МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ ГИДРОМЕТЕОРОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра Морские информационные системы

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

(Магистр)

На тему:Разработка макета малогабаритного подводного аппарата дистанционного контроля корпуса судна в ледовых условиях. Программный комплекс системы управления малогабаритным подводным аппаратом

Исполнитель Спасский М.А

(фамилия, имя, отчество)

Руководитель профессор каф. МИС д.т.н

(ученая степень, ученое звание)

Завгородний Владимир Николаевич

(фамилия, имя, отчество)

«К защите допускаю»

Заведующий кафедрой

(подпись)

доктор технических наук, профессор

(ученая степень, ученое звание)

Сикарев Игорь Александрович

(фамилия, имя, отчество)

«08» 06 2022 г.

Санкт–Петербург

2022

РЕФЕРАТ

Магистерская диссертация: 85с., 61 рис., 34 табл., 18 источника литературы, 3 приложения.

Ключевые слова: ПРОГРАММИРОВАНИЕ, ЯЗЫК ПРОГРАММИРОВАНИЯ, NMEA, ГРАФИЧЕСКИЙ ИНТРЕЙФЕС.

Объект исследования — макет системы управления малогабаритным подводным аппаратом для дистанционного контроля корпуса судна в ледовых условиях.

Предмет исследования — Программное обеспечение макета системы управления малогабаритным подводным аппаратом для дистанционного контроля корпуса судна в ледовых условиях.

Цель работы: Разработка программного обеспечения макета комплекса системы управления малогабаритным подводным аппаратом для дистанционного контроля корпуса судна в ледовых условиях

В магистерской диссертации проводится анализ: компонентов из которых состоит малогабаритный подводный аппарат, анализ состава сообщений NMEA

В результате: Результатом работы стал макет графического интереса, который необходим для дистанционного управления малогабаритным подводным аппаратом, а так же для дистанционного контроля корпуса судна в ледовых условиях.

Оглавление

ВВЕДЕНИЕ

В современном мире цифровых технологий, прогресс не стоит на месте. Появляются новые задачи и работы, многие из которых несут высокий риск для человеческого здоровья, а иногда и для жизни. Именно из-за работ с таким высоким риском, человечество придумало механизмы, которые зачастую способны существенно облегчить жизнь человеку, а иногда и вовсе заменить его или. Сначала механизмы были простые: блоки, рычаги, вороты, затем последовали более сложные, помогающие человеку не только с физическим трудом, но и умственным, например, счёты и последовавшие за ними простейшие калькуляторы, способные на тот момент времени только на сложение и вычитание. Следом за ними начали появляться более продвинутые артефакты, позволявшие передвигать тяжёлые грузы на более далёкие расстояния, чем человек мог донести или поднять, позволявшие проложить маршрут в тех местах, где сам человек не мог пройти. И так , шаг за шагом , наука позволила создавать дистанционно-оперируемые механизмы .

Я хотел бы привести в пример один из подобных механизмов , который , в моём понимании , должен существенно упростить такую сложную и довольно опасную профессию как аквалангист . Зачастую при подозрениях на образование повреждений днища судна, необходимо в максимально короткие сроки устроить проверку и при необходимости начать ремонт. Самый надёжный вариант – поставить судно в док и устроить полную проверку. Однако, тем судам, что находятся в окружении льдов подобный вариант не подойдёт, так как зачастую, ближайший порт может находится в сотнях морских миль, что исключает всякую возможность на профессиональное обслуживание. Следовательно, осмотр придётся делать

прямо на месте, во льдах.

Данный дипломный проект посвящен разработке макета малогабаритного подводного аппарата дистанционного контроля корпуса судна в ледовых условиях, а именно его программный комплекс системы управления малогабаритным подводным аппаратом для работы в трудно доступных местах с ограниченным пространством, с тяжёлыми погодными условиями связанными с отрицательной температурой воды и возможно низкой температурой воздуха. В качестве платформы для разработки макета системы управления использовалась среда для программирования “Visual Studio”, использовался язык программирования C#?, так же для программирования органов передвижения использовалась среда программирования “Arduino IDE” и язык программирования C++. Основной плато при создании макета послужила Arduino Mega 3 ревизии , данная плата разработана на базе микроконтроллера ATmega2560.

Актуальность работы: Актуальность данного проекта определена необходимостью визуального контроля за подводным объекта, который может быть затруднен труднодоступностью для человека поэтому в данной работе был разработан комплекс системы управления малогабаритным подводным аппаратом для дистанционного контроля корпуса судна в ледовых условиях который включает в себя разработку графического интерфейса, программное обеспечение для управления двигателями при осуществлении маневрирования.

Объект исследования — Программно-аппаратный комплекс использованный для создания макета комплекса системы управления малогабаритным подводным аппаратом.

Предмет исследования — Структура и состав программно-аппаратного комплекса использованного для создания макета комплекса системы управления малогабаритным подводным аппаратом.

Цель исследования — Разработка программного обеспечения макета комплекса системы управления малогабаритным подводным аппаратом для дистанционного контроля корпуса судна в ледовых условиях

Прикладная задача — Разработка макета для управления малогабаритным подводным аппаратом.

Задачи исследовательской работы:

Анализ элементной базы и выбор среды разработки.

Разработка программного обеспечения управления двигателями:

- при всплытии;
- при погружении;
- при горизонтальном перемещении.

Разработка интерфейсов :

- приема и передачи данных с датчика GPS ;
- макета графического интерфейса для обработки данных от малогабаритного подводного аппарата.

Новизна выпускной квалификационной работы заключается в …

Теоретическая значимость: данную работу можно использовать для улучшения ориентирования малогабаритных аппаратов в условиях Арктики при осуществлении дистанционного контроля, так же данную работу можно использовать для уже доступных дистанционных малогабаритных подводных аппаратов с целью добавления системы отслеживания малогабаритного подводного аппарата.

Практическая ценность исследования: Данная работа предоставляет возможность более простого управления подводным аппаратом при

исследовании подводной части судна в условиях Арктики, без привлечения дополнительных человеческих ресурсов, уменьшение времени, которое может быть затрачено на осуществление контроля подводной части корабля.

Выпускная квалификационная работа состоит из: титульного листа, реферата, содержания, введения, основной части, заключения, списка используемых в работе литературных источников, приложений.

ГЛАВА 1. РАЗРАБОТКА СТРУКТУРНОЙ СХЕМЫ ПОДВОДНОГО АППАРАТА, АНАЛИЗ СУЩЕСТВУЮЩЕЙ КОМПОНЕНТНОЙ БАЗЫ И ВЫБОР КОМПЛЕКТУЮЩИХ

1.1. Разработка структурной схемы подводного аппарата

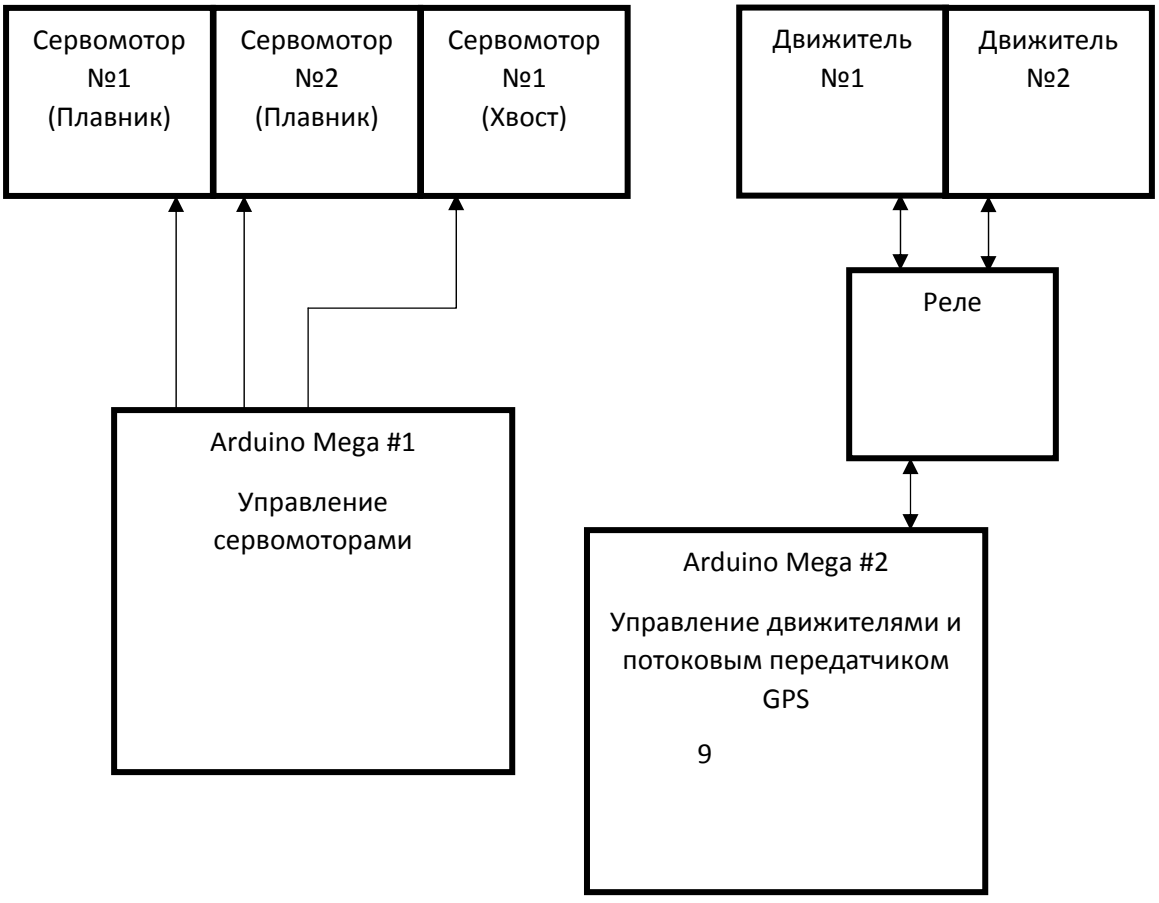
В современном мире, где продвинутые технологий не стоят на месте, подводные роботы обретают широкое использование в мире для выполнения всевозможных работ под водой. Они могут выполнять такие задачи как: поиск, инспекция, ремонт, исследование и так далее. Использование роботов под водой является непростой задачей, поскольку имеется риск связанный с их эксплуатацией в неизвестной подводной среде, с неизвестной навигационной обстановкой, так же роботам приходится работать под воздействием внешних воздействий, риски так же связаны с изменением их габаритных качеств, а так же с изменением гидродинамических параметров в ходе выполнения поставленной задачи в возможно агрессивной среде. Довольно крупной проблемой является задача автоматизации процесса управления. Разрабатываемый макет малогабаритного подводного аппарата предназначен в первую очередь для осуществления посково-иследовательных работ на относительно небольшой глубине, куда порой невозможно добраться человеку из-за трудно доступности места, отсутствия освещенности или мало

освещенности , перепадами температур воды и воздуха, а так же из-за других человеческих рисков для здоровья. Перед достижением заданной цели , необходимо начать с изучения структурной схемы которая представлена на рисунке 1.1. Данная схема создана, чтобы предоставить читателю примерное понимание о структуре работы нашего аппарата и о взаимодействии его компонентов друг с другом.

При последующем изучении и выборе и замене всех недостающих компонентов для создания рабочей базы, которая является основополагающей частью при создании макета малогабаритного подводного аппарата следует учитывать следующие пункты :

– малогабаритный подводный аппарат должен иметь модульную структуру, которая в свою очередь дает возможность замены сломанных или нерабочих элементов системы такие как: двигатель, плавники, корпус или же другие навесные элементы.

– аппарат должен иметь видеокамеру, для фиксирования повреждений подводных сооружений.



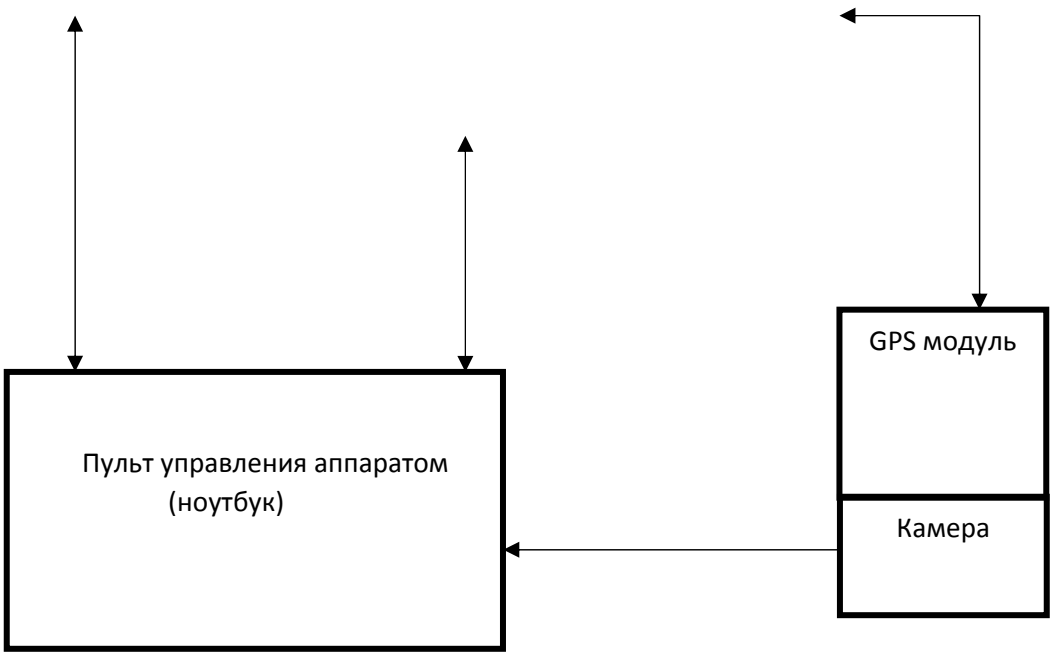


Рисунок 1.1 – структурная схема подводного аппарата

-аппарат должен иметь достаточно длинный кабель для передачи потоковой видеозаписи на пульт управления

- должны присутствовать фонари, мощности которых достаточно для освещения подводной части корабля

-аппарату необходима возможность подключения к GPS, для того, чтобы пользователь мог обнаружить местонахождение робота в случае непредвиденного течения или подобных неожиданных обстоятельств.

- батарея должна быть легкозаменяемой (как и другие компоненты), а также должна выдерживать все технические потребности аппарата во время работы.

1.2. Анализ комплектующих отечественного производства

Основная плата управления является “фундаментом” при создании макета малогабаритного подводного аппарата, так с её помощью можно

напрямую управлять всеми подключенными к ней модулями. Именно поэтому при разработке макета, выбор комплектующих мы начали с неё

Изначально в качестве платы управления решено было выбрать отечественный аналог компании “Arduino” , было решено выбрать один из отечественных аналогов продукции компании «Arduino», если быть точнее то от Петербуржского представителя электроники “Искра” ,а именно их платы серии “Iskra”. Выбор данного производителя позволил бы сократить время ожидания поставок интересующих нас компонентов, поскольку платы серии “Arduino” поставляются только из-за границы. В ходе изучения продукции отечественного производителя были выбраны 6 микроконтроллеров, такие как:

Iskra JS
Iskra JS mini
Iskra Neo
Iskra Mini
Iskra Mega
IskraN pro
Iskra Uno

Большая часть из вышеперечисленных плат не подходит для создания макета малогабаритного подводного аппарата.

Два микроконтроллера, первые из списка возможных вариантов «Iskra JS» и «Iskra JS mini» обладают такими техническими параметрами:

Основные характеристики «Iskra JS» указаны в приложении А

Основные характеристики «Iskra JS mini» Указаны в приложении Б

Программно-аппаратная часть платы написана на языке программирования “Javascript”, а не на “C++”, данный язык программирования имеет относительно не большую группу людей использующих его , не подходит для микроконтроллера поскольку при запуски сторонних приложении и при их отладке используется слишком

большое количество памяти, так же увеличивается количество программного кода необходимого для запуска этих приложений, не говоря уже о том , что данные платы не имеют заранее установленных расширений для подключения сторонних устройств в, что подразумевает «кастомное» (написание вручную) написание долгой версии кода для каждого модуля , что может потребовать от пары недель , до пары месяцев долгой кропотливой работы для каждого модуля , что указан у нас в чертеже. Так же хотелось бы отметить что , мой коллега предупредил меня о том, что при программировании данной платы используется язык “Javascript”, из этого следует, что эта плата нам не подходит, поскольку необходимо потратить гораздо больше времени на изучение данного языка, нежели чем на “C++” так как данный язык изучается со школьной программы. Микропроцессор «Iskra JS mini» не имеет выходов SPI, которые необходимы для подключения и управления сервмоторами . Не говоря уже о том , что данная плата имела отрицательную репутацию среди разработчиков за дурную тенденцию « слетающей» прошивки , что означало , что любое системное ПО (включая встроенное заводом изготовителем) могло просто перестать на ней работать в любой момент или же из-за некачественной переадресовки пакетов могло сжечь процессор и/или порты, делая плату полезной лишь на уровне пресс-папье и эффективной примерно на том же уровне.

Рассмотрим характеристики микроконтроллер «Iskra Neo:

Микроконтроллер	ATmega32U4
Тактовая частота	16 МГц
Флеш-память	32 КБ (из них 4 КБ занято загрузчиком)
SRAM-память	2,5 КБ
EEPROM-память	1 КБ
Рабочее напряжение	5 В
Рекомендуемое входное напряжение	7–12 В
Максимальный ток с пина 5V	1 А
Максимальный ток с пина ввода-вывода	40 мА
Максимальный суммарный ток с пинов ввода-вывода	200 мА
Портов ввода-вывода общего назначения	20
Портов с поддержкой ШИМ	7
Портов, подключённых к АЦП	12
Разрядность АЦП	10 бит
Аппаратные интерфейсы	UART, I²C, SPI
Габариты	69×53 мм

Данная плата имеет ограничения которые не подходят для наешго проекта , а именно количество цифровых входов, имея в виду того , что многие из пинов , рассчитаны на более слабое напряжение , что не позволяло бы нам «запитать» ни сервомоторы , ни реле , ни даже маленький дополнительный светильник , который собирался играть роль фонаря прямо с платы , но по остальным параметрам плата для нас подходила. Осталось только разрешить проблему с подключением , но при объединении несколько выходов в один имело несколько недочетов такие как: возможные нарушения

работы всей системы, а так же риск слишком перегрузить плату из-за чего могут возникнуть ряд других проблем.

Рассмотрим основные характеристики «Iskra mini»:

Микроконтроллер	ATmega328
Тактовая частота	16 МГц
Флеш-память	32 КБ (2 КБ используются
Оперативная память SRAM	2 КБ
Энергонезависимая память EEPROM	1 КБ
Рабочее напряжение	5 В
Входное напряжение	5–9 В
Портов ввода-вывода общего назначения	20
Портов с поддержкой ШИМ	6
Портов, подключённых к АЦП	8
Разрядность АЦП	10 бит
Аппаратные интерфейсы	UART, SPI, I²C
Максимальный ток одного вывода	40 мА
Максимальный ток с пина +5V	150 мА

Основные характеристики «Iskra Uno»

Микроконтроллер	ATmega328P
Тактовая частота	16 МГц
Флеш-память	32 КБ
SRAM-память	2 КБ
EEPROM-память	1 КБ
Портов ввода-вывода	20
Рабочее напряжение	5 В
Максимальный ток с пина 5V	1 А
Максимальный ток с пина ввода-вывода	40 мА
Максимальный суммарный ток с пинов ввода-вывода	200 мА
Аппаратные интерфейсы	UART, I²C, SPI

Основные характеристики «Iskra nano pro»:

Микроконтроллер	ATmega328PB
Ядро	8-битный AVR
Тактовая частота	16 МГц
Flash-память	32 КБ (2 КБ занимает загрузчик)
SRAM-память	2 КБ
EEPROM-памяти	1 КБ
Портов ввода-вывода всего	22
Портов с АЦП	8
Разрядность АЦП	10 бит
Портов с ШИМ	9
Разрядность ШИМ	8 бит
Аппаратных интерфейсов SPI	2
Аппаратных интерфейсов I²C / TWI	1
Аппаратных интерфейсов UART / Serial	2
Номинальное рабочее напряжение	5 В
Максимальный выходной ток пина 5V	1 Ам
Максимальный выходной ток пина 3V3	1 Ам
Максимальный ток с пина или на пин	40 мА
Допустимое входное напряжение от внешнего источника	7–12 В

Платы «Iskra mini» и «Iskra nano pro» имеют в своем расположении ограниченное количество цифровых выходов в размере 5 штук из-за чего они так же нам не подходят.

Последним подходящим вариантом остается плата “Iskra Mega”

Её характеристики:

Микроконтроллер	ATmega2560
Ядро	8-битный AVR
Тактовая частота	16 МГц
Flash-память	256 КБ (8 КБ занимает загрузчик)
SRAM-память	32 КБ
EEPROM-память	4 КБ
Порты ввода-вывода	70
Порты с АЦП	16
Разрядность АЦП	10 бит
Порты с ШИМ	15
Разрядность ШИМ:	8 бит
Аппаратный интерфейс SPI	1
Аппаратный интерфейс I²C / TWI	1
Аппаратный интерфейс UART / Serial	4
Номинальное рабочее напряжение	5 В
Максимальный выходной ток пина 5V	3 А
Максимальный выходной ток пина 3V3	300 мА
Максимальный ток с пина или на пин	40 мА
Допустимое входное напряжение от внешнего источника	5–27 В

Данная плата отвечала всеми необходимыми характеристики для создания макета малогабаритного подводного аппарата, но при её поиске выяснилось что она временно снята с продажи

После чего было принято решение о замене дополнительной платы расширения, которая отвечает за управление моторами и сервоприводами, а так же упрощает передачу команд, отладку и настройку. С камерой, серво моторами, радиомодулем возникла аналогичная проблема. В качестве

решения были куплены два двигателя постоянного тока от отечественного производителя, а именно два двигателя К140.

1.3. Оригинальные компоненты и импортные аналоги

Из-за отсутствия русских аналогов для платы серии “Arduino”,решение было принято в строну сборки макета с использованием оригинальных комплектующих и запчастей который в данный момент присутствуют на нашем рынке. На данный момент из плат серии “Arduino” сейчас присутствуют:

Arduino Nano
Arduino Uno
Arduino Mega
Arduino Due
Arduino Leonardo
Arduino Mini
Arduino Yún

Следует изучить основные характеристики данных плат.

Технические параметры Arduino Uno

Технические параметры плат Arduino Leonardo и Arduino mini

Микроконтроллер	ATmega32u4
Рабочее напряжение	5В
Напряжение питания (рекомендуемое)	7-12В
Напряжение питания (предельное)	6-20В
Цифровые входы/выходы	20
Каналы ШИМ	7
Аналоговые входы	12
Максимальный ток одного вывода	40 мА
Максимальный выходной ток вывода	3.3V 50 мА
Flash-память	32 КБ (ATmega32u4) из которых 4 КБ используются
SRAM	2.5 КБ (ATmega32u
EEPROM	1 КБ (ATmega32u4)
Тактовая частота	16 МГц

Микроконтроллер	ATmega328
Рабочее напряжение	5В
Напряжение питания	7-9В
Цифровые входы/выходы	14 (из которых 6могут использоваться в качестве ШИМ-выходов)
Аналоговые входы	8 (4 из которых на внешних выводах)
Максимальный ток одного вывода	40 мА
Flash-память	32 КБ (из которых 2 КБ используются загрузчиком)
SRAM	2 КБ
EEPROM	1 КБ
Тактовая частота	16 МГц

Технические параметры Arduino Due представлены в приложении В.

Технические параметры Arduino Nano:

Микроконтроллер	Atmel ATmega168 или ATmega328
Рабочее напряжение (логический уровень)	5В
Напряжение питания (рекомендуемое)	7-12В
Напряжение питания (предельное)	6-20В
Цифровые входы/выходы	14 (из которых 6 могут использоваться как ШИМ-выходы)
Аналоговые входы	8
Максимальный ток одного вывода	40 мА
Flash-память	16 КБ (ATmega168) или 32 КБ (ATmega328) из которых 2 КБ используются загрузчиком
SRAM	1 КБ (ATmega168) или 2 КБ (ATmega328)
EEPROM	512 байт (ATmega168) или 1 КБ (ATmega328)

Технические параметры Arduino Mega:

Микроконтроллер	ATmega2560
Рабочее напряжение	5В
Напряжение питания (рекомендуемое)	7-12В
Напряжение питания (предельное)	6-20В
Цифровые входы/выходы	54 (из которых 15 могут использоваться в качестве ШИМ-выходов)
Аналоговые входы	16
Максимальный ток одного вывода	40 мА
Максимальный выходной ток вывода	3.3V 50 мА
Flash-память	256 КБ из которых 8 КБ используются загрузчиком
SRAM	8 КБ
EEPROM	4 КБ
Тактовая частота	16 МГц

Технические параметры Arduino Yun

Входное напряжение питания	5 В
Максимальный выходной ток пина	3.3V: 800 мА
Размеры	69×53×9 мм
Микроконтроллер	ATmega32U4
Архитектура	AVR
Тактовая частота	16 МГц
Напряжение логических уровней	5 В
Контакты ввода-вывода общего назначения	20
Максимальный ток с пина ввода-вывода	40 мА
Контакты с поддержкой АЦП	6
Разрядность АЦП	10 бит
Контакты с поддержкой ШИМ	12
Разрядность ШИМ	8 бит
Flash-память	32 КБ (4 КБ занимает загрузчик)
SRAM-память	2,5 КБ
EEPROM-память	1 КБ
Аппаратные интерфейсы	1× I²C/TWI 1× UART/Serial

Платы: Arduino Nano, Arduino mini, Iskra Js, Iskra mini не подходили для нашего подводного аппарата поскольку все эти перечисленные платы имеют малое количество пинов. Платы: Arduino Leonardo и Arduino Uno имеют мощный встроенный микроконтроллер на своей базе, а так же имеют внушительное количество цифровых входов и выходов которые в полной мере покрывают нашу нужду при создании малогабаритного подводного аппарата Единственный их недостаток, это высокий спрос на них, средний срок ожидания доставки достигал отметки в 5 месяцев. Но имея обширные технические знания и несколько принтеров которые используются в создании 3Д моделей, мы смогли демонтировать плату Arduino Mega. Однако, после некоторого анализа, стало известно, что у плат Arduino есть

некоторые проблемы с многозадачностью. Это означает, что если мы хотим одновременно начать погружение и повернуть аппарат влево, то одна из этих двух команд будет выполняться с задержкой, примерно в 0.5 секунд, что может являться большой помехой в нашей работе. По этой причине было принято решение взять две платы Arduino Mega, чтобы разделить их по функционалу. Одна из плат будет управлять

плавниками, ответственными за возможность всплытия/погружения аппарата и движение вперёд (тк это потоковая команда, то её выполнение не требует много ресурсов платы в отличии от команд «поворот» в виду отсутствия необходимости отсчитывать текущий угол аппарата) После того как в наших руках оказалась плата которая стала базисом для нашего проекта, необходимо было купить оставшееся подвесное оборудования а именно сервомоторы.

Для начала были куплены сервомоторы MG995

Технические параметры MG995:

Рабочее напряжение	4.8-7.2 В
Угол поворота	120 градусов
Крутящий момент	8,5 кг/см (при 4.8 В), 10 кг/см (при 6 В)
Скорость	0,20 сек/60° (при 4,8 В), 0,16 сек/60° (при 6 В)
Вес	55 г
Размер	40x20x42 мм

Данные сервомоторы обладают всеми необходимыми нам характеристиками и уже зарекомендовали себя на моём опыте как качественные компоненты с

высоким показателем надёжности и эффективности, поэтому было решено взять их как для плавников, так и для хвоста нашего робота.

Далее необходимо было выбрать , хотим ли мы сделать робота на проводном управлении или попробовать создать некий гибрид , используя плавающий буй+ радио модуль , для передачи сигнала с расстояния на буй , а с буя через кабели передать сигнал на плату , однако после некоторого размышления , было решено , что на текущий момент , мы оставим разработку полностью проводной , без радио модуля , это позволит сначала отработать нам все возможные подключения к платам , уточнить необходимые диаметры кабелей для более долгой и качественной работы , не нарушая хорошее соотношение цена/качество и позволит нам не умещать тяжёлый блок питания на борту судна , что позволит нам точнее и лучше скорректировать его остойчивость и плавучесть .

После чего нам необходимо было выбрать камеру, после некоторого анализа модулей камер как с потоковой передачей данных, так и не с потоковой передачей данных, представленных на рынке на текущий момент, были найдены следующие варианты.

На рынке на момент выбора были представлены следующие варианты: OV7670, OV2640 и Eachine 1000TVL

Технические параметры OV2640

Датчик размер	1/4 дюйма
Пиксель	1600 * 1200 (200 Вт)
Выходной формат	RGB565 / JPEG / YUV / YCbCr и т. Д.
Интерфейс управления	SCCB
Рабочее напряжение	3,3 В
Размер модуля	27 мм x 27 мм

Технические параметры OV7670

Количество пикселей	0.3 mpx
Интерфейс	Standard SCCB interface compatible with I2C interface
Выходной формат	– RawRGB to RGB (GRB4: 2:2, RGB565/555/444), YUV (4:2:2) and YCbCr (4:2:2)
Поддерживает	VGA, CIF
Разрешение	640x480
Напряжение питания	2.5-3.3В
Потребляемая мощность	60мВт
Ток покоя	20мкА
Оптика	1/6"
Угол обзора	25°
Чувствительность	1.3 В/(Lux-sec)
Частота дискретизации	50/60Гц
Скорость записи	максимум 30кадр/с
соотношение сигнал/шум	46 дБ
динамический диапазон	52 дБ
режим развертки	по рядам
размер пикселя	3.6мкм x 3.6мкм

Характеристики Eachine 1000TVL

Потребляемая мошность	55mA @ 5v
Питание	5-20 Вольт
Минимальная освешенность (люкс)	0.08lux / f1.2
Размер матрицы	1/3 "
Формат	NTSC / PAL (переключатель)
Разрешение	PAL: 976x582 NTSC: 976x494
Объектив	2.8мм с IR покрытием
FOV	110°
Рабочая температура	от 0 до 40°C
Размер камеры	28 * 24.5 * 17.5 мм
Размер	17мм * 14мм

Для визуального обозрения повреждений была выбрана Eachine 1000TVL. Данная камера специально создана для управляемых высокоскоростных дронов, при помощи которых участвуют в гоночных соревнованиях между друг другом. При помощи этой камеры игроки получают прямую видеотрансляцию с дрона с минимальной задержкой. Такая камера не имеет большой сложности в монтаже, что будет очень полезно, при необходимости замены/починки.

Для определения местоположения аппарата было решено использовать GPS модуль NEO-6M , имеющий следующие характеристики :

GPS модуль	U-Blox NEO-6M-0-001
Чувствительность	-161 dBm;
Скорость обновления	5 Гц
Интерфейсы	UART (выведен), SPI, DDC, IIC
Передает координаты в формате	NMEA
Скорость подключения по умолчанию по UART	9600 бод
Напряжение питания	3 – 5 В
Возможность работы с программами	U-Center и т.п

1.4. Сборка и теоретическое описание проекта.

Проект будет представлять из себя дистанционно управляемый подводный аппарат, оперирующий на двух платах Arduino Mega 2560 (rev3), к которым подключён motorshield Adafruit V2, через который будут питаться двигатели и сервоприводы. Видеотрансляция будет осуществляться напрямую на дисплей пульта через аналоговый кабель, в то время как данные GPS будут поступать с небольшого буя, который будет находиться над аппаратом и будет служить дополнительной мерой для определения его текущего местоположения.

Заключение по первой главе.

В данной главе проведён подробный анализ комплектующих отечественного и импортного производства. На основании, которого для реализации проекта были выбраны следующие комплектующие: Arduino mega 2560 в количестве двух единиц, , камера Eachine 1000TVL и GPS щит

NEO-6M. В роли пульта было решено использовать портативный компьютер для облегчения управления и уменьшения количества компонентов.

Глава 2. Разработка модели программного обеспечения для управления малогабаритным подводным аппаратом

2.1 Программное обеспечение для двигателей при осуществлении всплытия

При создании макета малогабаритного подводного аппарата нами были выбранные двигатели : F130-16165 , который представлены на риссинуке 2.1.1

Рисунок 2.1.1 5-ти вольтовый электро мотор F130-16165

Данный мотор должен обеспечивать возможность всплытия нашего аппарата. Основные преимущества при выборе мотора были его : номинальное напряжение (4,5В) , диапазон рабочего напряжения (от 1.0 до 5.0 В) , кпд (53.5%) , мощность (0,67 Вт) , а так же небольшой размер. Основной принцип подключения данного типа мотора , это непосредственное подключение к главное плате Arduino или к допп. Плате Arduino через порты Плата управления сервомоторами подключается через такие порты PWMH ,к портам коммуникации ,два подключения к выходу на питание VCC,два выхода на заземление GND , 6 подключений на аналоговые порты .Питание подключается к отдельному порту для питания. Модуль радио управления

подключается к портам MISO, MOSI и SCK.Модуль фото камеры используются 7 цифровых портов,6 аналоговых портов, 3 выхода на заземление и 4 выхода на питание 3.3В , согласно портам указанным на принципиальной схеме на рисунке 2.1.2 платы управления подводным аппаратом

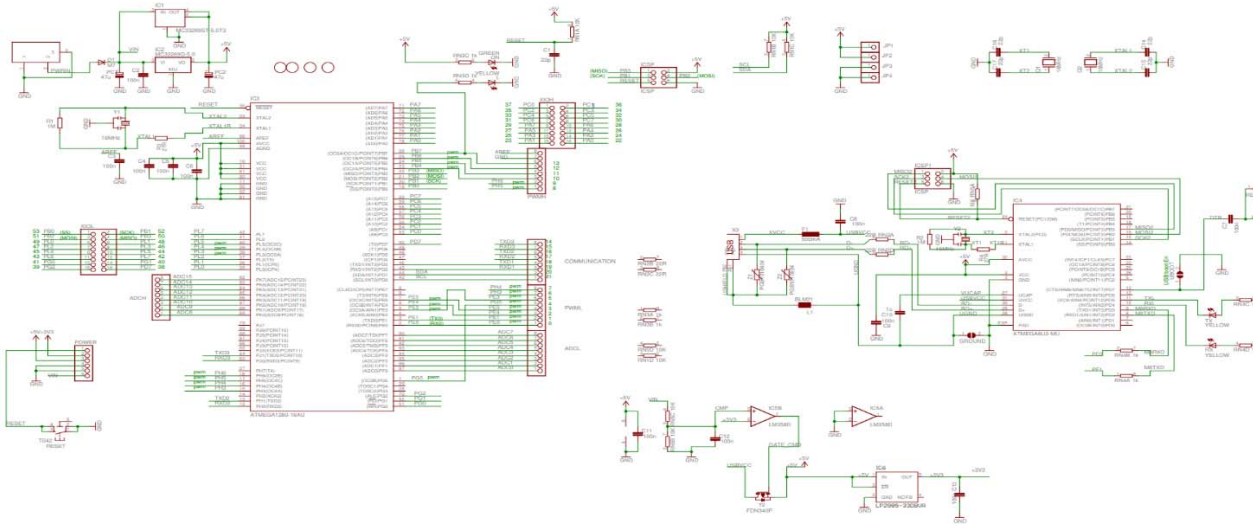


Рисунок 2.1.2. Принципиальная схема платы управления подводным аппаратом.

Движение нашего подводного аппарата на напрямую зависит от правильной работы выбранных нами сервомотор и серводвигателей. Следовательно для их правильной и бесперебойной работы требуется правильное подключение к плате управления сервомоторами и двигателями исходя из её принципиальной схемы , указанной на рисунке 2.1.3 и правильное написание программной составляющей

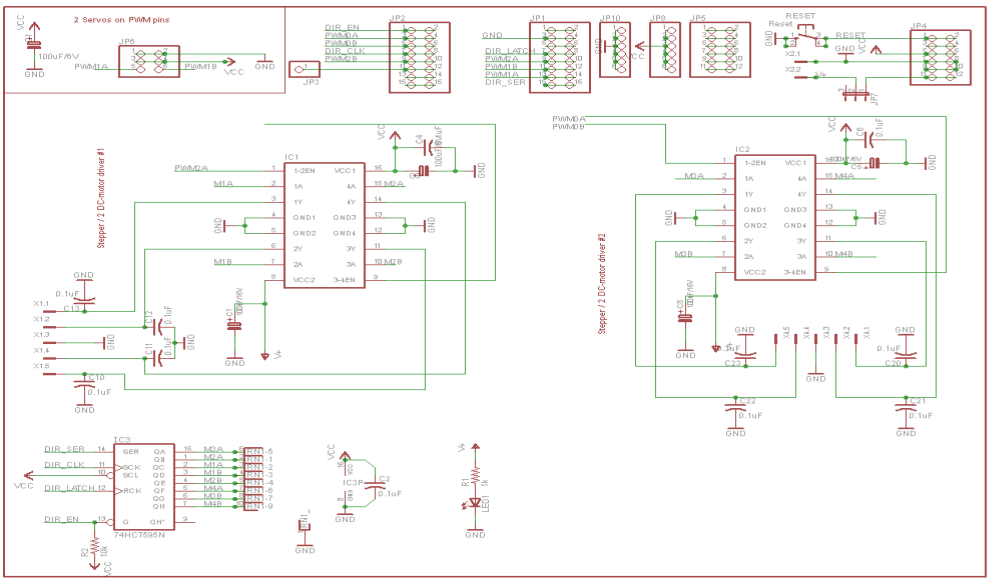


Рисунок 2.1.3. Электрическая принципиальная схема платы управления серводвигателями

Для того чтобы, аппарат имел возможность осуществлять всплытия, необходимо предварительно запрограммировать основную плату Arduino, путем передачи ей кода написсного на языке C++\C#.

Таким образом для настройки двигателей на всплытие мы делаем следующие шаги:

1. Установить библиотеку отвечающую за поддержку моторов : #include <Servo.h>

2. Добавляем библиотеку SPI для обмена протоколов данных между нашей основой платой и подключенными устройствами. Вместе с I2C и UART этот протокол часто используется для многих типов периферийных устройств

3.Задаем ячейку для моторов и пины как на рисунке 2.1.4

```
byte servoPin = 5;
byte servoPin = 4;
byte PWMA = 6;
byte AIN1 = 8;
byte AIN2 = 7;
byte AIN3 = 9;
byte AIN4 = 11;
```

Рисунок 2.1.4. Установка пинов.

3.Устанавливаем максимальную мощность на двигатели (рисунок 2.1.5)

```
byte motorSpeedA = 0;
byte motorSpeedAMAX = 255;
const byte motorSpeedAReverse = 185;
const byte servoNeutral = 65;
```

Рисунок 2.1.5 Максимальная скорость

4. Сохраняем предполагаемый угол поворота рулевого колеса для отправки в мотор и устанавливаем значение поворота по горизонтали. Рисунок 2.1.6.

```
byte servoPosition;
byte steerLeftAngleMAX = servoNeutral - 37;
byte steerRightAngleMAX = servoNeutral + 37;
unsigned long lastSpeedButtonReadTime = 0;
unsigned long lastControllerReadTime = 0;
unsigned long currentMillis = 0;
```

Рисунок 2.1.6. Установка угла и значений поворота.

5. Проводим основную настройку для выходов и входов как на рисунке 2.1.7.

```
void setup() {
  servoPosition = servoNeutral; //startup with neutral alignment
  steeringServo.attach(servoPin);
  steeringServo.write(servoPosition);

  pinMode(PWMA, OUTPUT);
  pinMode(AIN1, OUTPUT);
  pinMode(AIN2, OUTPUT);
  pinMode(AIN3, OUTPUT);
  pinMode(AIN4, OUTPUT);

  Serial.begin(9600);
  radio.begin();
  radio.openReadingPipe(0, address);
  radio.setAutoAck(false);
  radio.setDataRate(RF24_250KBPS);
  radio.setPALevel(RF24_PA_LOW);
  radio.startListening();
    // Reset data on startup to default values
  resetData();
```

Рисунок 2.1.7. Код настройки выходов в цикле программы.

6. Основной цикл. В основном кицкле представленном на рисунке 2.1.8 , мы можем наблюдать следующие строчки: получение данных от контроллера, установка серво моторов, проверка времени передачи, а так же кнопкн выбора режима проверки скорости, в том числе период времени 500 мс для устранения дребезга и ход действий если данные кнопки получены и кнопка нажата, и прошло необходимое время. Строчки кода где настраиваем максимальную скорость.

```
void loop() {
    readController();
    setMotorSpeeds();
        currentMillis = millis();
    if ((radio.available()) && (data.joystick2Button == LOW) && ((currentMillis - lastSpeedButtonReadTime) > 500)) {
        if (motorSpeedAMAX == 120) {
            motorSpeedAMAX = 180;} else if (motorSpeedAMAX == 180) {
                motorSpeedAMAX = 255;} else if (motorSpeedAMAX == 255) {
                    motorSpeedAMAX = 120;
            }
        lastSpeedButtonReadTime = millis();
    }
}
```

Рисунок 2.1.8. Код основного цикла.

Имеются допольнительные командные строчки что представлены на рисунке 2.1.9. Они необходимы для установки сигнала контроллера в нейтральное положение , для преобразования увеличивающиеся показания оси Y для перехода от 50до 90в 0 до максимального значения для сигнала ШИМ для увеличения скорости двигателя, преобразования значение правого контроллера в угол для рулевого сервопривода, записи определенной скорости и положения руля в моторы

```
void resetData() {
    data.joystick1xAxisBytes = 127;
    data.joystick1yAxisBytes = 127;
    data.joystick2xAxisBytes = 127;
    data.joystick2yAxisBytes = 127;
    data.joystick2Button = 1;
    //     Serial.println("reset data");
  }

void readController() {
  if (radio.available()) {
    radio.read(&data, sizeof(Data_Package));
     Serial.println("joystick1-Y: ");
     Serial.print(data.joystick1yAxisBytes);
     Serial.println("joystick2-Y: ");
     Serial.print(data.joystick2yAxisBytes);
      Serial.println("joystick1-X: ");
      Serial.print(data.joystick1xAxisBytes);
      Serial.println("joystick2-X: ");
      Serial.print(data.joystick2xAxisBytes);
      Serial.println("joystick2-Button: ");
      Serial.print(data.joystick2Button); */
      // Record the last time the controller was successfully read
      lastControllerReadTime = millis();
  }
  currentMillis = millis();
  if ((currentMillis - lastControllerReadTime) > 1000) {
    resetData();
    }
}
```

Рисунок 2.1.9. Дополнительные команды для двигателей и серво.

2.2 Программное обеспечение для двигателей при осуществлении погружения

Для изменения угла атаки двигателей а так же для изменения курса, нами были выбраны серво привода Сервопривод MG995S представленный на рисунке 2.2.1.

Рисунок 2.2.1 Сервопривод MG995S

Данный серво привод был выбран нами за следующий характеристики : Рабочее напряжение от 4,8 до 7,2 В. Максимальный потребляемый ток – 1000мА. Диапазон вращения составляет 180 градусов. Скорость вращения без нагрузки при4,8В : 60 градусов за 0,17секунд, скорость вращения без нагрузки при 6В: 60 градусов за 0,14с. Максимальный крутящий момент 11 кг\см.

Данный тип сервопривода так же как и двигатель подключается к плате Arduino или же к плате расширения Arduino в нужные выходы как соответственно представлено на рисунке 2.2.2.

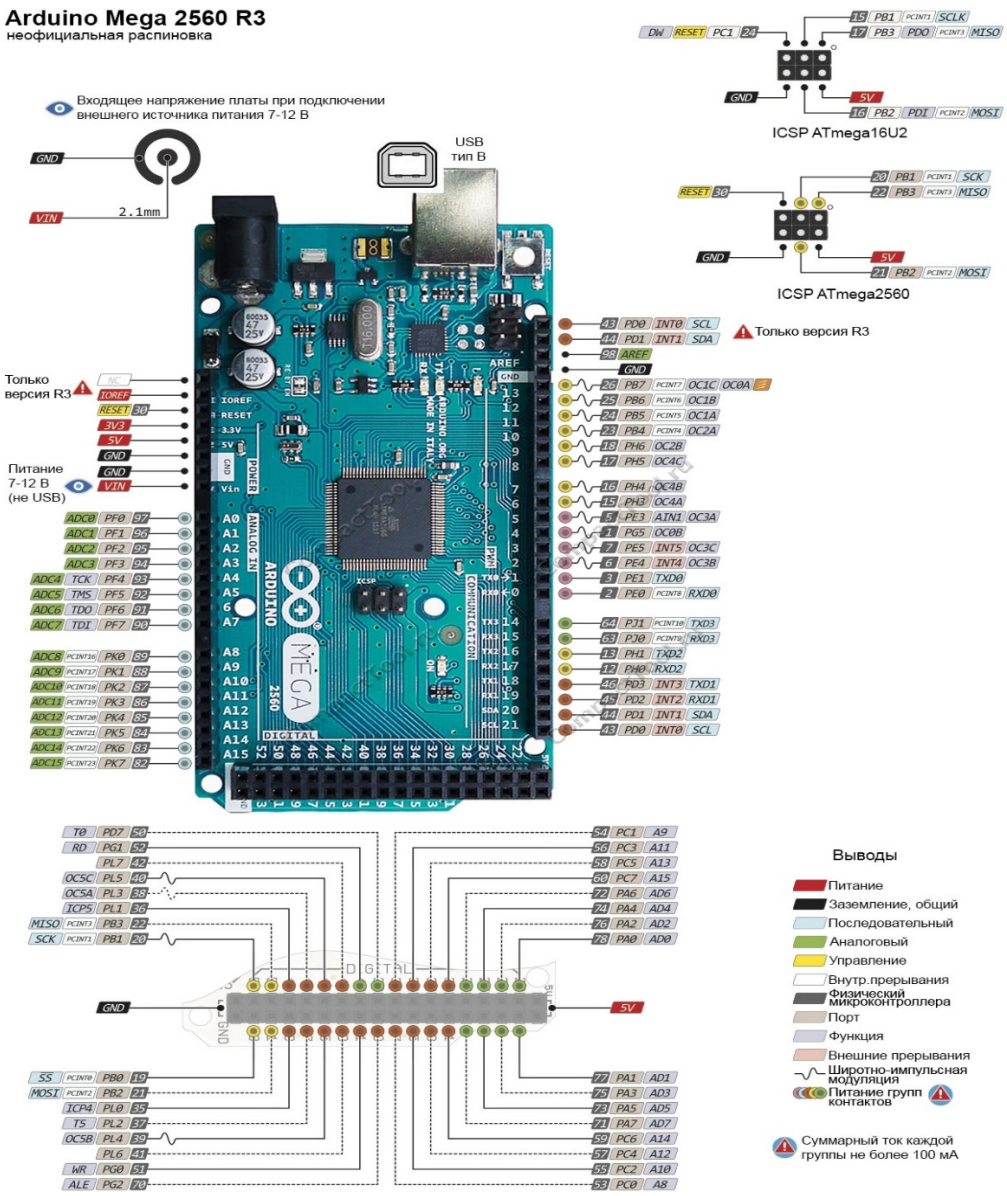


Рисунок 2.2.2. Расположение входов и выходов на плате Arduino Mega

Если же подключать сервопривода или моторы к плате расширения, то они подключаются так же на прямую как могут подключаться к основной плате Arduino, стоит заметить что количество входов и выходов с дополнительной платы управления сервомоторами, что представлена на рисунке 2.2.3, меньше чем на основной, но подключение осуществляется проще.

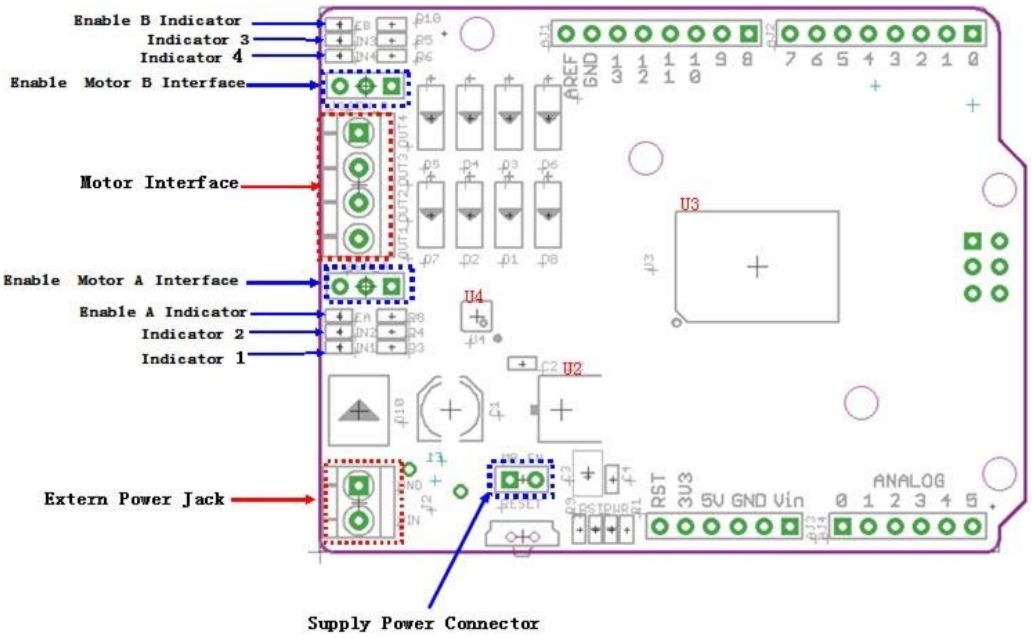


Рисунок 2.2.3 Расположение входов и выходов на плане управления сервомоторами.

Для того чтобы, аппарат имел возможность осуществлять всплытия, необходимо предварительно запрограммировать основную плату Arduino, путем передачи ей кода написсного на языке C++\C#.

Таким образом для настройки двигателей на всплытие мы делаем следующие шаги:

1. Установить библиотеку отвечающую за поддержку моторов : #include <Servo.h>

2. Добавляем библиотеку SPI для обмена протоколов данных между нашей основой платой и подключенными устройствами. Вместе с I2C и UART этот протокол часто используется для многих типов периферийных устройств

3.Задаем ячейку для моторов и пины рисунке 2.2.4

```
byte servoPin = 5;
byte servoPin = 4;
byte PWMA = 6;
byte AIN1 = 8;
byte AIN2 = 7;
byte AIN3 = 9;
byte AIN4 = 11;
```

Рисунок 2.2.4. Установка пинов.

4.Устанавливаем максимальную мощность на двигатели рисунок 2.2.5

```
byte motorSpeedA = 0;
byte motorSpeedAMAX = 255;
const byte motorSpeedAReverse = 185;
const byte servoNeutral = 65;
```

Рисунок 2.2.5. Максимальная скорость

6. Сохраняем предполагаемый угол поворота рулевого колеса для отправки в мотор и устанавливаем значение поворота по горизонтали. Рисунок 2.2.6.

```
byte servoPosition;
byte steerLeftAngleMAX = servoNeutral - 37;
byte steerRightAngleMAX = servoNeutral + 37;
unsigned long lastSpeedButtonReadTime = 0;
unsigned long lastControllerReadTime = 0;
unsigned long currentMillis = 0;
```

Рисунок 2.2.6. Установка угла и значений поворота.

8. Проводим основную настройку для выходов и входов как на рисунке 2.2.7.

```
void setup() {
  servoPosition = servoNeutral; //startup with neutral alignment
  steeringServo.attach(servoPin);
  steeringServo.write(servoPosition);

  pinMode(PWMA, OUTPUT);
  pinMode(AIN1, OUTPUT);
  pinMode(AIN2, OUTPUT);
  pinMode(AIN3, OUTPUT);
  pinMode(AIN4, OUTPUT);

  Serial.begin(9600);
  radio.begin();
  radio.openReadingPipe(0, address);
  radio.setAutoAck(false);
  radio.setDataRate(RF24_250KBPS);
  radio.setPALevel(RF24_PA_LOW);
  radio.startListening();
    // Reset data on startup to default values
  resetData();
```

Рисунок 2.2.7. Код настройки выходов и входов в цикле программы.

9. Основной цикл. В основном цикле представленном на Рисунке 2.2.8 , мы можем наблюдать следующие строчки: получение данных от контроллера, установка серво моторов, проверка времени передачи, а так же кнопке выбора режима проверки скорости, в том числе период времени 500 мс для устранения дребезга и ход действий если данные кнопки получены и кнопка нажата, и прошло необходимое время. Строчки кода где настраиваем максимальную скорость.

```
void loop() {
    readController();
    setMotorSpeeds();
        currentMillis = millis();
    if ((radio.available()) && (data.joystick2Button == LOW) && ((currentMillis - lastSpeedButtonReadTime) > 500)) {
        if (motorSpeedAMAX == 120) {
            motorSpeedAMAX = 180;} else if (motorSpeedAMAX == 180) {
                motorSpeedAMAX = 255;} else if (motorSpeedAMAX == 255) {
                    motorSpeedAMAX = 120;
            }
        lastSpeedButtonReadTime = millis();
    }
}
```

Рисунок 2.2.8. Код основного цикла.

Имеются дополнительные командные строчки что представлены на рисунке 2.2.9. Они необходимы для установки сигнала контроллера в нейтральное положение , для преобразования уменьшающиеся показания оси Y для перехода назад от 45до 0 4. В 0 до максимального значения для сигнала ШИМ для увеличения скорости двигателя.

```
void resetData() {
    data.joystick1xAxisBytes = 127;
    data.joystick1yAxisBytes = 127;
    data.joystick2xAxisBytes = 127;
    data.joystick2yAxisBytes = 127;
    data.joystick2Button = 1;
    //     Serial.println("reset data");
  }

void readController() {
  if (radio.available()) {
    radio.read(&data, sizeof(Data_Package));
     Serial.println("joystick1-Y: ");
     Serial.print(data.joystick1yAxisBytes);
     Serial.println("joystick2-Y: ");
     Serial.print(data.joystick2yAxisBytes);
      Serial.println("joystick1-X: ");
      Serial.print(data.joystick1xAxisBytes);
      Serial.println("joystick2-X: ");
      Serial.print(data.joystick2xAxisBytes);
      Serial.println("joystick2-Button: ");
      Serial.print(data.joystick2Button); */
      // Record the last time the controller was successfully read
      lastControllerReadTime = millis();
  }
  currentMillis = millis();
  if ((currentMillis - lastControllerReadTime) > 1000) {
    resetData();
    }
}
```

Рисунок 2.2.9. Дополнительные команды для сервоприводов и моторов.

2.3 Программное обеспечение подводного аппарата для двигателей для горизонтального перемещения

Для передвижения подводного аппарата по горизонтальной линии, необходимо было выбрать и установить двигатели. Нами были выбраны . 5-ти вольтовый электро мотор F130-16165 .

Для написания кода управления был использован язфк программирования C++\C# , так же как и для сервоприводов.

1. Перед началом написания программного кода для двигателей, нам необходимо подключить библиотеки “RF24.h” и “ SPI.h” как указано на рисунке 2.3.1 . Эти две библиотеки отвечают за связь между устройствами, за прием и передачу данных.

```
#define R_IR 3
#define L_IR 4
#define L_MR 5
#define R_MR 6
#define min_speed 200
#define max_speed 800
#include <SPI.h>
#include "RF24.h"
RF24 myRadio (7, 8);
```

Рисунок 2.3.1. Начало написания программы для дивгатлей.

2. Далее необходимо задать структуру данных как на рисунке 2.3.2., обазначить входы и выходы, задаем скорость приема и передачи,задаем массив для адресса

```
struct package
{
  int msg;
};
typedef struct package Package;
Package data;
byte addresses[][6] = {"0"};
void setup() {
  pinMode(R_IR, INPUT);
  pinMode(L_IR, INPUT);
  pinMode(L_MR, OUTPUT);
  pinMode(R_MR, OUTPUT);
  Serial.begin (9600);
  myRadio.begin();
  myRadio.setChannel(115);
  myRadio.setPALevel(RF24_PA_MIN);
  myRadio.setDataRate( RF24_250KBPS ) ;
}
int received;
```

Рисунок 2.3.2. Начальная организация данных перед основной настройкой.

3. В цикле настройки программы задаем минимальное и максимальное значение скорости, указываем параметры аналогового чтения , а так же условия при приеме значения скорости как на рисунке 2.3.3.

```
void loop() {
ReadData();
}
void Control_Car()
{
  if (received>=1 && received <=10)
  {
    int PWM_Value =  map (received, 1, 10, min_speed, max_speed);
    analogWrite(R_MR,PWM_Value);
    analogWrite(L_MR,PWM_Value);
  }
    if (received>=11 && received <=20)
  {
    int PWM_Value =  map (received, 11, 20, min_speed, max_speed);
    analogWrite(R_MR,0);
    analogWrite(L_MR,0);
  }
    if (received>=21 && received <=30)
  {
    int PWM_Value =  map (received, 21, 30, min_speed, max_speed);
    analogWrite(R_MR,PWM_Value);
    analogWrite(L_MR,0);
  }
      if (received>=31 && received <=40)
  {
    int PWM_Value =  map (received, 31, 40, min_speed, max_speed);
    analogWrite(R_MR,0);
    analogWrite(L_MR,PWM_Value);
  }
}
```

Рисунок 2.3.3. Цикл настройки программы.

4. В цикле настройки приема и чтения данных мы указываем то что необходимо с помощью программного кода проверить приходят ли данные , доступен ли прием данных , откуда осуществлять чтение самих данных, устанавливаем размер принимаемых данных, а так же можем остановить или запустить прием данных как на рисунке 2.3.4.

```
void ReadData()
{
  myRadio.openReadingPipe(1, 0xF0F0F0F0AA);
  myRadio.startListening();
  if ( myRadio.available())
  {
    while (myRadio.available())
    {
      myRadio.read( &data, sizeof(data) );
    }
    Serial.print("\nReceived:");
    Serial.println(data.msg);
    received = data.msg;
    Control_Drone();
  }
  else
  {
  //analogWrite(R_MR,0);
  //analogWrite(L_MR,0);
  }
}
```

Рисунок 2.3.4. Цикл чтения данных

5.Цикл записи данных представленный на рисунке 2.3.5 , отвечает за написание данные , выводе их для пользователя ,определение массива сообщения и размера. Так же в цикл включена программа остановки приема данных и команда для приема сообщения определенного размера.

```
void WriteData()
{
  myRadio.stopListening();
  myRadio.openWritingPipe(0xF0F0F0F066);
  myRadio.write(&data, sizeof(data));
  Serial.print("\nSent:");
  Serial.println(data.msg);
  delay(300);
}
```

Рисунок 2.3.5. Цикл записи данных.

Вывод по главе 2.

Во второй главе нами были рассмотрены компоненты с помощью которых осуществляется движение , технические характеристики, принцип подключения к основной плате Arduino или же к доп. плате Arduino. Так же был рассмотрен основной программный код , который был разработан для управления всплытием, погружением и так для горизонтального передвижения мало габаритного подводного аппарата.

Глава 3. Разработка макета графического интерфейса для управления малогабаритным подводным аппаратом.

3.1 Настройка приема и передачи данных местоположения малогабаритного подводного аппарата по протоколу "NMEA"

NMEA (от «National Marine Electronics Association» Национальная Ассоциация Морской Электроники) — стандарт, определяющий текстовый протокол связи морского (как правило, навигационного) оборудования (или оборудования, используемого в транспорте) между собой. NMEA стал особенно популярен в связи с распространением GPS/GLONASS-приёмников, использующих этот стандарт.

Интерфейс обмена данными большинства GPS приемников реализован в соответствии с этой спецификацией. Основные навигационные программы, которые обеспечивают отображение данных в реальном времени, поддерживают и «понимают» NMEA протокол. Эти данные содержат полные навигационные пакеты GPS приемника – координаты, высоту, время и скорость. Все NMEA сообщения состоят из последовательного набора данных, разделенных запятыми. Каждое отдельное сообщение не зависит от других и является полностью «завершенным». NMEA сообщение включает заголовок, набор данных, представленных ASCII символами, и поле «чек-суммы» для проверки достоверности переданной информации. Все устройства установленные на судне или корабле взаимодействуют друг с другом с целью приема или передачи данных как на рисунке 3.1.1.

Рис.3.1.1 – Пример схемы взаимодействия устройств NMEA

Протокол NMEA 0183

NMEA — это формат передачи сообщений между корабельными приборами. Он включает в себя систему сообщений для обмена информацией между навигационными GPS-приемниками и потребителями навигационной информации. Все команды и сообщения передаются в текстовом ASCII-виде. Сообщения относящиеся к GPS-приемнику начинаются с $GP, в конце строки сообщения должны быть символы. В последнем поле сообщения может быть указана контрольная сумма текущего сообщения, начинающаяся с разделителя *. Контрольная сумма всех символов сообщения, включая пробелы, расположенных между разделителями $и *, не включая последних. Шестнадцатеричный результат переводится в два ASCII-символа (0-9, A-F).

Спецификация оборудования

Один “**TALKER**” и несколько “**LISTENERS**” могут быть соединены параллельно соединительным проводом. количество “**СЛУШАТЕЛЕЙ**” зависит от выходной мощности и требований к входному приводу отдельных устройства.

1.Соединительный провод.

Соединение между устройствами может осуществляться с помощью двухжильного экранированного провода типа «витая пара».

2. Определения проводников.

Проводниками, упомянутыми в настоящем стандарте, являются сигнальные линии «А» и «В» и экран.

3.Электрические соединения/требования к экрану.

Все соединения сигнальной линии «А» соединены параллельно со всеми соединениями устройства «А» и всеми сигнальными соединениями**.** Соединения линии «В» подключаются параллельно со всеми соединениями «В» устройства. Экраны всех кабелей ”LISTENERS” должны быть подключены только к раме “TALKERS” и не должны быть подключены к “LISTENERS”

4.Соединитель (Connector) .

Стандартный разъем не указан. Везде, где это возможно, следует использовать легкодоступные коммерческие соединители. Изготовители должны предоставить средства для идентификации пользователем используемых соединителей.

5. Характеристика электрического сигнала.

5.1 Определение состояние сигнала.

Состояние простоя, маркировки, логической "1", ВЫКЛ или стопового бита определяется отрицательным напряжением на линии "A" относительно линии «B». Состояние активного, интервального, логического "0", ВКЛ или стартового бита определяется положительным напряжением на линии "A" относительно линии «B». Обратите внимание, что приведенные выше уровни «A» по отношению к уровням «B» инвертированы из требований к входу/выходу напряжения стандартных UART и что многие линейные драйверы и приемники обеспечивают логическую инверсию.

5.2 Схема управления "TALKER"

Не предусмотрено подключение к шине более одного "TALKER". Цепь управления, используемая для подачи сигнала «A» и возврата «B», должна как минимум соответствовать требованиям EIA-422-A.

5.3 Схема приема "LISTENER"

Несколько "LISTENER" могут быть подключены к одному "TALKER". Цепь приема "LISTENER" должна состоитm из оптоизолятора и должен иметь защитные цепи для ограничения тока, обратного смещения и мощности рассеивания на оптодиоде как на рисунке 3.1.2

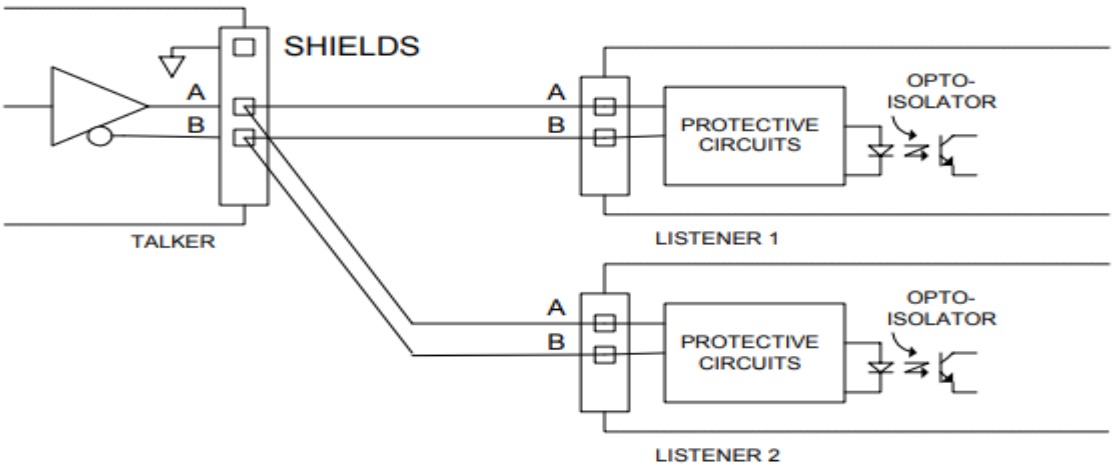


Рис 3.1.2. Пример изолятор и защитной цепи.

Цепь приема должна быть рассчитана на работу с минимальным дифференциальным входным напряжением 2,0 В и не должна потреблять от линии более 2,0 мА при этом напряжении. По соображениям совместимости с оборудованием, разработанным в соответствии с более ранними версиями этого стандарта, отмечается, что «холостой ход, маркировка, логическая «1», ВЫКЛ или состояние стопового бита» ранее определялись в диапазоне от -15 до + 0,5 Вольт. Было определено, что «активный, интервальный, логический «0», состояние «Вкл.» или «Стартовый бит» находится в диапазоне от +4,0 до +15 В при подаче не менее 15 мА.

5.4 Электрическая изоляция

В “LISTENER”не должно быть прямого электрического соединения между сигнальной линией «А», обратной линией «В» или экраном и заземлением или питанием судна. Требуется заземление на корпус судна.

5.5 Максимальное напряжение на шине

Максимальное приложенное напряжение между сигнальными линиями «А» и «В», а также между любой линией и землей должно соответствовать спецификации EIA-422. Для защиты от неправильного подключения и для использования с более ранними конструкциями “TALKER” все устройства приемной цепи должны выдерживать напряжение 15 вольт между сигнальными линиями «А» и «В», а также между любой линией и землей в течение неопределенного периода времени.

6. Передача данных

Данные передаются в последовательной асинхронной форме в соответствии со стандартами ANSI. Первый бит является начальным битом, за ним следуют биты данных, начиная с младшего бита, как показано на рис. 3.1.3. Используются следующие параметры:

- Скорость передачи 4800
- Биты данных 8 (d7 = 0)
- Паритет Нет
- Стоповые биты Один

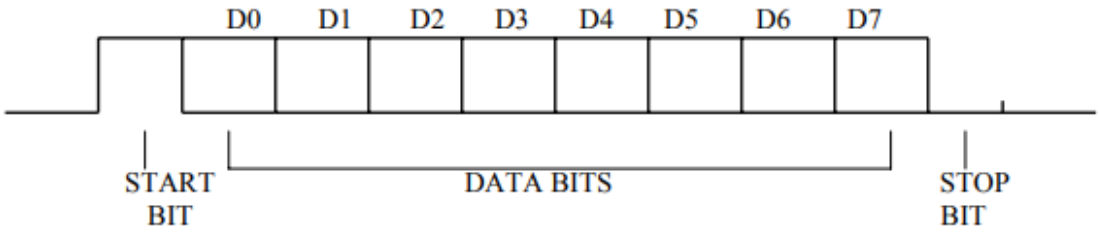


Рис. 3.1.3

7. Протокол формата данных

7.1 Символы

Все передаваемые данные должны интерпретироваться как символы ASCII. Старший бит 8-битового символа всегда должен передаваться как ноль (d7 = 0).

7.2 Зарезервированные символы

Зарезервированный набор символов состоит из тех символов ASCII, которые показаны в таблице 1, где “HEX” это шестнадцатеричный код, а “DECIMAL”- десятичные значения.

Symbol	HEX	DECIMAL	Value
<CR>	0D	13	Окончание строки. Разделитель конца предложения
<LF>	0A	10	Перевод строки
$	24	36	Начало параметрического разделителя предложений
*	2A	42	Разделитель поля контрольной суммы
,	2C	44	Разделитель полей
!	21	33	Разделитель предложения начала инкапсуляции
\	5C	92	Зарезервировано для будущего использования
^	5E	94	Разделитель кода для HEX-представления
			Символы ISO 8859-1
~	7E	126	Зарезервировано для будущего использования
<del>	7F	127	Зарезервировано для будущего использования

Таблица 1. Таблица с перечнем зарезервированных символов.

Эти символы используются для определенных целей форматирования, таких как разграничение предложений и полей, и, за исключением разграничения кода, не должны использоваться в полях данных.

7.2 Допустимые символы

Допустимый набор символов состоит из всех печатаемых символов ASCII (от HEX 20 до HEX 7E), кроме тех, которые определены как зарезервированные символы. В таблице 2 перечислен допустимый набор символов.

	Hex	Dec		Hex	DEC		Hex	DEC
Space	20	32	@	40	64	`	60	96
Reserved	21	33	A	41	65	a	61	97
"	22	34	B	42	66	b	62	98
#	23	35	C	43	67	c	63	99
Reserved	24	36	D	44	68	d	64	100
%	25	37	E	45	69	e	65	101
&	26	38	F	46	70	f	66	102
'	27	39	G	47	71	g	67	103
(	28	40	H	48	72	h	68	104
)	29	41	I	49	73	i	69	105
Reserved	2A	42	J	4A	74	j	6A	106
+	2B	43	K	4B	75	k	6B	107
Reserved	2C	44	L	4C	76	l	6C	108
-	2D	45	M	4D	77	m	6D	109
.	2E	46	N	4E	78	n	6E	110
/	2F	47	O	4F	79	o	6F	111
0	30	48	P	50	80	p	70	112
1	31	49	Q	51	81	q	71	113
2	32	50	R	52	82	r	72	114
3	33	51	S	53	83	s	73	115
4	34	52	T	54	84	t	74	116
5	35	53	U	55	85	u	75	117
6	36	54	V	56	86	v	76	118
7	37	55	W	57	87	w	77	119
8	38	56	X	58	88	x	78	120
9	39	57	Y	59	89	y	79	121
:	3A	58	Z	5A	90	z	7A	122
;	3B	59	[	5B	91	{	7B	123
<	3C	60	Reserved	5C	92	|	7C	124
=	3D	61	]	5D	93	}	7D	125
>	3E	62	Reserved	5E	94	Reserved	7E	126
?	3F	63	_	5F	95	Reserved	7F	127

Таблица 2. Таблица с перечнем недопустимых символов.

7.3 Недопустимые символы.

Значения ASCII, не указанные как «зарезервированные символы» или «допустимые символы», исключаются и не должны передаваться в любое время. Когда необходимо передать 8-битный символ, определенный в ISO 8859-1, который является зарезервированным символом (таблица 1) или не указан в таблице 2 в качестве допустимого символа (например, в собственном предложении или текстовом предложении) должны

использоваться три символа. За зарезервированным символом «^» (HEX 5E) следуют два символа ASCII (0–9, A–F), представляющие шестнадцатеричное значение передаваемого символа. Например, чтобы отправить заголовок как «127,5º», передайте: «127,5^F8». для отправки зарезервированных символов <CR><LF> передать: «^0D^0A»

для отправки зарезервированного символа «^» передать: «^5E»

7.4 Значения символов

Когда в настоящем стандарте используются отдельные символы для определения единиц измерения, указания типа поля данных, типа предложения и т. д., они должны интерпретироваться в соответствии с таблицей значения символов в таблице 3.

A	Status symbol; Yes; Data Valid; Warning Flag Clear; Auto; Ampere; ASCII
a	Alphabet character variable A through Z or a through z
B	Bars (pressure, 1000 Mb = 1 Std. Atm. = 100kPa); Bottom
C	Celsius (Degrees); Course-up
c	Valid character; Calculating
D	Degrees (of Arc)
E	Error; East; Engine
F	Fathoms
f	Feet
G	Great Circle; Green
g	Good
H	Compass Heading; Head-up; Hertz; Humidity
h	Hours; HEX number
I	Inches
J	Input operation completed
K	Kilometers; km/hour
k	Kilograms
L	Left; Local; Lost Target
l	Latitude; Liters; Liters/second
M	Meters; Meters/second; Magnetic; Manual; Cubic Meters
m	Minutes; message
N	Nautical miles; Knots; North; North-up; Newton
n	Numeral; address
P	Purple; Proprietary (only when following $); Position sensor; Percent; Pascal (pressure)
Q	Query; Target-Being-Acquired
R	Right; Rhumb line; Red; Relative; Reference; Radar Tracking; Rev/min (RPM)
S	South; Statute miles; Statute miles/hour; Shaft; Salinity in parts per thousand
s	Seconds; Six bit number
T	Time difference; True; Track; Tracked-Target
t	Test
U	Dead Reckoning Estimate
u	Sign, if minus "-" (HEX 2D)
V	Data invalid; No; Warning Flag Set; Manual; Volt
W	West; Water; Wheelover
x	Numeric character variable
y	Longitude
Z	Time

Таблица 3. Таблица с перечнем отдельно используемых символов согласно стандарту.

8. Поле.

Поле состоит из строки допустимых символов или без символов (нулевое поле), расположенной между двумя соответствующие символы-разделители.

8.1 Поле адреса

Поле адреса является первым полем в предложении и следует за «$» или «!». разделитель, он служит для определения предложение. Разделитель "$" идентифицирует предложения, которые соответствуют общепринятым параметрическим правилам и правилам композиции полей с разделителями. Знак"!" разделитель идентифицирует предложения, которые соответствуют правилам инкапсуляции специального назначения и правилам композиции полей без разделителей. Символы в поле адреса ограничены цифрами и заглавными буквами. Поле адреса не должно быть пустым полем.

Передаются только предложения со следующими тремя типами адресных полей :

8.1.1 Поле утвержденного адреса

Утвержденные поля адреса состоят из пяти цифр и символов верхнего регистра, определенных настоящим стандартом. Первые два символа — это

идентификатор TALKER, указанные в таблице 4.

TALKER DEVICE	IDENTIFIER
Heading Track Controller (Autopilot): General	AG*
Magnetic	AP
Automatic Identification System	AI
COMMUNICATIONS: Digital Selective Calling (DSC)	CD*
Data Receiver	CR
Satellite	CS*
Radio-Telephone (MF/HF)	CT*
Radio-Telephone (VHF)	CV*
Scanning Receiver	CX*
DECCA Navigator	DE
Direction Finder	DF*
Electronic Chart System (ECS)	EC
Electronic Chart Display & Information System (ECDIS)	EI
Emergency Position Indicating Beacon (EPIRB)	EP*
Engine room Monitoring Systems	ER
GLONASS Receiver	GL
Global Navigation Satellite System (GNSS)	GN
Global Positioning System (GPS)	GP
HEADING SENSORS: Compass, Magnetic	HC*
Gyro, North Seeking	HE*
Gyro, Non-North Seeking	HN
Integrated Instrumentation	II
Integrated Navigation	IN
Loran C	LC
Proprietary Code	P
Radar and/or Radar Plotting	RA*
Sounder, depth	SD*
Electronic Positioning System, other/general	SN
Sounder, scanning	SS
Turn Rate Indicator	TI*
VELOCITY SENSORS: Doppler, other/general	VD*
Speed Log, Water, Magnetic	VM
Speed Log, Water, Mechanical	VW
Voyage Data Recorder	VR
Transducer	YX
TIMEKEEPERS, TIME/DATE: Atomic Clock	ZA
Chronometer	ZC
Quartz	ZQ
Radio Update	ZV
Weather Instruments	WI

Таблица 4. Таблица с перечнем индикаторов “TALKER”

Идентификатор “TALKER” служит для определения характера передаваемых данных. Устройства, способные передавать данные из нескольких

источников, должны передавать соответствующий идентификатор “TALKER” (например, устройство с приемником GPS и приемником Loran-C должно передавать GP, когда положение основано на GPS,

LC при местоположении Loran-C и IN для комплексной навигации следует использовать, если линии положения от Loran-C и GPS объединены в определение местоположения). Устройства, способные ретранслировать данные с других источники должны использовать соответствующий идентификатор (например, приемники GPS, передающие данные о курсе, не должны передавать $GPHCD, если направление по компасу фактически получено из сигналов GPS).

8.1.2 Поле адреса запроса

Адрес запроса состоит из пяти символов и используется для запроса передачи определенного предложения по отдельной шине от идентифицированного “TALKER”. Первые два символа — это идентификатор “TALKER” устройства, запрашивающего данные, следующие два символа — это идентификатор TALKER вызываемого устройства, а последний символ — это символ запроса «Q».

8.1.3 Поле собственного адреса

Поле собственного адреса состоит из фирменного символа «P», за которым следуют три символа. Мнемонический код изготовителя, используемый для идентификации ГОВОРИТЕЛЯ, выдающего собственное предложение, и любые дополнительные символы по мере необходимости Список действительных мнемонических кодов производителя содержится в Приложение III.

8.2 Поле данных

Поля данных в утвержденных предложениях следуют за разделителем «,» и содержат допустимые символы (и разделители кода «^») в соответствии с

форматами, показанными в таблице 5

Field Type	Symbol	Definition
Special Format Fields:		
Status	A	Single character field: A = Yes, Data Valid, Warning Flag Clear V = No, Data Invalid, Warning Flag Set
Latitude	llll.ll	Fixed/Variable length field: degreesminutes.decimal - 2 fixed digits of degrees, 2 fixed digits of minutes and a variable number of digits for decimal-fraction of minutes. Leading zeros always included for degrees and minutes to maintain fixed length. The decimal point and associated decimal-fraction are optional if full resolution is not required.
Longitude	yyyyy.yy	Fixed/Variable length field: degreesminutes.decimal - 3 fixed digits of degrees, 2 fixed digits of minutes and a variable number of digits for decimal-fraction of minutes. Leading zeros always included for degrees and minutes to maintain fixed length. The decimal point and associated decimal-fraction are optional if full resolution is not required.
Time	hhmmss.ss	Fixed/Variable length field: hoursminutesseconds.decimal - 2 fixed digits of hours, 2 fixed digits of minutes, 2 fixed digits of seconds and a variable number of digits for decimal-fraction of seconds. Leading zeros always included for hours, minutes and seconds to maintain fixed length. The decimal point and associated decimal-fraction are optional if full resolution is not required.
Defined field		Some fields are specified to contain pre-defined constants, most often alpha characters. Such a field is indicated in this standard by the presence of one or more valid characters. Excluded from the list of allowable characters are the following that are used to indicate field types within this standard: "A", "a", "c", "hh", "hhmmss.ss", "llll.ll", "x", "yyyyy.yy"

Таблица 5. Таблица с перечнем форматов допустимых для пустых символов.

Поля данных в частных предложениях содержат только допустимые символы и символы-разделители «,» и «^», но не определены настоящим стандартом. Из-за наличия переменных полей данных и пустых полей определенные поля данных должны располагаться внутри предложения только с соблюдением разделителей полей ",". Поэтому для “LISTENER” важно находить поля,

подсчитывая разделители, а не подсчитывая общее количество символов, полученных с самого начала предложения

8.2.1 Поля с переменной длиной

Хотя некоторые поля данных имеют фиксированную длину, многие из них имеют переменную длину, чтобы позволить устройствам передавать информацию и предоставлять данные с большей или меньшей точностью в соответствии с возможностями или требованиями конкретного устройства. Поля переменной длины могут быть буквенно-цифровыми или числовыми полями. Переменные числовые поля могут содержать десятичную точку и могут содержать начальные или конечные «нули».

8.2.2 Типы данных полей

Поля данных могут быть буквенными, числовыми, буквенно-цифровыми, переменной длины, фиксированной длины, фиксированной/переменной (с частью фиксированной длины, в то время как остальная часть варьируется). Некоторые поля являются постоянными, их значение определяется

конкретное определение фразы. Допустимые типы полей приведены в таблице 6 со сводкой по типам полей.

Field Type	Symbol	Definition
Numeric Value Fields:		
Variable numbers	x.x	Variable length integer or floating numeric field. Optional leading and trailing zeros. The decimal point and associated decimal-fraction are optional if full resolution is not required. (example: 73.10 = 73.1 = 073.1 = 73)
Fixed HEX field	hh___	Fixed length HEX numbers only, MSB on the left.
Variable HEX field	h--h	Variable length HEX numbers only, MSB on the left.
Fixed Six bit field	ss___	Fixed length Six bit coded characters only. See Table 7 and Figures 3 & 4 for field conversions.
Variable Six bit field	s--s	Variable length Six bit coded characters only. See Table 7 and Figures 3 & 4 for field conversions.
Information Fields:		
Variable text	c--c	Variable length valid character field.
Fixed alpha field	aa___	Fixed length field of upper-case or lower-case alpha characters
Fixed number field	xx___	Fixed length field of numeric characters
Fixed text field	cc___	Fixed length field of valid characters

NOTES:

1. Spaces shall only be used in variable text fields.

2. A negative sign "-" (HEX 2D) is the first character in a Field if the value is negative. When used, this increments the specified size of fixed length fields by one. The sign is omitted if the value is positive.

3. Units of measure fields are appropriate characters from the Symbol Table (Table 3) unless a specific unit of measure is indicated.

4. Fixed length field definitions show the actual number of characters. For example, a field defined to have a fixed length of 5 HEX characters is represented as hhhhh between delimiters in a sentence definition.

Таблица 6. Таблица с перечнем допустимых типов полей.

8.2.3 Пустые поля

Нулевое поле — это поле нулевой длины, т. е. в поле не передаются никакие символы. Нулевые поля должны использоваться, когда значение ненадежно или недоступно. Например, если информация о заголовке недоступна, отправка данных «000» вводит в заблуждение, поскольку пользователь не может отличить «000», означающее отсутствие данных, от допустимого заголовка «000». Однако пустое поле без каких-либо символов явно указывает на то, что данные не передаются. Нулевые поля с их разделителями могут иметь следующий вид в зависимости от того, где они расположены в предложении: ",," ",*" Символ ASCII NULL (HEX 00) не должен использоваться в качестве нулевого поля.

8.3 Поле контрольных сумм

Поле контрольной суммы должно передаваться во всех предложениях. Поле контрольной суммы является последним полем в предложении и следует за символом-разделителем контрольной суммы "*".Контрольная сумма представляет собой 8-битное исключающее ИЛИ (без начальных или стоповых битов) всех символов в предложении, включая разделители "," и "^", между символами "$" или "!", но не включая их. и разделители "*". Шестнадцатеричное значение старших и наименее значащих 4 битов результата преобразуется в два символа ASCII (0-9, A-F (верхний регистр)) для передачи. Самый значащий символ передается первым. Примеры использования поля контрольной суммы:
$GPGLL,5057.970,N,00146.110,E,142451,A*27<CR><LF> ;
$GPVTG,089.0,T,,,15.2,N,,*7F<CR><LF>

8.4 Поле идентификатора последовательного сообщения

Это поле имеет решающее значение для идентификации групп из 2 или более предложений, составляющих сообщение, состоящее из нескольких предложений. Это поле увеличивается каждый раз, когда новое сообщение, состоящее из нескольких предложений, создается с помощью того же модуля форматирования предложений. Значение сбрасывается на ноль, когда оно превышает определенное максимальное значение.

При создании макета малогабаритного подводного аппарата для GPS модуля был выбран модуль “Trema GPS модуль ATGM336H”. Это навигационное устройство которое позволяет определить координаты по широте, долготе и высоте. В качестве дополнительных функций модуль может предоставить данные о текущей дате , времени, скорости и направления движения.

Модуль получает данные на основе информации поступающей со спутников навигационных систем. GPS модуль самостоятельно обрабатывает полученную информацию и передает данные по шине UART. Данные передаются в виде сообщений в текстовом формате по протоколу NMEA 0183. Данный модуль отличается низким энергопотреблением и высокой чувствительностью.

Основные характеристики модуля :

- Напряжение питания: 3,3 В или 5 В, поддерживаются оба напряжения.
- Питание резервной батареи: 3 В.
- Ток потребляемый модулем: до 25 мА.
- Интерфейс: UART.
- Скорость шины UART: 4800, 9600, 19200, 38400, 57600, 115200 бит/с.
- Конфигурация шины UART: 8 бит данных, без проверки четности, с одним стоповым битом.
- Протокол передачи данных: NMEA 0183
- Частота обновления выводимых данных: от 1 до 10 Гц.
- Поддерживаемые навигационные системы: GPS (США), Глонасс (Россия) и Beidou (Китай).
- Точность позиционирования: < 2 м.
- Точность измерения скорости: < 0,1 м/с.
- Максимальная высота: 18000 м.
- Рабочая температура: от -40 до +85 °C.

Для настройки модуля нами использовалось две библиотеки :

Библиотека iarduino_GPS_ATGM336 позволяет настроить работу Trema GPS модуля ATGM336H (задать скорость UART, частоту обновления данных, версию NMEA 0183, настроить состав выводимых сообщений, выбрать спутниковые навигационные системы, указать динамическую модель, перезагрузить модуль или сохранить его настройки).
Библиотека iarduino_GPS_NMEA позволяет получать данные из текстовых сообщений NMEA 0183 отправляемых GPS-модулем по шине UART (получить широту, долготу, высоту, скорость, курс, дату, время, количество спутников и данные о них).

Так же данный модуль имеет следующие возможности :

1. Получать широту, долготу и высоту над уровнем моря.
2. Получать скорость и курс на истинный полюс.
3. Получать дату и время UTC.
4. Получать количество наблюдаемых спутников и спутников участвующих в позиционировании.
5. Получать данные о спутниках: уровень приёма, положение и тип навигационной системы.
6. Получать геометрические факторы ухудшения точности и точность позиционирования.
7. Определять ошибку определения координат, даты, времени, скорости и курса.
8. Настраивать скорость передачи данных по шине UART.
9. Настраивать частоту обновления выводимых данных.
10. Выбирать версию протокола NMEA 0183 для формирования отправляемых сообщений.
11. Настраивать состав сообщений NMEA 0183.
12. Выбирать спутниковые навигационные системы, данные которых требуется получать.
13. Выбирать динамическую модель навигационной платформы.
14. Перезагружать модуль с выбором типа старта (холодный / горячий).
15. Перезагружать модуль со сбросом настроек в заводские.
16. Сохранять настройки в энергонезависимую память модуля.

Настройка модуля происходит в среде программирования Arduino IDE на языкеC+\ C++\C#.
Для первичной настройки и для первого получения координат долготы и широты 5 раз в секунду, делаем следующие действия :

1.Подключаем библиотеки iarduino_GPS_ATGM336 и iarduino_GPS_.
Задаем цель для gps. Для работы с функциями и методами библиотеки iarduino_GPS_NMEA. Используем команду: iarduino_GPS_NMEA gps;
Объявляем объект SettingsGPS для работы с библиотекой iarduino_GPS_ATGM336 с помощью команды: iarduino_GPS_ATGM336 SettingsGPS; как представлено на рисунке 3.1.3.

```
include <iarduino_GPS_NMEA.h>
#include <iarduino_GPS_ATGM336.h>

iarduino_GPS_NMEA    gps;
iarduino_GPS_ATGM336 SettingsGPS;
```

Рисунок 3.1.3. Начало работы с датчиком GPS

2. В основном цикле программы выполняем следующие действия как на рисунке 3.1.4. :

1. Инициировать работу библиотек
2. Инициировать работу с GPS.
3. Инициировать расшифровку строк NMEA

```
void setup(){
     Serial.begin(9600);
     SettingsGPS.begin(Serial1);
     gps.begin(Serial1);
```

Рисунок 3.1.4. Начало основного цикла программы.

3. Переходим к настройке модуля как указано на рисунке 3.1.5 Для настройки модуля нам необходимо :

1. Установить скорость передачи данных
2. Установить что данные нужно получать от спутников:
3. Установить что каждый пакет данных NMEA должен содержать только одну: SettingsGPS.composition(NMEA_RMC);
4. Установить что модуль используется в портативном устройстве:
5. Установить обновлять данные 10 раз в секунду

```
     SettingsGPS.baudrate(9600);
     SettingsGPS.system(GPS_GP, GPS_GL);
     SettingsGPS.composition(NMEA_RMC);
     SettingsGPS.model(GPS_PORTABLE);
     SettingsGPS.updaterate(10);
}
```

Рисунок 3.1.5 Настройка модуля.

4. Создаем цикл для чтения данных “void loop ( ) {“ как на рисунке 3.1.6 Следующим шагом будет начало чтения данных, затем проверка достоверности координат. Затем необходимо установить условие, что если данные не прочитаны или координаты не достоверны. Если координаты не достоверны, тогда выводим сообщение об ошибке, ждем 2 секунды и выполняем функцию “loop () ” с начала.

```
void loop(){
     gps.read();
     if(gps.errPos){
       Serial.println("Координаты недостоверны");
       delay(2000); return;
     }
```

Рисунок 3.1.6 Цикл “void loop”

5. Следующим шагом будет вывод координат как на рисунке 3.1.7. Вывод координат мы начинаем с широты, а потом долгота. Все показания выводятся в градусах.

```
Serial.print("Широта: ");
Serial.print(gps.latitude, 5);
Serial.print("°, ");
Serial.print("Долгота: ");
Serial.print(gps.longitude, 5);
Serial.println("°.");
```

Рисунок 3.1.7. Вывод координат широты и долготы в градусах.

Следует обратить внимание что не указывается скорость шины Serial имеет автоматическую настройку относительно скорости модуля с помощью функции SettingsGPS.begin(Serial); , а затем устанавливается функцией SetingsGPS.baudrate(9600);

Состав сообщений которые отправляет модуль , урезан из-за обращения к функции SettingsGPS.composition(); до одной строки с идентификатором NMEA_RMC. Вместо данного идентификатора можно указать “NMEA_GGA” так как в их строках так же содержится информация о широте и долготе.

Получение скорости и курса.

1. Для получения скорости и курса, нам необходимо использовать программный код. Для начала работы с GPS модулем мы инициализируем библиотеку которая отвечает за расшифровку строк протокола NMEA которые мы получаем по UART. Подключаем библиотеку для настройки параметров работы GPS #include <iarduino_GPS_ATGM336.h>Объявляем

объект gps : iarduino_GPS_NMEAgps; Объявляем объект: iarduino_GPS_ATGM336 SettingsGPS как на рисунке 3.1.8.

```
#include <iarduino_GPS_NMEA.h>
#include <iarduino_GPS_ATGM336.h>

iarduino_GPS_NMEA    gps;
iarduino_GPS_ATGM336 SettingsGPS;
```

Рисунок 3.1.8. Начальные команды перед работой с модулем.

2. Основной цикл

Основной цикл программы представлен на рисунке 3.1.9.

Первым действием будет инициализация работы библиотек. Начинаем работу с UART, это необходимо для вывода данных на монитор последовательного порта, при этом скорость вывода составляет 9600 бит/сек. Далее начинаем работу с GPS модулем по указанной шине UART. Данная функция должна сама определять следующие параметр: текущую скорость GPS модуля ATGM336. Запускаем расшифровку строк NMEA, не забывая указывать объект для используемой шины UART;

```
void setup(){
    Serial.begin(9600);
    SettingsGPS.begin(Serial1);
    gps.begin(Serial1);
```

Рисунок 3.1.9. Настройка основного цикла.

3. Начинаем настраиваем работу модуля как на рисунке 3.1.10.

Задаем скорость для передачи данных ,а так же скорость работы шины в 9600 бит/сек. Необходимо указать, что данные будут приходить от спутников навигационных систем GPS (GPS_GP) и Glonass (GPS_GL). Задаем параметр что каждый пакет данных NMEA должен содержать только одну строку с индикатором скорости и курса, относительно к земле. Затем, указываем что модуль используется в дроне. Задаем установку для обновления данных 10 раз в секунду. Стоит отметить что команда “gps.read()” имеет скорость чтения данных в 2 раза медленнее чем их выводит модуль.

```
SettingsGPS.baudrate(9600);
SettingsGPS.system(GPS_GP, GPS_GL);
SettingsGPS.composition(NMEA_VTG);
SettingsGPS.model(GPS_DRONE);
SettingsGPS.updaterate(10);
```

Рисунок 3.1.10 Настройка модуля.

4. Основной цикл программы и код для вывода курса и скорости представлен на рисунке 3.1.11.

Первое с чего начинаем это чтение данные, затем проверяем достоверность скорости и курса. Если данные не могут быть обработаны или прочитаны (gps.errCrs=1) или же скорость,курс недостоверны (gps.errCrs=2), то выводим сообщение об ошибке. Ждём 2 секунды и выполняем функцию loop() с начала. Проверяем достоверность скорости и курса. Выводим текущие скорость и курс:

Выводим скорость в м\с.

```
Serial.print(gps.speed);
Serial.print("м\с. ");
Serial.print("Курс: ");
```

Выводим курс в градусах.

```
Serial.print(gps.course);
Serial.println("°.\r\n");
```

```
void loop(){

    gps.read();

    if(gps.errCrs){
      Serial.println("Скорость недостоверна");
      delay(2000); return;
    }
    Serial.print("Скорость: ");
    Serial.print(gps.speed);
    Serial.print("км/ч. ");
    Serial.print("Курс: ");
    Serial.print(gps.course);
    Serial.println("°.\r\n");
}
```

Рисунок 3.1.11. Основной цикл программы и вывод курса и скорости

Следует обратить внимание что не указывается скорость шины Serial имеет автоматическую настройку относительно скорости модуля с помощью функции SettingsGPS.begin(Serial); , а затем устанавливается функцией SetingsGPS.baudrate(9600);

Если уменьшать строки входящие в состав сообщения NMEA , это приведет к уменьшению количества данных содержащихся в нём. Например, строка с идентификатором NMEA_VTG содержит информацию только о скорости и курсе, без координат, даты, времени и т.д.

Получение данных о текущей дате и времени

1.Подключаем библиотеки как на рисунке 3.1.12. Необходимо определить массив строк в которых содержится по две первых буквы из названий дня недели.

```
#include <iarduino_GPS_NMEA.h>
#include <iarduino_GPS_ATGM336.h>

iarduino_GPS_NMEA    gps;
iarduino_GPS_ATGM336 SettingsGPS;

char* wd[]={"Вс","Пн","Вт","Ср","Чт","Пт","Сб"};
```

Рисунок 3.1.12. Подключение библиотек и определение массива строк.

2.Инициализация системы

Начинаем работу с инициализации работы библиотек как на рисунке 3.1.13. Затем инициализируем работу шины UART для вывода данных на последовательный порт, при этом скорость вывода составляет 9600 бит/сек. Инициализируем работу GPS модуля по указанной шине UART. Функция должна сама определить текущую скорость модуля. Следующей командой запускаем расшифровку строк NMEA указав объект используемой шины UART.

```
void setup(){
    Serial.begin(9600);
    SettingsGPS.begin(Serial1);
    gps.begin(Serial1);
```

Рисунок 3.1.13. Начало работы с модулем.

3.Настраиваем работу модуля:

1. Настройка модуля GPS начинается как на рисунке 3.1.14 с установки скорость передачи данных модулем и скорость работы шины Serial1 в 9600 бит/сек. Указываем что данные должны приходить на модуль от навигационных систем GPS (GPS_GP) и Glonass (GPS_GL). Далее следует указать что каждый пакет данных NMEA должен содержать только одну строку с идентификатором информации о дате и времени. Указываем что модуль используется стационарно. Задаем частоту обновления данных 10 раз в секунду. Стоит отметить , что функция gps.read() имеет скорость чтения данных в 2 раза меньше чем скорость вывода.

```
    SettingsGPS.baudrate(9600);
    SettingsGPS.system(GPS_GP, GPS_GL);
    SettingsGPS.composition(NMEA_ZDA);
    SettingsGPS.model(GPS_STATIC);
    SettingsGPS.updaterate(10);
}
```

Рисунок 3.1.14. Первоначальная настройка модуля.

4. Основной цикл

Основной цикл начинаем с Чтения данных как на рисунке 3.1.15. Затем идет отображение времени (час, минуты, секунды). Отображение даты (день, месяц, год). Отображение дня недели. Данные об ошибках и количество секунд со страта Unix. Затем завершаем строку.

```
void loop(){                                        //
     gps.read();                                    //
     Serial.print(gps.Hours  ); Serial.print(":"); //  Выводим час.
     Serial.print(gps.minutes); Serial.print(":"); //  Выводим минуты.
     Serial.print(gps.seconds); Serial.print(" "); //  Выводим секунды.
     Serial.print(gps.day    ); Serial.print("."); //  Выводим день месяца.
     Serial.print(gps.month  ); Serial.print("."); //  Выводим месяц.
     Serial.print(gps.year   ); Serial.print("г.");//  Выводим год.
     Serial.print(" (");                            //
     Serial.print(wd[gps.weekday]);                 //  Выводим день недели.
     Serial.print("), ");                           //
     Serial.print("UnixTime: ");                    //
     Serial.print(gps.Unix);                        //  Выводим время UnixTime.
     Serial.print("с.");                            //
     if( gps.errTim ){                              //
         Serial.print(" Время недостоверно.");      //  Выводим информацию о недостоверном времени.
     }                                              //
     if( gps.errDat ){                              //
         Serial.print(" Дата недостоверна.");       //  Выводим информацию о недостоверной дате.
     }                                              //
     Serial.print("\r\n");                          //
}                                                   //
```

Рисунок 3.1.15. Программный код основного цикла для вывода данных.

Получение данных о спутниках

Данный программный код выводит данные о доступных для связи спутниках, если быть точнее то выводится их ID, уровень приёма, положение относительно GPS-модуля, тип навигационной системы , а так же частота вывода данных 1 раз в 2 секунды.

1. Начало работы с модулем

Инициализируем библиотеки которые отвечают за построчную расшифровку протокола NMEA, данные которые приходят по UART и для настройки параметров работы GPS модуля ATGM336 как на рисунке 3.1.16. Объявляем выбранный объект для работы с библиотекой iarduino_GPS_NMEA , для использования функций и методов смой библиотеки. Задаем объект SettingsGPS , который необходим для использования функций и методов библиотеки iarduino_GPS_ATGM336. Задаем массив в котором будут данные о 30 спутниках в формате: {ID спутника (1...255), Отношение сигнал/шум в дБ, тип навигационной системы Угол возвышения спутника (0°-горизонт ... 90°-зенит), Азимут положения спутника (0°...255°), Остаток азимута до 360°. Определяем массив строк содержащих названия навигационных систем спутников.

```
#include <iarduino_GPS_NMEA.h>
#include <iarduino_GPS_ATGM336.h>

iarduino_GPS_NMEA    gps;
iarduino_GPS_ATGM336 SettingsGPS;

uint8_t i[30][7];
char* sa[]={"NoName ","GPS    ","Глонасс"};
```

Рисунок 3.1.16. Начало работы с модулем.

2. Инициализация системы и библиотек

Начинаем инициализацию работу шины UART для вывода данных на последовательный порт со скоростью вывода и приема данных 9600 бит/сек как на рисунке 3.1.17. Затем инициируем работу GPS модуля по шине UART. Функция сама определит текущую скорость для модуля ATGM336. Производим настройку модуля. Задаем скорость передачи данных и скорость работы шины в 9600 бит/сек. Указываем что данные нужно получать от спутников навигационных систем. Указываем что каждый пакет данных NMEA должен содержать только строки идентификатора информации активных спутников и информации о доступных Указываем что модуль используется в портативном устройстве. Задаем частоту обновления данных 1 раз в секунду. Стоит отметить , что функция gps.read() имеет скорость чтения данных в 2 раза меньше чем скорость вывода.

```
void setup(){
     Serial.begin(9600);
     SettingsGPS.begin(Serial1);
     gps.begin(Serial1);
     SettingsGPS.baudrate(9600);
     SettingsGPS.system(GPS_GP, GPS_GL);
     SettingsGPS.composition(NMEA_GSA, NMEA_GSV);
     SettingsGPS.model(GPS_PORTABLE);
     SettingsGPS.updaterate(1);
}
```

Рисунок 3.1.17. Инициализация системы и библиотек.

3.Основной цикл программы

Начинаем работу с чтения данных как на рисунке 3.1.18. Читаем данные о доступных спутниках и записываем данные по этим спутникам в массив i , так же стоит отметить что чтение данных может занимать свыше ожной секунды. При указании второго параметра Если указать второй параметр “gps.read(i,true);” то в выше указанном массиве будут записаны данные только о тех спутниках, которые участвуют при процессе позиционирования.

Далее происходит вывод информации по спутникам. Проходим по всем элементам массива «i». Если спутник имеет идентификационный номер, то выводим информацию о нём. Выводим ведущий 0. Выводим номер строки в мониторе последовательного порта. Выводим название навигационной системы спутника. Выводим ведущий 0. Выводим ID спутника. Выводим ведущий 0. Выводим отношение сигнал/шум (SNR) в дБ. Выводим ведущий 0. Выводим возвышение спутника относительно модуля (0°-горизонт ... 90°-зенит). Выводим ведущий 0. Выводим ведущий 0.Выводим азимут нахождения спутника относительно модуля (0...360°).

Состав сообщения отправленных модулем NMEA 0183, урезан из-за обращения к функции SettingsGPS.composition(); до строк с идентификаторами NMEA_GSA и NMEA_GSV. Из-за уменьшения количества строк в входящих в сообщение , уменьшается и количество принимаемых данных. К примеру, строки с идентификаторами NMEA_GSA и NMEA_GSV содержат только информацию о спутниках и геометрические факторы ухудшения точности.

```
void loop(){
     gps.read(i);
     for(uint8_t j=0; j<30; j++){
        if( i[j][0] ){
            if(j<9) Serial.print(0);
            Serial.print(j+1);
            Serial.print(") Спутник ");
            Serial.print(sa[i[j][2]]);
            Serial.print(" ID: ");
            if(i[j][0]<10) Serial.print(0);
            Serial.print(i[j][0]);
            Serial.print(". Уровень: ");
            if(i[j][1]<10) Serial.print(0);
            Serial.print(i[j][1]);
            Serial.print("дБ. Возвышение: ");
            if(i[j][4]<10) Serial.print(0);
            Serial.print(i[j][4]);
            Serial.print("°. Азимут: ");
            if(i[j][5]<100) Serial.print(0);
            if(i[j][5]<10) Serial.print(0);
            Serial.print(i[j][5]+i[j][6]);
            Serial.print("°. ");
            if(i[j][3]){Serial.print("Участвует в позиционировании.");}
            Serial.print("\r\n");
        }
```

Рисунок 3.1.18. Основной цикл программы для получения данных о спутниках.

3.2 Программно-графическая части протокола приема передачи данных "NMEA"

При приеме сигнала с приема передатчика GPS/DGPS по методу передачи NMEA мы можем наблюдать не расшифрованные сообщения которые содержат в себе строчки с данным, которые включают в себя набор букв, цифр и символов. Каждый символ отвечает за то или иное обозначение. В таблице ниже приводятся примеры взаимодействия с каждым навигационным датчиком.

Датчик	Тип данных, передаваемых датчиком	Сообщения поддерживающие передаваемые	Канал в System Configuration Utility/Service/Sensors/Accu
GPS (GLONASS, DECCA, LORAN)	Position (Lat. & Lon.)	GGA GLL RMC SNU	Pos. system 1 (Pos. system 2)
	COG & SOG	RMC VTG	Pos. system 1 (Pos. system 2)
	atellite informatio	GGA	Pos. system 1 (Pos. system 2)
	Datum	DTM	Pos. system 1 (Pos. system 2)
	ORAN-C data statu	SNU	Pos. system 1 (Pos. system 2)
	UTC time	ZDA ZLZ ZZU GGA GLL	Time
	UTC date	ZDA	Time
ARPA	Targets	TTM	ARPA_A (ARPA_B)
	VRM's EBL's Cursor	RSD	ARPA_A (ARPA_B)
	Heading	OSD VHW	Compass
	Speed through the water	OSD VHW	Log
Sounder	Depth under the sounder vibrator	DBT DPT DBK DBS	Sounder
Compass	Heading	HDT HDM OSD VHW	Compass
Speed log (dopler log)	Dpeed through the water	OSD VHW VBW	Log
	SOG	VBW	Log
Digital anemometer	Wind direction speed	VWR MWV VWT	Wind
Digital thermometer	Water temperature	MTW	Temperature
Yeoman digitizer	WP position	WPL	Yeoman

В нашей программе при получении кода, сообщения NMEA выглядят как на рисунке 3.2.1. В данном разделе будет описана функциональная связь с приемом данных от навигационных датчиков , поддерживающих стандарт NMEA-0183.

Ниже так же приводится расшифровка сообщений NMEA-0183

1. DBK (Depth Below Keel) – глубина воды под килем. Пример сообщения “$--DBK, x.x1, f2, x.x3, M4, x.x5, F6*hh<CR><LF>.” и его расшифровка

N	Поле	Название	Значение
1	x.x	Глубина	Значение глубины
2	f	Единица измерения	"f"-футы
3	x.x	Глубина	Значение глубины
4	M	Единица измерения	"m"-метры
5	x.x	Глубина	Значение глубины
6	F	Единица измерения	"f"-сажень

2. DBS (Depth Below Surface) – глубина воды относительно водной поверхности. Сообщение и его расшифровка - $--DBS, x.x1, f2, x.x3, M4, x.x5, F6*hh<CR><LF>.

N	Поле	Название	Значение
1	x.x	Глубина	Значение глубины
2	f	Единица измерения	"f"-футы
3	x.x	Глубина	Значение глубины
4	M	Единица измерения	"m"-метры
5	x.x	Глубина	Значение глубины

3. DBT (Depth Below Transducer)– глубина, измеренная от вибратора эхолота. Сообщение и его расшифровка - $--DBT,x.x1,f2,x.x3,M4,x.x,F6*hh<CR><LF>.

N	Поле	Название	Значение
1	x.x	Глубина	Значение глубины
2	f	Единица измерения	"f"-футы
3	x.x	Глубина	Значение глубины
4	M	Единица измерения	"m"-метры
5	x.x	Глубина	Значение глубины

4. DPT (Depth) – глубина. Содержание включает в себя значение глубины, измеренной от вибратора эхолота, и значение поправки за расположение вибратора. Положительная поправка означает расстояние от вибратора до поверхности воды, отрицательная – расстояние от вибратора до киля. $--DPT, x.x1, x.x2, x.x3*hh<CR><LF>.

N	Поле	Название	Значение	Примичание
1	x.x	Глубина (в метрах)		
2	x.x	Поправка (в метрах)		не обрабатывается
3	x.x	Maximum range scale in use		не обрабатывается

5. DTM (Datum reference) – данные о геодезической основе. В кодировке содержится информация о референц-эллипсоиде, к которому отнесены принимаемые координаты. $--DTM, ccc1, a2, x.x3, a4, x.x5, a6, x.x7, ccc8*hh<CR><LF>.

N	Поле	Название	Значание	Примичание
1	ccc	Local datum	“W72” – WGS 72 “W84” – WGS 84 “IHO” – референц-эллипсоид по терминологии IHO “999” – референц-эллипсоид, определенный пользователем “S85” – SGS85 “P90” – PE90	Обрабатывается только “W84”
2	a	Local datum subdivision code		не обрабатывается
3	x.x	Lat offset, min		не обрабатывается
4	a	Hemisphere	“N” – North; “S” –South	не обрабатывается
5	x.x	Lon offset, min		не обрабатывается
6	a	Hemisphere	“E” – East; “W” – West	не обрабатывается
7	x.x	Altitude offset, m		не обрабатывается
8	ccc	Reference datum	“W84” – WGS84 “W72” – WGS72 “S85” – SGS85 “P90” – PE90	не обрабатывается

6. GGA (Global Positioning System Fix Data) – данные о координатах по gps. Состав принимаемого сообщения включает в себя: время, координаты и сопутствующие данные от приемоиндикатора.
GPS. $--GGA, hhmmss.ss1, llll.ll2, a3, yyyyy.yy4, a5, x6, xx7, x.x8, x.x9, M10, x.x11, M12, x.x13, xxxx14*hh<CR><LF>.

N	Поле	Название	Значение	Примичание
1	hhmmss.ss	Время UTC обсер-вации	Часы, минуты, секунды	не обрабатывается в ns 3000 ecdis
2	llll.ll	Широта маршрут-ной	Градусы, минуты, десятые доли	
3	a	Полушарие	“N” – северное; “S” – южное	
4	yyyyy.yy	Долгота маршрут-ной	Градусы, минуты, десятые доли	
5	a	Полушарие	“E” – восточное; “W” – западное	
6	x	Индикатор качес-тва обсервации GPS	“0” – Обсервация невозможна или недействительна; “1” – Обсервация действительна, GPS тип SPS; “2” – Обсервация действительна, дифф. GPS тип SPS “3” – Обсервация действительна, GPS тип PPS “4” – Обсервация действительна, RTK	значение “4” не обрабатывается
7	xx	Количество исполь-зуемых спутников	от 0 до 12	
8	xx	Горизонтальное ослабление точ-ности		
9	x.x	Высота антенны от уровня моря		не обрабатывается
10	m	Единицы измерения	“M” – метры	не обрабатывается
11	x.x	Геоидальная сепарация	Вертикальная разность между эллипсоидом “WGS-84” и уров-нем моря (геоидом), “минус” – уровень моря ниже эллипсоида	не обрабатывается
12	m	Единицы измерения	“M” – метры	не обрабатывается
13	x.x	Возраст данных дифф. gps (в секундах)		
14	xxxx	Код станции дифф. gps	от 0000 до 1023	

7. GLL (Geographic Position – Latitude/Longitude) – географические координаты.Состав сообщения состоит из кодировки координат текущего МС, время обсервации и статус данных.
$--GLL, llll.ll1, a2, yyyyy.yy3, a4, hhmmss.ss5, A6, a7*hh<CR><LF>.

4	a	Полушарие	“E” – восточное; “W” – западное			
5	hhmmss.ss	Время UTC	Часы, минуты,	не обрабат		
6	a	Статус данных, по-лученных от прие-мо индикатора	“A” – данные надежные; “V” – данные ненадежные			
7	a	Positioning system mode indicator	“A” – Autonomous; “D” – Differential; “E” – Estimated (dead reckoning); “M” – Manual input; “S” – Simulator; “N” – Data not valid	не обрабатывается		

8. HDM (Heading, Magnetic) – магнитный курс в градусах.
$--HDM, x.x1, M2*hh<CR><LF>.

N	Поле	Название	Значение	Примичание
1	x.x	Курс		
2	m	Тип	“V” – магнитный	

9. HDT (Heading, True) – курс истинный.
$--HDT, x.x1, T2*hh<CR><LF>.

N	Поле	Название	Значение	Примичание
1	x.x	Курс (в градусах)		
2	t	Тип курса	“T” – истинный	

10. MTW (Water Temperature) – температура воды.
$--MTW, x.x1, C2*hh<CR><LF>.

N	Поле	Название	Значение	Примичание
1	x.x	Температура		
2	c	Единицы измерения	“C” – градусы цельсия	

11. MWV (Wind Speed and Angle) – скорость и направление ветра.
$--MWV, x.x1, a2, x.x3, a4, A5*hh<CR><LF>.

N	Поле	Название	Значение	Примичание
1	x.x	Направление	в градусах	
2	a	Тип ветра	“R” – относительный;	
3	x.x	Скорость ветра		
4	a	Единицы измерения	“K” – км/час; “N” – узлы; “M” – м/сек.	
5	a	Статус данных	“A” – данные надежные; “V” – данные ненадежные	

12. OSD (Own Ship Data) – данные своего судна. Состав принимаемого сообщения включает в себя: курс и путевой угол, статус данных, скорость, метод ее получения и единицы измерения, направление и скорость суммарного сноса.
$--OSD, x.x1, A2, x.x3, a4, x.x5, a6, x.x7, x.x8, a9*hh<CR><LF>.

N	Поле	Название	Значение	Примечание		
1	х.х	Курс истинный относи-тельно воды (в гра-дусах)				
2	а	Статус данных	“A” – данные надежные; “V” – данные ненадежные			
3	х.х	Курс истинный от-носительно грунта (в градусах)		не обрабатывается		
4	а	Тип ввода курса		не обрабатывается		
5	х.х	Значение скорости				
6	а	Метод получения и/ или тип скорости	“D” – скорость отно-сительно грунта; “M” – скорость, вве-денная вручную; “W” – скорость относительно воды; “R” – скорость, опре-деленная по захва-ченной сарп цели; “P” – скорость, опре-деленная системой позиционирования			
7	х.х	Истинное направле-ние		не обрабатывается		

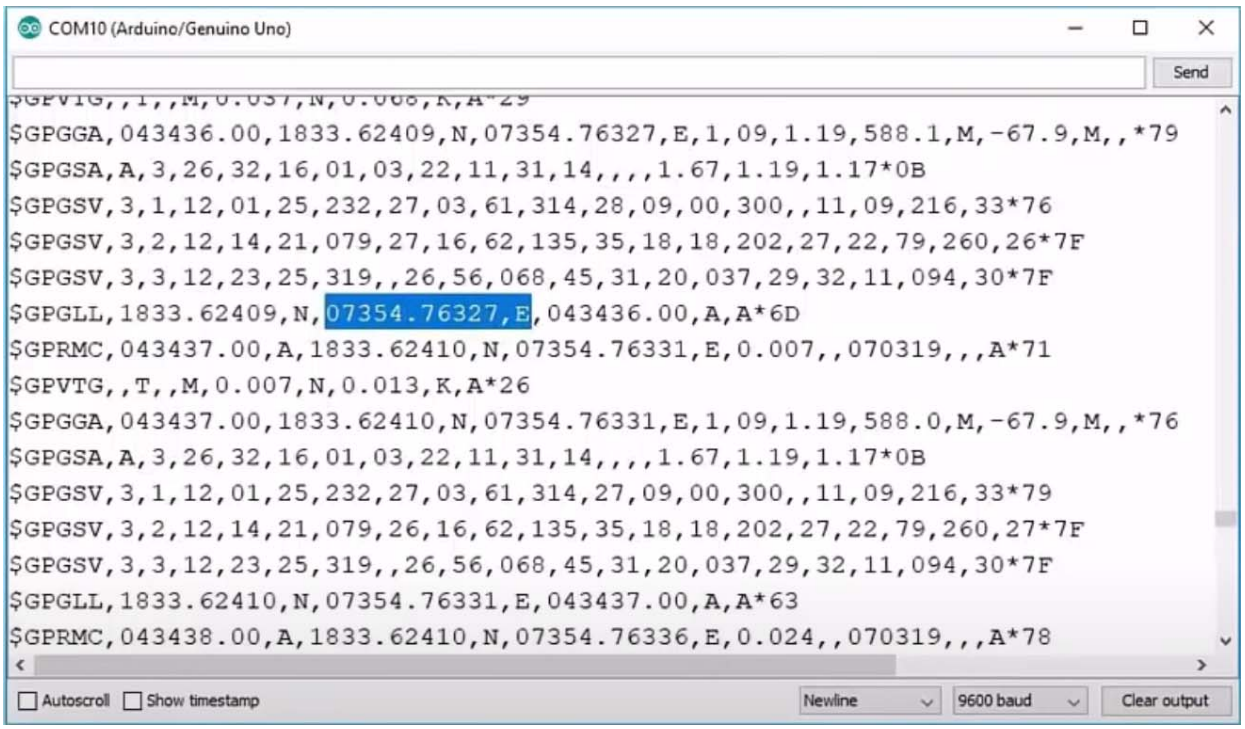


Рисунок. 3.2.1 пример полученных сообщений NMEA

Таким образом. При получении сигнала NMEA, опираясь на выше приведенные таблицы, мы можем расшифровать то или иное сообщение в удобный для понимания формат. Для программирования данные сообщения переводить не обязательно. В аппаратной части они обрабатываются и при использовании графического интерфейса выводятся автоматически в упрощенной форме для пользователя.

3.3 Программный обеспечение графического интерфейса для обработки показаний местоположения малогабаритного подводного аппарата.

GUI (Graphical User Interface) или ГИП (графический интерфейс пользователя) — это одна из разновидностей пользовательских интерфейсов, элементы которого выполнены в виде графических изображений. То есть все основные объекты, присутствующие в этом интерфейсе — иконки, функциональные кнопки, объекты меню и т.д. — выполнены в виде изображений.

Если сравнить GUI с обычной командной строкой, то в первом варианте перед пользователем открывается полный доступ к абсолютно всем элементам, который он видит на дисплее. Реализовать этот доступ можно с

использованием разных устройств ввода: оптической мыши, трекбола, клавиатуры, джойстика и пр.

Обычно в GUI каждый графический объект передает смысл функции с помощью понятного образа, чтобы пользователю было проще разобраться с определенным программным обеспечением и легче взаимодействовать с ОС в целом. Но важно понимать, что GUI — это лишь составная часть графического интерфейса. Функционирует он на уровне визуализации данных и таким же образом взаимодействует с пользователем.

Разновидности GUI

- *Трехмерные* — 3D-объекты интерфейса, транслирующиеся на дисплей в виде трехмерных графических элементов.
- *Простые* — обычные дисплейные формы и штатные объекты интерфейса, работа которых обеспечивается подсистемой ГИП.
- *Двухмерные* — уникальные интерфейсные элементы и изображения, которые были созданы сторонней библиотекой или внутренними средствами какой-либо программы.

Одно из ключевых требований к хорошему GUI — реализация концепции DWIM (Do What I Mean или дословно «делай то, что я имею в виду»). То есть система должна функционировать предсказуемо, чтобы пользователь интуитивно понимал, что произойдет после его определенного действия (ввода команды).

Для создания макета графического интерфейса для малогабаритного подводного аппарата использовалась программа Visual studio. Visual Studio — это стартовая площадка для написания, отладки и сборки кода, а также последующей публикации приложений. Помимо стандартного редактора и отладчика, которые есть в большинстве сред IDE, Visual Studio включает в себя компиляторы, средства автозавершения кода, графические конструкторы и многие другие функции для улучшения процесса разработки.

Перечень особо полезный функций Visual Studio :

1. *Волнистые линии и быстрые действия*

Волнистые линии которые можно заметить на рисунке 3.3.1 обозначают ошибки или потенциальные проблемы кода прямо во время ввода. Эти визуальные подсказки помогают немедленно устранить проблемы, не дожидаясь появления ошибок во время сборки или выполнения. Если навести указатель мыши на волнистую линию, на экран будут выведены

дополнительные сведения об ошибке. Также в поле слева может отображаться лампочка, указывающая на наличие сведений о быстрых действиях для устранения ошибки.

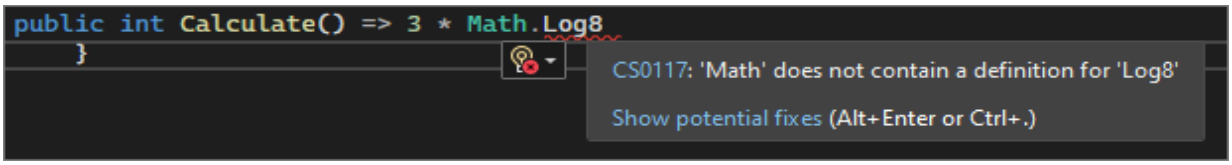


Рисунок 3.3.1. Визуализация указания на строчки с ошибкой и варианты исправления.

2. Очистка кода

Вы можете одним нажатием кнопки отформатировать код и применить к нему исправления, предложенные параметрами стиля кода, соглашениями в файле .editorconfig и (или) анализаторами Roslyn. Очистка кода, которая сейчас доступна только для кода C#, помогает устранять проблемы в коде перед переходом к его проверке. Пример данной функции отображен на рисунке 3.3.2.

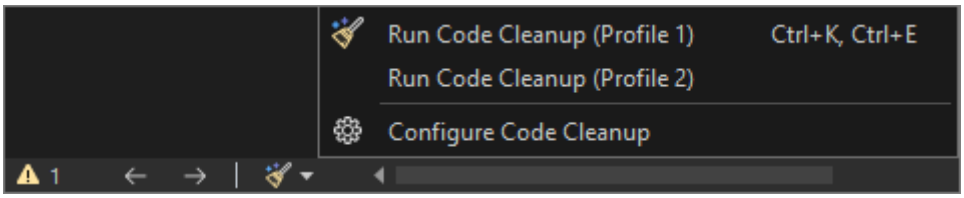


Рисунок 3.3.2. Функция отчистки кода

3.Рефакторинг

Данная функция отображена на рисунке 3.3.3. Рефакторинг включает в себя такие операции, как интеллектуальное переименование переменных, извлечение одной или нескольких строк кода в новый метод и изменение порядка параметров методов.

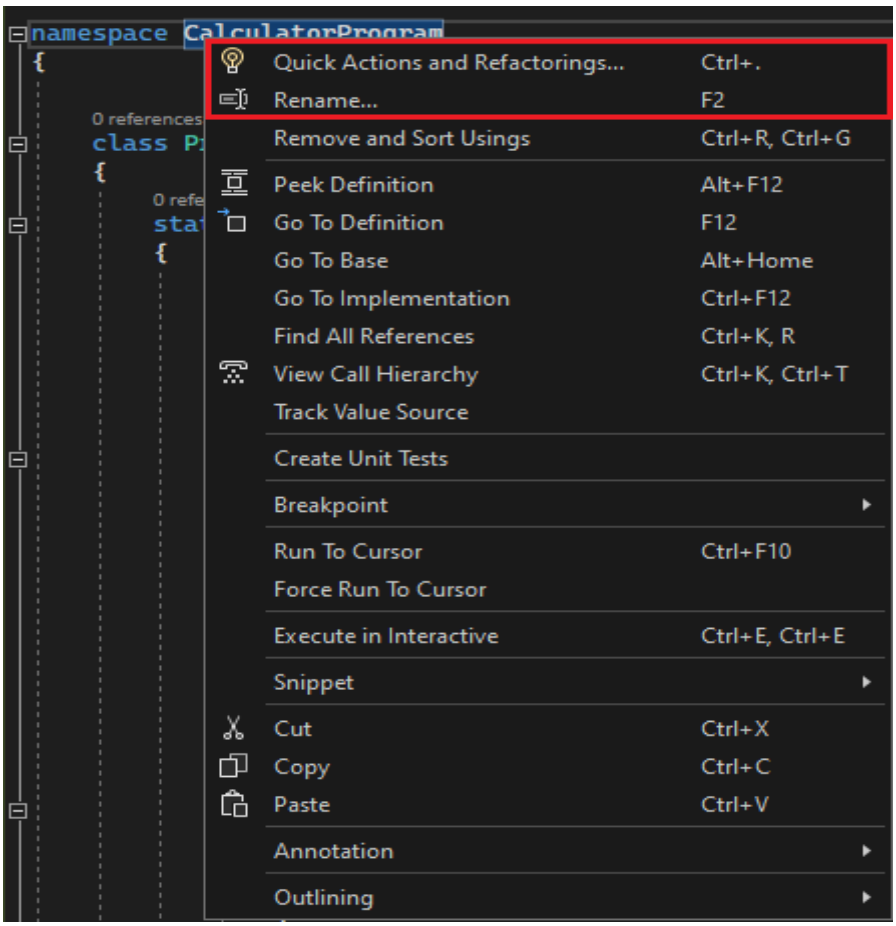


Рисунок 3.3.3. Функция Рефраторинга.

4.IntelliSense

IntelliSense — это набор возможностей, отображающих сведения о коде непосредственно в редакторе и в некоторых случаях автоматически создающих небольшие отрывки кода. По сути, это встроенная в редактор базовая документация, которая избавляет от необходимости искать информацию в других источниках.

На рисунке 3.3.4 показано, как IntelliSense отображает список членов типа:

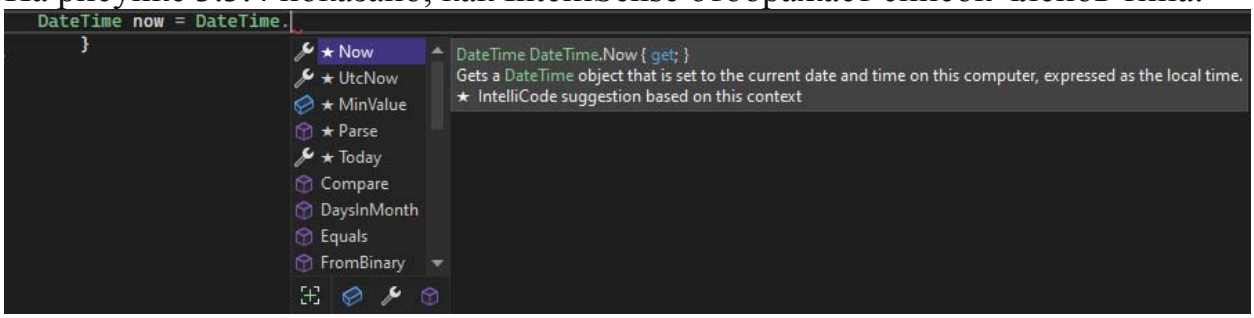


Рисунок. 3.3.4 Отображение функции “IntelliSence”

Функции IntelliSense зависят от языка. Дополнительные сведения см. в руководствах по IntelliSense для C# , IntelliSense для Visual C++, IntelliSense для JavaScript и IntelliSense для Visual Basic.

5.Поиск в Visual Studio

Иногда будет казаться, что в Visual Studio слишком много меню, действий и свойств. Чтобы быстро находить функции интегрированной среды разработки или элементы кода, в Visual Studio представлен единый компонент поиска (CTRL+Q) как продемонстрированно на рисунке 3.3.5

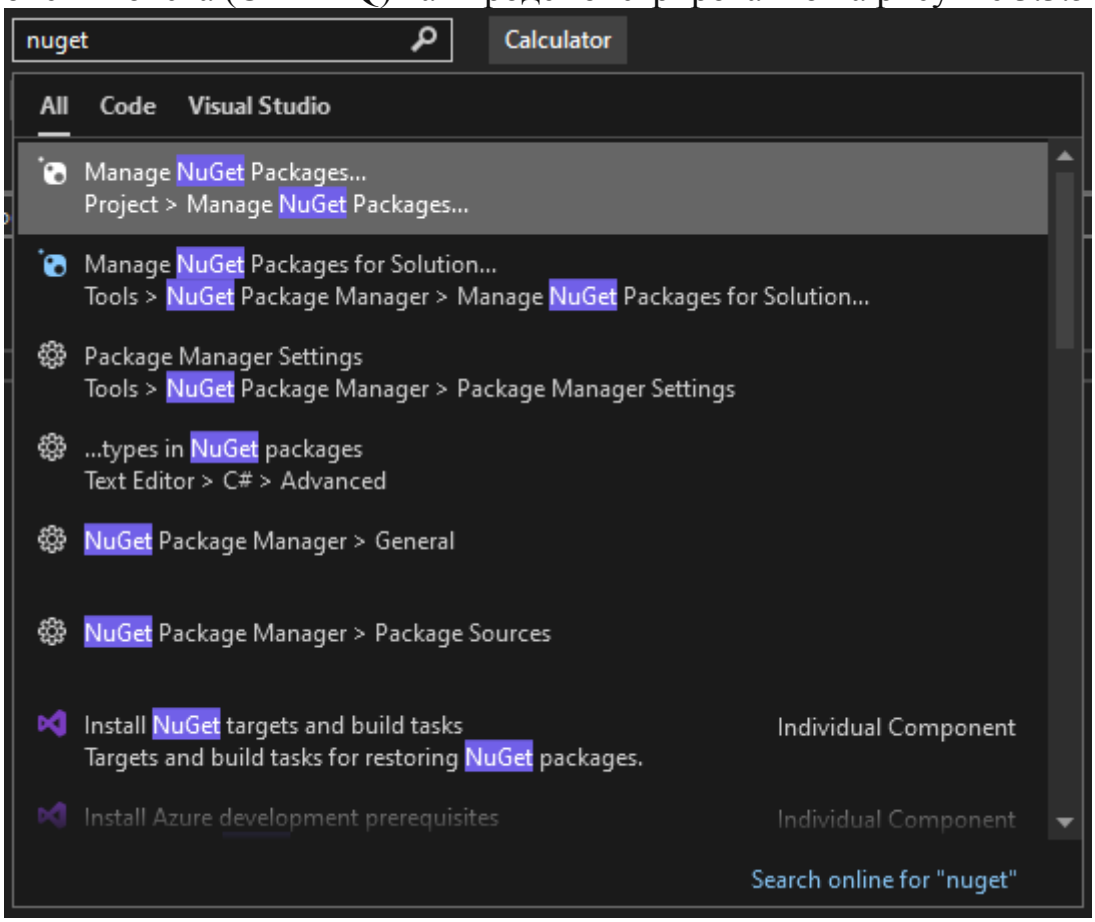


Рисунок 3.3.5. Функция поиска.

6.Live Share

Предоставляет возможности совместного редактирования и отладки в реальном времени независимо от типа приложения или языка. Вы можете мгновенно предоставлять общий доступ к своему проекту с поддержкой

высокого уровня безопасности. Кроме того, вы можете предоставлять общий доступ к сеансам, экземплярам терминала, веб-приложениям на локальном компьютере, голосовым звонкам и т. п.

7. Иерархия вызовов

В окне Иерархия показанном на рисунке 3.3.6 вызовов показаны методы, вызывающие выбранный метод. Это может быть полезно, если мы собираетесь изменить либо удалить метод или хотите отследить ошибку.

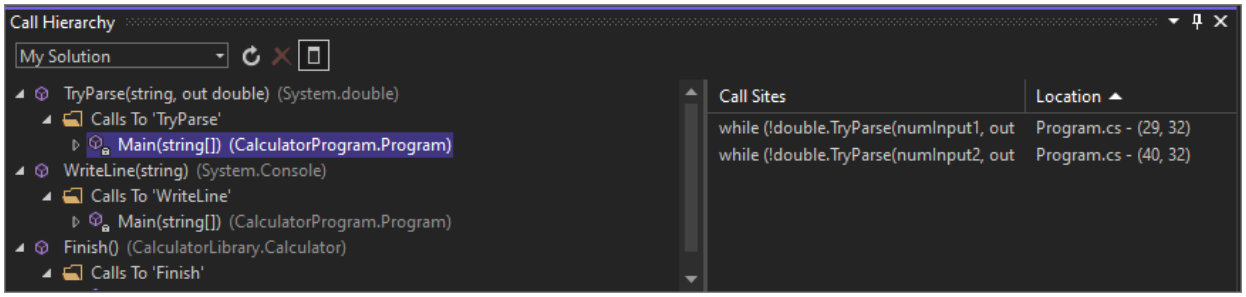


Рис 3.3.6. Окно иерархии.

7.CodeLens

CodeLens показанная на рисунке 3.3.7 помогает находить ссылки на код, изменения кода, связанные с кодом ошибки, рабочие элементы, проверки кода и модульные тесты — не выходя из редактора.

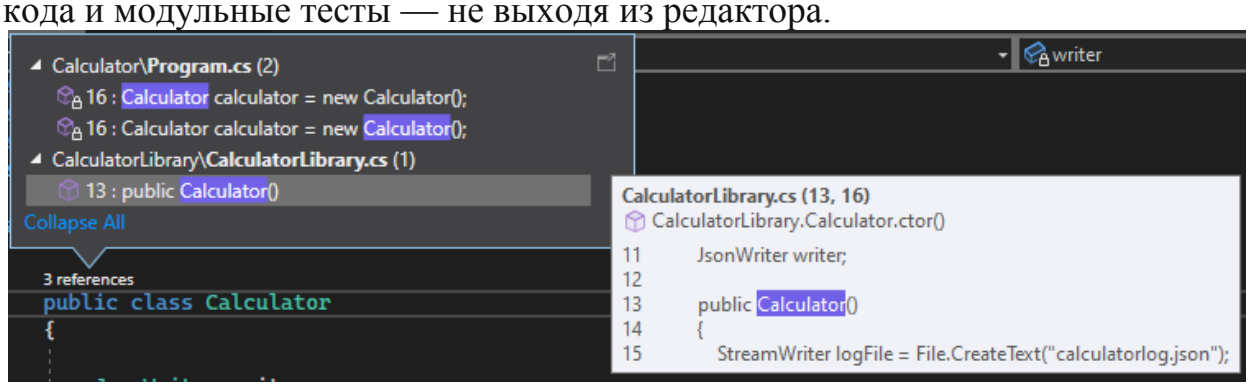


Рисунок 3.3.7. Функция “CodeLens”

8. Перейти к определению

Функция Перейти к определению показанная на рисунке 3.3.8, позволяет перейти к расположению, где определена выбранная функция или тип.

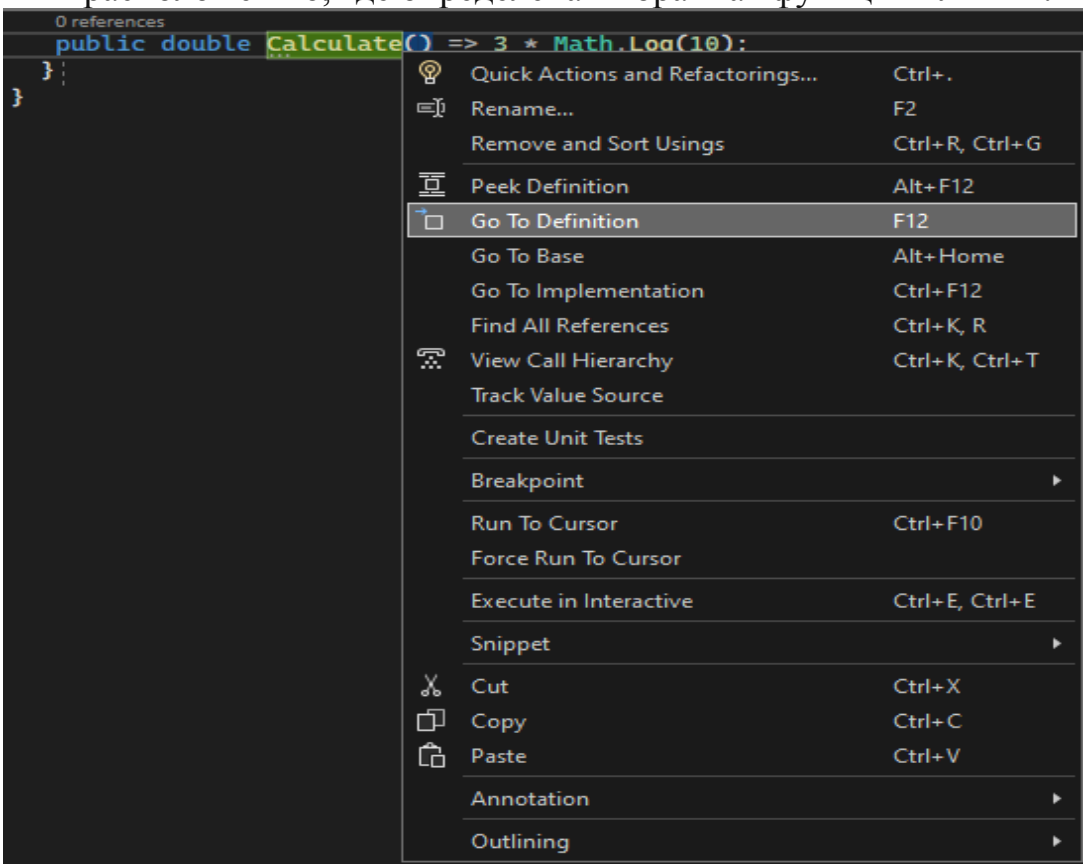


Рисунок 3.3.8. Изображение вкладки функции “перейти к определению”.

9. Показать определение

В окне Показать определение можно отобразить метод или определение типа, не открывая отдельный файл. Внешний вид функции указан на рисунке 3.3.9

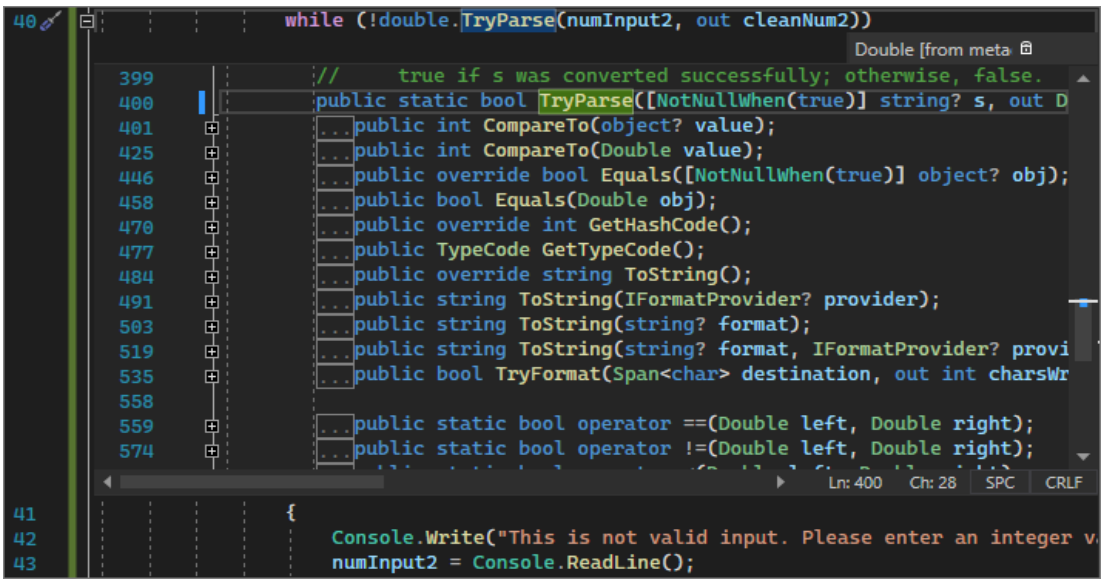


Рисунок 3.3.9. Функция для отображения определения

При создании графического интерфейса использовался язык C#.

В начале перед создание окон с показаниями , необходимо было установить условия приема сигнала, условий начала и остановки передачи данных. Устанавливались плашки выбора, от куда приходит сигнал : с Веб сокета или серийного порта, так же кнопки для остановки передачи и приема и кнопка для выхода из программы, начало программы показано на рисунке 3.3.10

```
#include "mainwindow.h"

MainWindow::MainWindow(QWidget *parent) : QMainWindow(parent) {
    this->setCentralWidget(new QWidget);
    this->setStatusBar(new QStatusBar(this));
    QGridLayout *grid = new QGridLayout(this->centralWidget());

    QToolBar *toolbar = this->addToolBar("Main Toolbar");
    QAction *startSerial = toolbar->addAction("START FROM SERIAL");
    QObject::connect(startSerial, SIGNAL(triggered()), this, SLOT(startSerial()));
    QAction *startSocket = toolbar->addAction("START FROM WEB SOCKET");
    QObject::connect(startSocket, SIGNAL(triggered()), this, SLOT(startSocket()));
    QAction *stop = toolbar->addAction("STOP");
    QObject::connect(stop, SIGNAL(triggered()), this, SLOT(stop()));
    QAction *quit = toolbar->addAction("QUIT");
    QObject::connect(quit, SIGNAL(triggered()), this, SLOT(close()));

    _gps = new GPS;
    _serial = NULL;
    _socket = NULL;
    _receivedPackages = 0;
```

Рисунок 3.3.10

Затем необходимо выводить данных на экран как га рисунке 3.3.11. Для удобства работы пользователя , было приятно такое решение как вывод данных о дате и времени приема пакета данных.

```
grid->addWidget(new QLabel("PACKAGE DATE:"), 1, 1);
_dataPacote = new QLCDNumber(this);
_dataPacote->setDigitCount(10);
grid->addWidget(_dataPacote, 1, 2);
grid->addWidget(new QLabel("PACKAGE TIME:"), 1, 3);
_horaPacote = new QLCDNumber(this);
_horaPacote->setDigitCount(8);
grid->addWidget(_horaPacote, 1, 4);
```

Рисунок 3.3.11

Так же для упрощения чтения показаний оператором необходимо выводить данные о долготе и широте. Команды представлены на рисунке 3.3.12

```
grid->addWidget(new QLabel("LATITUDE:"), 2, 1);
_latitude = new QLCDNumber(this);
_latitude->setDigitCount(8);
grid->addWidget(_latitude, 2, 2);
grid->addWidget(new QLabel("LONGITUDE:"), 2, 3);
_longitude = new QLCDNumber(this);
_longitude->setDigitCount(8);
grid->addWidget(_longitude, 2, 4);
```

Рисунок 3.3.12. Вывод данных о долготе и широте.

Необходимо так же учитывать возможный скорость перемещения буя с приёма передатчиком, для этого так же будут отображаться его скорость и его относительная высота по отношению к палубе судна где находится оператор рис 3.3.14

```
grid->addWidget(new QLabel("SPEED IN KM/H:"), 3, 1)
_velocidade = new QLCDNumber(this);
_velocidade->setDigitCount(6);
grid->addWidget(_velocidade, 3, 2);
grid->addWidget(new QLabel("ALTITUDE:"), 3, 3);
_altitude = new QLCDNumber(this);
grid->addWidget(_altitude, 3, 4);
```

Рисунок 3.3.14. Команды для вывода скорости и относительной высоты буя с приёма передатчиком.

Для удобства пользования, при запуске программы оператор имеет возможность увидеть количество доступных ему спутников, а так же возможность переключаться. В программе так же будет отображаться горизонтальная погрешность , это необходимо поскольку приема передатчик не стационарен.

Затем следует настройка вывода даты и времени , а так же задается настройка их отображения на экране. Программный код для выполнения данной задачи представлен на рисунке 3.3.13

```
void MainWindow::showGPS(QByteArray gpsPackage) {
    _gps->parseGPS(gpsPackage);
    if(_gps->_gpgga != NULL) {
        _latitude->display(_gps->_gpgga->_latitude);
        _longitude->display(_gps->_gpgga->_longitude);
        _numeroSatelites->display(_gps->_gpgga->_satelites);
        _altitude->display(_gps->_gpgga->_altitude);
        int hours = (int) _gps->_gpgga->_utcTime / 10000;
        int minutes = (int) _gps->_gpgga->_utcTime / 100 % hours;
        int seconds = (int) _gps->_gpgga->_utcTime % (hours * 100) % minutes;
        _horaPacote->display(QTime(hours, minutes, seconds).toString(QString("hh:mm:ss")));
        _precisaoPacote->display(_gps->_gpgga->_hdop);
    }
    if(_gps->_gprmc != NULL) {
        int day = _gps->_gprmc->_utDate / 10000;
        int month = _gps->_gprmc->_utDate / 100 % (day * 100);
        int year = _gps->_gprmc->_utDate % (day * 100000) % (month * 100) + 2000;
        _dataPacote->display(QDate(year, month, day).toString(QString("dd/MM/yyyy")));
        _velocidade->display(_gps->_gprmc->_speed);
    }
```

Рисунок 3.3.13. Команды для настройки вывода даты и времени.

После этого настраиваем условия для отображения количества доступных спутников для пользователя как показано на рисунке 3.3.15.

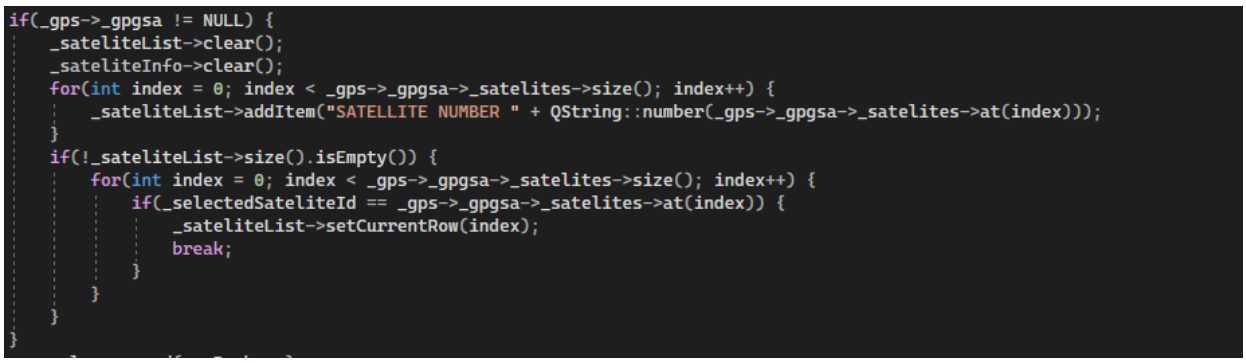


Рисунок 3.3.15. Команды, ответственные за вывод количества спутников.

Следующим шагом будет выставление ограничений для отображения координат как на рисунке 3.3.16. Надстройка максимума и минимума для вывода на экран, так же установка откуда должны браться данные.

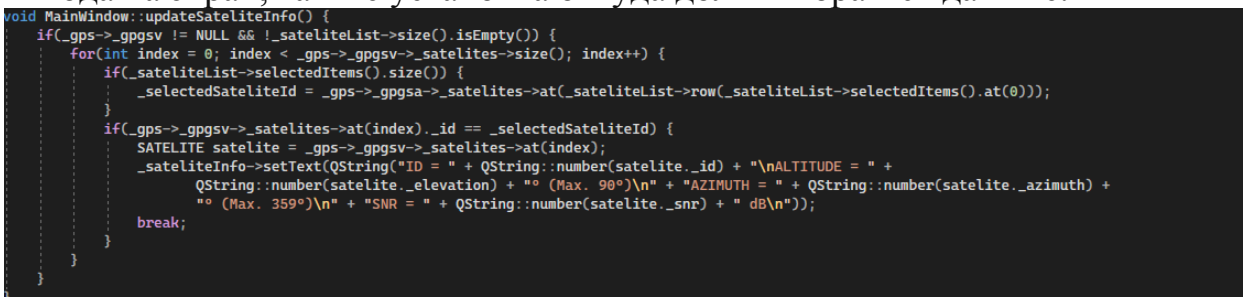


Рисунок 3.3.16. Программный код для выставления ограничений для отображения координат.

После проделанной работы, мы настраиваем нашу программа на считывание данных с серийного порта или же веб сокета, варианты для установки скорости чтения данных, устанавливаем программный порядок действий при ошибке загрузки софта или при других возможных ошибках связанных с серийным поротом или же веб сокетом. Далее проверяем нашу программу на наличие ошибок, если ошибок нет то мы можем запустить программу. Прототип визуального интерфейса мы можем наблюдать на рисунке 3.3.17.

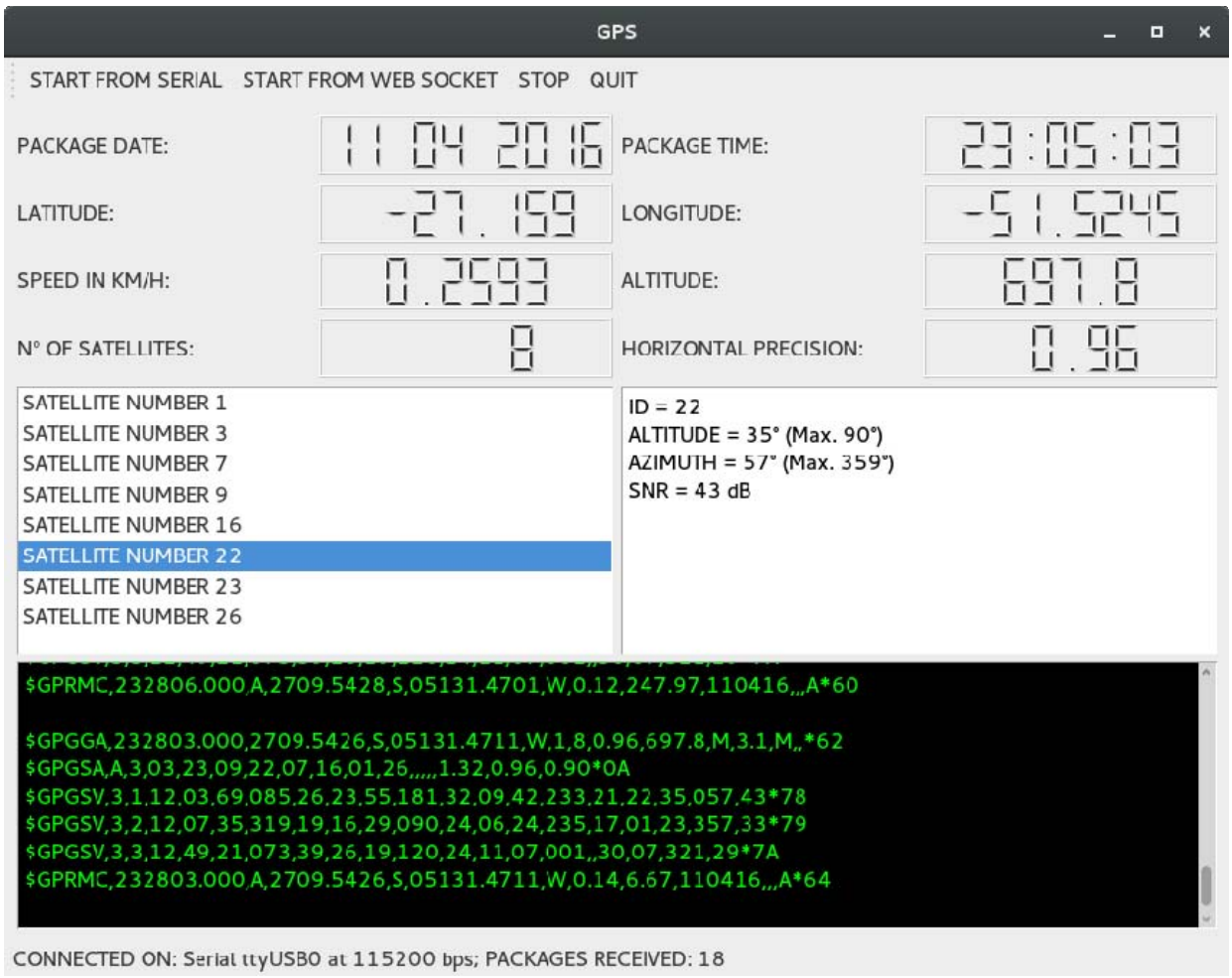


Рисунок 3.3.17. Макет графического интерфейса.

Вывод по 3 главе.

В третей главе были расмотрены способы настройки приема и передачи местоположения малогабаритного подводного аппарата по протоколу NMEA. Основные функции протокола, из каких он частей состоит, его описание, возможности и в каких сферах применяется. Были рассмотрены возможные варианты кодировки сигнала в виде необработанных сообщений, их расшифровку. Была рассмотрена программно-графическая часть протокола и из чего она состоит, за что отвечает тот или иной принимаемый или отправляемый символ. Так же был расмотрен способ создныия графического интерфейса на языке C#\C++ для использования операторм при управлении аппаратом. Целью интерфейса является упрощение понимания приходящих сигналов с NMEA для человека для более быстрой и эффективной работы. Были так же рассмотрены способы программирования приема передатчика для получения координат, даты и времени, количество доступных спутников и возможность переключаться между ними, время и дата получения последнего сигнала, а так же местоположения.

ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы поставленная цель, макета программного комплекса системы управления малогабаритным подводным аппаратом для дистанционного контроля корпуса судна в ледовых условиях, была достигнута.

Для этого были решены следующие задачи :

1) Проведён анализ существующей элементной базы и сделан выбор комплектующих .

2) создано аппаратно-программное обеспечение для:

- управления двигателями при осуществлении всплытия;

- управления двигателями при осуществлении погружения;

-управление двигателями при осуществлении горизонтального перемещения.

3) разработано ПО для:

- для приема и передачи данных с датчика GPS ;

- для создания макета графического интерфейса для управления малогабаритным подводным аппаратом.

При разработке программного обеспечения для подводного аппаратного возникла проблема связанная с подключением модуля радио управления. Решение было найдено за счёт добавления специальной библиотеки с расширениями для использования радио модуля. Так же при создании графического интерфейса возникла проблема с доступностью сред для разработки. В качесвте решения была выбрана единственная доступная среда на данный момент “Visual studio”

Данный подводный аппарат можно использовать в узких и трудно доступных места, куда не может добраться человек или есть риск для жизни человека, так же если есть риск ухудшения погодных условий, которые могут помешать человеку находится как в воде так и под водой.

Из-за недостатка времени и ресурсов, как технических, так и человеческих, при создании прототипа подводного аппарата не был добавлен модуль для видео камеры, который должен осуществлять непрерывную видео запись в хорошем качестве и сразу же передавать видео изображение непосредственно человеку , что управляет данным подводным аппаратом. Дальнейшим шагом к усовершенствованию прототипа является добавление данного модуля. Кроме этого, при дальней доработке аппарата имеется возможность на уровне программного обеспечения добавить возможность возврата подводного аппарата в зону приема сигнала от радио передатчика установленного на пульте, при потере этого сигнала это будет возможно при реализации автопилота. Так же, не будет лишним установить более мощное оборудование для осуществления приема-передачи управляющего сигнала для подводного аппарата , а так же для отправки исходящего сигнала от подводного аппарата на пульт управления.

СПИСОК ИСТОЧНИКОВ

1. Техническая спецификация ArduinoMega2560. Режим доступа: https://www.microchip.com/wwwproducts/en/ArduinoMega2560 . Дата обращения : 13.04.2022
2. Техническая спецификация Adafruit Motor Shield V2 . Режим доступа: https://learn.adafruit.com/adafruit-motor-shield-v2-for-arduino. Дата обращения : 13.04.2022
3. Техническая спецификация электродвигателя F130-16155. Режим доступа: https://elkomp.ru/cat/elektrodvigateli/elektrodvigatel-f130-16155. . Дата обращения : 13.04.2022
4. Техническая спецификация сервопривода MG995S. Режим доступа: https://3d-diy.ru/wiki/arduino-mechanics/servo-mg995S/. Дата обращения : 13.04.2022
5. Техническая спецификация радио модуля NRF24L01. Режим доступа : https://3d-diy.ru/wiki/arduino-moduli/radio-modul-nrf24l01/ . /. Дата обращения : 13.04.2022
6. Спасский М.А., Соболев М.В., Миклуш В.А. Создание подводного аппарата для визуального контроля повреждений морских объектов // Информационные технологии и системы: управление, экономика, транспорт, право. – 2019. – № 3 (35). – С. 149-152.
7. Техническая спецификация Atmega2650 Режим доступа: https://www.microchip.com/wwwproducts/en/ATmega2560.Дата обращения 20.03.2022.
8. Техническая спецификация микросхема L293Dю Режим доступа: https://static.chipdip.ru/lib/222/DOC000222839.pdf Дата обращения 20.03.2022.
9. Техническая спецификация микроконтроллера ATmega328p. Режим доступа: https://static.chipdip.ru/lib/303/DOC000303014.pdf. Дата обращения 20.03.2022.

10. Онлайн библиотека по Arduino. Режим доступа: http://electrik.info/microcontroller/1503-sposoby-chteniya-i-upravleniya-portami-vvoda-vyvoda-arduino.html. Дата обращения 20.04.2022
11. Онлайн справочник по Arduino. Режим доступа: https://wiki.iarduino.ru/page/GPS-module/. Дата обращения 21.04.2022
12. Расшифровка сигналов NMEA. Режим доступа: https://cirspb.ru/blog/other-articles/nmea-part1/ . Дата обращения 25.
13. Онлайн ресурс национальная ассоциация морской электроники по NMEA 0183 “Стандарт для сопряжения морских электронных устройств” - 1 января 2002 г. Дата обращения 10.05.2022
14. Белов А.В. Микроконтроллеры AVR: от азов программирования до создания практических устройств/ А.В. Белов. – СПб.: Наука и техника, 2016. – 544 с.
15. Алехин В.А. Микроконтроллеры PIC: основы программирования и моделирования в интерактивных средах MPLAB IDE, mikroC, TINA, Proteus. Практикум / В.А. Алехин. – М.: ГЛТ, 2016. – 248 с.
16. Аандрэ Ф. Микроконтроллеры семейства SX фирмы Ubicom / Ф. Аандрэ. – М.: ДМК, 2016. – 272 с.
17. Статья Алексея Бартоша. Режим доступа: http://electrik.info/main/school/1537-chto-takoe-tverdotelnoe-rele-i-kak-ego-pravilno-ispolzovat.html. Дата обращения 15.04.2022
18. Статья с ресурса RedComrade. Режим доступа: https://redcomrade.ru/security/kak-upravlyat-servoprivodom-dvumya-knopkami-upravlenie-servoprivodom-sg90/. Дата обращения 10.03.2022

ПРИЛОЖЕНИЕ А

Микроконтроллер	STM32F405RG (32-битный ARM Cortex M4)
Тактовая частота	168 МГц
Флеш-память	1024 кБ
SRAM-память	192 кБ
Номинальное рабочее напряжение	3,3 В
Рекомендуемое входное напряжение	7–15 В или 3,6–12 В
Максимальный ток с шины 5V	1000 мА
Максимальный ток с шины 3.3V	300 мА (включая питание микроконтроллера)
Максимальный ток с пина или на пин	25 мА
Максимальный суммарный ток с пинов или на пины	240 мА
Портов ввода-вывода общего назначения	26
Портов с поддержкой ШИМ	22
Портов с АЦП	12 (12 бит)
Портов с ЦАП	2 (12 бит)
Доступные аппаратные интерфейсы	4×UART/Serial, 3×I²C/TWI, 2×SPI

ПРИЛОЖЕНИЕ Б

Микроконтроллер	STM32F411CEU6 (32-битный ARM Cortex M4)
Тактовая частота	100 МГц
Объём Flash-памяти	512 КБ
Объём SRAM-памяти	128 КБ
Портов ввода-вывода всего	20
Портов с АЦП	8
Разрядность АЦП	12 бит
Портов с ШИМ	19
Разрядность ШИМ	16 бит
Аппаратных интерфейсов SPI	3
Аппаратных интерфейсов I²C / TWI	2
Аппаратных интерфейсов UART / Serial	1
Номинальное рабочее напряжение	3,3 В
Портов толерантных к 5 В	17
Максимальный ток с шины 3.3V	2 А (включая питание микроконтроллера)
Максимальный ток с пина или на пин	25 мА
Допустимое входное напряжение от внешнего источника	7–14 В
Габариты	33×21 мм
Габариты	69×53 мм

ПРИЛОЖЕНИЕ В

Микроконтроллер	AT91SAM3X8E
Рабочее напряжение	3.3В
Напряжение питания (рекомендуемое)	7-12В
Напряжение питания (предельное)	6-16В
Цифровые выводы	54 (из них 12 могут работать как ШИМ-выходы)
Аналоговые входы	12
Аналоговые выходы	2 (ЦАП)
Суммарный выходной ток всех выводов (максимальный)	130 мА
Максимальный выходной ток вывода	3.3V 800 мА
Максимальный выходной ток вывода	5V 800 мА
Flash-память	– 512 КБ в полном объеме доступна пользовательским программам
SRAM	– 96 КБ (два банка памяти: 64 КБ и 32 КБ)
Тактовая частота	84 МГц