Региональный оператор детского технопарка «Кванториум»
Государственное автономное образовательное учреждение дополнительного профессионального образования Владимирской области «Владимирский институт развития образования имени Л.И. Новиковой», детский технопарк «Кванториум-33»

ИНЖЕНЕРНАЯ КНИГА

Соревнования: **«Конкурс детских инженерных задач » - Боремся с гололедом**

Название команды: **«IceRobot 33»**

Разработала команда:

Румянцев Егор

Вергулес Кирилл

Колпаков Вячеслав

Лоллин Даниил

Лебедев Григорий

Коршунов А.И.

Владимир,2022

Содержание

Глава 1. Концепция

1.1 Концепция соревнований

Преамбула:

Несмотря на технологический прорыв последних десятилетий, до сих пор не решены многие вопросы, связанные с суровыми климатическими условиями. Проблема климатических условий особо актуальна для Российской Федерации из-за территориальных масштабов и разнообразия климатических зон. Многие регионы нашей страны находятся в местах с суровыми зимами, где и житель города миллионника, и житель районного центра с населением менее 1000 человек – каждый сталкивается с последствиями плохих погодных условий (гололеды, снежные заносы, экстремальное количество осадков и т.д.). Гололед на улицах приводит к снижению проходимости дорог, а в некоторых случаях даже к физическим травмам. Сами травмы негативно влияют на качество жизни и затраты конкретных людей, а косвенно – на потенциальные экономические издержки региона. Только в России, по статистике, травматизм, связанный с этими ситуациями, является одной из причин сезонной временной нетрудоспособности населения (травматизм из-за гололеда составляет порядка 15 %).

Задание заочного отборочного этапа конкурса: Разработать роботизированную транспортную платформу с дистанционным управлением, способную эффективно бороться с гололедом и снежными заносами.

Задача соревнований: Робот команды должен быть оснащен устройством для борьбы со снегом, гололедом и способен преодолеть

маршрут включающий в себя городскую и пересеченную местность а также участок с уклоном.

Требования к проведению контрольных тестов:

- Тестирование платформы должно проводится в городских условиях. Участникам необходимо предоставить фрагмент карты с обозначениями улиц/домов.
- Маршрут для представления результатов работы должен включать в себя повороты, движение по прямой, движение по участку дороги с уклоном. Также необходимо произвести испытания в движении по участку пересечённой местности.
- Необходимо продемонстрировать эффективность выбранного способа борьбы с наледью и снежными заносами высотой не менее 35 мм.
- Тестирование устройства для борьбы со льдом можно проводить на искусственно подготовленной поверхности льда.

Форма представления результатов выполнения задания заочного отборочного этапа конкурса:

- Видеопрезентации разработанного технического устройства.
- Запись контрольных тестов.
- Инженерная книга, включающая в себя анализ конкурсного задания, аналогов, особенности предлагаемого решения и этапы разработки платформы. Объём – не более 30 страниц, не включая приложение.

Требования к роботу:

- Время работы платформы – не менее 120 мин. В режиме активной работы - не менее 60 минут.

- Радиус действия дистанционного управления и систем связи с устройством, в городских условиях – не менее 200 метров
- Минимальный размер платформы 40 см х 30см х 30см (длина, ширина, высота), максимальный размер платформы 70 см х 60см х 60см.
- Платформа должна иметь массу не более 15 кг.
- Использование тепловых двигателей в качестве источника механической энергии запрещено.
- Требования к конструктивному исполнению не предъявляются. Требования к наличию и типу источника питания или способам приведения в действие не предъявляются. Требования к материалам изготовления платформы не предъявляются. Требования к составу комплектующих (деталей и составных частей) и происхождению комплектующих не предъявляются.

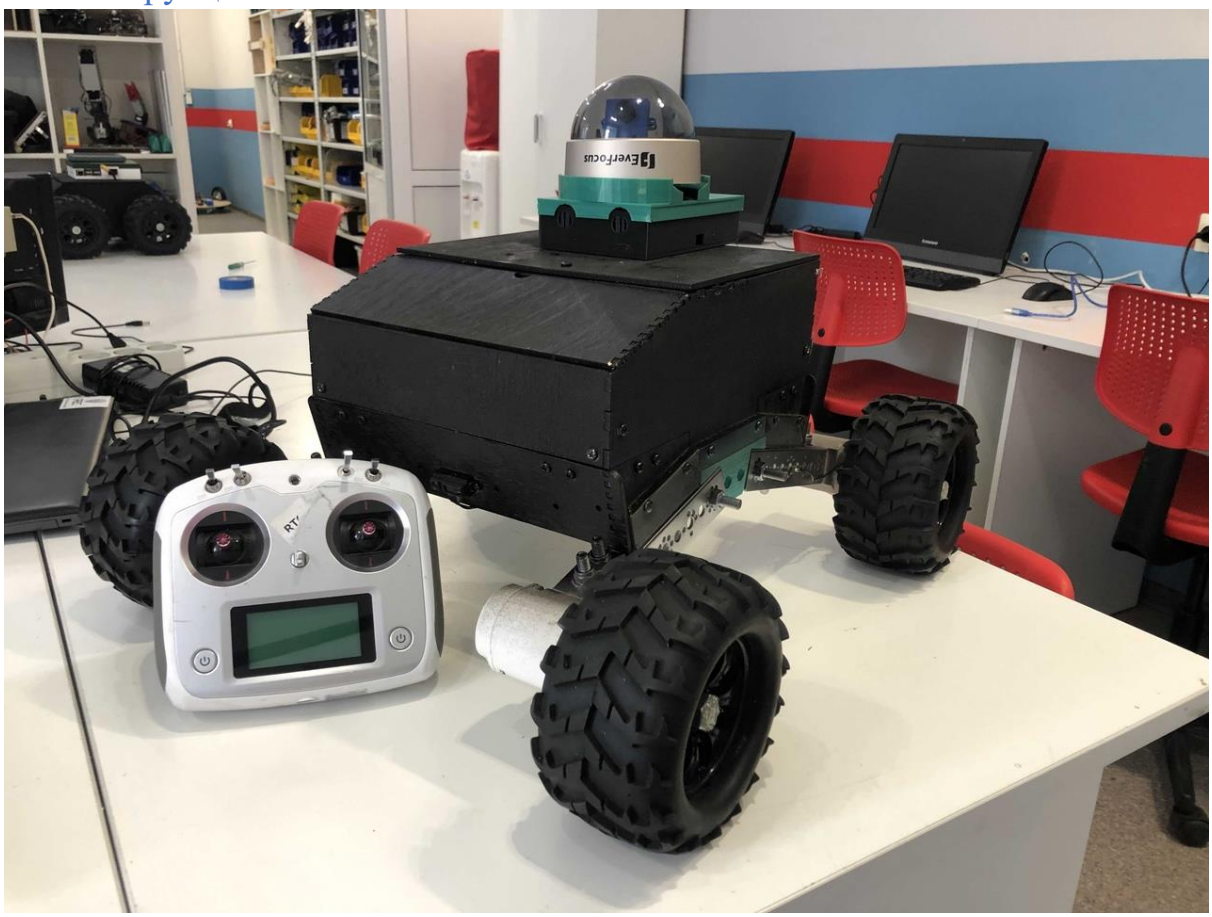
1.2 Концепция проекта

Целью проекта является разработка роботизированной транспортной платформы с дистанционным управлением, способную эффективно бороться с гололедом и снежными заносами.

Для достижения указанной цели необходимо решить следующие задачи:

- Разработать конструкцию робота
- Разработать 3D-модели робота;
- Разработать программное обеспечение, позволяющее, с помощью которого мы бы смогли управлять роботом дистанционно и считывать показания с датчиков
- Разработать инженерную книгу

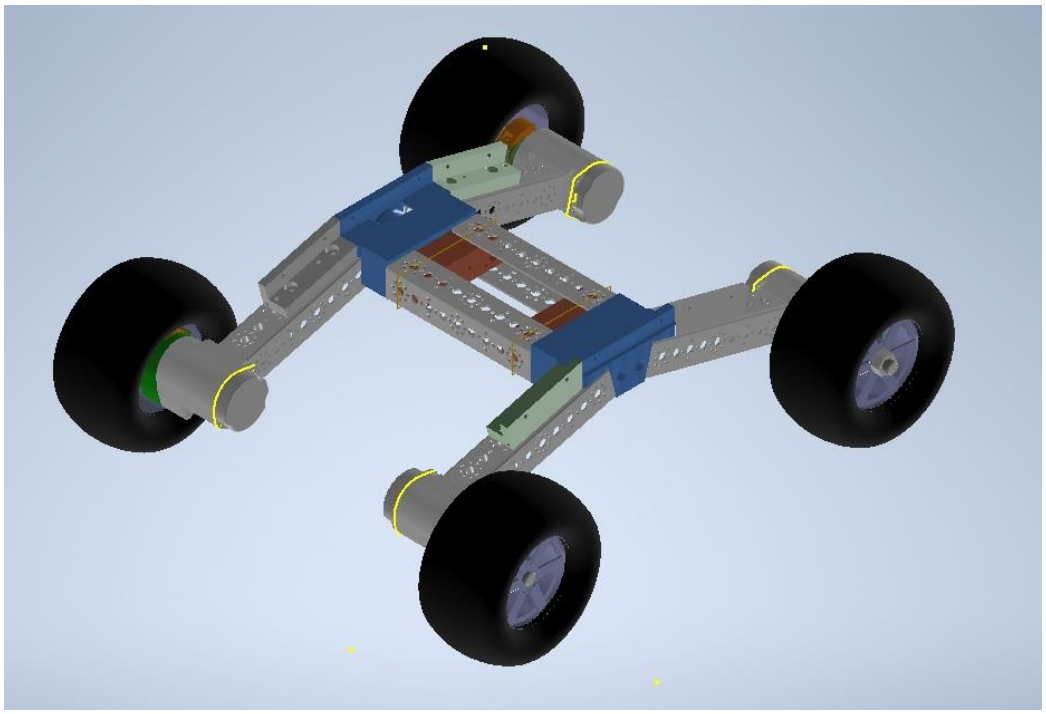
Глава 2. Конструкция



Депозитарий проекта: https://github.com/korsh624/TECHNOCAMP2022

Конструкция устройства состоит из следующих компонентов:

- Платформа для крепления основных компонентов
- Ходовая часть
- Узел электропривода
- Кузов робота
- Диск оптического датчика количества оборотов колес;
- Камера

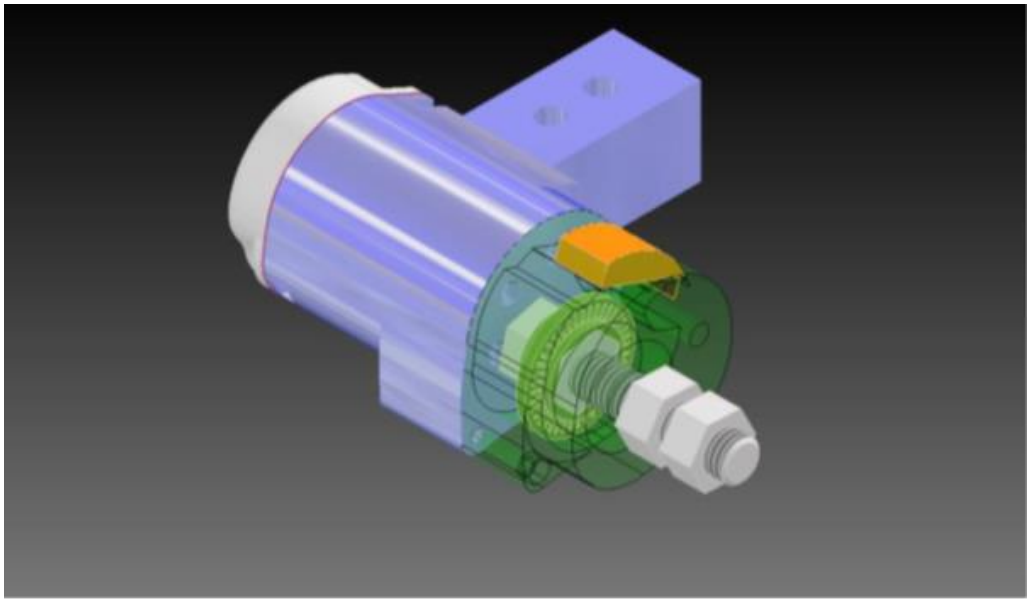


Ходовая часть робота представлена в виде Н-образной рамы, состоящей из шести П-образных профилей длиной 160 мм. Соединяются профили с помощью распечатанных на 3D-принтере деталей. Лучи наклонены вниз под углом 15° относительно горизонтали для увеличения дорожного просвета. Стороны стянуты друг с другом с помощью шпилек М8х320 и гаек соответствующего размера.

Узел электропривода состоит из:

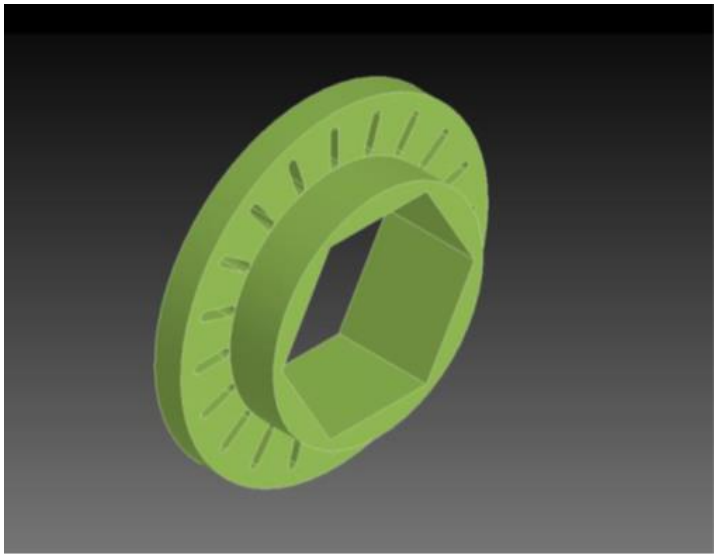
- Крепления мотора
- Втулки
- Колеса энкодера
- Кожуха энкодера
- Крышки энкодера
- Крышки мотора
- Колеса

Для приведения платформы используются коллекторные мотор-редукторы диаметром 37мм на 120 об/мин. Моторы крепятся к лучам рамы с помощью пластикового держателя на болты М8х50 и самоконтрящиеся гайки.

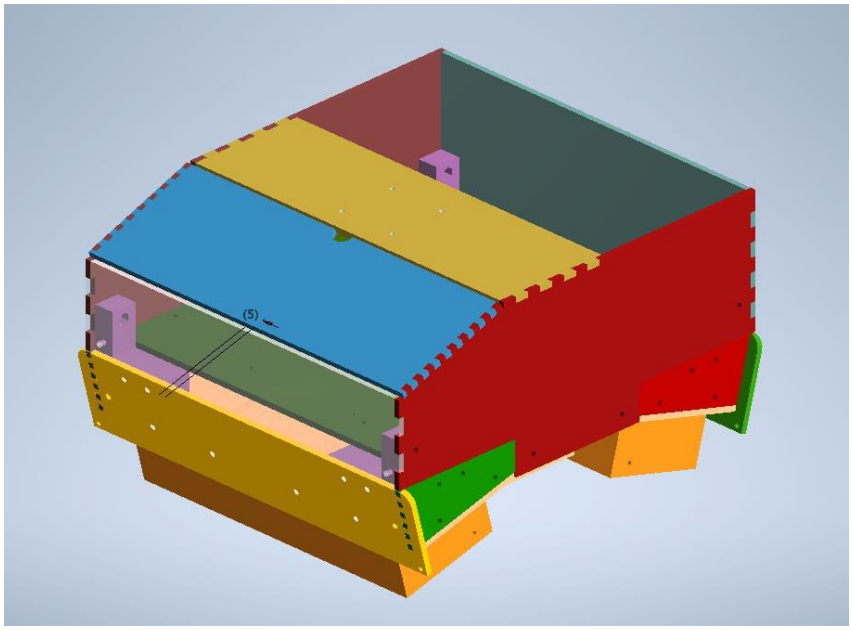


Втулки изготовлены из болтов М12х60, для которых применяется сталь Х40. В шляпке имеется глухое отверстие диаметром 6 мм и глубиной 20 мм, а. Данное отверстие необходимо для установки на выходной вал мотор-редуктора.

Диск для оптического датчика подсчета количества оборотов устанавливается на гайку, которая уже накручивается на втулку. Диск имеет 20 отверстий позволяющие изменять состояния оптического датчика. Диаметр диска составляет 35 мм. Толщина диска составляет 3 мм.



Колеса состоят из пластмассового диска, резиновой шины и поролоновой набивки. Внешний диаметр колес равен 144 мм. Колеса являются покупным элементом.



Внутри корпуса комплекс разделен на насколько отсеков:

1.Отсек с измерительными приборами и микроконтроллером

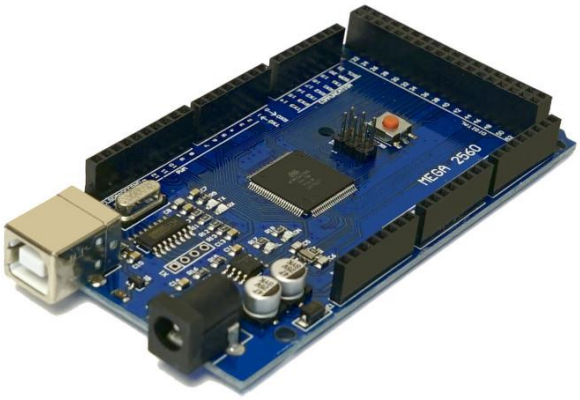
2.Отсек с драйверами и элементами питания

Под роботом расположены аккумуляторные батареи

в кол-ве 2 штук

Отсеки хранения АКБ-ей представлены в виде коробок с толщиной стенок 5 мм и выступом для крепления к раме и корпусу. Выполнены из пластика PLA на 3D-принтере. Отсеки находятся между лучами рамы, ближе к центральной перекладине.

Глава 3 Электроника

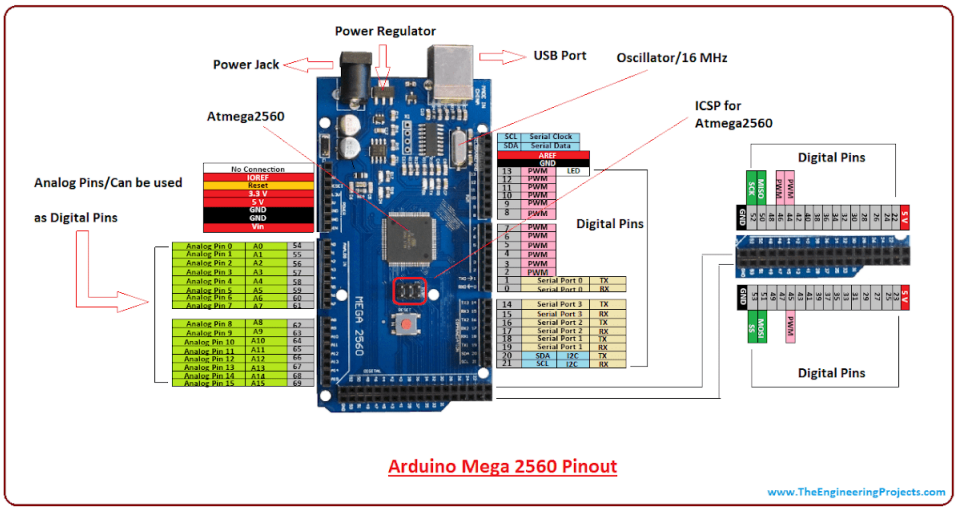


3.1 Система управления низкого уровня

3.1.1 Главный контроллер

Главным контроллером является Arduino Mega 2560.Она отвечает за передвижение робота. Мы разделили “обязанности” между разными микроконтроллерами. Часть характеристик представлены в таблице.

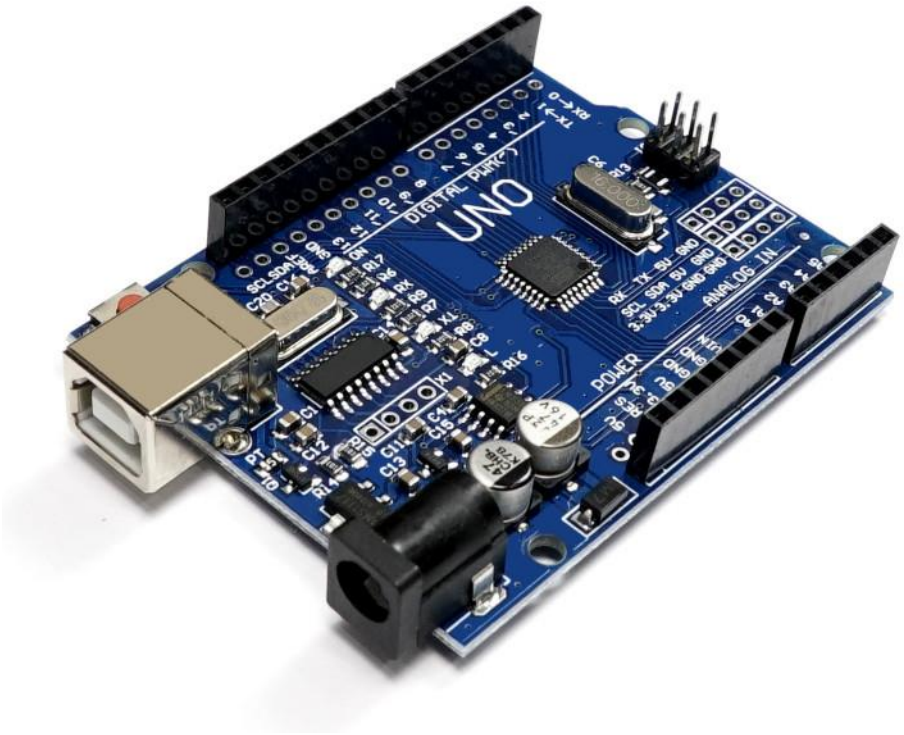
Наименование характеристик	Значение
Ядро	8-битный AVR
Ширина шины данных	10 бит
Тактовая частота	16 МГц
Объем памяти программ	256 кб
Тип памяти программ	flash
Объем SRAM	8 кб
Рабочее напряжение	5 В
Входное напряжение(рекомендуемое)	7-12 В
Входное напряжение(предельное)	6-20 В
Портов с ШИМ	15
Разрядность ШИМ	8 бит



Данный контроллер отвечает за управление движением робота. (Программа №3) Сигнал поступает на приемник с пульта, по протоколу i-bus на интерфейс UART3. Полученные данные ограничиваются и преобразовываются в значения для PWM портов, к которым подключены драйверы двигателей приводящие в движение моторы.

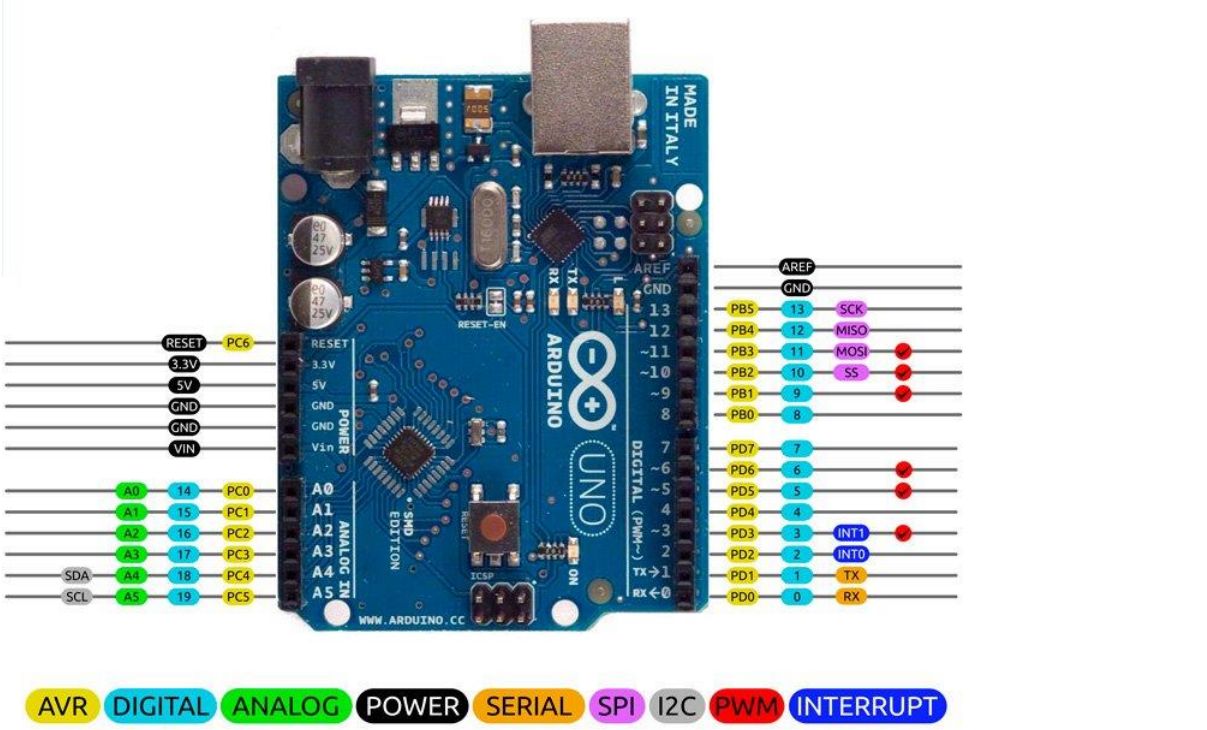
3.1.2 Arduino UNO

Также в нашем роботе присутствует Arduino UNO, к которой подключены все датчики и измерительные приборы. Мы используем ее только для считывания показаний с МПУ, контроллера заряда батарей и GPS-крекера. (Программа №1)



Наименование характеристик	Значение
Ядро	8-битный AVR
Ширина шины данных	10 бит
Тактовая частота	16 МГц
Объем памяти программ	32 кб
Тип памяти программ	flash
Объем RAM	2 кб
Рабочее напряжение	5 В
Входное напряжение(рекомендуемое)	7-12 В
Входное напряжение(предельное)	6-20 В
Портов с ШИМ	15

Разрядность ШИМ	8 бит



3.1.3 Драйвер управления двигателями постоянного тока

Для управления электродвигателями постоянного тока используется драйвер L298N.

Наименование	Значение
Размер платы	43 мм x 43.75 мм
Питание платы	5 В
Питание двигателей	2.5-46 В
Рабочий ток на канал	до 2 А
Логический "0" управляющего напряжения	0…1.5 В
Логическая "1" управляющего напряжения	2.3…7 В
Максимальная частота управляющего ШИМ	до 5 кГц
Защита от перегрева	Есть

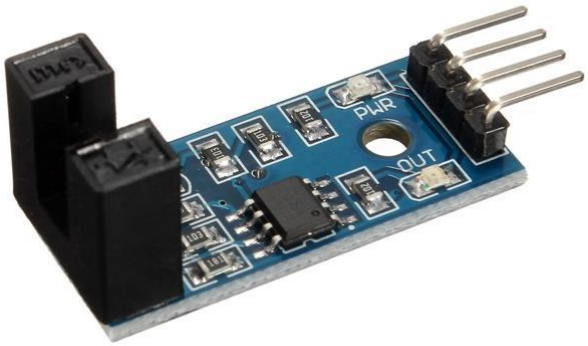
L298N содержит сразу два драйвера для управления электродвигателями (четыре независимых канала, объединенных в две пары). Имеет две пары входов для управляющих сигналов и две пары выходов для подключения электромоторов. Кроме того, у L298N есть два входа для включения каждого из драйверов. Эти входы используются для управления скоростью вращения электромоторов с помощью широтно-импульсной модуляции сигнала (ШИМ).

Обозначение	Наименование
VCC	положительный контакт питания платы (управляющей логики) 5В
5-36V	положительный контакт питания электродвигателей 5 – 36 В
GND	отрицательный контакт питания
+5V	контакт, на котором формируется 5 вольт встроенным в плату стабилизатором напряжения, отключается размыкание контактов помеченные JUMPER_5
IN1,IN2,IN3, IN4	управление направлением вращения и скоростью двигателей

Одна микросхема L298N способна управлять двумя двигателями по 2А каждый двигатель, а если задействовать параллельное включение для одного двигателя, то можно поднять максимальный ток до 4А.

3.1.4 Оптический датчик числа оборотов колеса

Для определения количества оборотов колеса используются бесконтактные оптические датчики положения RKP-MWES-LM393 (второе название FC-03). Они состоят из оптического излучателя и фотоприемника – оптической пары ITR9608.



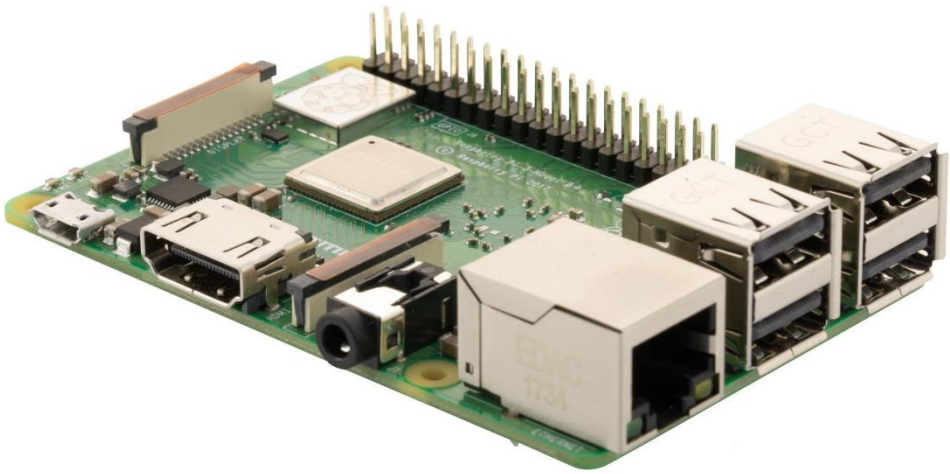
Световой поток от излучателя попадает на фотоприемник, что вызывает определенное состояние датчика. а. Наличие непрозрачного объекта на пути светового луча приводит к изменению светового потока на фотоприемнике, а значит и к другому состоянию датчика.

Непрозрачным объектом выступает смоделированное образцовый диск с отверстиями, устанавливаемый на вал мотора (во внутренней части колеса).

3.2 Система управления высокого уровня

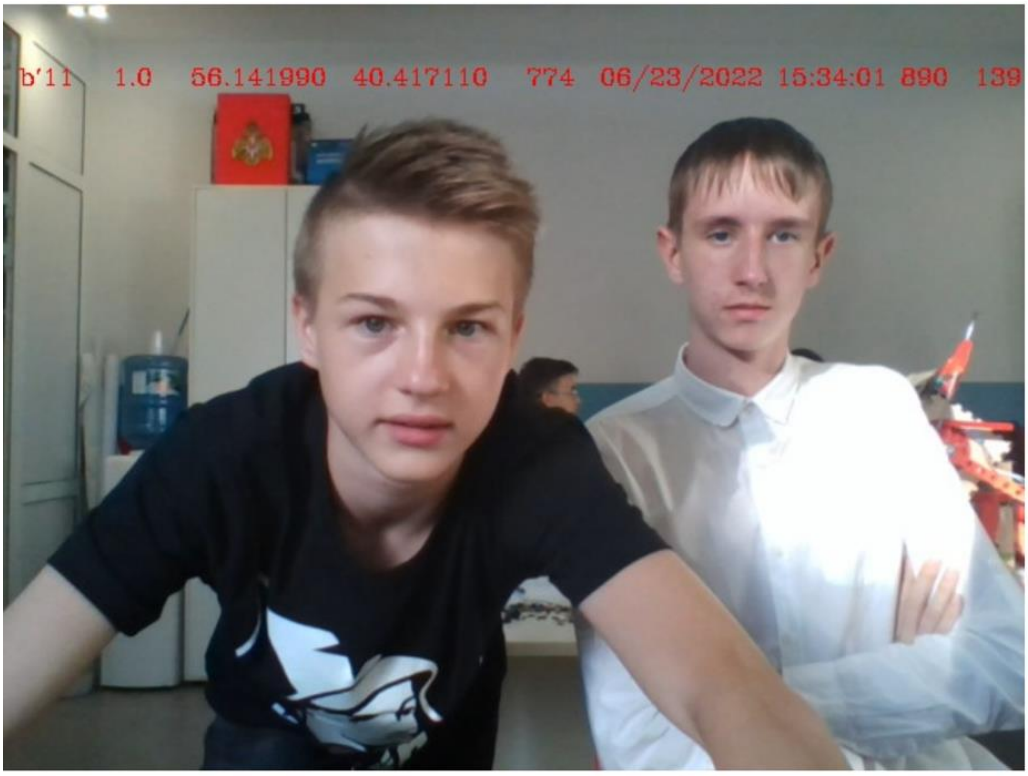
3.2.1 Одноплатный компьютер Raspberry pi3 B

Одноплатный компьютер мы используем для стриминга и вывода телеметрии поверх изображения.



Наименование характеристики	Значение характеристики
Процессор	64-битный четырехъядерный Cortex-A53 (ARM v8) с тактовой частотой 1.2 ГГц
Оперативная память	1ГБ LPDDR2 SDRAM (900 MHz)
Постоянная память	microSD
Интерфейсы взаимодействия	DSI, CSI, 40-pin GPIO
Сетевые интерфейсы	WIFI 802.11n + Bluetooth 4.1 Low Energy
Графический процессор	2-ядерный сопроцессор VideoCore IV 3D

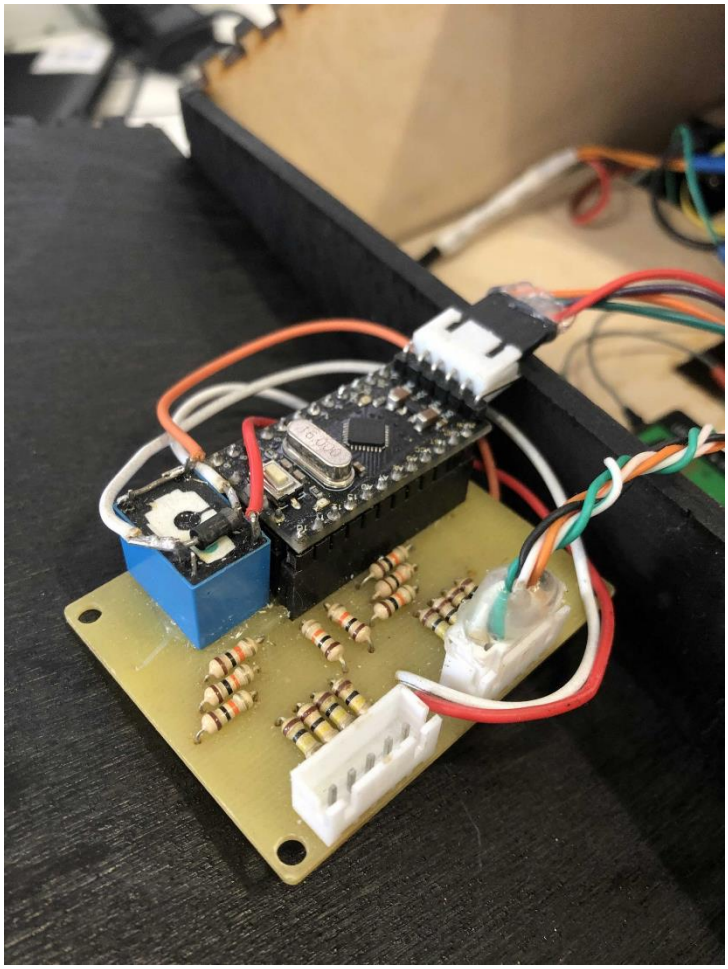
Для того чтобы выводить видео изображение и данные(координаты, время, заряды аккумуляторов) наложенные поверх него, мы используем код написанный на Python(Программа №2). Для вывода телеметрии и трансляции видео мы используем фреймворк Flask. Для обработки изображения и наложения телеметрии используется библиотека OpenCV. Когда мы запускаем код на Python, он запускает свой веб сайт, на котором выводится картинка с показателями и трансляцией видео.



3.2.2 Контроллер заряда батарей

В нашем роботе присутствует самодельный контроллер заряда батарей. (Программа №4).

Благодаря нему мы знаем точный заряд на каждой ячейки аккумулятора



Наше устройство состоит из:

- Из 8 резисторов номиналом 10кОм
- Из 8 резисторов номиналом 100кОМ
- Платы которая фиксирует заряд

3.2.3 GPS трекер

.Для получение полезных данных(точное время, дату и координаты нашего робота)используется модуль Ublox GPS на базе микросхемы Neo M8N. Режимы работы ГЛОНАСС/GPS (по умолчанию), GLONASS, GPS. Рабочая температура –40…+85 °C.

3.2.4 RPI-Камера

Мы используем RPI-камеру для того чтобы видеть, что происходит впереди робота. Наша камера находится в специальном корпусе для снижения риска повреждений и попаданий пыли

Наименование	Характеристика
Модель	Caddx Ant lite
Изображение	4:3
FOV	165 градусов
Горизонтальное разрешение:	1200 ТВЛ
Объектив	1,8 мм/F2.5
WDR	Global WDR

3.3 Система питания

Бортовым источником напряжения комплекса служат две LiPo аккумуляторных батареи, имеющие следующие характеристики:

Наименование	Характеристика
Тип ячеек	LiPo
Количество ячеек	4 шт.
Конфигурация ячеек	4S (4 ячейки соединены последовательно)
Емкость батареи	5000 mAh
Заявленная токоотдача	60C (числовой коэффициент умножается на емкость батареи)

Расчет времени работы комплекса при пиковой нагрузке на каждом из компонентов

Наименование	Токопотребление, Ампер
Драйвера двигателя	3+3
Одноплатный компьютер	2
GPS Ublox Neo M8N	0.04
Arduino Mega	0.07
Arduino Uno	0.04
МПУ	0.03
Дальномер Sharp	0.04

Так как в нашем роботе 2 аккумуляторных батареи

Время работы (h) = Емкость батареи (Ah) / Суммарное токопотребление компонентов (A)

$10/6+2+0.04+0.07+0.04+0.03+0.04 \approx 0{,}97$ часа ≈ 58.2 минут

4. Программное обеспечение

Это код написанный в Arduino IDE, с помощью которого мы считываем показания с МПУ, с GPS трекера и системы контроля заряда АКБ

Программа №1

```
#include <SharpIR.h>
#include <TinyGPSPlus.h>
#include <SoftwareSerial.h>

static const int RXPin = 9, TXPin = 8;
static const uint32_t GPSBaud = 9600;
#define sensor A0

TinyGPSPlus gps;



SoftwareSerial ss(RXPin, TXPin);

void setup()
{
  Serial.begin(9600);

  ss.begin(GPSBaud);
}

void loop()
{
   float volts = analogRead(sensor)*0.0048828125;
  int distance = 13*pow(volts, -1);
  delay(100);

  if (distance <= 30){
    Serial.println(distance);
```

```
  }
// Дальше GPS!

  static const double LONDON_LAT = 51.508131, LONDON_LON = -0.128002;

  printInt(gps.satellites.value(), gps.satellites.isValid(), 5);
  printFloat(gps.hdop.hdop(), gps.hdop.isValid(), 6, 1);
  printFloat(gps.location.lat(), gps.location.isValid(), 11, 6);
  printFloat(gps.location.lng(), gps.location.isValid(), 12, 6);
  printInt(gps.location.age(), gps.location.isValid(), 5);
  printDateTime(gps.date, gps.time);
  printFloat(gps.altitude.meters(), gps.altitude.isValid(), 7, 2);
  printFloat(gps.course.deg(), gps.course.isValid(), 7, 2);
  printFloat(gps.speed.kmph(), gps.speed.isValid(), 6, 2);
  printStr(gps.course.isValid() ? TinyGPSPlus::cardinal(gps.course.deg()) : "*** ", 6);

  printInt(gps.charsProcessed(), true, 6);
  printInt(gps.sentencesWithFix(), true, 10);
  printInt(gps.failedChecksum(), true, 9);
  Serial.println();

  smartDelay(1000);

  if (millis() > 5000 && gps.charsProcessed() < 10)
    Serial.println(F("No GPS data received: check wiring"));
}

static void smartDelay(unsigned long ms)
{
  unsigned long start = millis();
  do
  {
    while (ss.available())
      gps.encode(ss.read());
```

```
  } while (millis() - start < ms);
}

static void printFloat(float val, bool valid, int len, int prec)
{
  if (!valid)
  {
    while (len-- > 1)
      Serial.print('*');
    Serial.print(' ');
  }
  else
  {
    Serial.print(val, prec);
    int vi = abs((int)val);
    int flen = prec + (val < 0.0 ? 2 : 1);
    flen += vi >= 1000 ? 4 : vi >= 100 ? 3 : vi >= 10 ? 2 : 1;
    for (int i=flen; i<len; ++i)
      Serial.print(' ');
  }
  smartDelay(0);
}

static void printInt(unsigned long val, bool valid, int len)
{
  char sz[32] = "******************";
  if (valid)
    sprintf(sz, "%ld", val);
  sz[len] = 0;
  for (int i=strlen(sz); i<len; ++i)
    sz[i] = ' ';
  if (len > 0)
    sz[len-1] = ' ';
  Serial.print(sz);
```

```
  smartDelay(0);
}

static void printDateTime(TinyGPSDate &d, TinyGPSTime &t)
{
  if (!d.isValid())
  {
    Serial.print(F("********** "));
  }
  else
  {
    char sz[32];
    sprintf(sz, "%02d/%02d/%02d ", d.month(), d.day(), d.year());
    Serial.print(sz);
  }

  if (!t.isValid())
  {
    Serial.print(F("******** "));
  }
  else
  {
    char sz[32];
    sprintf(sz, "%02d:%02d:%02d ", t.hour(), t.minute(), t.second());
    Serial.print(sz);
  }

  printInt(d.age(), d.isValid(), 5);
  smartDelay(0);
}

static void printStr(const char *str, int len)
{
  int slen = strlen(str);
```

```
  for (int i=0; i<len; ++i)
    Serial.print(i<slen ? str[i] : ' ');
  smartDelay(0);
}
```

Это код написанный на Python , благодаря нему мы выводим изображение и поверх изображения данные с Arduino UNO(время, даты, координаты).

Изображение можно увидеть на странице 18.

Программа №2

```
import cv2
import serial
import time
from flask import Flask, render_template, Response, request, redirect, url_for
data='open'
read="___"

data='open'
arduino = serial.Serial('COM14',9600)
cap=cv2.VideoCapture(0)
print("connected camera")
def sendmessage():
    count=0
    global read
    while (count<1):
        arduino.write(data.encode())
        print ("Send message")
        read=str(arduino.readline())
        print(read)
        time.sleep(1)
        count=count+1
    count=0

app = Flask(__name__)
def gen_frames():
```

```
    while True:
        success,frame=cap.read()
        font = cv2.FONT_HERSHEY_COMPLEX
        if read!="___":
            print("print teleometry")
            cv2.putText(frame, read, (10, 50), font, 0.7, color=(255, 255, 255), thickness=2)
        if not success:
            break
        else:
            ret, buffer = cv2.imencode('.jpg', frame)
            frame = buffer.tobytes()

            yield (b'--frame\r\n'
                   b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')
@app.route('/')
def index():
    return render_template('index.html')
@app.route('/video_feed')
def video_feed():
    return Response(gen_frames(), mimetype='multipart/x-mixed-replace; boundary=frame')

@app.route('/update_telemetry')
def update_telemetry():
    sendmessage()
    return ("nothing")

if __name__ == "__main__":
    app.run(host='0.0.0.0', debug=False)
```

Программа № 3

```
#include "FlySkyIBus.h"

//правое переднее   f11    r10

//правое заднее   f12    r9
```

```
//левое переднее   f4    r7
//левое заднее   f6    r5
#define rff 11
#define rfr 10
#define rrf 12
#define rrr 9
#define lff 7
#define lfr 4
#define lrf 6
#define lrr 5
int x = 0;
int y = 0;
int b = 0;
int c = 0;
int r = 0;
int l = 0;
void setup()
{
  Serial.begin(9600);
  IBus.begin(Serial3);
  for (int i = 2; i < 14; i++) {
    pinMode(i, OUTPUT);
    digitalWrite(i, 0);
  }
  pinMode(22,OUTPUT);
  pinMode(24,OUTPUT);
```

```
  pinMode(26,OUTPUT);
  pinMode(28,OUTPUT);
  digitalWrite(22,0);
  digitalWrite(24,0);
  digitalWrite(26,0);
  digitalWrite(28,0);
  delay(2000);
}

void loop()
{
  IBus.loop();
  x = IBus.readChannel(0);
  y = IBus.readChannel(1);
  b = IBus.readChannel(6);
  c = IBus.readChannel(7);
  x = map(x, 1000, 2000, -255, 255);
  y = map(y, 1000, 2000, -255, 255);
  b = map(b, 1000, 2000, -255, 255);
  c = map(c, 1000, 2000, -255, 255);
  if ((x < 50) && (x > -50)) {
    x = 0;
  }
  if (x > 150) {
    x = 255;
  }
```

```
if (x < -150) {
  x = -255;
}
if ((y < 50) && (y > -50)) {
  y = 0;
}
if (y > 150) {
  y = 255;
}
if (y < -150) {
  y = -255;
}
if ((b < 50) && (b > -50)) {
  b = 0;
}
if (b > 150) {
  b = 255;
}
if (b < -150) {
  b = -255;
}
if ((c < 50) && (c > -50)) {
  c = 0;
}
if (c > 150) {
  c = 255;
```

```
}
if (c < -150) {
  c = -255;
}
l = x + y;
r = y - x;
if ((r < 50) && (r > -50)) {
  r = 0;
}
if (r > 150) {
  r = 255;
}
if (r < -150) {
  r = -255;
}
if ((l < 50) && (l > -50)) {
  l = 0;
}
if (l > 150) {
  l = 255;
}
if (l < -150) {
  l = -255;
}
if(l>0){
  digitalWrite(lfr,0);
```

```
  digitalWrite(lrr,0);
  analogWrite(lff,l);
  analogWrite(lrf,l);
 } else{
  digitalWrite(lff,0);
  digitalWrite(lrf,0);
  analogWrite(lfr,-l);
  analogWrite(lrr,-l);
 }
if(l==0){
  digitalWrite(lff,0);
  digitalWrite(lrf,0);
  digitalWrite(lfr,0);
  digitalWrite(lrr,0);
 }

  if(r>0){
  digitalWrite(rfr,0);
  digitalWrite(rrr,0);
  analogWrite(rff,r);
  analogWrite(rrf,r);
 } else{
  digitalWrite(rff,0);
  digitalWrite(rrf,0);
  analogWrite(rfr,-r);
  analogWrite(rrr,-r);
```

```
}
if(r==0){
  digitalWrite(rff,0);
  digitalWrite(rrf,0);
  digitalWrite(rfr,0);
  digitalWrite(rrr,0);
}

if(b>0){
  digitalWrite(22,0);
  digitalWrite(24,1);
}else{
  digitalWrite(24,0);
  digitalWrite(22,1);
}
if(b==0){
  digitalWrite(24,0);
  digitalWrite(22,0);
}

if(c>0){
  digitalWrite(26,0);
  digitalWrite(28,1);
}else{
  digitalWrite(28,0);
  digitalWrite(26,1);
```

```
 }
 if(c==0){
   digitalWrite(28,0);
   digitalWrite(26,0);
 }
 Serial.print("  x=");
 Serial.print(x);
 Serial.print("  y=");
 Serial.print(y);
 Serial.print("  b=");
 Serial.print(b);
 Serial.print("  c=");
 Serial.print(c);
 Serial.println();
 delay(300);

}
```

Программа №4

```
float data[8];
String message= "";
void setup() {

  Serial.begin(9600);

}
```

```
void loop() {
  String message= "";
  for (int i=0; i<4;i++){
    data[i]=analogRead(i);
    float s=18.8*(i+1);
    data[i]=data[i]/s;
//    Serial.print("cel ");
//    Serial.print(i);
//    Serial.print("= ");
//    Serial.print(data[i]);
//    Serial.print(" ");
    message=message + "cel-"+String(i+1)+" = "+String (data[i])+" ";
  }
  message=message +"_________________";
  for (int i=4; i<8;i++){
    data[i]=analogRead(i);
    float s=18.8*(i-3);
    data[i]=data[i]/s;
//    Serial.print("cel ");
//    Serial.print(i);
//    Serial.print("= ");
//    Serial.print(data[i]);
//    Serial.print(" ");
    message=message + "cel-"+String(i-3)+" = "+String (data[i])+" ";
  }
```

```
  Serial.println(message);


}
```