Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«Нижегородский государственный университет им. Н.И. Лобачевского»

Физический факультет

Кафедра информационных технологий в физических исследованиях

**Отчет по научно-исследовательской практике**

Отчёт выполнили

студенты 1 курса магистратуры группы 05172м

**Попутникова Е.В.,**

**Зюзин А.П.,**

**Леонов С.В.,**

**Герасимов А.Ю.,**

**Руденко А. П.,**

**Сорокина М.Н.,**

**Кочетков А.А.**

Научный руководитель:

доцент кафедры ИТФИ,

к.ф.-м.н. **Минеев С.А.**

Нижний Новгород

2018 г.

**Содержание**

[Введение 3](#_Toc515995153)

[1. Навигация 4](#_Toc515995154)

[1.1. Цель работы: 4](#_Toc515995155)

[1.2.Задачи: 4](#_Toc515995156)

[1.3.Результаты: 4](#_Toc515995157)

[1.3.1.Освоение технологии создания прошивок и программ для полетного контроллера Pixhawk. 4](#_Toc515995158)

[1.3.1.1. Pixhawk 4](#_Toc515995159)

[1.3.1.2. Предварительные настройки. Установка Toolchain 4](#_Toc515995160)

[1.3.1.3. Сборка прошивки. Тестовый проект 4](#_Toc515995161)

[1.3.1.4. Работа с сообщениями uORB 5](#_Toc515995162)

[1.3.2.Подготовка предложений по интеграции модулей RTK с платформой Pixhawk. 8](#_Toc515995163)

[1.3.Заключение 12](#_Toc515995164)

[2. Система управления электроприводами RINF 13](#_Toc515995165)

[2.1. Цель: 13](#_Toc515995166)

[2.3.Этапы разработки: 13](#_Toc515995167)

[2.3.1.Первый этап 13](#_Toc515995168)

[2.3.2.Этап второй 14](#_Toc515995169)

[2.3.Заключение: 16](#_Toc515995170)

[3. Подсистема управления движением (CNTR) 25](#_Toc515995171)

[3.1.Постановка задачи 25](#_Toc515995172)

[3.2.Описание 2D-имитатора роботизированной работы RINF 25](#_Toc515995173)

[3.3.Описание имитатора полетного контроллера Pixhawk и интеграция в него 2D-имитатора 25](#_Toc515995174)

[3.4.Моделирование алгоритма ручного движения 26](#_Toc515995175)

[3.5.Выводы 28](#_Toc515995176)

[4. Электро-механическая подсистема (ELMH) 29](#_Toc515995177)

[4.1.Цель 29](#_Toc515995178)

[4.2.Задачи: 29](#_Toc515995179)

[4.3.Результаты 29](#_Toc515995180)

[4.1.1. 3D модель: 30](#_Toc515995181)

[4.1.2.Примеры чертежей: 33](#_Toc515995182)

[4.1.3.Перечень конструктивных элементов и материалов 38](#_Toc515995183)

[4.2.Заключение: 39](#_Toc515995184)

[5. Сенсорно-сигнальная подсистема(SSRL) 40](#_Toc515995185)

[5.1. Цель 40](#_Toc515995186)

[5.2. Задачи 40](#_Toc515995187)

[5.3. Результаты 40](#_Toc515995188)

[5.4. Выводы 46](#_Toc515995189)

[Литература 47](#_Toc515995190)

# Введение

Современные робототехнические автономные транспортные системы представляют собой сложные многокомпонентные программно-аппаратные комплексы. Создание подобных комплексов требует навыков:

- разработки схем и чертежей конструктивных и электрических компонентов;

- разработки  электронного оборудования (систем электропитания, управления)

- разработки программного обеспечения различного назначения (базового и прикладного).

- проведения испытаний и экспериментальных исследований сложных технических систем;

- организации взаимодействия в проектной группе;

- владения профессиональными инструментами разработчика.

Развитие перечисленных навыков является основной целью научно-исследовательской практики, в рамках которой создана студенческая проектная группа по разработке мобильного автономного робототехнического комплекса RINF, представляющего собой скоростную инвалидную коляску с возможностью трансформации из положения "сидя" в положение "полулежа" с фиксацией промежуточных состояний.

В рамках научно-исследовательской практики 2018 г. основными задачами проектной группы были:

* Освоение технологии создания прошивок и программ для полетного контроллера Pixhawk (Попутникова Е.В., Зюзин А.П.);
* Создание системы управления электроприводами электротранспортной тележки RINF (Леонов С.В.);
* Создание программного обеспечения для получения из 3D сцены распределение дальности по азимутальному углу.(Кочетков А.А.);
* Доработка 3D-модели электрической тележки RINF в части размещения и крепления источников питания и контроллеров управления двигателями.(Руденко А.П., Сорокина М.Н.);
* Разработка алгоритма управления движением (Герасимов А.Ю.).

Общее руководство практикой: Минеев С.А. (доцент каф.ИТФИ физического факультета ННГУ им. Н.И. Лобачевского).

# Навигация

## 1.1. Цель работы:

1. Освоение технологии создания прошивок и программ для полетного контроллера Pixhawk.
2. Оценка возможностей использования модулей RTK совместно с полетным контроллером Pixhawk для высокоточного определения координат мобильных роботов.

## 1.2.Задачи:

1. Разработать программу для Pixhawk, позволяющую получать координаты мобильного робота, и обеспечить ее интеграцию в прошивку.
2. Подготовить предложение по интеграции модулей RTK с платформой Pixhawk.

## 1.3.Результаты:

### 1.3.1.Освоение технологии создания прошивок и программ для полетного контроллера Pixhawk.

#### 1.3.1.1. Pixhawk

Контроллер Pixhawk [1]— это решение, объединяющее полетный контроллер PX4 и плату ввода-вывода PX IO в одном. Он является высокопроизводительным автопилотом в одном модуле, подходит для установки на любой носитель (самолет, крыло, квадрокоптер, вертолет, автомобили, лодки) или любую другу роботиризированную платформу, которая может двигаться. Работа Pixhawk требует наличия прошивки, в которую в качестве модуля может быть включена пользовательская программа. Для включения модуля требуется пересобрать прошивку.

#### 1.3.1.2. Предварительные настройки. Установка Toolchain

Для сборки прошивки для контроллера Pixhawk была установлена ОС UbuntuLinux LTS (16.04), которая обеспечивает наиболее полные возможности при работе с данным устройством. Согласно инструкциям с официального сайта Pixhawk были установлены Toolchain, компилятор для сборки прошивки, а также загружен проект с GitHub, содержащий файлы, необходимые для сборки ядра. Также были установлены средства, позволяющие управлять из консоли на PC через USB-кабель выполнением программ из загруженной прошивки на Pixhawk.

#### 1.3.1.3. Сборка прошивки. Тестовый проект

Для проверки правильности работы описанной инфраструктуры был собрана и загружена в контроллер тестовая прошивка "Hello, sky!"[2]. Рассмотрим процесс сборки подробнее.

1. Создание папки нового модуля (например, Firmware/src/examples/px4\_simple\_app). Для того чтобы модуль был добавлен в прошивку при сборке, нужно добавить строку с его названием в Firmware/cmake/configs/nuttx\_px4fmu-v2\_default.cmake.

2. Создание файла исходного кода px4\_simple\_app.c, содержащего инструкции для запуска модуля и вывода информационного сообщения.

3. Создание файла CMakeLists.txt с инструкциями для сборки модуля.

4. Сборка прошивки выполняется из папки Firmware командой make px4fmu-v2\_default. После чего залить прошивку в Pixhawk можно, используя USB-кабель, командой make px4fmu-v2\_default upload.

5. Проверка работы может быть выполнена при подключении к системной консоли.

Необходимо отметить, что архитектура Pixhawk организована таким образом, что в ней уже существуют модули для работы с различными внешними устройствами, которые при необходимости могут быть включены в прошивку. Однако для правильного взаимодействия с ними нужно освоить работу с сообщениями uORB.

#### 1.3.1.4. Работа с сообщениями uORB

ThemicroObjectRequestBroker (uORB) - приложение, необходимое для обмена структурами данных между потоками и модулями с использованием простой реализации шаблона публикации-подписки. На рис. 1 можно видеть, что uORB является связующим звеном между блоками разного уровня, в том числе обеспечивает обмен данными между пользовательскими программами и различными видами датчиков.

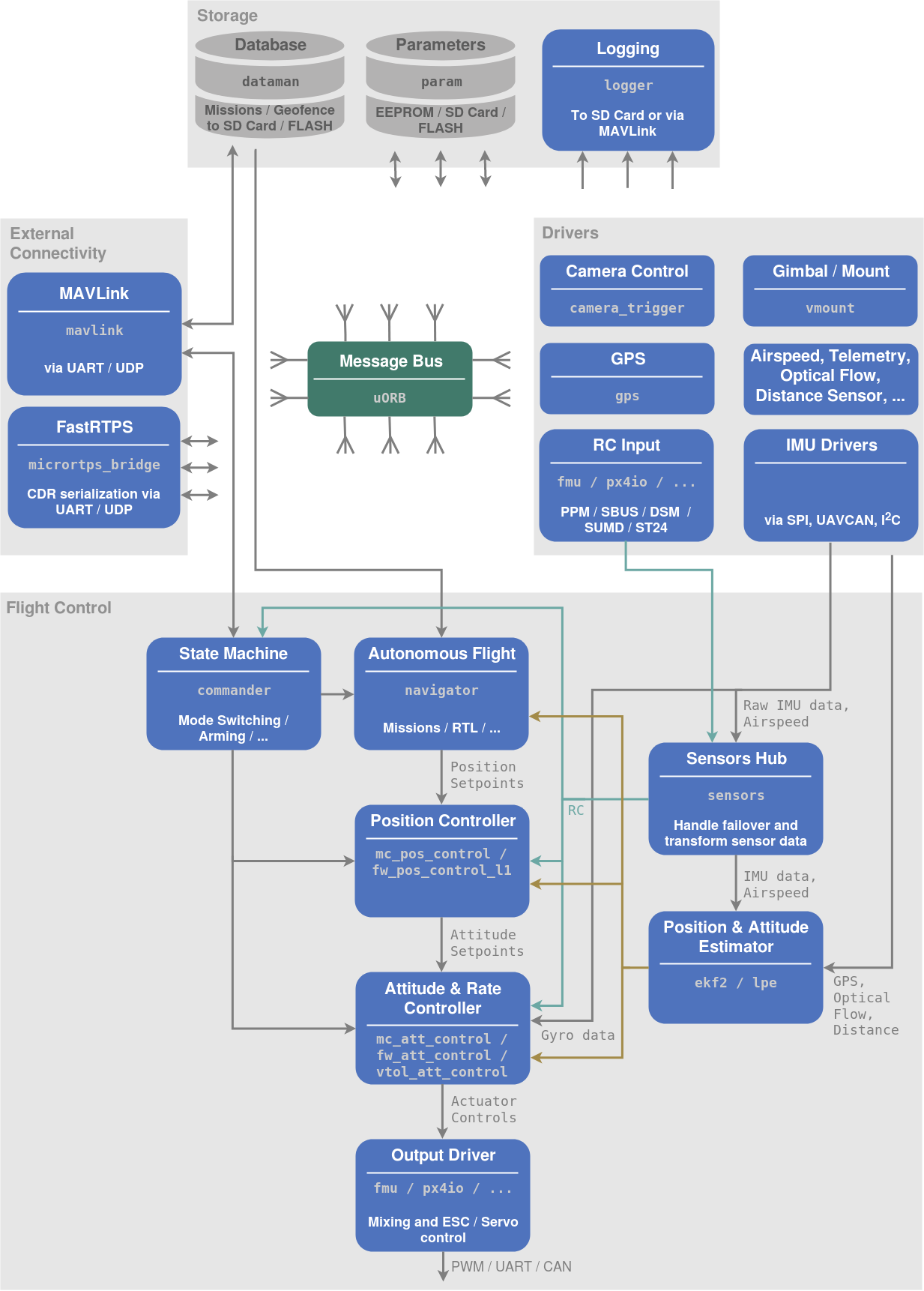


Рис. 1. Структурная схема организации PX4

Таким образом, следующий этап работы представлял собой настройку обмена данными между датчиками и контроллером, в частности были получены данные с одного из внутренних датчиков (акселерометра) и приемника GPS. Для этого была организована подписка на соответствующие сообщения uORB(sensor\_combined\_s, vehicle\_gps\_position\_s ).

|  |
| --- |
| *Пример кода для получения данных GPS* |
| /\*subscribe\*/  Intsensor\_sub\_fd\_gps=orb\_subscribe(ORB\_ID(vehicle\_gps\_position));  /\*limittheupdaterateto5Hz\*/  orb\_set\_interval(sensor\_sub\_fd\_gps,200);  pollfdfds[1];  fds[0].events=POLLIN;  fds[0].fd=sensor\_sub\_fd\_gps;  interror\_counter=0;  for(inti=0;i<5;i++){  /\*waitforsensorupdateof1filedescriptorfor1000ms(1second)\*/  intpoll\_ret=px4\_poll(fds,1,1000);  /\*handlethepollresult\*/  if(poll\_ret==0){  PX4\_ERR("Gotnodatawithinasecond");  }elseif(poll\_ret<0){  if(error\_counter<10||error\_counter%50==0){  PX4\_ERR("ERRORreturnvaluefrompoll():%d",poll\_ret);  }  error\_counter++;  }else{  if(fds[0].revents&POLLIN){  PX4\_INFO("Work");  /\*obtaineddataforthefirstfiledescriptor\*/  structvehicle\_gps\_position\_sraw;  /\*copysensorsrawdataintolocalbuffer\*/  orb\_copy(ORB\_ID(vehicle\_gps\_position),sensor\_sub\_fd\_gps,&raw);  PX4\_INFO("GPS:\t%8.4f\t%8.4f\t%8.4f\t%8.4f",  (double)raw.lat,  (double)raw.lon,  (double)raw.eph,  (double)raw.epv);  }  }  }  PX4\_INFO("exiting"); |

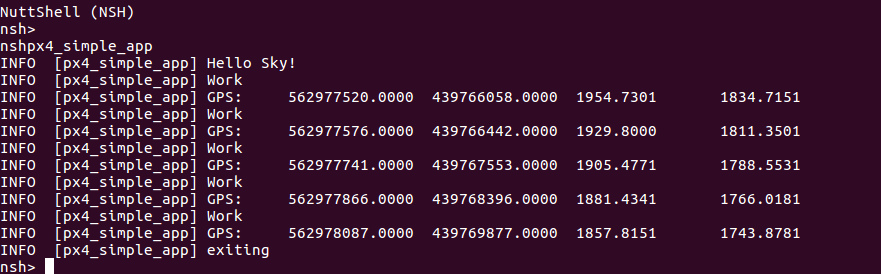
Сборка прошивки и запуск модуля производятся аналогично описанному выше. После чего в консоли можно увидеть показания датчиков.

Рис. 2. Пример работы разработанного приложения: получение данных с датчиков GPS

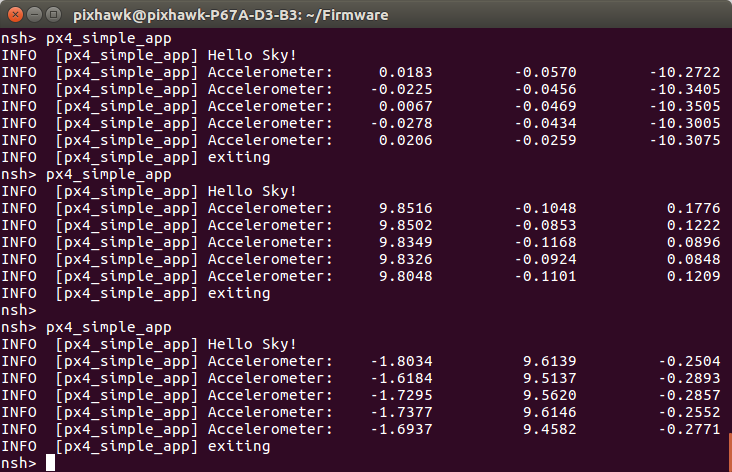


Рис. 3. Пример работы разработанного приложения: получение данных с акселерометра

### 1.3.2.Подготовка предложений по интеграции модулей RTK с платформой Pixhawk.

Платформа PX4 совместима с чипом u-blox M8P GPS[3] и поддерживает работу с устройствами, основанными на данном чипе:

* CUAV C-RTK GPS
* Drotek XL RTK GPS
* Here+ RTK GPS

Первые два из перечисленных модулей работают только с Pixhawk 3. CUAV C-RTK GPS лишь недавно стал поддерживаться, поэтому на данный момент не сформирована документация, необходимая для работы с ним. Наиболее распространенным модулем из перечисленных является Here+ RTK [4], совместимый к тому же с разными версиями Pixhawk.

Для загрузки поправок в RTK модуль необходимо опубликовать получаемые с базовой станции поправки. Механизм публикации предполагает формирование сообщения соответствующего типа (gps\_inject\_data\_s) и непосредственно публикацию в uORB. (см. пример кода). Было разработано приложение, позволяющее реализовать заполнение сообщения для передачи RTCM-поправок и опубликовать его.

|  |
| --- |
| *Пример кода для публикации поправок* |
| /\*subscribe\*/  intsensor\_sub\_fd\_gps=orb\_subscribe(ORB\_ID(vehicle\_gps\_position));  /\*limittheupdaterateto5Hz\*/  orb\_set\_interval(sensor\_sub\_fd\_gps,200);  structgps\_inject\_data\_satt;  memset(&att,0,sizeof(att));  att.timestamp=1;  att.data[0]='a';  att.data[1]='b';  att.data[2]='c';  att.data[3]='d';  orb\_advert\_tatt\_pub=orb\_advertise(ORB\_ID(gps\_inject\_data),&att);  pollfdfds[1];  fds[0].events=POLLIN;  fds[0].fd=sensor\_sub\_fd\_gps;  interror\_counter=0;  for(inti=0;i<5;i++){  intpoll\_ret=px4\_poll(fds,1,1000);  if(poll\_ret==0){  PX4\_ERR("Gotnodatawithinasecond");  }elseif(poll\_ret<0){  if(error\_counter<10||error\_counter%50==0){  PX4\_ERR("ERRORreturnvaluefrompoll():%d",poll\_ret);  }  error\_counter++;}  else{  if(fds[0].revents&POLLIN){  PX4\_INFO("Work");  /\*get data if needed\*/  /\*...\*/  /\*publish\*/  orb\_publish(ORB\_ID(gps\_inject\_data),att\_pub,&att);}  }  } |

Получить данные о точных координатах после применения модулем поправок и вычисления точных координат можно способом, аналогичным описанному выше: подписавшись на сообщения uORB о GPS координатах.

Также необходимо обеспечить возможность передачи данных от базовой станции к полетному контроллеру. Это можно сделать с использованием комплементарных Wi-Fi модулей (например, APM / PixhawkWirelessWifiRadioModule или Waveshare WI232-B). Для подключения Wi-Fi модулей нужно соединить разъем телеметрии полетного контроллера с интерфейсами модулей. В частности, для APM / PixhawkWirelessWifiRadioModule можно использовать Serialport, а для Waveshare WI232-B подключиться к имеющимся на плате разъемам UART. После этого может быть произведена настройка требуемых характеристик передачи данных (скорости передачи, количества информационных бит и т. д.) через приложение APM Planner [5]. Параметры должны соответствовать параметрам, которые заданы при конфигурировании COM-порта. Таким образом, для передачи данных необходимо выполнить на стороне базовой станции (PC) запись сообщений с поправками в последовательный порт, эти сообщения будут переданы по Wi-Fi на Pixhawk, где их можно будет опубликовать описанным выше способом.

|  |
| --- |
| *Пример записи в COM-порт* |
| charbuf[128];  char\*pbuf=buf;  int counter=0;    HANDLE hCom, hEvent;  DWORD bytesRead, bWritten;    BOOL fSuccess;  HANDLE hThread;  DWORD dwParam, dwThreadId;    VOID WINAPI ThreadProc(PVOID\* dummy){      printf("Ждите данные из порта СОМ4...**\n**");      while(true){          ReadFile(hCom, pbuf, 1, &bytesRead, NULL);          if(bytesRead>0)          {              WriteFile(hCom, pbuf++, 1, &bWritten, NULL);              counter++;              if(counter==10){                  SetEvent(hEvent);                  buf[counter]='**\0**';                  counter=0;                  pbuf=buf;              }          }      }  }  int main(void){      char\*pcComPort="COM4";      DCB dcb;      hCom=CreateFile(pcComPort, GENERIC\_READ | GENERIC\_WRITE,          0, NULL, OPEN\_EXISTING, 0, NULL);      if(hCom==INVALID\_HANDLE\_VALUE){          printf("Ошибка открытия COM4!**\n**");          while(!kbhit());          return1;      }      GetCommState(hCom, &dcb);      dcb.BaudRate=CBR\_9600;      dcb.ByteSize=8;      dcb.Parity=NOPARITY;      dcb.StopBits=ONESTOPBIT;      fSuccess=SetCommState(hCom, &dcb);      if(!fSuccess){          printf("Попытка вызвать SetCommState провалилась!**\n**");          while(!kbhit());          return1;      }      printf("COM порт %s успешно сконфигурирован**\n**", pcComPort);    hEvent=CreateEvent(NULL, FALSE, FALSE, NULL);      hThread=CreateThread(NULL, 0, (LPTHREAD\_START\_ROUTINE)ThreadProc,          &dwParam, 0, &dwThreadId);    printf("Нажмитеклавишудлявыхода...**\n**");      do{          WaitForSingleObject(hEvent, INFINITE);          printf("%s**\n**", buf);          fwrite(buf, sizeof(char), 10, fout);      }while(!kbhit());  CloseHandle(hCom);      return0;  } |

|  |  |
| --- | --- |
|  |  |
| а) | б) |
| Рис. 4. Wi-Fi модули, которые могут быть использованы для обмена данными между базовой станцией и полетным контроллером а) APM / PixhawkWirelessWifiRadioModule, б) Waveshare WI232-B | |

Другим важным аспектом разработки системы навигации является получение поправок, т.е. организация базовой станции. Для этого существуют две возможности:

* использование платных поправок, получаемых по сети Интернет (например, сервис RTKNet.ru [6]),
* покупка и настройка собственной базовой станции.

## 1.3.Заключение

Таким образом, в ходе выполнения работы было сделано:

1. Освоена технология создания прошивок и программ для полетного контроллера Pixhawk.
2. Разработана программа для Pixhawk, позволяющая получать координаты мобильных роботов и показания датчиков полетного контроллера, обеспечена ее интеграция в прошивку.
3. Подготовлено предложение по интегрированию модулей RTK с платформой Pixhawk. Реализован механизм публикации для передачи поправок в RTK модуль.

# 2. Система управления электроприводами RINF

## 2.1. Цель:

Создание системы управления электроприводами электротранспортной тележки RINF.

**2.2. Задачи:**

1. На базе библиотеки управления приводами экзоскелета (разработчик Севрюков О.) создать библиотеку управления мотор-колесами электротранспортной тележки RINF;
2. Разработать тестовое приложение для отработки и демонстрации режимов управления мотор-колесами.

## 2.3.Этапы разработки:

### 2.3.1.Первый этап

В ходе работы была разработана библиотека для управления электроприводами мотор-колес роботизированной транспортной тележки. Разработка велась на основе имеющейся библиотеки управления приводами экзоскелета, ориентированной на работу с контроллерами BLSD-20 НПФ «Электропривод». Для применения библиотеки в составе ПОэлектротранспортной тележки RINF необходима ее коррекция и переход на контроллеры BLSD-50 НПФ «Электропривод».

Режимы работы приводов, которые должны обеспечиваться библиотекой:

1) Вращение по часовой стрелки с заданным значением кода, пропорционального скважности ШИМ-импульса (прямой ход);

2) Вращение против часовой стрелки с заданным значением кода, пропорционального скважности ШИМ-импульса (прямой ход);

3) Блокировка вращения мотор-колеса в обоих направлениях (по часовой и против часовой стрелки) с заданным кода, пропорционального тормозящему моменту.

Для решения поставленных задач функции управления контроллерами мотор-колес роботизированной тележки RINF, были скорректированы согласно вышеуказанным требованиям.

Управление скважностью ШИМ-импульса приводов (пункты 1 и 2) осуществляется с помощью метода GroupOfEngines::SetADC(), который позволяет задать желаемые пользователем скважности ШИМ-импульса. Для установки заданного значения скважности ШИМ-импульса для всех приводов группы, включенных в нее с помощью метода GroupOfEngines::AddEngineToGroup, был использован метод GroupOfEngines::SendMomentsForAllEngines().

После вызова данного метода желаемые скважности ШИМ-импульса устанавливаются для всех приводов и происходит вращение мотор-колес в заданных направлениях.

Для блокировки мотор-колеса требуется выполнить перечень последовательных действий:

1) Вызвать метод GroupOfEngines::LockEngine();

2) Далее необходимо одной командой установить заданные моменты у всех приводов группы (GroupOfEngines::SendMomentsForAllEngines());

3) Задать усилие, с которой осуществляется блокировка. ЭтоможносделатьвызвавGroupOfEngines::SetADC() илиGroupOfEngines::SetMoment();

4) Необходимо одной командой установить заданные моменты у всех приводов группы (GroupOfEngines::SendMomentsForAllEngines()).

### 2.3.2.Этап второй

Создание тестового приложения для отработки и демонстрации работы мотор-колеса.

Для тестирования работы мотор-колеса было создано тестовое приложение позволяющее управлять режимами вращения посредством задания пользователем значений кода, пропорционального скважности ШИМ-импульса. Данное приложение позволяет безопасно тестировать созданную библиотеку, учитывая недопустимые значения и защищая контроллер от перегорания.

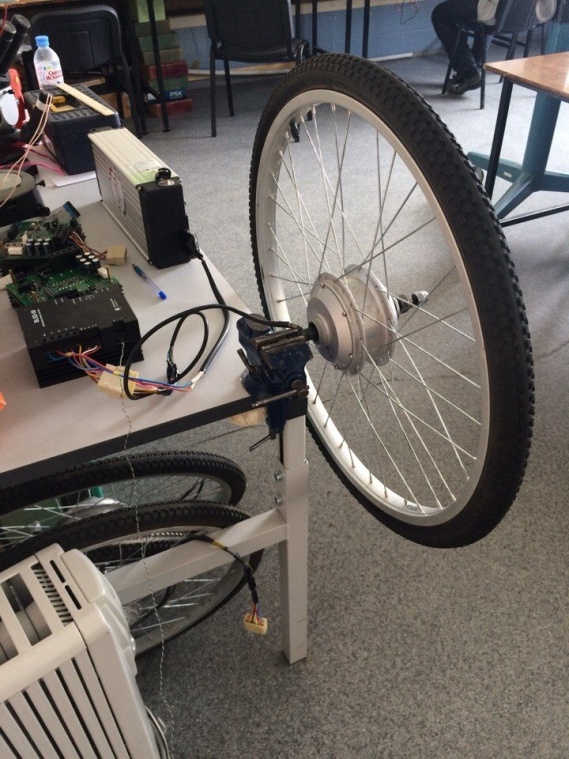


Рис. 5. Установка для тестирования работы библиотеки управления контроллером BLSD-50

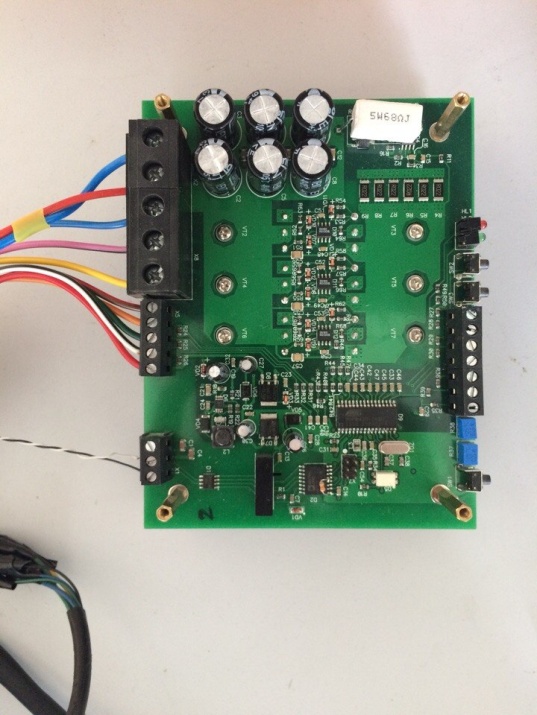


Рис. 6. Контроллер BLSD-50 для управления мотор-колесами

## 2.3.Заключение:

Выполнена корректировка библиотеки, позволяющая управлять мотор-колесами. Создано приложение для отработки и демонстрации работы режимов управления мотор-колесами.

Система готова к переносу на полетный контроллер Pixhawk с целью создания встроенной системы управления электротранспортной тележки RINF.

В ходе работы была было выявлено ограничение в встроенном программном обеспечении контроллера BLSD-50, не позволяющее достигать требуемой максимальной скорости. В ходе дальнейшей работы требуется произвести корректировку данного программного обеспечения.

**Приложение 1.**

Заголовочный файл скорректированной библиотеки

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Организация: кафедра ИТФИ ННГУ, г. Нижний Новгород

//Проект: EXO-NN

//Версия проекта: 1.0

//Спецификация: ANSI C++ 98

//Создан: 22.09.2014

//Изменен: 20.03.2015

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\* \authorСеврюков О.Ф., Леонов С.В., Минеев С.А.\*/

/\*\* \file group\_of\_engines\_API.h\*/

/\*\* \brief Описание классов для работы с группой бесколлекторных электродвигателей.\*/

#ifndef \_GROUP\_OF\_ENGINES\_API\_H\_

#define \_GROUP\_OF\_ENGINES\_API\_H\_

#include <string>

#include <vector>

#include <stdexcept>

#include <cmath>

//#include "blsd.h"

#include "send\_functions.h"

using namespace std;

#define MAX\_SIZE\_OF\_ENGINES\_GROUP 256 //<! Максимальный размер группы приводов.

#define ACCELERATION\_VALUE 24//<! Максимальное значение ускорения вала (только для штатной прошивки BLSD-xx).

#define DECELERATION\_VALUE 24//<! Максимальное значение ускорения торможения вала (только для штатной прошивки BLSD-xx).

namespaceBlsd

{

/\*\*Идентификатор привода.\*/

enumEngineID

{

LeftThighEngineID=0, //<! Идентификатор привода левого бедра.

LeftKneeEngineID=1, //<! Идентификатор привода левого колена.

RightThighEngineID=3, //<! Идентификатор привода правого бедра.

RightKneeEngineID=4 //<! Идентификатор привода правого колена.

};

/\*\*Калибровочная таблица, связывающая момент,

\*развиваемый приводом, и код АЦП токового датчика контроллера привода BLSD-xx.\*/

classCalibrateTable

{

public:

/\*\*КодАЦП.\*/

vector<unsigned int> \_ADCCode;

/\*\*Моменты, нм.\*/

vector<double>moment;

/\*\*Заполнить калибровочную таблицу.

\*@paramMoment момент, развиваемый приводом, нм.

\*@paramADCCode код АЦП токового датчика.\*/

voidFillTable(unsigned intADCCode, double Moment);

/\*\*Выгрузить калибровочную таблицу в файл.

\*@paramfilepath абсолютный путь к файлу с калибр.данными.

\*@exceptionruntime\_error не удалось выгрузить данные в файл.\*/

voidFlashToFile(string filepath);

/\*\*Конструктор.\*/

CalibrateTable();

};

/\*\*Привод.\*/

class Engine

{

/\*\*Адресприводанашине RS485.\*/

unsignedchar \_RS485\_addr;

/\*\*Наименование привода в группе.\*/

string \_name;

/\*\*Калибровочный коэффициент для расчета угловой скорости.\*/

double \_coef\_aspeed;

/\*\*Получить калибровочную таблицу 'момент - код АЦП токового датчика.'.

\*@paramfileWithCalibrateData абсолютный путь к файлу с калибровочными данными.

\*@return калибровочная таблица.

\*@exceptionruntime\_error не удалось открыть файл с калибровочной таблицей.\*/

CalibrateTableGetTable(stringfileWithCalibrateData);

/\*\*Прочитать калибровочный коэффициент длярасчет угловой скорости.

\*@paramfileWithAngleSpeedCoefs абсолютный путь к файлу с калибровочными данными.\*/

voidGetAngleSpeedCoefFromFile(string fileWithAngleSpeedCoefs);

public:

/\*\*Момент, развиваемый приводом, Нм.\*/

doublemoment;

/\*\*Направление вращения привода.\*/

booldirection;

/\*\*КодАЦП.\*/

intadc\_code;

/\*Калибровочная таблица 'момент - код АЦП токового датчика'.\*/

CalibrateTable \_cal\_table;

/\*\*Идентификатор привода в группе (индекс в массиве приводов).\*/

unsignedint \_engineID;

/\*\*Конструктор.

\*@paramaddr адрес привода на шине RS485;

\*@paramname наименование привода в группе;

\*@paramengineID идентификатор привода в группе (индекс в массиве приводов);

\*@paramfileWithCalibrateData абсолютный путь к файлу с калибровочными данными;

\*@paramfileWithAngleSpeedCoefs абсолютный путь к файлу с калибровочными коэффициентом для расчета угловой скорости.

\*@exceptionruntime\_error не удалось открыть файл с калибровочной таблицей.\*/

Engine(unsigned char addr, string name, unsigned intengineID, string fileWithCalibrateData, string fileWithAngleSpeedCoefs);

/\*\*Получить наименование привода в группе.\*/

stringGetName();

/\*\*Получить адрес привода на шине RS485.

\*@return адрес привода на шине RS485.\*/

unsignedintGetAddr();

/\*\*Получить калибровочный коэффициент для расчета угловой скорости.

\*@return калибровочный коэффициент.\*/

doubleGetAngleSpeedCoef();

};

/\*\*Группаприводов.\*/

classGroupOfEngines

{

private:

/\*\*IP адрес преобразователя интерфейсов MOXA NPORT 5232.\*/

string \_ip;

/\*\*Преобразователь интерфейсов Ethernet-RS485,

\*используемый для взаимодействия с контроллерами BLSD-xx.\*/

//CNPort5232 \_nPort;

/\*\*Массивприводов.\*/

vector<Engine> \_engines;

/\*\*Массив угловых скоростей приводов в град/с.\*/

vector<double> \_angleSpeeds;

/\*\*Массив значений счетчика оборотов привода.\*/

vector<int> \_masCounter;

/\*\*Порт преобразователя интерфейсов MOXA NPORT 5232.\*/

int \_port;

/\*\*Контроллер соединения с контроллером привода.\*/

Controller \_control;

/\*\*Протокол управления контроллером привода.\*/

Protokolprot;

/\*\*Войти в режим регулировки от ПК (только для штатной прошивки BLSD-xx).

\*@paramengineID идентификатор привода в группе (индекс в массиве приводов).\*/

voidPCReg(unsigned intengineID);

/\*\*Задать торможение.

\*@paramengineID идентификатор привода в группе (индекс в массиве приводов).\*/

voidSetDec(unsigned intengineID);

public:

/\*\*Добавить привод в группу.

\*@paramengine добавляемый привод.

\*@exceptionsystem\_error не удалось соединиться с контроллером BLSD-xx.\*/

voidAddEngineToGroup(Engine&engine);

/\*\*Проверить наличие привода в группе.

\*@paramengineID идентификатор привода в группе (индекс в массиве приводов).

\*@returntrue привод находится в группе.\*/

boolIsContain(unsigned intengineID);

/\*\*Имеется ответ от контроллеров на групповую команду задания кода АЦП.\*/

boolisBLSDResponce;

/\*\*Получить угловую скорость привода в град/сек.

\*@paramengineID идентификатор привода в группе (индекс в массиве приводов);

\*@return угловая скорость;

\*@exceptionruntime\_error невозможно получить значение (выставить флаг isBLSDResponce).\*/

doubleGetAngleSpeed(unsigned intengineID);

/\*\*Получить число оборотов привода для калибровки.

\*@paramengineID идентификатор привода в группе (индекс в массиве приводов);

\*@return число оборотов привода для калибровки;

\*@exceptionruntime\_error невозможно получить значение (выставить флаг isBLSDResponce).\*/

intGetNumRevCalibration(unsigned intengineID);

/\*\*Получить группу приводов.

\*@return массив приводов.\*/

vector<Engine>GetEnginesOfGroup();

/\*\*Задать момент для конкретного привода.

\*@paramengineID идентификатор привода в группе (индекс в массиве приводов);

\*@parammoment устанавливаемый момент, нм.

\*@exceptioninvalid\_argument некорректное значение аргумента метода;

\*@exceptionsystem\_error не удалось соединиться с контроллером BLSD-xx.\*/

voidSetMoment(unsigned intengineID, double moment);

/\*\*Задать код АЦП для конкретного привода.

\* @paramengineID идентификатор привода в группе (индекс в массиве приводов);

\* @paramadc\_code устанавливаемый код АЦП.

\* @exceptioninvalid\_argument некорректное значение аргумента метода;

\* @exceptionsystem\_error не удалось соединиться с контроллером BLSD-xx.\*/

voidSetADC(unsigned intengineID, intadc\_code);

/\*\*Задать ускорение.

\* @paramengineID идентификатор привода в группе (индекс в массиве приводов);

\* @paramaccValueзначениеускорения.\*/

voidSetAcc(unsigned intengineID, unsigned intaccValue);

/\*\*Установить заданные моменты (@seeSetMoment(unsignedintengineID, doublemoment)) для каждого

\*привода группы.

\*@exceptionsystem\_error не удалось соединиться с контроллером BLSD-xx.\*/

voidSendMomentsForAllEngines();

/\*\*Получить количество оборотов привода.

\*@paramengineID идентификатор привода в группе (индекс в массиве приводов).

\*@return число оборотов привода.

\*@exceptioninvalid\_argument некорректное значение аргумента метода;

\*@exceptionsystem\_error не удалось соединиться с контроллером BLSD-xx.\*/

shortGetRounds(unsigned intengineID);

/\*\*Получить код АЦП привода.

\*@paramengineID идентификатор привода в группе (индекс в массиве приводов).

\*@return код АЦП.

\*@exceptioninvalid\_argument некорректное значение аргумента метода;

\*@exceptionsystem\_error не удалось соединиться с контроллером BLSD-xx.\*/

intGetADCCode(unsigned intengineID);

/\*\*Остановить привод.

\*@paramengineID идентификатор привода в группе (индекс в массиве приводов).

\*@return true приводостановлен.\*/

boolStopEngine(unsigned intengineID);

/\*\*Задать параметры привода.

\*@paramengineID идентификатор привода в группе (индекс в массиве приводов);

\*@parampairMagnetPolesNum количество пар полюсов магнита;

\*@parammaxRevPerSecNum максимальное количество об/сек.\*/

voidSetEngineOptions(unsigned intengineID, intpairMagnetPolesNum, intmaxRevPerSecNum);

/\*\*Сбросить счетчик оборотов привода.

\* @paramengineID идентификатор привода в группе (индекс в массиве приводов).

\* @exceptioninvalid\_argument некорректное значение аргумента метода;

\* @exceptionsystem\_error не удалось соединиться с контроллером BLSD-xx.\*/

voidResetRoundsCounter(unsigned intengineID);

/\*\*Заблокировать вал привода (уже не используется)

\*@paramengineID идентификатор привода в группе (индекс в массиве приводов);\*/

voidBlockEngine(unsigned intengineID);

/\*\*Заблокировать вал привода

\*@paramengineID идентификатор привода в группе (индекс в массиве приводов);\*/

voidLockEngine(unsigned intengineID);

/\*\*Разблокировать вал привода

\*@paramengineID идентификатор привода в группе (индекс в массиве приводов);\*/

voidUnlockEngine(unsigned intengineID);

/\*\*Конструктор.

\*@paramip IP-адрес преобразователя интерфейсов;

\*@paramport порт преобразователя интерфейсов.

\*@parampr протокол управления контроллером привода.\*/

GroupOfEngines(string ip, int port, Protokolpr);

/\*\*Конструктор.

\*@paramcom\_portcom-порт, к которому подключен интерфейс RS232-RS485.

\*@parampr протокол управления контроллером привода.\*/

GroupOfEngines(string com\_port, Protokolpr);

};

}

#endif // \_GROUP\_OF\_ENGINES\_API\_H\_

**Приложение 2.**

Пример использования библиотеки.

intmain(intargc, char \*argv[])

{

//идентификатор привода, параметры тестовых режимов работы привода

unsignedintengineID;

int moment;

int mode;

cout<<"Input mode: 0-block test, 1-speed test "<<endl;

cin>>mode;

cout<<"Input engine number from -255 to 255: "<<endl;

cin>>engineID;

if ((engineID>256))

throwstd::logic\_error("Input value from -255 to 255");

switch(mode)

{

case 0:

{

cout<<"Input power from 0 to 40: "<<endl;

cin>>moment;

break;

}

case 1:

{

cout<<"Input speed from -255 to 255: "<<endl;

cin>>moment;

break;

}

}

if ((moment>40) || (moment<0))

throwstd::logic\_error("Input value from -255 to 255");

stringcomPort = "/dev/ttyS0"; //"/dev/ttyS1";

GroupOfEnginesgrp(comPort, NIFTI); //создаемгруппуприводов

Engine left\_engine(engineID, "left\_engine", engineID, "", "");

grp.AddEngineToGroup(left\_engine); //добавляемприводвгруппу

//grp.isBLSDResponce = true;

if(mode == 0)

{

while(true)

{

grp.LockEngine(engineID);

grp.SendMomentsForAllEngines();//Одновременнаявыдачазаданныхмоментоввсемиприводами

usleep(10000);//Пауза не менее 10 мс

grp.SetADC(engineID, moment);//Задание усилия, с которым выполняется блокировка

grp.SendMomentsForAllEngines();//Одновременная выдача заданных моментов всеми приводами

}

}

else if (mode == 1)

{

while(true)

{

grp.SetADC(engineID, moment);

grp.SendMomentsForAllEngines();

//intangle\_speed = grp.GetAngleSpeed(engineID);

usleep(20000);

}

}

return 0;

}

# 3. Подсистема управления движением (CNTR)

## 3.1.Постановка задачи

Для управления автономной транспортной платформой RINF (далее - платформа) необходимо разработать подсистему управления движением(CNTR). Разработанный алгоритм должен удовлетворять следующим требованиям:

1. Разработать 2D-имитатор роботизированной платформы RINF в окружении;
2. Разработать макет программы управления движения платформы RINF.
3. Обеспечить интеграцию 2D-имитатора роботизированной платформы RINF и макета программы управления движения с полётным контроллером Pixhawk.

## 3.2.Описание 2D-имитатора роботизированной работы RINF

Имитатор состоит из самого прямоугольного мира, имеющего крайние угловые точки, и роботов, добавляемых в данный мир.

Сам робот имеет следующие характеристики:

1. Декартовые координаты.
2. Ориентацию, то есть угол относительно оси ОХ.
3. Карту дальности: массив дальностей от робота до ближайшего препятствия с сенсора от минимального до максимального угла с дискретным шагом
4. Размеры робота
5. Поступательную и вращательную(угловую) скорости.

Робот извне получает скорости вращениям левого и правого колес. На основе полученных скоростей робот высчитывает поступательную и вращательную скорости. Перед выполнением движения робота рассчитывается карта дальности. Расчёт происходит следующим образом: сама просматриваемая сенсором область представляет собой сектор, центр которого расположен на переднем краю робота, а центр дуги находится на оси направления робота. Сам сектор делится на множество частей, в пределах которых и проверяется наличие препятствий, то есть на данном участке сектора высчитывается расстояние от центра сектора до края попавшего на данный участок препятствия. Если препятствия нет, то просто возвращается радиус сектора. После вычисления карты дальностей робот делает сам шаг: сначала поворот, а потом само поступательное движение.

Имитатор извне принимает скорости вращения колёс, а на выходе возвращает текущие координаты робота и его карту дальности.

## 3.3.Описание имитатора полетного контроллера Pixhawk и интеграция в него 2D-имитатора

Имитатор контроллера представляет собой скрипт в виде консольного приложения, где с имитатора роботизированной работы RINF выводятся данные о положении робота и информации с датчиков, а скрипт вводит в систему скорости колёс в асинхронном режиме.

При запуске самого скрипта запускается двухпоточный процесс, где в дополнительном потоке выполняется работа имитатора: расчёт карты дальности, скоростей и координат, а также отправка информации основному потоку.

Основной поток выводит полученную информацию на экран, а также ждёт нажатия какой-либо клавиши, символизирующего о желании пользователя передать сигналы на колёса. При получении о сообщении нажатия клавиши, основной поток считывает вводимые с экрана скорости колёс и передаёт их дополнительному потоку, который продолжает работать уже с учётом введенных данных.

В целом логика работы скрипта такова: запускается моделирование с начальными параметрами. Во время работы робот движется в свободном режиме, если же пользователь хочет поменять его маршрут, то он вводит соответствующие данные с клавиатуры. Таким образом робот управляется в ручном режиме. Также имеется одна особенность: из-за ограниченной ширины консоли одномерный массив дальностей пришлось представить в виде матрицы, где элементы массива расположены построчно.

## 3.4.Моделирование алгоритма ручного движения

Ниже приведены результаты работы программы, изображающие разные этапы моделирования. Изначально присутствует следующая конфигурация: Робот располагается в центре мира, откуда потом движется по прямой.

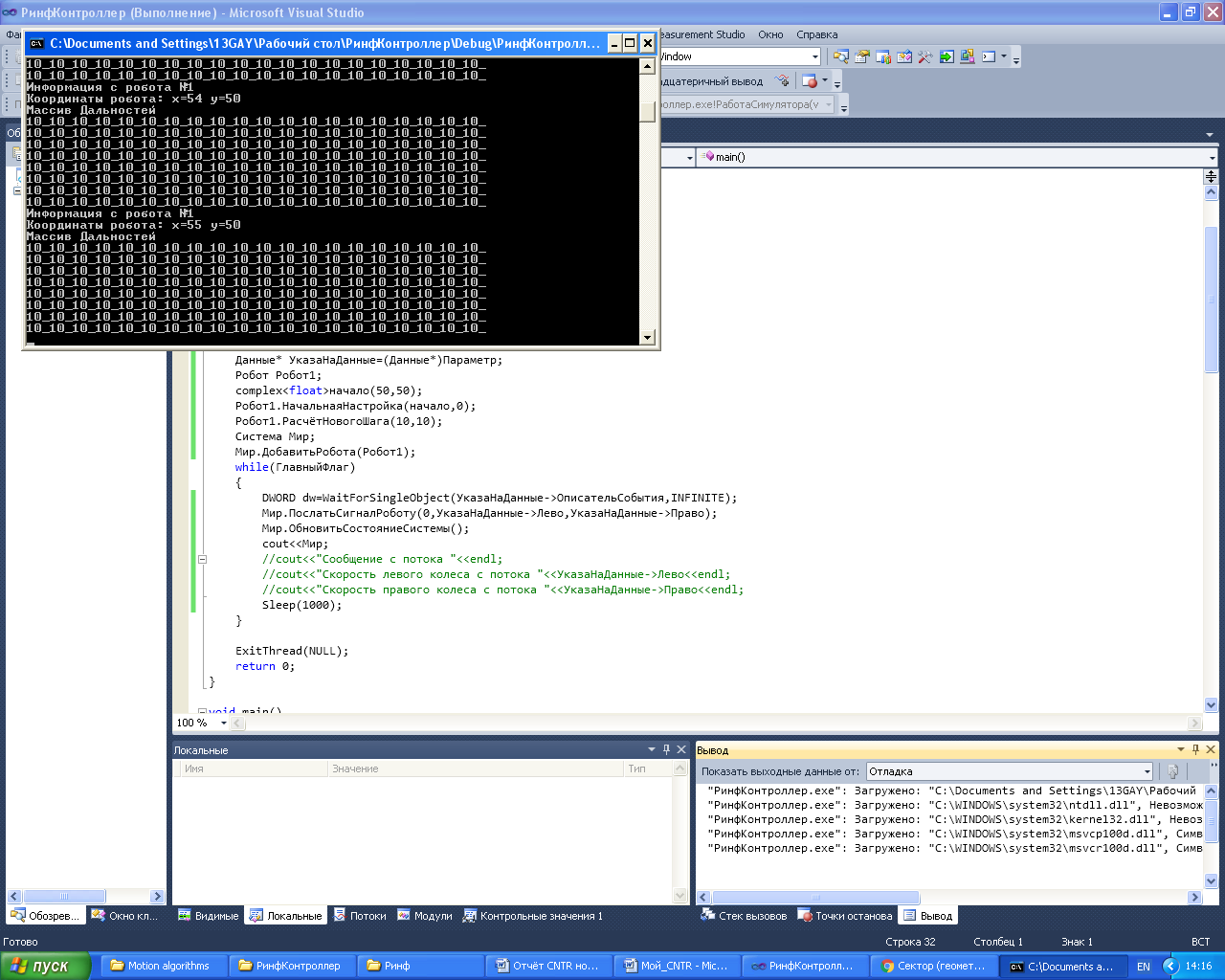


Рис. 7. Работа в штатном режиме.

Здесь изображено состояние, когда робот перемещается в свободном режиме и передаёт основному потоку свои данные.

При нажатии любой клавиши появляется возможность вводить данные с экрана.

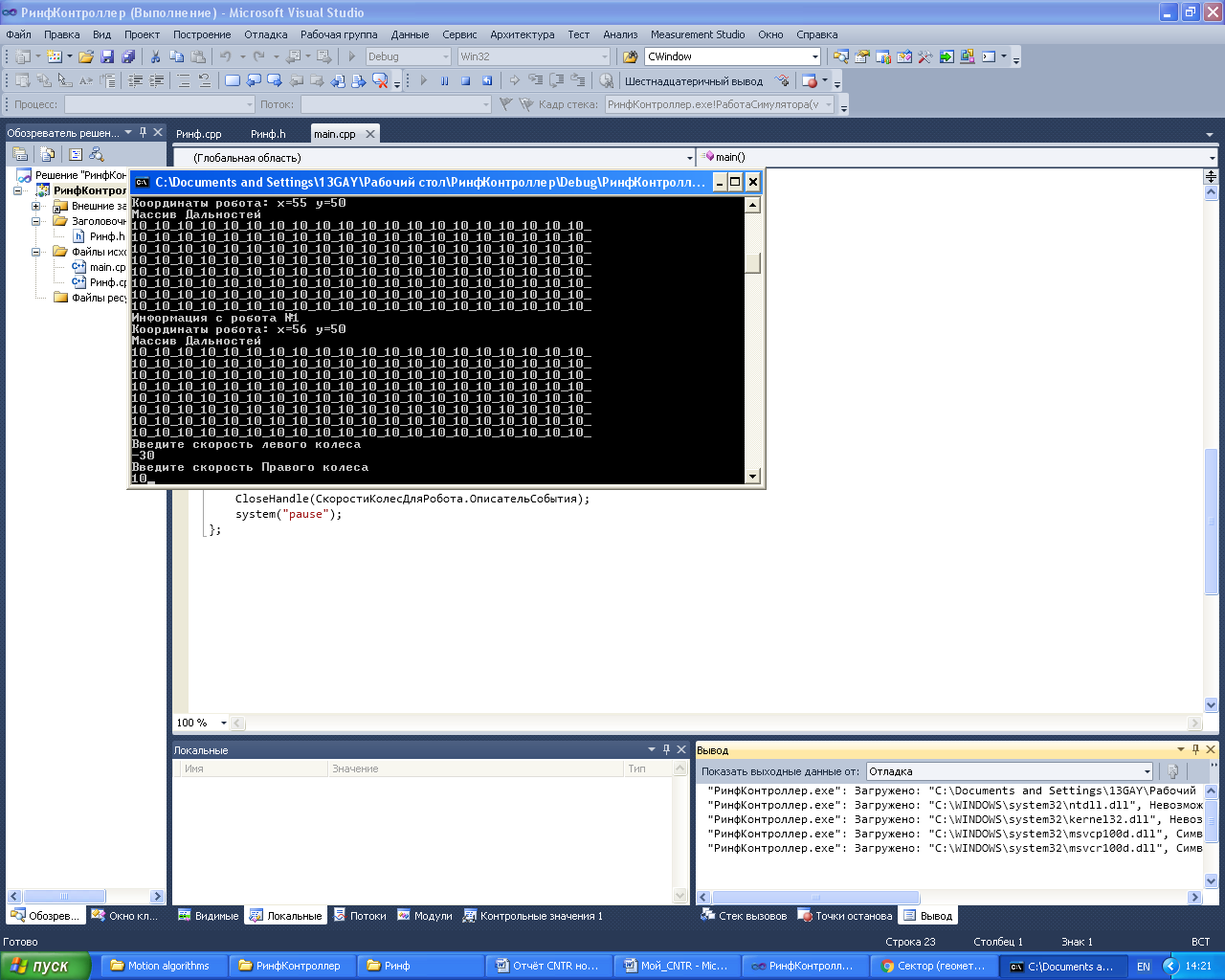


Рис.8. Ввод данных.

После чего продолжается моделирование с учётом введённых данных.

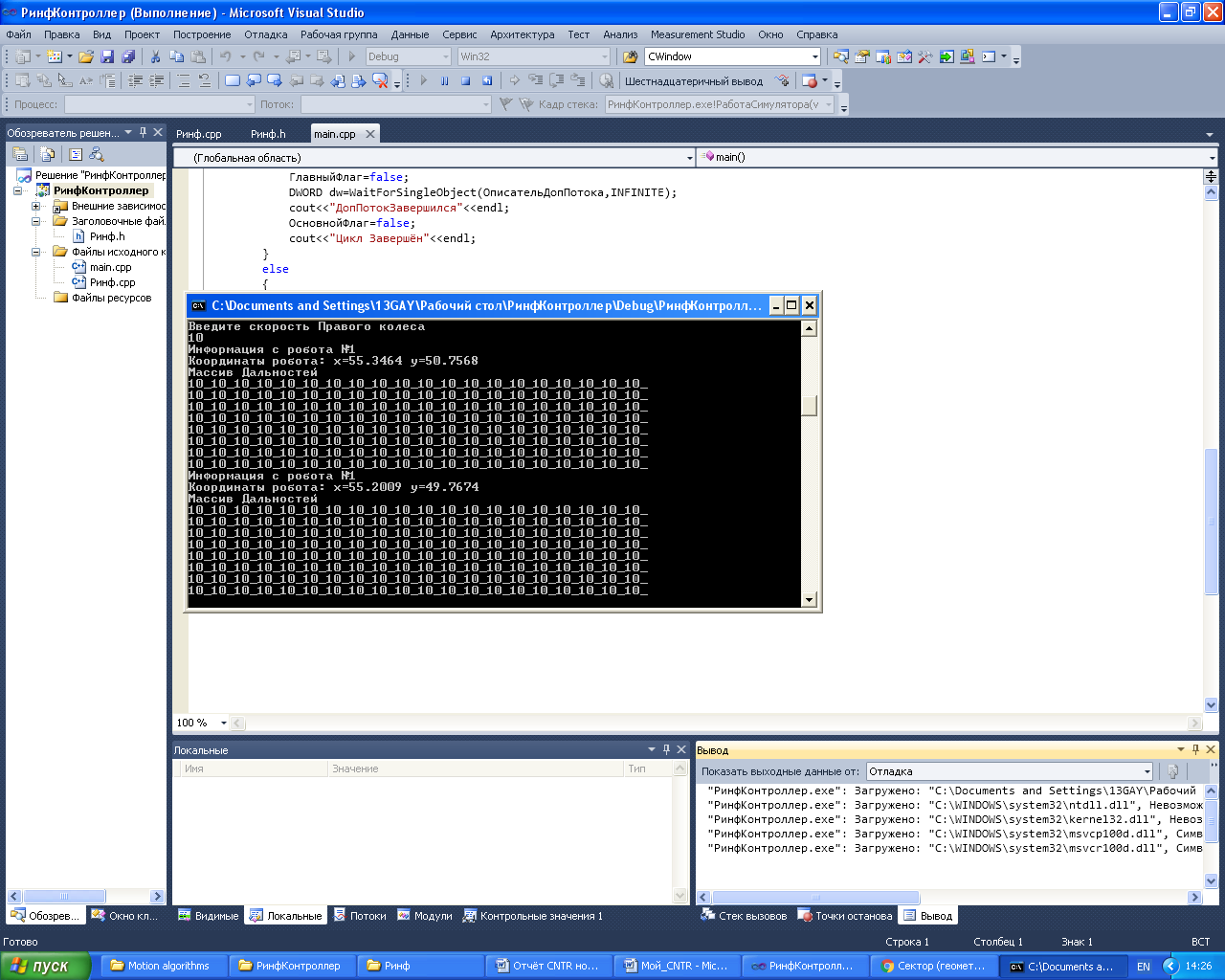


Рис. 9. Работа в штатном режиме с учётом введённых данных.

В данном случае робот станет двигаться по кругу. Если робот наткнётся на препятствие, то это будет заметно на карте дальности. Но добавить препятствия внутри мира не удалось, поэтому датчики могут зафиксировать только приближение к границам мира.

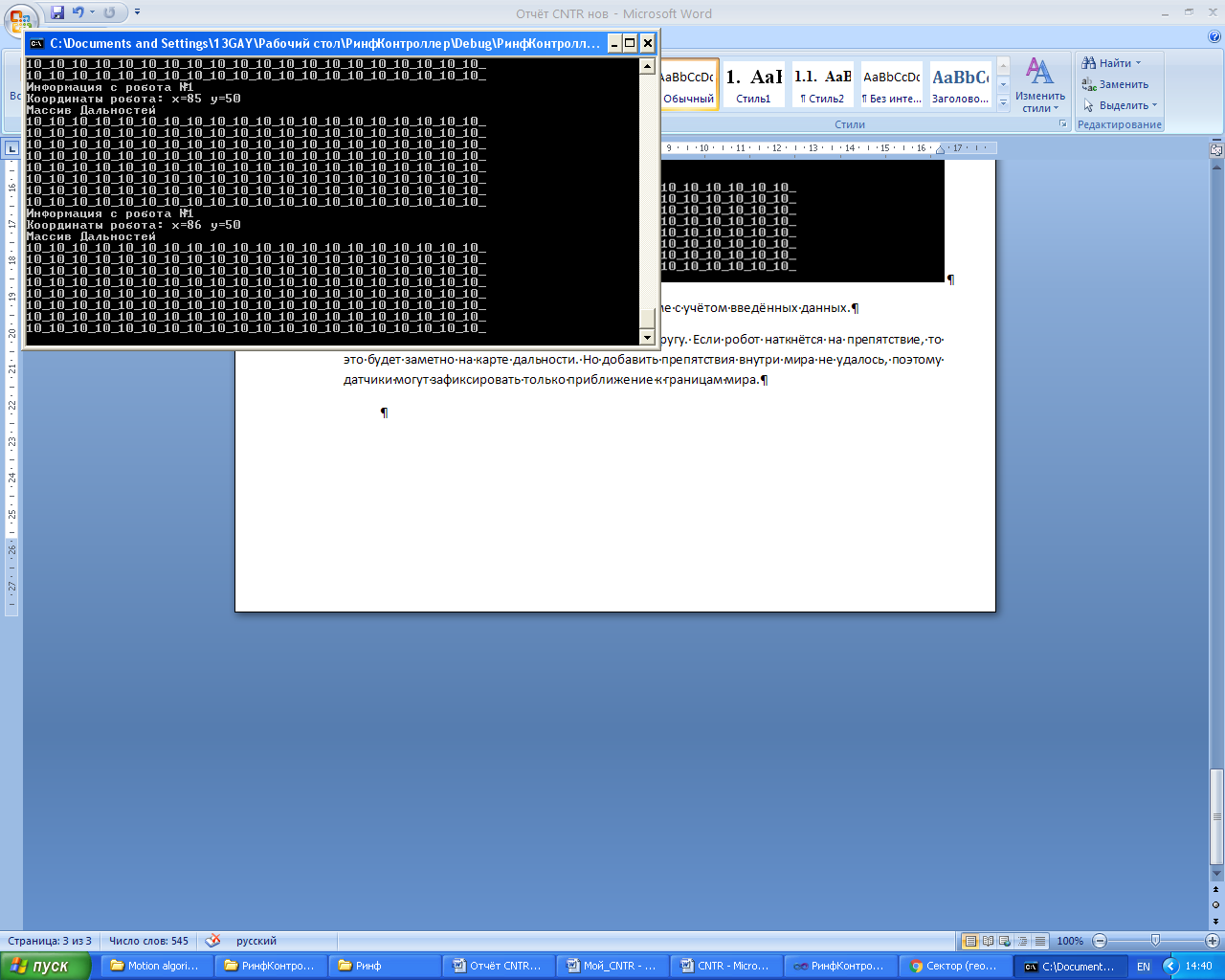


Рис. 10. Робот подходит к границе мира.

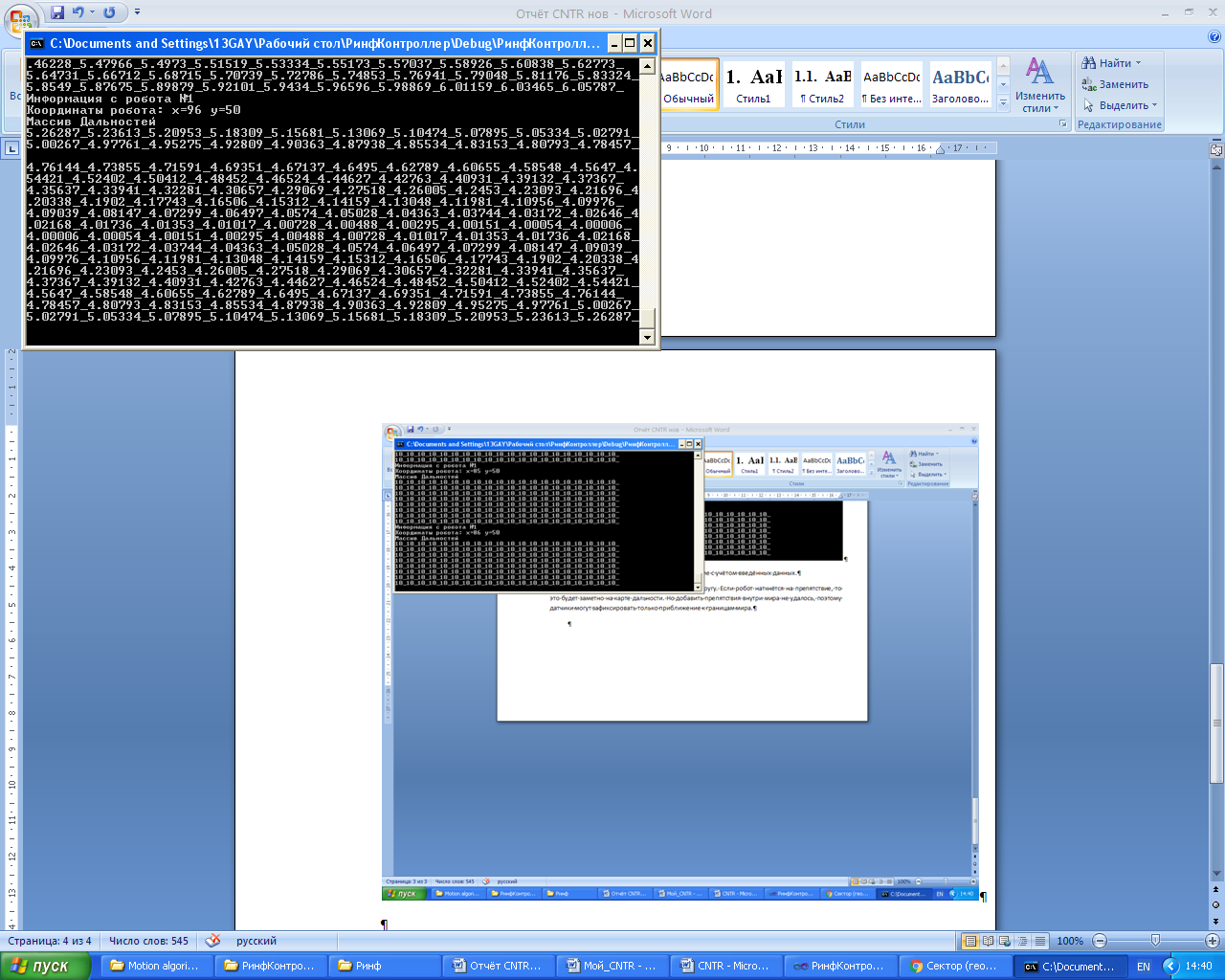


Рис. 11. Робот уже вблизи границы мира.

На рисунке 10 граница мира пока ещё не попала в область сканирования сенсора, поэтому пока возвращается радиус сектора, а на рисунке 5 граница уже попала в область, поэтому мы можем наблюдать варьирование значений дальности.

Так как робот движется по прямой линии, и середина дуги ближе к краю мира, чем её края, то соответственно минимальная дальность приходится на середину дуги, а максимумы расположены по краям.

## 3.5.Выводы

1. Разработан 2D-имитатор роботизированной системы RINF в окружении.
2. 2D-имитатор интегрирован в скрипт, имитирующий часть работы полётного контроллера Pixhawk.
3. По сравнению с прошлым разом, робот имеет конечные размеры, а не является материальной точкой, а также добавлена карта дальности.

# 4. Электро-механическая подсистема (ELMH)

## 4.1.Цель

Доработка 3D-модели электрической тележки RINF в части размещения и крепления источников питания и контроллеров управления двигателями.

## 4.2.Задачи:

а) Скорректировать 3D-модель в части размещения и крепления источников питания и контроллеров;

б)Подготовить комплект чертежей элементов крепления;

в) Подготовить перечень конструктивных элементов и материалов.

## 4.3.Результаты

В ходе выполнения работы была скорректирована 3D-модель электротранспортной тележки RINF в части:

- размещения и крепления источников питания;

- размещения и крепления контроллеров;

- изменение конструкции основания с учетом дополнительных элементов крепления.

Проект 3D-модели разрабатывался в среде КОМПАС вер. 14 по завершении работ был размещен в репозиторииBITBUCKET:

https://bitbucket.org/VPRG/rinf/ [chertezhi/](https://bitbucket.org/VPRG/rinf/commits/6e11c084e98921f6d8d3f949a40a29dfb5c2a28c#chg-chertezhi/%D0%B4%D0%B5%D1%82%D0%B0%D0%BB%D0%B8/%D0%BA%D1%80%D0%B5%D0%BF%D0%B5%D0%B6%20%D0%B4%D1%80%D0%B0%D0%B9%D0%B2%D0%B5%D1%80%D0%BE%D0%B2.cdw)

Далее приведены виды 3D-модели и примеры чертежей, генерируемых из нее.

### 4.1.1. 3D модель:

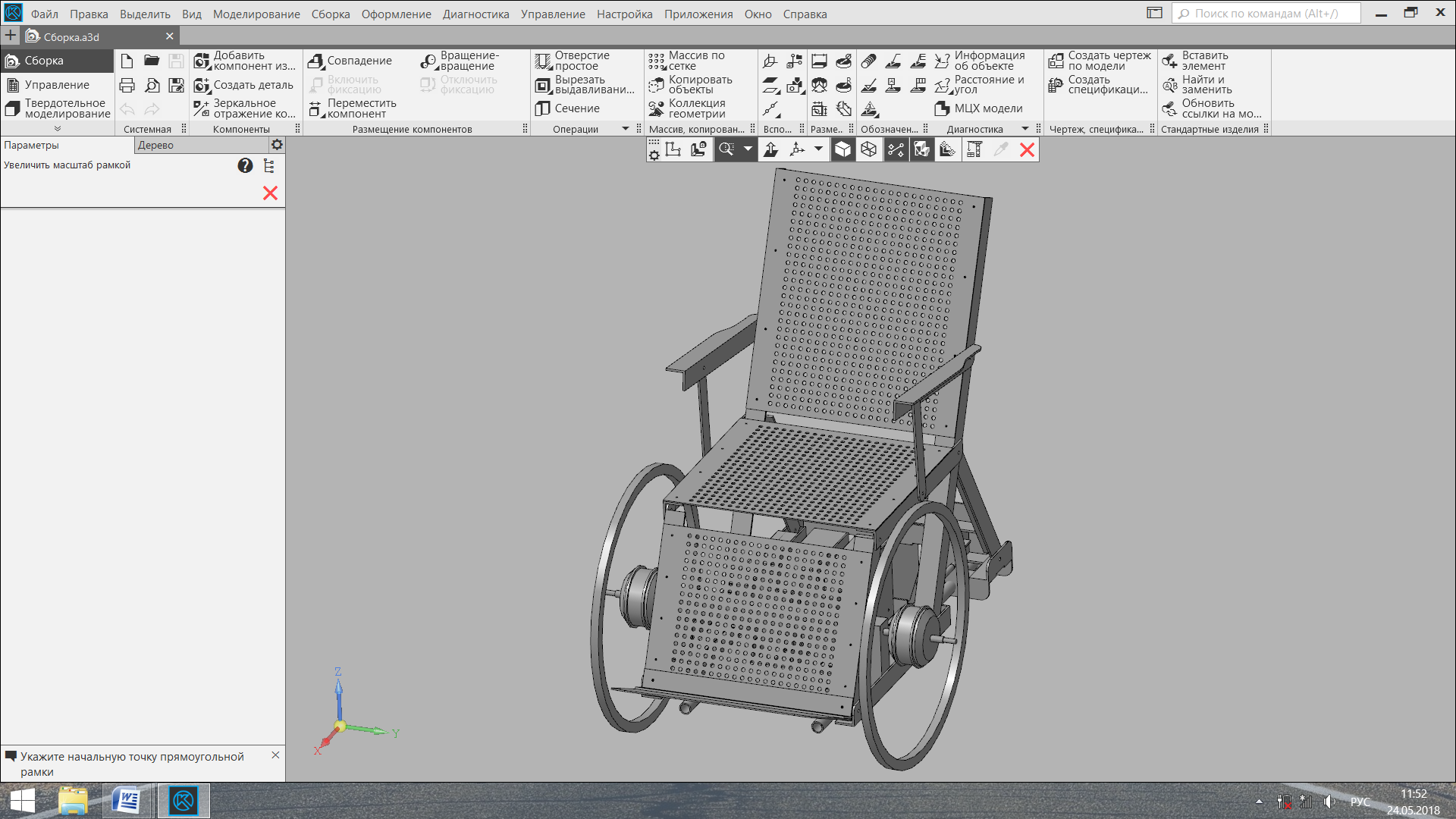


Рис. 12.Вид 3D-модели спереди

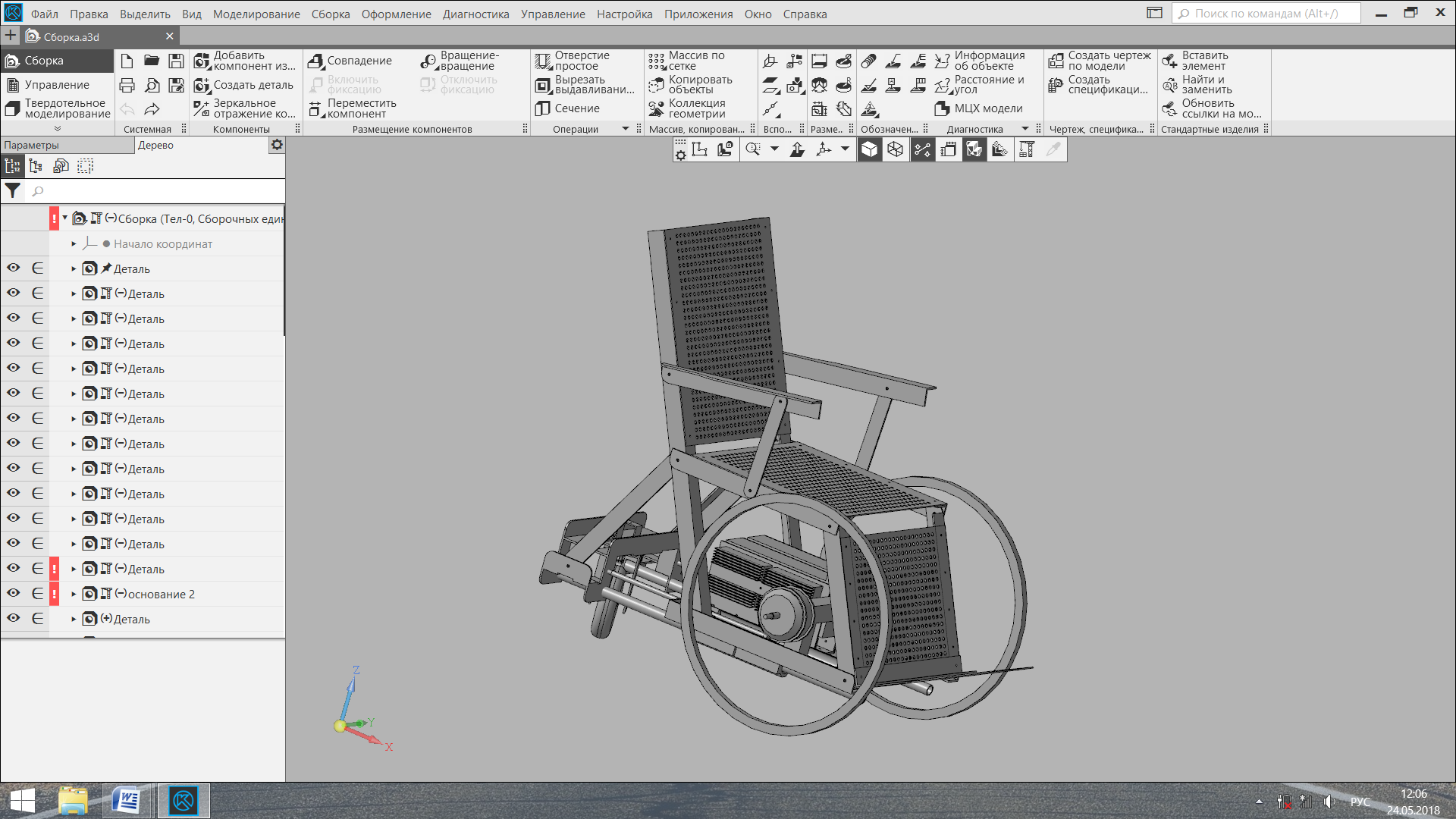


Рис. 13.Вид 3D-модели справа с установленными компонентами (источники питания и контроллеры)

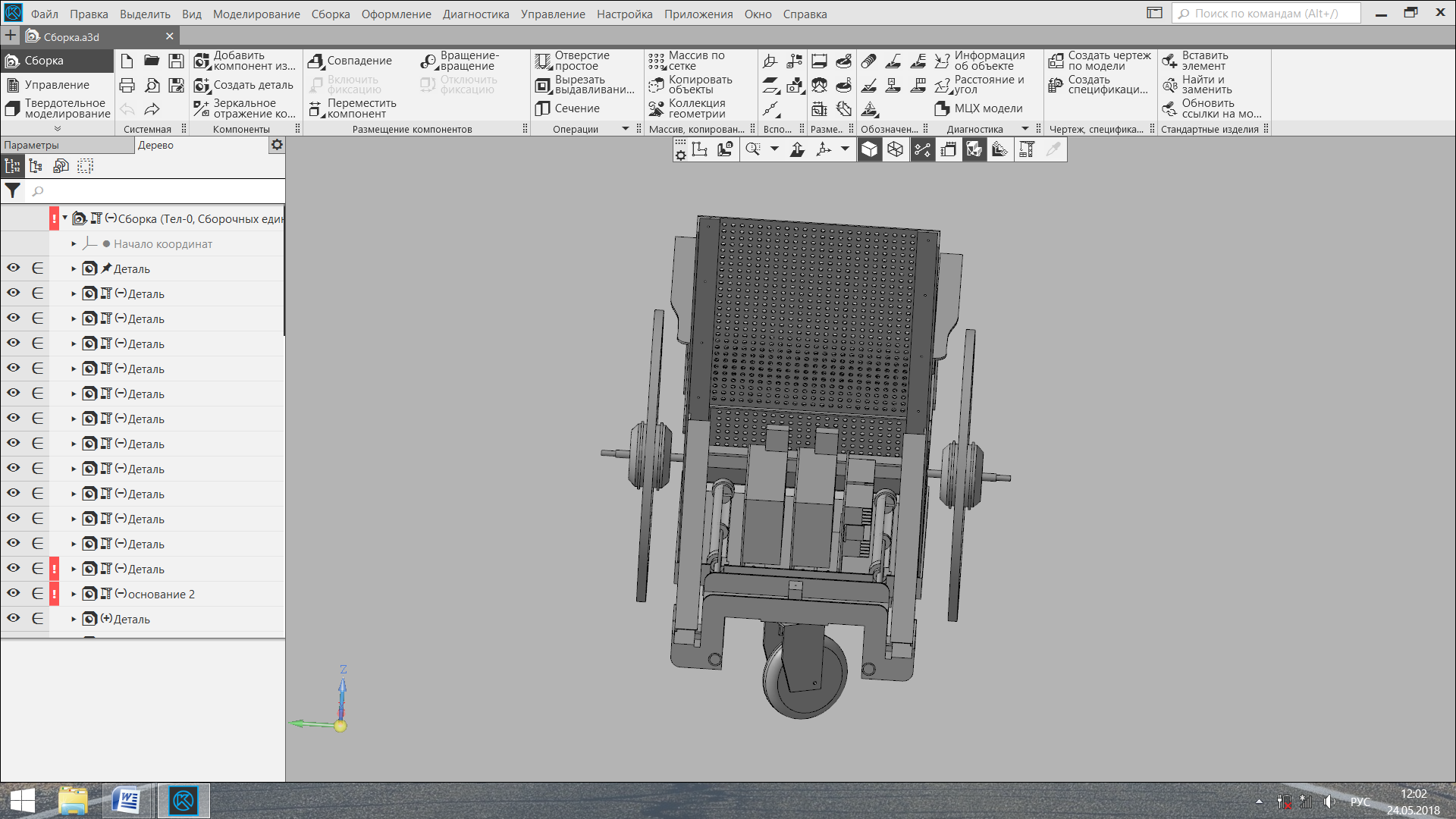


Рис. 14. Вид 3D-модели сзади с добавленными элементами крепления

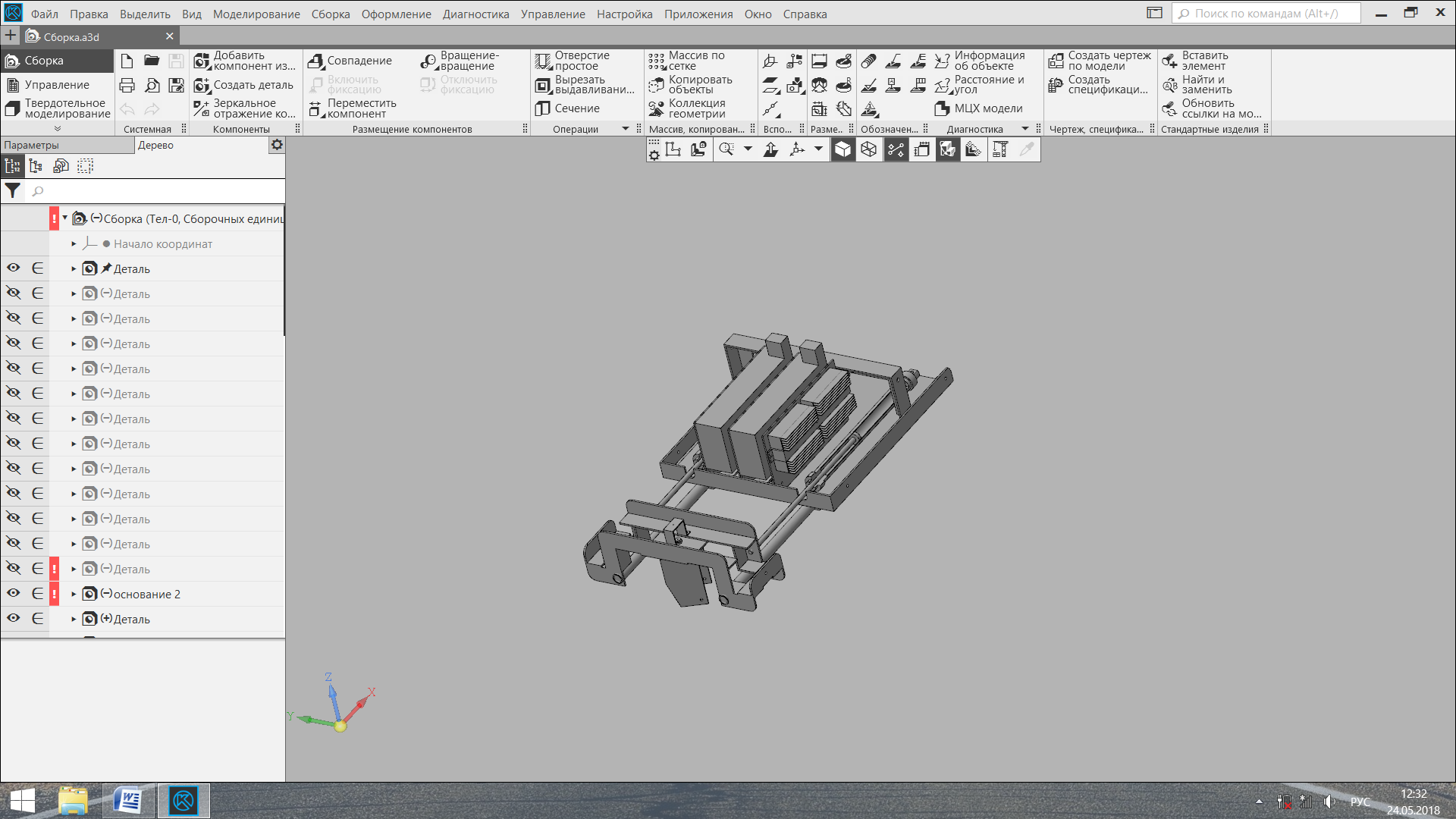


Рис. 15.Основание 3D-модели с добавленными компонентами (источники питания и контроллеры) и элементами крепления



Рис. 16. Источник питания

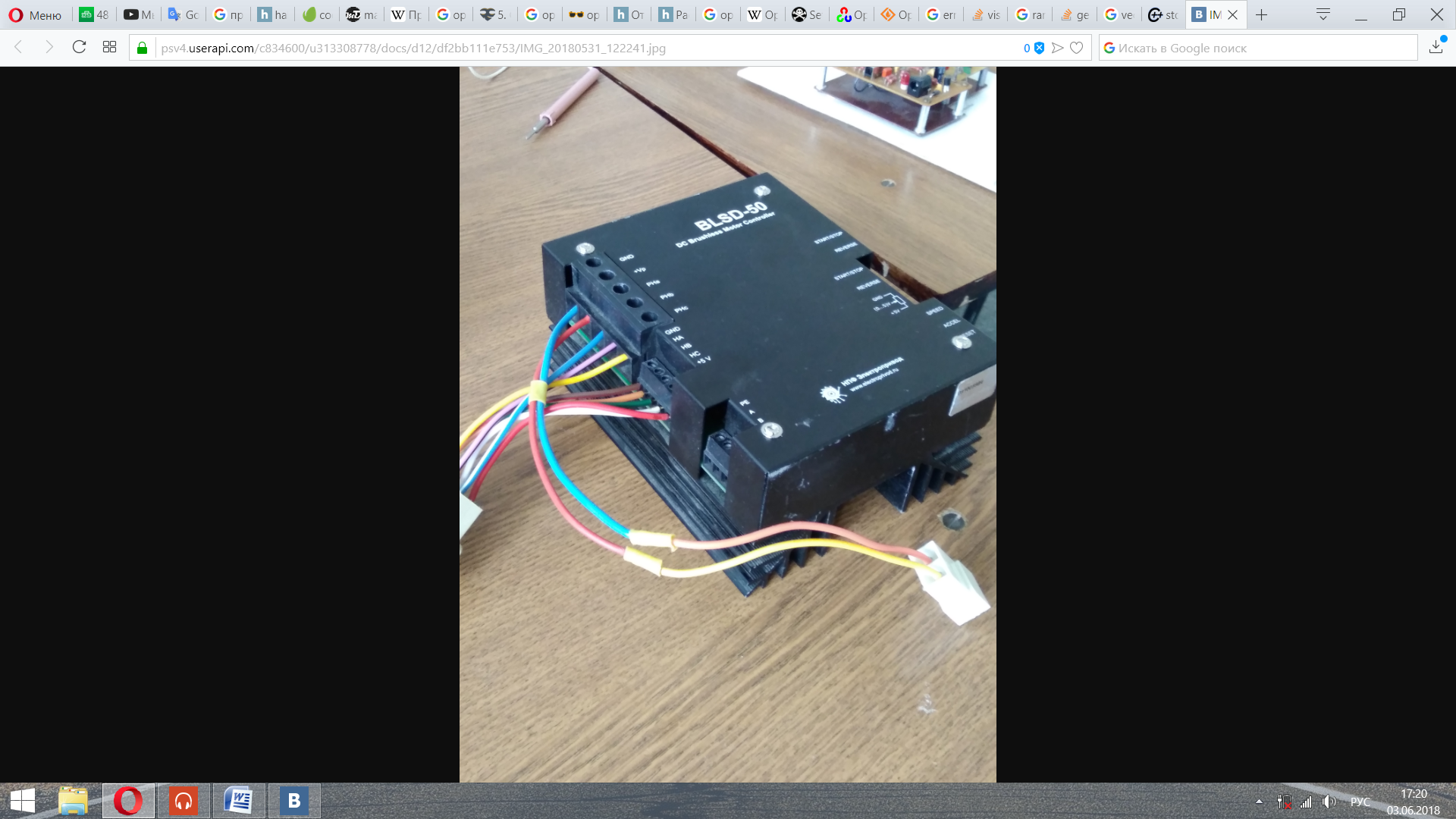


Рис. 17. Контроллер управления двигателями

### 4.1.2.Примеры чертежей:

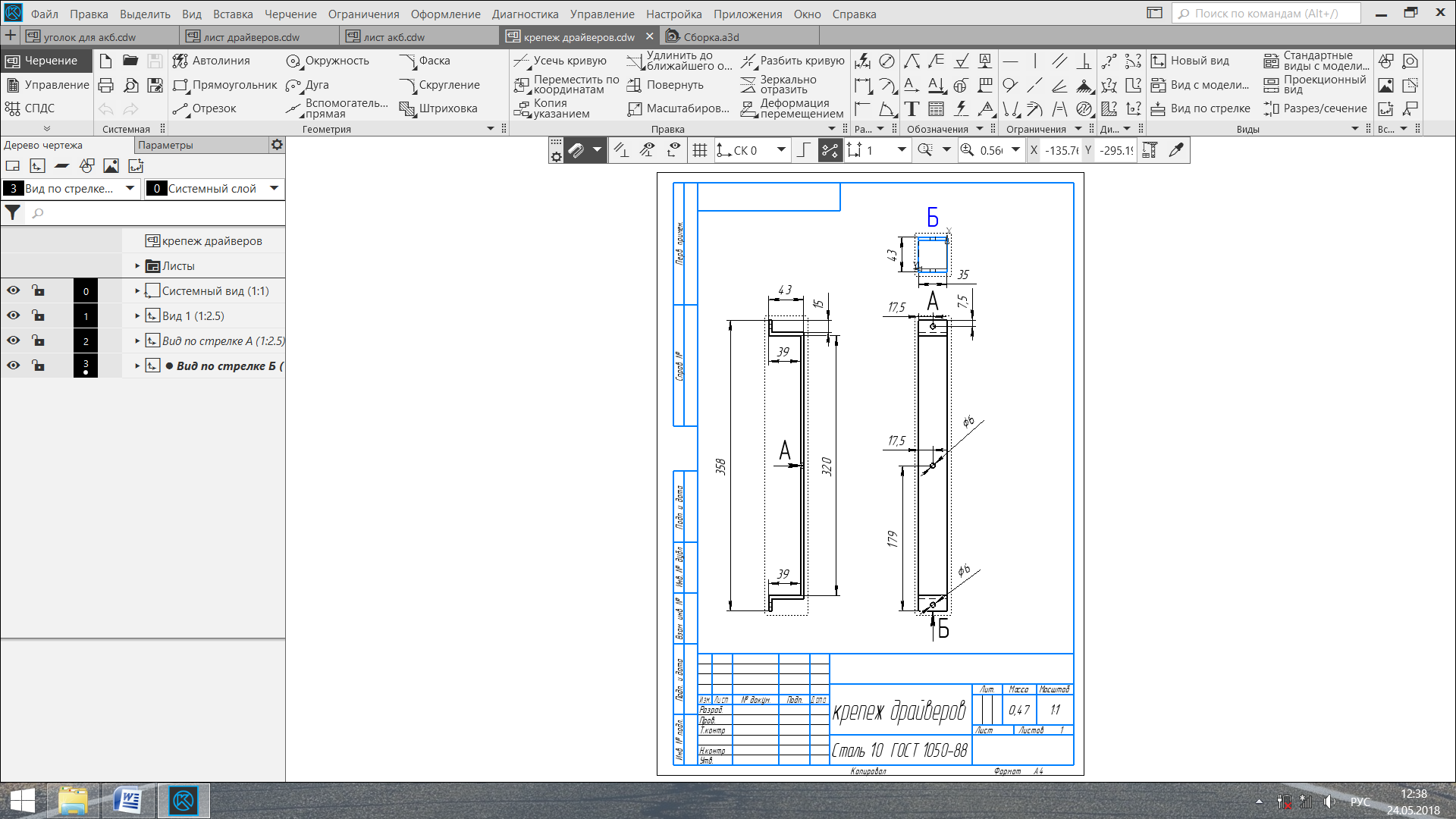


Рис. 18.Чертеж элемента крепления драйвера (крепежная скоба)

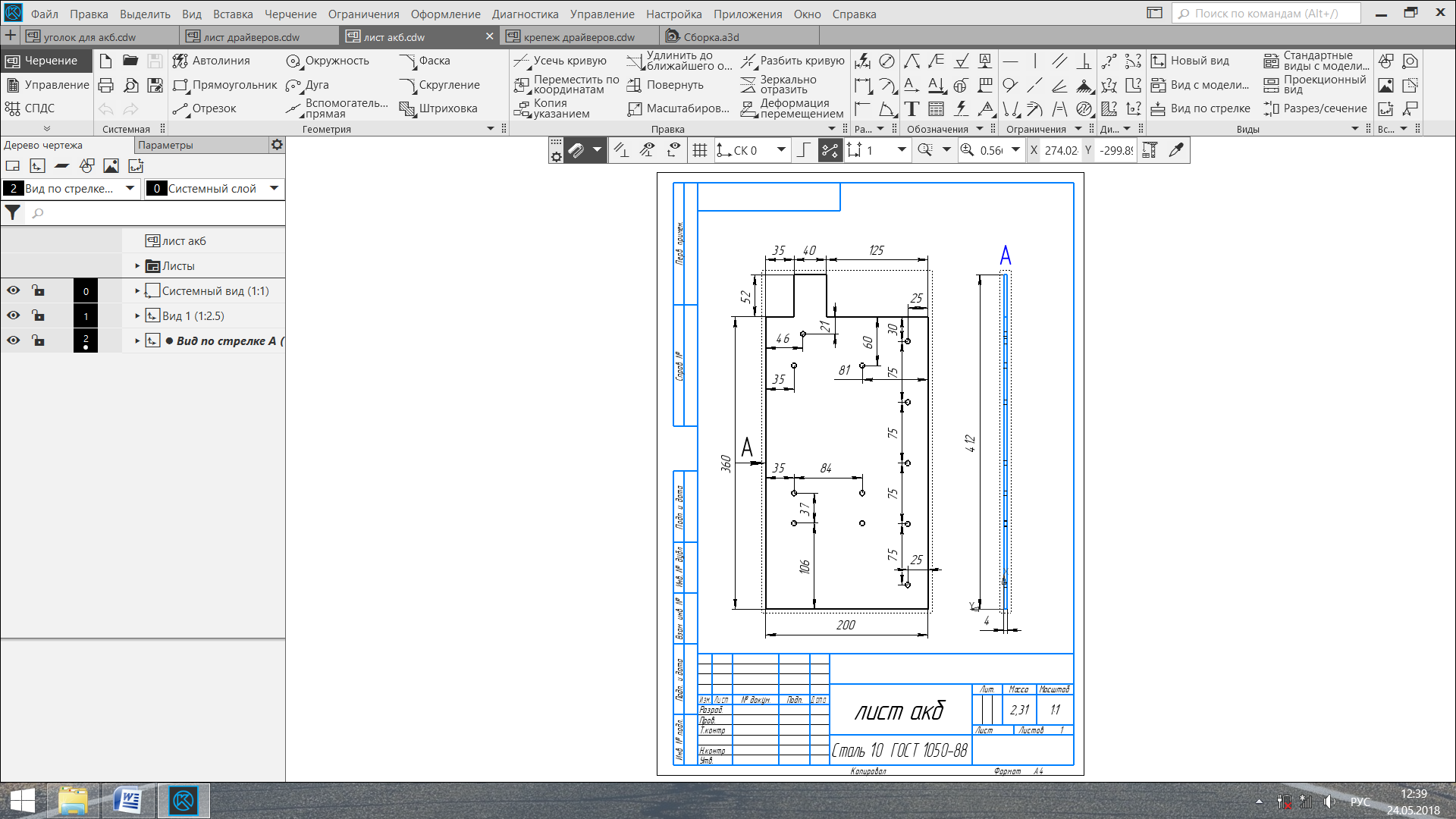


Рис. 19.Чертеж основания крепления аккумуляторов

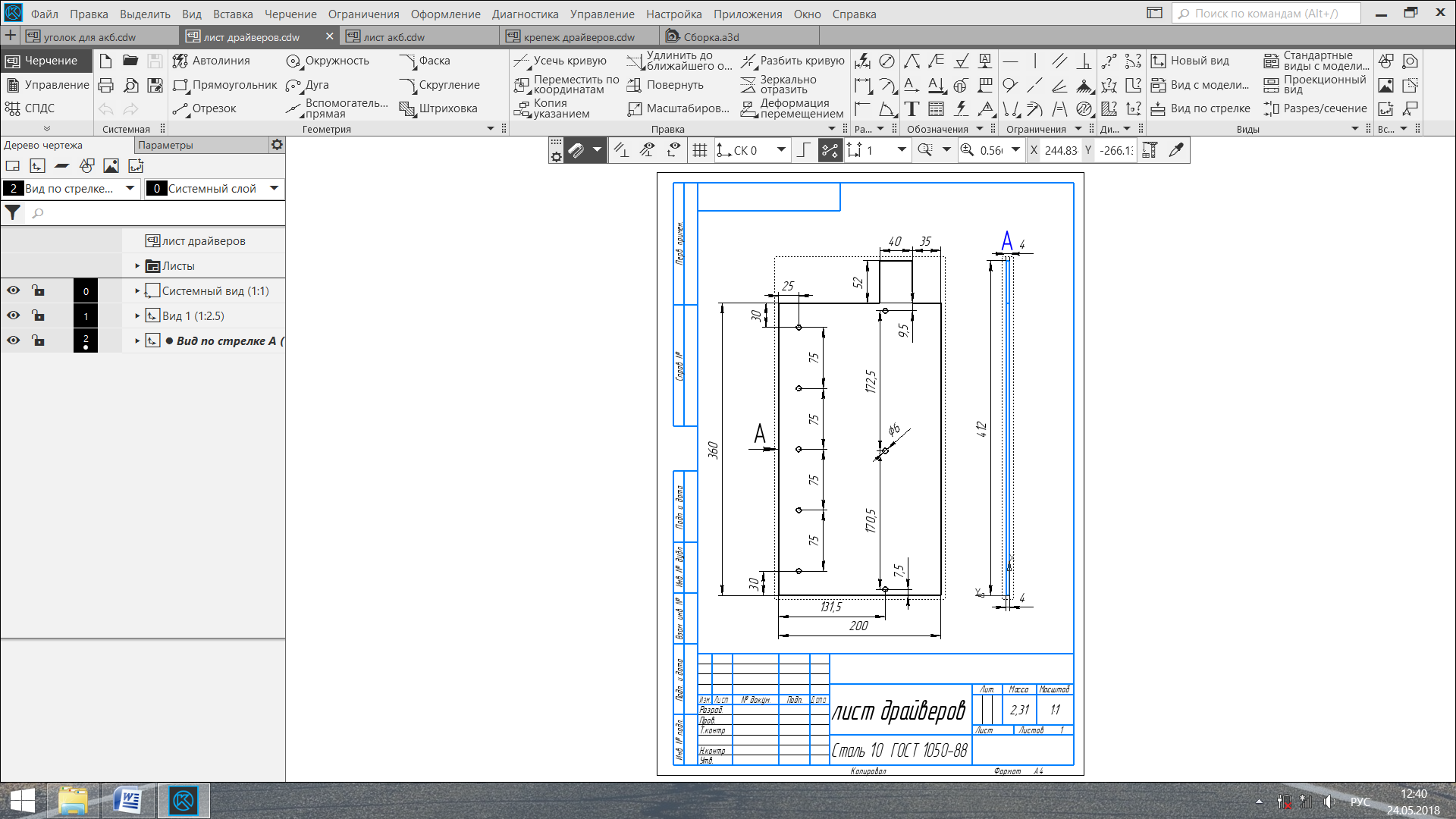


Рис. 20.Чертеж основания крепления драйверов

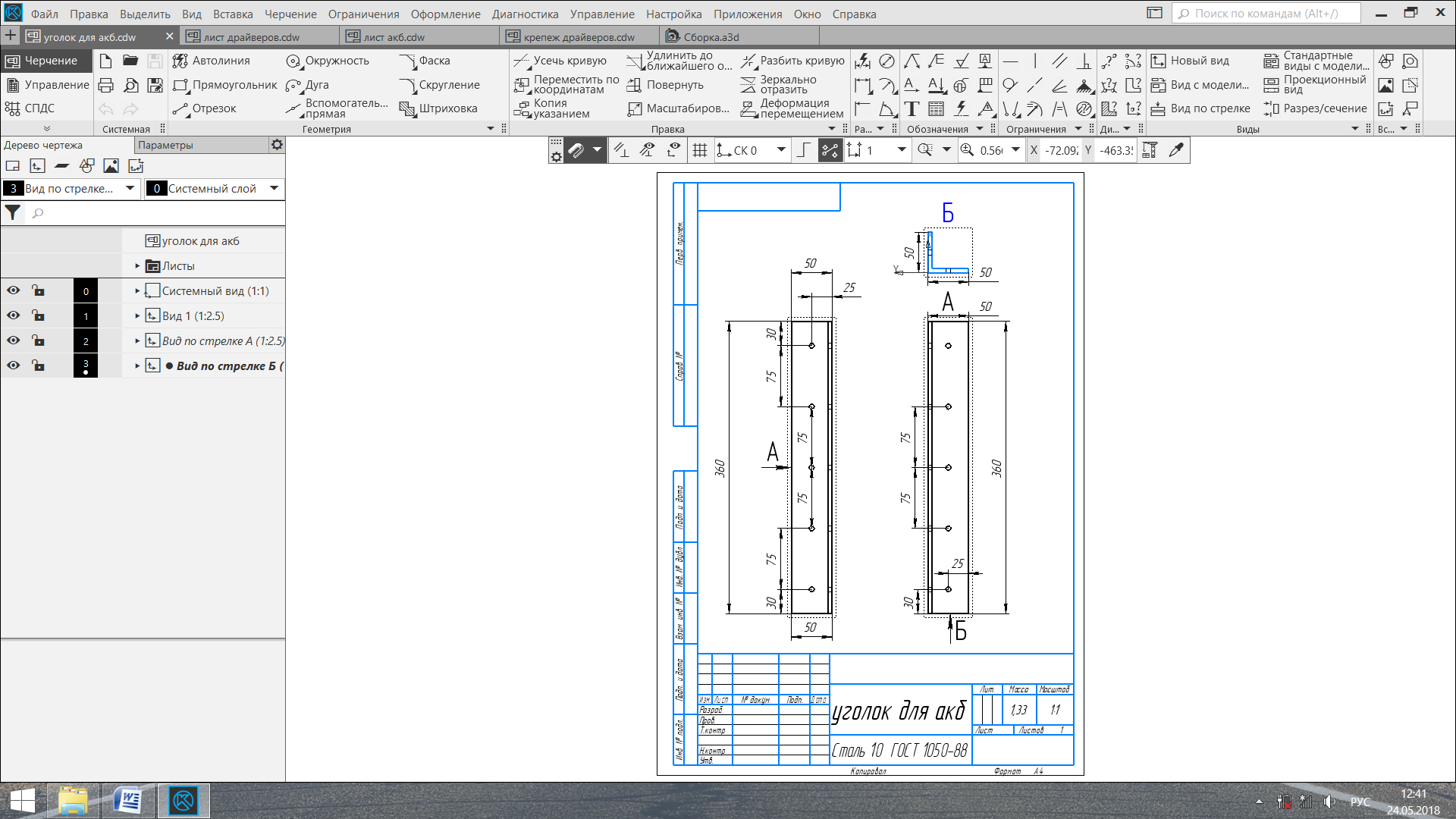


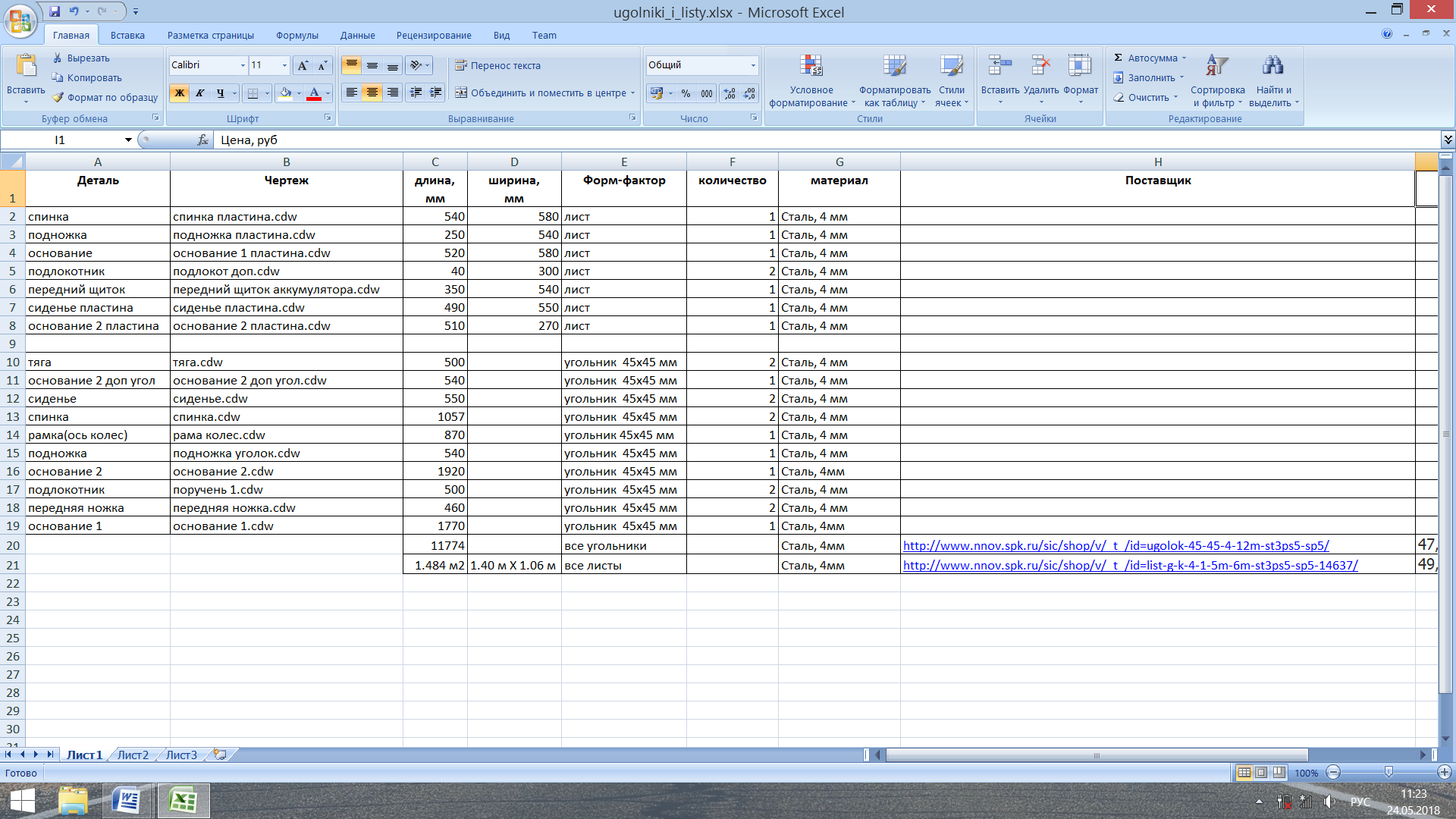
Рис. 21.Чертеж элемента крепления оснований добавленных элементов (аккумуляторов и драйверов) к основанию конструкции

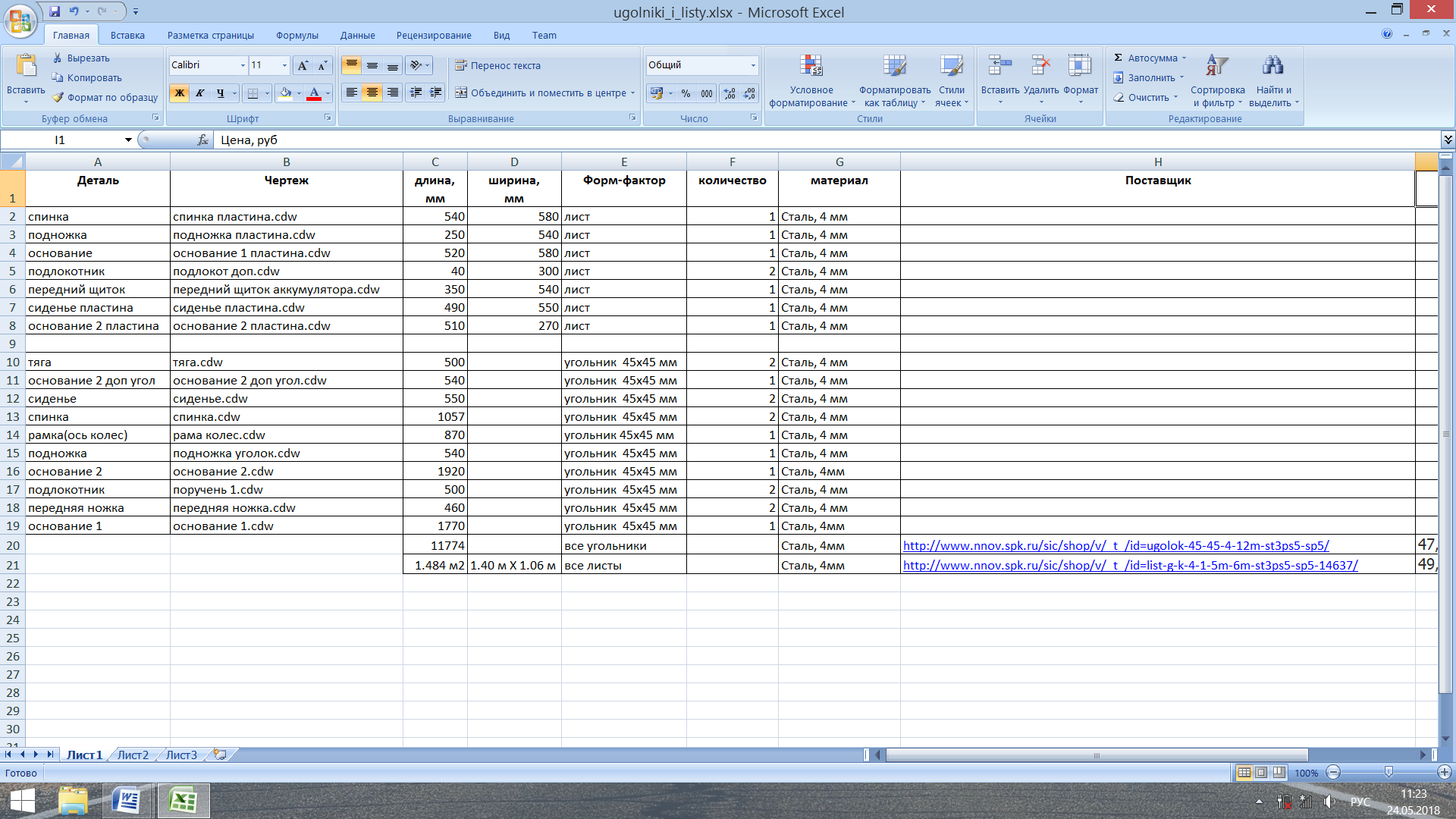
Чертежи разрабатывались в среде КОМПАС вер. 14 по завершении работ были размещены в репозиторииBITBUCKET:

https://bitbucket.org/VPRG/rinf/ [chertezhi/](https://bitbucket.org/VPRG/rinf/commits/6e11c084e98921f6d8d3f949a40a29dfb5c2a28c#chg-chertezhi/%D0%B4%D0%B5%D1%82%D0%B0%D0%BB%D0%B8/%D0%BA%D1%80%D0%B5%D0%BF%D0%B5%D0%B6%20%D0%B4%D1%80%D0%B0%D0%B9%D0%B2%D0%B5%D1%80%D0%BE%D0%B2.cdw)

### 4.1.3.Перечень конструктивных элементов и материалов

В ходе выполнения работы был проведен поиск необходимых конструктивных материалов для закупки в местных розничных базах. По результатам поиска был составлен прайс-лист. Расчет необходимого количества конструктивных материалов был выполнен оптимальным образом.





## 4.2.Заключение:

В ходе выполнения работы:

- была доработана 3D модель электрической тележки RINF в части размещения и крепления источников питания и контроллеров управления двигателями;

- был подготовлен комплект чертежей элементов крепления;

- был подготовлен перечень конструктивных элементов и материалов для закупки.

# 5. Сенсорно-сигнальная подсистема(SSRL)

## 5.1. Цель

Создание программного обеспечения для получения из 3D сцены распределение дальности по азимутальному углу.

## 5.2. Задачи

Для обеспечения корректного движения платформы «Ринф» необходима система, позволяющая сканировать окружающую обстановку и обнаруживать препятствия. Данная функция реализована в сенсорно‑сигнальной подсистеме. Её основу составляет времяпролётная3D-камера ETOF‑114, которая позволяет получить зависимость дальности до ближайших объектов, окружающих роботизированную транспортную платформу, от направления в пространстве. На данном этапе для работы алгоритмов навигации и автономного движения необходимо разработать программное обеспечение, решающее следующие задачи:

1. Потоковая обработка карт дальности, полученных с помощью камеры ETOF‑114.
2. Создание алгоритма для преобразования карт двумерного распределения дальности в одномерное распределение дальности по азимутальному углу.

## 5.3. Результаты

Для решения поставленных задач была создана библиотека потоковой обработки 3D-сцен. Разработка программного обеспечения проводилась на языке C++ в среде разработке Eclipseи интегрировано с библиотекой алгоритмов компьютерного зрения и обработки изображений OpenCV. Интерфейс библиотеки реализован в классе STOFRadar (листинг 1).

Данные об окружающей обстановке выводится в форме карты дальности (объект типа Mat), в котором интенсивность пикселей пропорциональна дальности,или в виде вектора (объект типа vector<int>), т. е. одномерной зависимости дальности от горизонтальной координаты пикселей. В зависимости от передаваемых функции GetDistanceArray (см. листинг 1) параметров вектор может формироваться либо одним из сечений карты дальности, либо усреднёнными значениями по каждому столбцу изображения (листинг 2). Режим работы камеры задаётся соответствующими параметрами, передаваемыми в конструктор класса в виде структур типа STOFParameters и STOF\_Radar\_Parameters. Структура типа STOFParameters задаёт следующие параметры камеры (листинг 3):

а.) частоту экспонирующего излучения;

б.) минимальный порог обрезания по интенсивности;

в.) минимальное и максимальное возможные значения регистрируемой дистанции;

г.) кадровую частоту;

д.) время накопления;

|  |
| --- |
| Листинг 1. Класс интерфейса библиотеки |
| classSTOFRadar  {  public:  /\*\*Конструктор с параметрами.  \*@paramSTOFParameters - параметрыкамеры.\*/  STOFRadar( STOFParametersparam, STOF\_Radar\_Parametersradar\_param);  /\*\*Зафиксировать карту дальности.  \* @return возвращает карту дальности. \*/  MatGetDistanceMap();  /\*\*Зафиксировать карту интенсивности.  \*@return - возвращает карту интенсивности. \*/  MatGetIntenseMap();  /\*\*Получить вектор дальности, который описывает зависимость  \* измеренной дальности от индекса пикселя.  \*@paramap - режимы захвата;  \*@return - возвращает вектор дальности.\*/  voidGetDistanceArray(vector<int>& buffer, array\_paramap);  private:  /\*\*Драйверсервиса TOF-камеры ETOF-110.\*/  STOF\_Grabber\* s\_grab;  /\*\*Параметрыфильтров.\*/  STOF\_Radar\_Parametersradar\_parameters;  /\*\*Рабочиепараметрыкамеры.\*/  STOFParametersSTOF\_parameters;  /\*\*Временная выборка карт дальности (последние захваченные кадры).\*/  vector<Mat>history;  /\*\*Параметры карт дальности.\*/  intbyte\_in\_depth\_pixel, byte\_in\_intensity\_pixel , width, height;  }; |

|  |
| --- |
| Листинг 2. Режимы захвата вектора дальности. |
| enumarray\_param {  MIN, //возвращает минимальные значения дистанции для каждого столбца  MIDDLE, //возвращает значения строки, находящейся на уровне 1/2  TOP\_QUARTER, //возвращает значения строки, находящейся на уровне 1/4  LOW\_QUARTER, //возвращает значения строки, находящейся на уровне 3/4  SUMM //SUMM - для каждой строки возвращает сумму пикселей, находящихся на  уровне от 1/4 до 3/4  }; |

|  |
| --- |
| Листинг 3. Рабочие параметры камеры. |
| /\*\* STOFParameters - задаёт рабочие параметры камеры\*/  structSTOFParameters  {  constchar\* url; //urlкамеры  unsignedintfreq; //частота, кГц  unsignedint fps; //количествокадроввсекунду  boolauto\_freq\_tunning; //включить автоматическую подстройку частоты  boolon\_filter; //включить фильтрацию  unsignedintcummulateTime; //время накопления, тысяч отсчётов  unsignedintthreshold; //порог отсекания по интенсивности, тыс. ед.  }; |

Для устранения шумов может производиться медианная фильтрация полученных карт по координатам изображений, а также их усреднение по временной выборке из последних захваченных кадров. Размеры фильтрующих окон и временной выборки могут контролироваться соответствующими параметрами, передаваемыми вконструктор класса STOFRadarв виде структуры типа STOF\_Radar\_Parameters (листинг 4).

В листинге 5 приведён пример использования библиотеки. Здесь показаны инициализация и передача параметров камеры, захват и запись в yml-файл карт дальности и интенсивности, а также вектора дальности. Для фильтрации данных по времени необходимо заполнить выборку до определённого размера. При последующем захвате карты выборка начинает перезаписываться с первого элемента, подобно кольцевому буферу.

|  |
| --- |
| Листинг 3. Параметры фильтрации. |
| /\*\*image\_filter - задаёт размер фильтра по координате.\*/  enumimage\_filter { NO\_IFILTER, X3, X5, X7 };  /\*\* time\_filter - задаёт размер временной выборки.\*/  enumtime\_filter { NO\_TFILTER, S3, S5, S7, S9 };  /\*\*  \* Параметры фильтров  \* im\_filter - размер фильтра по координатам  \* t\_filter - размер фильтра по времени  \*/  structSTOF\_Radar\_Parameters  {  image\_filterim\_filter;  time\_filter t\_filter;  }; |

|  |
| --- |
| Листинг 4. Простой пример использования библиотеки. |
| #include"stof.nsmap"  #include"STOF\_Grabber.h"  #include<opencv2/opencv.hpp>  #include"STOFRadar.h"  usingnamespacestd;  usingnamespace cv;  int main(intargc, char\* argv[])  {  //Инициализируемпараметрыкамеры  STOFParametersparam = { "http://192.168.9.110/soap/",  26000, 25, true, true, 550000, 120 };  //Инициализируемпараметрыфильтрации  STOF\_Radar\_Parametersradar\_param;  radar\_param.im\_filter = X5;  radar\_param.t\_filter = S3;  //Передаёмпараметрывконструктор  STOFRadar\* radar = newSTOFRadar(param, radar\_param);  vector<int>distance\_info;  Mat distance\_frame;  Matintensity\_frame;  /\* Ждём полного заполнения временной выборки карт дальности до размера,  определённого в параметре radar\_param.t\_filter  for (inti = 0; i<radar\_param.t\_filter; i++)  {  //Захваткартыдальности.  distance\_frame = radar->GetDistanceMap();  //ЗахваткартыИнтенсивности.  intensity\_frame = radar->GetIntenseMap();  //Захватвекторадальности.  radar->GetDistanceArray(distance\_info, MIDDLE);  }  //Сохраняем данные в yml-файл.  FileStorage storage("results//frame.yml", cv::FileStorage::WRITE);  storage <<"distance\_frame"<<distance\_frame;  storage <<"intensity\_frame"<<intensity\_frame;  storage <<"distance\_vector"<<distance\_info;  storage.release();  distance\_frame.release();  intensity\_frame.release();  return 0;  } |

Примеры карт дальности, полученных при разных параметрах фильтрации приведены на рис. 22. На них можно ясно различить дверной проём, стены и различные препятствия. Соответствующие этим изображениям сечения на уровне ½ по высоте приведены на рис. 23.

|  |  |
| --- | --- |
| E:\Хранилище  документов\антон\учёба\5 курс (физфак)\2 семестр\Робототехника (Минеев)\results\NO_FILTERS\frame.bmp | E:\Хранилище  документов\антон\учёба\5 курс (физфак)\2 семестр\Робототехника (Минеев)\results\TF&PF\frame1.bmp |
| а.) нет фильтрации | в.) фильтрация по времени |
| E:\Хранилище  документов\антон\учёба\5 курс (физфак)\2 семестр\Робототехника (Минеев)\results\PF\frame.bmp | E:\Хранилище  документов\антон\учёба\5 курс (физфак)\2 семестр\Робототехника (Минеев)\results\TF&PF\frame1.bmp |
| б.) фильтрация по координатам | г.) фильтрация по времени и по координатам |
| Рис. 22. Карты дальности при различных параметрах фильтрации. | |

|  |  |
| --- | --- |
| E:\Хранилище  документов\антон\учёба\5 курс (физфак)\2 семестр\Робототехника (Минеев)\картинки\no_filters.png | E:\Хранилище  документов\антон\учёба\5 курс (физфак)\2 семестр\Робототехника (Минеев)\картинки\spatial_filter.png |
| а.) нет фильтрации | б.) фильтрация по координатам |
| E:\Хранилище  документов\антон\учёба\5 курс (физфак)\2 семестр\Робототехника (Минеев)\картинки\time_filter.png | E:\Хранилище  документов\антон\учёба\5 курс (физфак)\2 семестр\Робототехника (Минеев)\картинки\time-spatial_filter.png |
| в.) фильтрация по времени | г.) фильтрация по времени и по координатам |
| Рис. 23. Сечения 3D-сцены при различных параметрах фильтрации. | |

## 5.4. Выводы

На данный момент были решены следующие задачи:

1. В ходе работы была разработана библиотека для потоковой обработки 3D-сцены.
2. Преобразование карт двумерного распределения дальности в одномерное распределение дальности по азимутальному углу.

Для дальнейшей работы с системой необходимо провести калибровку камеры, т. е. сопоставить горизонтальные координаты полученных сечений с азимутальным углом.

# Литература

1. Официальный сайт Pixhawk. – https://pixhawk.org.
2. Pixhawk Developer Guide. – https://dev.px4.io/en/apps/hello\_sky.html.
3. NEO-M8P u-blox M8 High Precision GNSS Modules Data Sheet. Rev. 0.6 16.05.2017.
4. Here+ GNSS User Guide. V1.0 Aug 2017.
5. Документация по приложению APMPlanner. – http://ardupilot.org/planner2.
6. Cеть базовых станций RTK. – http://rtknet.ru.