Государственное бюджетное общеобразовательное учреждение

Школа № 2103 города Москвы

ДОКУМЕНТАЦИЯ ПО КОМАНДНОМУ КЕЙСУ №2

“Роботизированная система грузов”

МОСКОВСКОЙ ПРЕДПРОФЕССИОНАЛЬНОЙ ОЛИМПИАДЫ ШКОЛЬНИКОВ

Работу выполнила команда

«Эйлер Плюс»

Ученики 11 «Т» и 10 «Т» классов

ГБОУ Школы № 2103:

Колдашов Иван Сергеевич

Коняхина Евгения Александровна

Жучков Матвей Алексеевич

Тарасова Валентина Андреевна

Кондратенко Данила Денисович

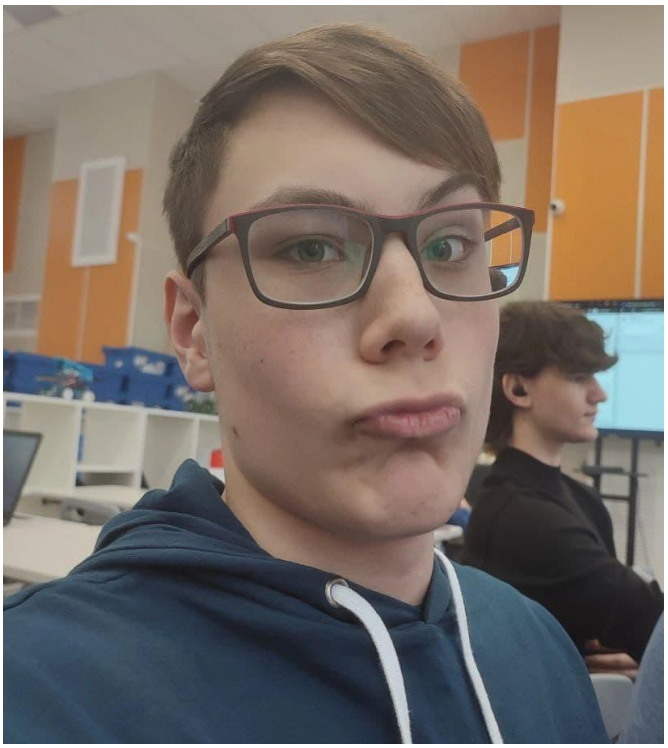
Научный руководитель, инженер:

Ткаченко Артем Алексеевич

Москва 2023

Оглавление

Наша команда:



Колдашов Иван Сергеевич

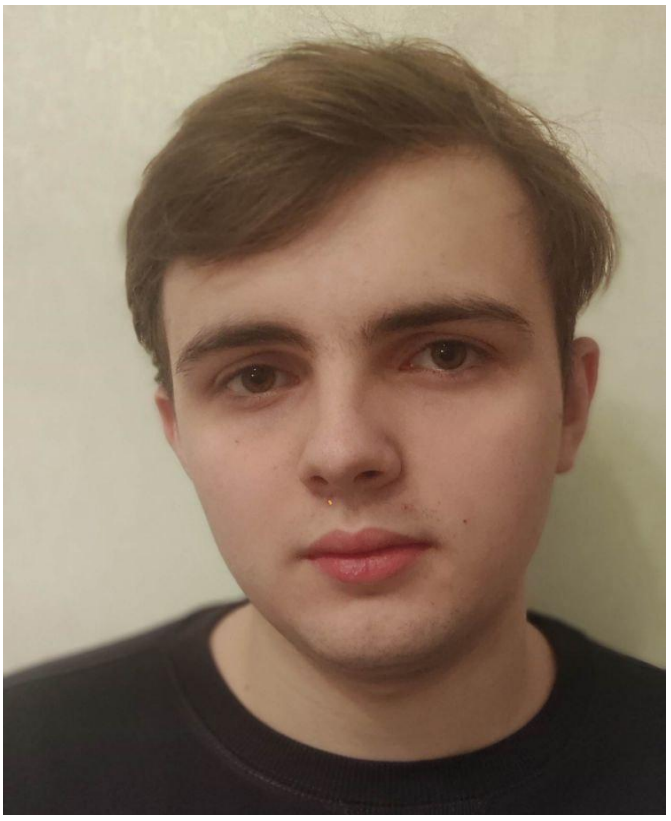
Программист C++, Электронщик

Коняхина Евгения Александровна

Ответственная за документацию и талисман

Тарасова Валентина Андреевна

Программист Python



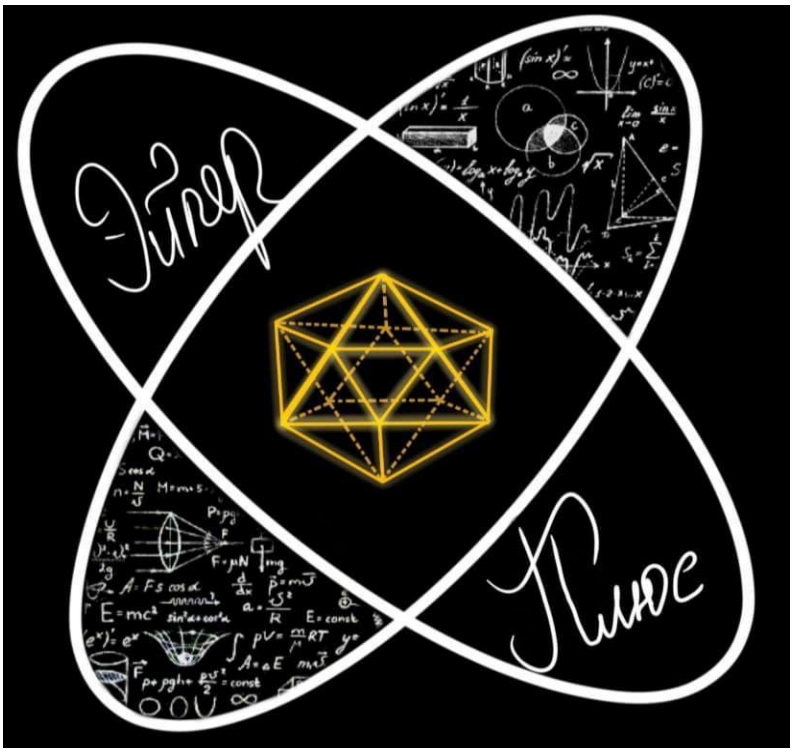
Жучков Матвей Алексеевич

Главный 3D дизайнер

Кондратенко Данила Денисович

3D дизайнер

Наш логотип:



Введение

В современном мире для всевозможных производственных процессов - от выпечки хлеба и сортировки лекарств до сборки смартфонов и космических кораблей - просто необходимы механические системы, способные непрерывно или в соответствии с заданным графиком перемещать в нужное место отдельные детали или готовые изделия. Это экономит не только время (ведь вручную подобный процесс длится гораздо дольше!), но и силы тех, кто занят в производстве.

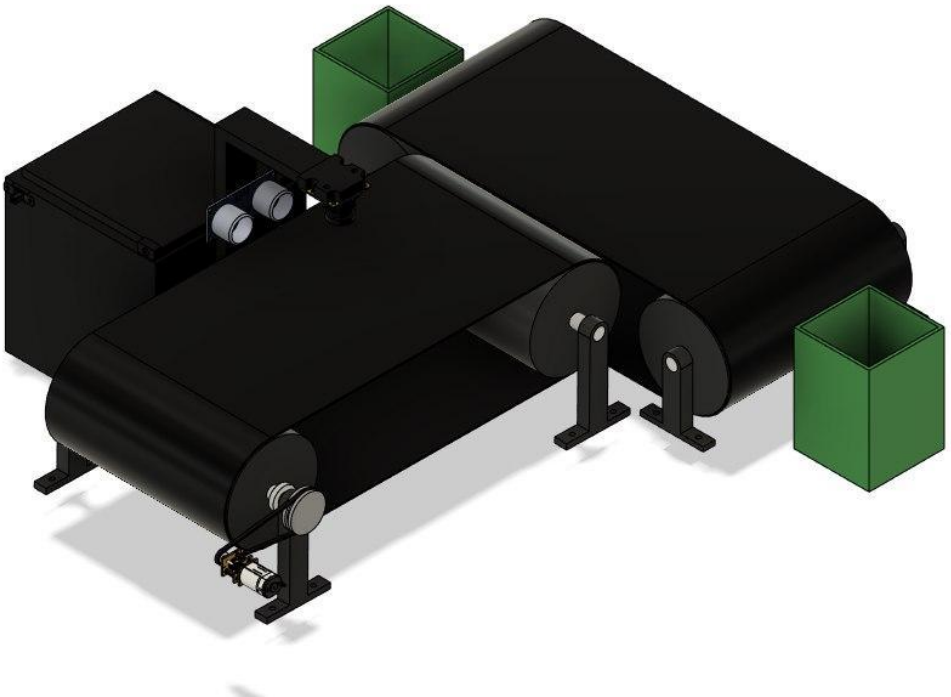
Об облегчении своего труда и ускорении рабочего процесса человек задумывался давно. И именно для решения этой задачи был создан конвейер. Многие ошибочно полагают, что впервые его использовал Генри Форд, на рубеже XIX и XX веков. Однако, если углубиться в историю, то окажется, что механические системы для подачи воды, являющиеся своеобразными прототипами современных конвейеров, использовались ещё до Нашей Эры - в Древнем Египте, Месопотамии, Китае и Индии. В Средневековой Европе похожие устройства использовались сначала на мельницах, а затем, постепенно совершенствуясь, и на других предприятиях. Генри Форд лишь создал "конвейерное производство", завершив изобретательский процесс, длившийся не одно столетие. Впрочем, именно после Форда конвейер (как устройство и как система производства) стал массово и повсеместно внедряться в различные производственные процессы, что привело к огромному скачку технического прогресса. Впоследствии системы совершенствовались с появлением новых видов конвейеров.

В наши дни от конвейеров ждут помощи не только в транспортировке и сборке продукции, но и в ее сортировке. Например, в современных складских системах требуется обеспечить распределение и хранение деталей, заготовок, готовых товаров и т.д.. Для непрерывности их подачи и распределения необходимо применять конвейерные системы с элементами роботизации. Преимущества роботизации технологических процессов - это прежде всего обеспечение точности, качества и производительности, а также замена человека при выполнении опасных, монотонных, тяжелых работ. Поэтому у нашей команды возникла идея спроектировать и реализовать конструкцию и программное обеспечение роботизированной системы распределения грузов, товаров, деталей. Конечно, данная работа не является открытием или самостоятельным изобретением, однако мы лишь учимся и для нас это является отправной точкой для создания новых технологий, которые решали бы задачи по облегчению жизни человечества в различных сферах его деятельности.

Описание устройства “Т-образ”

Наше устройство представляет собой стационарную систему распределения грузов, способную различать груз по цвету и сортировке в нужные ячейки. При движении грузов по ленточному конвейеру система автоматически выполняет необходимые манипуляции. Благодаря камере система определяет нужные параметры отсеивания, а ультразвуковой датчик дает необходимую задержку для обработки информации. Далее следует второй конвейер, выполняющий самую примитивную работу (доставка до нужного контейнера). Управляется система с помощью мобильного приложения, которое беспроводным путем отдает команды.

На фотографии ниже представлена 3D-модель всей установки:



Функциональность:

Функционал, реализованный согласно техническому заданию:

1. Соответствие между цветами грузов и контейнерами задается в мобильном приложении

2. Приложение запускает конвейер

3. При достижении груза зоны видимости ультразвукового датчика конвейер автоматически останавливается

4. Цвета грузов распознаются

5. Активируется механизм сбрасывания

Функционал, реализованный сверх требований ТЗ:

1.Вывод кол-во отсортированных грузов в каждом боксе

2.Возможность смена цвета в любой момент времени

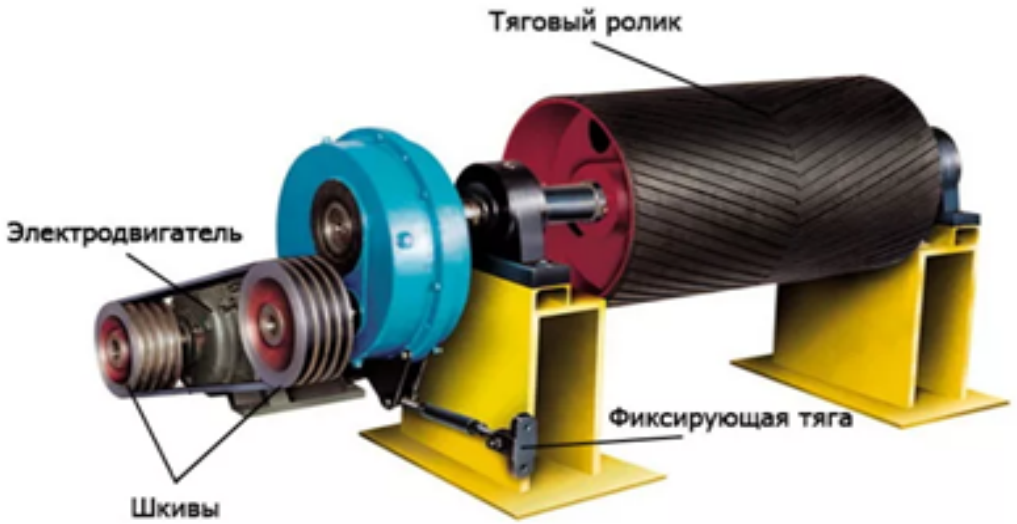
3.Возможность остановки конвейера в любой момент времени

Узлы устройства:

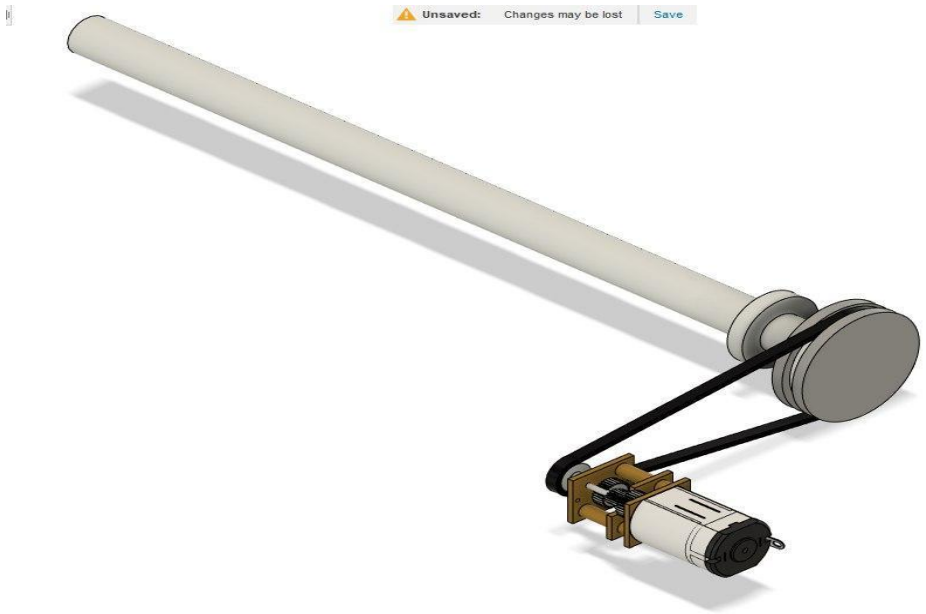
1. Конвейер (*2 шт. ; Будут реализованы посредством ременной передачи*)

ТЗ: Привод конвейера может быть на основе мотор-редуктора, сервопривода, шагового двигателя. Запуск/остановку привода, задание/изменение цветов необходимо осуществить беспроводным способом

Пример ременной передачи:

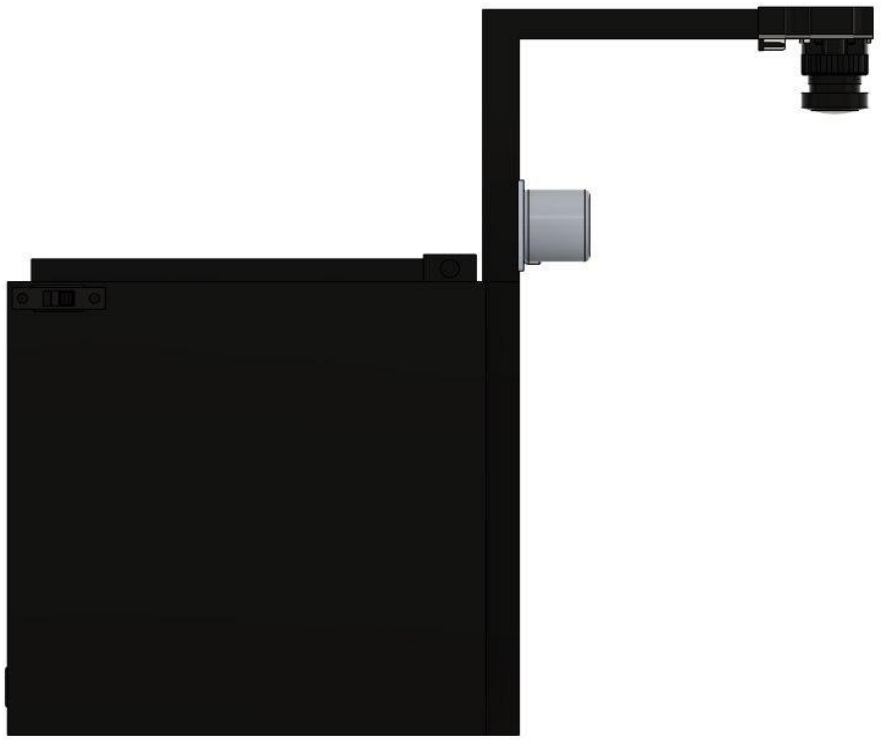


Сама передача:



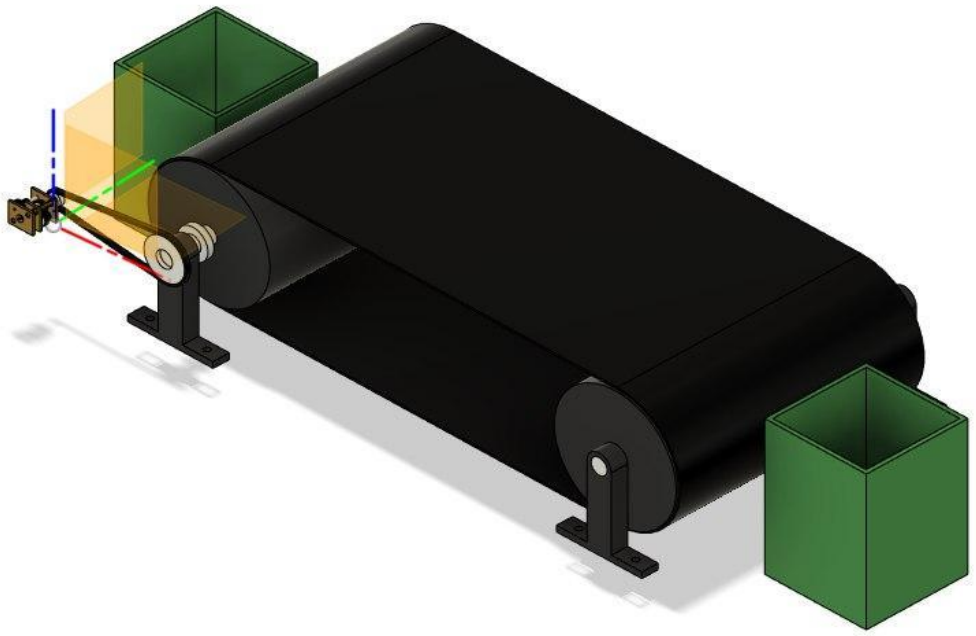
2. Системы сканирования для обнаружения грузов

Тз: Система сканирования может быть реализована с помощью камеры или датчика цвета. (в нашей работе система сканирования будет реализован при помощи камеры)



3. Механизм сбрасывания

Второй конвейер. *(В зависимости от цвета, конвейер будет скидывать грузы в контейнеры)*



4. Контейнеры для сборки грузов

ТЗ: Не менее 70х50х50 мм, Количество контейнеров - 2

5. Грузы

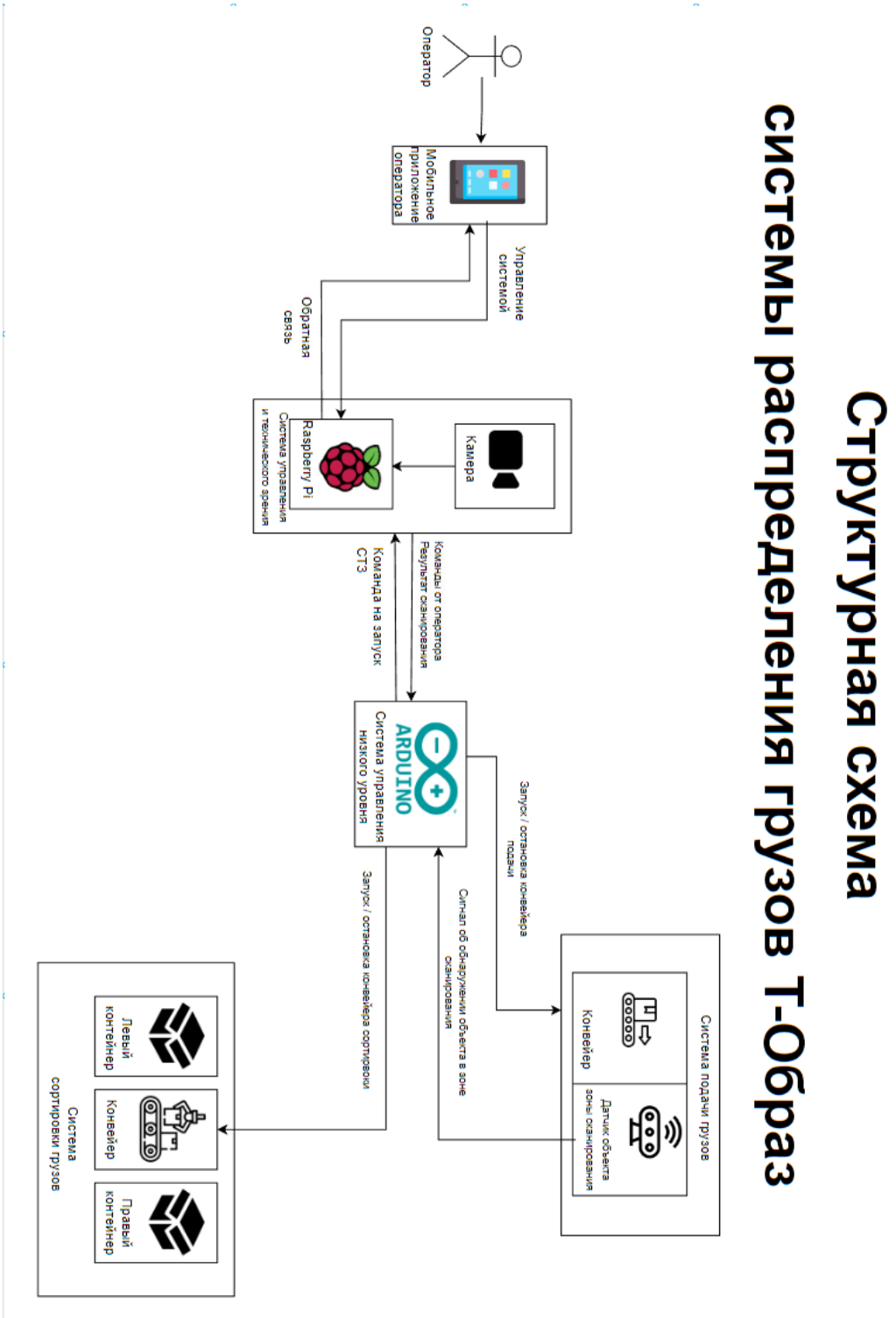
ТЗ: Необходимо применить четыре любых цвета для грузов. Количество грузов - 4. материала грузов можно использовать пластик, дерево, металл.

6. Мобильное приложение (Запуск/остановка устройства)

Тз: Каждый из контейнеров необходим для сбора грузов двух цветов. Соответствие между цветами грузов и контейнерами задается в мобильном приложении

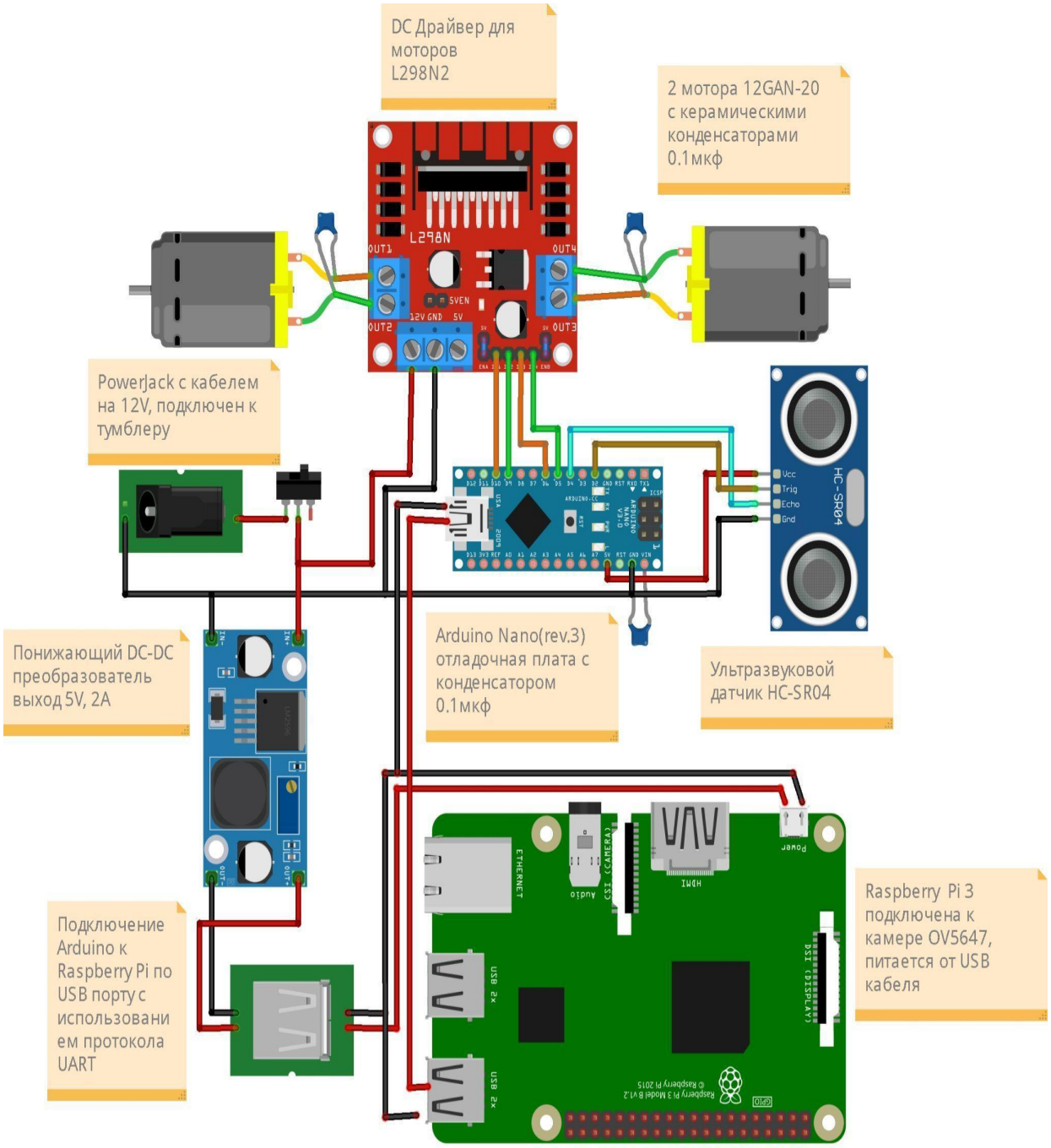
7. Автоматическая остановка Конвейера

Принцип работы

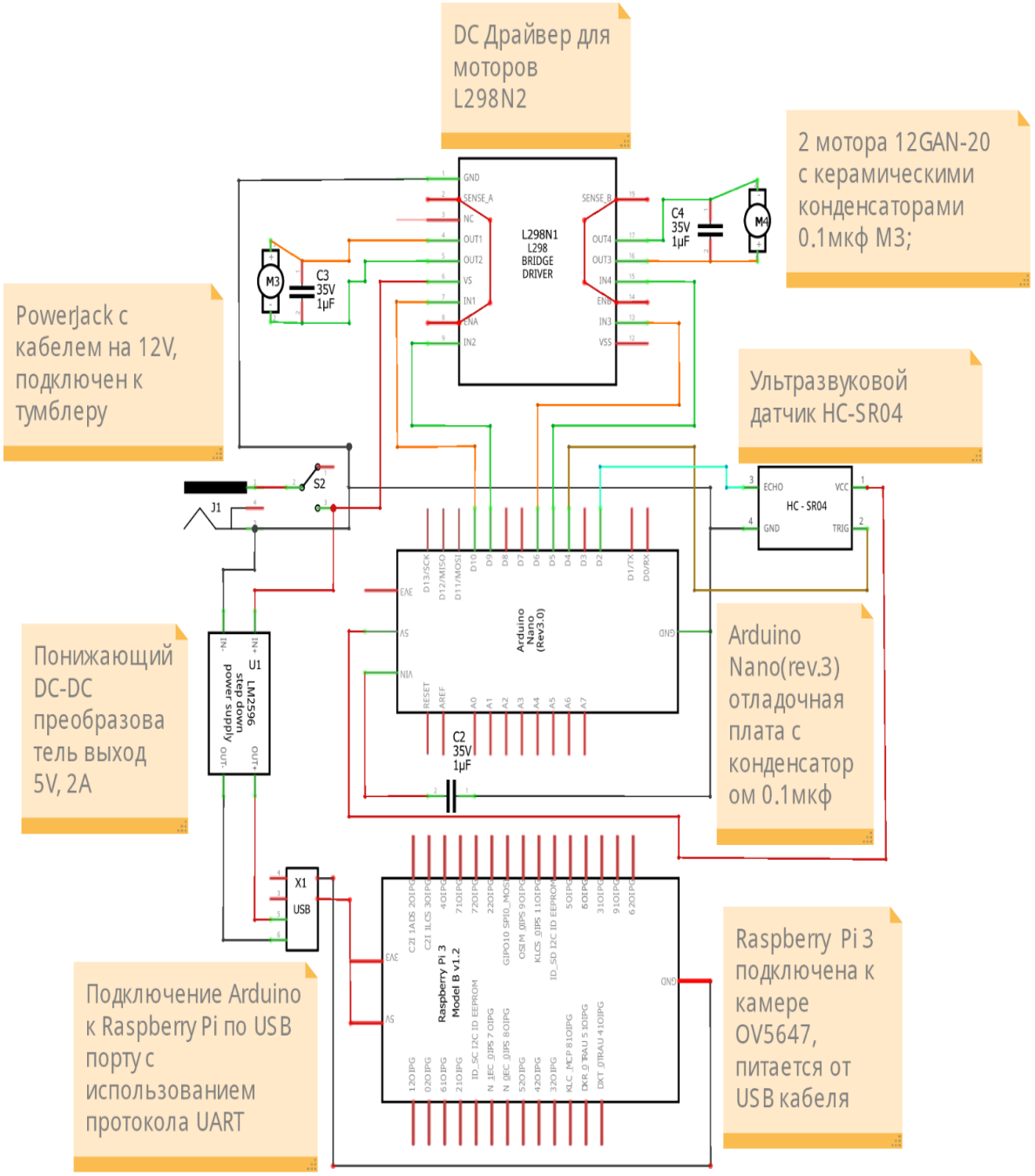


Электрические схемы:

Макетная схема:



Принципиальная схема:



Список компонентов:

Имя	Устройства	Модель	Параметры	Комментарий
micro	Отладочная Плата	Arduino Nano (Rev3.0)	5-20V 40 mAh ATmega328P	Управление двигателями и считывание данных с узлов
HC-SR04 Sensor 1	Ультразвуковой датчик	HC-SR04	5V 15 мА 15* 2см-4м	Определение расстояния, подключение к Nano
M3	DC Motor	12 GAN-20	12 В 0.75A 1200 об\мин	Мотор
M4				
S2	Toggle Switch	SS12F15	5мм 1p2т	Переключатель
C1	Керамические конденсаторы (Ceramic capacitor		35V 1μF	Защита от перепадов тока
C2				
C3				
Dc	DC DC Понижающий преобразователь	LM2596HVS	In: 4,5-40V Out: 3-40V 2A (3A max)	Понижает напряжение до 5V для Raspberry Pi
POW	Блок питания	Сетевой универсальный адаптер	12V 3A 5.5*2.5mm	Блок питания подключенный к коннектору питания
J1	Коннектор питания	Power Jack	12V 1.5A	Коннектор питания
L298N	DC motor driver	L298N	5-35V 36mA	Драйвер для двигателей
cam	Камера для Raspberry Pi	OV5647	1080p30, 720p60 2592×1944 5 мегапикселей	Получение видеоизображения

Pi 3B	Одноплатный компьютер Raspberry Pi	Raspberry Pi 3B+	5-3.3V, 1.5A max - 2.5A 40 пинов GPIO 4 USB, HDMI, Audio Jack 1200 МГц 64bit	Обработка данных

Таблица подключений:

Arduino Nano

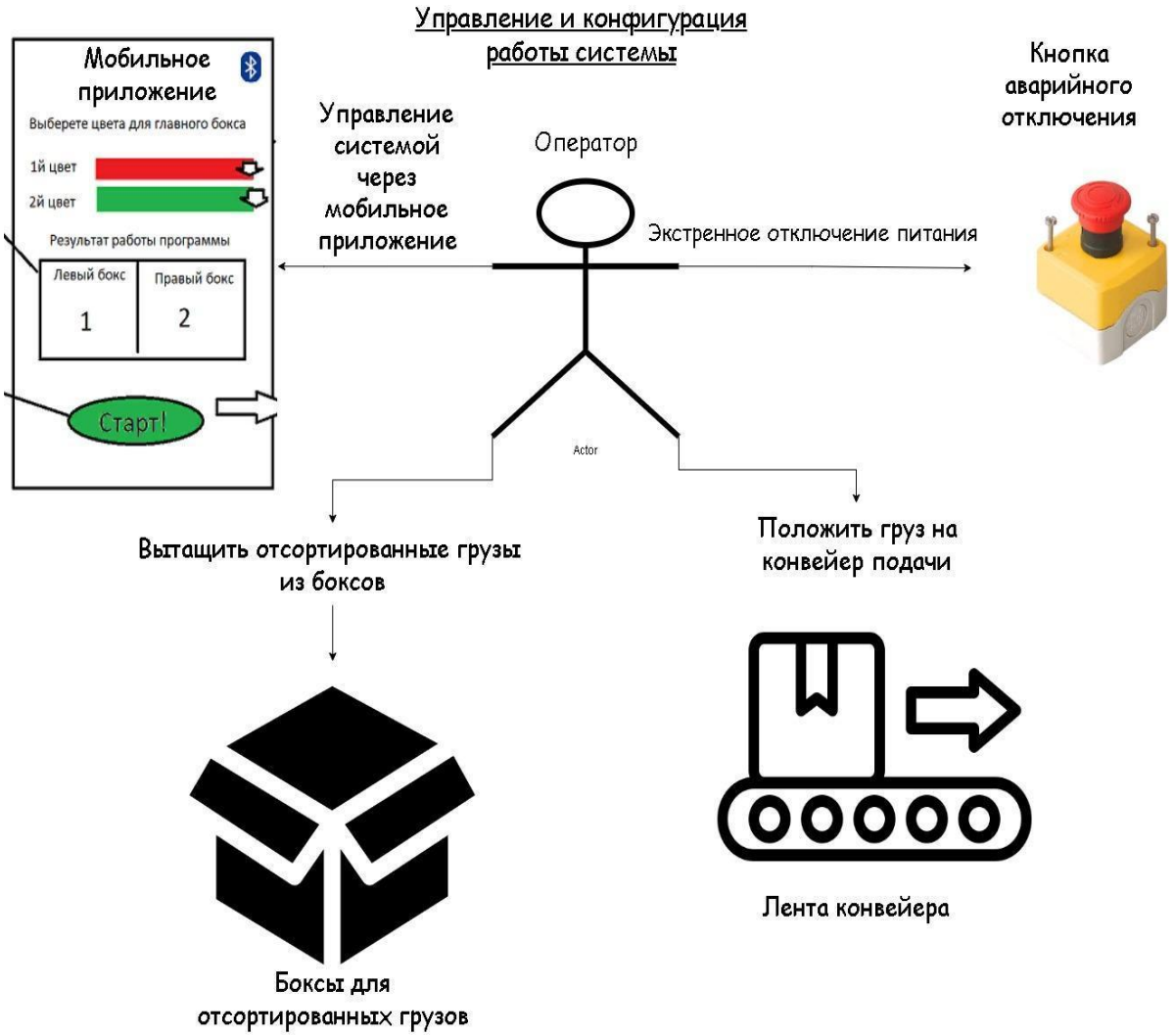
Пин	Назначение пина	Устройство, обозначение	Пин устройства	Комментарий
GND	Заземление компонентов	L298N1 HC-SR04 Конденсатор C2 Powerjack J1	GND -	Общее заземление всех компонентов
VIN	Входной пин для подключения внешнего источника	Конденсатор C2	+	Защита платы
5V	обеспечивает питание HC-SR04	HC-SR04	VSS	Необходим для подачи нужного напряжения на датчик
USB	Возможность обмена информацией с Raspberry Pi	Raspberry Pi	USB	Использование протокола UART
D10	Аналоговый пин подачи сигнала	L298N1 Bridge driver	IN1	Для работы мотора
D9	Аналоговый пин подачи сигнала	L298N1 Bridge driver	IN2	Для работы мотора
D6	Аналоговый пин подачи сигнала	L298N1 Bridge driver	IN3	Для работы мотора
D5	Аналоговый пин подачи сигнала	L298N1 Bridge driver	IN4	Для работы мотора
D4	Цифровой пин	HC-SR04	TRIG	Для работы датчика
D2	Цифровой пин	HC-SR04	ECHO	Для работы датчика

Raspberry Pi 3B

Пин	Назначение пина	Устройство	Пин устройства	Комментарий
Шина CSI	Camera Serial Interface	Raspberry Pi Cam Module	Шина CSI	Камера, разработана для Raspberry Pi. Подключается шлейфом CSI

Блок схемы и алгоритмы:

Диаграмма пользовательского взаимодействия:



Перечень пользовательского взаимодействия в текстовом формате и возможные сценарии:

Перечень пользовательских взаимодействий:

Физические действия:

Включение/выключение тумблера питания

Поставить/убрать груз ***только*** на ленту или контейнер(расчет на это)

Взаимодействие с мобильным приложением:

1. Выбрать любые цвета из предложенного перечня(подразумевает выбор нового цвета в любой момент времени);
2. Старт/Стоп для работы конвейера(подразумевает возможность “отложенной” постановки в любой момент времени);
3. Просмотр результатов сортировки(определение какой груз по цвету и сколько в каждом контейнере);

Сценарии работы устройства:

Номер действия	**Действующее лицо**	**Действие лица**	**Альтернативный сценарий, проблемы ошибки и тд, возможное решение, комментарий**
1	Пользователь	Нажал на кнопку питания на устройстве	Кнопка питания не работает
2	Пользователь	Входит в мобильное приложение (попытка подключения)	Приложение не запускается
3	Система	Включает Bluetooth подключение (включено всегда до отключения питания), **подключение есть**	**Подключение не происходит** или Bluetooth неисправен (другая техническая неполадка)
4	Пользователь	Выбирает в приложении цвет (1 для левого конвейера, 2 соответственно)	Пользователь выбрал ***одинаковые*** цвета в обоих случаях (нужна блокировка выбранного цвета в приложении) Если пользователь выбирает

			только один цвет, то по умолчанию выставляется либо рандомный, либо конкретный(срабатывает при нажатии старт)
5		Нажимает на «Старт (немедленный запуск главной ленты)	Лента не двигается, проскальзывает и др
5	Система	Запускает 1 ленту	☝
6		Готовится заметить груз в предполагаемой точке	Груз проходит мимо (неверное расстояние или угол крепежа датчика)
7		После обнаружения камерой проверяет цвет каждого груза	Цвет не может определиться **или** он не соответствует реальности (последнее не может обнаружиться программой, рекомендуется нажать Стоп пользователю)
8		После проверки цвета груз перебрасывается на 2 ленту (1 лента двигаться пока не обнаружиться новый груз, одновременно двигается 2 лента заданное время)	Если груз 1 цвета – движение 2 ленты влево, иначе вправо
9		Считает кол-во грузов по цветам	Лучше реализовать счет в малине
10	Система	Отключает конвейер или 👇	Допустим автоматическое отключение если грузы не обнаруживаются в течении 2 мин
10	Пользователь	Нажимает в приложении кнопку «Стоп»	Система завершает работу сразу после завершения необратимого действия (кнопка перестает отображаться или просто не отвечает (подает сигнал, но так как процесс уже запущен, игнорируется))
		Может запустить процесс - Старт	Система запускает работу с последнего цикла действия
11	Пользователь	Нажимает кнопку отключения от Bluetooth	

Интерфейс мобильного приложения:

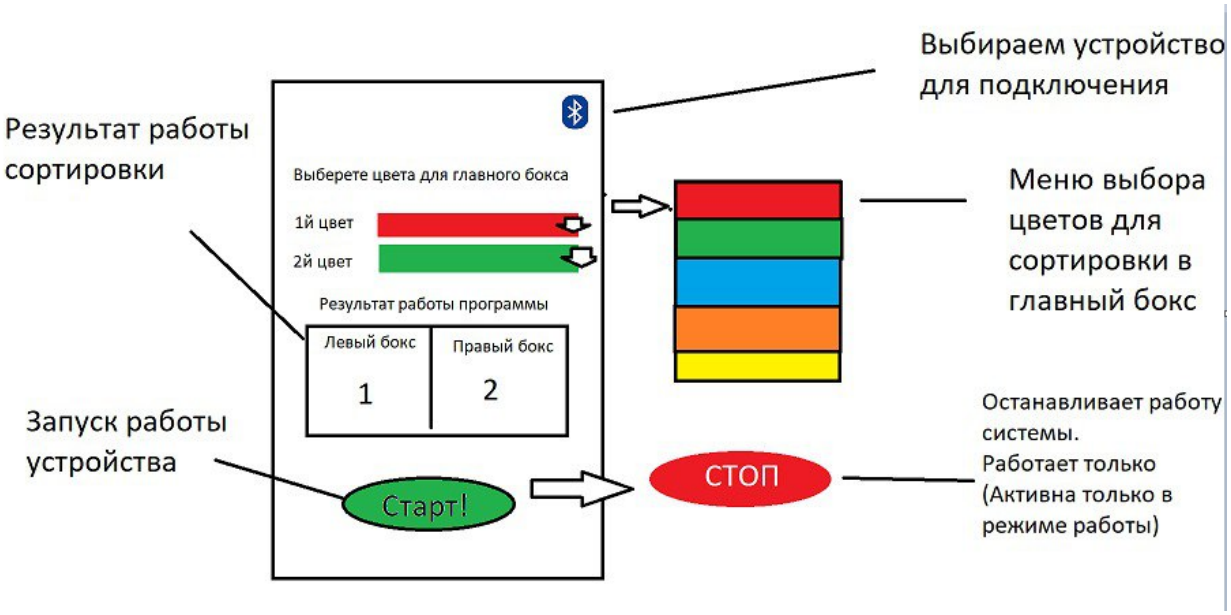
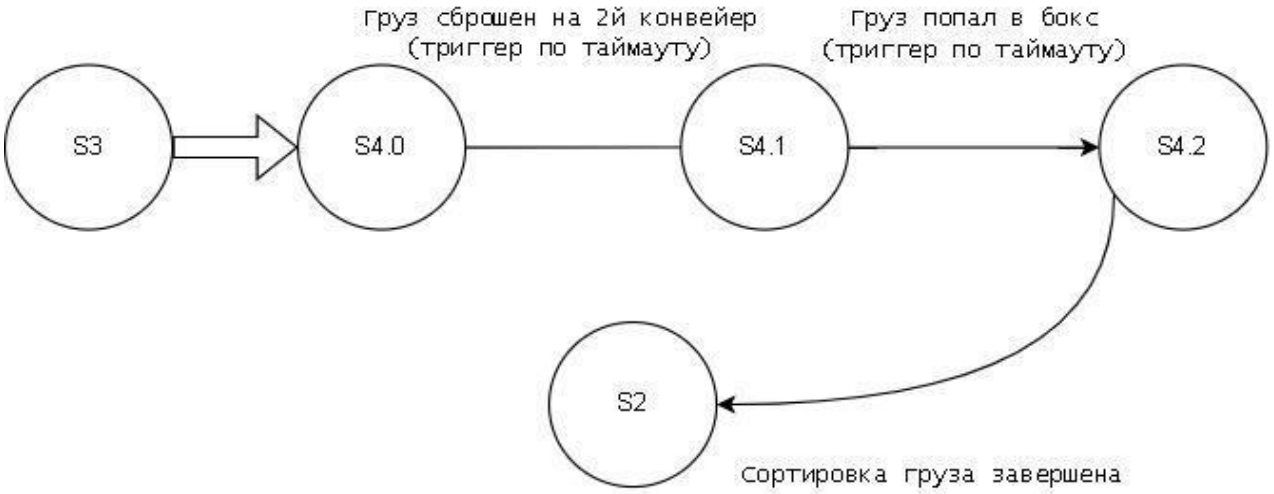
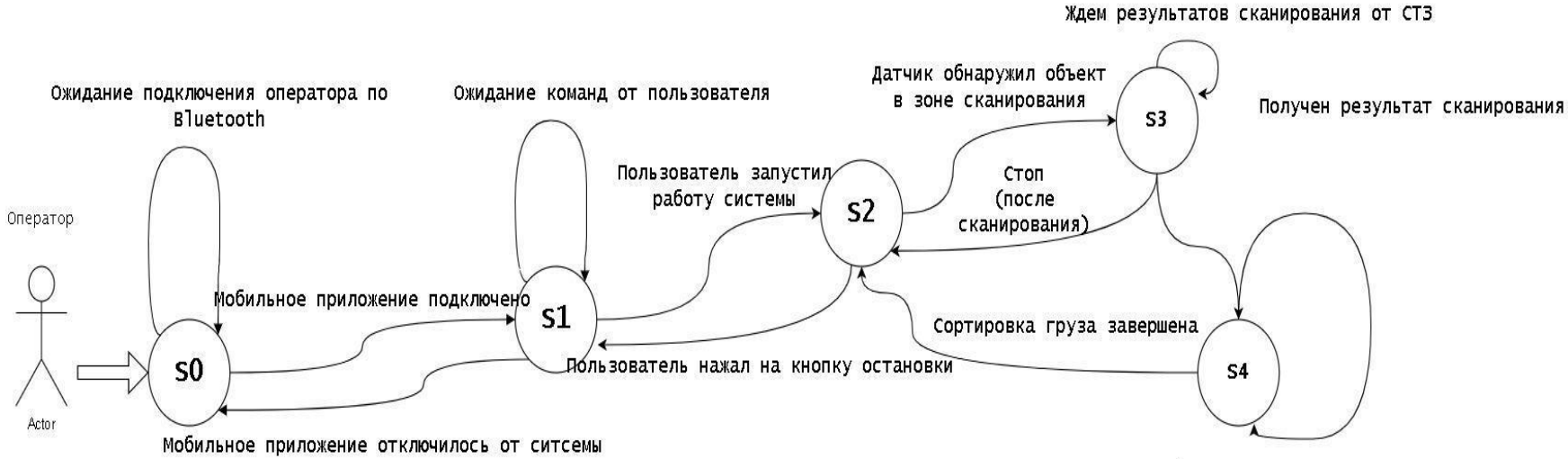


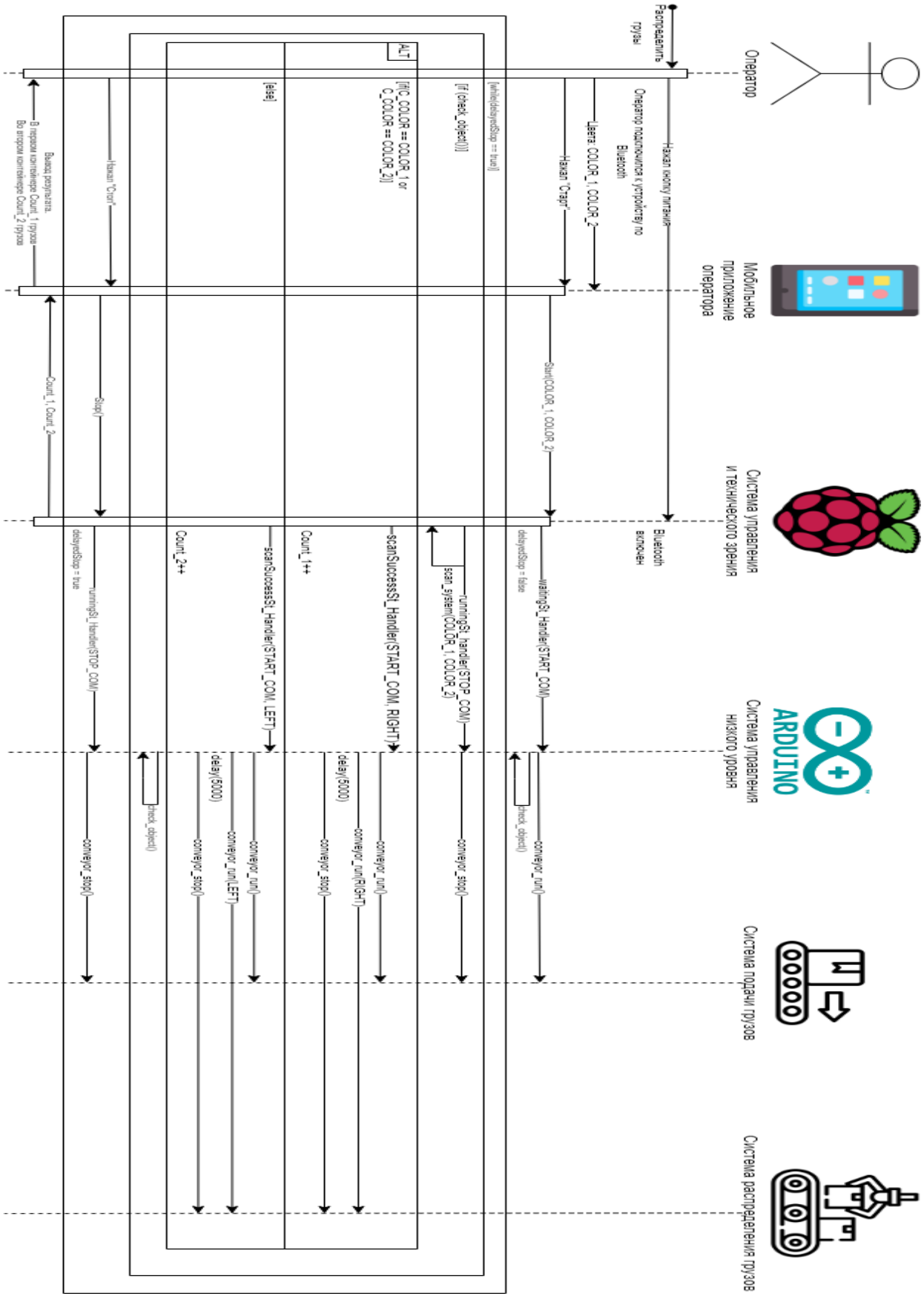
Диаграмма состояния(statemachine):



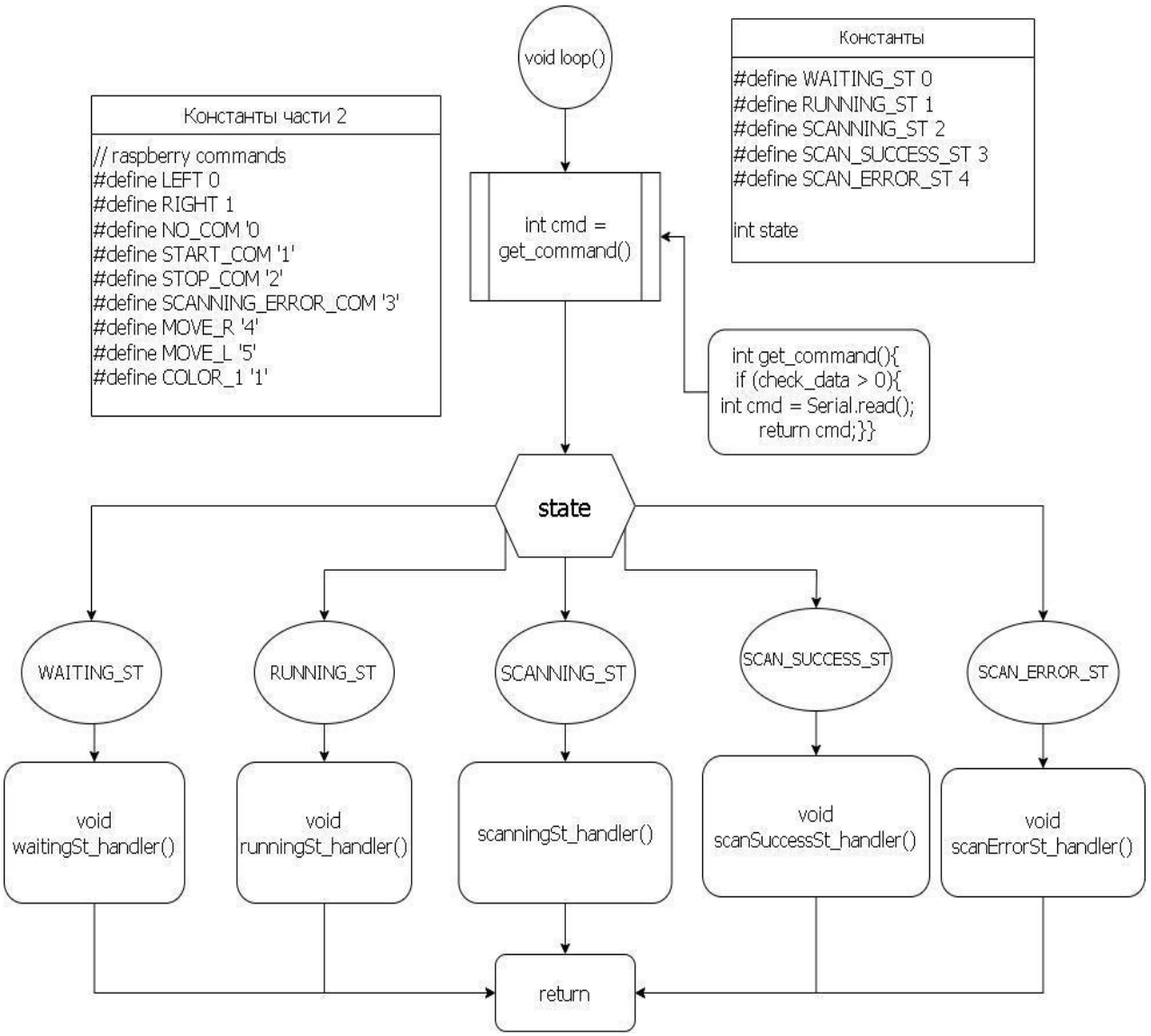


Легенда	
S0	Устройство включено и ждёт соединения с мобильным приложением (включение Bluetooth); (знач. ожидание)
S1	Устройство ждет завершения настроек в приложении и команды на запуск (ожидание пока пользователь не выберет цвет и затем нажмет Старт);(ожидание)
S2	Устройство двигает конвейер до обнаружения датчиком груза (знач. движение)
S3	Устройство определяет цвет кубика
S4	Устройство распределяет груз по контейнерам

Диаграмма последовательности(sequence):



Схематичное построение основы кода:



Исходный код:

```
#define SERIAL_SPEED 115200
#define OBJ_FIND_DISTANCE_CM 7
#define PIN_ECHO D2
#define PIN_TRIG D4
#define MOT_1_+_PIN D10
#define MOT_1_-_PIN D9
#define MOT_2_+_PIN D6
#define MOT_2_-_PIN D5
#define WAITING_ST 0
#define RUNNING_ST 1
#define SCANNING_ST 2
#define SCAN_SUCCESS_ST 3
#define SCAN_ERROR_ST 4

int state = WAITING_ST; // S0
bool delayedStop = false;
uint32_t myTimer1
////////////////////////////////////////////////////////////////////////////
void conveyor_stop(){// первая лента остановка

        digitalWrite(MOT_1_+_PIN, 0);// M3
        digitalWrite(MOT_1_-_PIN, 0);



}
void conveyor_stop_2(){// вторая лента остановка
        digitalWrite(MOT_2_+_PIN, 0);// M4
        digitalWrite(MOT_2_-_PIN, 0);
         // тайм аут
}
void conveyor_run(){//первая лента запуск
        digitalWrite(MOT_1_+_PIN, 1);
        digitalWrite(MOT_1_-_PIN, 0);
        // тайм аут
}
void conveyor_run_2_L(){// вторая лента запуск влево
        digitalWrite(MOT_2_+_PIN, 1);
        digitalWrite(MOT_2_-_PIN, 0);
        if (millis() - myTimer1 >= 6000) {
                myTimer1 = millis();
                return;
        }
}
void conveyor_run_2_R(){// вторая лента запуск вправо
        digitalWrite(MOT_2_+_PIN, 0);
        digitalWrite(MOT_2_-_PIN, 1);
```

```
        if (millis() - myTimer1 >= 6000) {
                myTimer1 = millis();
                return;
        }
}
////////////////////////////////////////////////////////////////////////////
void setup(){
        Serial.begin(SERIAL_SPEED);
        pinMode(PIN_TRIG, OUTPUT);
        pinmode(PIN_ECHO, INPUT);
        pinMode(MOT_1_+_PIN, OUTPUT);
        pinmode(MOT_2_+_PIN, OUTPUT);
        pinmode(MOT_2_-_PIN, OUTPUT);
}
////////////////////////////////////////////////////////////////////////////
void waitingSt_handler(int cmd){
        switch(cmd){
                case NO_COM:{
                        break;
                }
                case START_COM:{
                        state = RUNNING_ST;
                        break;
                }
                default:{
                        //WRONG COMMAND
                }
        }
}
////////////////////////////////////////////////////////////////////////////
void get_distance(){
        long duration, cm;
   Serial.begin (9600);
   digitalWrite(PIN_TRIG, LOW);
   delayMicroseconds(2);
   digitalWrite(PIN_TRIG, HIGH);
   delayMicroseconds(10); // В этот момент датчик будет посылать сигналы с частотой 40 КГц.
   digitalWrite(PIN_TRIG, LOW);
   duration = pulseIn(PIN_ECHO, HIGH);//  Время задержки акустического сигнала на эхолокаторе.
   cm = duration / 58 ; //  преобразовать время в расстояние
   Serial.print(cm);
   delay(300);
}
bool check_object(){
        float dist = get_distance();
        if (dist <= OBJ_FIND_DISTANCE_CM){
        return true;
        } else {
        return false;
        }
```

```
}
////////////////////////////////////////////////////////////////////////////////
void runningSt_handler(int cmd){
	if (delayedStop){
		conveyor_stop();
		state = WAITING_ST;
		return;
	}

	if (check_object()){ // если объект в зоне обнаружения
		conveyor_stop();// то остановка и переход в S3
		state = SCANNING_ST;
		return;
	}
	conveyor_run(); // работает постоянно пока первые if не сработают

	switch(cmd){
		case NO_COM:{
			break;
		}
		case STOP_COM:{
			conveyor_stop();
			state = WAITING_ST;
			break;
		}
		default:{
			//WRONG COMMAND
			break;
		}
	}
}
////////////////////////////////////////////////////////////////////////////////
void scanningSt_handler(int cmd){
	if (delayedStop){
		conveyor_stop();
		state = WAITING_ST;
		return;
	}
	switch(cmd){
		case COLOR_1:{
			state = SCAN_SUCCESS_ST;
			break;
		}
		default:{
			state = SCAN_ERROR_ST;
			break;
		}
	}
}
////////////////////////////////////////////////////////////////////////////////
```

```
void scanSuccessSt_handler(int cmd){
	switch(cmd){
		case COLOR_1:{
			conveyor_run_2_L();
			state = RUNNING_ST;
			break;
		}
		default:{ // прописать выход в S3
			conveyor_run_2_R();
			state = RUNNING_ST;
			break;
		}
	}
}
////////////////////////////////////////////////////////////////////////////////
void scanErrorSt_handler(int cmd){
	Serial.begin(9600);
	Serial.println("Ошибка сканирования, поправьте кубики");
	state = SCANNING_ST;
	return;
}
// raspberry commands
#define LEFT 0
#define RIGHT 1
#define NO_COM '0'
#define START_COM '1'
#define STOP_COM '2'
#define SCANNING_ERROR_COM '3'
#define MOVE_R		'4'
#define MOVE_L		'5'
#define COLOR_1 '1'

int check_data(){
	return Serial.available(); // Получение инфо с малины запись в функцию
}

int get_command(){
	if (check_data > 0){
		int cmd = Serial.read();// локальная смд
	}
	if (cmd < NO_COM) or  (cmd > MOVE_L) //если команда не пуста или команда не подходит
под 1 цвет, то ожидание
		cmd = NO_COM;
	return cmd;
}////////////////////////////////////////////////////////////////////////////////////////////////////////////////
void loop(){
	int cmd = get_command();
	if (cmd == STOP_COM){
		delayedStop = true;
	}
```

```
        switch(state){
                case WAITING_ST:{ //S1
                        waitingSt_handler(cmd);
                        break;
                }
                case RUNNING_ST:{ // S2
                        runningSt_handler(cmd);
                        break;
                }
                case SCANNING_ST:{ //S3
                        scanningSt_handler(cmd);
                        break;
                }
                case SCAN_SUCCESS_ST:{ //S3 successes
                        scanSuccessSt_handler(cmd);
                        break;
                }
                case SCAN_ERROR_ST:{// S3 error
                        scanErrorSt_handler(cmd);
                        break;
                }
        }
}
```

Список используемой литературы:

https://eti.su/articles/over/over_227.html
https://dzen.ru/a/XKMoPRhDDQCzHsSQ
https://dzen.ru/a/XnYheM5_X3Wqx4nS
https://ru.m.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D0%B2%D0%B5%D0%B9%D0%B5%D1%80
https://alexgyver.ru/gyvertimer/
https://voltiq.ru/raspberry-pi-arduino-serial-communication/
http://robotosha.ru/arduino/fritzing-library.html