

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Институт машиностроения

(наименование института полностью)

Кафедра «Промышленная электроника»

(наименование)

11.03.04 Электроника и наноэлектроника

(код и наименование направления подготовки/ специальности)

Электроника и робототехника

(направленность (профиль) / специализация)

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
(БАКАЛАВРСКАЯ РАБОТА)**

на тему Индикационная панель для электрического гоночного болида

Обучающийся

Н.Д. Коротаяев

(Инициалы Фамилия)

(личная подпись)

Руководитель

к.т.н., доцент, М.В. Позднов

(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант

к.ф.н., доцент, М.М. Бажутина

(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2022

Аннотация

Изложенная работа описывает разработку и создание электронной индикационной панели приборов, собирающей и отображающей информацию о состоянии различных систем транспортного средства в реальный момент времени

Целью работы является создание прототипа универсального устройства, подготовленного для интеграции в электрический гоночный болид Formula Student

Задачами работы являются анализ существующих решений, изучение и подбор компонентной базы, разработка печатной платы и корпуса устройства, написание кода программы и сборка готового устройства.

Abstract

The title of the bachelor's thesis is "Race car display panel".

The work consists of a 77-page explanatory note including 58 figures, 11 tables, a list of 21 references including 17 foreign sources, 5 appendix and the graphic part on 6 A1 sheets.

The aim of the work is to design a display panel which will be used as a dashboard and telemetry unit in a Formula Student Electric car

In the first part, we examine existing technical solutions and draw up criteria for device design.

In the second part, we directly designing the device: its structure, electronic part, power circuit and casing.

In the third part, we develop an algorithm, create a program flowchart and write the code for it.

In the fourth part, we design the PCB for this device.

In the next part, we conduct an experimental check of the performance of all components of the device, aimed at identifying errors.

It can be concluded that all the tasks being accomplished, the presented work can be used for its main task: be installed in a Togliatti Racing Team race car during racing events

Содержание

Введение.....	5
1 Постановка задачи.....	7
1.1 Поиск и анализ аналогичных устройств.....	7
1.2 Определение критериев для разработки.....	9
2 Проектирование устройства.....	11
2.1 Подбор компонентов	11
2.2 Разработка схемы питания	26
2.3 Разработка структурной схемы	27
2.4 Разработка печатной платы.....	28
3 Создание печатной платы.....	44
4 Разработка алгоритма и управляющей программы.....	46
5 Разработка корпуса устройства	54
Заключение	58
Список используемой литературы	59
Приложение А – Схема соединения в системе EasyEDA.....	62
Приложение Б – Листинг управляющей программы	63

Введение

Электромобили – набирающий популярность тип транспорта. Несмотря на окружающую их ауру высоких технологий и светлого будущего, они не являются технологиями завтрашнего дня. Тихие и чистые самодвижущиеся повозки, приводимые в движение силой электричества не просто давно известны, они были популярны и составляли конкуренцию автомобилям с двигателем внутреннего сгорания еще на заре автомобилестроения. Так, например, в первом десятилетии 20-го века в США автомобили на электрической тяге составляли 38% от общего парка, лишь незначительно уступая автомобилям на паровой тяге (40%) и серьезно опережая автомобили с ДВС, которые занимали лишь 28% [2].

К сожалению, электроавтомобили были вытеснены и забыты на почти что столетие, не смотря на редкие попытки создать «транспорт будущего» вновь, реальную популярность и внимание они получили совсем недавно, ответственным за это можно считать компанию Tesla, выпустившую в 2015 году Model S, которая качественно превосходила все предыдущие автомобили и могла полноценно конкурировать со ставшим уже классическим бензиновыми и дизельными машинами. Открывшийся заново рынок подстегнул интерес как различных правительств, увидевших в электротранспорте решение проблемы загрязнения воздуха, так и производителей, желающих занять пустующую нишу, потребитель же увидел что-то новое, необычное, ведь электротранспорт дает ощущения, совершенно отличные от привычных машин.

Спорт не остался в стороне, так в 2014 году дебютировали гонки серии Formula-E, аналог популярной во всем мире Формулы-1[1]. Студенческие соревнования Formula SAE, ввиду своей специфики подготовки студентов и меньшего стеснения бюрократическими рамками, запустили полностью электрическую дисциплину раньше – в 2010 году [3].

Современные требования требуют современных решений, поэтому для того, чтобы оставаться актуальной, команде нашего университета, Togliatti Racing Team (TRT), потребовалось создание собственного гоночного автомобиля на электрической тяге. Одной из проблем, требовавшей решения, оказалась приборная панель. Использование проверенного временем решения: приборной панели мотоцикла, взятой вместе с двигателем с работающей модели, оказалось невозможным ввиду их отсутствия – силовая установка создается с нуля, а значит требуется новое отдельное устройство.

В данной работе было разработано устройство, отвечающее регламенту Formula Student Electric, а также требованиям команды TRT.

1 Постановка задачи

1.1 Поиск и анализ аналогичных устройств

Ввиду того, что приборная панель согласно регламенту соревнований является «low voltage system» (системой низкого напряжения), специальных требований к ней не предъявляется. Однако непосредственно проводящие проверку непосредственно перед соревнованиями члены технической комиссии в личном общении рекомендовали обеспечить защиту от попадания влаги. Главным требованием команды TRT был бюджет, не превышающий 500\$.

Перед принятием решения о разработки были рассмотрены варианты закупки уже готового прибора.

Неожиданной проблемой стало малое количество универсальных решений, обусловленное спецификой применения. Первым подходящим вариантом стала приборная панель для электрических мотоциклов от Unizen Technologies Pvt Ltd [4].



Рисунок 1 – Unizen EV Instrument Cluster

Однако его применение не является целесообразным ввиду отсутствия возможности конфигурировать выводимую информацию. Похожая проблема свойственна всем бюджетным готовым решениям: для удешевления и

упрощения производства они узкоспециализированы, предназначены для установки на конкретный тип транспорта с узким спектром возможных регулировок.

Очевидным решением в данной ситуации стал поиск универсального устройства профессионального класса, обладающего широким диапазоном настроек. Хорошим примером такого устройства является Andromeda EVIC, 7" дисплей, предназначенный для установки на любой электрический автомобиль. [7]



Рисунок 2 – Andromeda EVIC

Универсальные порты соединений, поддержка соединения по CAN-шине, наличие всей необходимой информации о коннекторах и схемы подключений как самого устройства, так и различных датчиков на сайте производителя, а также возможность полной настройки выводимой информации под конкретные требования делают эту панель прекрасным выбором, отвечающим всем требованиям.

К сожалению, такой широкий функционал и удобство пользования отражаются на цене – 825\$ без учета доставки, эта сумма серьезно превышает допустимый бюджет, а следовательно, несмотря на всю привлекательность, данная панель не может быть использована.

Аналогов этого дисплея, ценой дешевле 500\$ найдено не было.

Ввиду отсутствия возможных вариантов, отвечающих как техническим требованиям, так и требованиям бюджета, было принято решение разработать прибор самостоятельно.

1.2 Определение критериев для разработки

Определение критериев разработки началось с определения способа вывода информации. Без промедлений выбор пал на жидкокристаллический дисплей, ввиду отсутствия каких-либо альтернатив.

Следующим шагом стало определение информации, которую необходимо выводить на приборную панель. Консультируясь с членами команды TRT, а также отталкиваясь от готовых решений, было решено совместить в одном устройстве приборную панель и устройство сбора телеметрии. Это обуславливается необходимостью сбора телеметрии гоночного болида, а совмещение в одном устройстве как функций приборной панели, так и бортового самописца, позволяет в целом упростить электронную систему болида, ведь приборная панель и так собирает практически всю информацию для вывода ее пилоту. Так же ввиду близкого расположения, устройство должно отвечать за обработку кнопок включения низко и высоковольтных систем, датчика установки степени рекуперации и кнопок для связи с пилотом.

По итогам проделанной работы был сформирован следующий список требований:

- Установка прибора за рулем на месте приборной панели
- Подключение по CAN-шине
- Вывод информации на дисплей
- Сбор данных телеметрии

Вывод информации на дисплей подразумевает отображение следующих данных:

- Степень заряда батареи
- Температура батареи
- Температура левого инвертора
- Температура правого инвертора
- Температура левого мотора
- Температура правого контроллера
- Выбранный режим езды
- Состояние высоковольтной системы
- Состояние GPS-сигнала
- Состояние LORA-передатчика

Сбор телеметрии подразумевает запись, сохранение и отправку всех вышеперечисленных параметров, а также:

- Уровень нажатия педали акселератора
- Степень рекуперации
- Нажатие кнопки связи водителя
- Данные GPS-приемника
- Данные с широкополосного датчика, совмещающего в себе акселерометр, гироскоп, термометр и др.

Вывод

Произведя анализ рынка и проконсультировавшись с членами команды TRT, был сформирован список задач, необходимых для реализации, после чего была начата работа по проектированию устройства.

2 Проектирование устройства

2.1 Подбор компонентов

Проектирование будущего прибора было начато с подбора компонентов, необходимых для решения поставленных задач.

Так как устройство должно было обрабатывать множество разных сигналов, а также выводить информацию на ЖК дисплей, основной создаваемого прибора должен был стать микроконтроллер. Из множества доступных вариантов, для дальнейшего рассмотрения были отобраны следующие решения: Arduino Mega, Raspberry Pi, ESP32 и Teensy.

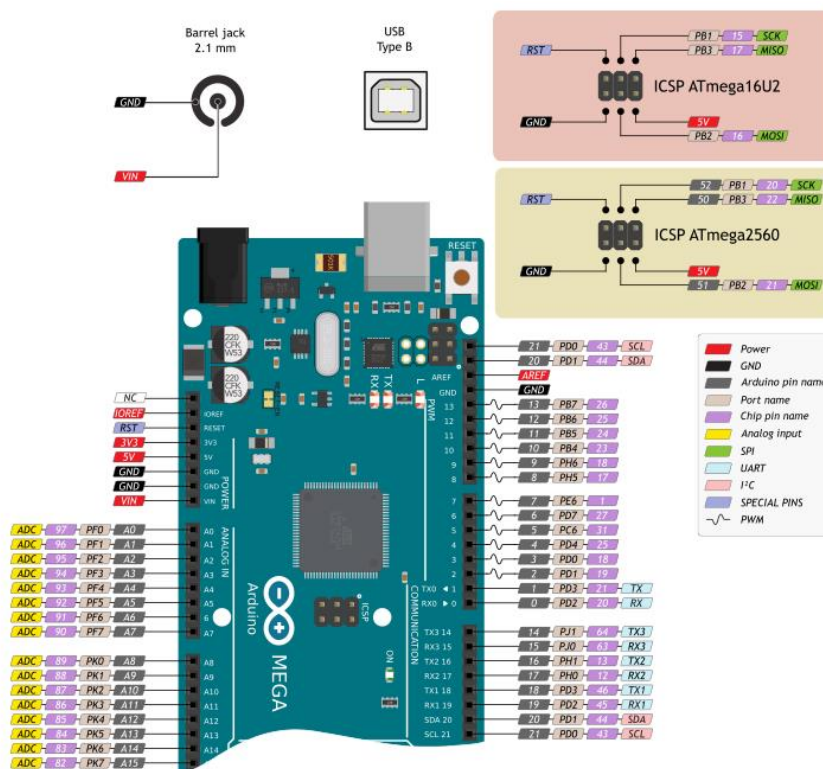


Рисунок 3 – Arduino Mega

Первым рассмотренным вариантом сразу же стал микроконтроллер Arduino Mega [8]. Проходя обучение в университете, данный вид микроконтроллеров был хорошо изучен, с чего помощью была сделана не одна лабораторная работа. Еще одним преимуществом является его распространенность, наличие широкой поддержки в интернете и сравнительно небольшой бюджет, всего 5-6 тысяч рублей.

Основные характеристики представлены в таблице 1.

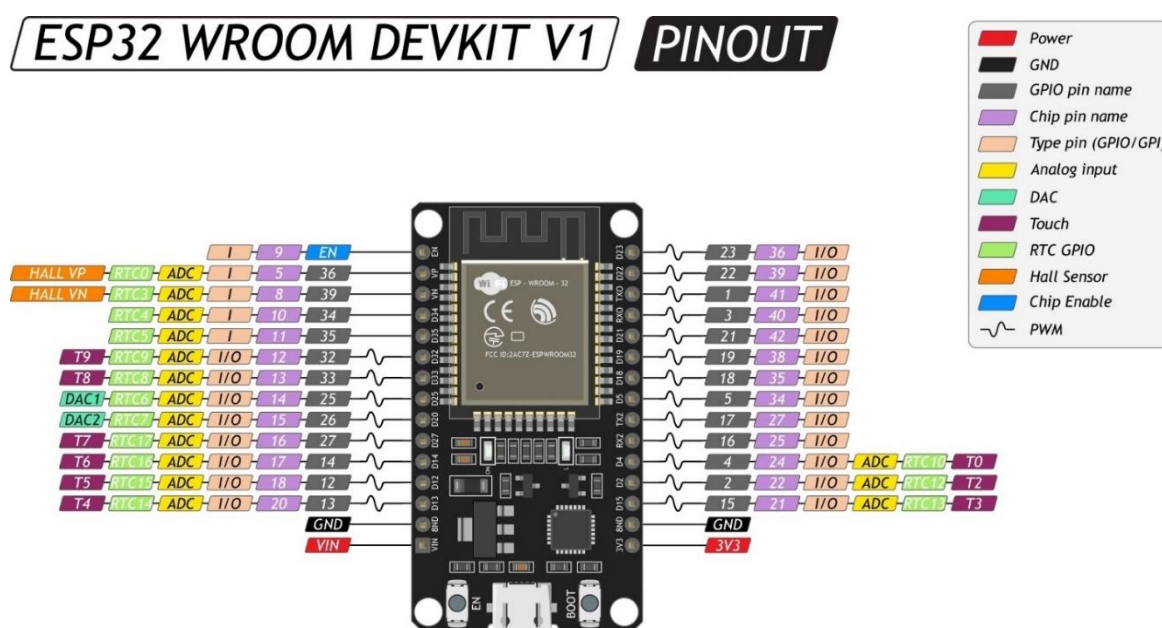
Таблица 1 – основные характеристики Arduino Mega Rev.3

Параметр	Значение
Рабочее напряжение	5 V
Входное напряжение	7-12 V
Рекомендованное входное напряжение	6-20 V
Цифровые порты входа/выхода (с поддержкой ШИМ)	54 (15)
Аналоговые порты входа/выхода	16
Максимальный выходной ток порта	20 мА
Максимальный ток порта 3.3V	50 мА
FLASH-память	256 КБ
SRAM	8 Кб
EPROM	4 Кб
Частота процессора	16 МГц
Длина	101.52 мм
Ширина	53.3 мм
Вес	37 г

Arduino Mega обладает широким набором портов, однако данный микроконтроллер не имеет в своем составе контроллера CAN и для его реализации требуется отдельная плата. Помимо этого опасения вызывала относительно слабая производительность и малая память микропроцессора ATmega 2560.

Система должна обеспечивать стабильное быстроедействие, если задержки и ошибки в сборе телеметрии или отображении информации на дисплее являются хоть и серьезной, но не критичной проблемой, то передача этих данных и обработка сигнала степени рекуперации являются критически-важными элементами, от которых зависит не только результат болида на треке, но и безопасность пилота.

Общий размер программы и затраты оперативной памяти так же представлялись достаточно объемными. В связи с этим было решено, что тактовая частота в 16 МГц и всего 256 Кб FLASH и 8 Кб SRAM памяти не обеспечат достаточной производительности, тем более в случае недостаточной оптимизации кода программы управления. Так же этого явно было недостаточно для установки RTOS – операционной системы реального времени, применение которой предполагалось на данном этапе.



беспроводные стандарты связи, такие как Bluetooth и Wi-Fi, однако для подключения CAN-шины все так же требуется отдельный контроллер.

Таблица 2 – основные характеристики ESP32-WROOM

Параметр	Значение
Микропроцессор	Tensilica Xtensa LX6
Тактовая частота	240 МГц
Рабочее напряжение	3.3V
АЦП	12-битный, 18 каналов
ЦАП	8-битный, 2 канала
Цифровые порты входа/выхода	25 (4 только на вход)
Аналоговые входы с АЦП	15
Аналоговые выходы с ЦАП	2
Максимальный ток цифрового порта	40 мА
Максимальный ток порта 3.3V	50 мА
SRAM	520 Кб
FLASH	448 Кб
Внешняя FLASH	4 Мб
Стандарты передачи данных	SPI(4), I2C(2), I2S(2), CAN, UART(3)
Wi-Fi	Tensilica Xtensa LX6
Bluetooth	240 МГц

Благодаря использованию двухъядерного микропроцессора Xtensa LX6 от компании Tensilica, в два раза большим объемом оперативной памяти и возможностью использования внешней FLASH-памяти данный микроконтроллер куда больше подходит для выполнения задачи, однако в отличие от Arduino, где используется AVR-процессор, LX6 построены на проприетарная архитектура FLIX, которая является совершенно не знакомой.

Возможность использовать ESP32-C или ESP32-H, построенные на архитектуре RISC-V хоть и облегчает доступ к поддержке сообщества, однако не решает проблемы.

Следующим рассмотренным вариантом стала Raspberry Pi, являющаяся по сути полноценным мини-компьютером под управлением полноценной операционной системы Linux [23].

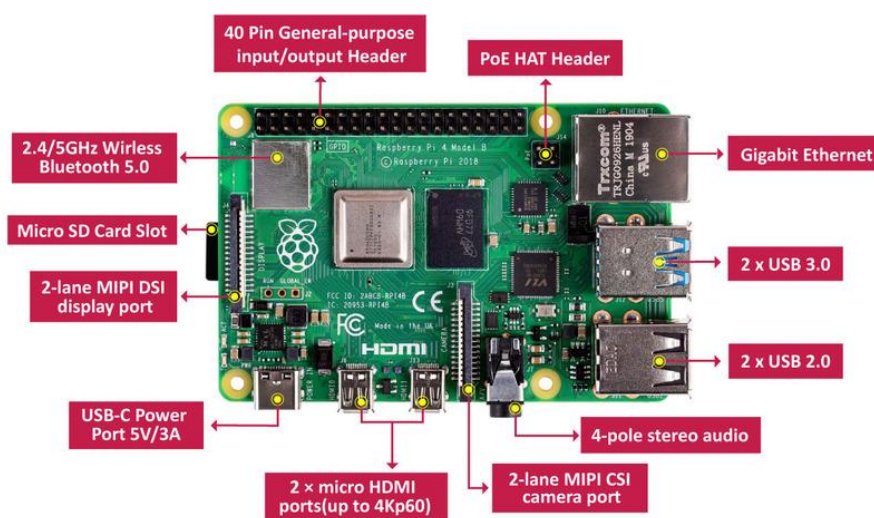


Рисунок 5 - Raspberry Pi

Таблица 3 – основные характеристики Raspberry Pi

Параметр	Значение
Микропроцессор	Cortex-A72 64-bit
Тактовая частота	1,5 ГГц
ОЗУ	4 GB LPDDR4-2400
Порты входа/выхода	40
Wi-Fi	5.0 ГГц IEEE 802.11b/g/n/ac
HDMI	2 micro HDMI 4k 60 ГГц
Интернет	Ethernet 1 Гбит
Требуемое питание	5.1V 3A через USB-C или один из портов.

Так как данный микроконтроллер является по сути полноценным компьютером, сомнений в его производительности не возникало. Однако в этом же заключалась и проблема: для работы требуется полноценная операционная система на базе Linux, что не является приемлемым вариантом из-за вероятности полного зависания системы.

Не найдя подходящего решения, поиски продолжились пока не был найден еще один вариант, которым стала Teensy 4.0 [25].

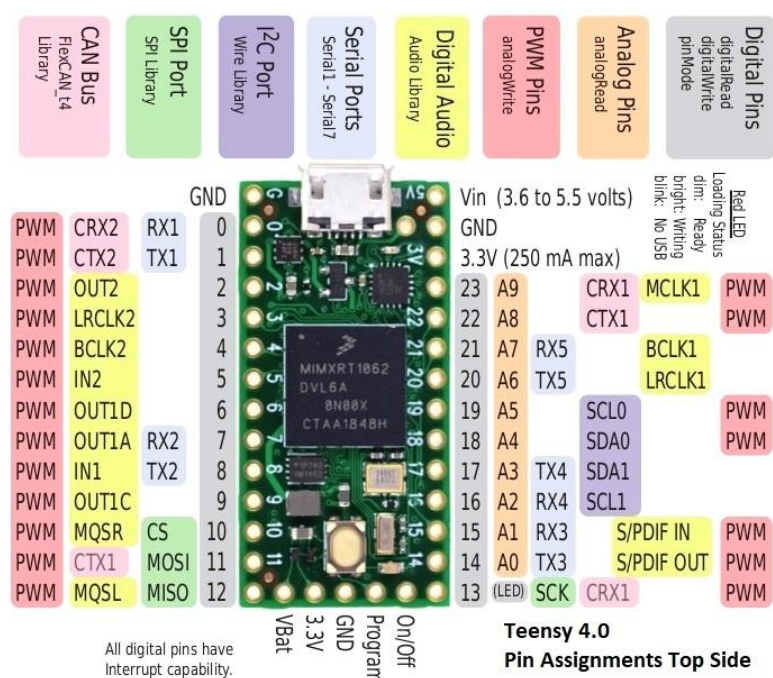


Рисунок 6 – Teensy 4.0

Данный микроконтроллер является ардуино-подобным, что позволило применить уже имеющиеся знания, а также получить поддержку большого сообщества. Помимо этого, он обладает мощным процессором, большим объемом памяти, большим числом портов, встроенным CAN-контроллером, Serial и I²C портами.

Таблица 4 – основные характеристики Teensy 4.0

Параметр	Значение
Процессор	ARM Cortex-M7
Тактовая частота	600 MHz
FLASH	1986 Кб
ОЗУ	1024 Кб
Цифровые порты входа/выхода (с ШИМ)	40 (31)
Аналоговые порты входа/выхода	10
Аналоговые порты входа	14
Порты связи	Serial (7), SPI (3), I2C (3)
CAN-шины	3 (1 CAN FD)
Каналы прямого доступа к памяти	32
Особенности	криптографическая акселерация, наличие генератора случайных чисел, часов реального времени, возможность разгона микропроцессора до 1.2 ГГц

Обладая Cortex-процессором, работающим на частоте 600 МГц при возможном разгоне до 1.2, внушительным объемом памяти, при этом оставаясь ардуино-совместимым микроконтроллером, данный вариант оказался идеально подходящим для поставленной задачи. Компактные размеры, наличие встроенного CAN-контроллера, возможность подключения FLASH-накопителей оказались приятным бонусом.

Определившись с самым главным – микроконтроллером, следующим шагом стал подбор дисплея. Для адекватного размещения всех параметров,

необходима была полноценная матрица достойного разрешения, так необходима была относительно большая диагональ – в районе 5 дюймов.

Все это сильно сузило круг подходящих вариантов: Простые строчные дисплеи не обладали достаточной гибкостью настройки, а OLED-панели не подходили по размерам.

Наличие библиотек для управления дисплеем было принято как само собой разумеющееся.

Логичным шагом стал поиск вариантов среди китайских производителей на сайте Aliexpress, где после непродолжительных поисков был найден подходящий вариант: JLX256160 [18].



Рисунок 7 – Дисплей JLX256160

Таблица 5 – основные характеристики JLX256160

Параметр	Значение
Диагональ	5 дюймов
Разрешение	256 × 160
Тип матрицы	TFT
Количество отображаемых цветов	Монохромный
Количество градаций оттенка	16
Необходимое питание	3.3 или 5V
Тип подключения	SPI или I ² C или Параллельный
Длина	86 мм
Ширина	116 мм

В нашем случае была выбрана конфигурация с подключением по I²C и с питанием от 5 Вольт. Данный дисплей показывает хорошую видимость даже при на улице при солнечном свете. Не смотря на монохромность, наличие множества оттенков позволяет обеспечить четкую видимость отображаемой информации. Большая диагональ и сравнительно большое разрешение позволяет удобно разместить всю необходимую информацию, в том числе динамически изменяемую.

Так как выбранный микроконтроллер уже обладает встроенным CAN-контроллером, для подключения к CAN-шине необходим лишь CAN-трансивер.

После недолгого поиска, был выбран трансивер на основе чипа TJA1051 производства NXP Semiconductors [16]. Это хорошее, проверенное решение, прекрасно зарекомендовавшее себя в автомобильной промышленности.



Рисунок 8 – Трансивер на основе TJA1051

Таблица 6 – основные характеристики TJA1051

Параметр	Значение
Основные	<p>Соответствие стандарту ISO 11898-2:2003</p> <p>Гарантированная синхронизация передачи данных на скоростях до 5 МБит/с</p> <p>Низкое ЭМ излучение и высокая ЭМ стойкость</p> <p>Возможность прямого подключения к питанию 3 V и 5 V микроконтроллера</p> <p>Соответствие стандарту AEC-Q100</p>
Режим низкого энергопотребления	<p>Функциональное поведение предсказуемо при любых условиях питания</p> <p>Отключение от CAN-шины при нулевой нагрузке</p>

Продолжение таблицы 6

Защита	Высокая стойкость к электростатическим разрядам Защита выводов шины от скачков тока и напряжения Защита от перегрева и переохлаждения Защита от блокировки передачи данных TXD в доминантном состоянии
--------	---

После определения что будет управлять устройством, как выводить информацию и как подключить устройство к информационной шине болида, следующим шагом стал подбор подходящего GPS приемника и акселерометра

В качестве приемника GPS был выбран модуль на базе ATGM366H , он компактен, имеет съемную антенну, встроенную память, при малой цене обладая высокой точностью позиционирования и скоростью нахождения координат. Так же для работы с ним есть удобная библиотека TinyGPS++ [6].



Рисунок 9 – GPS приемник

Таблица 7 – основные характеристики ATGM366H

Параметр	Значение
Типы принимаемого сигнала	DS/GPS/GLONASS/GALILEO/QZSS/SBAS
Время холодного старта	≤ 35 сек.
Время горячего старта	≤ 1 сек.
Точность позиционирования	< 2 метра
Частота обновления	1 – 10 Гц
Формат данных	8 бит + 1 стоп-бит
Интерфейс подключения	UART
Напряжение питания	5V
Потребляемый ток	< 25 мА

В качестве акселерометра было решено использовать 9-ти позиционный IMU-сенсор, который позволил бы, используя всего 1 чип, производить многочисленные измерения. Выбор пал на сравнительно бюджетный MPU-9250 [20].

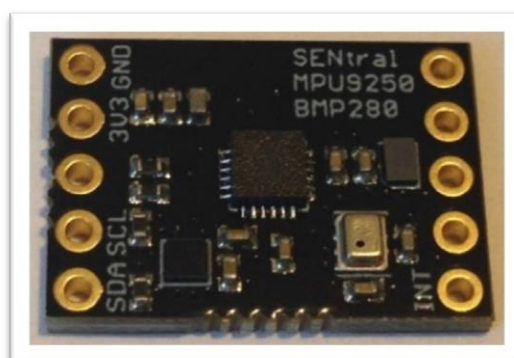


Рисунок 10 – IMU-датчик MPU-9250

Таблица 8 – основные характеристики MPU-9250

Параметр	Значение
Датчики	Акселерометр Магнитометр Гироскоп
Число осей измерения	3
Требуемое питание	3V

Далее необходимо было определить, как передавать данные телеметрии в реальном времени. Были рассмотрены такие варианты как установка LTE модуля и подключение к веб-серверу, запущенном на Raspberry Pi, а также использование Wi-Fi и Bluetooth. Однако необходимо было обеспечить не только саму передачу данных и ее скорость, но и дальность. Обусловлено это тем, что расстояние от самого трека до командных боксов на разных соревнованиях разное, так, если в Москве оно составляет менее 300 метров, то на отдельных этапах в Европе оно может достигать 1 километра. Таким образом было найдено решение в виде использования технологии LoRa – проприетарной маломощной технологии передачи данных на дистанции в от 1-2 до 10-15 километров [19].

LoRa-передатчики компактны, а выбрав частоту и подобрав подходящую антенну можно добиться быстрой и стабильной передачи данных на необходимой дистанции. При этом для приема сигнала на компьютер необходим всего лишь USB-адаптер.

Из доступных вариантов, оптимальным был выбран E32-433T30D производства EBYTE — это компактный передатчик со съемной стандартной антенной, обладающий большой дальностью передачи.



Рисунок 11 – Модуль LoRa E32-433T30D

Таблица 9 – основные характеристики E32-433T30D

Параметр	Значение
Тип модуля	LoRa
Диапазон	433 МГц
Интерфейс подключения	UART
Чип	SX1278
Мощность	30 dBm
Дистанция	8 км
Тип антенны	SMA-K

Так как все низковольтные системы болида питаются от мотоциклетного 12V аккумулятора, прямое включение компонентов не является возможным, ввиду их различных требований к питанию, таким образом было решено использовать DC-DC преобразователь.

Среди доступных вариантов, подходящим оказался Mini560 DC-DC на основе чипа Joulwatt JW5068A. Это компактный понижающий модуль с регулируемым выходным напряжением.

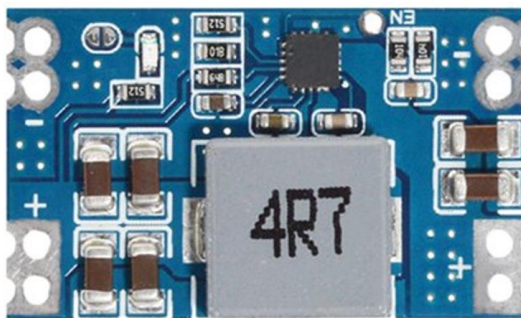


Рисунок 12 – Модуль Mini560

Таблица 10 – основные характеристики Mini560 DC-DC

Параметр	Значение
Микросхема	JW5068A
Тип преобразования	широотно-импульсный
Частота преобразования	500 кГц
Максимальный КПД	98%
Входное напряжение	6,7 – 20 V
Выходное напряжение	3.3V; 5V; 9V; 12V;
Номинальный ток	4 A
Максимальный ток	5A
Диапазон рабочих температур	-40° – +85°

Кнопки были выбраны из списка доступных на Aliexpress, оптимально-подходящие по цене и условиям доставки, с подсветкой и рабочим напряжением 3.3-6 V [4].



Рисунок 13 – Кнопки управления

После определения списка компонентов, с которыми предстоит работать, следующим шагом являлась разработка схемы питания для них.

2.2 Разработка схемы питания

Использование в одном устройстве компонентов, требующих для работы различные уровни напряжений, требует разработки подходящей схемы питания.

Таблица 11 – Список компонентов и необходимый для них уровень напряжения

Параметр	Значение
Компонент	Необходимый уровень напряжения, V
Микроконтроллер	5
ЖК-дисплей	3.3
CAN-трансивер	5
GPS-приемник	3.3
LoRa-передатчик	5

Продолжение таблицы 11

IMU-датчик	3.3
Кнопки управления	5

На основе этой таблицы была, с помощью онлайн-сервиса Progr@m4you была составлена блок-схема системы питания [10].

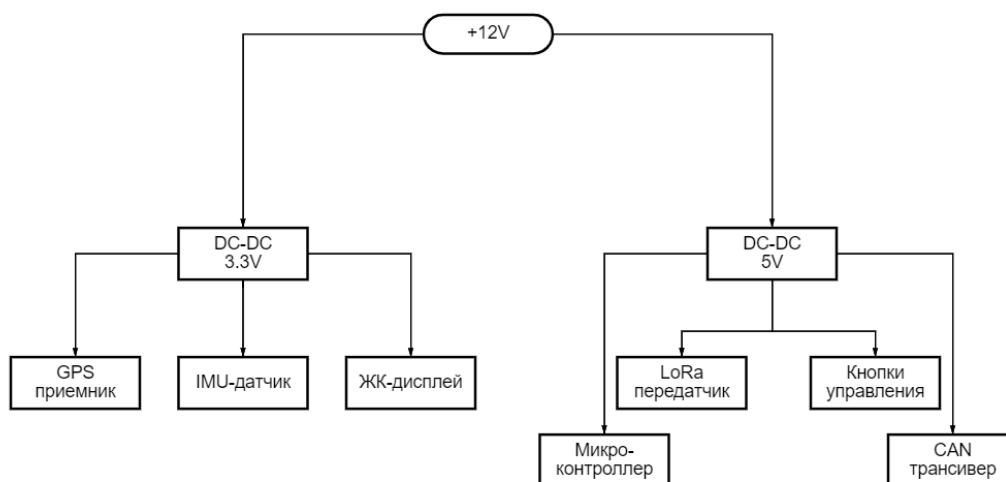


Рисунок 14 – Блок-схема системы питания

Разработка блок-схемы для схемы питания позволит упростить последующую разработку самой печатаной платы, ввиду удобства восприятия графического материала.

2.3 Разработка структурной схемы

Для упрощения последующего проектирования печатной платы, следующим шагом является создание структурной схемы будущего устройства.

Для этого с помощью графического редактора создается схема, на которой изображены все модули в связи между собой. Так же на схеме отображена CAN-шина и система питания.

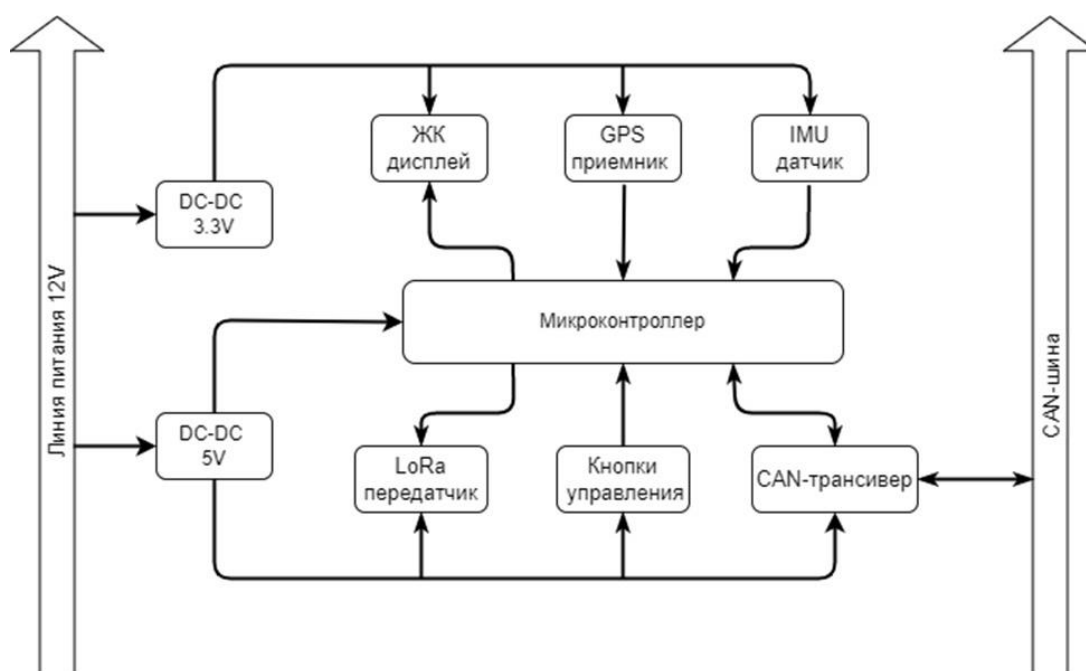


Рисунок 15 – Структурная схема устройства

Структурная схема, включающая в себя схему питания, позволит упростить проектирование платы, уменьшив вероятность ошибок.

2.4 Разработка печатной платы

Проектирование печатной платы началось с поиска подходящей специализированной САПР, так как использование цифровых технологий позволяет сократить время проектирования и повысить качество финального образца. Были отобраны два варианта – EasyEDA и DipTrace

DipTrace — это многофункциональная САПР для разработки печатных плат, а также документации для проектов различной сложности. Эта САПР предлагает простой пользовательский интерфейс, многолистовую и многоуровневую иерархию, скоростной авто-трассировщик, стандартные библиотеки на 130 тыс. компонентов и бесплатную поддержку. Так же имеется учебник и обучающие программы, доступные на 22 языках [11].

DipTrace включает в себя 4 редактора: редактор электрических схем, редактор печатных плат, редактор компонентов и редактор посадочных мест.

Так же имеется возможность предварительного просмотра 3D-модели в реальном времени и ее экспорта для последующего моделирования.

DipTrace Schematics — инструмент по разработке сложных многоуровневых иерархических принципиальных схем со множеством функций по созданию визуальных и логических связей между выводами компонентов. Он позволяет провести обратную аннотацию с печатной платы, либо импортировать проекты из других САПР, в том числе в виде списков соединений. Имеется система верификации и Spice-экспорт для полноценного анализа проекта. Есть проверка ERC и экспорт в Spice для внешнего моделирования.

DipTrace PCB Layout — инструмент для разработки печатных плат с мощным автотрассировщиком и автоматическим позиционированием компонентов, возможностью копирования трассировки между иерархическими блоками и всеми необходимыми функциями для работы в том числе с высокоскоростными и дифференциальными сигналами Real-Time DRC предупреждает о возможных ошибках в удобной форме, а импорт/экспорт «нетлистов» и производственных файлов, таких как Gerber, позволяет интегрировать DipTrace в современный мир электроники.

EasyEDA — кроссплатформенный веб-набор инструментов автоматизации проектирования электроники, позволяющий проектировать, симулировать, делиться и обсуждать схемы и печатные платы. Так же она позволяет создавать техническую документацию, производственные файлы и экспортировать изображения [15]. Программа дает возможность создавать и редактировать принципиальные схемы, проводить Spice-моделирование смешанных аналоговых и цифровых схем, проектировать и редактировать печатные платы. В состав EasyEDA входят следующие компоненты:

- редактор принципиальных схем;
- редактор печатных плат;
- SPICE-симулятор;

- редактор электронных компонентов;
- генератор файлов формата GERBER;
- программа просмотра файлов формата GERBER;
- система управления проектами;
- система заказа изготовления печатных плат;
- облачное хранилище файлов.

Среда является браузерным веб-приложением, независимым от локальных данных пользователя. Так как сервис является бесплатным, не имеет ограничений, кроме количества частных проектов и не требует инсталляции каких-либо программ, а файлы хранятся на облачном сервере, то программа позволяет продолжать работу над проектом на любом компьютере, подключённом к сети Интернет.

Ввиду вышеперечисленного, было принято решение проектировать в среде EasyEDA.

Первый шаг при проектировании платы – создание принципиальной схемы в соответствующей рабочей области

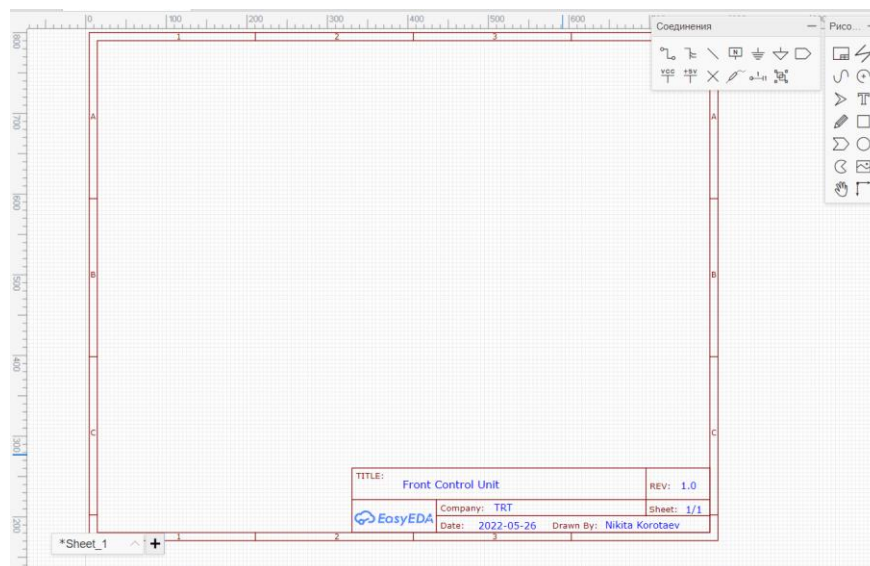


Рисунок 16 – Рабочая область создания принципиальной схемы

Компоненты размещаются опираясь на структурную схему, представленную на рисунке 15. Первым – микроконтроллер. Для этого,

нажав правой кнопкой мыши на рабочей области, выбирается пункт «Place Component» (Разместить Компонент)

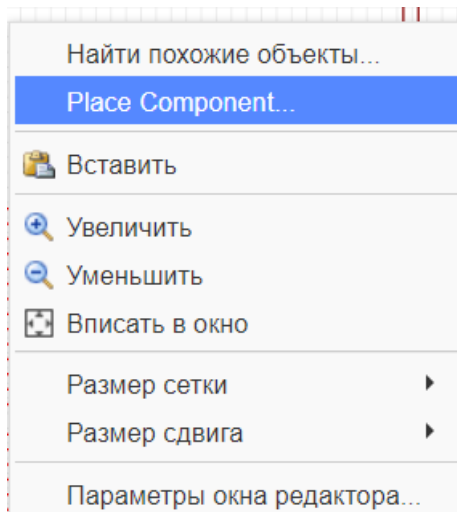


Рисунок 17 – Контекстное меню выбора

В открывшемся окне, в строке поиска, вбивается название необходимого компонента, после чего из появившегося списка выбирается подходящий.

Ввиду того, что EasyEDA это бесплатное веб-приложение, каждый созданный компонент зарегистрированных пользователей так же доступен к выбору. Эта отдельная библиотека является отдельным плюсом, потому как невозможно даже в самую богатую библиотеку внести все возможные элементы. Именно так и произошло с выбранным микроконтроллером: он присутствовал только среди пользовательских элементов, причем в различных вариантах исполнения его как на принципиальной схеме, так и непосредственно на плате.

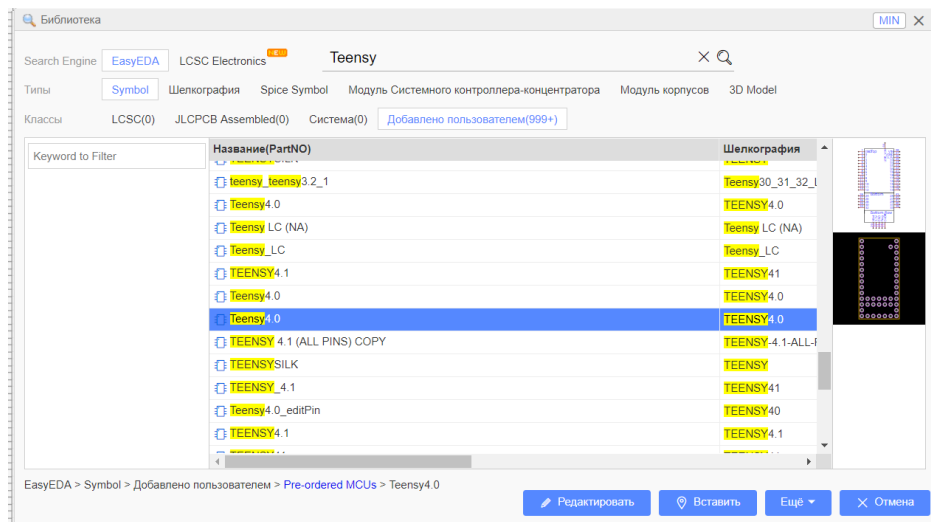


Рисунок 18 – Меню поиска компонентов

Выбрав подходящий компонент, размещаем его на рабочей поверхности

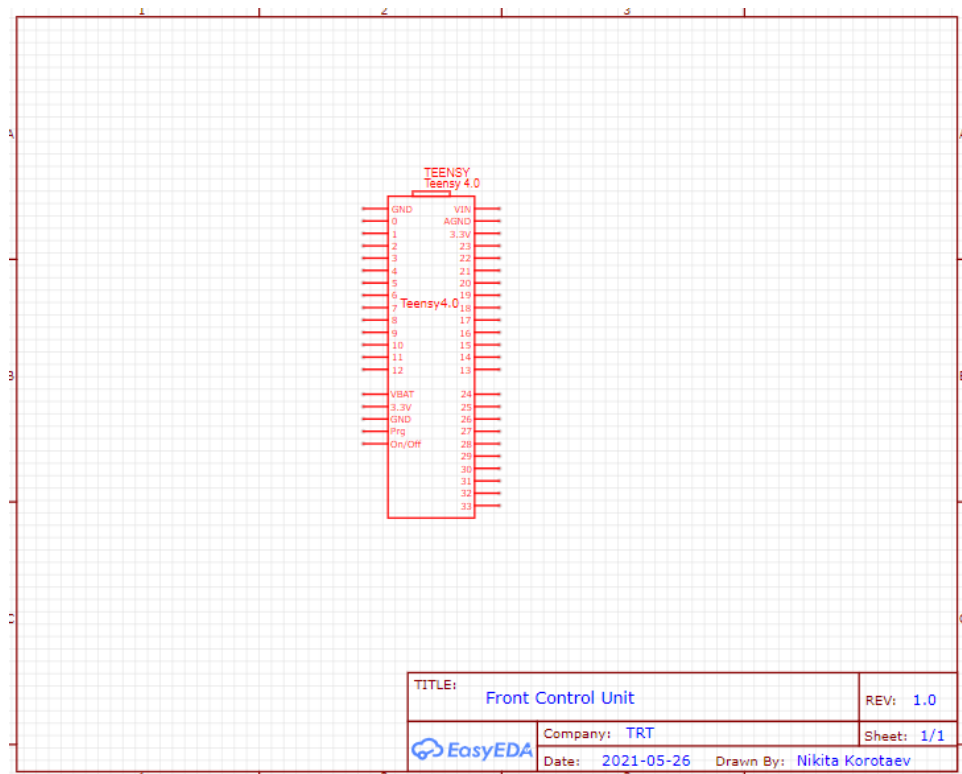


Рисунок 19 – Микроконтроллер, размещенный на рабочей плоскости

Последующие компоненты размещаются аналогичным образом

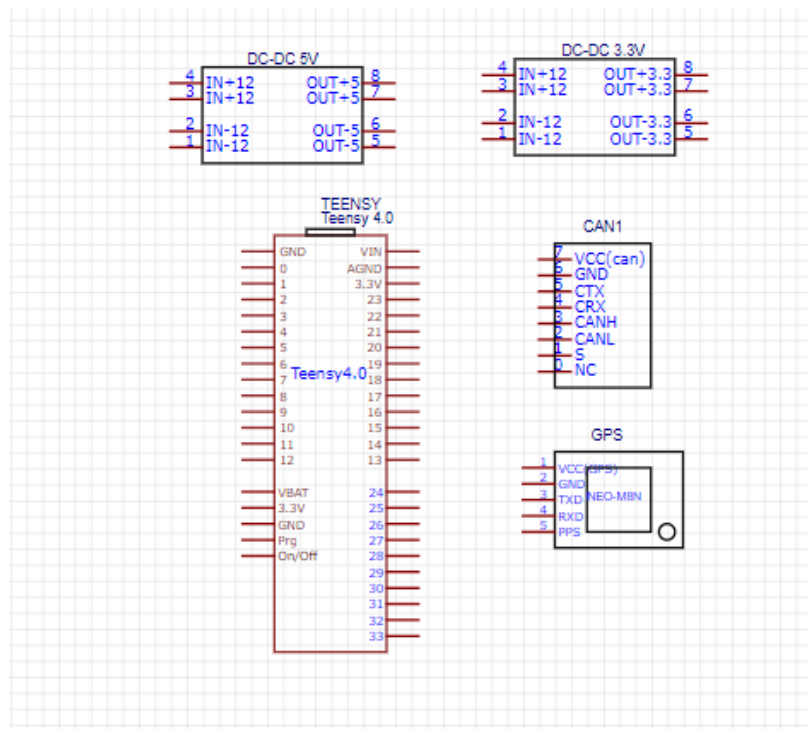


Рисунок 20 – Модули, размещенные на рабочей плоскости

Размещение модуля дисплея, подключения которого осуществляется через 20-ти разъемный прямой коннектор не является целесообразным. Беспроводной передатчик LoRa не был найден, его размеры и расположение будут отдельно учитываться при проектировании платы.

Следующим шагом является размещение всех необходимых коннекторов: питание, кнопки управление, CAN-шина и т.д.

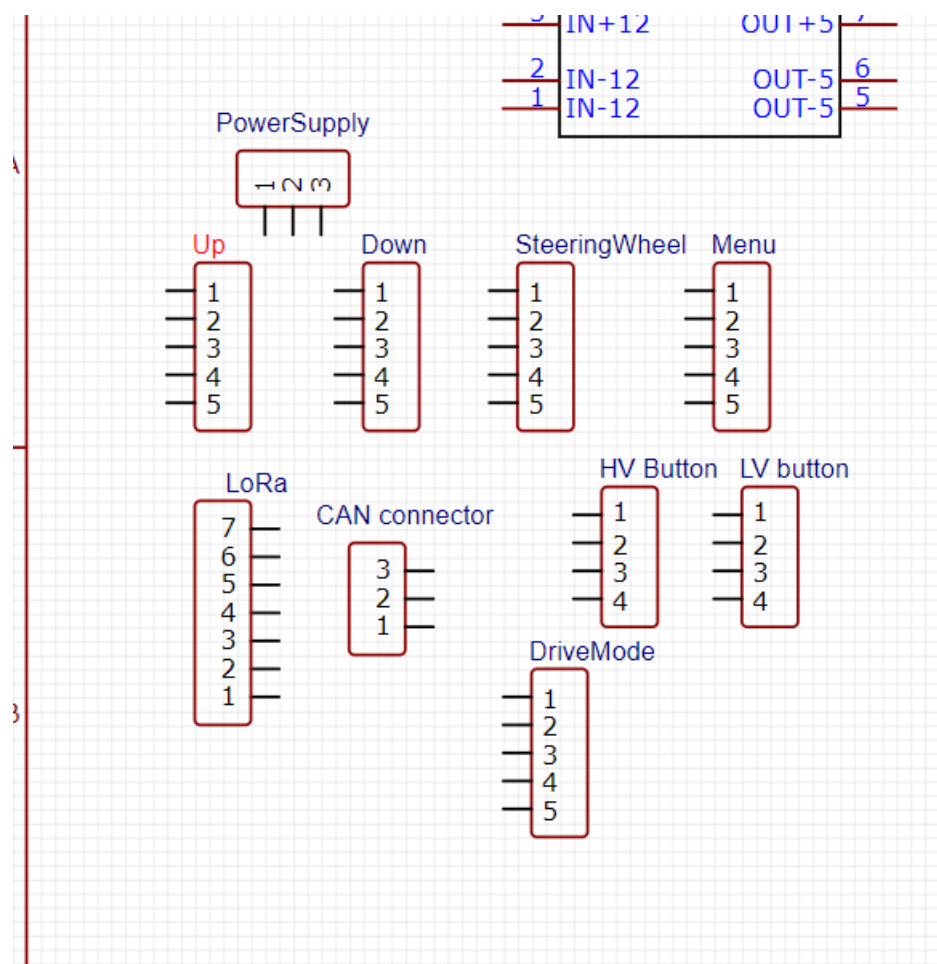


Рисунок 21 – Коннекторы, размещенные на рабочей плоскости

В виду того, что некоторые входные сигналы могут превышать допустимый для Teensy уровень напряжения 3.3V, было решено разместить 6 делителей напряжения. В случае, когда в их задействовании нет необходимости, вместо SMD резисторов будут использованы перемычки.

Резистор – стандартны компонент, он имеется во встроенной библиотеке, расположенной слева от рабочей плоскости

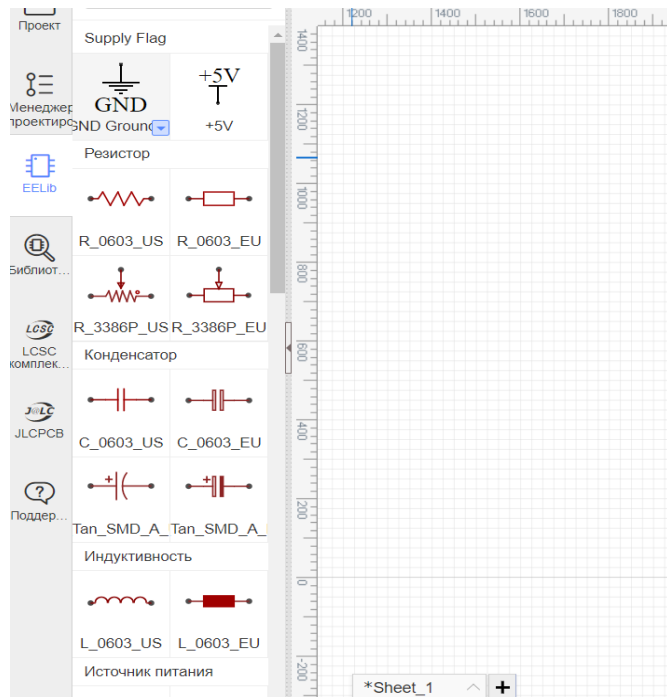


Рисунок 22 – Область стандартной библиотеки

В данной библиотеке есть различный набор элементов, изображенных в соответствии как с Европейскими, так и с Американскими стандартами.

В данной САПР доступны различные способы создания соединений. Так же доступно проставление размеров, рисование и добавление текста – все это используется из перемещаемых подменю, по умолчанию находящихся в правом верхнем углу рабочей плоскости.

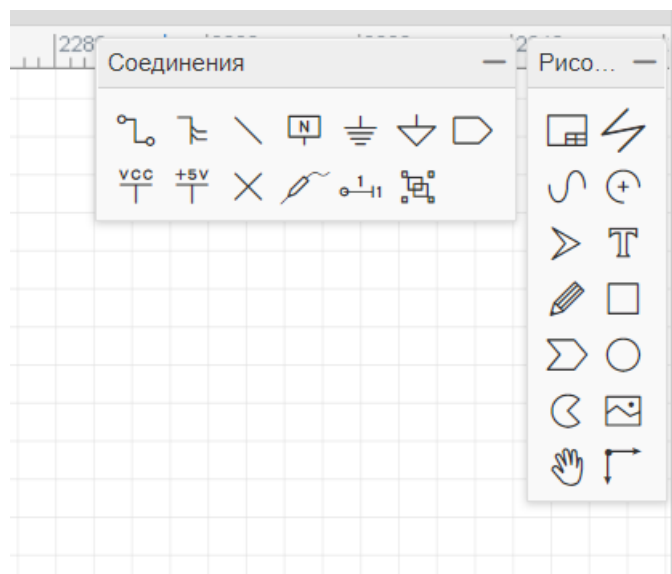


Рисунок 23 – Подменю соединений и рисования

Далее создается простой делитель напряжения. Для подключения к «земле» используется советующий флажок, означающий подключение к общей шине. Это упрощает проектирование.

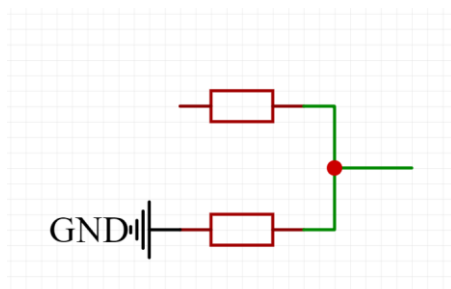


Рисунок 24 – Делитель напряжения

По аналогии создаются остальные 5 необходимых.

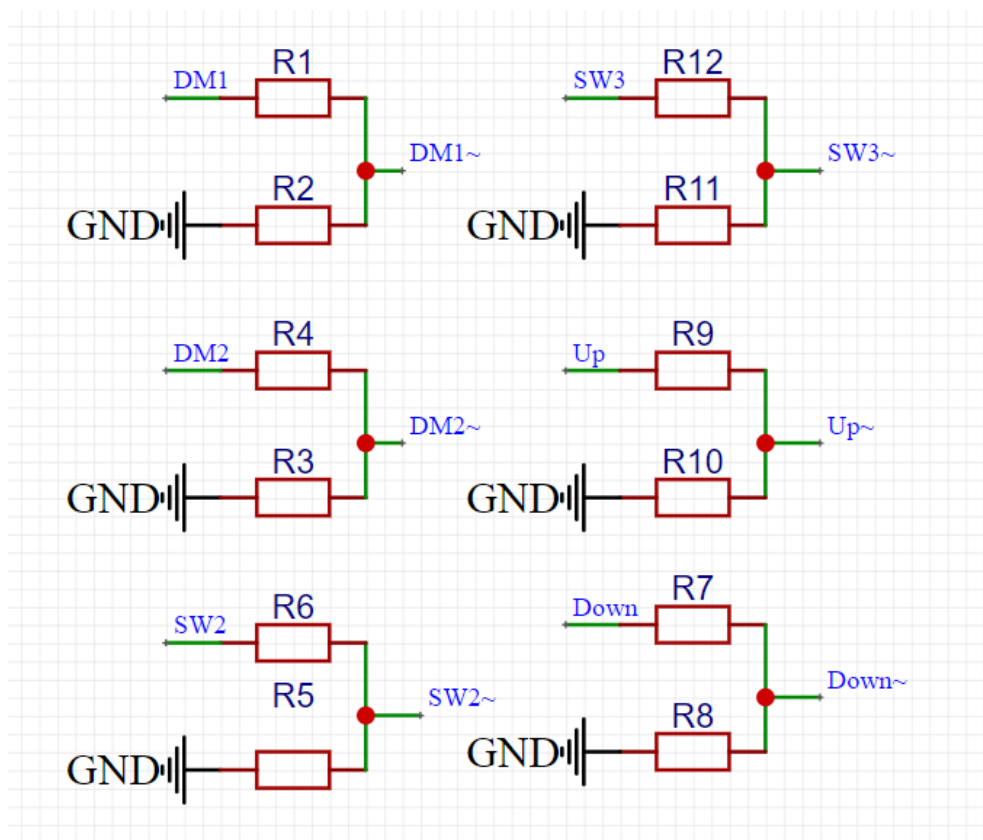


Рисунок 25 – Все делители напряжения

Расположив все элементы, необходимо их соединить. На примере создания делителя можно увидеть два возможных варианта: линиями и помощью портов, подключенных к общей шине. Второй вариант был выбран как более простой и удобный.

Каждый порт с одинаковым названием подключен ко всем другим портам с таким же названием.

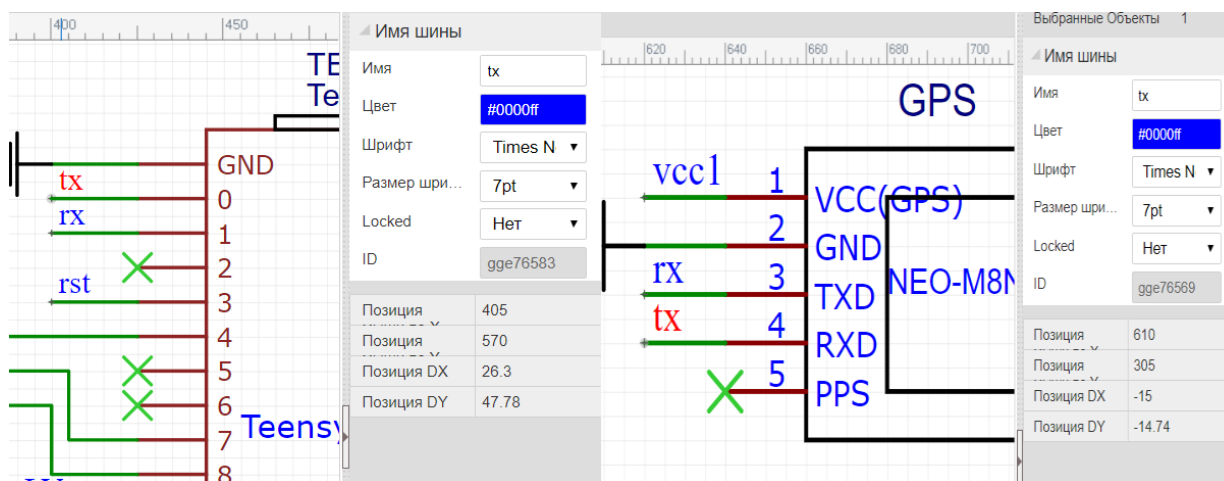


Рисунок 26 – Соединение с помощью портов

Таким образом делаются все остальные соединения. Те коннекторы, подключение которых не предполагается, помечаются зеленым крестиком.

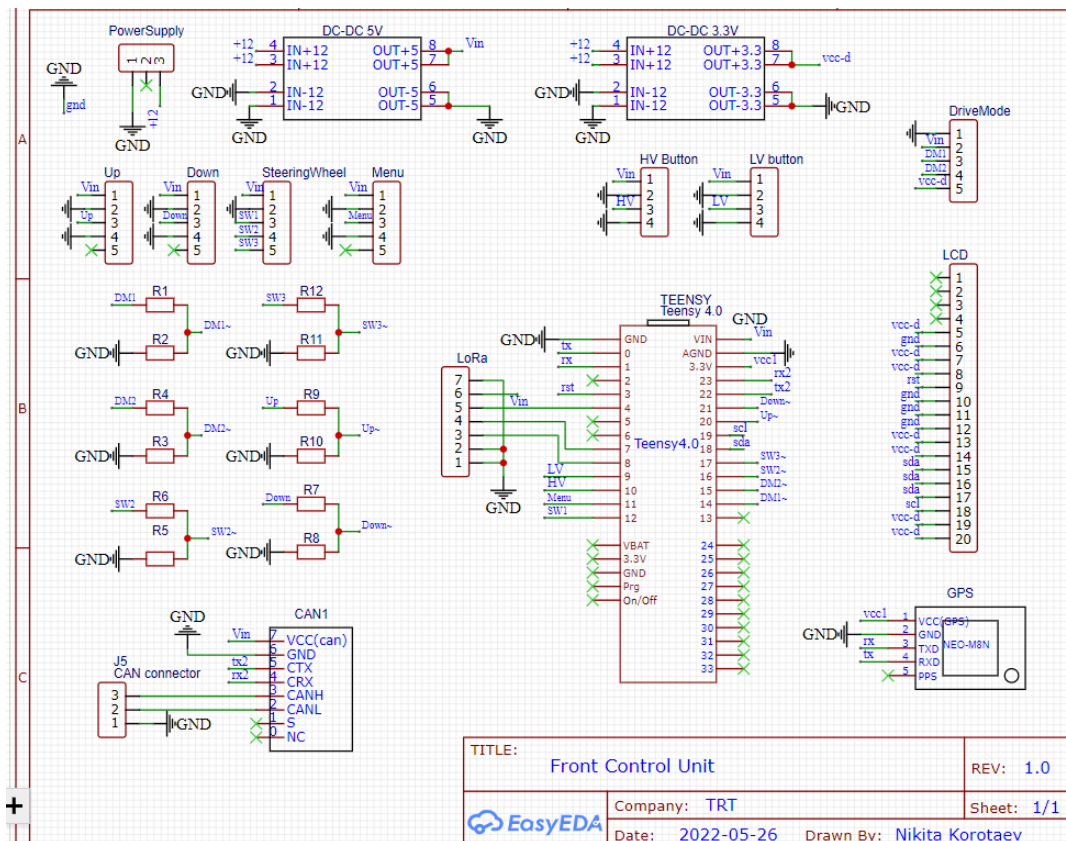


Рисунок 27 – Общий вид принципиальной схемы

Разместив и связав все компоненты, можно приступить к проектированию непосредственно самой печатной платы. Принципиальная схема в системе EasyEDA представлена в Приложении А.

Для начала проектирования печатаной платы требуется сохранить проект, и в выпадающем меню Design выбрать пункт преобразовать схему в печатную плату.

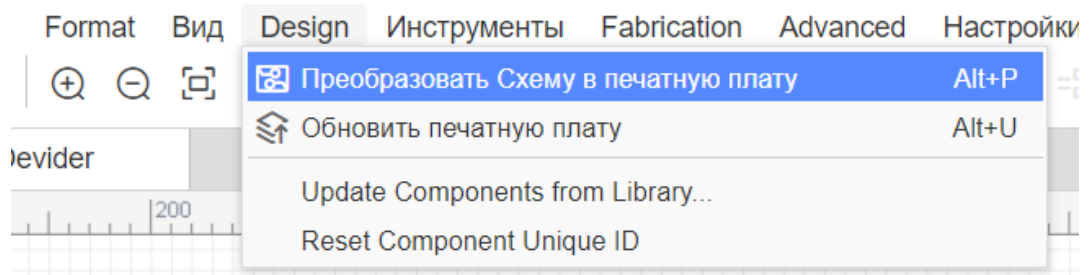


Рисунок 28 – Пункт преобразования в печатную плату

После загрузки, появляется окно, предлагающее задать параметры будущей платы, такие как размер, количество слоев и тип контура.

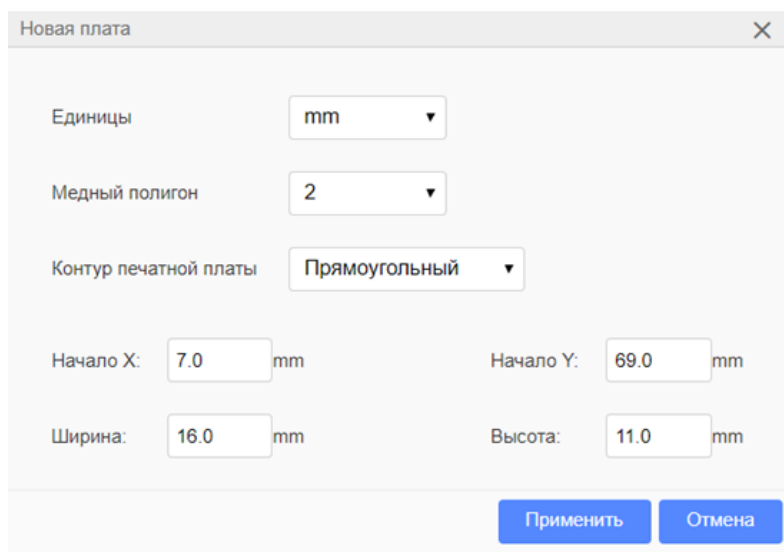


Рисунок 29 – Параметры платы

Применив заданные параметры, открывается следующий вид: контурные изображения всех добавленных элементов были помещены на экран, а фиолетовым цветом обозначены границы самой печатной платы

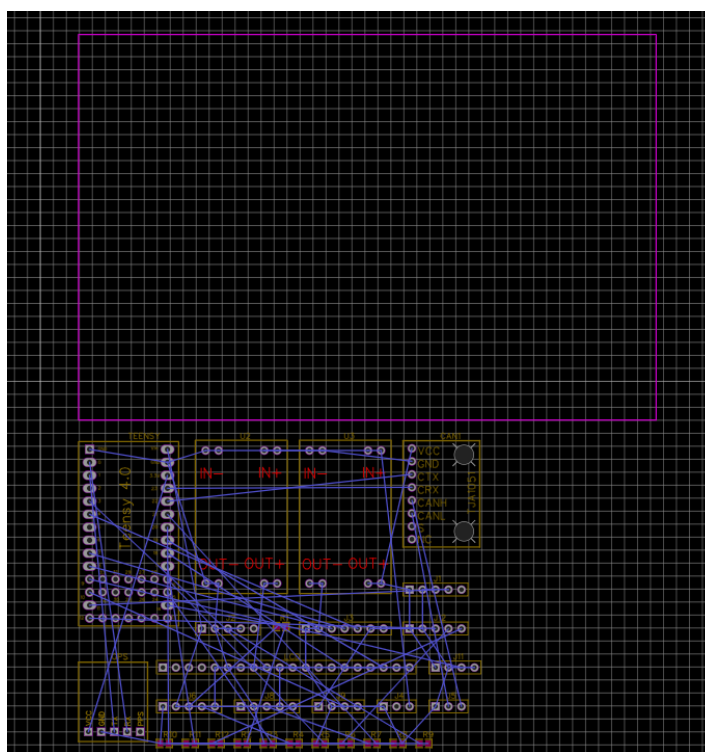


Рисунок 30 – Рабочая плоскость проектирования печатной платы

В правом верхнем углу расположены инструменты для работы и окно переключения между слоями

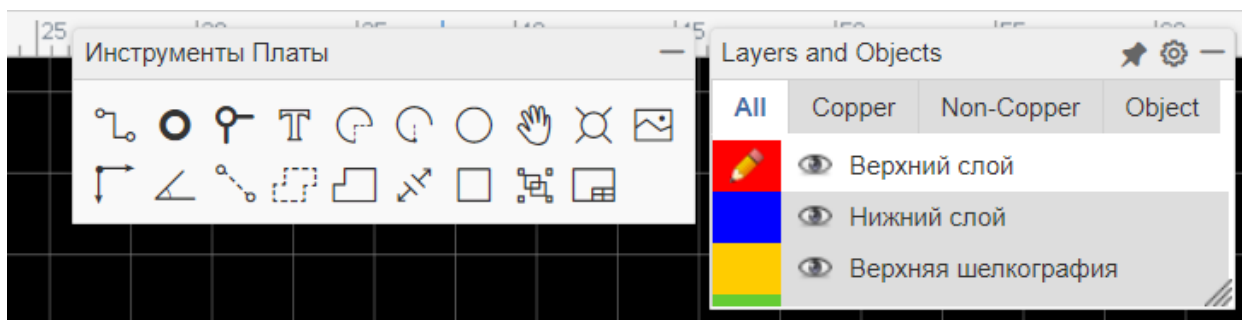


Рисунок 31 – Доступные инструменты и слои

В зависимости от выбранного слоя и инструмента, можно редактировать шелкографию, добавлять текст, проставлять размеры, создавать медные полигоны и проводить медные дорожки.

Все элементы размещаются в пределах контура печатной платы. С помощью соответствующего инструмента построен медный полигон для общей «земли».

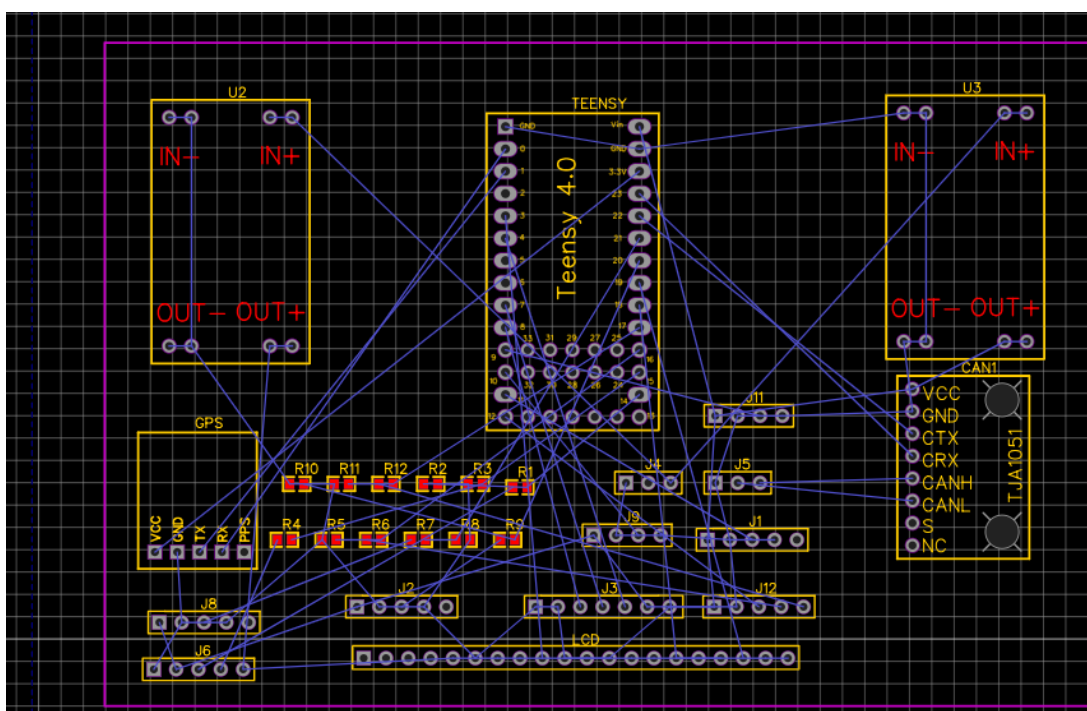


Рисунок 32 – Элементы, размещенные в габаритах печатной платы.

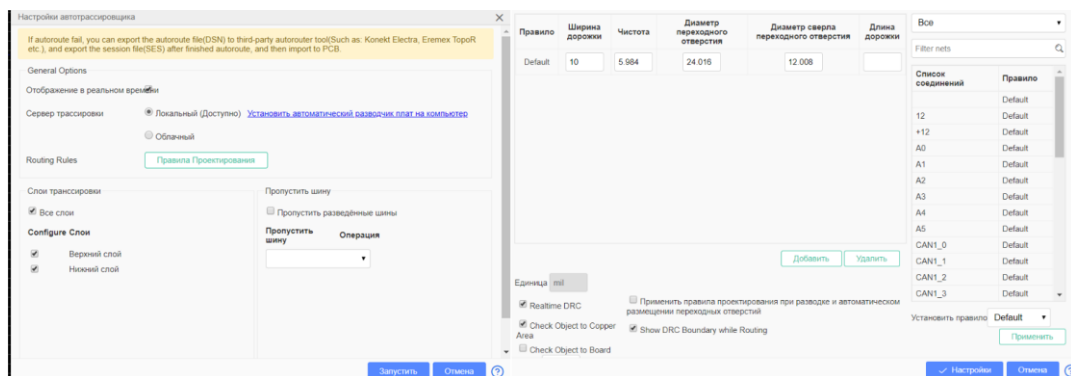


Рисунок 35 – Окно настройки автоматической трассировки и выставления специальных правил трассировки

Разместив все элементы и произведя трассировку, были добавлены крепежные отверстия, справочные таблицы, логотип команды, а также подписаны все коннекторы.

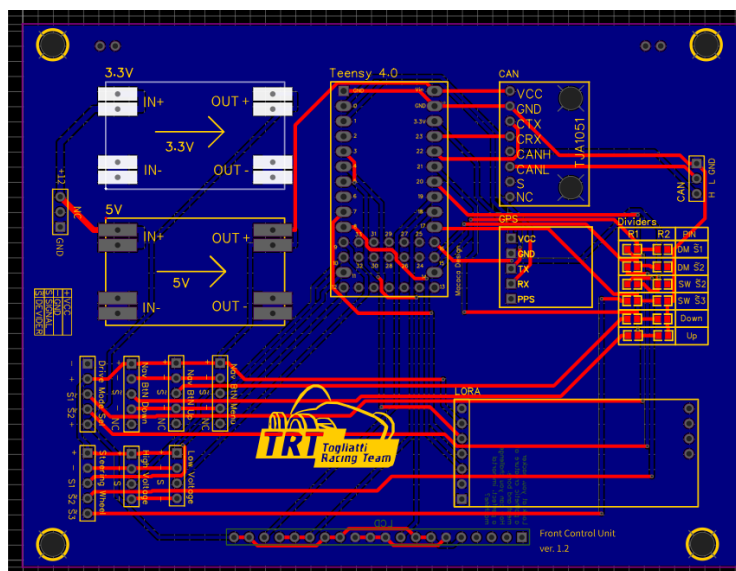


Рисунок 36 – Вид спроектированной платы

Так же программа имеет возможность предпросмотра готового изделия как в двухмерном, так и трехмерном виде. Для последнего необходимо, чтобы каждый элемент имел привязанную 3D-модель.

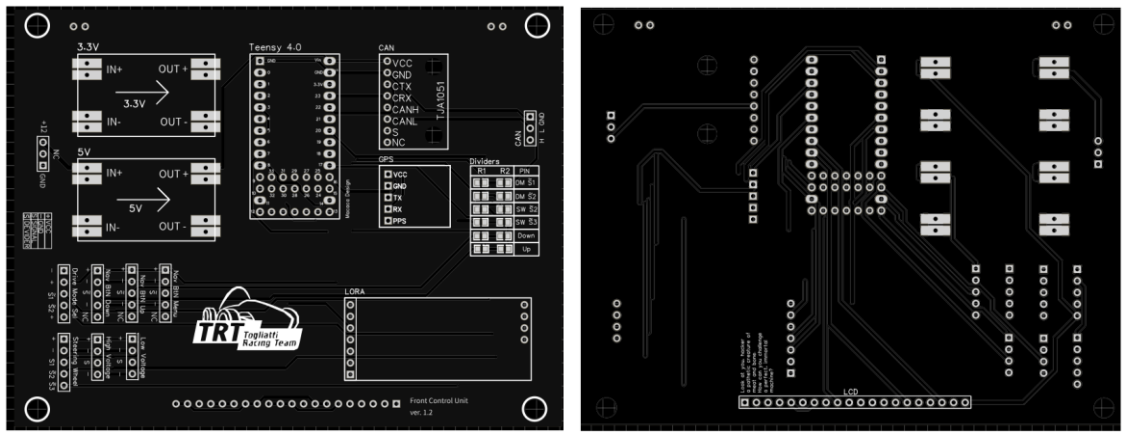


Рисунок 37 – Двухмерный общий вид готовой платы с двух сторон

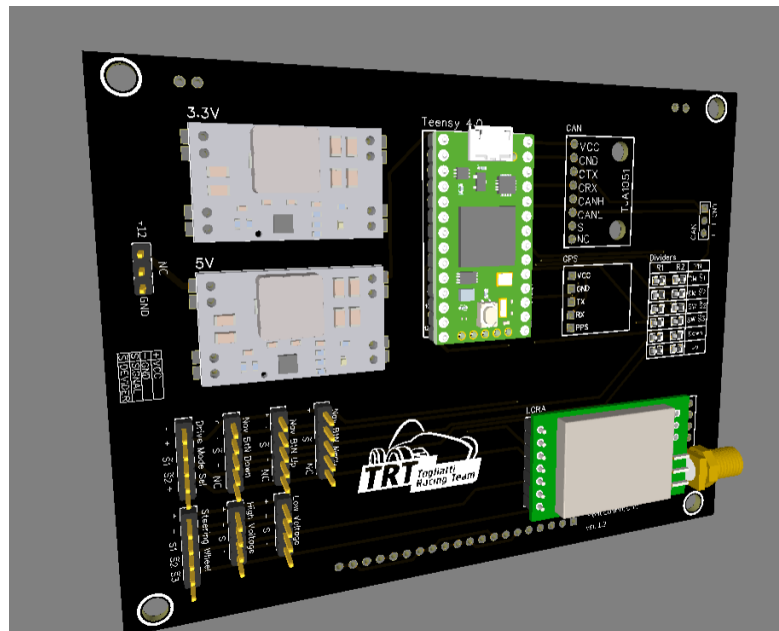


Рисунок 38 – 3D вид платы

На данном трехмерном виде наглядно виден главный минус трехмерного предпросмотрщика – не все элементы обладают качественными моделями, ввиду того что сделаны они простыми пользователями. Для некоторых элементов трехмерных моделей не существует вовсе.

Вывод

По завершению данного этапа, на основе подобранных компонентов и необходимых требований, был разработан дизайн печатной платы, позволяющий произвести дальнейшее создание ее создания.

3 Создание печатной платы

Самостоятельное производство платы достаточно сложный процесс, а разработана она была в САПР, поддерживающей создание GERBER-файлов для заказа платы на фабрике в Китае. Фабричное производство во всех отношениях лучше самостоятельного.

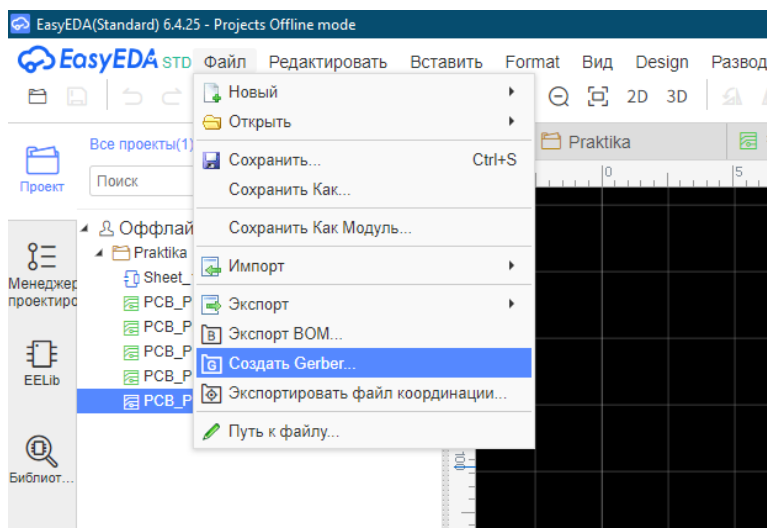


Рисунок 39 – Пункт меню создания Gerber-файла

После этого в появившемся окне необходимо выбрать цвет паяльной маски, обработку поверхности под пайку, количество плат и толщину текстолита.

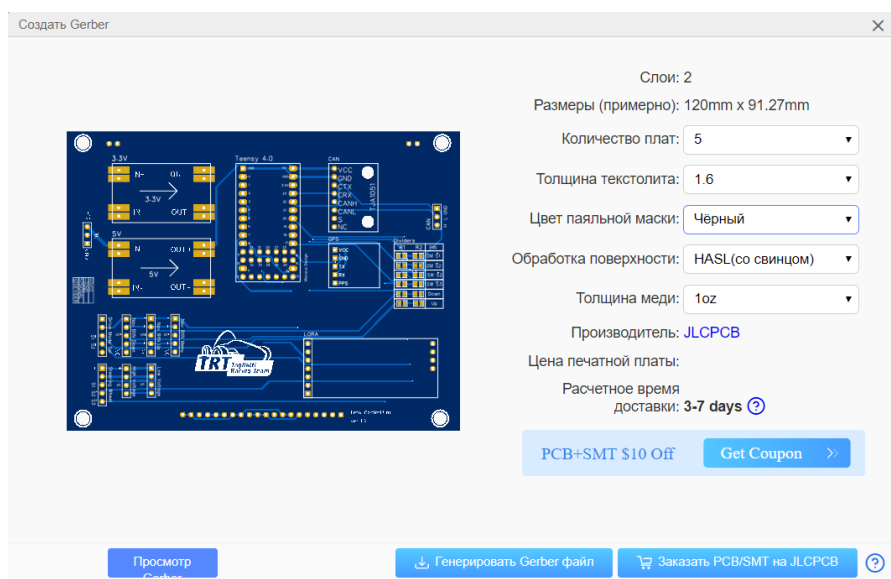


Рисунок 40 – Окно настройки файла

Далее необходимо сгенерировать файл gerber.zip и перейти на сайт JLPCB [17] и кликнуть на кнопку загрузки файла.

The image shows a screenshot of the JLPCB website's PCB configuration interface. At the top, it displays the detected board dimensions: "Detected 2 layer board of 91.27x120mm(3.59x4.72 inches)". Below this, there are two preview images of the PCB. The main configuration area includes the following options:

- Base Material: FR-4 (selected), Aluminum
- Layers: 1, 2 (selected), 4, 6
- Dimensions: 91.27 x 120 mm
- PCB Qty: 5
- Product Type: Industrial/Consumer electronics (selected), Military/Aerospace, Medical
- Different Design: 1, 2, 3, 4
- Delivery Format: Single PCB (selected), Panel by Customer, Panel by JLPCB
- PCB Thickness: 0.4, 0.6, 0.8, 1.0, 1.2, 1.6 (selected), 2.0
- PCB Color: Green, Purple, Red, Yellow, Blue, White, Black (selected)
- Silkscreen: White (selected)
- Surface Finish: HASL(with lead) (selected), LeadFree HASL-RoHS, ENIG-RoHS
- Outer Copper Weight: 1 oz (selected), 2 oz
- Gold Fingers: No (selected), Yes
- Confirm Production File: No (selected), Yes
- Flying Probe Test: Fully Test (selected), Not Test
- Castellated Holes: No (selected), Yes
- Remove Order Number: No (selected), Yes, Specify a location

On the right side, there is a "Charge Details" summary:

Engineering fee	\$4.00
Board	\$4.50
Build Time	
PCB: 3 days	\$0.00
Calculated Price	\$8.50
Additional charges may apply for special cases	
Weight	0.23kg

Below the summary is a "SAVE TO CART" button and a "Shipping Estimate" section with a "Charge" dropdown menu.

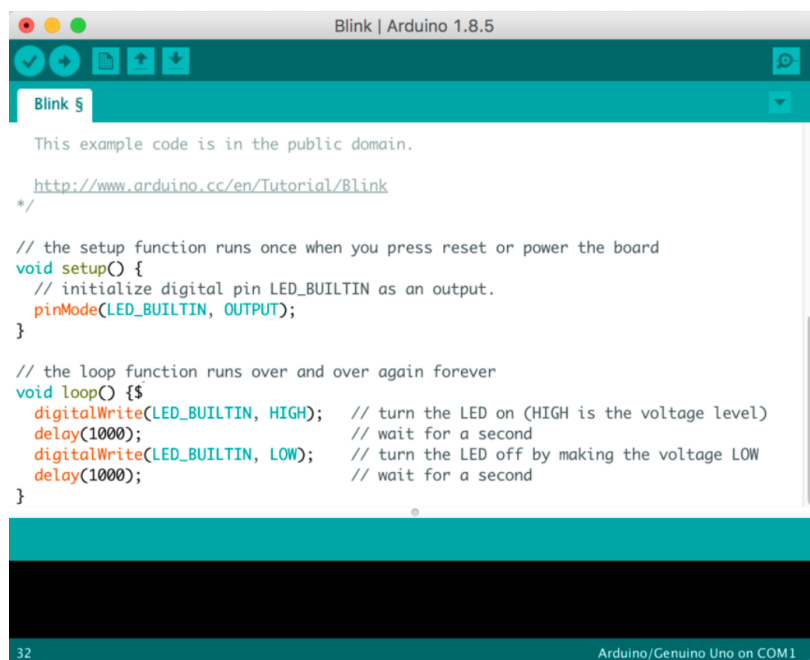
Рисунок 41 – Форма заказа платы на сайте jlpcb

Вывод

Благодаря развитию современных технологий и интеграции различных стран в международную торговлю, необходимость вручную разводить печатные платы отпала, так как заказать производство на профильном предприятии, с соответствующим качеством производства, является достаточно бюджетным способом получения печатной платы, даже для студента или школьника.

4 Разработка алгоритма и управляющей программы

Разработка управляющей программы началась с выбора среды программирования. Так как контроллер используется ардуино-совместимый, то было два варианта: Arduino IDE [9] и Visual Studio Code [28].

The image shows a screenshot of the Arduino IDE interface. The title bar reads "Blink | Arduino 1.8.5". The main window displays the source code for the "Blink" example. The code is as follows:

```
This example code is in the public domain.

http://www.arduino.cc/en/Tutorial/Blink
*/

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000); // wait for a second
  digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
  delay(1000); // wait for a second
}
```

The status bar at the bottom indicates "32" on the left and "Arduino/Genuino Uno on COM1" on the right.

Рисунок 42 – Среда разработки Arduino IDE

Это стандартная среда разработки Arduino, ввиду этого она так же, как и сами микроконтроллеры Arduino не очень хорошо подходит для написания больших и сложных программ ввиду малого удобства интерфейса и навигации. Также встроенный компилятор обладает слишком большими допусками, например позволяет сравнивать signed и unsigned переменные.

В связи с этим выбор заранее был сделан в пользу Visual Studio Code. Это легкий, кроссплатформенный редактор исходного кода, разработанный Microsoft.

Он обладает возможностью настройки «под себя», а так же широкой библиотекой пользовательских расширений, одно из которых – PlatformIO

[22], среда разработки, позволяющая создавать программы для Arduino и доступная в качестве расширения.

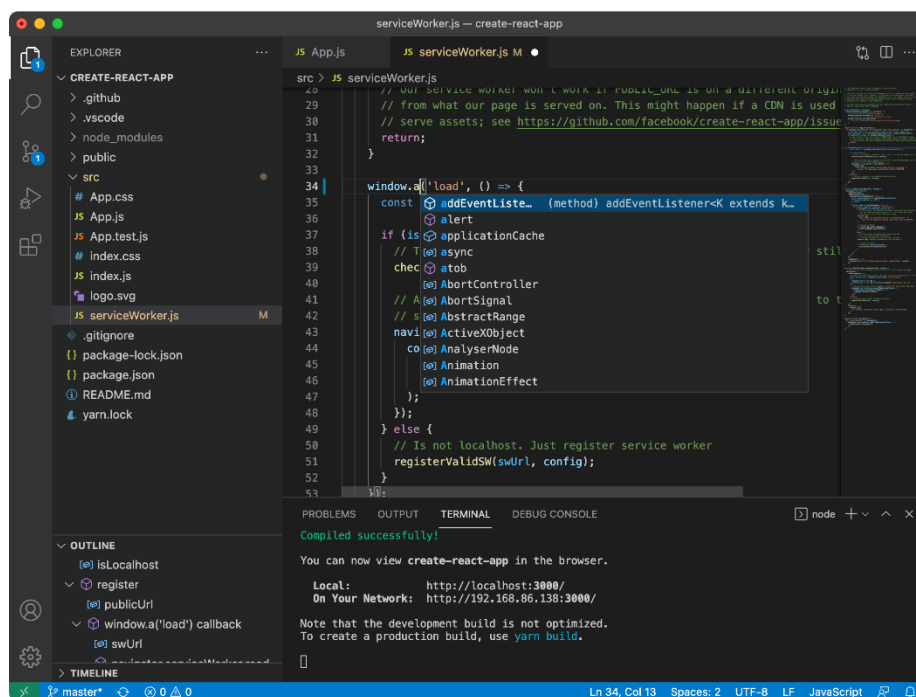


Рисунок 43 – Visual Studio Code

Разработка велась блоками для каждого модуля по отдельности, а затем все это было объединено воедино.

Сначала было подключение дисплея, для этого использовалась библиотека U8glib [27]. На рисунке 44 приведен пример функций, отвечающих за вывод на дисплей информации о заряде батареи, оставшемся запасе мощности, температуре моторов и инверторов в виде шкал

```

void printBatteryCharge(int &BatteryChargeBar)
{
  u8g2.drawFrame(0, 18, 256, 25);           //Charge Bar box
  u8g2.drawRBox(2, 20, BatteryChargeBar, 21, 0); //Charge bar graphic
}
void printPowerGraph (int &PowerGraph)
{
  u8g2.drawFrame(0, 62, 134, 25);          //Power bar box
  u8g2.drawRBox(2, 64, PowerGraph, 21, 0); //Power bargraphic
}
void printBatteryTemp(int &BatteryTempBar)
{
  u8g2.drawFrame(137, 62, 119, 25);        //Battery temp bar box
  u8g2.drawRBox (139, 64, BatteryTempBar, 21, 0); //Battery bar graphic
}
void printMotorTemp(int &MotorTempGraph)
{
  u8g2.drawFrame(137, 106, 119, 25);       //Motor temp bar box
  u8g2.drawRBox (139, 108, MotorTempGraph, 21, 0); //Motor bar graphic
}
void printInvertorTemp(int &InvertorTempGraph)
{
  u8g2.drawFrame(0, 106, 134, 25);         //Invertor temp bar box
  u8g2.drawRBox(2, 108, InvertorTempGraph, 21, 0); //Invertor temp bargraphic
}

```

Рисунок 44 – Пример использования библиотеки

Общее графическое оформление интерфейса выведено в отдельные функции. Применяя полученные знания, удалось сделать анимацию некоторых иконок, так, например, до тех порка не установится подключение LoRa-передатчика к ответному приемнику, иконка моргает, а после – горит постоянно.

```

void printLORAicon ()
{
  u8g2.drawRBox(240, 146, 3, 12,0 );
  u8g2.drawDisc(241, 144, 3);
  if (millis() - LoraSignalIcon1 > 999){
    u8g2.drawEllipse(238, 144, 3, 5, U8G2_DRAW_LOWER_LEFT);
    u8g2.drawEllipse(238, 144, 3, 5, U8G2_DRAW_UPPER_LEFT);
    u8g2.drawEllipse(244, 144, 3, 5, U8G2_DRAW_UPPER_RIGHT);
    u8g2.drawEllipse(244, 144, 3, 5, U8G2_DRAW_LOWER_RIGHT);
  }
  if (millis() - LoraSignalIcon1 > 1222){
    u8g2.drawEllipse(238, 144, 5, 7, U8G2_DRAW_LOWER_LEFT);
    u8g2.drawEllipse(238, 144, 5, 7, U8G2_DRAW_UPPER_LEFT);
    u8g2.drawEllipse(244, 144, 5, 7, U8G2_DRAW_UPPER_RIGHT);
    u8g2.drawEllipse(244, 144, 5, 7, U8G2_DRAW_LOWER_RIGHT);
  }
  if (millis() - LoraSignalIcon1 > 1555){
    u8g2.drawEllipse(238, 144, 7, 9, U8G2_DRAW_LOWER_LEFT);
    u8g2.drawEllipse(238, 144, 7, 9, U8G2_DRAW_UPPER_LEFT);
    u8g2.drawEllipse(244, 144, 7, 9, U8G2_DRAW_UPPER_RIGHT);
    u8g2.drawEllipse(244, 144, 7, 9, U8G2_DRAW_LOWER_RIGHT);
  }
  if (millis() - LoraSignalIcon1 > 2500){
    LoraSignalIcon1 = millis();
  }
}

```

Рисунок 45 – Анимация иконок

Так как информация должна быть точной, но при этом понятной при быстром взгляде пилота во время гонки, использование исключительно цифр или шкал было не целесообразным. Простая идея совместить как цифровой, так и графический способ вывода информации оказалась достаточно громоздкой в реализации, ввиду особенностей работы библиотеки.

```
void printText(int &BatteryChargePercent, int &BatteryTempNumbers, int &MotorTempNumbers, int &InvertorTempNumbers)
{
    u8g2.setCursor(217, 2); // Battery % position
    u8g2.print(BatteryChargePercent); // Battery %
    u8g2.drawStr(0, 2, "Charge");
    u8g2.drawStr(243, 2, "%");
    u8g2.drawStr(136, 45, "Battery C");
    u8g2.setFont(u8g2_font_10x20_tf); //changing font for the °
    u8g2.drawUTF8(223,45, DEGREE_SYMBOL);
    u8g2.setFont(u8g2_font_VCR_OSD_tf); //changing it back
    u8g2.setFontMode(1);

    u8g2.setFont(u8g2_font_10x20_tf); //changing font for Battery temp
    u8g2.setCursor(140,67); //Battery temp position
    u8g2.print(BatteryTempNumbers); //Battery temp
    u8g2.setFont(u8g2_font_VCR_OSD_tf); //Changing it back
    u8g2.setDrawColor(1);

    u8g2.drawStr(136, 89, "Motor C");
    u8g2.setFont(u8g2_font_10x20_tf); //changing font for the °
    u8g2.drawUTF8(199,89, DEGREE_SYMBOL);
    u8g2.setFont(u8g2_font_VCR_OSD_tf); //Changing it back
    u8g2.setFontMode(1);
    u8g2.setFont(u8g2_font_10x20_tf); //Changing fon for Motor temp
    u8g2.setCursor(140,111); //Motor temp position
    u8g2.print(MotorTempNumbers); //Motor temp
    u8g2.setFont(u8g2_font_VCR_OSD_tf); //Changing it back
    u8g2.setDrawColor(1);

    u8g2.drawStr(1, 89, "Invertor C");
    u8g2.setFont(u8g2_font_10x20_tf); //Changing font for the °
    u8g2.drawUTF8(100,89, DEGREE_SYMBOL);
    u8g2.setFont(u8g2_font_VCR_OSD_tf); //Changing it back
}
```

Рисунок 46 – Вывод текста и числовых параметров

После подключения дисплея, была начата работа над подключением к CAN-шине. Для этого использовалась библиотека ACAN T4 [5], разработанная специально для микроконтроллера Teensy.

```
ACAN_T4_Settings settings (250 * 1000); // 250 kbit/s
settings.mLoopBackMode = false; //never set true
settings.mSelfReceptionMode = false; //never set true
```

Рисунок 47 – Настройка параметров работы CAN-шины

```

CANMessage CanTelemetry ;
CANMessage BMScontrollerRight ;
CANMessage BMScontrollerLeft ;
CANMessage BMSMotorLeft ;
CANMessage BMSMotorRight ;
if (ACAN_T4::can1.receive (CanTelemetry)) {
  if (CanTelemetry.id == 0x10a) {
    Errorcode = CanTelemetry.data[0] ;
    TeensyTemp = CanTelemetry.data[1] ;
    R2Dbutton = CanTelemetry.data[2] ;
    ECO_mode = CanTelemetry.data[3] ;
    Normal_Mode = CanTelemetry.data[4] ;
    Endurance_mode = CanTelemetry.data[5] ;
    HV = CanTelemetry.data[6] ;
    BrakeState = CanTelemetry.data[7] ;
  } else if (BMScontrollerRight.id == 0x0CF11E7F) {
    ThrottleSignalRight = BMScontrollerRight.data[0] ;
    ControllerTempRight = (BMScontrollerRight.data[1] - 40) ;
    MotorTempCanR = (BMScontrollerRight.data[2] - 30) ;
    ECO_mode = BMScontrollerRight.data[3] ;
    HV = BMScontrollerRight.data[4] ;
    Endurance_mode = BMScontrollerRight.data[5] ;
  } else if (BMScontrollerLeft.id == 0x0CF11E7F) {
    ThrottleSignalLeft = BMScontrollerLeft.data[0] ;
    ControllerTempLeft = (BMScontrollerLeft.data[1] - 40) ;
    MotorTempCanL = BMScontrollerLeft.data[2] ;
    ECO_mode = BMScontrollerLeft.data[3] ;
    HVbutton = BMScontrollerLeft.data[4] ;
    Endurance_mode = BMScontrollerLeft.data[5] ;
  } else if (BMSMotorLeft.id == 0x0CF11E05) {
    RPMLoraLeft = (BMSMotorLeft.data[0] * 256 - RPMLoraLSBleft) ; //RPM 0-6000
    RPMLoraLSBleft = BMSMotorLeft.data[1] ;
  }
}

```

Рисунок 48 – Пример обработки CAN-сообщения

Получение данных рекуперации не сложное, простое считывание аналогового сигнала с фильтрацией помех

```

Flap = analogRead (Recup) ;
filteredVal += (Flap - filteredVal) * 0.1;
if (filteredVal > max2) { max2 = filteredVal; } ;
RecupDataRaw = map(filteredVal, 780, max2, -125, 1223) ;
RecupData = constrain(RecupDataRaw, 0, 1023) ;

```

Рисунок 49 – Обработка датчика рекуперации

Для передачи данных использовалась библиотека SoftwareSerial [24], которая позволяет реализовать последовательный интерфейс на любых цифровых выводах Ардуино с помощью программных средств, дублирующих функциональность UART.

```

int ThrottleLora = map (ThrottleSignalRight, 0, 255, 0, 100) ;
if (LORASerial.available()){
  LORASerial.print(ThrottleLora) ;
  LORASerial.print(";") ;
  LORASerial.print(RPMLoraLeft) ;
  LORASerial.print(";") ;
  LORASerial.print(RPMLoraRight) ;
  LORASerial.print(";") ;
  LORASerial.print (BrakeState) ;
  LORASerial.print(";") ;
  LORASerial.print (MotorTempCanL) ;
  LORASerial.print(";") ;
  LORASerial.print (MotorTempCanR) ;
  LORASerial.print(";") ;
  LORASerial.print ("50") ;
  LORASerial.print(";") ;
  LORASerial.print (CurrentLoraL) ;
  LORASerial.print(";") ;
  LORASerial.print (CurrentLoraR) ;
  LORASerial.print(";") ;
  LORASerial.print (VoltageLora) ;
  LORASerial.print(";") ;
  LORASerial.print (ControllerTempLeft) ;
  LORASerial.print(";") ;
  LORASerial.print (ControllerTempRight) ;
  LORASerial.print(";") ;
  LORASerial.print (R2Dbutton) ;
  LORASerial.print(";") ;
  LORASerial.print (STBLora) ;

```

Рисунок 50 – Отправка данных с помощью LoRa

Данные с 9-ти позиционного IMU-датчика обрабатываются с помощью библиотеки с таким же названием - MPU9250 [21].

```

if (mpu.update()) {
  static uint32_t prev_ms = millis();
  if (millis() > prev_ms + 25) {
    print_roll_pitch_yaw();
    prev_ms = millis();
  }
}

void print_roll_pitch_yaw() {
  LORASerial.print("Yaw, Pitch, Roll: ");
  LORASerial.print(mpu.getYaw(), 2);
  LORASerial.print(", ");
  LORASerial.print(mpu.getPitch(), 2);
  LORASerial.print(", ");
  LORASerial.println(mpu.getRoll(), 2);
}

```

Рисунок 51 – Обработка данных IMU-датчика

Общее устройство программы представлено на рисунке 42. Полный код программы представлен в приложении Б.

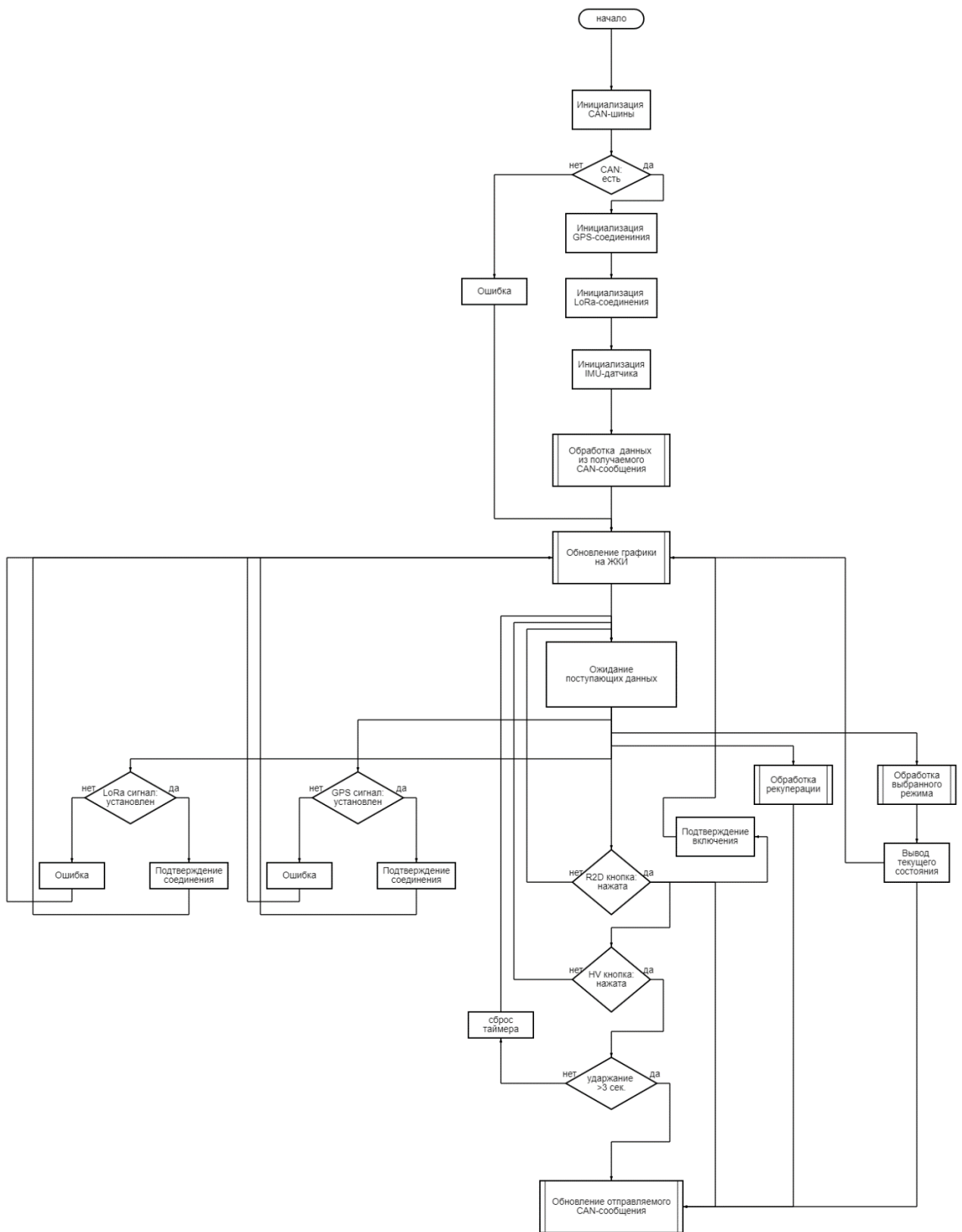


Рисунок 52 – Общее устройство программы

Вывод

Написание управляющей программы является, пожалуй, самым сложным и ответственным этапом среди всех при проектировании устройства. Именно этот этап является основой, от которой зависит качество выполненной работы, потому как эффективность использования выбранных компонентов напрямую зависит от оптимизации и надежности управляющей программы. Просчеты чреваты медленной работой, даже не смотря на завидные характеристики микроконтроллера, скорость передачи CAN-шины и качество остальных подобранных компонентов.

Объем получившейся программы серьезно превысил возможности просты микроконтроллеров, полностью подтвердив опасения, связанные с использованием микроконтроллера Arduino, высказанные в самом начале работы на этапе подбора компонентов.

5 Разработка корпуса устройства

Корпус предназначен для защиты от внешних механических воздействий и влаги. Сама плата крепится напрямую к крепежным отверстиям ЖКИ на винтах, этому способствуют ее размеры и расположение отверстий. Корпус выполняет роль защитного чехла с задней стороны, так как с передней плата встраивается в приборную панель гоночного болида.

Ввиду всего этого было решено сделать простую 3Д модель, которую можно было бы использовать для печати на 3Д принтере.

САПР для работы была выбрана самая простая, понятная и, что самое главное – бесплатная. Сервис Tinkercad [26] позволяет, помимо моделирования схем для ардуино, производить 3Д моделирование и сохранять полученные модели.

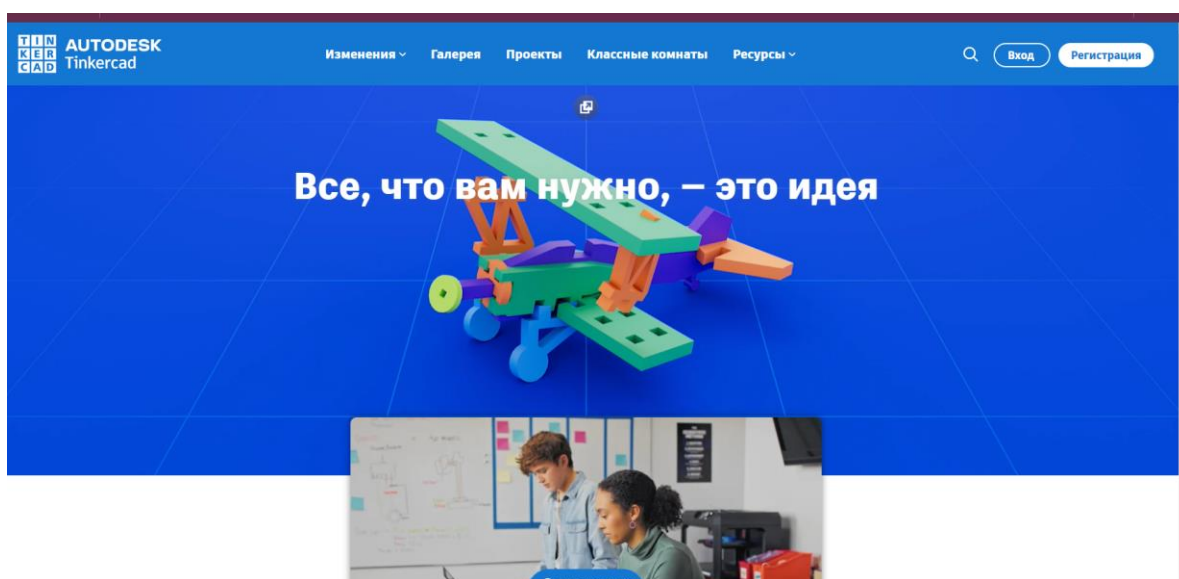


Рисунок 53 – Официальный сайт Tinkercad

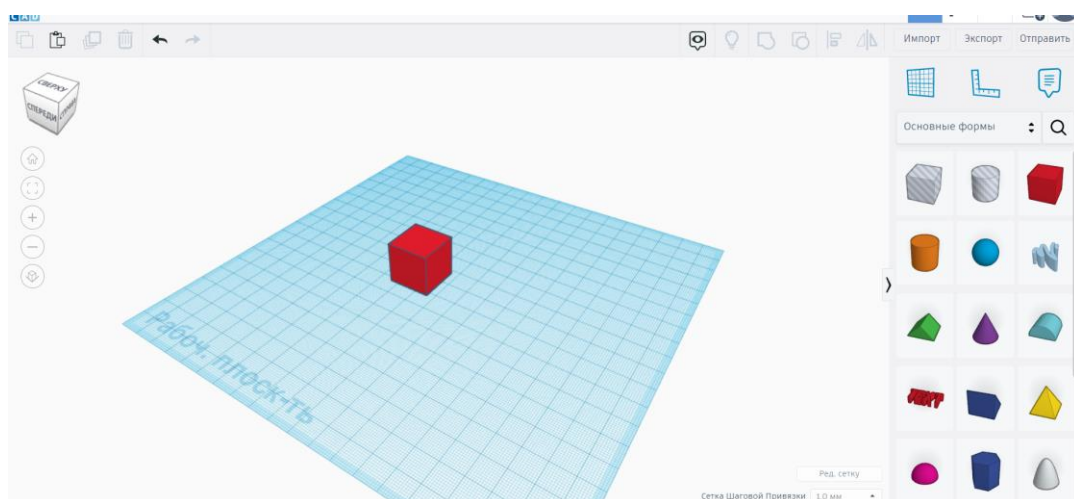


Рисунок 54 – Рабочая область 3Д-редактора

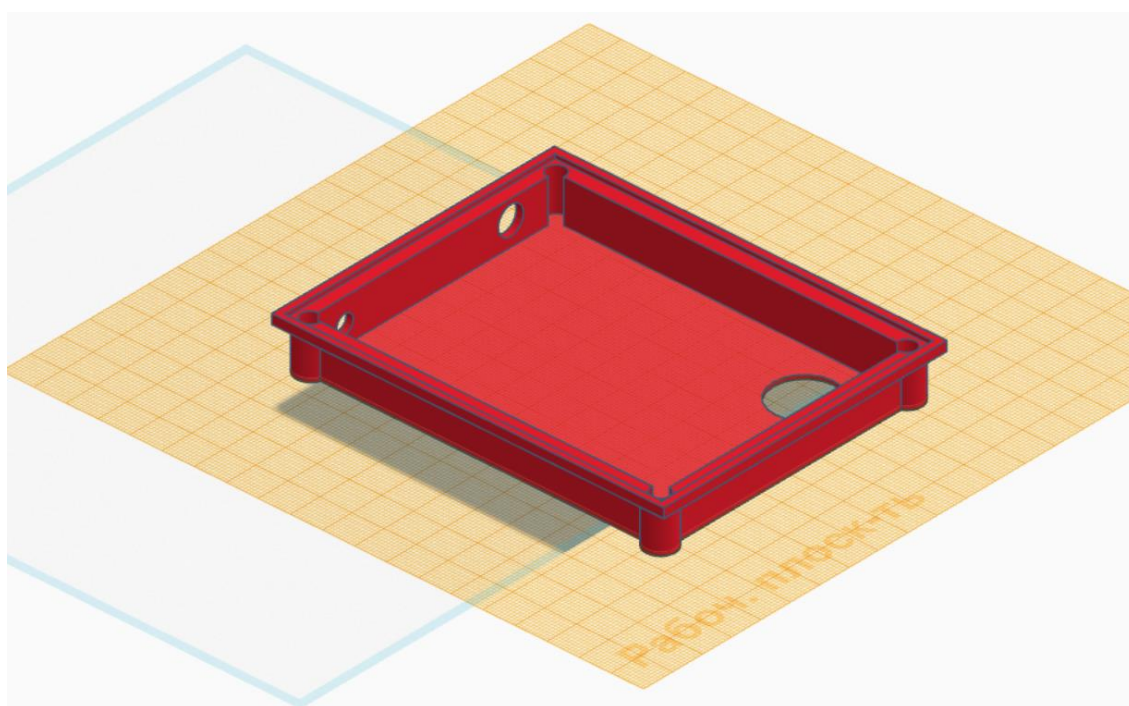


Рисунок 55 – Готовая 3Д модель корпуса

При разработке корпуса была учтена необходимость вывода проводов, для чего были сделаны специальные отверстия. Крепление устройства предполагается с помощью винтов, для чего предусмотрено место для установки специальных металлических резьбовых втулок для пластиковых деталей. Их установка производится путем нагрева и вплавления в пластиковую деталь.



Рисунок 56 – Резьбовая втулка, установленная в пластиковую деталь

Далее требуется лишь скачать модель в необходимом формате для последующей печати на 3Д принтере, для скачивания доступны форматы .stl и .obj

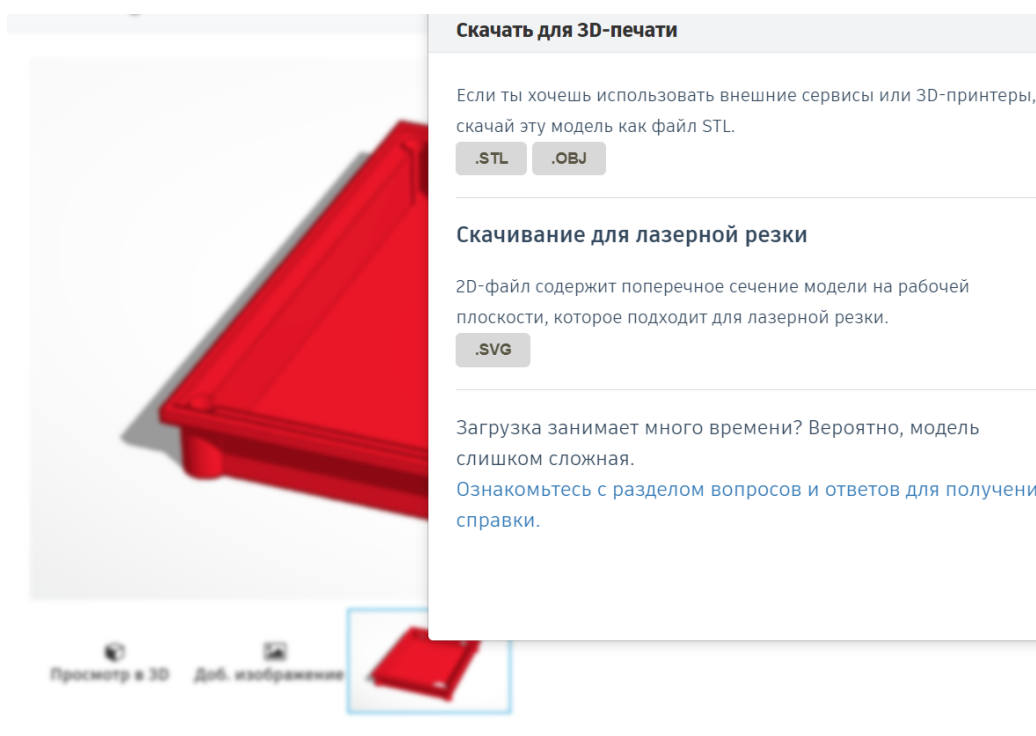


Рисунок 57 – Область выбора формата перед сохранением 3Д модели

Вывод

Так же, как и с производством печатной платы, при разработке и производстве корпуса устройства было применено обширное использование современных технологий, которое позволило серьезно упростить весь процесс работы, от разработки и проектирования, до непосредственного создания корпуса устройства.

Само по себе распространение 3Д печати, в частности бюджетных и доступных 3Д принтеров, позволяет осуществить печать детали по заказу практически в любом городе, в том числе в Тольятти.

В рамках выполнения данной работы, был использован профессиональный 3Д принтер от Нидерландской компании Ultimaker, ввиду его наличия у команды Togliatty Racing Team в Центре машиностроения ТГУ

Заключение

Результатом проведенных анализа, разработки и проектировки является собранное, полностью функционирующее готовое устройство – приборная индикационная панель, предусмотренная для установки на электрический гоночный болид. Помимо этого, важным отдельным личным результатом является приобретение и улучшение уже имеющихся навыков по разработке и проектирования сложных печатных плат, программированию микроконтроллеров и сборке устройств на печатных платах.

Благодаря правильной постановке задачи, основанной на анализе существующих предложений, а также на требовании по соответствию нормам и положениям регламента соревнований Formula Student Electric, имеется возможность избежать каких-либо вероятных проблем при прохождении технической инспекции во время соревнований как в России, так и за рубежом. Консультация и принятие во внимания пожеланий членов команды Togliatty Racing Team, позволило полностью реализовать все необходимые для команды функции, соответствуя при этом заявленным требованиям.

Данная индикационная панель предполагается для установки на строящийся в данный момент электрический гоночный болид первого поколения команды Togliatty Racing Team, ближайшее полноценное использование которого должно будет осуществляться в рамках ближайших международных соревнований Formula Student Electric в Москве.

По итогам данных соревнований возможно появление дополнительных требований и пожеланий, которые будут учтены при разработке последующих устройств, а также модернизации и доработках уже имеющегося, с целью достижения оптимального результата.

Список используемой литературы

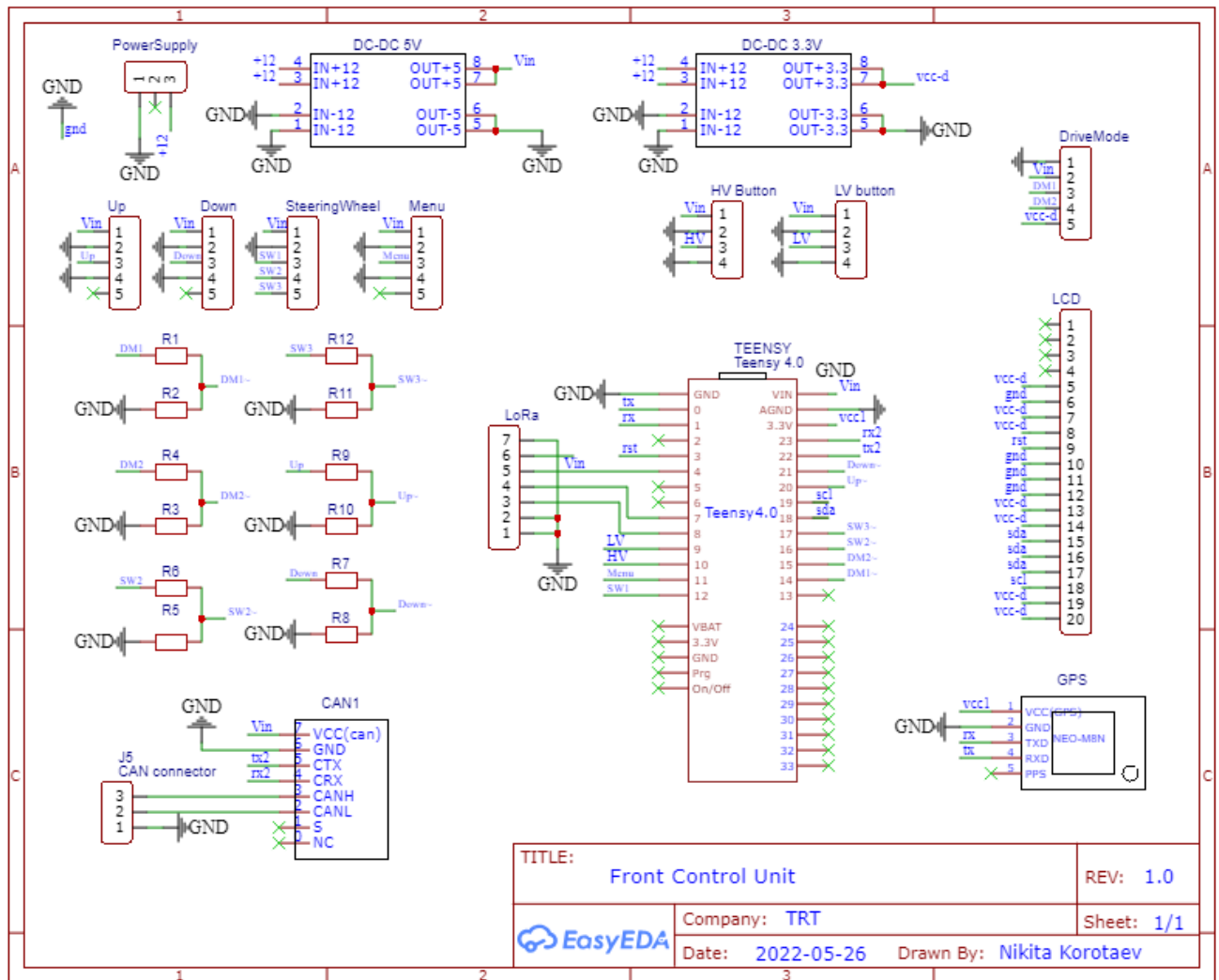
1. Формула Е [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Формула_Е (дата обращения: 15.05.2022)
2. Электромобиль [Электронный ресурс]. URL: <https://ru.wikipedia.org/wiki/Электромобиль> (дата обращения: 15.05.2022)
3. Formula Student [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Formula_Student (дата обращения: 15.05.2022)
4. 16mm Metal LED indicator Light press push button switches reset Momentary/Latching self-lock Car power ring 12V 3V Start Button [Электронный ресурс]. URL: <https://aliexpress.ru/item/33060639491.html> (дата обращения: 04.06.2022)
5. ACAN T4 [Электронный ресурс]. URL: <https://github.com/pierremolinaro/acan-t4> (дата обращения: 06.06.2022)
6. ATGM336H GPS Module [Электронный ресурс]. URL: <https://www.tinytronics.nl/shop/nl/communicatie-en-signalen/draadloos/gps/modules/atgm336h-gps-module> (дата обращения: 04.06.2022)
7. Andromeda EVIC 7" Display [Электронный ресурс]. URL: <https://stealthev.com/product/andromeda-evic-7-display/> (дата обращения: 04.06.2022)
8. Arduino Mega 2560 [Электронный ресурс]. URL: <http://arduino.ru/Hardware/ArduinoBoardMega2560> (дата обращения: 04.06.2022)
9. Arduino IDE [Электронный ресурс]. URL: <https://www.arduino.cc/en/software> (дата обращения: 06.06.2022)

10. Block Diagram Redactor [Электронный ресурс]. URL: <https://programforyou.ru/block-diagram-redactor> (дата обращения: 06.06.2022)
11. DipTrace [Электронный ресурс]. URL: www.diptrace.com (дата обращения: 06.06.2022)
12. E32-TTL-1W (433T30D) [Электронный ресурс]. URL: <https://www.platan.ru/cgi-bin/qwery.pl/id=2014435739> (дата обращения: 04.06.2022)
13. ESP32-WROOM [Электронный ресурс]. URL: <https://www.electronicclinic.com/esp32-wroom-32d-pinout-features-and-specifications/> (дата обращения: 04.06.2022)
14. EV (Electric Vehicle) Instrument Cluster / Development Services [Электронный ресурс]. URL: <https://www.indiamart.com/unizen-tech-limited/bike.html> (дата обращения: 15.05.2022)
15. EasyEDA [Электронный ресурс]. URL: <https://easyeda.com> (дата обращения: 06.06.2022)
16. High Speed CAN transceiver TJA1051 [Электронный ресурс]. URL: <https://www.nxp.com/products/interfaces/can-transceivers/can-with-flexible-data-rate/high-speed-can-transceiver:TJA1051> (дата обращения: 04.06.2022)
17. JLCPCB [Электронный ресурс]. URL: <https://jlcpcb.com> (дата обращения: 06.06.2022)
18. JLX256160G-910-PL [Электронный ресурс]. URL: <http://www.jlxlcd.cn/html/zh-detail-857.html> (дата обращения: 04.06.2022)
19. LoRa [Электронный ресурс]. URL: <https://ru.wikipedia.org/wiki/LoRa> (дата обращения: 04.06.2022)
20. MPU 9250 [Электронный ресурс]. URL: <https://invensense.tdk.com/products/motion-tracking/9-axis/mpu-9250/> (дата обращения: 04.06.2022)

21. MPU9250 [Электронный ресурс]. URL: <https://www.arduino.cc/reference/en/libraries/mpu9250/> (дата обращения: 08.06.2022)
22. PlatformIO [Электронный ресурс]. URL: <https://platformio.org> (дата обращения: 06.06.2022)
23. Raspberry Pi 4 Tech Specs [Электронный ресурс]. URL: <https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/> (дата обращения: 04.06.2022)
24. SoftwareSerial Library [Электронный ресурс]. URL: <https://docs.arduino.cc/learn/built-in-libraries/software-serial> (дата обращения: 08.06.2022)
25. Teensy® 4.0 Development Board [Электронный ресурс]. URL: <https://www.pjrc.com/store/teensy40.html> (дата обращения: 04.06.2022)
26. Tinkercad [Электронный ресурс]. URL: <https://www.tinkercad.com> (дата обращения: 08.06.2022)
27. U8glib [Электронный ресурс]. URL: <https://github.com/olikraus/u8g2> (дата обращения: 06.06.2022)
28. Visual Studio Code [Электронный ресурс]. URL: <https://code.visualstudio.com> (дата обращения: 06.06.2022)

Приложение А

Схема соединений в системе EasyEDA



Приложение Б

Листинг управляющей программы

```
#include <Arduino.h>

#include <U8g2lib.h>

#include <ACAN_T4.h>

#include <SoftwareSerial.h>

#include <MPU9250.h>

#ifdef U8X8_HAVE_HW_SPI

#include <SPI.h>

#endif

#ifdef U8X8_HAVE_HW_I2C

#include <Wire.h>

#endif

U8G2_ST75256_JLX256160_F_SW_I2C u8g2 (U8G2_R0, 19, 18, 3) ;

static uint32_t gSendDate = 0 ; // Smth important for CAN

SoftwareSerial LORAserial(7, 8) ; // RX, TX

static const int RXPin = 0, TXPin = 1 ; //GPS RX, TX

static const int Encoder = 14 ; //Drive Mod selector

const char DEGREE_SYMBOL[] = { 0xB0, '\0' };

const int StartButton = 9 ; //StartStop Button
```

Продолжение приложения Б

```
const int HighVoltageButton = 10 ; //High Voltage Button on the dashboard
const int ledPin = 13 ;
const int Recup = 15 ; //Recupiration
unsigned long ButtonMillis ;
unsigned long HVmillis ;
unsigned long EncoderMillis ;
unsigned long GoSendOnAt ;
unsigned long HVSendOnAt ;
unsigned long EncSendOnAt ;
unsigned long SendOnDelay = 3000 ;
unsigned long SendOffDelay = 100 ;
unsigned long HVondelay = 1000 ;
unsigned long HVoffdelay = 100 ;
unsigned long EnOnDelay = 1000 ;
unsigned long EnOffDellay = 100 ;
unsigned long CurrentMillis ;
unsigned long timing ;
unsigned long timingGPS ;
unsigned long LoraSignalIcon1 ;
unsigned long LoraSignalIcon2 ;
unsigned long LoraSignalIcon3 ;
```


Продолжение приложения Б

```
float filteredVal = 0 ;  
int serialCanData = 0;  
int Flap = 0 ;  
int max2 = 0 ;  
int RecupDataRaw = 0 ;  
int RecupData = 0 ;  
int EncoderValue = 0 ;  
int EncoderValueRaw = 0 ;  
int MotorTempCanR = 0 ;  
int MotorTempCanL = 0 ;  
int ThrottleSignalLeft = 0 ;  
int ThrottleSignalRight = 0 ;  
int ControllerTempRight = 0 ;  
int ControllerTempLeft = 0 ;  
int RMPrawL = 0 ;  
int RMPrawR = 0 ;  
int RPMLoraLeft = 0 ;  
int RPMLoraRight = 0 ;  
int RPMLoraLSBleft = 0 ;  
int RPMLoraLSBrigt = 0 ;  
int TeensyTemp = 0 ;
```

Продолжение приложения Б

```
int BrakeState = 0 ;  
  
int CurrentLoraL = 0 ;  
  
int CurrentLoraR = 0 ;  
  
int CurrentLSBleft = 0 ;  
  
int CurrentLSBright = 0 ;  
  
int VoltageLora = 0 ;  
  
int VoltageLSB = 0 ;  
  
int Errorcode = 0 ;  
  
bool HV = 0 ;  
  
bool HVSendOk = 0 ;  
  
bool SendOk = 0 ;  
  
bool GoReady = false ;  
  
bool HVready = false ;  
  
bool GoState = false ;  
  
bool CanR2D = false ;  
  
bool HVCan = false ;  
  
bool HVState = false ;  
  
bool ButtonZero = false ;  
  
bool DashCan = false ;  
  
bool EncoderDash = false ;  
  
bool GPSok = false ;
```

Продолжение приложения Б

```
bool LoraOK = false ;

bool HVbutton = false ;

bool R2Dbutton = false ;

bool SteeringButton = false ;

bool STBlora = false ;

bool STBloreReady = false ;

bool STBState = false ;

bool ECO_mode = false ;

bool Normal_Mode = false ;

bool Endurance_mode = false ;

MPU9250 mpu;

void setup()

{

    u8g2.begin() ;

    u8g2.setFont(u8g2_font_VCR_OSD_tf) ;

    u8g2.setFontPosTop();

    u8g2.setContrast(0x2c * 4) ; // put your setup code here, to run once:

    pinMode (9, INPUT_PULLUP) ;

    pinMode (10,INPUT_PULLUP) ;

    pinMode (15, INPUT_PULLUP) ;

    pinMode (11, INPUT_PULLUP);
```

Продолжение приложения Б

```
pinMode(ledPin, OUTPUT) ;

digitalWrite(ledPin, LOW);

ACAN_T4_Settings settings (250 * 1000) ; // 250 kbit/s

settings.mLoopBackMode = false ; //never set true

settings.mSelfReceptionMode = false ; //never set true

const uint32_t errorCode = ACAN_T4::can1.begin (settings) ; //can1 or can2 or
can3

if (0 == errorCode) {

    Serial.println ("can1 ok") ;

    DashCan = true;

    Serial.flush(); //Waiting for all data to be transferred to the serial

} else {

    Serial.print ("Error can1: 0x") ;

    Serial.println (errorCode, HEX) ;

    DashCan = false; }

Serial.begin(115200) ;

LORAserial.begin(9600) ;
```

Продолжение приложения Б

```
if (!mpu.setup(0x68)) {  
    while (1) {  
        Serial.println("MPU connection failed. Please check your connection with  
`connection_check` example.");  
        delay(5000);  
    }  
}  
  
void loop()  
{  
    int BatteryChargeRaw = random(0, 365) ;           //Battery graphic  
    int BatteryChargeBar = map (BatteryChargeRaw, 0, 365, 0, 252) ;    //Battery  
    graphic  
    int BatteryChargePercent = map (BatteryChargeRaw, 0, 365, 0, 99) ; //Battery %  
  
    int PowerRaw = random (0, 1600) ;                 //Power numbrers  
    int PowerGraph = map (PowerRaw, 0, 1600, 0, 130 ) ;           //Power  
    graphic  
  
    int BatteryTempRaw = random(0,60) ;              //Battery temperature  
    raw
```

Продолжение приложения Б

```
int BatteryTempNumbers = map(BatteryTempRaw, 0, 60, 0, 75); //Battery
temperature numbers

int BatteryTempBar = map(BatteryTempRaw, 0, 60, 0, 115); //Battery
temperature graph

int MotorTempRawMid = (MotorTempCanR + MotorTempCanL) / 2 ;
//Motor temp numbers right

int MotorTempNumbers = map (MotorTempRawMid, 0, 1943, 0, 60) ;
//Motor temp numbers Настроить МАП после установки на болид

int MotorTempGraph = map (MotorTempRawMid, 0, 1943, 0, 115) ;
//Motor temp graph Настроить МАП после установки на болид

int InvertorTempRaw = random (0, 162) ; //Invertor numbers

int InvertorTempNumbers = map (InvertorTempRaw, 0, 162, 0, 70) ;
//Пересчитать для МАР-а после установки на болид

int InvertorTempGraph = map (InvertorTempRaw, 0, 162, 0, 130) ;

if (millis() - timing > 100) // Pause number
{
    timing = millis() ;
    u8g2.clearBuffer() ; // clear the internal memory

    printBatteryCharge(BatteryChargeBar);
```

Продолжение приложения Б

```
printPowerGraph (PowerGraph);

printBatteryTemp(BatteryTempBar);

printMotorTemp(MotorTempGraph);

printInvertorTemp(InvertorTempGraph);

printBoxes();

printCanStatus();

printDriveModeSelected();

printText(BatteryChargePercent, BatteryTempNumbers, MotorTempNumbers,
InvertorTempNumbers);

changeColor(BatteryTempBar, MotorTempGraph, InvertorTempGraph);

if (!GPSok) {
    printGPSicon(); }

if (!LoraOK){
    printLORAicon();}

if (!HVbutton) {
    printHV(); }

if (R2Dbutton) {
    printR2D();}

u8g2.sendBuffer() ; // transfer internal memory to the display
}
```

Продолжение приложения Б

```
bool temp = digitalRead(StartButton) ;

bool HVbutton = digitalRead(HighVoltageButton) ;

bool STB = digitalRead (SteeringButton); //STB means STeering Button

//EncoderValueRaw = analogRead (Encoder) ;

CurrentMillis = millis() ;

if (!temp) {

    ButtonMillis = CurrentMillis;

    GoReady = true;}

if(GoReady) {

    if ((CurrentMillis - ButtonMillis) >= SendOnDelay) {

        CanR2D = true ;

        GoState = true ;

        GoSendOnAt = CurrentMillis ;

        GoReady = false ;

        SendOk = true ;

        //Serial.print("button");

    }

}
```


Продолжение приложения Б

```
if (GoState) {  
    if((CurrentMillis - GoSendOnAt) >= SendOffDelay) {  
        GoState = false ;  
        CanR2D = false ;  
    }  
}
```

```
if(HVbutton) {  
    HVmillis = CurrentMillis;  
    HVready = true;}  
}
```

```
if(HVready) {  
    if ((CurrentMillis - HVmillis) >= HVondelay) {  
        HVCan = true ;  
        HVState = true ;  
        HVSendOnAt = CurrentMillis ;  
        HVready = false ;  
        HVSendOk = true ;  
        Serial.print("HVbutton"); }  
}
```

Продолжение приложения Б

```
if (HVState) {  
    if((CurrentMillis - HVSendOnAt) >= HVoffdelay) {  
        HVState = false ;  
        HVCan = false ;  
    }  
}  
  
if (!STB){  
    STBlora = true ;  
}  
  
Flap = analogRead (Recup) ;  
filteredVal += (Flap - filteredVal) * 0.1;  
if (filteredVal > max2) { max2 = filteredVal; } ;  
RecupDataRaw = map(filteredVal, 780, max2, -125, 1223) ;  
RecupData = constrain(RecupDataRaw, 0, 1023) ;  
  
//EncoderValue = 1 ;  
  
if(SendOk) {  
    CANMessage StartStop;
```

Продолжение приложения Б

```
if (gSendDate <= millis ()) {  
    StartStop.id = 0xBB41 ;  
    StartStop.len = 1 ;  
    StartStop.data [0] = CanR2D ; //StartStop Button  
    const bool ok = ACAN_T4::can1.tryToSend (StartStop) ; //can1 or can2 or  
can3  
    if (ok) {  
        gSendDate += 100 ; // every 250ms send can message  
        Serial.print("CAN");  
        SendOk = false ;  
    }  
}  
}  
}  
  
if(HVSendOk) {  
    CANMessage HighVoltage;  
    if (gSendDate <= millis ()) {  
        HighVoltage.id = 0xBB42 ;  
        HighVoltage.len = 1 ;  
        HighVoltage.data [0] = HVCan ; // High Voltage Data  
        const bool ok = ACAN_T4::can1.tryToSend (HighVoltage) ; //can1 or can2 or  
can3
```

Продолжение приложения Б

```
if (ok) {  
    gSendDate += 100 ; // every 250ms send can message  
    HVSendOk = false ;  
}  
}  
}
```

CANMessage EncoderRecup;

```
if (gSendDate <= millis ()) {
```

```
    EncoderRecup.id = 0xBB43 ;
```

```
    EncoderRecup.len = 2 ;
```

```
    EncoderRecup.data [0] = EncoderValue ; //Encoder Data
```

```
    EncoderRecup.data [1] = RecupData ; //Recupirations
```

```
    const bool ok = ACAN_T4::can1.tryToSend (EncoderRecup) ; //can1 or can2 or  
can3
```

```
    if (ok) {
```

```
        gSendDate += 100 ; // every 250ms send can message
```

```
    }
```

```
}
```

CANMessage CanTelemetry ;

Продолжение приложения Б

```
CANMessage BMScontrollerRight ;  
  
CANMessage BMScontrollerLeft ;  
  
CANMessage BMSMotorLeft ;  
  
CANMessage BMSMotorRight ;  
  
if (ACAN_T4::can1.receive (CanTelemetry)) {  
  
    if (CanTelemetry.id == 0x10a) {  
  
        Errorcode = CanTelemetry.data[0] ;  
  
        TeensyTemp = CanTelemetry.data[1] ;  
  
        R2Dbutton = CanTelemetry.data[2] ;  
  
        ECO_mode = CanTelemetry.data[3] ;  
  
        Normal_Mode = CanTelemetry.data[4] ;  
  
        Endurance_mode = CanTelemetry.data[5] ;  
  
        HV = CanTelemetry.data[6] ;  
  
        BrakeState = CanTelemetry.data[7] ;  
  
    } else if (BMScontrollerRight.id == 0x0CF11E7F) {  
  
        ThrottleSignalRight = BMScontrollerRight.data[0] ;  
  
        ControllerTempRight = (BMScontrollerRight.data[1] - 40) ;  
  
        MotorTempCanR = (BMScontrollerRight.data[2] - 30) ;  
  
        ECO_mode = BMScontrollerRight.data[3] ;  
  
        HV = BMScontrollerRight.data[4] ;  
  
        Endurance_mode = BMScontrollerRight.data[5] ;  
  
    }  
  
}
```

Продолжение приложения Б

```
} else if (BMScontrollerLeft.id == 0x0CF11E7F) {  
    ThrottleSignalLeft = BMScontrollerLeft.data[0] ;  
    ControllerTempLeft = (BMScontrollerLeft.data[1] - 40) ;  
    MotorTempCanL = BMScontrollerLeft.data[2] ;  
    ECO_mode = BMScontrollerLeft.data[3] ;  
    HVbutton = BMScontrollerLeft.data[4] ;  
    Endurance_mode = BMScontrollerLeft.data[5] ;  
} else if (BMSMotorLeft.id == 0x0CF11E05) {  
    RPMLoraLeft = (BMSMotorLeft.data[0] * 256 - RPMLoraLSBleft) ; //RPM  
0-6000  
    RPMLoraLSBleft = BMSMotorLeft.data[1] ;  
    CurrentLoraL = ((BMSMotorLeft.data[2]*256 - CurrentLSBleft) / 10) ;  
    CurrentLSBleft = BMSMotorLeft.data[3] ;  
    VoltageLora = ((BMSMotorLeft.data[4] *256 - VoltageLSB) / 10) ;  
    VoltageLSB = BMSMotorLeft.data[5] ;  
    HVbutton = BMSMotorLeft.data[6] ;  
} else if (BMSMotorRight.id == 0x0CF11F7F) {  
    RPMLoraRight = (BMSMotorLeft.data[0] *256 - RPMLoraLSBright) ;  
    RPMLoraLSBright = BMSMotorLeft.data[1] ;  
    CurrentLoraR = ((BMSMotorLeft.data[2] *256 - CurrentLSBright) / 10) ;  
}  
}
```

Продолжение приложения Б

```
if (mpu.update()) {  
    static uint32_t prev_ms = millis();  
  
    if (millis() > prev_ms + 25) {  
        print_roll_pitch_yaw();  
        prev_ms = millis();  
    }  
}  
  
int ThrottleLora = map (ThrottleSignalRight, 0, 255, 0, 100) ;  
  
if (LORASerial.available()){  
    LORASerial.print(ThrottleLora) ;  
  
    LORASerial.print(";") ;  
  
    LORASerial.print(RPMLoraLeft) ;  
  
    LORASerial.print(";") ;  
  
    LORASerial.print(RPMLoraRight) ;  
  
    LORASerial.print(";") ;  
  
    LORASerial.print (BrakeState) ;  
  
    LORASerial.print(";") ;  
  
    LORASerial.print (MotorTempCanL) ;  
  
    LORASerial.print(";") ;  
  
    LORASerial.print (MotorTempCanR) ;
```

Продолжение приложения Б

```
LORAserial.print(";") ;  
  
LORAserial.print ("50") ;  
  
LORAserial.print(";") ;  
  
LORAserial.print (CurrentLoraL) ;  
  
LORAserial.print(";") ;  
  
LORAserial.print (CurrentLoraR) ;  
  
LORAserial.print(";") ;  
  
LORAserial.print (VoltageLora) ;  
  
LORAserial.print(";") ;  
  
LORAserial.print (ControllerTempLeft) ;  
  
LORAserial.print(";") ;  
  
LORAserial.print (ControllerTempRight) ;  
  
LORAserial.print(";") ;  
  
LORAserial.print (R2Dbutton) ;  
  
LORAserial.print(";") ;  
  
LORAserial.print (STBlora) ;  
  
LORAserial.print(";") ;  
  
LORAserial.print (HV) ;  
  
LORAserial.print(";") ;  
  
LORAserial.print (Normal_Mode) ;  
  
LORAserial.print(";") ;
```


Продолжение приложения Б

```
LORASerial.print (Endurance_mode) ;

LORASerial.print(";") ;

LORASerial.print (ECO_mode) ;

LORASerial.print(";") ;

LORASerial.print (Errorcode) ;

LORASerial.println("");

    }

}

void printBatteryCharge(int &BatteryChargeBar)

{

    u8g2.drawFrame(0, 18, 256, 25);           //Charge Bar box

    u8g2.drawRBox(2, 20, BatteryChargeBar, 21, 0) ; //Charge bar graphic

}

void printPowerGraph (int &PowerGraph)

{

    u8g2.drawFrame(0, 62, 134, 25) ;           //Power bar box

    u8g2.drawRBox(2, 64, PowerGraph, 21, 0) ; //Power bargraphic

}

void printBatteryTemp(int &BatteryTempBar)

{

    u8g2.drawFrame(137, 62, 119, 25) ;           //Battery temp bar box
```

Продолжение приложения Б

```
u8g2.drawRBox (139, 64, BatteryTempBar, 21, 0 ) ; //Battery bar graphic
}

void printMotorTemp(int &MotorTempGraph)
{
    u8g2.drawFrame(137, 106, 119, 25) ; //Motor temp bar box
    u8g2.drawRBox (139, 108, MotorTempGraph, 21, 0 ) ; //Motor bar graphic
}

void printInvertorTemp(int &InvertorTempGraph)
{
    u8g2.drawFrame(0, 106, 134, 25); //Invertor temp bar box
    u8g2.drawRBox(2, 108, InvertorTempGraph, 21, 0) ; //Invertor temp bargraphic
}

void printBoxes()
{
    u8g2.drawFrame(0, 132, 50, 28) ; //NORM
    u8g2.drawFrame(49, 132, 50, 28) ; //ENDR
    u8g2.drawFrame(98, 132, 38, 28) ; //ECO
    u8g2.drawFrame(135, 132, 31, 28) ; //High Voltage
    u8g2.drawFrame(165, 132, 31, 28) ; //Ready 2 Drive
    u8g2.drawFrame(195, 132, 31, 28) ; //GPS
```

Продолжение приложения Б

```
u8g2.drawCircle(210, 168, 17); //GPS Earth
```

```
u8g2.drawFrame(225, 132, 31, 28); //LoRa
```

```
}
```

```
void printGPSicon()
```

```
{
```

```
if (millis() - timingGPS > 1500){
```

```
    u8g2.drawDisc(205, 142, 4);
```

```
    u8g2.drawLine(202, 139, 220, 135);
```

```
    u8g2.drawLine(202, 142, 222, 138);
```

```
    u8g2.drawLine(202, 144, 222, 143);
```

```
    u8g2.drawLine(202, 145, 220, 147);
```

```
}
```

```
if (millis() - timingGPS > 3000){
```

```
    timingGPS = millis();
```

```
}
```

```
}
```

```
void printLORAicon ()
```

```
{
```

```
u8g2.drawRBox(240, 146, 3, 12,0);
```

```
u8g2.drawDisc(241, 144, 3);
```

Продолжение приложения Б

```
if (millis() - LoraSignalIcon1 > 999){  
    u8g2.drawEllipse(238, 144, 3, 5, U8G2_DRAW_LOWER_LEFT) ;  
    u8g2.drawEllipse(238, 144, 3, 5, U8G2_DRAW_UPPER_LEFT) ;  
    u8g2.drawEllipse(244, 144, 3, 5, U8G2_DRAW_UPPER_RIGHT) ;  
    u8g2.drawEllipse(244, 144, 3, 5, U8G2_DRAW_LOWER_RIGHT) ;  
}  
  
if (millis() - LoraSignalIcon1 > 1222){  
    u8g2.drawEllipse(238, 144, 5, 7, U8G2_DRAW_LOWER_LEFT) ;  
    u8g2.drawEllipse(238, 144, 5, 7, U8G2_DRAW_UPPER_LEFT) ;  
    u8g2.drawEllipse(244, 144, 5, 7, U8G2_DRAW_UPPER_RIGHT) ;  
    u8g2.drawEllipse(244, 144, 5, 7, U8G2_DRAW_LOWER_RIGHT) ;  
}  
  
if (millis() - LoraSignalIcon1 > 1555){  
    u8g2.drawEllipse(238, 144, 7, 9, U8G2_DRAW_LOWER_LEFT) ;  
    u8g2.drawEllipse(238, 144, 7, 9, U8G2_DRAW_UPPER_LEFT) ;  
    u8g2.drawEllipse(244, 144, 7, 9, U8G2_DRAW_UPPER_RIGHT) ;  
    u8g2.drawEllipse(244, 144, 7, 9, U8G2_DRAW_LOWER_RIGHT) ;  
}  
  
if (millis() - LoraSignalIcon1 > 2500){  
    LoraSignalIcon1 = millis();  
}
```

Продолжение приложения Б

```
}  
  
void printText(int &BatteryChargePercent, int &BatteryTempNumbers, int  
&MotorTempNumbers, int &InvertorTempNumbers)  
{  
  
    u8g2.setCursor(217, 2);      // Battery % position  
  
    u8g2.print(BatteryChargePercent); // Battery %  
  
    u8g2.drawStr(0, 2, "Charge");  
  
    u8g2.drawStr(243, 2, "%");  
  
    u8g2.drawStr(136, 45, "Battery C");  
  
    u8g2.setFont(u8g2_font_10x20_tf); //Changing font for the °  
  
    u8g2.drawUTF8(223,45, DEGREE_SYMBOL);  
  
    u8g2.setFont(u8g2_font_VCR_OSD_tf); //Changing it back  
  
    u8g2.setFontMode(1);  
  
  
    u8g2.setFont(u8g2_font_10x20_tf); //Changing font for Battery temp  
  
    u8g2.setCursor(140,67);      //Battery temp position  
  
    u8g2.print(BatteryTempNumbers); //Battery temp  
  
    u8g2.setFont(u8g2_font_VCR_OSD_tf); //Changing it back  
  
    u8g2.setDrawColor(1);  
  
  
    u8g2.drawStr(136, 89, "Motor C");  
  
    u8g2.setFont(u8g2_font_10x20_tf); //Changing font for the °
```

Продолжение приложения Б

```
u8g2.drawUTF8(199,89, DEGREE_SYMBOL) ;  
  
u8g2.setFont(u8g2_font_VCR_OSD_tf) ; //Changing it back  
  
u8g2.setFontMode(1) ;  
  
u8g2.setFont(u8g2_font_10x20_tf); //Changing fon for Motor temp  
  
u8g2.setCursor(140,111) ; //Motor temp position  
  
u8g2.print(MotorTempNumbers); //Motor temp  
  
u8g2.setFont(u8g2_font_VCR_OSD_tf) ; //Changing it back  
  
u8g2.setDrawColor(1) ;  
  
  
u8g2.drawStr(1, 89,"Invertor C") ;  
  
u8g2.setFont(u8g2_font_10x20_tf); //Changing font for the °  
  
u8g2.drawUTF8(100,89, DEGREE_SYMBOL) ;  
  
u8g2.setFont(u8g2_font_VCR_OSD_tf) ; //Changing it back  
  
u8g2.setFontMode(1) ;  
  
u8g2.setFont(u8g2_font_10x20_tf); //Changing font for Invertor temp  
  
u8g2.setCursor(4,111) ; //Invertor temp position  
  
u8g2.print(InvertorTempNumbers); //Invertor temp  
  
u8g2.setFont(u8g2_font_VCR_OSD_tf) ; //Changing it back  
  
u8g2.setDrawColor(1) ;  
  
}
```

Продолжение приложения Б

```
void changeColor(int &BatteryTempBar, int &MotorTempGraph, int
&InvertorTempGraph)
{
    if (BatteryTempBar > 30) {
        u8g2.setDrawColor(0);
    } else if (BatteryTempBar < 30) {
        u8g2.setDrawColor(1);
    }

    if (MotorTempGraph > 20) {
        u8g2.setDrawColor(0);
    } else if (MotorTempGraph < 20) {
        u8g2.setDrawColor(1);
    }

    if (InvertorTempGraph > 20) {
        u8g2.setDrawColor(0);
    } else if (InvertorTempGraph < 20) {
        u8g2.setDrawColor(1);
    }
}
```

Продолжение приложения Б

```
void printCanStatus()
{
    if (DashCan){
        u8g2.drawStr (1, 45, "POWER");
    } else if (!DashCan) {
        u8g2.drawStr(65, 2,"Connecting..." );
    }
}
```

```
void printDriveModeSelected()
{
    if (Endurance_mode) {
        u8g2.drawStr(50, 138,"ENDR");
    } else if (Normal_Mode ) {
        u8g2.drawStr(1, 138,"NORM" );
    } else if (ECO_mode ) {
        u8g2.drawStr(99, 138,"ECO");
    }
}
```

```
void printHV()
```


Продолжение приложения Б

```
{  
  
    u8g2.drawStr(138, 138,"H" );  
  
    u8g2.drawStr(152, 138,"V" );  
  
}  
  
void printR2D()  
  
{  
  
    u8g2.drawStr(167, 138,"R" );  
  
    u8g2.setFont(u8g2_font_5x7_tf );    //Changing the font for the Little 2 after the  
R  
  
    u8g2.drawStr(179, 142,"2" );  
  
    u8g2.setFont(u8g2_font_VCR_OSD_tf ); //Changin it back  
  
    u8g2.drawStr(183, 138,"D" );  
  
}  
  
void print_roll_pitch_yaw() {  
  
    Serial.print("Yaw, Pitch, Roll: ");  
  
    Serial.print(mpu.getYaw(), 2);  
  
    Serial.print(", ");  
  
    Serial.print(mpu.getPitch(), 2);  
  
    Serial.print(", ");  
  
    Serial.println(mpu.getRoll(), 2);  
  
}
```