Министерство науки и высшего образования Российской Федерации
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Российский экономический университет имени Г.В. Плеханова»
Московский приборостроительный техникум

Выпускная квалификационная работа
(Дипломная работа)

На тему: Разработка программного комплекса с предиктивной коррекцией ошибок управления (на примере ООО «Центр инновационных разработок ВАО»)

Дрюпина Андрея Александровича
Студент 4 курса группы П50-1-18

по специальности 09.02.07 «Информационные системы и программирование»
для присвоения квалификации: программист
Форма обучения: очная

Руководитель: _____ / Копылов Олег Валерьевич /
(подпись)
«____»_____2022 г.

Консультант: _____ / Попова Любовь Юрьевна /
(подпись)
«____»_____2022 г.

Студент: _____ / Дрюпин Андрей Александрович /
(подпись)
«____»_____2022 г.

Допущена к защите
Распоряжение от «_____»_____2022 г. № _____

2022

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Робототехника в современном мире является достаточно важной его частью. Это сфера, совершенствованию которой уделяется особое внимание ввиду пользы, которую она приносит человечеству. Промышленность, медицина, военно-промышленный комплекс, сельское хозяйство – лишь немногие примеры сфер, где робототехнические механизмы нашли обширное применение. Однако робототехническая отрасль, несмотря на все достижения, не является совершенной, и имеет массу проблем самого разного плана. Одной из таких проблем - скорость смены полезной нагрузки. Для большинства платформ вспомогательное оборудование не является универсальным и время разработки и установки зависит от его сложности и функциональности. Одна из целей выпускной квалификационной работы – спроектировать, разработать и запрогрограммировать универсальный интерфейс связи вспомогательного оборудования с ведущей платформой. Вторая цель выпускной квалификационной работы – разработка предиктивной системы коррекции ошибок управления. Эта система является вспомогательной для оператора платформы и призвана максимально снизить человеческий фактор.

Универсальные платформы используются для проведения широкого спектра работ на территориях, опасных для человека. Используя быстрозаменяемое вспомогательное оборудование, оператор робота может осуществлять: обследование подозрительных вещей, извлечение людей из-под завалов, радиационный и химический контроль, забор проб воды и грунта.

Было принято решение создать универсальный роботизированный аппаратно-программный комплекс для выполнения работ в условиях опасных и вредных факторов с возможностью быстрой замены полезной нагрузки.

Актуальность данной работы заключается в необходимости создания отечественной универсальной роботизированной платформы для использования его в гражданских (в том числе коммерческих) целях, требующих быстрой и легкой замены вспомогательной нагрузки на дистанционно-управляемых роботах. Проект особенно актуален для служб экстренного реагирования.

1. ОБЩАЯ ЧАСТЬ

1.1. Цель разработки

Создать универсальный роботизированный аппаратно-программный комплекс для выполнения работ в условиях опасных и вредных факторов (опасность обрушения, радиационная, биологическая или химическая угроза, высокие температуры) с возможностью быстрой замены полезной нагрузки.

1.2. Средства разработки

В качестве средств разработки указаны программы, представленные в таблице 1.

Таблица 1 – Программные средства

№	Тип средства	Название средства	Назначение
1	2	3	4
1	Текстовый редактор	Microsoft Word 2019 MSO (16.0.13530.20368)	Разработка документации, формирование отчетных документов
2	Инструментальное средство разработки программных решений	Visual Studio 2019 Community 16.8.4	Разработка десктоп приложения
3	Инструментальное средство разработки программных решений	Arduino IDE 1.8.10	Разработка встраиваемого приложения
4	Инструментальное средство разработки программных решений	Atmel Studio 4.19	Разработка встраиваемого приложения
5	Средство проектирования	Draw.io 2.0.9	Разработка схем для проектирования приложения
6	Средство логического анализа сигналов	Saleae Logic 2.3.45	Логический анализ сигналов
7	Операционная система	Microsoft Windows 10 Pro	Операционная система

В качестве средств вычислительной техники использовался персональный компьютер. Его характеристики представлены в таблице 2.

Таблица 2 – Технические средства

№	Тип оборудование	Наименование оборудование
1	2	3
Ноутбук HP Envy x360 13-ar0010ur		
1	Размер экрана	13.3”
2	Разрешение экрана	1920x1080
3	Линейка процессора	AMD Ryzen 5 3500U
4	Количество ядер процессора	4
5	Оперативная память	8 ГБ
6	Тип видеокарты	встроенная
7	Видеокарта	AMD Radeon Vega 8
8	Конфигурация накопителей	SSD
9	Общий объем всех накопителей	128 ГБ

В качестве средств периферийной техники использовались устройства, приведённые в таблице 3.

Таблица 3 – Периферийные устройства

№	Наименование	Описание
1	2	2
1	Логический анализатор	Logic Pro 16
2	4-х канальный осциллограф	Hantek DSO4254C
3	Генератор сигналов	UNI-T UTG901C
4	AVR программатор	USBAsp
5	AVR JTAG Debugger	AVR ISP II

2. СПЕЦИАЛЬНАЯ ЧАСТЬ

2.1. Постановка задачи

Разработать встроенное приложение для управления универсальной роботизированной платформой-носителем с предиктивной системой коррекции ошибок управления (на примере ООО «Центр инновационных разработок ВАО»).

В рамках выпускной квалификационной работы ставится задача разработать и собрать образец автоматизированного универсального роботизированного комплекса, в который входит:

1) Универсальная платформа-носитель;
2) Универсальный интерфейс ввода/вывода для подключения полезной нагрузки.

2.2. Входные и выходные данные

Входные данные - поток данных в виде структурированного текста, представляющий собой строку из ASCII символов, конвертируемые программой в структуру данных;

Выходные данные – поток данных в виде структурированного текста, представляющий собой строку из ASCII символов, конвертируемые программой в структуру данных.

2.2.1. Подробные требования к проекту

В качестве требований проекта необходимо выполнить следующие задачи:

- Реализовать аппаратную часть платформы-носителя: основной вычислительный модуль, модуль радиосвязи, модуль получения пространственных и физических координат;
- Реализовать подключение к управляющему компьютеру через радиоканал для управления платформой-носителем на расстоянии, а также для получения телеметрии оператором;
- Реализовать SDK для сторонних разработчиков внешних модулей,

путем создания библиотеки с готовыми программными модулями управления, контроля и тестирования аппаратной части;

- Реализовать возможность передвижения платформы с задаваемой оператором программно-аппаратного комплекса скоростью и направлением;
- Реализовать возможность передачи телеметрии на управляющий компьютер;
- Реализовать определение текущих физических координат платформы;
- Реализовать определение текущих пространственных координат;
- Реализовать систему предиктивной коррекции ошибок управления для упрощения работы оператора с платформой, вывод подробной информации о всех узлах и модулях платформы-носителя и полезной нагрузки, динамического поддержания параметров движения, а также вывод предупредительных сообщений о некорректной работе программно-аппаратного комплекса;
- Реализовать программно-аппаратный интерфейс для подключения внешней полезной нагрузки к платформе-носителю по технологии Plug and Play («Подключи и работай»).

2.3. Внешняя спецификация

2.3.1. Описание задачи

Робототехнический комплекс состоит из универсальной роботизированной платформы – носителя полезной нагрузки с интегрированным интерфейсом для подключения внешних устройств (рабочего инструмента), манипулятора с уникальной системой контроля перемещения и программного обеспечения, осуществляющего управление робототехнической системой как единым целым. Предполагается организация техподдержки и сервиса на территории

России.

Основные технические параметры:

- Масса – 20 кг;
- Габаритные размеры – 400x250x110 мм
- Грузоподъёмность – 15 кг;
- Максимальная скорость – 5 км/час;
- Максимальная дальность управления на прямой видимости – 1 км;
- Тип разъема порта ввода/вывода – D-sub 15;
- Максимально выдаваемая мощность порта ввода/вывода – 80 Вт;
- Максимальная скорость порта ввода/вывода в командном режиме – 2 Мб/сек;
- Модель связи порта ввода/вывода – Мастер/ведомый - Мастер/ведомый;
- Вид механического крепления полезной нагрузки – универсальные рельсы;
- Время автономной работы на одном элементе питания – 40 минут;
- Максимально возможное количество встраиваемых элементов питания – 6 шт;
- Предполагаемая цена разрабатываемой платформы-носителя – 400 тыс. руб.

Разработчик встраиваемого приложения скачивает приложение для станции управления, SDK для платформы и подключает скачанные библиотеки к своему проекту. Далее он создает новый объект класса платформы, содержащий все необходимые функции для работы с аппаратной частью. В самом начале необходимо провести

инициализацию объекта, введя в аргументы метода функциональное название платформы и уникальный ключ, и провести инициализацию метода-обработчика команд с персонального компьютера. После первоначальной инициализации разработчик приложения пишет свою управляющую программу, используя методы класса платформы.

Работа с универсальным интерфейсом осуществляется в двух режимах: ручном и командном. Ручной режим позволяет управляющей программе конфигурировать любой контакт интерфейса. В командном режиме передача осуществляется при помощи командных пакетов и пакетов с данными, в полудуплексном режиме на скорости до 2 Мб/сек.

2.3.2. Входные и выходные данные

В Таблице 4 представлены входные данные, получаемые с компьютера.

Таблица 4 – Входные данные

№	Наименование	Тип	Ограничения	Формат ввода	Описание
1	2	3	4	5	6
1	move	[A-Z, a-z] {1,1}	Char	Пакет, передаваемый с компьютера на платформу	Направление движения
2	Speed	[0-9, ,] {1, 100}	Int		Скорость в процентах
3	Value	[A-Z, a-z] {1,1}	Char		Тип разгона
4	azimutloc	[0-9, ,] {1, 100}	Int		Азимут движения
5	Gpio1	[0-9 , ,] {1, 100}	Double		значение gpio1 masterLink (masterLink - универсальный интерфейс подключения устройств)
6	Gpio2	[0-9 , ,] {1, 100}	Double		значение gpio2 masterLink (masterLink - универсальный интерфейс подключения

№	Наименование	Тип	Ограничения	Формат ввода	Описание
1	2	3	4	5	6
					устройств)
7	Gpio3	[0-9 , ,] {1, 100}	Double		значение gpio3 masterLink (masterLink - универсальный интерфейс подключения устройств)
8	Gpio4	[0-9 , ,] {1, 100}	Double		значение gpio4 masterLink (masterLink - универсальный интерфейс подключения устройств)
9	Systemstatus	[0-9 , ,] {1, 100}	Int		Статус системы
10	data	[A-Z, a-z] {1,256}	String		Дополнительные данные

В Таблице 5 представлены выходные данные, отправляемые на компьютер.

Таблица 5 – Выходные данные

№	Наименование	Тип	Ограничения	Формат ввода	Описание
1	2	3	4	5	6
1	Move	Char	[A-Z, a-z] {1,1}	Пакет, передаваемый с платформы на компьютер	Направление движения
2	speed	Int	[0-9, ,] {1, 100}		Скорость в процентах
3	value	Char	[A-Z, a-z] {1,1}		Тип разгона
4	Lcurr	Double	[0-9 , ,] {1, 100}		Значение тока левого мотора
5	Rcurr	Double	[0-9 , ,] {1, 100}		Значение тока правого мотора
6	Accx	Double	[0-9 , ,] {1, 100}		Значение акселерометра по оси x
7	Accy	Double	[0-9 , ,] {1, 100}		Значение акселерометра по оси y
8	Accz	Double	[0-9 , ,] {1, 100}		Значение акселерометра по

№	Наименование	Тип	Ограничения	Формат ввода	Описание
1	2	3	4	5	6
					оси z
9	Gyrox	Double	[0-9 , ,] {1, 100}		Значение гироскопа по оси x
10	Gyroy	Double	[0-9 , ,] {1, 100}		Значение гироскопа по оси y
11	Gyroz	Double	[0-9 , ,] {1, 100}		Значение гироскопа по оси z
12	Magx	Double	[0-9 , ,] {1, 100}		Значение магнетометра по оси x
13	Magy	Double	[0-9 , ,] {1, 100}		Значение магнетометра по оси y
14	Magz	Double	[0-9 , ,] {1, 100}		Значение магнетометра по оси z
15	Lan	String	[A-Z, a-z] {1,256}		Значение широты
16	Lon	String	[A-Z, a-z] {1,256}		Значение долготы
17	vbat	Double	[0-9 , ,] {1, 100}		Значение напряжения батареи в вольтах
18	Exitd	int	[0-9] {1, 7}		Идентификатор внешнего устройства
19	exitstatus	int	[0-9] {1, 1}		Статус внешнего устройства

2.3.3. Методы

При разработке приложения для управления роботизированной платформы-носителем использовались следующие методы:

- Объектно-ориентированное программирование: работа с классами, методами и структурами;
- Данные, передаваемые платформой на компьютер представлены на рисунке 1.
- Данные, передаваемые компьютером на платформу представлены на рисунке 2.

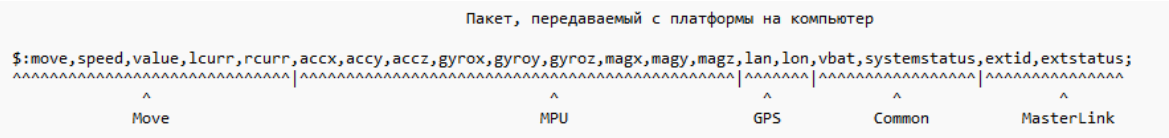

Рисунок 1 – Пакет Platform to PC

Аргументы пакета:

move - направление движения, char

speed - скорость в процентах, int

value - тип разгона, char

lcurr - значение тока левого мотора, double

rcurr - значение тока правого мотора, double

accx - значение акселерометра по оси x, double

accy - значение акселерометра по оси y, double

accz - значение акселерометра по оси z, double

gyrox - значение гироскопа по оси x, double

gyroy - значение гироскопа по оси y, double

gyroz - значение гироскопа по оси z, double

magx - значение магнетометра по оси x, double

magy - значение магнетометра по оси y, double

magz - значение магнетометра по оси z, double

lan - широта, string

lon - долгота, string

vbat - напряжение батареи в вольтах, double

extid - идентификатор внешнего устройства

extstatus - статус внешнего устройства, int

В получаемом пакете находятся значения с конкретных датчиков изменения которых, непосредственно отображаются в приложении. Например, если значения «move» - направление движения платформы будет меняться, это означает что платформа меняет своё направление движения.

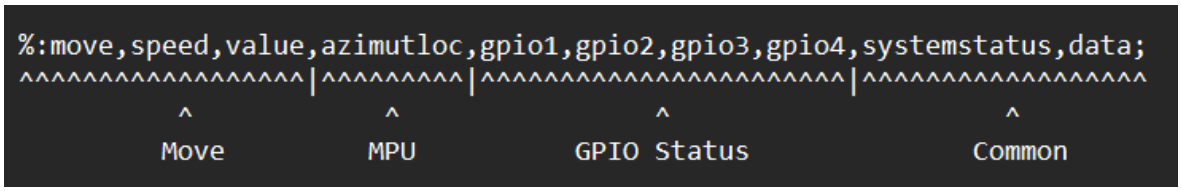

Рисунок 2 - Пакет PC to Platform

Аргументы пакета:

move - направление движения, char

speed - скорость в процентах, int

value - тип разгона, char

azimutloc - азимут движения, int

gpio1 - значение gpio1 masterLink, double

gpio2 - значение gpio2 masterLink, double

gpio3 - значение gpio3 masterLink, double

gpio4 - значение gpio4 masterLink, double

systemstatus - статус системы, int

data - Дополнительные данные, String

2.3.4. Тесты

После разработки приложения был отобран следующие виды тестирования:

Функциональное тестирование:

- Тестирование методом черного ящика;
- Метод «Входной двери» (Бинарное тестирование yes/no).

Тестирование производительности:

- Нагрузочное тестирование;
- Стресс тестирование.

Далее определившись с видами тестирования, был произведен ряд тестовых испытаний:

Тестирование методом черного ящика было применено при тестировании работоспособности функционала встроенного приложения для передачи команд на станцию управления.

Метод «Входной двери» применялся при тестировании работы приложения со станцией управления.

Нагрузочное тестирование - тестирование предназначено для проверки работоспособности системы при стандартных нагрузках и для определения максимально возможного пика, при котором система работает правильно.

Стресс тестирование - Тестирование предназначено для проверки работоспособности системы при нестандартных нагрузках и для определения максимально возможного пика, при котором система работает правильно. Так же предназначено для выявления результатов, при которых система переходит в нерабочее состояние.

Тестовые сценарий и результаты тестирования указаны в приложении Б «Результаты и сценарий тестовых испытаний».

2.4. Проектирование

2.4.1. Схема архитектуры программы

Схема архитектуры программы представлена на рисунке 3.

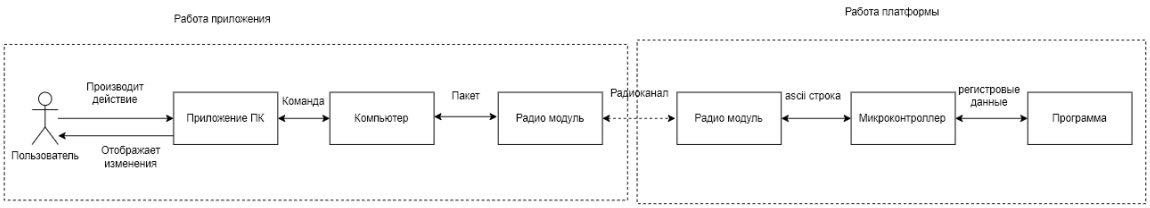

Рисунок 3 – Схема архитектуры программы

«Производит действие» - работа с приложением при помощи компьютерной мыши или клавиатуры;

«Команда» - работа приложения с ПК на аппаратном системном уровне;

«Пакет» - работа ПК с радио модулем на аппаратном уровне периферии;

«Радиоканал» - обмен байтами по радиоканалу на частоте 443.920 МГц.

«ascii строка» - передача данных на уровне представления;

«Регистровые данные» - работа программы с микроконтроллером на уровне представления регистров;

«Отображает изменения» - пользователь видит изменения данных.

2.4.2. Структурная схема программы

Структурная схема программы представлена на рисунке 4.

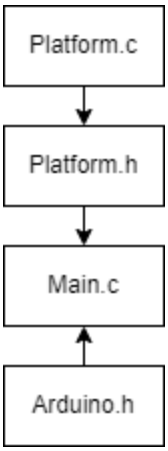

Рисунок 4– Структурная схема приложения

Описание модулей представлено в таблице 6.

Таблица 6 – Описание модулей

№	Название модуля	Описание модуля
1	Main.c	Точка входа
2	Platform.h	Заголовочный файл библиотеки управления
3	Platform.c	Файл логики библиотеки управления
4	Arduino.h	Заголовочный файл вспомогательных функций платформы Ардуино

2.4.3. Функциональная схема

Функциональная схема программы представлена на рисунке 5.

Рисунок 5 - Функциональная схема приложения

2.4.4. Аппаратная схема

Аппаратная структура платформы представлена на рисунке 6.

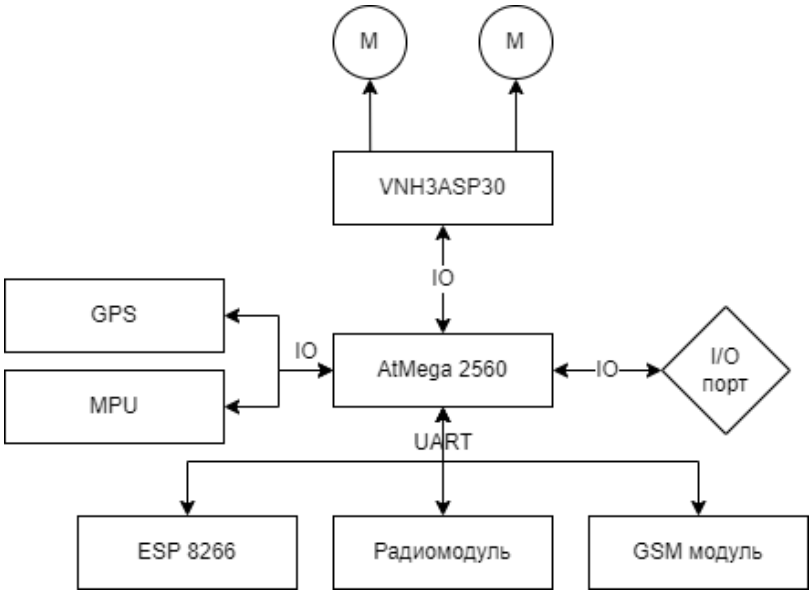

Рисунок 6 - Аппаратная схема

Описание компонентов:

Atmega2560 – AVR Микроконтроллер Atmel серии Mega;

M – Тяговый мотор;

VNH3ASP30 – Сдвоенный драйвер моторов;

GPS – GNSS модуль точных геокоординат;

MPU – Модуль отслеживания пространственных перемещений;

ESP 8266 – Модуль Wi-Fi;

Радиомодуль – Приемопередающий модуль для связи с ПК посредством защищенного радиоканала;

GSM модуль - Приемопередающий модуль для связи с ПК посредством передачи пакетов через станции подвижной связи (сотовая связь);

I/O Порт – Порт интерфейса MasterLink для связи с полезной нагрузкой.

2.5. Результаты работы программы

Так как у встраиваемого приложения нет интерфейса, на рисунке 7 приведены основные управляющие сигналы, полученные при помощи логического анализатора.

Рисунок 7- Основные сигналы платформы

Описание сигналов:

- Канал 0 – Исходящие данные от платформы
- Канал 1 – Входящий сигнал
- Канал 2 – Тактовый сигнал драйвера мотора
- Канал 3 – Отладочный сигнал векторов прерываний UART
- Время обработки полученного сигнала микроконтроллером: 2,1186 ms;
- Время работы прерывания UART: 9,999687 ms;
- Длительность отладочных сигналов между пакетами: 16 ns;

3. ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ

3.1. Инструментальные средства разработки

Для разработки приложения использовалась среда разработки Arduino IDE, которая позволяет писать код на языке C++ и Wiring с встроенным компилятором под архитектуру AVR. Преимуществами данной среды разработки является удобный графический интерфейс, быстрая компиляция программного кода и возможность быстрого редактирования программного кода. Архитектура AVR была выбрана из-за простоты разработки под нее, легкой поддержки, а также из-за встроенной богатой периферии.

Для работы с сигналами использовался программно-аппаратный комплекс Saleae Logic PRO. Это один из немногих комплексов для работы с сигналами. Его преимуществами является информативный графический интерфейс, большая база предустановленных библиотек для анализа цифровых сигналов, большое количество вводных каналов и скорость работы близкая к 1 ГГц.

Для работы с документацией использовался текстовый процессор, предназначенный для создания, просмотра и редактирования текстовых документов, с локальным применением простейших форм таблично-матричных алгоритмов. Выпускается корпорацией Microsoft в составе пакета Microsoft Office. Используемой версией является Microsoft Office Word 2016 для Windows. Его преимуществами являются большая функциональность, поддержка мультиоперационности, а также удобный интерфейс.

Для создания схем использовался векторный графический редактор «Draw.io», редактор диаграмм и блок-схем для различных операционных систем. Его преимуществом является Web-расположение, без необходимости установки.

В качестве операционной системы использовалась операционная

система для персональных компьютеров и рабочих станций, разработанная корпорацией Microsoft в рамках семейства Windows NT. Она является распространенной и проработанной.

3.2. Отладка программы

Основными ошибками были ошибки переполнения входного буфера UART, получение неточных координат GNSS, получение неправильных данных с датчиков.

Исправление данных ошибок проводилось путем сокращения количества отправляемых с рабочей станции оператора команд, ошибки GNSS путем программной перезагрузки модуля, а неточные данные с датчиков компенсировались при помощи различных фильтров значений.

3.3. Характеристика программы

Характеристика программы представлена в таблице 7.

Таблица 7 – Характеристика программы

№	Название модуля	Описание модуля	Размер модуля	Кол-во строк
1	PlatformMain.cpp	Модуль основной программы	1,2 кб	36
2	Platform.h	Заголовочный файл библиотеки Platform	4,6 кб	147
3	Platform.cpp	Модуль логики библиотеки Platform	23,9 кб	628
4	Arduino.h	Заголовочный файл библиотеки Arduino	7,2 кб	260
5	Display.cpp	Модуль программы полезной нагрузки «Дисплей»	3,2 кб	116

ЗАКЛЮЧЕНИЕ

В изложенной выпускной квалификационной работе стояла задача упростить работу в условиях опасных для человека, снизить вероятность получения травм, снизить вероятность получения вредных факторов, способных навредить здоровью человека на предприятиях. Опасность обрушения, радиационная, биологическая или химическая угроза, высокие температуры и т.д. все этими проблемы нам предстояло решить благодаря созданию универсальной платформы-носителя с возможностью быстрой замены полезной нагрузки.

На этапе анализа необходимо было продумать, как создать устройство, при помощи которого можно обезопасить нахождение на опасных для человека предприятиях. Далее был проанализирован рынок на наличие аналогов. Прямых аналогов предстоящей разработки не нашлось, а косвенные аналоги имели существенные недостатки, например такие как большая масса и габариты, отсутствие возможности замены полезной нагрузки «на ходу», отсутствие автономных режимов управления и гораздо большую стоимость.

После выявления необходимых требований и анализа поставленных задач предстояло спроектировать аппаратную часть и интерфейс подключения внешних устройств к платформе. Проектирование затрудняла задача автоматической перенастройки платформы внешним устройством под себя и необходимость использовать промышленные стандарты соединений, что успешно решилось использованием разъема D-SUB15 и разработкой встроенного программного модуля-программатора.

Разработка встроенной управляющей программы производилась на языке C++ с вставками ассемблерного кода для увеличения производительности и уменьшения конечного размера программы. Также были использованы аппаратные реализации протоколов UART, SPI, I2C

и блоков арифметики для ускорения работы программы и реализации вычислительного конвейера. Далее были написаны основные функции для взаимодействия с платформой по радиоканалу.

После разработки аппаратная и программная части были протестированы различными способами, включая тесты сигналов, анализ временных задержек, помехоустойчивость радиоканала, устойчивость к механическим воздействиям и устойчивость к ошибкам управления.

Выполненная задача отвечает поставленным методам, основными преимуществами программно-аппаратного комплекса являются: быстрая реакция на команды управления, интуитивное управление, позволяющее работать в визуальном режиме, подробная телеметрия позволяющая работать в полувизуальном режиме, а также работа с любыми типами полезной нагрузки. Из недостатков: отсутствие возможности работы в инструментальном режиме, отсутствие возможности работы по управляющей программе без перенастройки платформы, а также требуются минимальные навыки работы с промышленными роботами.

СПИСОК ИСПОЛЬЗУЕМЫХ МАТЕРИАЛОВ

1) ГОСТ Р 7.0.5-2008 БИБЛИОГРАФИЧЕСКАЯ ССЫЛКА. Общие требования и правила составления.

2) Arduino Library List [Электронный ресурс] URL: https://www.arduinolibraries.info/ (дата обращения 06.02.2022).

3) Avr-libc: Example using the two-wire interface (TWI) [Электронный ресурс] URL: https://www.nongnu.org/avr-libc/user-manual/group__twi__demo.html (дата обращения 04.02.2022).

4) TWI on ATMEGA2560 | AVR Freaks [Электронный ресурс] URL: https://www.avrfreaks.net/forum/twi-atmega2560 (дата обращения 06.02.2022).

5) Электронная свободная энциклопедия [Электронный ресурс] URL: https://ru.wikipedia.org/wiki/I%C2%B2C (дата обращения 07.02.2022).

6) atmega microcontroller registers setting for i2C | AVR Freaks [Электронный ресурс] URL: https://www.avrfreaks.net/forum/atmega-microcontroller-registers-setting-i2cы (дата обращения 15.02.2022).

7) Atmega2560/i2c.h at master · JoshAshby/Atmega2560 [Электронный ресурс] URL: https://github.com/JoshAshby/Atmega2560 (дата обращения 02.03.2022).

8) ATmega2560: работа с UART [Электронный ресурс] URL: http://microsin.net/programming/avr/atmega2560-working-with-uart.html (дата обращения 02.03.2022).

9) Serial Communication using UART on ATMEGA2560 [Электронный ресурс] URL: https://www.avrfreaks.net/forum/serial-communication-using-uart-atmega2560 (дата обращения 03.03.2022).

10) Atmel ATmega640/V-1280/V-1281/V-2560/V-2561/V [Электронный ресурс] URL: https://ww1.microchip.com/downloads/en/devicedoc/atmel-2549-8-bit-avr-

microcontroller-atmega640-1280-1281-2560-2561_datasheet.pdf (дата обращения 03.03.2022).

11) ATmega328PA [Электронный ресурс] URL: https://files.amperka.ru/datasheets/ATmega328.pdf (дата обращения 03.03.2022).

12) SPI on Arduino Mega not working in Assembler [Электронный ресурс] URL: https://www.avrfreaks.net/forum/spi-arduino-mega-not-working-assembler (дата обращения 04.03.2022).

13) avr-sbi-for-spi-ports-h [Электронный ресурс] URL: https://stackoverflow.com/questions/40807274/avr-sbi-for-spi-ports-h-atmega2560 (дата обращения 06.03.2022).

14) ATmega2560 as an SPI slave [Электронный ресурс] URL: https://rpc.gehennom.org/2013/09/atmega2560-as-an-spi-slave/ (дата обращения 08.03.2022).

15) Справка по Ассемблеру для Atmel AVR [Электронный ресурс] URL: https://dfe.karelia.ru/koi/posob/avrlab/avrasm-rus.htm (дата обращения 08.03.2022).

16) Первая программа для AVR микроконтроллера на Ассемблере [Электронный ресурс] URL: https://ph0en1x.net/79-avr-asm-first-program-for-microcontroller.html (дата обращения 10.03.2022).

17) Примеры на Ассемблере для микроконтроллеров Atmel AVR [Электронный ресурс] URL: https://smartep.ru/index.php?page=avr_asm_examples (дата обращения 16.03.2022).

18) <avr/interrupt.h>: Interrupts [Электронный ресурс] URL: https://www.nongnu.org/avr-libc/user-manual/group__avr__interrupts.html (дата обращения 16.03.2022).

19) AVR External Interrupts [Электронный ресурс] URL: https://exploreembedded.com/wiki/AVR_External_Interrupts (дата

обращения 16.03.2022).

Приложение А. Текст программы

АННОТАЦИЯ

В данном программном документе приведен текст приложения с предиктивной коррекцией ошибок управления (на примере ООО «Центр инновационных разработок ВАО»).

В разделе «Текст программы» указано назначение программы, краткая характеристика области применения программы, описание модулей и их программный код.

СОДЕРЖАНИЕ

1. ТЕКСТ ПРОГРАММЫ

1.1. Наименование программы

Наименование – Встроенное приложение с предиктивной коррекцией ошибок управления.

1.2. Область применения программы

Программа должна эксплуатироваться в составе программно-аппаратного комплекса в виде платформы-носителя с универсальным интерфейсом связи «MasterLink». Конечными пользователями программы должны являться сотрудники с допуском работы на промышленном оборудовании с автоматическим управлением подвижными частями.

1.3. Модули

Таблица 1 - Модули.

№	Название модуля	Описание модуля	Размер модуля	Кол-во строк
1	PlatformMain.cpp	Модуль основной программы	1,2 кб	36
2	Platform.h	Заголовочный файл библиотеки Platform	4,6 кб	147
3	Platform.cpp	Модуль логики библиотеки Platform	23,9 кб	628
4	Arduino.h	Заголовочный файл библиотеки Arduino	7,2 кб	260
5	Display.cpp	Модуль программы полезной нагрузки «Дисплей»	3,2 кб	116

1.4. Код программы

1.4.1. PlatformMain.cpp

```
#include "Platform.h"

Platform platform;

void setup() {
  pinMode(13, OUTPUT); //Debug signal
  Serial.begin(115200); //Debug or platform's load
  Serial1.begin(9600); //GPS
```

```
  platform.begin("testPlatf", "8tegqHu6VZ");
  platform.GPIOSetup(GPIO_DIGITALOUT, GPIO_DIGITALOUT, GPIO_DIGITALOUT,
GPIO_DIGITALOUT);
  platform.initUARTControlData(platform);
  platform.initMPU();
}

void loop() {
  while (1) { //Speed-up bug
    //PORTB |= (1 << 7); //13 test square generator
    //PORTB &= ~ (1 << 7); //13

    if (millis() % 50 == 0) {
      //platform.sendUARTControlData("^:asd;\r\n");
      platform.getGPSData(&Serial1);
      platform.getMPUData();
    }

    //  if (Serial.available() > 0) { //Segment for test bridge between PC and platform's load
    //    platform.sendUARTCommandData("^:" + Serial.readString() + ";");
    //  }

    //platform.startBench();
    //delay(500);
    //platform.getGPSData(&Serial1);
    //platform.stopBench(&Serial);
  }
}
```

1.4.2. Platform.h

```
#pragma once
#include <Arduino.h>
#include <avr/interrupt.h>
#include <Wire.h>
#define _LIB_VERSION        1.0

#define DEBUGGYRO           false
#define DEBUGACC            false
#define DEBUGUART           false
#define DEBUGGPS            false

#define MPU6050_ADDRESS     0x68
```

```
#define BACKWARD          0      // Move backward
#define FORWARD           1      // Move forward
#define LEFT              2      // Move counterclock-wise
#define RIGHT             3      // Move counterclock
#define FORWARDLEFT       4      // Move forward and left
#define FORWARDRIGHT      5      // Move forward and right
#define BACKWARDLEFT      6      // Move backward and left
#define BACKWARDRIGHT     7      // Move backward and right

#define BRAKE             1      // Value for rapid braking
#define STOP              0      // Value for inertional braking
#define FAST              0      // Value for rapid acceleration
#define SLOW              1      // Value for soft acceleration

#define STATUS_STOP       0      // Stop, command processing is discontinued
#define STATUS_WORK       1      // Work, exchange of commands
#define STATUS_SHUTDOWN   2      // Ready to Shut Down
#define STATUS_ECO        3      // Energy saving mode
#define STATUS_EMODE      4      // Emergency mode
#define STATUS_ERROR      5      // Unexpected system error
#define STATUS_EXEPTION   6      // Work, have problems

#define GPIO_OFF          0      // GPIO off
#define GPIO_DIGITALIN    1      // GPIO as digital input
#define GPIO_DIGITALOUT   2      // GPIO as digital output
#define GPIO_ANALOGIN     3      // GPIO as analog input

struct DataIncome {                          // Structure of data coming from PC to UART
  char move;
  uint8_t speed;
  char value;
  uint8_t azimuthloc;
  uint8_t gpio1 = 0;
  uint8_t gpio2 = 0;
  uint8_t gpio3 = 0;
  uint8_t gpio4 = 0;
  uint8_t systemstatus = 0;
  String data;
};

struct DataOutcome {                         // Data structure from UART to PC
  char move;
```

```
  uint8_t speed;
  char value;
  uint16_t lcurr;
  uint16_t rcurr;
  float accx;
  float accy;
  float accz;
  float gyrox;
  float gyroy;
  float gyroz;
  float magx;
  float magy;
  float magz;
  String lan;
  String lon;
  float vbat;
  uint8_t systemstatus = 0;
  uint16_t extid = 0;
  uint8_t extstatus = 0;
};

struct MainParameters {        // Data structure of platform parameters
  uint8_t systemstatus = 0;
  uint16_t extid = 0;
  uint8_t extstatus = 0;

  String GPSTimestamp ="";
  String GPSLatitude = "0.000000";
  String GPSLongitude = "0.000000";
};

class Platform {   // class Platform
  public:
    DataIncome controlDataIn;
    DataOutcome controlDataOut;
    MainParameters mainParameters;

    //GPIO mode
    uint8_t GPIO1 = 0;
    uint8_t GPIO2 = 0;
    uint8_t GPIO3 = 0;
    uint8_t GPIO4 = 0;
```

```
  //MPU6050 sensor
  volatile float AccX, AccY, AccZ;
  volatile float GyroX, GyroY, GyroZ;
  volatile float AccErrorX, AccErrorY, GyroErrorX, GyroErrorY, GyroErrorZ;
  volatile float Temperature;
  volatile int MPU_Calib_Counter = 0;
  volatile float AccDevider, GyroDevider = 0;

  Platform();
  void begin(String name, String key);

  //Movements section
  void makeMove(uint8_t direction, uint8_t speed, uint8_t acceleration);
  void brake(uint8_t mode);

  //Telemetry section
  bool initUARTControlData(Platform platform, int baudrate);
  bool initUARTControlData(Platform platform);
  void getUARTControlData(void);
  void sendUARTControlData(String outgoingDataString);
  bool getGPSData(Stream* _serial);
  void initMPU();
  void getMPUData();

  //MasterLink section
  void GPIOSetup(uint8_t GPIO_1, uint8_t GPIO_2, uint8_t GPIO_3, uint8_t GPIO_4);

  //Another useful functions
  void startBench();
  void stopBench(Stream* _serial);
  float convertRawCoordinatesToDegrees(float RawDegrees);
  void I2Cread(uint8_t Address, uint8_t Register, uint8_t Nbytes, uint8_t* Data);
  void I2CwriteByte(uint8_t Address, uint8_t Register, uint8_t Data);

private:
  String PlatformKey = "";              // Platform's private key
  String PlatformName = "";             // Platform's name

  //Move UART command section
  String stringUARTCommand = "";        // Variable of collection of accepted command characters per line
  volatile bool startedUARTCommandRecieve; // Variable odf uart command data recieve begin
  volatile uint8_t indexUARTCommand = 0;   // Index of accepted command mode argument
```

```
    //Load UART command section
    String stringUARTLoad = "";             // Variable of collecting accepted platform load symbols per string
    volatile bool startedUARTLoadRecieve;    // Platform load data start variable by uart
};
```

1.4.3. Platform.cpp

```
#include "Platform.h"

#define cbi(sfr, bit) (_SFR_BYTE(sfr) &= ~_BV(bit))
#define sbi(sfr, bit) (_SFR_BYTE(sfr) |= _BV(bit))

Platform ptf; // Class instance call

Platform::Platform() {} // Class constructor

void Platform::begin(String name, String key) {
  PlatformName = name;
  PlatformKey = key;



  pinMode(7, OUTPUT);     // Motor key A, 7
  pinMode(4, OUTPUT);     // Motor key A, 4
  pinMode(8, OUTPUT);     // Motor key B, 8
  pinMode(9, OUTPUT);     // Motor key B, 9
  pinMode(5, OUTPUT);     // Motor PWM pin, 5
  pinMode(6, OUTPUT);     // Motor PWM pin, 6
  pinMode(A2, INPUT);     // Current sensor pin, A2
  pinMode(A3, INPUT);     // Current sensor pin, A3
  pinMode(A7, INPUT);     // Voltage sensor pin, A7

  pinMode(52, OUTPUT);    // GPIO1 pin
  pinMode(50, OUTPUT);    // GPIO2 pin
  pinMode(51, OUTPUT);    // GPIO3 pin
  pinMode(53, OUTPUT);    // GPIO4 pin

  sbi(TCCR3A, COM3A1);    // PWM, 5
  sbi(TCCR4A, COM4A1);    // PWM, 6

  mainParameters.systemstatus = STATUS_WORK;
}

void Platform::makeMove(uint8_t direction, uint8_t speed, uint8_t acceleration) {
```

```
uint8_t dividerForRightMotor = 0;
uint8_t dividerForLeftMotor = 0;

PORTH &= ~ (1 << 4); //7, LOW A
PORTG &= ~ (1 << 5); //4, LOW A
PORTH &= ~ (1 << 5); //8, LOW B
PORTH &= ~ (1 << 6); //9, LOW B

switch (direction) {
  case 0:
    PORTH |= (1 << 4); //7, HIGH A
    PORTH |= (1 << 6); //9, HIGH B
    break;
  case 1:
    PORTG |= (1 << 5); //4, HIGH A
    PORTH |= (1 << 5); //8, HIGH B
    break;
  case 2:
    PORTH &= ~ (1 << 4); //7, LOW A
    PORTH |= (1 << 5); //8, HIGH B

    PORTG &= ~ (1 << 5); //4, LOW A
    PORTH |= (1 << 6); //9, HIGH B
    break;
  case 3:
    PORTH |= (1 << 4); //7, HIGH A
    PORTH &= ~ (1 << 5); //8, LOW B

    PORTG |= (1 << 5); //4, HIGH A
    PORTH &= ~ (1 << 6); //9, LOW B
    break;
  case 4:
    PORTG |= (1 << 5); //4, HIGH A
    PORTH |= (1 << 5); //8, HIGH B

    dividerForRightMotor = 5;
    dividerForLeftMotor = 0; //Decrease left speed
    break;
  case 5:
    PORTG |= (1 << 5); //4, HIGH A
    PORTH |= (1 << 5); //8, HIGH B

    dividerForRightMotor = 0; //Decrease right speed
```

```
    dividerForLeftMotor = 5;
    break;
  case 6:
    PORTH |= (1 << 4); //7, HIGH A
    PORTH |= (1 << 6); //9, HIGH B

    dividerForRightMotor = 5;
    dividerForLeftMotor = 0; //Decrease left speed
    break;
  case 7:
    PORTH |= (1 << 4); //7, HIGH A
    PORTH |= (1 << 6); //9, HIGH B

    dividerForRightMotor = 0; //Decrease right speed
    dividerForLeftMotor = 5;
    break;
 }

 /*if(acceleration == 1) {
        for(uint16_t i = 0; i <= map(speed, 0, 100, 0, 255); i++) { //Не работает. работает. да...
            analogWrite(pwmpin[0], i);
            analogWrite(pwmpin[1], i);
            delay(10);
        }
  }
  else {*/

 OCR3A = map(speed<5?speed:speed-dividerForLeftMotor, 0, 100, 0, 255); // set pwm duty
 OCR4A = map(speed<5?speed:speed-dividerForRightMotor, 0, 100, 0, 255);

 //}
}

void Platform::brake(uint8_t mode) {
 if (mode == 1) {
  OCR3A = 0; // set pwm duty
  OCR4A = 0;

  //Rapid braking, short circuit motor
  PORTH |= (1 << 4); //7, HIGH
  PORTG |= (1 << 5); //4, HIGH
  PORTH |= (1 << 5); //8, HIGH
  PORTH |= (1 << 6); //9, HIGH
```

```
    delay(50);

    //Return keys to low state
    PORTH &= ~ (1 << 4); //7, LOW
    PORTG &= ~ (1 << 5); //4, LOW
    PORTH &= ~ (1 << 5); //8, LOW
    PORTH &= ~ (1 << 6); //9, LOW
  }
  else {
    OCR3A = 0; // set pwm duty
    OCR4A = 0;

    //Soft inertional braking
    PORTH &= ~ (1 << 4); //7, LOW
    PORTG &= ~ (1 << 5); //4, LOW
    PORTH &= ~ (1 << 5); //8, LOW
    PORTH &= ~ (1 << 6); //9, LOW
  }
}

//Telemetry section

bool Platform::initUARTControlData(Platform platform, int baudrate) {
  UCSR2A = 1 << U2X1; //UCSR2A = 1 << U2X1 for 115200
  // assign the baud_setting, a.k.a. ubrr (USART Baud Rate Register)
  /* Set baud rate */
  UBRR2H = baudrate >> 8;
  UBRR2L = baudrate;

  //Permission to receive and transmit via USART, interrupts on arrival and on devastation
  UCSR2B = (1 << RXCIE2) | (1 << TXCIE2) | (1 << RXEN2) | (1 << TXEN2);
  UCSR2C = (1 << UCSZ21) | (1 << UCSZ20); //Word's size 8 bits
  sei();

  ptf = platform;
  return true;
}

bool Platform::initUARTControlData(Platform platform) {
  UCSR2A = 1 << U2X1;
  // assign the baud_setting, a.k.a. ubrr (USART Baud Rate Register)
  /* Set baud rate */
```

```
  UBRR2H = 34 >> 8; //Value '34' for 57600 baudrate
  UBRR2L = 34;

  //Permission to receive and transmit via USART, interrupts on arrival and on devastation
  UCSR2B = (1 << RXCIE2) | (1 << TXCIE2) | (1 << RXEN2) | (1 << TXEN2);
  UCSR2C = (1 << UCSZ21) | (1 << UCSZ20); //Word's size 8 bits
  sei();

  ptf = platform;
  return true;
}



ISR(USART2_RX_vect) { //ISR UART2 handler
  if(ptf.mainParameters.systemstatus != STATUS_EMODE) ptf.getUARTControlData();
}

void Platform::getUARTControlData(void) {
  while ( !(UCSR2A & (1 << RXC2)) );
  char incomingByte = UDR2; // Read income char

  //--------------------------------------------------Who am I section-----------------------------------------------
  if (incomingByte == '@' && !startedUARTCommandRecieve && !startedUARTLoadRecieve) {
    sendUARTControlData("@:"+PlatformName+","+PlatformKey+";");
  }

  //--------------------------------------------------Load UART command section-------------------------------------
-------
  if (incomingByte == '*') {
    startedUARTLoadRecieve = true;
    stringUARTLoad = "";
  }
  if (incomingByte != ';' && startedUARTLoadRecieve) stringUARTLoad += incomingByte;
  else {
    stringUARTLoad += ";";

    for (uint32_t i = 0; i <= strlen(stringUARTLoad.c_str()); ++i) { //UART0 transmit
      /* Wait for empty transmit buffer */
      while ( !( UCSR0A & (1 << UDRE0)) );
      /* Put data into buffer, sends the data */
      UDR0 = stringUARTLoad[i];
    }
```

```
    startedUARTLoadRecieve = false;
    stringUARTLoad = "";
  }

  //-------------------------------------------------Move UART command section----------------------------------------
--------
  if (incomingByte != ',' && incomingByte != ';' && startedUARTCommandRecieve &&
!startedUARTLoadRecieve) {  // if it isn't space and end
    stringUARTCommand += incomingByte;                                  // Add to sting
  } else {                                                  // If it's a space or ;
      switch (indexUARTCommand) {
        case 0:
          controlDataIn.move = stringUARTCommand[1];
          break;
        case 1:
          controlDataIn.speed = stringUARTCommand.toInt();
          break;
        case 2:
          controlDataIn.value = stringUARTCommand[0];
          break;
        case 3:
          controlDataIn.azimuthloc = stringUARTCommand.toInt();
          break;
        case 4:
          controlDataIn.gpio1 = stringUARTCommand.toFloat();
          if(GPIO1 == GPIO_DIGITALOUT) digitalWrite(52, stringUARTCommand.toFloat());
          break;
        case 5:
          controlDataIn.gpio2 = stringUARTCommand.toFloat();
          if(GPIO2 == GPIO_DIGITALOUT) digitalWrite(50, stringUARTCommand.toFloat());
          break;
        case 6:
          controlDataIn.gpio3 = stringUARTCommand.toFloat();
          if(GPIO3 == GPIO_DIGITALOUT) digitalWrite(51, stringUARTCommand.toFloat());
          break;
        case 7:
          controlDataIn.gpio4 = stringUARTCommand.toFloat();
          if(GPIO4 == GPIO_DIGITALOUT) digitalWrite(53, stringUARTCommand.toFloat());
          break;
        case 8:
          controlDataIn.systemstatus = stringUARTCommand.toInt();
          ptf.mainParameters.systemstatus = controlDataIn.systemstatus;
          break;
```

```
      case 9:
        controlDataIn.data = stringUARTCommand;
        break;
      }
    stringUARTCommand = "";                    // Clear string
    indexUARTCommand++;                        // Select next parsing section of array
  }
  if (incomingByte == '%') {
    startedUARTCommandRecieve = true;
    indexUARTCommand = 0;
    stringUARTCommand = "";
  }
  if (incomingByte == ';' && startedUARTCommandRecieve) {
    startedUARTCommandRecieve = false;

    //Заполняем структуру и передаем её
    if(mainParameters.systemstatus != STATUS_STOP && mainParameters.systemstatus !=
STATUS_EMODE) {
      controlDataOut.move = controlDataIn.move;
      controlDataOut.speed = controlDataIn.speed;
      controlDataOut.value = controlDataIn.value;
    }
    controlDataOut.lcurr = analogRead(A3) * 0.038;   //Current in Amps
    controlDataOut.rcurr = analogRead(A2) * 0.038;
    // controlDataOut.accx = AccX;
    // controlDataOut.accy = AccY;
    // controlDataOut.accz = AccZ;
    // controlDataOut.gyrox = GyroX;
    // controlDataOut.gyroy = GyroY;
    // controlDataOut.gyroz = GyroZ;
    controlDataOut.magx = 0;
    controlDataOut.magy = 0;
    controlDataOut.magz = 0;
    controlDataOut.lan = mainParameters.GPSLatitude;
    controlDataOut.lon = mainParameters.GPSLongitude;
    controlDataOut.vbat = ((analogRead(A7)* 5.0) / 1024.0)/0.337;
    controlDataOut.systemstatus = mainParameters.systemstatus;
    controlDataOut.extid = mainParameters.extid;
    controlDataOut.extstatus = mainParameters.systemstatus;

    //Serial.println(ptf.controlDataOut.gyrox);

    String outgoingDataString = "&:" + String(controlDataOut.move) + "," + String(controlDataOut.speed) +
```

```
"," + String(controlDataOut.value) + "," + String(controlDataOut.lcurr) + "," + String(controlDataOut.rcurr)
+ "," + String(controlDataOut.accx) + "," + String(controlDataOut.accy) + "," + String(controlDataOut.accz)
+ "," + String(controlDataOut.gyrox) + "," + String(controlDataOut.gyroy) + "," +
String(controlDataOut.gyroz) + "," + String(controlDataOut.magx) + "," + String(controlDataOut.magy) +
"," + String(controlDataOut.magz) + "," + controlDataOut.lan + "," + controlDataOut.lon + "," +
String(controlDataOut.vbat) + "," + String(controlDataOut.systemstatus) + "," + String(controlDataOut.extid)
+ "," + String(controlDataOut.extstatus) + ";\r\n";
   //String outgoingDataString = "&:" +PlatformName+"," + String(controlDataOut.move) + "," +
String(controlDataOut.speed) + "," + String(controlDataOut.value) + "," + String(controlDataOut.lcurr) + ","
+ String(controlDataOut.rcurr) + "," + String(controlDataOut.accx) + "," + String(controlDataOut.accy) + ","
+ String(controlDataOut.accz) + "," + String(controlDataOut.gyrox) + "," + String(controlDataOut.gyroy) +
"," + String(controlDataOut.gyroz) + "," + String(controlDataOut.magx) + "," +
String(controlDataOut.magy) + "," + String(controlDataOut.magz) + "," + controlDataOut.lan + "," +
controlDataOut.lon + "," + String(controlDataOut.vbat) + "," + String(controlDataOut.extid) + "," +
String(controlDataOut.extstatus) + ";\r\n";

   sendUARTControlData(outgoingDataString);

   if(ptf.mainParameters.systemstatus != STATUS_STOP && ptf.mainParameters.systemstatus !=
STATUS_EMODE) {
     switch (controlDataIn.move) {
       case 'f':
         makeMove(FORWARD, controlDataIn.speed, (controlDataIn.value == 'f') ? FAST : SLOW);
         break;
       case 'b':
         makeMove(BACKWARD, controlDataIn.speed, (controlDataIn.value == 'f') ? FAST : SLOW);
         break;
       case 'l':
         makeMove(LEFT, controlDataIn.speed, (controlDataIn.value == 'f') ? FAST : SLOW);
         break;
       case 'r':
         makeMove(RIGHT, controlDataIn.speed, (controlDataIn.value == 'f') ? FAST : SLOW);
         break;
       case 'a':
         makeMove(FORWARDLEFT, controlDataIn.speed, (controlDataIn.value == 'f') ? FAST : SLOW);
         break;
       case 'c':
         makeMove(FORWARDRIGHT, controlDataIn.speed, (controlDataIn.value == 'f') ? FAST : SLOW);
         break;
       case 'd':
         makeMove(BACKWARDLEFT, controlDataIn.speed, (controlDataIn.value == 'f') ? FAST : SLOW);
         break;
       case 'e':
```

```
        makeMove(BACKWARDRIGHT, controlDataIn.speed, (controlDataIn.value == 'f') ? FAST : SLOW);
        break;
       case 's':
        brake(STOP);
        break;
     }
    }
    else brake(BRAKE);
  }
}

void Platform::sendUARTControlData(String outgoingDataString)
{
  for (uint32_t i = 0; i <= strlen(outgoingDataString.c_str()); ++i) {
    /* Wait for empty transmit buffer */
    while ( !( UCSR2A & (1 << UDRE2)) );
    /* Put data into buffer, sends the data */
    UDR2 = outgoingDataString[i];
  }
}

bool Platform::getGPSData(Stream* _serial) {
  String stringGPS = "";
  if (_serial->available() > 0) {
    stringGPS = _serial->readStringUntil(13); //NMEA data ends with 'return' character, which is ascii(13)
    stringGPS.trim();                    // they say NMEA data starts with "$", but the Arduino doesn't think so.
    //Serial.println(stringGPS);         //All the raw sentences will be sent to monitor, if you want them, maybe
to see the labels and data order.

    //Start Parsing by finding data, put it in a string of character array, then removing it, leaving the rest of thes
sentence for the next 'find'
    if (stringGPS.startsWith("$GPGLL") || stringGPS.startsWith("$GLGLL") ||
stringGPS.startsWith("$GAGLL") || stringGPS.startsWith("$BDGLL") || stringGPS.startsWith("$GQGLL") ||
stringGPS.startsWith("$GNGLL")) { //I picked this sentence, you can pick any of the other labels and
rearrange/add sections as needed.
      //Serial.println(stringGPS);     // display raw GLL data in Serial Monitor
      // mine looks like this: "$GPGLL,4053.16598,N,10458.93997,E,224431.00,A,D*7D"

      //This section gets repeated for each delimeted bit of data by looking for the commas
      //Find Lattitude is first in GLL sentence, other senetences have data in different order
      int Pos = stringGPS.indexOf(','); //look for comma delimetrer
      stringGPS.remove(0, Pos + 1); // Remove Pos+1 characters starting at index=0, this one strips off
"$GPGLL" in my sentence
```

```
    Pos = stringGPS.indexOf(','); //looks for next comma delimetrer, which is now the first comma because I removed the first segment
    char Lat[Pos];           //declare character array Lat with a size of the dbit of data
    for (int i = 0; i <= Pos - 1; i++) { // load charcters into array
      Lat[i] = stringGPS.charAt(i);
    }
    //Serial.print(Lat);         // display raw latitude data in Serial Monitor, I'll use Lat again in a few lines for converting
    //repeating with a different char array variable
    //Get Lattitude North or South
    stringGPS.remove(0, Pos + 1);
    Pos = stringGPS.indexOf(',');
    char LatSide[Pos];          //declare different variable name
    for (int i = 0; i <= Pos - 1; i++) {
      LatSide[i] = stringGPS.charAt(i); //fill the array
      //Serial.println(LatSide[i]);       //display N or S
    }

    //convert the variable array Lat to degrees Google can use
    float LatAsFloat = atof (Lat);           //atof converts the char array to a float type
    float LatInDeg;
    if (LatSide[0] == char(78)) {     //char(69) is decimal for the letter "N" in ascii chart
      LatInDeg = convertRawCoordinatesToDegrees(LatAsFloat);  //call the conversion funcion (see below)
    }
    if (LatSide[0] == char(83)) {     //char(69) is decimal for the letter "S" in ascii chart
      LatInDeg = -( convertRawCoordinatesToDegrees(LatAsFloat));  //call the conversion funcion (see below)
    }
    if(LatInDeg > 0 && String(LatInDeg, 8) != "") ptf.mainParameters.GPSLatitude = String(LatInDeg, 8); //TEMP SOLUTION
    //Serial.println(LatInDeg, 15); //display value Google can use in Serial Monitor, set decimal point value high
    //repeating with a different char array variable
    //Get Longitude
    stringGPS.remove(0, Pos + 1);
    Pos = stringGPS.indexOf(',');
    char Longit[Pos];           //declare different variable name
    for (int i = 0; i <= Pos - 1; i++) {
      Longit[i] = stringGPS.charAt(i);    //fill the array
    }
    //Serial.print(Longit);      //display raw longitude data in Serial Monitor
    //repeating with a different char array variable
    //Get Longitude East or West
```

```
    stringGPS.remove(0, Pos + 1);
    Pos = stringGPS.indexOf(',');
    char LongitSide[Pos];        //declare different variable name
    for (int i = 0; i <= Pos - 1; i++) {
      LongitSide[i] = stringGPS.charAt(i);    //fill the array
      //Serial.println(LongitSide[i]);        //display raw longitude data in Serial Monitor
    }
    //convert to degrees Google can use
    float LongitAsFloat = atof (Longit);    //atof converts the char array to a float type
    float LongInDeg;
    if (LongitSide[0] == char(69)) {     //char(69) is decimal for the letter "E" in ascii chart
      LongInDeg = convertRawCoordinatesToDegrees(LongitAsFloat); //call the conversion funcion (see
below
    }
    if (LongitSide[0] == char(87)) {      //char(87) is decimal for the letter "W" in ascii chart
      LongInDeg = -(convertRawCoordinatesToDegrees(LongitAsFloat)); //call the conversion funcion (see
below
    }
    if(LongInDeg > 0 && String(LongInDeg, 8) != "") ptf.mainParameters.GPSLongitude =
String(LongInDeg, 8); //TEMP SOLUTION
    //Serial.println(LongInDeg, 15); //display value Google can use in Serial Monitor, set decimal point value
high
    //repeating with a different char array variable
    //Get TimeStamp - GMT
    stringGPS.remove(0, Pos + 1);
    Pos = stringGPS.indexOf(',');
    char TimeStamp[Pos];         //declare different variable name
    for (int i = 0; i <= Pos - 1; i++) {
      TimeStamp[i] = stringGPS.charAt(i);       //fill the array
    }
    ptf.mainParameters.GPSTimestamp = TimeStamp; //TEMP SOLUTION
    //Serial.print(TimeStamp);   //display raw longitude data in Serial Monitor, GMT
    //Serial.println(String(LongInDeg, 8));
   }
  }
  return true;
}

void Platform::initMPU() {
  Wire.begin();
  Wire.setClock(400000);

  I2CwriteByte(MPU6050_ADDRESS, 29, 0x06);// Set accelerometers low pass filter at 5Hz !
```

```
 I2CwriteByte(MPU6050_ADDRESS, 26, 0x06); // Set gyroscope low pass filter at 5Hz !

 // Configure gyroscope range
 I2CwriteByte(MPU6050_ADDRESS, 27, 0x6B); GyroDevider = 131; //GYRO_FULL_SCALE_250_DPS !
 //I2CwriteByte(MPU6050_ADDRESS, 27, 0x08); GyroDevider = 65.5; //GYRO_FULL_SCALE_500_DPS
 //I2CwriteByte(MPU6050_ADDRESS, 27, 0x10); GyroDevider = 32.8;
//GYRO_FULL_SCALE_1000_DPS
 // I2CwriteByte(MPU6050_ADDRESS, 27, 0x18); GyroDevider = 16.4;
//GYRO_FULL_SCALE_2000_DPS

 // Configure accelerometers range
 I2CwriteByte(MPU6050_ADDRESS, 28, 0x00); AccDevider = 16384; //ACC_FULL_SCALE_2_G !
 //I2CwriteByte(MPU6050_ADDRESS, 28, 0x08); AccDevider = 8192; //ACC_FULL_SCALE_4_G
 //I2CwriteByte(MPU6050_ADDRESS, 28, 0x10); AccDevider = 4096; //ACC_FULL_SCALE_8_G
 //I2CwriteByte(MPU6050_ADDRESS, 28, 0x18); AccDevider = 2048; //ACC_FULL_SCALE_16_G


 while (MPU_Calib_Counter < 200) {
  uint8_t Buf[14];
  I2Cread(MPU6050_ADDRESS, 0x3B, 14, Buf);

  //Get values from sensor
  GyroX = -(Buf[0] << 8 | Buf[1]);
  GyroY = -(Buf[2] << 8 | Buf[3]);
  GyroZ = Buf[4] << 8 | Buf[5];

  // Sum all readings
  GyroErrorX = GyroErrorX + (GyroX / GyroDevider);
  GyroErrorY = GyroErrorY + (GyroY / GyroDevider);
  GyroErrorZ = GyroErrorZ + (GyroZ / GyroDevider);
  MPU_Calib_Counter++;
 }

 //Divide the sum by 200 to get the error value
 GyroErrorX = GyroErrorX / 200;
 GyroErrorY = GyroErrorY / 200;
 GyroErrorZ = GyroErrorZ / 200;
 MPU_Calib_Counter = 0;

 while (MPU_Calib_Counter < 200) {
  uint8_t Buf[14];
  I2Cread(MPU6050_ADDRESS, 0x3B, 14, Buf);
```

```
    //Get values from sensor
    AccX = (Buf[8] << 8 | Buf[9]) / AccDevider;
    AccY = (Buf[10] << 8 | Buf[11]) / AccDevider;
    AccZ = (Buf[12] << 8 | Buf[13]) / AccDevider;

    // Sum all readings
    AccErrorX = AccErrorX + ((atan((AccY) / sqrt(pow((AccX), 2) + pow((AccZ), 2))) * 180 / PI));
    AccErrorY = AccErrorY + ((atan(-1 * (AccX) / sqrt(pow((AccY), 2) + pow((AccZ), 2))) * 180 / PI));
    MPU_Calib_Counter++;
  }

  //Divide the sum by 200 to get the error value
  AccErrorX = AccErrorX / 200;
  AccErrorY = AccErrorY / 200;
  MPU_Calib_Counter = 0;

#if DEBUGGYRO || DEBUGACC
  Serial.print(F("AccErrorX: "));
  Serial.println(AccErrorX);
  Serial.print(F("AccErrorY: "));
  Serial.println(AccErrorY);
  Serial.print(F("GyroErrorX: "));
  Serial.println(GyroErrorX);
  Serial.print(F("GyroErrorY: "));
  Serial.println(GyroErrorY);
  Serial.print(F("GyroErrorZ: "));
  Serial.println(GyroErrorZ);
#endif
}

void Platform::getMPUData() {
  uint8_t Buf[14];

  I2Cread(MPU6050_ADDRESS, 0x3B, 14, Buf); // Read accelerometer and gyroscope

  //Gyroscope
  GyroX = (Buf[0] << 8 | Buf[1]) / GyroDevider;
  GyroY = (Buf[2] << 8 | Buf[3]) / GyroDevider;
  GyroZ = (Buf[4] << 8 | Buf[5]) / GyroDevider;

  // Correct the outputs with the calculated error values
  GyroX = GyroX + abs(GyroErrorX); // GyroErrorX ~(-0.56)
  GyroY = GyroY + abs(GyroErrorY); // GyroErrorY ~(2)
```

```
  GyroZ = GyroZ + abs(GyroErrorZ); // GyroErrorZ ~ (-0.8)

  //Temperature
  Temperature = (Buf[6] << 8 | Buf[7]) / 340.0 + 36.53;

  // Accelerometer
  AccX = (Buf[8] << 8 | Buf[9]) / AccDevider;
  AccY = (Buf[10] << 8 | Buf[11]) / AccDevider;
  AccZ = (Buf[12] << 8 | Buf[13]) / AccDevider;

  // Display values
  ptf.controlDataOut.accx = AccX;
  ptf.controlDataOut.accy = AccY;
  ptf.controlDataOut.accz = AccZ;
  ptf.controlDataOut.gyrox = GyroX;
  ptf.controlDataOut.gyroy = GyroY;
  ptf.controlDataOut.gyroz = GyroZ;
  //Serial.println(ptf.controlDataOut.gyrox);

  // Gyroscope
#if DEBUGGYRO
  Serial.print(F("GyroX: "));
  Serial.println((int)GyroX, DEC);
  Serial.print(F("GyroY: "));
  Serial.println((int)GyroY, DEC);
  Serial.print(F("GyroZ: "));
  Serial.println((int)GyroZ, DEC);
  Serial.println((int)Temperature, DEC);
#endif

  // Accelerometer
#if DEBUGACC
  Serial.print(F("AccX: "));
    Serial.println(AccX, DEC);
    Serial.print(F("AccY: "));
    Serial.println(AccY, DEC);
    Serial.print(F("AccZ: "));
    Serial.println (AccZ, DEC);
#endif

}

//MasterLink section
```

```
void Platform::GPIOSetup(uint8_t GPIO_1, uint8_t GPIO_2, uint8_t GPIO_3, uint8_t GPIO_4) {
  GPIO1 = GPIO_1;
  GPIO2 = GPIO_2;
  GPIO3 = GPIO_3;
  GPIO4 = GPIO_4;

  if(GPIO_1 == GPIO_OFF || GPIO_1 == GPIO_DIGITALOUT) pinMode(52, OUTPUT);
  else pinMode(52, INPUT);

  if(GPIO_2 == GPIO_OFF || GPIO_2 == GPIO_DIGITALOUT) pinMode(50, OUTPUT);
  else pinMode(50, INPUT);

  if(GPIO_3 == GPIO_OFF || GPIO_3 == GPIO_DIGITALOUT) pinMode(51, OUTPUT);
  else pinMode(51, INPUT);

  if(GPIO_4 == GPIO_OFF || GPIO_4 == GPIO_DIGITALOUT) pinMode(53, OUTPUT);
  else pinMode(53, INPUT);
}

//Another useful functions
void Platform::startBench() {
  TCCR1A = 0x00;          // Turn off
  TCCR1B = 0x00;          // Turn off
  TCNT1 = 0x00;           // Reset counter
  TCCR1B = 0x01;          // Start timer
}

void Platform::stopBench(Stream* _serial) {
  TCCR1B = 0x00;          // Stop timer
  uint32_t count = TCNT1 - 2; // Minus 2 ticks on actions

  _serial->print("ticks: ");
  _serial->print(count);
  _serial->print("  ");
  _serial->print("time (us): ");
  _serial->println(count * (float)(1000000.0f / F_CPU), 4);
}

float Platform::convertRawCoordinatesToDegrees(float RawDegrees) {
  float RawAsFloat = RawDegrees;
  int firstdigits = ((int)RawAsFloat) / 100; // Get the first digits by turning f into an integer, then doing an
integer divide by 100;
  float nexttwodigits = RawAsFloat - (float)(firstdigits * 100);
```

```
  float Converted = (float)(firstdigits + nexttwodigits / 60.0);
  return Converted;
}

void Platform::I2Cread(uint8_t Address, uint8_t Register, uint8_t Nbytes, uint8_t* Data)
{
  // Set register address
  Wire.beginTransmission(Address);
  Wire.write(Register);
  Wire.endTransmission();

  // Read Nbytes
  Wire.requestFrom(Address, Nbytes);
  uint8_t index = 0;
  while (Wire.available())
    Data[index++] = Wire.read();
}

void Platform::I2CwriteByte(uint8_t Address, uint8_t Register, uint8_t Data)
{
  // Set register address
  Wire.beginTransmission(Address);
  Wire.write(Register);
  Wire.write(Data);
  Wire.endTransmission();
}
```

1.4.4. Arduino.h

```
/*
  Arduino.h - Main include file for the Arduino SDK
  Copyright (c) 2005-2013 Arduino Team.  All right reserved.

  This library is free software; you can redistribute it and/or
  modify it under the terms of the GNU Lesser General Public
  License as published by the Free Software Foundation; either
  version 2.1 of the License, or (at your option) any later version.

  This library is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
  Lesser General Public License for more details.

  You should have received a copy of the GNU Lesser General Public
```

```
  License along with this library; if not, write to the Free Software
  Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
*/

#ifndef Arduino_h
#define Arduino_h

#include <stdlib.h>
#include <stdbool.h>
#include <string.h>
#include <math.h>

#include <avr/pgmspace.h>
#include <avr/io.h>
#include <avr/interrupt.h>

#include "binary.h"

#ifdef __cplusplus
extern "C"{
#endif

void yield(void);

#define HIGH 0x1
#define LOW  0x0

#define INPUT 0x0
#define OUTPUT 0x1
#define INPUT_PULLUP 0x2

#define PI 3.14159265358979323846264338327950
#define HALF_PI 1.57079632679489661923132169163975
#define TWO_PI 6.28318530717958647692528676655900
#define DEG_TO_RAD 0.017453292519943295769236907684886
#define RAD_TO_DEG 57.295779513082320876798154814105
#define EULER 2.71828182845904523536028747135266

#define SERIAL  0x0
#define DISPLAY 0x1

#define LSBFIRST 0
#define MSBFIRST 1
```

```
#define CHANGE 1
#define FALLING 2
#define RISING 3

#if defined(__AVR_ATtiny24__) || defined(__AVR_ATtiny44__) || defined(__AVR_ATtiny84__)
  #define DEFAULT 0
  #define EXTERNAL 1
  #define INTERNAL1V1 2
  #define INTERNAL INTERNAL1V1
#elif defined(__AVR_ATtiny25__) || defined(__AVR_ATtiny45__) || defined(__AVR_ATtiny85__)
  #define DEFAULT 0
  #define EXTERNAL 4
  #define INTERNAL1V1 8
  #define INTERNAL INTERNAL1V1
  #define INTERNAL2V56 9
  #define INTERNAL2V56_EXTCAP 13
#else
#if defined(__AVR_ATmega1280__) || defined(__AVR_ATmega2560__) ||
defined(__AVR_ATmega1284__) || defined(__AVR_ATmega1284P__) || defined(__AVR_ATmega644__) ||
defined(__AVR_ATmega644A__) || defined(__AVR_ATmega644P__) ||
defined(__AVR_ATmega644PA__)
#define INTERNAL1V1 2
#define INTERNAL2V56 3
#else
#define INTERNAL 3
#endif
#define DEFAULT 1
#define EXTERNAL 0
#endif

// undefine stdlib's abs if encountered
#ifdef abs
#undef abs
#endif

#define min(a,b) ((a)<(b)?(a):(b))
#define max(a,b) ((a)>(b)?(a):(b))
#define abs(x) ((x)>0?(x):-(x))
#define constrain(amt,low,high) ((amt)<(low)?(low):((amt)>(high)?(high):(amt)))
#define round(x)     ((x)>=0?(long)((x)+0.5):(long)((x)-0.5))
#define radians(deg) ((deg)*DEG_TO_RAD)
#define degrees(rad) ((rad)*RAD_TO_DEG)
```

```
#define sq(x) ((x)*(x))

#define interrupts() sei()
#define noInterrupts() cli()

#define clockCyclesPerMicrosecond() ( F_CPU / 1000000L )
#define clockCyclesToMicroseconds(a) ( (a) / clockCyclesPerMicrosecond() )
#define microsecondsToClockCycles(a) ( (a) * clockCyclesPerMicrosecond() )

#define lowByte(w) ((uint8_t) ((w) & 0xff))
#define highByte(w) ((uint8_t) ((w) >> 8))

#define bitRead(value, bit) (((value) >> (bit)) & 0x01)
#define bitSet(value, bit) ((value) |= (1UL << (bit)))
#define bitClear(value, bit) ((value) &= ~(1UL << (bit)))
#define bitWrite(value, bit, bitvalue) (bitvalue ? bitSet(value, bit) : bitClear(value, bit))

// avr-libc defines _NOP() since 1.6.2
#ifndef _NOP
#define _NOP() do { __asm__ volatile ("nop"); } while (0)
#endif

typedef unsigned int word;

#define bit(b) (1UL << (b))

typedef bool boolean;
typedef uint8_t byte;

void init(void);
void initVariant(void);

int atexit(void (*func)()) __attribute__((weak));

void pinMode(uint8_t, uint8_t);
void digitalWrite(uint8_t, uint8_t);
int digitalRead(uint8_t);
int analogRead(uint8_t);
void analogReference(uint8_t mode);
void analogWrite(uint8_t, int);

unsigned long millis(void);
unsigned long micros(void);
```

```
void delay(unsigned long);
void delayMicroseconds(unsigned int us);
unsigned long pulseIn(uint8_t pin, uint8_t state, unsigned long timeout);
unsigned long pulseInLong(uint8_t pin, uint8_t state, unsigned long timeout);

void shiftOut(uint8_t dataPin, uint8_t clockPin, uint8_t bitOrder, uint8_t val);
uint8_t shiftIn(uint8_t dataPin, uint8_t clockPin, uint8_t bitOrder);

void attachInterrupt(uint8_t, void (*)(void), int mode);
void detachInterrupt(uint8_t);

void setup(void);
void loop(void);

// Get the bit location within the hardware port of the given virtual pin.
// This comes from the pins_*.c file for the active board configuration.

#define analogInPinToBit(P) (P)

// On the ATmega1280, the addresses of some of the port registers are
// greater than 255, so we can't store them in uint8_t's.
extern const uint16_t PROGMEM port_to_mode_PGM[];
extern const uint16_t PROGMEM port_to_input_PGM[];
extern const uint16_t PROGMEM port_to_output_PGM[];

extern const uint8_t PROGMEM digital_pin_to_port_PGM[];
// extern const uint8_t PROGMEM digital_pin_to_bit_PGM[];
extern const uint8_t PROGMEM digital_pin_to_bit_mask_PGM[];
extern const uint8_t PROGMEM digital_pin_to_timer_PGM[];

// Get the bit location within the hardware port of the given virtual pin.
// This comes from the pins_*.c file for the active board configuration.
//
// These perform slightly better as macros compared to inline functions
//
#define digitalPinToPort(P) ( pgm_read_byte( digital_pin_to_port_PGM + (P) ) )
#define digitalPinToBitMask(P) ( pgm_read_byte( digital_pin_to_bit_mask_PGM + (P) ) )
#define digitalPinToTimer(P) ( pgm_read_byte( digital_pin_to_timer_PGM + (P) ) )
#define analogInPinToBit(P) (P)
#define portOutputRegister(P) ( (volatile uint8_t *)( pgm_read_word( port_to_output_PGM + (P)) ) )
#define portInputRegister(P) ( (volatile uint8_t *)( pgm_read_word( port_to_input_PGM + (P)) ) )
#define portModeRegister(P) ( (volatile uint8_t *)( pgm_read_word( port_to_mode_PGM + (P)) ) )
```

```
#define NOT_A_PIN 0
#define NOT_A_PORT 0

#define NOT_AN_INTERRUPT -1

#ifdef ARDUINO_MAIN
#define PA 1
#define PB 2
#define PC 3
#define PD 4
#define PE 5
#define PF 6
#define PG 7
#define PH 8
#define PJ 10
#define PK 11
#define PL 12
#endif

#define NOT_ON_TIMER 0
#define TIMER0A 1
#define TIMER0B 2
#define TIMER1A 3
#define TIMER1B 4
#define TIMER1C 5
#define TIMER2  6
#define TIMER2A 7
#define TIMER2B 8

#define TIMER3A 9
#define TIMER3B 10
#define TIMER3C 11
#define TIMER4A 12
#define TIMER4B 13
#define TIMER4C 14
#define TIMER4D 15
#define TIMER5A 16
#define TIMER5B 17
#define TIMER5C 18

#ifdef __cplusplus
} // extern "C"
#endif
```

```
#ifdef __cplusplus
#include "WCharacter.h"
#include "WString.h"
#include "HardwareSerial.h"
#include "USBAPI.h"
#if defined(HAVE_HWSERIAL0) && defined(HAVE_CDCSERIAL)
#error "Targets with both UART0 and CDC serial not supported"
#endif

uint16_t makeWord(uint16_t w);
uint16_t makeWord(byte h, byte l);

#define word(...) makeWord(__VA_ARGS__)

unsigned long pulseIn(uint8_t pin, uint8_t state, unsigned long timeout = 1000000L);
unsigned long pulseInLong(uint8_t pin, uint8_t state, unsigned long timeout = 1000000L);

void tone(uint8_t _pin, unsigned int frequency, unsigned long duration = 0);
void noTone(uint8_t _pin);

// WMath prototypes
long random(long);
long random(long, long);
void randomSeed(unsigned long);
long map(long, long, long, long, long);

#endif

#include "pins_arduino.h"

#endif
```

1.4.5. Display.cpp

```
#include <SPI.h>
#include <Adafruit_GFX.h>
#include <Max72xxPanel.h>

Max72xxPanel matrix = Max72xxPanel(5, 1, 1);
int wait = 100; // In milliseconds
int spacer = 1;
int width = 5 + spacer; // The font width is 5 pixels
```

```
String stringUART = ""; // Переменная сбора принятых командных символов в строку
bool startedUART; // переменная начала приема командных данных по uart
uint8_t indexUART = 0; // Индекс принятого аргумента командного режима

byte mask[8] = {
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000,
  0b00000000
};

String receivedTicker = "";

void setup() {
  Serial.begin(115200);
  //ticker("");
  //pixelsDraw();
  matrix.fillScreen(LOW);
  matrix.write();
}

void loop() {
  if (Serial.available() > 0) {
    char incomingByte = Serial.read();
    if (incomingByte != ',' && incomingByte != ';') stringUART += incomingByte;
    else {
      switch (indexUART) {
        case 0:
          receivedTicker = stringUART;
          receivedTicker.replace(":", "");
          break;
        case 1:
          mask[0] = (byte)stringUART.toInt();
          break;
        case 2:
          mask[1] = (byte)stringUART.toInt();
          break;
        case 3:
```

```
        mask[2] = (byte)stringUART.toInt();
        break;
      case 4:
        mask[3] = (byte)stringUART.toInt();
        break;
      case 5:
        mask[4] = (byte)stringUART.toInt();
        break;
      case 6:
        mask[5] = (byte)stringUART.toInt();
        break;
      case 7:
        mask[6] = (byte)stringUART.toInt();
        break;
      case 8:
        mask[7] = (byte)stringUART.toInt();
        break;
    }
    stringUART = "";                    // очищаем строку
    indexUART++;                        // переходим к парсингу следующего элемента массива
   }
   if (incomingByte == '*') {                   // если это *
    startedUART = true;                 // поднимаем флаг, что можно парсить
    indexUART = 0;                      // сбрасываем индекс
    stringUART = "";                    // очищаем строку
   }
   if (incomingByte == ';') {  // если таки приняли ; - конец парсинга
    startedUART = false;                // сброс
    if(receivedTicker.length() > 0) ticker(receivedTicker);
    else pixelsDraw();
   }
  }
}

void pixelsDraw() {
  for (int y = 0; y < 8; y++ ) {          // Передача массива
    for (int x = 0; x < 8; x++ ) {
      matrix.drawPixel(x, y, mask[y] & (1 << x));
    }
  }
  matrix.write();
}
```

```
void ticker(String tape) {
  for ( int i = 0 ; i < width * tape.length() + matrix.width() - spacer; i++ )
  {
    matrix.fillScreen(LOW);

    int letter = i / width;                  // номер символа выводимого на матрицу

    int x = (matrix.width() - 1) - i % width;
    int y = (matrix.height() - 8) / 2;         // отцентрировать текст по вертикали

    while ( x + width - spacer >= 0 && letter >= 0 ) {
      if ( letter < tape.length() ) {
        matrix.drawChar(x, y, tape[letter], HIGH, LOW, 1);
      }
      letter--;
      x -= width;
    }
    matrix.write();                      // выведим значения на матрицу
    delay(wait);
  }
  receivedTicker = "";
}
```

Приложение Б. Сценарий и результаты тестовых испытаний

АННОТАЦИЯ

В данном программном документе приведен сценарий тестовых испытаний встроенного приложения с предиктивной коррекцией ошибок управления (на примере ООО «Центр инновационных разработок ВАО»)

В разделе «Цель испытаний» указана цель проведения испытаний.

В разделе «Требования к программе» указаны требования, подлежащие проверке во время испытаний и заданные в техническом задании на программу

В данном программном документе, в разделе «Требования к программной документации» указан состав программной документации, предъявляемый на испытания.

В разделе «Средства и порядок испытаний» указаны технические и программные средства, используемые во время испытаний, а также порядок проведения испытаний.

В разделе «Методы испытаний» приведено описание используемых методов испытаний.

В разделе «Тестовые примеры» приведены таблицы с результатами тестовых испытаний.

СОДЕРЖАНИЕ

1. ОБЪЕКТ ИСПЫТАНИЙ

1.1. Наименование объекта

Наименование – «приложение для управления универсальной роботизированной платформой-носителем (на примере ООО «Центр инновационных разработок ВАО)»

1.2. Область применения объекта

В условиях опасных для человека, снизив вероятность получения травм, вредных факторов, способных навредить здоровью на предприятиях (опасность обрушения, радиационная, биологическая или химическая угроза, высокие температуры).

1.3. Обозначение испытуемой программы

Наименование темы разработки «UniversalPlatform».

2. ТРЕБОВАНИЯ К ПРОГРАММЕ

2.1. Требования подлежащие к проверке

«Модуль движения» - включает в себя управление движением платформы, выставление параметров мощности двигателя, выставление режима разгона.

«Модуль телеметрии» - включает в себя получение текущего направления движения, графики значений гироскопа, акселерометра, магнитометра, датчиков тока моторов, вольтажа батареи, получение координат GNSS (Глобальная навигационная система) и команды экстренного останова.

3. РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

Сценарий и результаты тестовых испытаний методом черного ящика приложения для управления универсальной роботизированной платформой-носителем представлены в таблицах 1-2.

В таблице 1 представлен модуль движения.

Таблица 1 - Тестирование модуля движения

№ П/П	Действие (Входное значение)	Ожидаемый результат	Фактический результат	Ожидаемое действие платформы	Фактический результат платформы	Статус теста (пройден / не пройден)
1	Нажатие на кнопку «Вперёд»	Отправка команды движения вперёд на платформу	Отправка команды движения вперёд на платформу	Начало движения платформы вперёд.	Начало движения платформы вперёд	Пройден
2	Нажатие на кнопку «Вперёд и направо»	Отправка команды движения вперёд и направо на платформу	Отправка команды движения вперёд и направо на платформу	Начало движения платформы вперёд и направо.	Начало движения платформы вперёд и направо	Пройден
3	Нажатие на кнопку «направо»	Отправка команды движения направо на платформу	Отправка команды движения направо на платформу	Начало движения платформы направо.	Начало движения платформы направо	Пройден
4	Нажатие на кнопку «направо и назад»	Отправка команды движения направо и назад на платформу	Отправка команды движения направо и назад на платформу	Начало движения платформы направо и назад.	Начало движения платформы направо и назад	Пройден
5	Нажатие на кнопку «назад»	Отправка команды движения назад на	Отправка команды движения назад на платформу	Начало движения платформы назад.	Начало движения платформы назад	Пройден

№ П/П	Действие (Входное значение)	Ожидаемый результат	Фактический результат	Ожидаемое действие платформы	Фактический результат платформы	Статус теста (пройден / не пройден)
		платформу				
6	Нажатие на кнопку «назад и налево»	Отправка команды движения назад и налево на платформу	Отправка команды движения назад и налево на платформу	Начало движения платформы назад и налево.	Начало движения платформы назад и налево.	Пройден
7	Нажатие на кнопку «налево»	Отправка команды движения налево на платформу	Отправка команды движения налево на платформу	Начало движения платформы налево.	Начало движения платформы налево.	Пройден
8	Нажатие на кнопку «Вперёд и налево»	Отправка команды движения вперёд и налево на платформу	Отправка команды движения вперёд и налево на платформу	Начало движения платформы вперёд и налево.	Начало движения платформы вперёд и налево.	Пройден
9	Выставление мощности, к примеру на 25%	Отправка команды на платформу	Отправка команды на платформу	Изменение мощности движения платформы на 25%	Изменение мощности движения платформы на 25%	Пройден
10	Нажатие на кнопку «Быстро»	Отправка команды на платформу	Отправка команды на платформу	Изменение мощности движения платформы на 100%	Изменение мощности движения платформы на 100%	Пройдён
11	Нажатие на кнопку «Медленно»	Отправка команды на платформу	Отправка команды на платформу	Изменение мощности движения платформы на 50%	Изменение мощности движения платформы на 50%	Пройден

В таблице 2 представлен модуль телеметрии.

Таблица 2 - Тестирование модуля телеметрии

№ П/П	Действие (Входное значение)	Ожидаемый результат	Фактический результат	Ожидаемое действие платформы	Фактический результат платформы	Статус теста (пройден / не пройден)
1	Отправка запроса на платформу на получение значений с гироскопа	Отображение полученных данных в графике «гироскоп»	Отображение полученных данных в графике «гироскоп»	Отправка на приложение значений гироскопа	Отправка на приложение значений гироскопа	Пройден
2	Отправка запроса на платформу на получение значений с акселерометра	Отображение полученных данных в графике «акселерометр»	Отображение полученных данных в графике «акселерометр»	Отправка на приложение значений акселерометра	Отправка на приложение значений акселерометра	Пройден
3	Отправка запроса на платформу на получение значений с магнитометра	Отображение полученных данных в графике «магнитометр»	Отображение полученных данных в графике «магнитометр»	Отправка на приложение значений магнитометра	Отправка на приложение значений магнитометра	Пройден
4	Отправка запроса на платформу на получение значений с датчиков тока моторов	Отображение полученных данных в графике «ток»	Отображение полученных данных в графике «ток»	Отправка на приложение значений ток	Отправка на приложение значений ток	Пройден
5	Отправка запроса на платформу на получение	Отображение полученных данных в	Отображение полученных данных в	Отправка на приложение значений	Отправка на приложение значений вольтажа батареи	Пройден

№ П/П	Действие (Входное значение)	Ожидаемый результат	Фактический результат	Ожидаемое действие платформы	Фактический результат платформы	Статус теста (пройден / не пройден)
	значений с вольтажа батареи	графике «Батарея»	графике «Батарея»	вольтажа батареи		
6	Отправка запроса на платформу на получение значений координат текущего местоположения	Отображение полученных данных на карте	Отсутствие отображения полученных данных на карте	Отправка на приложение значений координат текущего местоположения	Отправка на приложение пустой строки	Не пройден
7	Нажатие на кнопку «Em» (экстренный останов)	Отправка команды экстренного останова на платформу	Команда экстренного останова на платформу не отправлена	Экстренное прекращение работы	Продолжение работы	Не пройден

Сценарий и результаты тестовых испытаний методом «Входной двери» (Бинарное тестирование yes/no) приложения для управления универсальной роботизированным комплексом представлены в таблице 3.

Таблица 3 - Тестирование методом «Входной двери».

№ П/П	Действие (Входное значение)	Ожидаемый результат	Фактический результат	Статус теста (пройден / не пройден)
1	Возможность включения автопилота по точкам	Да	Нет	Не пройден
2	Возможность включения следования по азимуту	Да	Нет	Не пройден
3	Возможность менять тип разгона платформы	Да	Да	Пройден

№ П/П	Действие (Входное значение)	Ожидаемый результат	Фактический результат	Статус теста (пройден / не пройден)
4	Возможность использования интерфейса MasterLink в простом режиме	Да	Да	Пройден
5	Возможность использования интерфейса MasterLink в режиме ведомого	Да	Да	Пройден
6	Возможность экстренного отключения платформы физической кнопкой	Да	Нет	Не пройден
7	Возможность работы по управляющей программе	Да	Да	Пройден

Сценарий и результаты тестовых испытаний методом UI-тестов приложения для управления универсальной роботизированной платформой-носителем представлены в таблице 4.

Таблица 4 - Тестирование методом UI-тестов.

Контроль интерфейса пользователя	Подтверждение	Статус теста (пройден / не пройден)
Использовать режим благоразумно; (возможность его редактировать, но в пределах разумного)	Отсутствие возможности использоваться режим благоразумно	Не пройден
Предоставить пользователю возможность выбирать: работать либо мышью, либо клавиатурой, либо их комбинацией;	Движение платформы приводится стрелками или клавишами W, A, S, D	Пройден
Позволить пользователю сфокусировать внимание;	Удобно расположена главная информация на интерфейсе	Пройден
Демонстрировать сообщения, которые помогут ему в работе;	Всплывающее окно с сообщением при работе	Пройден
Создавать условия для немедленных и обратимых действий, а также обратной связи;	Кнопка экстренного останова	Пройден
Обеспечить соответствующие пути и выходы;	Доступ ко вкладкам с любых окон	Пройден
Приспосабливайте систему к пользователям с различным уровнем подготовки;	Отсутствие вспомогательных объектов внутри приложения	Не пройден
Сделать пользовательский интерфейс более понятным;	Имеется надпись у кнопок и цветные элементы	Пройден

Контроль интерфейса пользователя	Подтверждение	Статус теста (пройден / не пройден)
Дать пользователю возможность настраивать интерфейс п своему вкусу;	Отсутствие возможности настраивать интерфейс	Не пройден
Разрешить пользователю напрямую манипулировать объектами интерфейса.	Наличие ползунка настройки мощности	Пройден
Уменьшение загрузки памяти пользователя		
Не загружать кратковременную память;	Более-менее используются стандартные элементы.	Пройден
Полагаться на распознание, а не на повторение; (распознавать элементы визуально)	Есть изображения на кнопках.	Пройден
Представить визуальные заставки;	Отсутствует визуальная заставка	Не пройден
Использовать метафоры из реального мира;	Кнопка стоп в виде знака «STOP»	Пройден
Применять раскрытие и объяснение понятий и действий;	Понятен смысл кнопки	Пройден
Увеличить визуальную ясность.	Внимание ясно выражается на всю необходимую информацию, сверено с золотым сечением.	Пройден
Последовательность пользовательского интерфейса	Придерживается единая цветовая палитра, шрифт и размер текста	Пройден
Общая совместимость всех программ;	Похожесть с аналогичными программными продуктами видна.	Продйен
Сохранение результатов взаимодействия;	Отсутствие результатов взаимодействия	Не пройден
Эстетическая привлекательность и цельность;	Простой и минималистичный дизайн	Пройден

Сценарий и результаты тестовых испытаний методом нагрузочного тестирования приложения для управления универсальной роботизированной платформой-носителем представлены в таблице 5.

Таблица 5 - Тестирование методом нагрузочного тестирования.

№ П/П	Действие (Входное значение)	Ожидаемый результат	Фактический результат	Ожидаемое действие платформы	Фактический результат платформы	Статус теста (пройден / не пройден)
1	Отправка запроса с интервалом в 100мс	Успешная отправка	Успешная отправка	Получение и обработка запроса	Получение и обработка запроса	Пройден
2	Отправка запроса с интервалом в 600мс	Успешная отправка	Успешная отправка	Получение и обработка запроса	Получение и обработка запроса	Пройден
3	Отправка запроса с интервалом в 10мс	Успешная отправка	Успешная отправка	Получение и обработка запроса	Отсутствие обработки запроса	Не пройден
4	Отсутствие отправки запроса	Запрос не отправлен	Запрос не отправлен	Продолжение текущего действия	Продолжение текущего действия	Пройден
5	Отправка запроса с интервалом в 80мс	Успешная отправка	Успешная отправка	Получение и обработка запроса	Получение и обработка запроса	Пройден
6	Отправка запроса с интервалом в 50мс	Успешная отправка	Успешная отправка	Получение и обработка запроса	Отсутствие обработки запроса	Не пройден
7	Отправка запроса с интервалом в 1мс	Успешная отправка	Успешная отправка	Получение и обработка запроса	Отсутствие обработки запроса	Не пройден
8	Отправка запроса с интервалом в 300мкс	Успешная отправка	Успешная отправка	Получение и обработка запроса	Отсутствие обработки запроса	Не пройден

Сценарий и результаты тестовых испытаний методом стресс тестирования приложения для управления универсальной роботизированной платформой-носителем представлены в таблице 6.

Таблица 6 - Тестирование методом стресс тестирования.

№ П/П	Действие (Входное значение)	Ожидаемый результат	Фактический результат	Ожидаемое действие платформы	Фактический результат платформы	Статус теста (пройден / не пройден)
1	Отправка запроса на платформу с двух ПК	Успешная отправка запроса	Успешная отправка запроса	Выполнение действий по очереди	Выполнение действий по очереди	Пройден
2	Отправка запроса с двух платформ на ПК	Успешная обработки запроса	Вызвано исключение переполнение стека	Успешная отправка запроса	Успешная отправка запроса	Не пройден
3	Внезапное отключение питания платформы	Окно с сообщением «Отсутствие связи проверьте платформы»	Окно с сообщением «Отсутствие связи проверьте платформы»	Платформ выключена	Платформ выключена	Пройден
4	Выключение компьютера	Отправка «Экстренного режима»	Режим «Экстренный режим» не отправлен	Ожидается переход в экстренный режим	Продолжение предыдущего действия	Не пройден
5	Линейное ускорение по любой оси более 8 м/с	Отображение сообщения «Опасно! Превышение допустимого ускорения.»	Отсутствие сообщения	Попытка само стабилизации	Отсутствие само стабилизации	Не пройден
6	Угол наклона по любой из осей более чем 75°	Отображение сообщения «Вниман	Отсутствие сообщения	Блокировка двигателей	Отсутствие блокировки двигателей	Не пройдён

№ П/П	Действие (Входное значение)	Ожидаемый результат	Фактический результат	Ожидаемое действие платформы	Фактический результат платформы	Статус теста (пройден / не пройден)
		ие, опасность переворота платформы»				
7	Блокировка двигателей	Отображение сообщения «Внимание! Проверьте двигатели.»	Отсутствие сообщения	Блокировка двигателей	Отсутствие блокировки двигателей	Не пройден

Приложение В. Руководство пользователя

АННОТАЦИЯ

В данном программном документе приведено руководство пользователя приложения с предиктивной коррекцией ошибок управления (на примере ООО «Центр инновационных разработок ВАО»).

В разделе «Назначение программы» указаны краткие сведения о программе.

В разделе «Условия выполнения программы» указаны требования к оборудованию и программному обеспечению для функционирования программы.

В разделе «Выполнение программы» указана инструкция для работы с программой.

СОДЕРЖАНИЕ

1. НАЗНАЧЕНИЕ ПРОГРАММЫ

Функциональным назначением программы является приложение для управления платформой-носителем, при помощи которого можно упростить работу в условиях опасных для человека, снизить вероятность получения травм, вредных факторов способных навредить здоровью на предприятиях (опасность обрушения, радиационная, биологическая или химическая угроза, высокие температуры) с возможностью быстрой замены полезной нагрузки.

2. УСЛОВИЯ ВЫПОЛНЕНИЯ ПРОГРАММЫ

В таблице 1 представлены минимальные технические средства для использования программы.

Таблица 1 – Технические средства.

№	Тип оборудования	Название средства
1	2	3
1	Размер экрана	10”
2	Разрешение экрана	1360x720
3	Линейка процессора	Intel Core i3
4	Количество ядер процессора	1
5	Оперативная память	6 ГБ
6	Тип видеокарты	Любой
7	Видеокарта	Любая
8	Конфигурация накопителей	Любая
9	Общий объём всех накопителей	Минимум 128 ГБ
10	Операционная система	Windows 10
11	Клавиатура	Есть
12	Компьютерная мышь	Есть

В таблице 2 представлены программные средства для использования программы.

Таблица 2 – Программные средства.

№	Тип средства	Название средства	Назначение
1	2	3	4
1	Операционная система	Microsoft Windows 10	Организация взаимодействия программ и пользователя

3. ВЫПОЛНЕНИЕ ПРОГРАММЫ

3.1. Действия для установки программы

Загрузите приложение и SDK платформы с официального репозитория проекта.

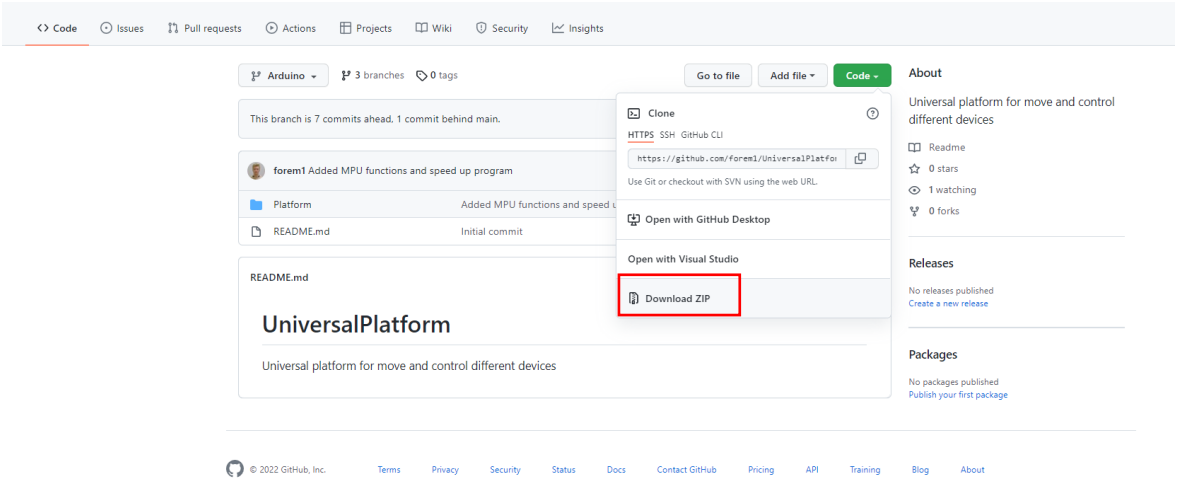

Рисунок 1 - Скачивание решения

Скопируйте скачанный SDK в папку библиотек вашего AVR-GCC компилятора.

```
#include "Platform.h"
```

Рисунок 2 - Подключение библиотеки к программе

Откройте файл "Main.c" из папки "Examples" и скомпилируйте его. Данный файл является базовым и может быть дополнен вашей управляющей программой. Документация по SDK находится в папке "Docs". Скопируйте полученный при компиляции бинарный файл "PlatformControl.hex" в корень карты памяти программатора.

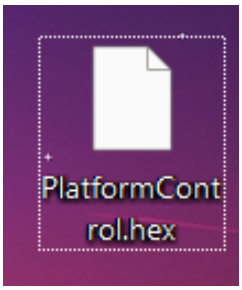

Рисунок 3 - Файл прошивки

Вставьте карту памяти в программатор и включите его. Подключите программатор к платформе-носителю через интерфейс MasterLink.

При помощи кнопок "вверх" и "вниз" на программаторе выберите необходимую версию прошивки для загрузки и нажмите на кнопку "OK". Программатор начнет прошивку платформы и в зависимости от результата на дисплее программатора отобразится надпись "OK" или "ERROR". Не отключайте программатор до завершения прошивки платформы.

Перезагрузите платформу, переподключив питание. Платформа готова к подключению к управляющему персональному компьютеру.

3.2. Действия для работы с программой

Для установки приложения, необходимо перейти в папку с инсталляционным проектом и запустить приложение «PlatformDesktop.exe»

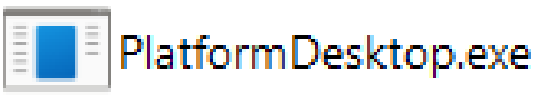

Рисунок 4 – Инсталляционный файл

3.3. Для ознакомления и изучения функциональных особенностей приложения, ниже приведена инструкция (см. Рис 2-4).

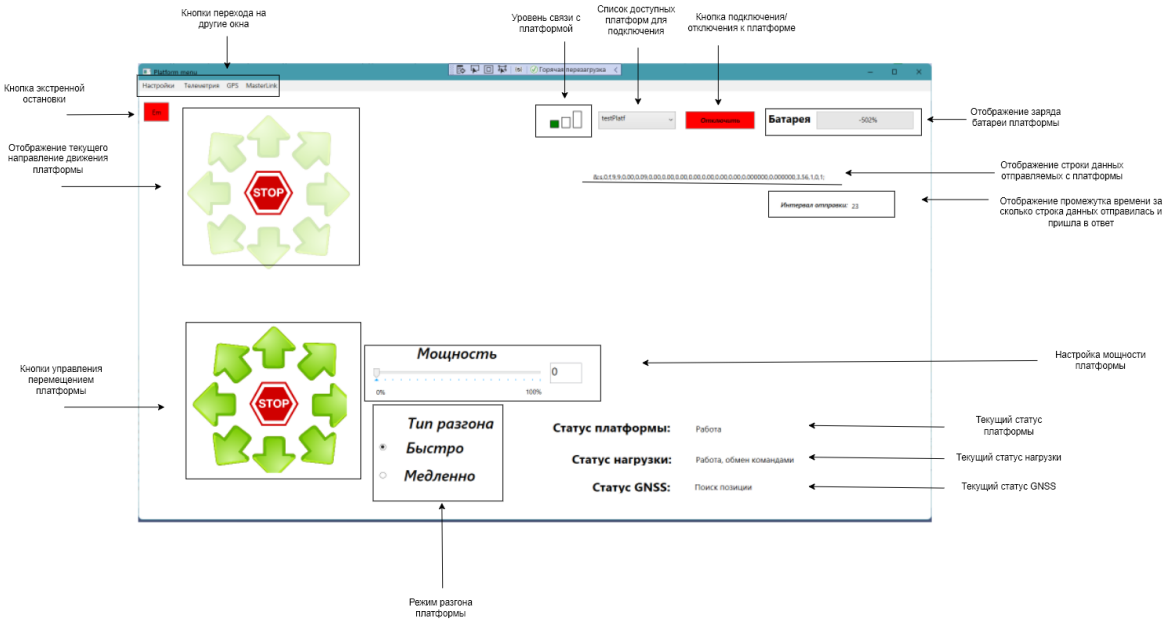

Рисунок 5 – Инструкция главного окна

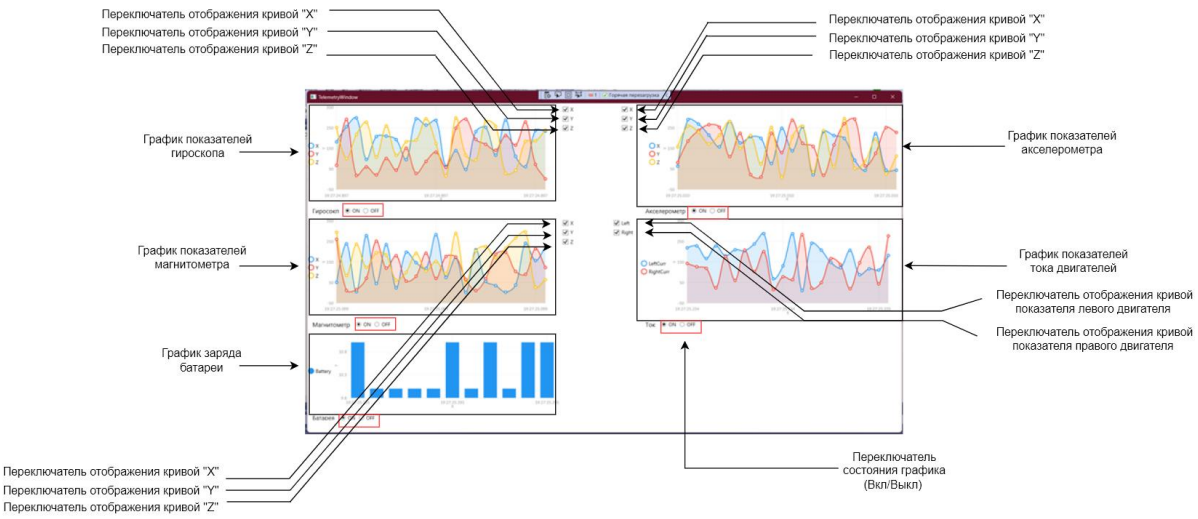

Рисунок 6 - Инструкция окна с графиками с телеметрии

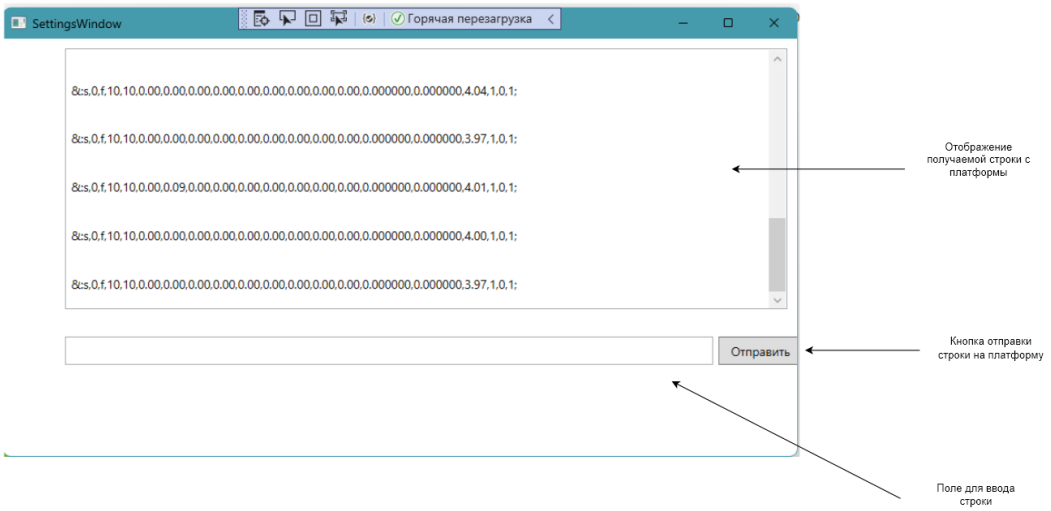

Рисунок 7 - Инструкция окна с настройками

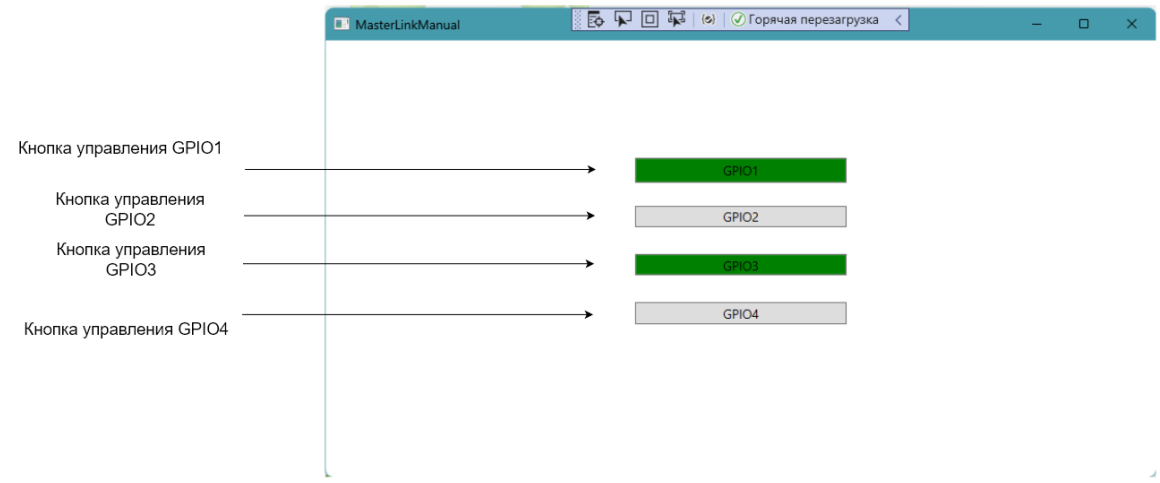

Рисунок 8 – Инструкция окна с ручным управлением порта MasterLink

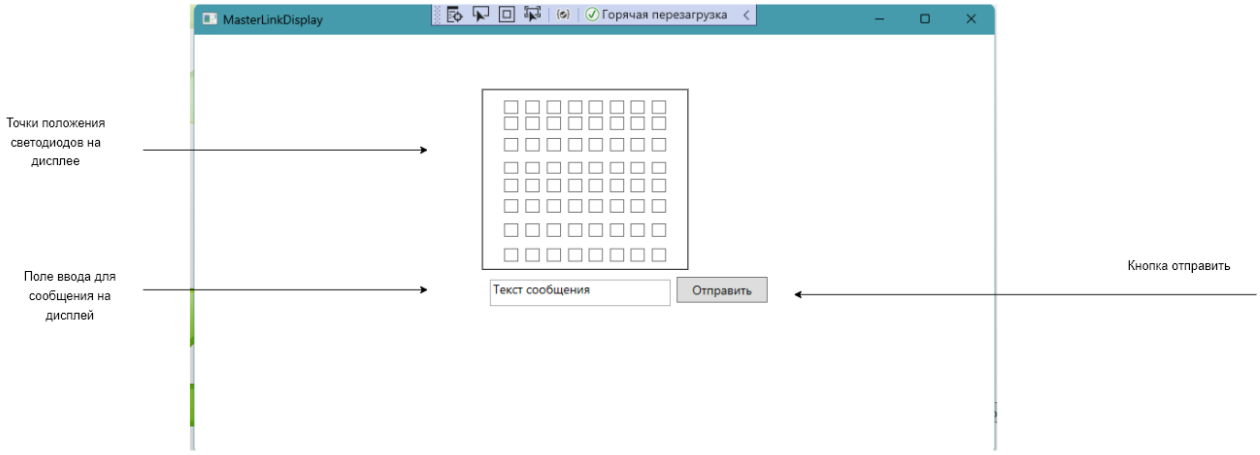

Рисунок 9 - Инструкция окна с нагрузкой дисплей

3.4. Действия для подключения приложения к платформе

После установки приложения и действий для установки работы с платформой, необходимо запустить программу, после чего программа начнет «стучать» во все COM-порты и ждать ответа. Если в ответ приходит строка, то программа инициализирует платформу по ID, далее отображает имя платформы в комбо-боксе, после чего нужно нажать на кнопку подключение.

Рисунок 10 - Подключение к платформе

3.5. Подробное описание строения графика

Шкала оси Y (вертикальная) означает значение на примере гироскопа, значение в градусах, ось X означает текущее время. При переключении отображения кривой X/Y/Z, на графике перестанет/начнет отображаться линия X/Y/Z. Линия имеет определённый цвет, который указан слева от оси Y. Кнопки снизу графика отключает/включает работу данного графика.

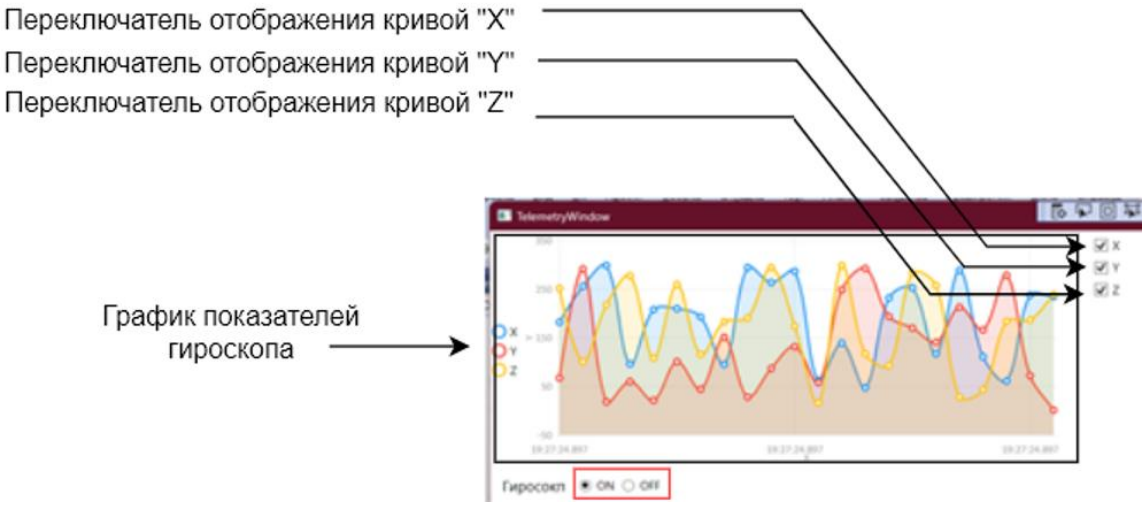

Рисунок 11 - Описание графика

3.6. Действия для удаления программы

Для удаления программы, необходимо удалить приложение и все скаченные библиотеки.

3.7. Пример кода программы для доработки разработчиком.

```
#include "Platform.h"

Platform platform;

void setup() {
  Serial1.begin(9600); //GPS
  platform.begin("Example", "12345678");
  platform.initControlUARTData(platform, 34); //57600 bod
  platform.initMPU();
}

void loop() {
  while (1) {
    platform.getMPUData();
    platform.getGPSData(&Serial1);

    //Put your code here
  }
}
```

Рисунок 12 - Пример кода для доработки