueueReceive(xQueue_LSM303DLHC_angles, &DataAngle, 0);
		/*
		 * Getting data.
		 */
		LSM303DLHC_MagGetData(&LSM303DLHC_Handle_main, &Data);
		LSM303DLHC_AccelGetData(&LSM303DLHC_Handle_main, &Data);
		/*
		 * Calibration.
		 */
		MATRIX_Vect_3f_Dif(Data.data_A, CalibMatix_h_A, Data.data_A);
		MATRIX_Vect_3f_Dif(Data.data_M, CalibMatix_h_M, Data.data_M);
		MATRIX_Vect_3f_Transf(Data.data_A, CalibMatix_A_A, vCalib_A);
		MATRIX_Vect_3f_Transf(Data.data_M, CalibMatix_A_M, vCalib_M);
		/*
		 * Getting two perpendicular vectors.
		 */
		MATRIX_Vect_3f_Reject(vCalib_M, vCalib_A, vX);
		MATRIX_Vect_3f_NotmalizeSelf(vX);
		MATRIX_Vect_3f_Notmalize(vCalib_A, vZ);
		/*
		 * Getting third perpendicular vectors.
		 */
		MATRIX_Vect_3f_Cross(vX, vZ, vY);
		/*
		 * Creation matrix.
		 */
		MATRIX_Matrix_3f_SetRow(vX, M, 0);
		MATRIX_Matrix_3f_SetRow(vY, M, 1);
		MATRIX_Matrix_3f_SetRow(vZ, M, 2);
		/*
		 * Getting angles.
		 */
		MATRIX_Matrix_3f_GetEuler(M, vElure);
		yaw = vElure[2] * CON_DEGREE;
```

```
            DataAngle.azimuth = (yaw > 0) ? 360-yaw : -yaw;
            DataAngle.grav_incline = MATRIX_Vect_3f_Ang(vZ, vZaxis)*CON_DEGREE;

            DataAngle.azimuth+= 40;
            if(DataAngle.azimuth > 360)
            {
                  DataAngle.azimuth -= 360;
            }
            /*
             * Filtration.
             */
            LSM303DLHC_Filt_EMA(&DataAngle.azimuth, &DataAngle.grav_incline, 0.1);

            xQueueSendToBack(xQueue_LSM303DLHC_angles, &DataAngle, 0);

            vTaskDelayUntil( &xLastWakeTime, ( LSM303DLHC_TASK_PERIOD / portTICK_RATE_MS ) );
      }
      vTaskDelete(NULL);
}
#endif

#if (LSM303_DEBUG == 1)

void LSM303DLHC_TransMagData (LSM303DLHC_HandleTypeDef* LSM303DLHC_Handler,
UART_HandleTypeDef* UART_Handle)
{
      LSM303DLHC_Data_t DataXYZ;
      uint8_t buffer[23];
      LSM303DLHC_MagGetData(LSM303DLHC_Handler, &DataXYZ);
      sprintf(buffer, "%6d;%6d;%6d;\r\n", (int)DataXYZ.data_M[0], (int)DataXYZ.data_M[1],
(int)DataXYZ.data_M[2]);
      HAL_UART_Transmit(UART_Handle, buffer, sizeof(buffer), 0xFF);
      HAL_Delay(100);
}

void LSM303DLHC_TransMagCalibData (LSM303DLHC_HandleTypeDef* LSM303DLHC_Handler,
UART_HandleTypeDef* UART_Handle)
{
      LSM303DLHC_Data_t DataXYZ;
      Vector_3f_t SendData;
      uint8_t buffer[23];
      LSM303DLHC_MagGetData(LSM303DLHC_Handler, &DataXYZ);
      MATRIX_Vect_3f_Dif(DataXYZ.data_M, CalibMatix_h_M, DataXYZ.data_M);
      MATRIX_Vect_3f_Transf(DataXYZ.data_M, CalibMatix_A_M, SendData);
      sprintf(buffer, "%6d;%6d;%6d;\r\n", (int)SendData[0], (int)SendData[1],
(int)SendData[2]);
      HAL_UART_Transmit(UART_Handle, buffer, sizeof(buffer), 0xFF);
      HAL_Delay(100);
}

void LSM303DLHC_TransAcellData (LSM303DLHC_HandleTypeDef* LSM303DLHC_Handler,
UART_HandleTypeDef* UART_Handle)
{
      LSM303DLHC_Data_t DataXYZ;
      uint8_t buffer[23];
      LSM303DLHC_AccelGetData(LSM303DLHC_Handler, &DataXYZ);
      sprintf(buffer, "%6d;%6d;%6d;\r\n", (int)DataXYZ.data_A[0], (int)DataXYZ.data_A[1],
(int)DataXYZ.data_A[2]);
      HAL_UART_Transmit(UART_Handle, buffer, sizeof(buffer), 0xFF);
      HAL_Delay(100);
}
void LSM303DLHC_TransAzimuth (LSM303DLHC_HandleTypeDef* LSM303DLHC_Handler,
UART_HandleTypeDef* UART_Handle)
{
```

```
	float azimuth, grav_incl;
	uint8_t buffer[23];
	LSM303DLHC_GetAzimuth_GravIvclin(LSM303DLHC_Handler, &azimuth, &grav_incl);
	float temp_azimuth = azimuth;
	//LSM303DLHC_Filt_EMA(&grav_incl, &azimuth, 0.1);
	sprintf(buffer, "%6d;%6d;%6d;\r\n", (int)temp_azimuth, (int)azimuth, (int)1);
	HAL_UART_Transmit(UART_Handle, buffer, sizeof(buffer), 0xFF);
	HAL_Delay(100);
}
#endif
```

6 Библиотека датчика мутности

Файл: turbidity.h

```
#ifndef INC_TURBIDITY_H_
#define INC_TURBIDITY_H_

#include "main.h"
#include "cmsis_os.h"

#define TURB_TASK_PRIOTITY		4
#define TURB_CHOSEN_ADC			hadc1
#define TURB_TASK_PERIOD		500
#define TURB_MAX_TURBIDITY		49

extern ADC_HandleTypeDef TURB_CHOSEN_ADC;
extern xQueueHandle xQueue_turb_percent_turbidity;

HAL_StatusTypeDef Turb_Init (void);
#endif /* INC_TURBIDITY_H_ */
```

Файл: turbidity.c

```
#include "turbidity.h"

static void vTaskTurb(void* pvParameter);
static void Turb_Filt_EMA (float* percent_turbidity,  float K);
/*
 * Creating a task and a queue.
 */
HAL_StatusTypeDef Turb_Init (void)
{
	if(xTaskCreate(vTaskTurb, NULL, configMINIMAL_STACK_SIZE, NULL, TURB_TASK_PRIOTITY,
NULL) == errCOULD_NOT_ALLOCATE_REQUIRED_MEMORY)
		return HAL_ERROR;

	xQueue_turb_percent_turbidity = xQueueCreate(1,sizeof(float));
	if(xQueue_turb_percent_turbidity == NULL)
		return HAL_ERROR;

	return HAL_OK;
}
/*
 * The task of obtaining turbidity sensor data using an ADC.
 */
static void vTaskTurb(void* pvParameter)
{
	TickType_t pxPreviousWakeTime = xTaskGetTickCount ();
	float percent_turbidity = 0.f;
```

```
	uint8_t turbidity = 0;

	for(;;)
	{
		xQueueReceive(xQueue_turb_percent_turbidity, &percent_turbidity, 0);

		HAL_ADC_Start(&TURB_CHOSEN_ADC);/*запустим ацп*/
		HAL_ADC_PollForConversion(&TURB_CHOSEN_ADC,100);/*100- таймаут, дожидаемся конца
преобразований*/
		turbidity = HAL_ADC_GetValue(&TURB_CHOSEN_ADC);
		HAL_ADC_Stop(&TURB_CHOSEN_ADC);

		percent_turbidity = (float)turbidity / ((float)TURB_MAX_TURBIDITY) * 100.f;
		Turb_Filt_EMA(&percent_turbidity, 0.33);

		xQueueSendToBack(xQueue_turb_percent_turbidity, &percent_turbidity, 0);

		vTaskDelayUntil(&pxPreviousWakeTime, TURB_TASK_PERIOD);
	}
	vTaskDelete(NULL);
}
/*
 * Implementation of EMA filter.
 */
static void Turb_Filt_EMA (float* percent_turbidity,  float K)
{
	static float filt_percent_turbidity = 0.f;

	filt_percent_turbidity += (*percent_turbidity - filt_percent_turbidity) * K;
	*percent_turbidity = filt_percent_turbidity;
}
```

7 Библиотека GSM-модуля

Файл: gsm.h

```
/*
 * gsm.h
 *
 *  Created on: 23 мар. 2023 г.
 *      Author: Mihail M.
 */

#ifndef INC_GSM_H_
#define INC_GSM_H_


#include "main.h"
#include "cmsis_os.h"


#define GSM_NUMBER                                  "+79515914568"
#define GSM_CHOSEN_UART                         huart1
#define GSM_RECIVE_BUFFER_SIZE                  300
#define GSM_SMS_BUFFER_SIZE                     300
#define GSM_SMS_SENDPERIOD_MS               3000
#define     GSM_TASK_PRIOTITY_SEND_SMS          2
#define     GSM_TASK_PRIOTITY_CHEK_RESP         6
#define     GSM_TASK_PRIOTITY_RECIVE_SMS  5

HAL_StatusTypeDef GSM_Init (void);

#endif /* INC_GSM_H_ */
```

Файл: gsm.c

```
/*
 * gsm.c
 *
 *  Created on: 23 мар. 2023 г.
 *      Author: Mihail M.
 */
#include <stdbool.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include "gsm.h"
#include "gps.h"
#include "lsm303dlhc.h"
#include "ds18b20.h"

xSemaphoreHandle xBinarySemaphore_GSM_send_data;
xSemaphoreHandle xBinarySemaphore_GSM_send_sms;
xSemaphoreHandle xBinarySemaphore_GSM_recive_data;
xSemaphoreHandle xBinarySemaphore_GSM_recive_sms;
uint8_t recive_buffer[GSM_RECIVE_BUFFER_SIZE] = {0,};

extern xQueueHandle xQueue_GSM_set_mode;
extern xQueueHandle xQueue_LSM303DLHC_angles;
extern xQueueHandle xQueue_GPS_data;
extern xQueueHandle xQueue_Ds18b20_temper;
extern xQueueHandle xQueue_turb_percent_turbidity;
extern UART_HandleTypeDef GSM_CHOSEN_UART;

static void send_data(uint8_t *data, uint16_t size);
static HAL_StatusTypeDef send_sms(const char *phone_number, uint8_t *sms_text);
static void SMS_processing(uint8_t* Text);
static void SMS_creation(uint8_t* Text);
static void vTask_GSM_Chek_Response(void* pvParameter);
static void vTask_GSM_Recive_SMS(void* pvParameter);
static void vTask_GSM_Send_SMS(void* pvParameter);
/*
 * Initialization of the tasks of semaphores and queues.
 */
HAL_StatusTypeDef GSM_Init (void)
{
	if(xTaskCreate(vTask_GSM_Chek_Response, NULL, configMINIMAL_STACK_SIZE*2, NULL,
GSM_TASK_PRIOTITY_CHEK_RESP, NULL) == errCOULD_NOT_ALLOCATE_REQUIRED_MEMORY)
		return HAL_ERROR;
	if(xTaskCreate(vTask_GSM_Recive_SMS, NULL, configMINIMAL_STACK_SIZE*2, NULL,
GSM_TASK_PRIOTITY_RECIVE_SMS, NULL) == errCOULD_NOT_ALLOCATE_REQUIRED_MEMORY)
		return HAL_ERROR;
	if(xTaskCreate(vTask_GSM_Send_SMS, NULL, configMINIMAL_STACK_SIZE*2, NULL,
GSM_TASK_PRIOTITY_SEND_SMS, NULL) == errCOULD_NOT_ALLOCATE_REQUIRED_MEMORY)
		return HAL_ERROR;

	vSemaphoreCreateBinary(xBinarySemaphore_GSM_send_data);
	if(xBinarySemaphore_GSM_send_data == NULL)
		return HAL_ERROR;
	xSemaphoreTake(xBinarySemaphore_GSM_send_data, portMAX_DELAY);

	vSemaphoreCreateBinary(xBinarySemaphore_GSM_send_sms);
	if(xBinarySemaphore_GSM_send_sms == NULL)
		return HAL_ERROR;
	xSemaphoreTake(xBinarySemaphore_GSM_send_sms, portMAX_DELAY);
```

```
	vSemaphoreCreateBinary(xBinarySemaphore_GSM_recive_data);
	if(xBinarySemaphore_GSM_recive_data == NULL)
		return HAL_ERROR;
	xSemaphoreTake(xBinarySemaphore_GSM_recive_data, portMAX_DELAY);

	vSemaphoreCreateBinary(xBinarySemaphore_GSM_recive_sms);
	if(xBinarySemaphore_GSM_recive_sms == NULL)
		return HAL_ERROR;
	xSemaphoreTake(xBinarySemaphore_GSM_recive_sms, portMAX_DELAY);

	xQueue_GSM_set_mode = xQueueCreate(1,sizeof(uint8_t));
	if(xQueue_GSM_set_mode == NULL)
		return HAL_ERROR;

	/*
	 * Start receiving data.
	 */
	HAL_UARTEx_ReceiveToIdle_IT(&GSM_CHOSEN_UART, recive_buffer, GSM_RECIVE_BUFFER_SIZE);


	return HAL_OK;
}

/*
 * Sending UART data using DMA.
 */
static void send_data(uint8_t *data, uint16_t size)
{
	HAL_UART_Transmit_DMA(&GSM_CHOSEN_UART, data, size);
	xSemaphoreTake(xBinarySemaphore_GSM_send_data, portMAX_DELAY);
	asm("NOP");
}
/*
 * Sending data tracking.
 */
void HAL_UART_TxCpltCallback(UART_HandleTypeDef *huart)
{
	if (huart == &GSM_CHOSEN_UART)
	{
		xSemaphoreGiveFromISR(xBinarySemaphore_GSM_send_data, NULL);
	}
}

/*
 * Response processing task.
 */
static void vTask_GSM_Chek_Response(void* pvParameter)
{
	/*
	 * Setting GSM
	 */
	HAL_UART_Transmit_IT(&GSM_CHOSEN_UART, (uint8_t *)"AT\r\n", 4); /*Starting string*/
	vTaskDelay(500);

	HAL_UART_Transmit_IT(&GSM_CHOSEN_UART, (uint8_t *)"AT+CMEE=2\r\n", 11); /*Advanced level
of error reporting*/
	vTaskDelay(500);

	HAL_UART_Transmit_IT(&GSM_CHOSEN_UART, (uint8_t *)"AT+CMGF=1\r\n", 11); /*Text message
mode*/
	vTaskDelay(500);

	HAL_UART_Transmit_IT(&GSM_CHOSEN_UART, (uint8_t *)"ATE0\r\n", 6); /*Disable echo mode*/
	vTaskDelay(500);
```

```
	HAL_UART_Transmit_IT(&GSM_CHOSEN_UART, (uint8_t *)"AT+CNMI=2,2,0,0,0\r\n", 19);
/*Automatic notification of a new incoming message.*/
	vTaskDelay(500);

	HAL_UART_Transmit_IT(&GSM_CHOSEN_UART, (uint8_t *)"AT+CMGDA=\"DEL ALL\"\r\n", 19);
/*Delete all massages.*/
	vTaskDelay(500);

	xSemaphoreTake(xBinarySemaphore_GSM_send_data, portMAX_DELAY);
	for(;;)
	{
		xSemaphoreTake(xBinarySemaphore_GSM_recive_data, portMAX_DELAY);
		if(strstr((char*)recive_buffer ,"+CMGS") != NULL && strstr((char*)recive_buffer
,"OK") != NULL) /*Tracking the success of sending a message.*/
			xSemaphoreGive(xBinarySemaphore_GSM_send_sms);
		else if(strstr((char*)recive_buffer ,"+CMT") != NULL) /*Tracking the arrival of a
new message.*/
			xSemaphoreGive(xBinarySemaphore_GSM_recive_sms);
		else
			memset(recive_buffer, '\0', GSM_RECIVE_BUFFER_SIZE);
	}
	vTaskDelete(NULL);
}
/*
 * The task of receiving and processing new messages.
 */
static void vTask_GSM_Recive_SMS(void* pvParameter)
{
	uint8_t sms_recive_buffer[GSM_RECIVE_BUFFER_SIZE] = {0,};
	for(;;)
	{
		xSemaphoreTake(xBinarySemaphore_GSM_recive_sms, portMAX_DELAY);
		strcpy((char*)sms_recive_buffer, (char*)recive_buffer);
		SMS_processing(sms_recive_buffer);
		memset(recive_buffer, '\0', GSM_RECIVE_BUFFER_SIZE);
	}
	vTaskDelete(NULL);
}
/*
 * Processing new massages.
 */
static void SMS_processing(uint8_t* Text)
{
	char* temp = NULL;
	uint8_t  mode = 0;
	temp = strstr((char*)Text,"Mode:");
	temp = strstr((char*)temp,":");
	mode = *(temp+1) - '0';
	xQueueSendToBack(xQueue_GSM_set_mode, &mode, 0);
}
/*
 * Tracking receive data.
 */
void HAL_UARTEx_RxEventCallback(UART_HandleTypeDef *huart, uint16_t Size)
{
	if(huart == &GSM_CHOSEN_UART)
	{
		xSemaphoreGiveFromISR(xBinarySemaphore_GSM_recive_data, NULL);
		HAL_UARTEx_ReceiveToIdle_IT(&GSM_CHOSEN_UART, recive_buffer,
GSM_RECIVE_BUFFER_SIZE);
	}
}
/*
```

```
 * The task of creating and sending SMS messages.
 */
static void vTask_GSM_Send_SMS(void* pvParameter)
{
	uint8_t sms_text[GSM_SMS_BUFFER_SIZE] = {'\0',};
	TickType_t pxPreviousWakeTime = xTaskGetTickCount ();
	for(;;)
	{
		SMS_creation(sms_text);
		send_sms(GSM_NUMBER, sms_text);
		memset(sms_text, '\0', GSM_SMS_BUFFER_SIZE);
		vTaskDelayUntil(&pxPreviousWakeTime, GSM_SMS_SENDPERIOD_MS);
	}
	vTaskDelete(NULL);
}
/*
 * Send SMS using AT commands
 */
static HAL_StatusTypeDef send_sms(const char *phone_number, uint8_t *sms_text)
{
	uint8_t temp_buffer_size = 100;
	char temp_buffer[100];
	memset(temp_buffer,'\0',temp_buffer_size);

	sprintf(temp_buffer, "AT+CMGS=\"%s\"\r\n", phone_number);
	send_data((uint8_t*)temp_buffer, strlen(temp_buffer));
	vTaskDelay(100/portTICK_RATE_MS);

	send_data(sms_text, (uint16_t)strlen((char*)sms_text));
	vTaskDelay(100/portTICK_RATE_MS);

	send_data((uint8_t *)"\r\n\x1A\r\n", 5);
	vTaskDelay(100/portTICK_RATE_MS);

	if(xSemaphoreTake(xBinarySemaphore_GSM_send_sms, 1000/portTICK_RATE_MS) != pdTRUE)
	{
		memset(recive_buffer, '\0', GSM_RECIVE_BUFFER_SIZE);
		return HAL_ERROR;
	}

	memset(recive_buffer, '\0', GSM_RECIVE_BUFFER_SIZE);
	return HAL_OK;
}
/*
 * Creating an SMS message using the data received from sensors.
 */
static void SMS_creation(uint8_t* Text)
{

	float percent_turbidity =0.f;
	uint16_t temp = 0;
	GPS_Data_t GPS_Data = {0, .latitude_direction = '0', .longitude_direction  = '0'};
	LSM303DLHC_Azimuth_GarvInkl_t LSM303DLHC_Azimuth_GarvInkl = {0,};
	Ds18b20SensorQu_t temp_sensor = {0,};

	xQueueReceive(xQueue_GPS_data, &GPS_Data, 30000 / portTICK_RATE_MS);
	xQueueReceive(xQueue_Ds18b20_temper, &temp_sensor, 30000 / portTICK_RATE_MS);
	xQueueReceive(xQueue_LSM303DLHC_angles, &LSM303DLHC_Azimuth_GarvInkl, 30000 /
portTICK_RATE_MS);
	xQueueReceive(xQueue_turb_percent_turbidity, &percent_turbidity, 30000 /
portTICK_RATE_MS);

	temp += sprintf((char*)Text+temp, "Azimuth: %d\r\n",(int)
LSM303DLHC_Azimuth_GarvInkl.azimuth);
```

```
      temp += sprintf((char*)Text+temp, "Angle of inclination: %d\r\n",(int)
LSM303DLHC_Azimuth_GarvInkl.grav_incline);
      temp += sprintf((char*)Text+temp, "Data fix: %d\r\n", (int)GPS_Data.data_fix);
      temp += sprintf((char*)Text+temp, "Date: %d.%d.%d\r\n", (int)GPS_Data.day,
(int)GPS_Data.month, (int)GPS_Data.year);
      temp += sprintf((char*)Text+temp, "Time: %d:%d\r\n", (int)GPS_Data.hours,
(int)GPS_Data.minutes);
      temp += sprintf((char*)Text+temp, "Latitude: %d.%d\r\n", (int)GPS_Data.latitude,
(int)((GPS_Data.latitude - (int)GPS_Data.latitude)*100000));
      temp += sprintf((char*)Text+temp, "Latitude direction: %c\r\n",
GPS_Data.latitude_direction);
      temp += sprintf((char*)Text+temp, "Longitude: %d.%d\r\n", (int)GPS_Data.longitude,
(int)((GPS_Data.longitude - (int)GPS_Data.longitude)*100000));
      temp += sprintf((char*)Text+temp, "Longitude direction: %c\r\n",
GPS_Data.longitude_direction);
      temp += sprintf((char*)Text+temp, "Temp1: %d.%d\r\n", (int)temp_sensor.temp_sensor1,
abs((int)((temp_sensor.temp_sensor1 - (int)temp_sensor.temp_sensor1)*10)));
      temp += sprintf((char*)Text+temp, "Temp2: %d.%d\r\n", (int)temp_sensor.temp_sensor2,
abs((int)((temp_sensor.temp_sensor2 - (int)temp_sensor.temp_sensor2)*10)));
      temp += sprintf((char*)Text+temp, "Turbidity percentage: %d\r\n", (int)
percent_turbidity);
```

ПРИЛОЖЕНИЕ Б

(Обязательное)

ФЮРА.436431.001 Э3

ФЮРА.468213.001 Э3

Перв. примен. ФЮРА.468213.001

Справ.№

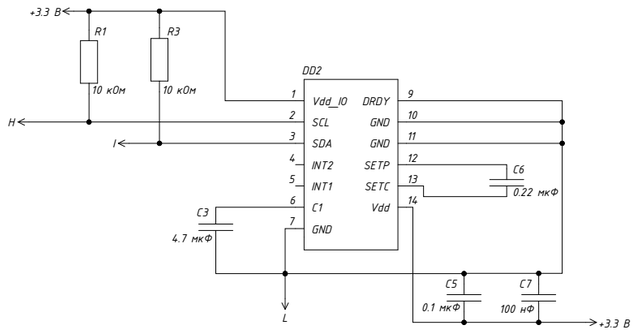
Подп. и дата

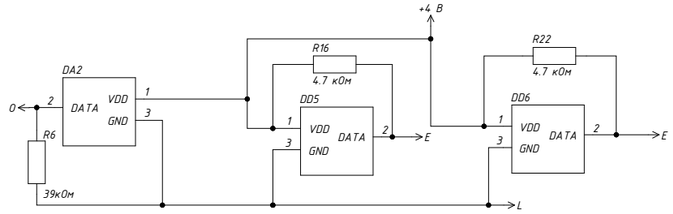
Инв.№ дубл.

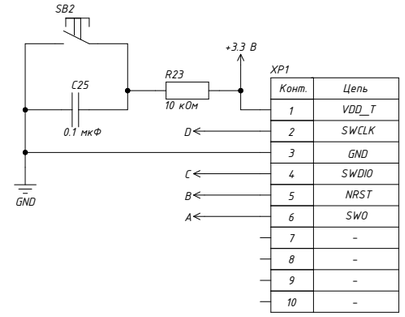
Взам.инв.№

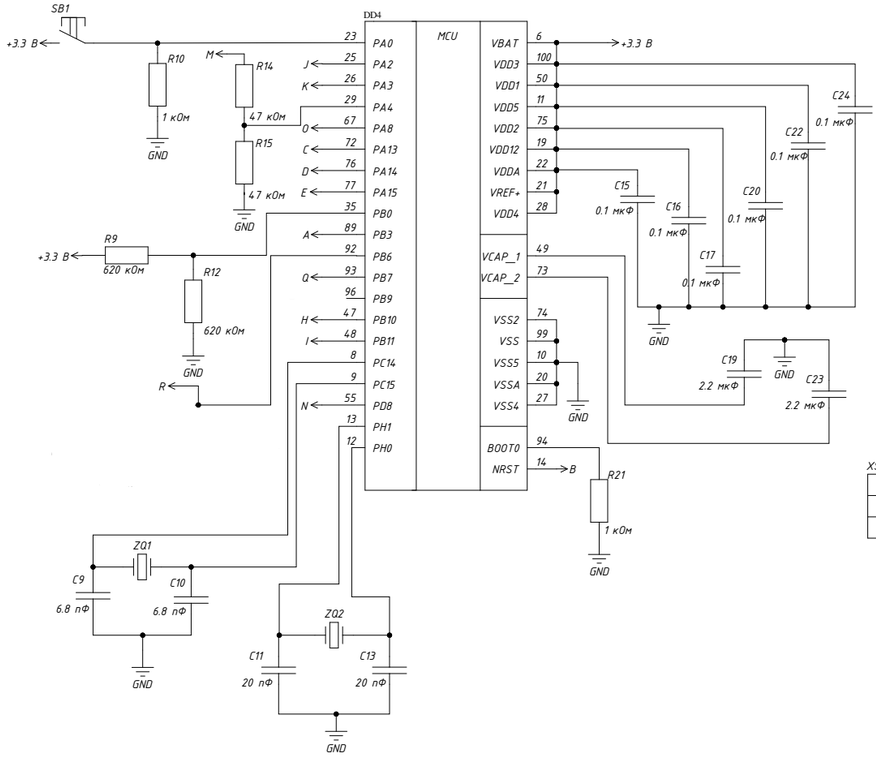
Подп.и дата

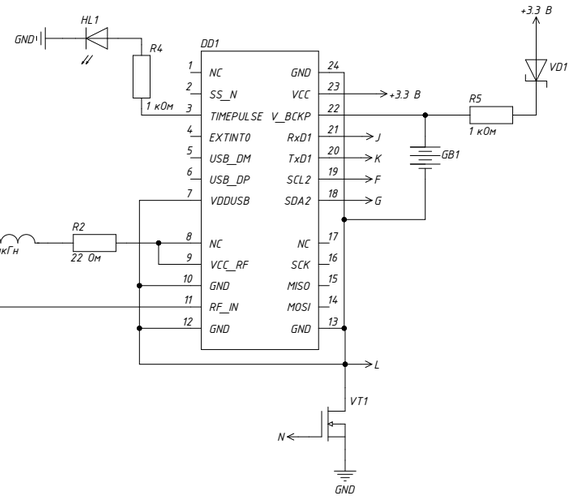
Инв.№ подл.

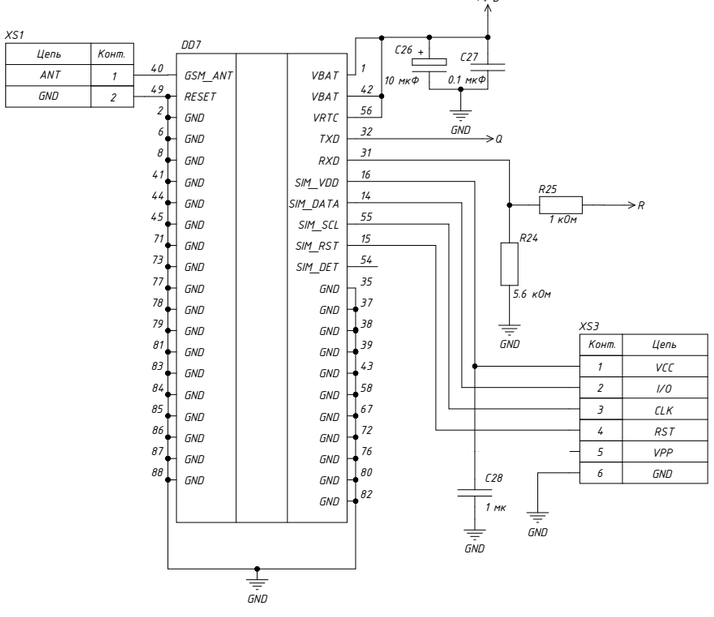


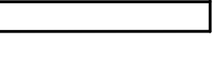












Изм.	Лист	№ докум.	Подп.	Дата
Разраб.		Мальцев		
Пров.		Солдатов		
Т.контр.				
Нач.сект.				
Н.контр.		Дикман		
Утв.				

ФЮРА.468213.001 Э3

Система автоматизации навигации речного судоходства

СХЕМА ЭЛЕКТРИЧЕСКАЯ ПРИНЦИПИАЛЬНАЯ

Лит.	Масштаб	Масса
У	1:1	

Лист 1 Листов 1

ТПУ ИШНКБ группа 1А91

Формат А2

ПРИЛОЖЕНИЕ В

(Обязательное)

ФЮРА.436431.001 ПЭ3

Поз. обозначение	Наименование	Кол.	Примечание.
	Конденсаторы		
C1,C5	0.1 мкФ ±10% 25 В	2	SMD 0603
C2,C4,C8	0.1 мкФ ±10% 25 В	3	SMD 0603
C3	4.7 мкФ ±10% 25 В	1	SMD 0603
C6	0.22 мкФ ±10% 10 В	1	SMD 0603
C7	100 нФ ±10% 16 В	1	SMD 0603
C9,C10	6.8 пФ ±10% 50 В	2	SMD 0603
C11,C13	20 пФ ±10% 50 В	2	SMD 0603
C12	ECAP 1 мкФ ±20% 25 В SMD	1	
C14	ECAP 100 мкФ ±20% 25 В	1	
C15,C16	0.1 мкФ ±10% 16 В	2	SMD 0603
C17,C20	0.1 мкФ ±10% 16 В	2	SMD 0603
C18	560 пФ ±10% 50 В	1	SMD 0603
C19,C23	2.2 мкФ ±10% 16 В	2	SMD 0603
C21	ECAP 330 мкФ ±20% 25 В	1	
C22,C24	0.1 мкФ ±10% 16 В	2	SMD 0603
C25	0.1 мкФ ±10% 16 В	1	SMD 0603
C26	ECAP 10 мкФ ±20% 16 В	1	
C27	0.1 мкФ ±10% 16 В	1	SMD 0603
	Микросхемы		
DA1	LM317BTG	1	SOT32
DA2	TS-300B	1	
DA3	MC34063ABN	1	DIP-8
DD1	NEO 6M	1	SMD-28

Подп. и дата | Инв. № дубл. | Взам. инв. № | Подп. и дата | Инв. № подл

Изм	Лист	№ докум	Подпись	Дата
Разраб.		Мальцев		
Пров.		Солдатов		
Н.контр		Дикман		
Утв.				

ФЮРА.436537.001ПЭ3

Система автоматизации навигации речного судоходства

Перечень элементов

Литера	Лист	Листов
У	1	4

ИШНКБ ТПУ гр. 1А91

Копировал

Формат А4

Поз. обозначение	Наименование	Кол.	Примечание.
DD2	LSM303DHLC	1	LGA-14
DD3	24AA32A	1	SOIC-8
DD4	STM32F407	1	LQFP-100
DD5,DD6	DS18B20	2	TO-92
DD7	SIM800L	1	LQFP-101
	Генераторы, источники питания		
G1	MS621FE-FL11E	1	QFP-128
	Светодиоды		
HL1	GNL-3012ED 20 мА	1	
	Катушки индуктивности		
L1	22 мкГн	1	SMD 1210
L2	120 мкГн	1	6,5x10,5
	Резисторы		
R1,R3	10 кОм ±5% 0.062 Вт	2	SMD 0402
R2	22 Ом ±5% 0.062 Вт	1	SMD 0402
R4,R5,R10	1 кОм ±5% 0.062 Вт	3	SMD 0402
R6	39 кОм ±5% 0.062 Вт	1	SMD 0402
R7,R8	4.7 кОм ±5% 0.062 Вт	2	SMD 0402
R9,R12	620 кОм ±5% 0.062 Вт	2	SMD 0403
R11	100 кОм ±5% 0.5 Вт	1	подстроечный 3296W
R13	100 Ом ±5% 0.062 Вт	1	SMD 0404
R14,R15	47 кОм ±5% 0.062 Вт	2	SMD 0404
R16,R22	4.7 кОм ±5% 0.062 Вт	2	SMD 0404

Подп. и дата | Инв. № дубл. | Взам. инв. № | Подп. и дата | Инв. № подл.

Поз. обозначение	Наименование	Кол.	Примечание.
R17	180 Ом ±5% 0.25 Вт	1	SMD 1206
R18	0.11 Ом ±5% 2 Вт	1	MF-2
R19	1.2 кОм ±5% 0.062 Вт	1	SMD 0402
R20	2.7 кОм ±5% 0.062 Вт	1	SMD 0402
R21	1 кОм ±5% 0.062 Вт	1	SMD 0402
R23	10 кОм ±5% 0.062 Вт	1	SMD 0402
R24	5.6 кОм ±5% 0.062 Вт	1	SMD 0402
R25	1 кОм ±5% 0.062 Вт	1	SMD 0402
	Выключатели		
SB1,SB2	1-1437565-7 24 В 0.05 А	2	
	Диоды		
VD1	1N5817 20 В 1 А	1	DO-41
VD2	BYV10-40 40 В 1 А	1	TO-220AB
	Транзисторы		
VT1	IRLL024NTR 55 В 3.1 А	1	SOT-223
	Антенны		
WA1	ANT-868-CPA	1	
	Разъёмы		
XP1,XP2	10-08-1071	2	
XS1	20579-001E	1	
XS2	5033981892 10 В 0.5 А	1	

Подп. и дата | Инв. № дубл. | Взам. инв. № | Подп. и дата | Инв. № подл.

Поз. обозначение	Наименование	Кол.	Примечание.
XS3	282837-2 250 В 13.5 А	1	
	Резонаторы		
ZQ1	МС-306 32.768 кГц	1	
ZQ2	NX5032GA 8 МГц	1	

Подп. и дата | Инв. № дубл. | Взам. инв. № | Подп. и дата | Инв. № подл.

Изм.	Лист	№ докум.	Подпись	Дата	ФЮРА.436537.001ПЭ3	Лист 4