RoboCup Russia Open - RoboCupJunior 2022

Team Description Paper

Лига RoboCup Junior: Rescue Line

Название команды: Катюша

Состав команды:

- Литвинова Мария, электронщик и программист
- Мозер Златослава, конструктор

Руководители команды:

- Иванов Василий Леонидович
- Мерзлякова Юлия Игоревна

Организация: ГБОУ «Президентский Физико-Математический Лицей №239»

Страна: Россия

Дата: 09.04.2022

Санкт-Петербург

2022

ОГЛАВЛЕНИЕ

1. КРАТКАЯ АННОТАЦИЯ ДОКУМЕНТА

Наша команда поставила перед собой цель создать робота, решающего задачу лиги Rescue Line с использованием продвинутых технических решений. В ходе разработки робота мы научились проектировать печатные платы, программировать новый для нас микроконтроллер STM32, а также существенно повысили навыки 3D-моделирования. Ознакомившись с опытом предыдущих команд из нашего лицея, мы решили использовать камеру и алгоритмы компьютерного зрения для прохождения полигона, что существенно расширяет возможности робота. Все модели и программы находятся в открытом доступе, и мы готовы к обмену опытом.

2. ВВЕДЕНИЕ

2.1. Информация о команде

Наша команда участвует в лиге Rescue Line первый раз. До этого мы принимали участие в RoboCup в лиге Onstage. Наша цель – реализация продвинутых технических решений в конструкции, электронике и программировании, а также получение нового опыта и совершенствование своих знаний и умений для создания робота, решающего задачу лиги Rescue Line.

2.2. Фото и сайт команды

Ссылки на Grabcad и Github нашего проекта, где представлены все модели и программы для нашего робота.

2.3. Опыт участия команды в RoboCup

Кто участвовал	Категория	Результат
Литвинова Мария	RoboCup Russia Open 2019 OnStage Novice	1 место
Литвинова Мария	RoboCup 2019 Sydney OnStage Novice	3 место
Литвинова Мария	RoboCup Asia-Pacific 2019 Moscow OnStage Novice	1 место
Литвинова Мария	Virtual RoboCup Asia-Pacific 2020 OnStage Advanced	1 место и награда Best Software Innovation
Литвинова Мария	RoboCup Russia Open 2021 OnStage Advanced	2 место
Мозер Златослава	RoboCup Russia Open 2021 OnStage Novice	3 место
Мозер Златослава	RoboCup Asia-Pacific 2021 OnStage Novice	3 место

3. ОСНОВНАЯ ЧАСТЬ

3.1. Конструкция

Каркас робота представляет собой фанерную основу на 4 колесах.

- **Конструкция фланца**

Для движения используются моторы с энкодерами 350RPM от DFROBOT. Для их крепления был смоделирован фланец и планетарный редуктор в Autodesk Inventor и напечатаны на 3D принтере из PLA. На фланце предусмотрено крепление для редуктора и крышки, которая закрывает механизм и не дает упасть колесу. В устройстве фланца предусмотрено место под подшипник, на который опирается колесо, тем самым уменьшая нагрузку на вал мотора (масса робота распределяется по подшипнику, что существенно облегчает нагрузку на вал мотора).

Во фланце есть крепление для валов, на которых закреплены сателлиты, а с другой стороны валы держатся в крышке. Коронная шестерня крепится винтами на колесо.

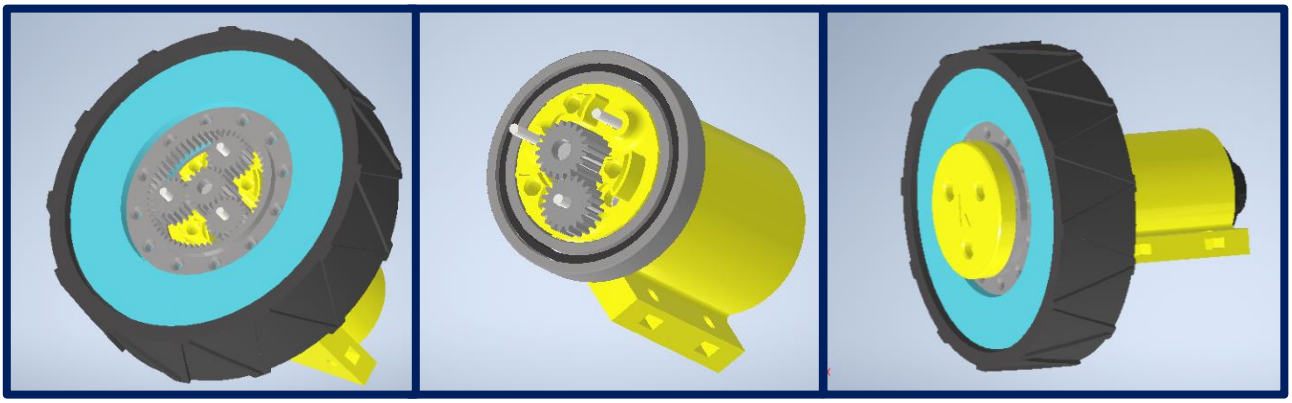


рис. Модель фланца

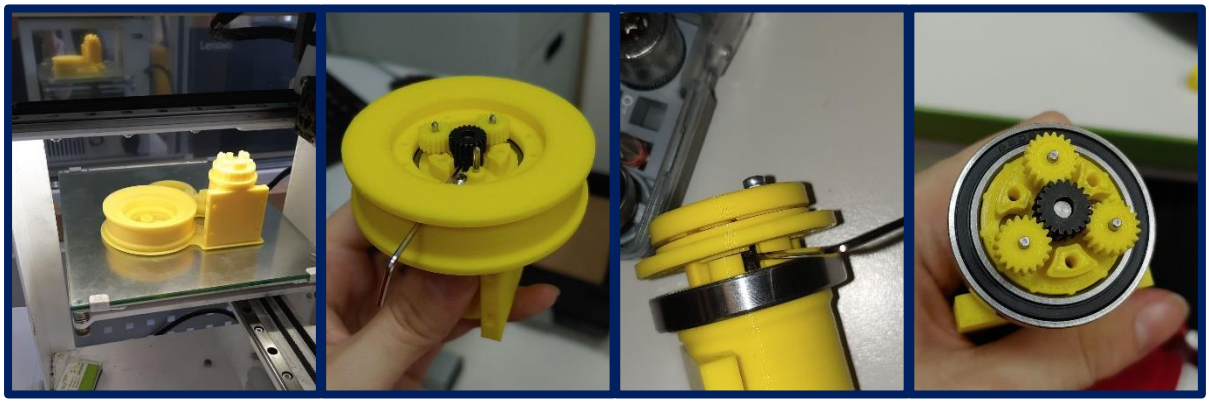
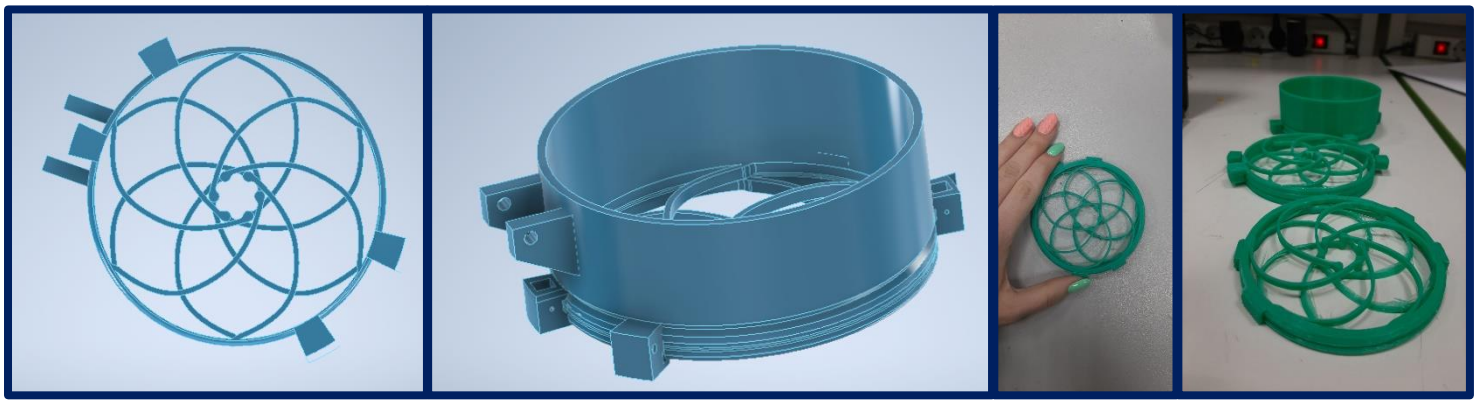


рис. Фото фланца

- **Конструкция захвата**

Для захвата шарика и спасательного набора мы сделали прототип Iris mechanism. Выбор именно этой конструкции в первую очередь был сделан потому, что захват может удерживать спасательный набор (т.е. кубик) - было трудно придумать, как совместить и квадратную, и кубическую форму захвата, чтобы при этом не делать два механизма. Но возникли проблемы с шариком – шарик выскальзывает из этого захвата, если его затянуть. Мы решили сделать цилиндр 3см, попадая в который шарик не может выпасть.

В движение ручки механизма приводятся с помощью тросов, которые закреплены на катушке и наматываются на нее. Катушка, в свою очередь, закреплена на валу мотора, а с другой стороны опирается на маленький подшипник, чтобы конструкция не перекашивалась. Крепление мотора с катушкой вынесено на отдельную лебедку. Так было сделано для удобства крепления – лебедку можно прикрепить где угодно, ее расположение зависит только от длины тросов, которая легко меняется.



Модель захвата и фото разных его версий

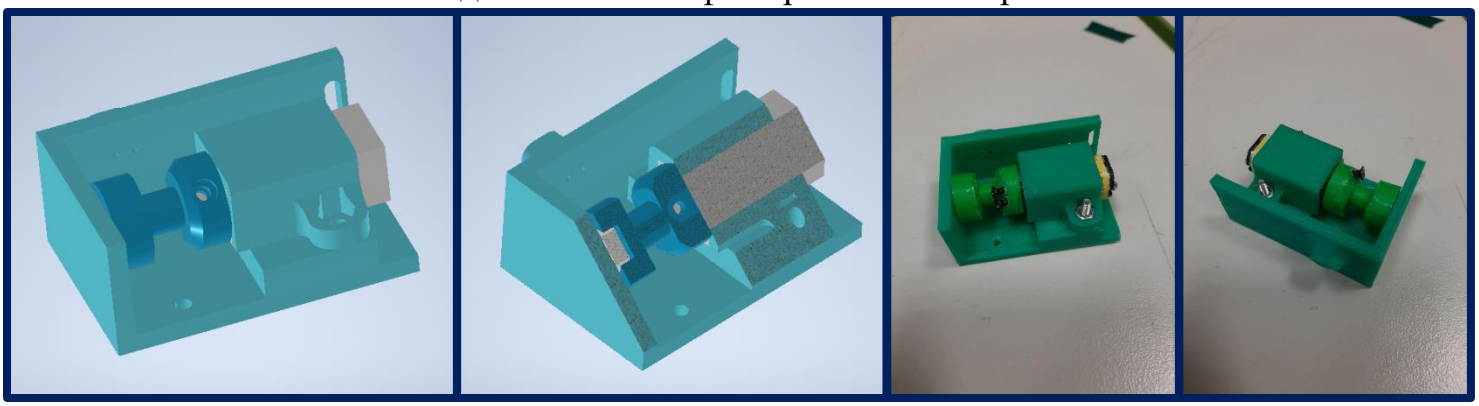
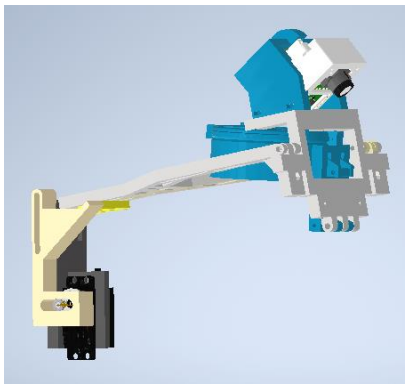


Фото и модель лебедки для зажима-отжима захвата

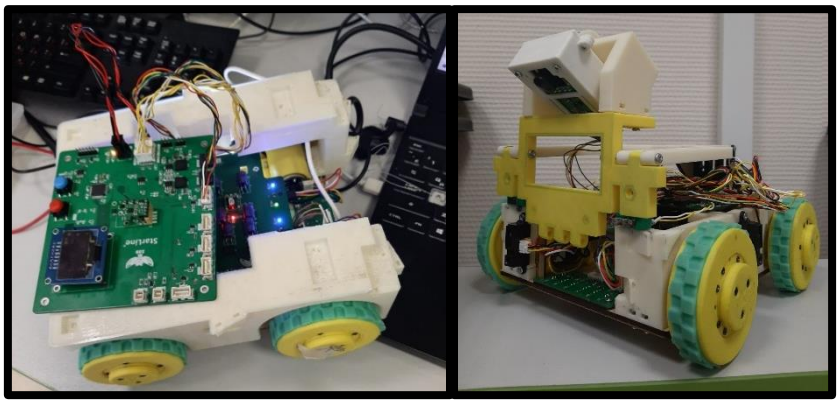
- **Конструкция механизма для подъема и поворота захвата на 180°**

Для поворота захвата на 180° мы используем лебедку, описанную выше, и пружину, которая работает на сжимание. То есть в закрытом положении пружина не натянута, при повороте мотора захват открывается тросами, при этом пружина натянута. Когда мотор возвращается в исходное положение, тросы перестают быть натянутыми и пружина стремится принять свое начальное положение.

Для свободного поворота захвата (т.е. чтобы он не задевал платы внутри робота) перед тем, как перевернуть захват, нужно всю конструкцию приподнять. Для подъема мы используем механизм с эксцентриком – на сервомоторе закреплен подшипник, который поднимает вверх-вниз балку с горизонтальной прорезью в процессе своего движения.



- **Корпус робота**



Платы и захватный механизм крепятся к корпусу робота. Он состоит из 2 больших пластиковых деталей.

рис. 2 версии робота

3.2. Электроника

Для робота был спроектированы и изготовлены 4 платы: материнская плата, плата управления моторами, коммутационная плата датчиков освещенности, коммутационная плата камеры.

Ниже представлена структурная схема робота.

К плате управления моторами подключены 4 мотора, 4 энкодера от них. К этой плате также подключается плата датчиков освещенности.

Аккумулятор КРВТ также подключается к плате управления моторами. На плате установлены 3 стабилизатора питания. Для управления моторами мы используем дополнительный микроконтроллер для удобства управления (скорости рассчитываются на материнской плате, а контроль реальных скоростей осуществляется на плате моторов), а также из-за большого количества пинов, на которых осуществляются прерывания, чтобы не загружать микроконтроллер.

STM32 на плате управления моторов и материнской плате соединены по UART. Также с платы моторов на материнскую плату идут 5В и 3,3В.

На материнской плате есть выводы для 2 моторов, 2 серводвигателей (для механизма захвата). К плате подключены коммутационная плата камеры, гироскоп Alt-IMU v5 от Pololu, OLED-дисплей, 4 лазерных дальномера vl53l0x, 2 концевых выключателя.

Камера OpenMV H7 подключена к коммутационной плате через PLS-разъемы. На плате реализована подсветка из 8 ярких светодиодов.

Перед созданием принципиальных схем была разработана структурная схема робота.

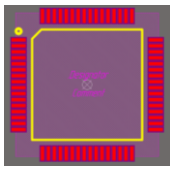
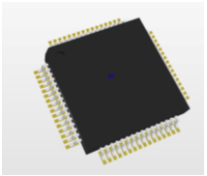
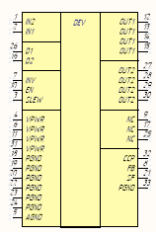
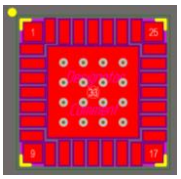
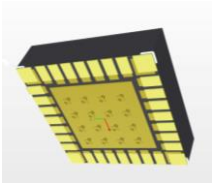
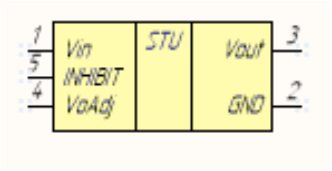
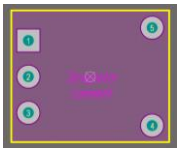
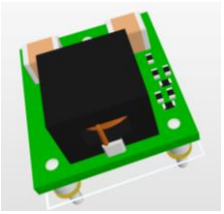
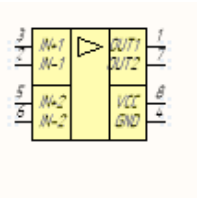
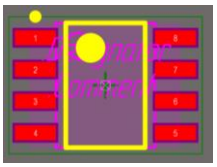
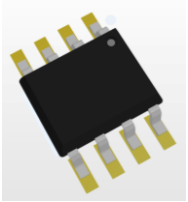
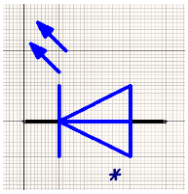
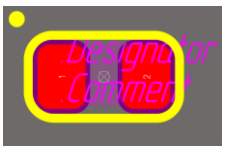
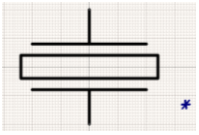
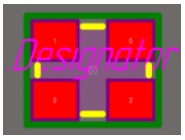
- **Выбор САПР**

Система автоматизированного проектирования Altium Designer — программно-аппаратный комплекс для построения электронных средств на базе печатных плат и программируемых логических интегральных схем это, давно ставший популярным среди профессиональных разработчиков печатных плат.

САПР был освоен в ходе создания робота и печатных плат для него.

- **Создание библиотеки компонентов**

Перед созданием электрических принципиальных схем были созданы библиотеки компонентов: таблица для примера – ниже. Библиотека содержит УГО компонента и посадочное место. Также к каждому посадочному место добавляется 3D-модель компонента.

Название компонента	Условное графическое обозначение	Посадочное место	3D-модель
Микроконтроллер STM32F103RBT6			
Интегральная микросхема управления двигателями MC33926			
Импульсный понижающий DC-DC преобразователь напряжения PTH08080W			
Операционный усилитель LMV358			
Светодиод			
Кварцевый резонатор			

- **Разработка принципиальных схем**

При оформлении схем мы руководствовались следующими документами:

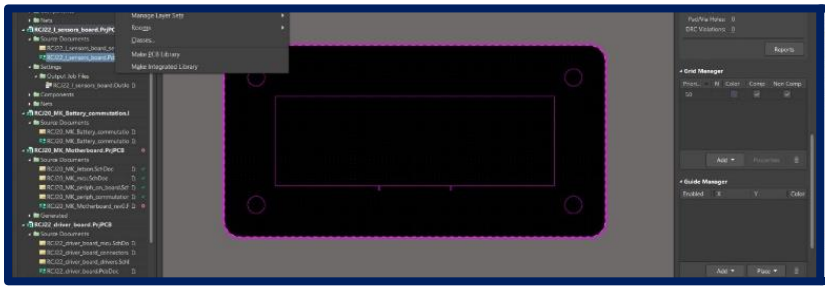
- ГОСТ 2.701-2008 ЕСКД. Схемы. Виды и типы. Общие требования к выполнению
- ГОСТ 2.702-2011 Правила выполнения электрических схем
- ГОСТ 2.710-81 Обозначения буквенно-цифровые в электрических схемах
- ГОСТ 2.725-68 Устройства коммутирующие
- ГОСТ 2.728-74 Резисторы, конденсаторы
- ГОСТ 2.730-73 Приборы полупроводниковые
- ГОСТ 2.759-82 Элементы аналоговой техники

Разработаны принципиальные схемы материнской платы, платы управления моторами, коммутационной платы датчиков освещенности и коммутационной платы камеры.

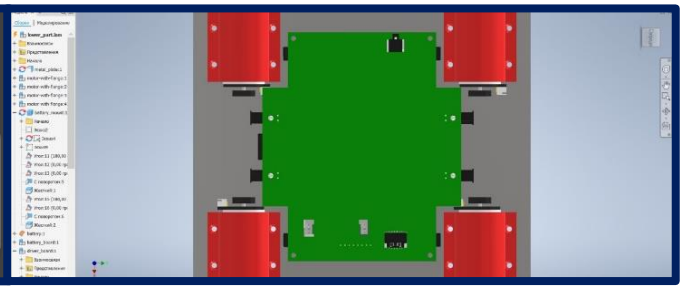
Для примера представлена принципиальная схема материнской платы.

- **Компоновка и трассировка печатных плат**

После разработки принципиальных схем компоненты мы приступили к созданию файла платы. Сначала в САПР для 3D-моделирования (Autodesk Inventor) в общей сборке робота были созданы 3D-модели контуров платы со всеми разъемами, чтобы заранее предусмотреть их расположение. Затем контур был экспортирован в DXF и перенесен в Altium.



Контур платы датчиков в Altium Designer



3D-модель контура платы управления моторами в Autodesk Inventor

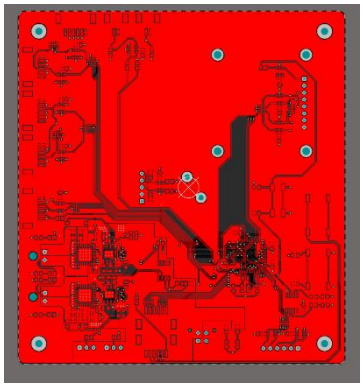
Компоненты из схем были перенесены в файл платы и скомпонованы.

Компоновка должна обеспечивать удобство сборки: все разъемы располагаются рядом с компонентами, для подключения которых они предназначены. Это позволяет использовать более короткие провода.

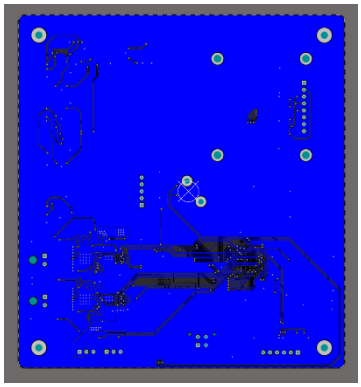
Материнская плата содержит 2 слоя, плата управления моторами – 4 слоя.

Из Altium Designer есть возможность загрузить полную 3D-модель готовой платы в формате STEP.

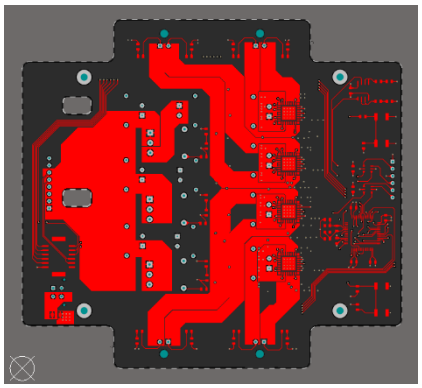
Трассировка верхнего слоя материнской платы



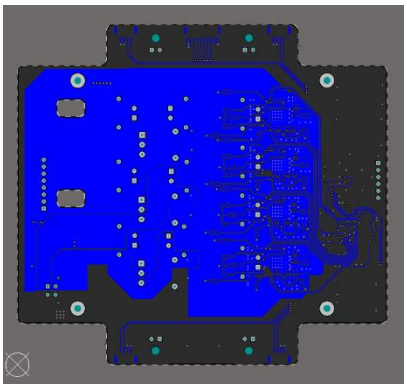
Трассировка нижнего слоя материнской платы



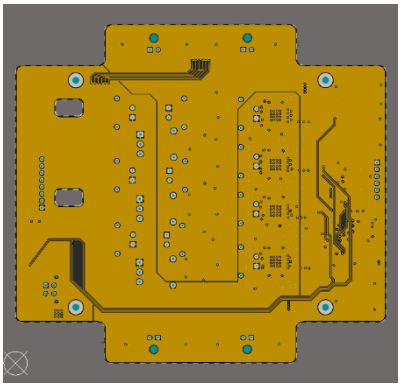
Трассировка верхнего слоя платы моторов



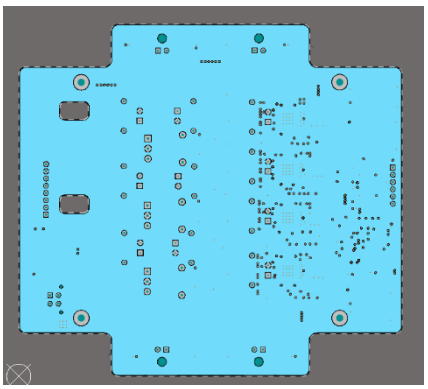
Трассировка верхнего слоя платы моторов



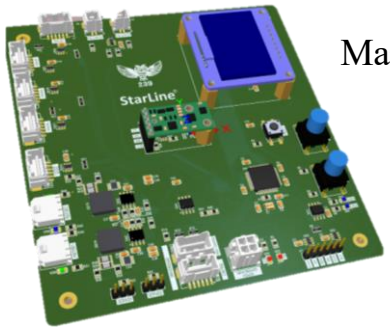
Трассировка первого сигнального слоя платы моторов



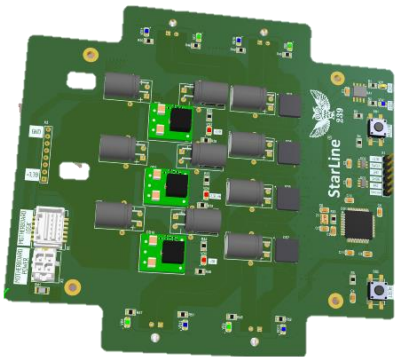
Трассировка второго сигнального слоя платы моторов



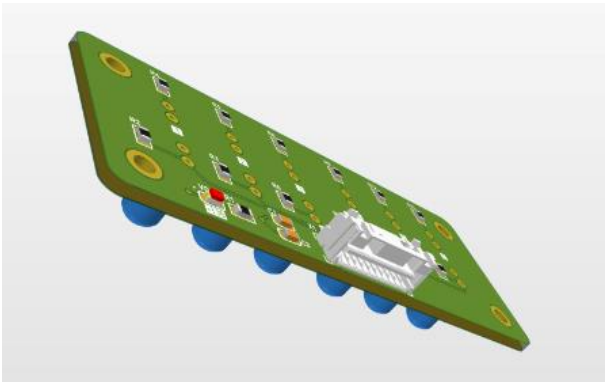
3D-модели плат в Altium Designer



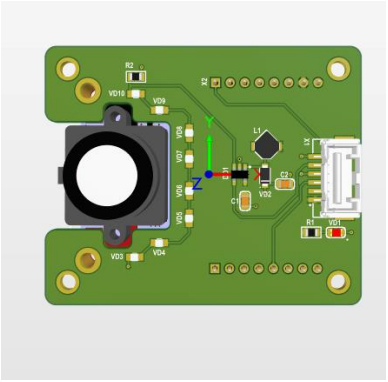
Материнская плата



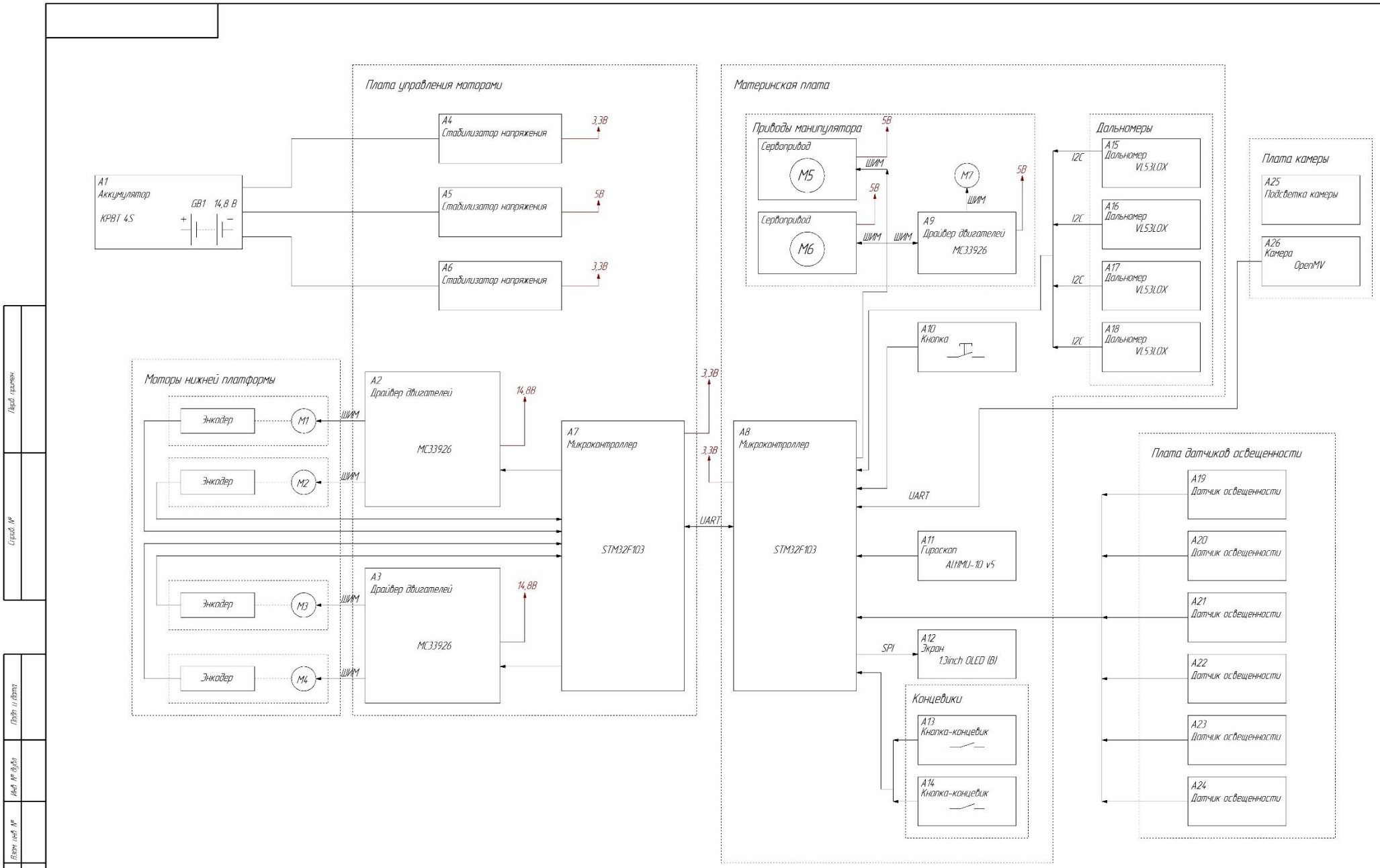
Плата управления моторами



Коммутационная плата датчиков освещенности



Коммутационная плата камеры



Изм.	Лист	№ докум.	Подпись	Дата
Разраб.		Лобанова Н.		10.11.2022
Провер.		Харченко А.В., Перминова В.И.		10.11.2022
Т. контр.				
Реценз.				
Н. контр.				
Утверд.		Харченко А.В., Перминова В.И.		10.11.2022

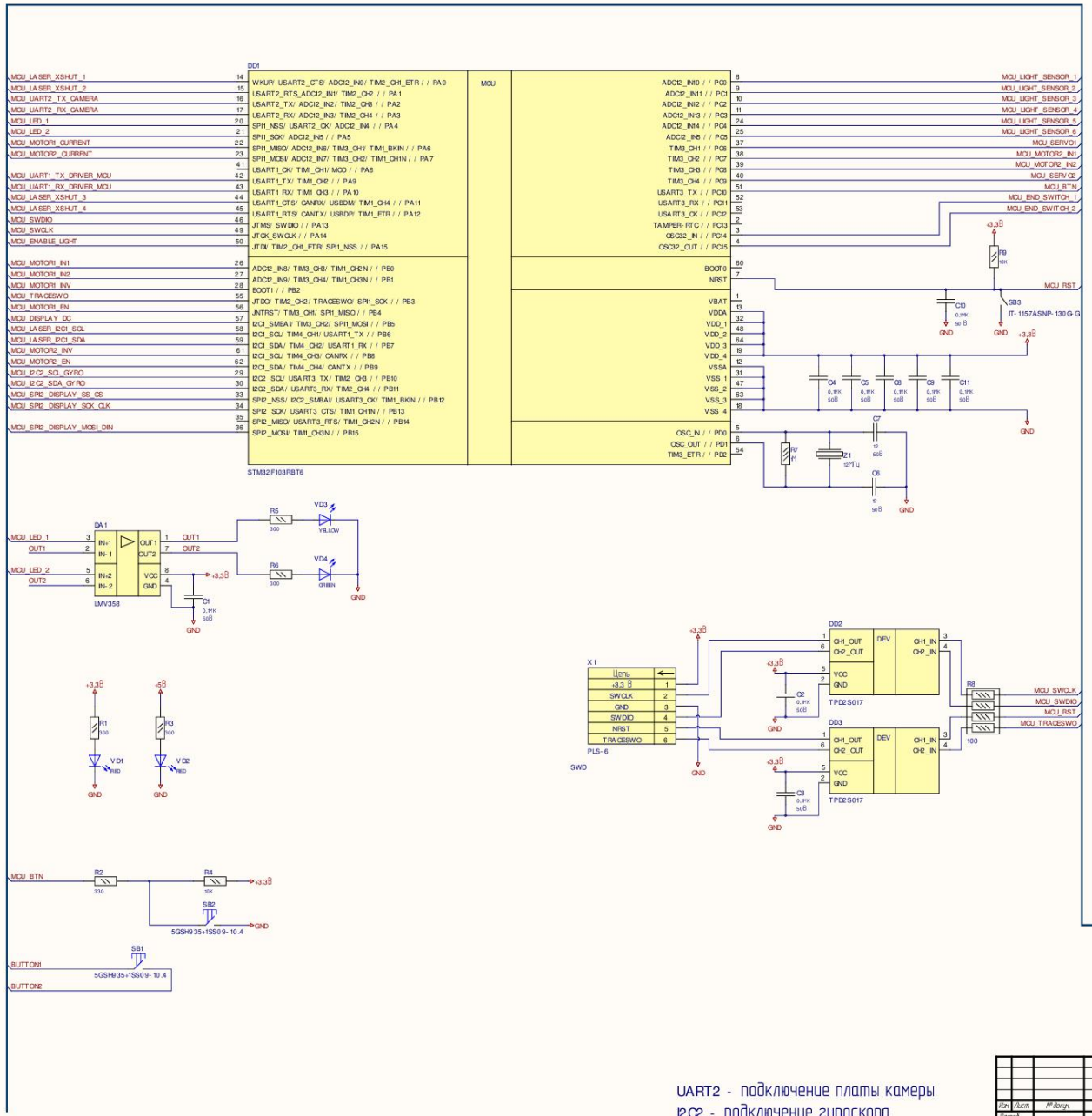
Робот для RCJ Rescue Line

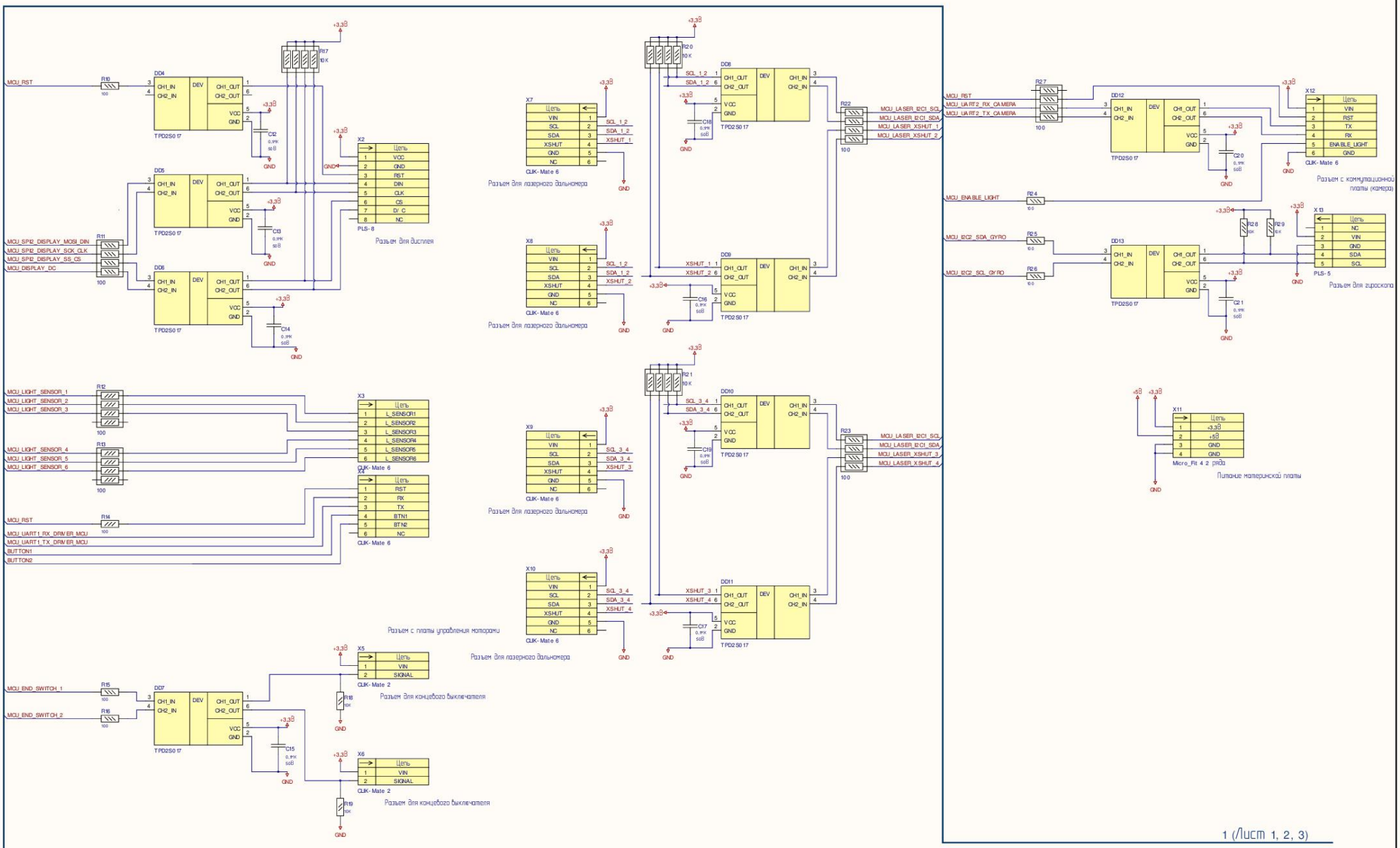
Схема электрическая структурная

Лит.	Масса	Масштаб
Лист	Листов	

Копировал

Формат А2


DD1
STM32F103RBT6
MCU
MCU_LASER_XSHUT_1
MCU_LASER_XSHUT_2
MCU_UART2_TX_CAMERA
MCU_UART2_RX_CAMERA
MCU_LED_1
MCU_LED_2
MCU_MOTOR1_CURRENT
MCU_MOTOR2_CURRENT
MCU_UART1_TX_DRIVER_MCU
MCU_UART1_RX_DRIVER_MCU
MCU_LASER_XSHUT_3
MCU_LASER_XSHUT_4
MCU_SWDIO
MCU_SWCLK
MCU_ENABLE_LIGHT
MCU_MOTOR1_IN1
MCU_MOTOR1_IN2
MCU_MOTOR1_INV
MCU_TRACESWO
MCU_MOTOR1_EN
MCU_DISPLAY_DC
MCU_LASER_I2C1_SCL
MCU_LASER_I2C1_SDA
MCU_MOTOR2_INV
MCU_MOTOR2_EN
MCU_I2C2_SCL_GYRO
MCU_I2C2_SDA_GYRO
MCU_SPI2_DISPLAY_SS_CS
MCU_SPI2_DISPLAY_SCK_CLK
MCU_SPI2_DISPLAY_MOSI_DIN
MCU_LIGHT_SENSOR_1
MCU_LIGHT_SENSOR_2
MCU_LIGHT_SENSOR_3
MCU_LIGHT_SENSOR_4
MCU_LIGHT_SENSOR_5
MCU_LIGHT_SENSOR_6
MCU_SERVO1
MCU_MOTOR2_IN1
MCU_MOTOR2_IN2
MCU_SERVO2
MCU_BTN
MCU_END_SWITCH_1
MCU_END_SWITCH_2
MCU_RST
+3,3В
GND
SB3
IT-1157ASNP-130G-G
DA1
LMV358
VD3
YELLOW
VD4
GREEN
X1
SWD
PLS-6
DD2
DD3
TPD2S017
R8
100
MCU_SWCLK
MCU_SWDIO
MCU_RST
MCU_TRACESWO
MCU_BTN
SB2
SB1
5GSH935+1SS09-10.4
BUTTON1
BUTTON2
UART2 - подключение платы камеры
I2C2 - подключение гироскопа
I2C1 - подключение лазерных дальномеров
SPI2 - подключение дисплея
Материнская плата
Схема электрическая принципиальная
1 (Лист 1, 2, 3)



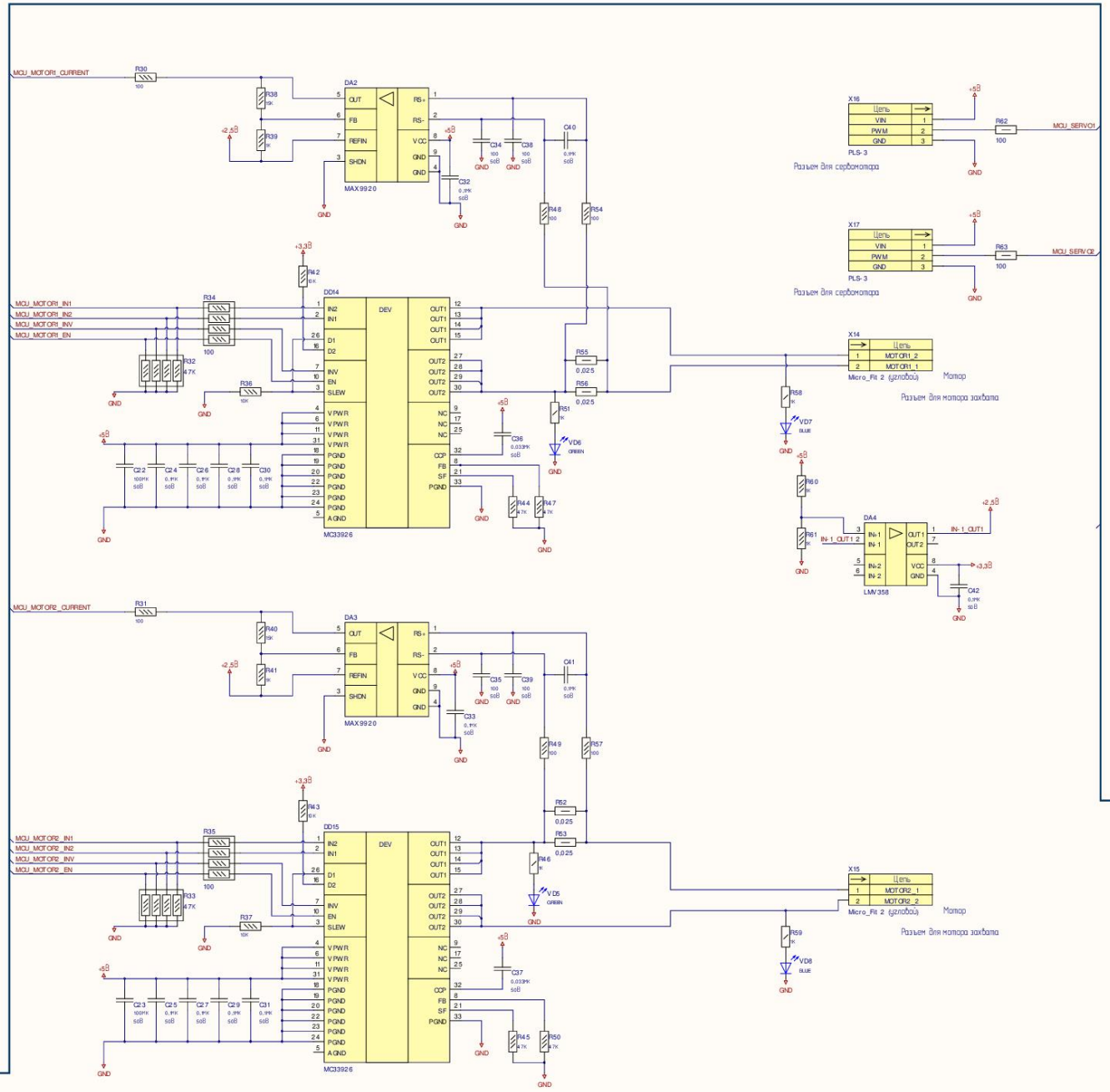
1 (Лист 1, 2, 3)

Изм.	Лист	№ докум.	Подп.	Дата

Копировал

Формат А1

Лист 2



1 (Лист 1, 2, 3)

Изм.	Лист	№ докум.	Подп.	Дата

Лист 3

Копировал

Формат А1

- **Изготовление печатных плат и монтаж компонентов**

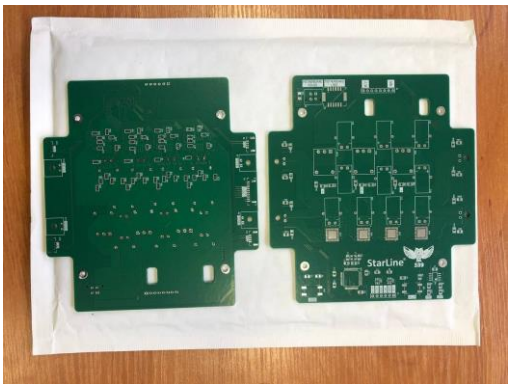
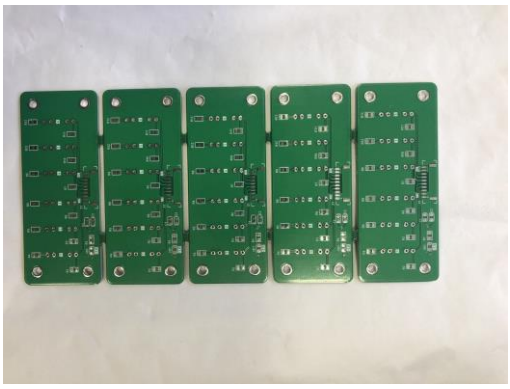



рис. Платы моторов и датчиков после изготовления на производстве

Компоненты мы монтировали на платы вручную, с помощью паяльника и термофена. Затем вручную обжимались провода в соответствующие разъемы.

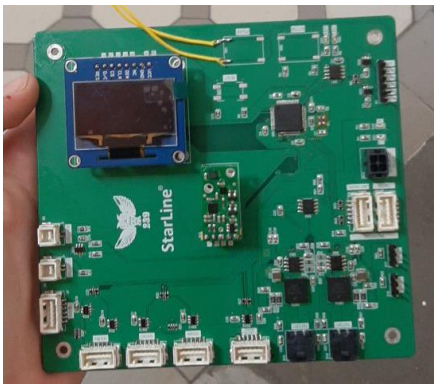


рис. Материнская плата после монтажа компонентов

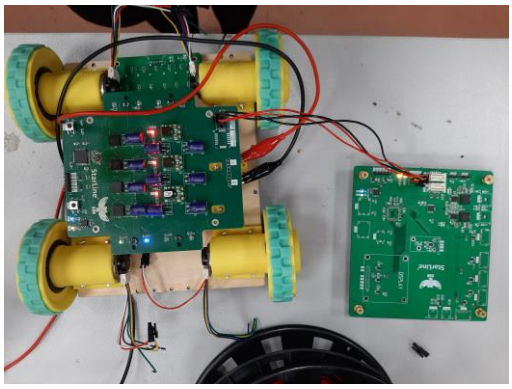
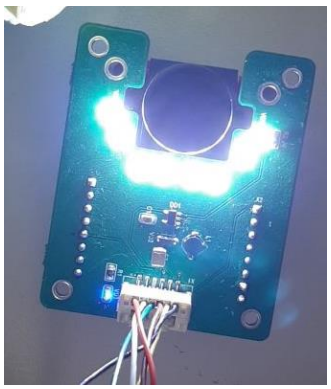



рис. Тестирование плат

3.3. Программное обеспечение

- **Компьютерное зрение**

Для прохождения полигона необходимо ориентироваться на линии и распознавать зеленые метки. Для решения поставленной задачи мы решили использовать технологии компьютерного зрения. Для упрощения реализации этой технологии мы решили использовать камеру OpenMV, которая программируется на microPython в специальной среде разработки.

Для начала: о распознавании линии.

Для распознавания линии изображение, получаемое с камеры, разбивается на несколько горизонтальных полос. Каждая полоса создается как отдельный регион интереса (roi). Далее в этом регионе интереса проводится поиск черных областей с помощью встроенной функции find_blobs. Для каждой области определяется ее центр по осям x и y и ширина области.

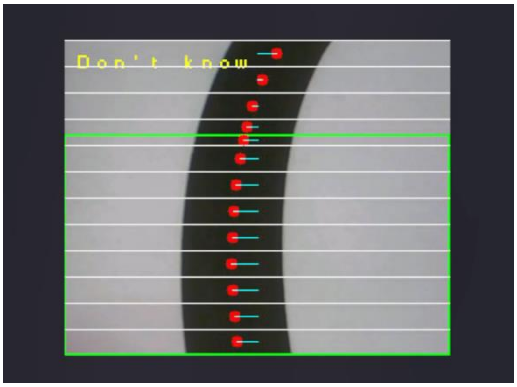


рис. Распознанная линия

Для каждой вычисляется отклонение по формуле:

$$sqr\_deviation = \left(\frac{x_{size}}{2} - x_{center}\right) * width$$

x_{size} – ширина изображения по горизонтали

x_center – x-координата черной области в регионе интереса.

$width$ - ширина области

$sqr\_deviation$ – величина, характеризующая расположение данной черной области на изображении

Умножение на ширину области нужно для того, чтобы более протяженные по горизонтали области (на резких поворотах) влияли на движение робота сильнее.

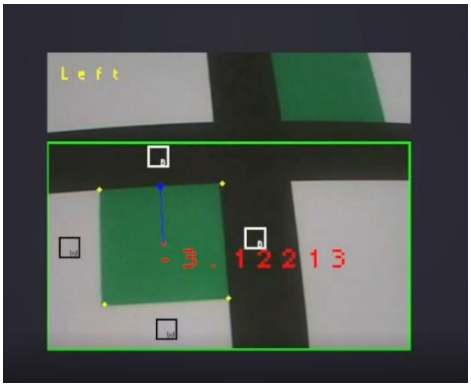
```
if len(self.lfr_centers) != 0:
    for i in range(len(self.lfr_centers)):

        deviation = (self.lfr_centers[i][0] - x_size//2)*self.line_widths[i]
        self.line_deviations.append(deviation)
```

рис. Фрагмент программы с расчетом отклонения

Теперь о распознавании зеленых меток.

Для начала ищем зеленые области с помощью функции find_blobs.



Далее с помощью функции min_corners определяем вершины минимального описывающего прямоугольника (он обозначается на изображении желтыми точками).

Вершины сортируются по возрастанию y-координаты.

```
arr.sort(key = lambda x: x[1])
```

рис. Фрагмент программы с сортировкой вершин

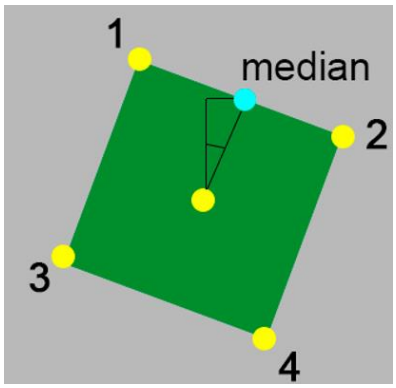


рис. Вычисление угла наклона

Далее вычисляются координаты точки – середины отрезка, соединяющего 2 верхние вершины и вычисляется угол поворота относительно вертикали по следующим формулам:

$$med_x = \frac{x_1 + x_2}{2} \qquad med_y = \frac{y_1 + y_2}{2} \qquad angle = arctg(\frac{med_x - c_x}{med_y - c_y})$$

```
def count_angles(self):

    self.angles_array = []

    for i in range(self.count):
        c_x = self.centres_array[i][0]
        c_y = self.centres_array[i][1]
        if self.DRAWING_SQR:
            self.cam_image.draw_circle(c_x, c_y, 2, (255,0,0), 3, True)
        med_x = int((self.m_array_converted[i][0][0] + self.m_array_converted[i][1][0])/2)
        med_y = int((self.m_array_converted[i][0][1] + self.m_array_converted[i][1][1])/2)
        angle = math.degrees(-math.atan((med_x - c_x)/(med_y-c_y)))
        self.angles_array.append(angle)
        if self.DRAWING_SQR:
            self.cam_image.draw_arrow(c_x, c_y, med_x, med_y, (0,0,255), 2)
            self.cam_image.draw_circle(med_x, med_y, 3,(0,0,255), 2, True)
            self.cam_image.draw_string(c_x - 10, c_y, str(angle),(255, 0, 0), 1)
```

рис. Функция вычисления углов отклонения меток

Далее определяются координаты точек справа, слева, сверху и снизу каждой метки. Вокруг каждой описывается окрестность в виде прямугольника. В каждом из них проверяется наличие черных областей, если они есть, считается, что это черная линия, если их нет, то значит, что это белое поле.

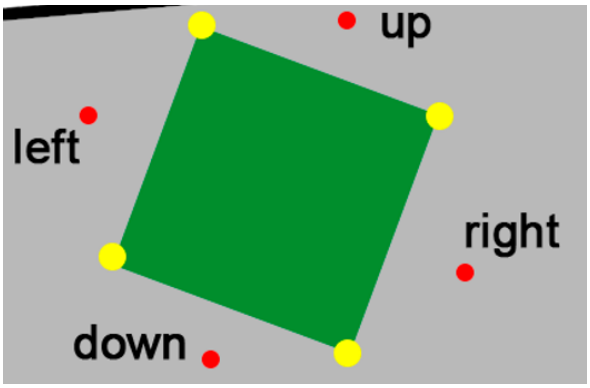


рис. Положение правой, левой, нижней и верхней областей поиска

$$up_x = \frac{(dist\_coeff + 1) * (x_1 + x_2 - c_x)}{2 * dist\_coeff}$$

$$up_y = \frac{(dist\_coeff + 1) * (y_1 + y_2 - c_y)}{2 * dist\_coeff}$$

На изображении прямоугольник обозначается белым, если в нем найдены черные области, и черным – если их нет. Цвета по перимеру метки кодируются 1 массивом из 4 элементов. 0 – белый, 1- черный.

Далее в зависимости от конфигурации меток определяется направление поворота.

Цифрами от 0 до 4 закодированы направления поворота.

Данные о линии и метках передаются от камеры на STM32 по протоколу UART.

```
self.uart.writechar(dir_info) #avg_data
time.sleep_ms(1)
self.uart.writechar(converted_line_info) #
time.sleep_ms(1)
self.uart.writechar(converted_angle_info)
time.sleep_ms(1)
self.uart.writechar(byte_signal)
time.sleep_ms(1)
```

рис. Отправка данных на микроконтроллер

- **Программирование STM32 на материнской плате**

Микроконтроллер программируется на C++ в среде разработки Keil µVision.

Для управления были написаны библиотеки: time_lib (функция задержки), uart_mb (для обработки сигналов UART от камеры и отправки сигналов на плату управления моторами), adc_dma (для работы с АЦП и считывания информации с датчиков с использованием DMA).

Основная программа – цикл езды вдоль линии.

1. Инициализация, запуск
2. Ожидание нажатия кнопки для начала езды
3. Считывание значений с датчиков, запоминание значений с камеры
4. Вычисление ошибки для каждого датчика.
   err_sensors[0] = sensor1()-grey;
5. ПИ-регулятор в зависимости от показаний датчиков освещенности
6. ПИД-регулятор с кубической составляющей ошибки для камеры
7. Сложение управляющих воздействий с датчиков и с камеры.
   Отправка скоростей моторов на плату управления моторами.

- **Программирование STM32 на плате управления моторами**

Для управления написаны библиотеки: uart (для обработки сигналов UART с материнской платы), pwm (для управления ШИМ на моторах), encoders (для обработки рперываний с энкодеров), speed_control (регулятор для корректировки скорости в зависимости от энкодеров).

```
int vell(int speed, long int encl)
{
    float speed_k = SPEED_COEFF;
    float enc_const = ENC_COEFF;

    float err = speed*enc_const + (Encoder1()-encl);
```

рис. Функция – регулятор скоростей моторов

Основная задача STM32 на плате моторов – контроль скоростей моторов. На материнской плате вычисляются скорости и передаются по UART на плату моторов. Плата моторов подает мощности на моторы, вычисляет текущие скорости по показаниям энкодеров и в зависимости от этого подает корректирующие мощности.

```
void EXTI9_5_IRQHandler(void)
{
    int signal1;
    if (EXTI_GetITStatus(EXTI_Line9) != RESET)
    {
    EXTI_ClearFlag(EXTI_Line9);

     signal1 = GPIO_ReadInputDataBit(GPIOC, GPIO_Pin_5);
    if (signal1)
    {
        encl = encl + 1;
    }
    else
    {
        encl = encl - 1;
    }
}
```

рис. Функция обработки прерываний с энкодеров

4. ЗАКЛЮЧЕНИЕ

4.1. Обсуждение

В ходе разработки робота мы научились проектировать печатные платы, программировать новый для нас микроконтроллер STM32, а также существенно повысили навыки 3D-моделирования.

Из-за того, что проектировали мы платы впервые, были ошибки. Например, неправильные размеры посадочных мест для драйверов моторов привели к усложнению монтажа для платы. Программирование STM значительно отличается от программирования Arduino, где не нужно вникать в работу периферии, т.к. за это отвечают встроенные функции. Программирование вызвало немало трудностей. Благодаря этому мы приобрели новый опыт.

До этого проекта мы уже умели работать с камерой OpenMV, но сейчас значительно углубили свои знания. Также мы продвинулись в целом в программировании на Python (например, на примере кода для камеры, научились работать с классами.)

Сильно продвинулись в 3D-моделировании. Мы старались строить сборки полностью на зависимостях, использовать проекции одних деталей на другие. В 3D-сборке робота присутствуют все элементы конструкции, в том числе платы.

4.2. Благодарности

4.1.1. Благодарим Президентский ФМЛ №239, компанию StarLine и Фонд «Финист», а также Кировский завод за финансовую поддержку проекта.

4.1.2. Благодарим наших преподавателей: Казанцеву Ольгу Юрьевну, Иванова Василия Леонидовича, Мерзлякову Юлию Игоревну, Хартанена Александра Вячеславовича

4.3. Список литературы

- https://www.youtube.com/playlist?list=PL8OgDYWys_b6XtOjCejd37aVv0ic24jqV - Курсы по программированию STM32
- https://www.youtube.com/watch?v=DBbG_yhcP9Q Iris-механизм
- https://rescue.rcj.cloud/events/2021/robocup2021/line/ RoboCup2021 Rescue Line