МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Воронежский государственный технический университет»

Кафедра радиоэлектронных устройств и систем

ОСНОВЫ ПРОЕКТИРОВАНИЯ ПРОГРАММНО-АППАРАТНЫХ КОМПЛЕКСОВ И СИСТЕМ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к выполнению лабораторной работы №2
для студентов специальности 11.05.01
«Радиоэлектронные системы и комплексы»
очной формы обучения

Воронеж 2024

УДК 621.396.6.001.63(07)
ББК 32.844я7

Составитель А. И. Сукачев

Основы проектирования программно-аппаратных комплексов и систем: методические указания к выполнению лабораторной работы №2 для студентов специальности 11.05.01 «Радиоэлектронные системы и комплексы» очной формы обучения / ФГБОУ ВО «Воронежский государственный технический университет»; сост.: А. И. Сукачев. – Воронеж: Изд-во ВГТУ, 2024. – 30 с.

В соответствии с рабочими учебными программами дисциплин приведены описания методов измерений и методик выполнения лабораторных работ, изложены теоретические сведения, содержащие основы использования микропроцессорной техники. По каждой лабораторной работе в описание включены: теоретические материалы, используемое оборудование и приборы, порядок подготовки и проведения работы.

Предназначены для студентов специальности 11.05.01 «Радиоэлектронные системы и комплексы» очной формы обучения.

Методические указания подготовлены в электронном виде и содержатся в файле МУ_ОППАКиС_ЛР2.pdf.

Ил. 29. Табл. 2. Библиогр.: 6 назв.

УДК 621.396.6.001.63(07)
ББК 32.844я7

Рецензент – А. В. Останков, д-р техн. наук, профессор кафедры радиотехники ВГТУ

Издается по решению редакционно-издательского совета Воронежского государственного технического университета

ЛАБОРАТОРНАЯ РАБОТА №2
ИЗУЧЕНИЕ ОСНОВ РАЗРАБОТКИ ПРОГРАММНО-АППАРАТНЫХ КОМПЛЕКСОВ НА БАЗЕ СЕМЕЙСТВА МИКРОКОНТРОЛЛЕРОВ ESP32

Цель лабораторной работы: Изучение основ программирования микроконтроллеров семейства ESP.

1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Первая лабораторная работа была связана только со знакомством с микроконтроллерами и лишь поверхностно раскрыла данную тему. Эта же лабораторная работа является гораздо более глубокой, так как в ней будет помимо всего прочего изучена довольно большая часть курсовой работы – получения изображения и передача его на сервер, где будет происходить его дальнейшая обработка.

2. ИНТЕРФЕЙСЫ SPI, I2C

SPI – Serial Peripheral Interface – 3-х проводная шина передачи данных, предназначенная для работы с переферийными устройствами. Состоит из 3-х проводов: SCK – тактовый сигнал, MOSI (Master Output Slave Input) - выход главного устройства – вход подчинённого, MISO (Master Input Slave Output) – вход главного, выход подчинённого. Также присутвует дополнительный 4-й провод, который отвечает за выбор подчинённого устройства – SS (CS) Slave Select (Chip Select). Данный интерфейс используется для работы с небольшими ЖК и TFT дисплеями, для работы с SD картами, когда не нужна сверхвысокая скорость обмена данными. В качестве примера работы с данным интерфейсом подключим ЖК дисплей от Nokia 5110 к Arduino Nano, как это показано в табл. 1.

Таблица 1

Подключение дисплея к Arduino Nano

Nokia 5110 LCD	Arduino Nano
RST	D3 (3)
CE	D4 (4)
DC	D12 (12)
DIN	D11 (11)
CLK	D13 (13)
BL	5V
GND	GND
VCC	5V

Далее необходимо скачать библиотеку LCD5110_Basic и установить (инструкция по установке библиотек приведена в следующем пункте). Далее необходимо запустить пример (Пример нужно выбрать для плат AVR и выбрать LCD5110_Bitmap, сам файл находится в формате .pde (старый формат Arduino проектов)).

В самом примере необходимо сконфигурировать выводы дисплея согласно табл. 1. После загрузки проекта на экране должен появиться логотип Arduino. Выводы для дисплея можно назначить и другие, но нужно помнить, что использование SCK и MOSI отличные от аппаратных снижают производительность программы.

Следующим этапом познакомимся с интерфейсом I2C на примере модуля часов реального времени DS1307. Для этого необходимо скачать и установить библиотеку RTClib из «Менеджера библиотек» Arduino.

I2C — это 2-х проводная шина, которая обеспечивает низкую скорость передачи данных, но позволяет подключить до 127 устройств. Главной же особенностью данного интерфейса является необходимости подключения подтягивающих резисторов к выводам SDA (Данные), и SCL (Тактовый сигнал), их номинал зависит от напряжения питания устройств, их количества, а также от длины самой линии данных. В модулях Arduino данные резисторы уже установлены, либо их можно подключить через микроконтроллер (многие микроконтроллеры позволяют подключить встроенный подтягивающий резистор, в том числе и ATMEGA328).

Для разнообразия вместо использования примера из библиотеки, напишем простую программу, которая выведет текущее для модуля время на экран из предыдущего задания. Программа приведена ниже:

```
#include <LCD5110_Basic.h> //Библиотека для работы с дисплеем
#include <Wire.h> //Библиотека для работы с I2C
#include <RTClib.h> //Библиотека для работы с DS1307
LCD5110 LCD(13,11,12,3,4);//Инициализируем объект LCD
extern uint8_t SmallFont[]; //Инициализируем шрифт
RTC_DS1307 rtc;// Инициализируем объект rtc
String time; //Переменные для хранения даты и времени в формате для вывода
String date;
//Модуль подключается следующим образом:
//SDA -> A4
//SCL -> A5
//VCC -> 5V
//GND -> GND
void setup() {
// put your setup code here, to run once:
Serial.begin(9600);
```

```
if (! rtc.begin()) {
Serial.println("Couldn't find RTC");
while (1);
}
if (! rtc.isrunning())
Serial.println("RTC is NOT running!");

LCD.InitLCD();
LCD.clrScr();
LCD.disableSleep();
LCD.setFont(SmallFont);
}

void loop() {
// put your main code here, to run repeatedly:
LCD.clrScr();
DateTime dateTime = rtc.now();
uint8_t hh = dateTime.hour(); // Получаем часы
uint8_t mm = dateTime.minute(); // Получаем минуты
uint8_t ss = dateTime.second(); // Получаем секунды
//Аналогично с датой, только возьмём для года uint16_t
uint16_t year = dateTime.year();
uint8_t month = dateTime.month();
uint8_t day = dateTime.day();

time = String(hh);
time += ':';
time += String(mm);
time += ':';
time += String(ss);

date = String(day);
date += '/';
date += String(month);
date += '/';
date += String(year);

LCD.print(time,10,10); // Выводим время
LCD.print(date,10,20); // Выводим дату
LCD.print("RP-193!",10,30);
Serial.println(time);
Serial.println(date);
delay(1000);
```

3. ЗНАКОМСТВО С МИКРОКОНТРОЛЛЕРАМИ СЕМЕЙСТВА ESP

В прошлой работе мы изучали микроконтроллеры AVR. Они являются на данный момент устаревшими, поэтому для сложных задач они не подходят. Из современных и популярных решений для платформы Arduino существуют микроконтроллеры ESP: ESP8266 и ESP32. Первая, является младшей моделью в ряде и в основном предназначена для работы с Wi-Fi и простой автоматизации. Второй же модуль на порядок производительнее, а также имеет полноценный Bluetooth, по которому помимо данных даже можно передать звук, т.е. сделать Bluetooth колонку. Именно с данного пункта мы и начнём знакомство с ESP32, тем более для работы с ESP32-CAM придётся сделать некоторые действия по настройки Arduino IDE.

Для того чтобы работать с микроконтроллерами ESP32 и ESP8266 необходимо добавить в Arduino IDE ссылки на информацию об этих платах в менеджер плат. Для этого нужно открыть Arduino IDE и в ней выполнить следующие действия: «Файл – Настройки – Дополнительные ссылки для менеджера плат». В данную строку вставить следующую ссылку:

```
https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json,http://arduino.esp8266.com/stable/package_esp8266com_index.json
```

В данном случае в этой строке целых 2 ссылки: для ESP8266 и ESP32, разделены они через запятую. После добавления этих ссылок надо зайти во вкладку: «Инструменты – Плата – Менеджер плат», где по ключевым словам ESP8266 и ESP32 установить необходимые пакеты.

Далее скачаем библиотеку для работы с A2DP. A2DP – это протокол передачи звуковых данных по Bluetooth, ссылка на Git-hub проекта приведена ниже (В интернете ищется по слову ESP32 A2DP):

```
https://github.com/pschatzmann/ESP32-A2DP
```

Скачанный архив нужно разархивировать (в Linux можно скачать папку через git clone). Далее нужно вспомнить директорию, куда была установлена Arduino IDE и перейти в папку с программой, там будет папка libraries. Папку из архива нужно переместить именно туда (рис. 1).

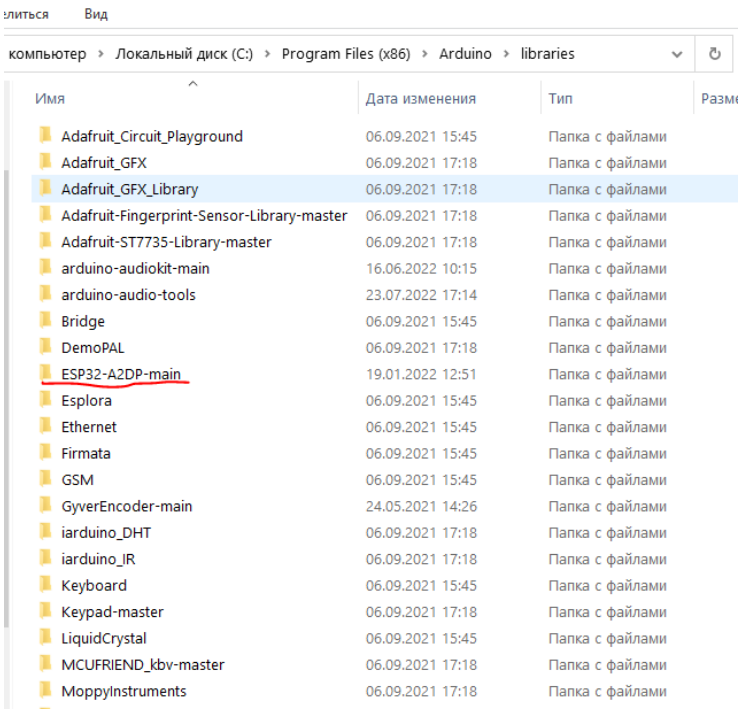

Рис. 1. Установленная библиотека

Далее нужно открыть папку с библиотекой, перейти в раздел «examples» и найти пример «bt_music_receiver_to_internal_dac» - это пример для работы со встроенным ЦАП, так как внешней микросхемы ЦАП в лаборатории на данный момент не имеется.

Для того, чтобы послушать музыку, через такую «импровизированную колонку» нужно подключить динамик согласно схеме (рис. 2). ЦАП в ESP32 стереофонический, но 8-битный, поэтому звук будет с явным шипением, особенно на малой громкости, выводы ЦАП находятся на GPIO25 и GPIO26. Также звук будет не громкий, так как динамик будет подключен напрямую к выход микроконтроллера, что делать не рекомендуется, так как в данном случае происходит перегрузка выхода.

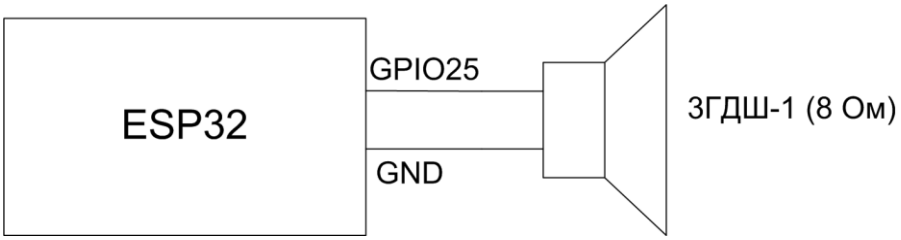

Рис. 2. Схема подключения динамика

Загружаем пример в микроконтроллер, в отличие от Arduino ESP требует удержание кнопки «BOOT» (GPIO0 или IO0) во время прошивки. Это запускает внутренний загрузчик в режим ожидания прошивки.

После прошивки микроконтроллера подключаемся к ESP32 по Bluetooth и запускаем любой звук или музыку. Должно начаться воспроизведение зву-

ка/музыки из динамика. Также в примере можно переименовать наше аудиоустройство.

4. ДОПОЛНИТЕЛЬНЫЕ СВЕДЕНИЯ

Писать программы в Arduino IDE крайне неудобно, поэтому существует как минимум два решения: одно платное, другое бесплатное. Платное решение основывается на использовании связки Visual Studio и расширения VisualMicro. Бесплатное – на Visual Studio Code и расширении Arduino. Бесплатное решение требует некоторого знания чёрной магии, так как настроить все зависимости для компилятора в ней сложно. Поэтому для начала рассмотрим первый вариант, у него есть 45-дневный пробный период, а даже после истечения срока годности заблокируются возможности прошивать плату и настраивать библиотеки, поэтому данный вариант можно рассматривать только совместно с Arduino IDE, как средой для настройки проекта.

4.1. VISUAL STUDIO + VISUALMICRO

1. Переходим на сайт Microsoft и скачиваем Visual Studio Community 2022 (Ссылка: https://visualstudio.microsoft.com/ru/).
2. Запускаем установщик (возможно, потребуется VPN в связи с санкциями). Устанавливаем пакеты C++.
3. После установки Visual Studio заходим в неё и открываем вкладку «Расширения – Управлять расширениями». Вбиваем в поиск фразу «Visual Micro». Устанавливаем.
4. Первые 45 дней будет доступен полный функционал Arduino IDE в IDE здорового человека. После этого периода он сократиться лишь до возможности редактировать код.
5. Для примера откроем пример, рассмотренный выше. Для этого выполняем следующие действия: «Файл – Открыть – Arduino Project». Выбираем пример из предыдущего пункта и если код не подсвечивается красным, то всё хорошо и все зависимости VisualMicro определил. При попытке написать что-либо Visual Studio попытается дополнить наш код с помощью подсказок – это то, чего не хватает Arduino IDE.

4.2. VISUAL STUDIO CODE + ARDUINO

Здесь дела обстоят несколько сложнее. Как минимум по причине ручного конфигурирования среды. И так план:

1. Переходим на сайт Microsoft и скачиваем Visual Studio Code (Ссылка: https://visualstudio.microsoft.com/ru/). Данная программа есть и под Linux.
2. Устанавливаем.
3. Здесь всё делается через расширения, устанавливаем следующие расширения (рис. 3).

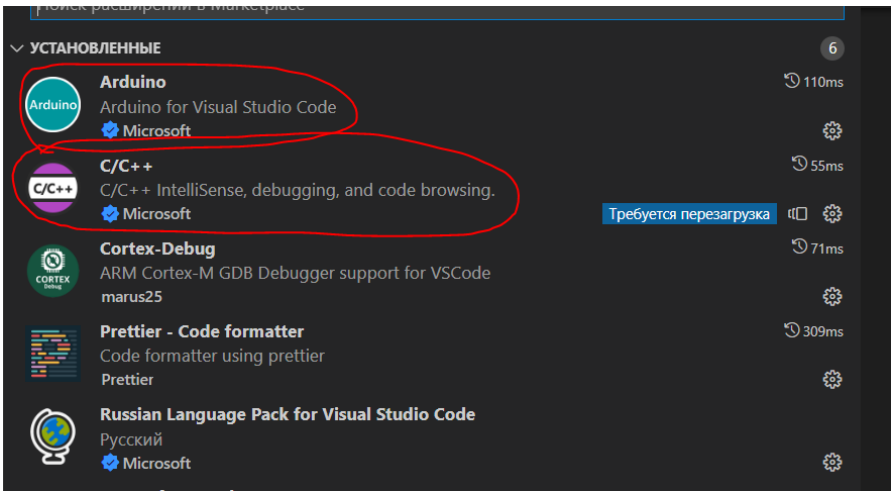

Рис. 3. Необходимые расширения

4. Выставляем первым делом Arduino Path (рис. 4).

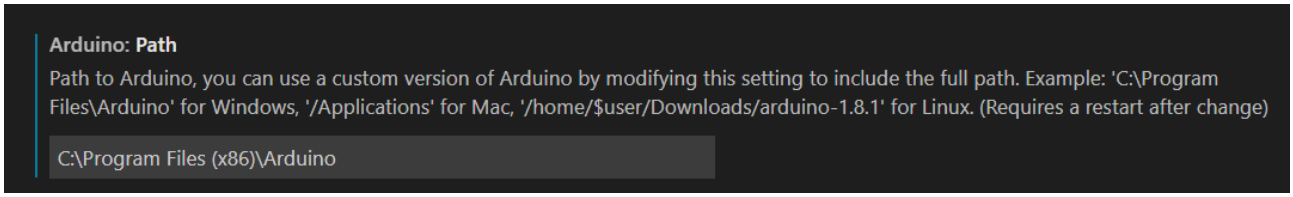

Рис. 4. Установка пути к Arduino IDE

5. Открываем файл проекта, зажимаем Ctrl + Shift + P и пишем «Arduino». В списке команд выбираем «Select Board» и «Select Serial Port».
6. Потом выбираем «Initialize», а вслед за ним и «Rebuild IntilliSense configuration». Потом нажимаем «Verify». После чего ошибки должны пропасть. Если ошибки продолжают сыпаться, то нужно будет прописать пути к библиотекам и компиляторам (Ищем в интернете).
7. Таким образом, получим код без ошибок, а самое главное будут работать подсказки (рис. 5).

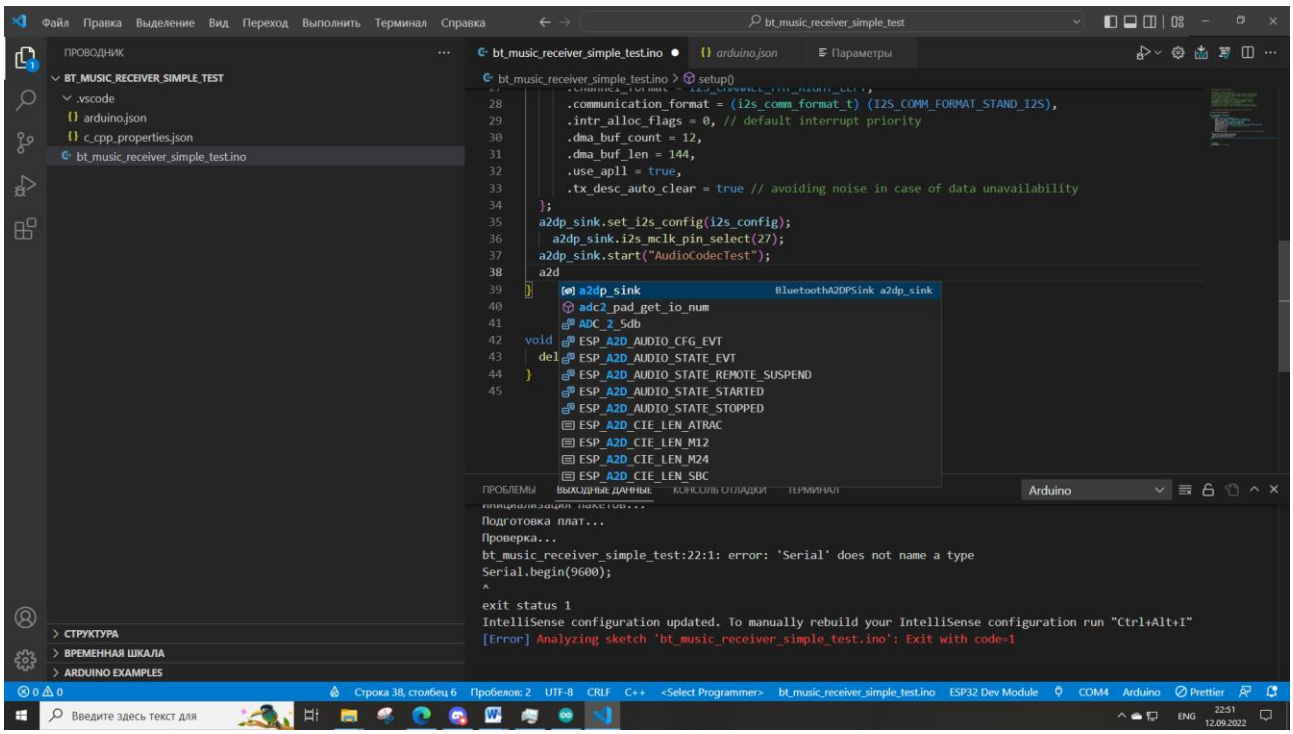

Рис. 5. Подсказки в Visual Studio Code

5. РАБОТА С ESP32-CAM

Теперь перейдём к теме курсовой работы – а именно охранной системе на базе ESP32-CAM. Будем реализовывать Wi-Fi камеру. Для этого загрузим пример из вкладки «Файл – Примеры – ESP32 – CameraWebServer». В качестве «ssid» указывается название Wi-Fi сети, а в поле «password» – пароль от неё. Плату выбираем «ESP32 – AI Thinker ESP32-CAM». И загружаем прошивку, удерживая кнопку «BOOT».

Далее нужно открыть терминал порта и увидеть прекрасную ошибку: «Camera unsupported». После этого нужно в настройках программы закомментировать один из #define и выбрать нужную модель, а именно: #define CAMERA_MODEL_AI_THINKER. Итоговый код приведён ниже на рисунке 6.

```
//
// WARNING!!! PSRAM IC required for UXGA resolution and high JPEG quality
//            Ensure ESP32 Wrover Module or other board with PSRAM is selected
//            Partial images will be transmitted if image exceeds buffer size
//

// Select camera model
//#define CAMERA_MODEL_WROVER_KIT // Has PSRAM
//#define CAMERA_MODEL_ESP_EYE // Has PSRAM
//#define CAMERA_MODEL_M5STACK_PSRAM // Has PSRAM
//#define CAMERA_MODEL_M5STACK_V2_PSRAM // M5Camera version B Has PSRAM
//#define CAMERA_MODEL_M5STACK_WIDE // Has PSRAM
//#define CAMERA_MODEL_M5STACK_ESP32CAM // No PSRAM
#define CAMERA_MODEL_AI_THINKER // Has PSRAM
//#define CAMERA_MODEL_TTGO_T_JOURNAL // No PSRAM

#include "camera_pins.h"
```

Рис. 6 . Настройка камеры

После этого снова загружаем скетч. Далее в терминале появиться ссылка на сайт, на котором будет видео с камеры (рис. 7).

```
.....................................................
.............
WiFi connected
Starting web server on port: '80'
Starting stream server on port: '81'
Camera Ready! Use 'http://192.168.1.139' to connect
```

Рис. 7. Ссылка на сайт

Убедившись в работоспособности камеры, необходимо скачать библиотеку:

https://github.com/yoursunny/esp32cam

Она является высокоуровневой обёрткой оригинальной библиотеки от Espressif. В ней также находятся примеры по работе с ней, которые можно изучить. В данном случае нас интересует пример «Wi-Fi Cam».

Для написания программы на данном этапе рекомендуется использовать одну из программ, которые были перечислены выше (в частности, будет использоваться (Visual Studio Code). Пример программы в Visual Studio Code показан на рис. 8.

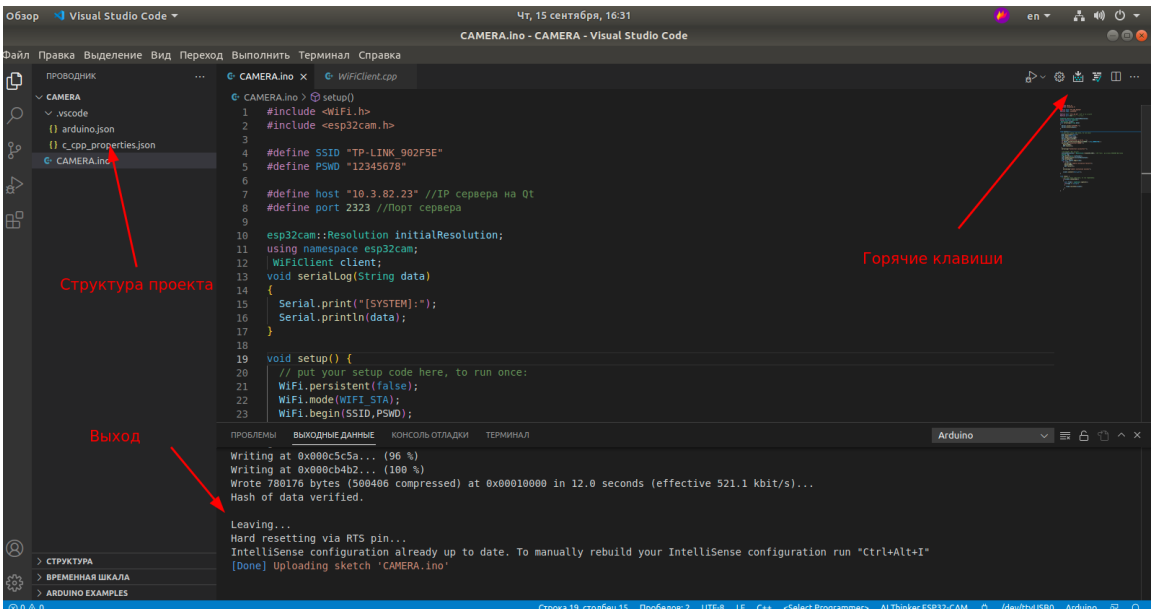

Рис. 8. Код программы для ESP32 в Visual Studio Code

Листинг программы для ESP32 приведён ниже:

```
#include <WiFi.h>
#include <esp32cam.h>

#define SSID "TP-LINK_902F5E"
#define PSWD "12345678"

#define host "10.3.82.23" //IP сервера на Qt
#define port 2323 //Порт сервера

esp32cam::Resolution initialResolution;
using namespace esp32cam;
 WiFiClient client;
void serialLog(String data)
{
  Serial.print("[SYSTEM]:");
  Serial.println(data);
}

void setup() {
  // put your setup code here, to run once:
  WiFi.persistent(false);
  WiFi.mode(WIFI_STA);
  WiFi.begin(SSID,PSWD);
  Serial.begin(115200);
  serialLog("Connecting to Wi-Fi...");
  if (WiFi.waitForConnectResult(20000) != WL_CONNECTED) {
```

```
      serialLog("Connection ERROR!");
       delay(5000);
       ESP.restart();
      }
      serialLog("Connection successful!");

      //Сконфигурируем камеру:
      initialResolution = Resolution::find(640,480); //Выставим разрешение 640x480 пикселей
      Config cfg;
      cfg.setPins(pins::AiThinker);
      cfg.setResolution(initialResolution);
      cfg.setJpeg(80);
      bool ok = Camera.begin(cfg);
       if (!ok) {
        serialLog("camera initialize failure");
        delay(5000);
        ESP.restart();
       }
       serialLog("camera initialize success");

       client.connect(host,port);
    }

    void loop() {
     // put your main code here, to run repeatedly:
       if(client.connected())
       {
        auto frame = esp32cam::capture();
        if(frame != nullptr)
        {
         frame->writeTo(client);
        }
       }
    }
```

Далее необходимо написать серверное ПО на Qt. Для этого необходимо вспомнить про QTcpServer и QTcpSocket. Результат программы показан на рис. 9.

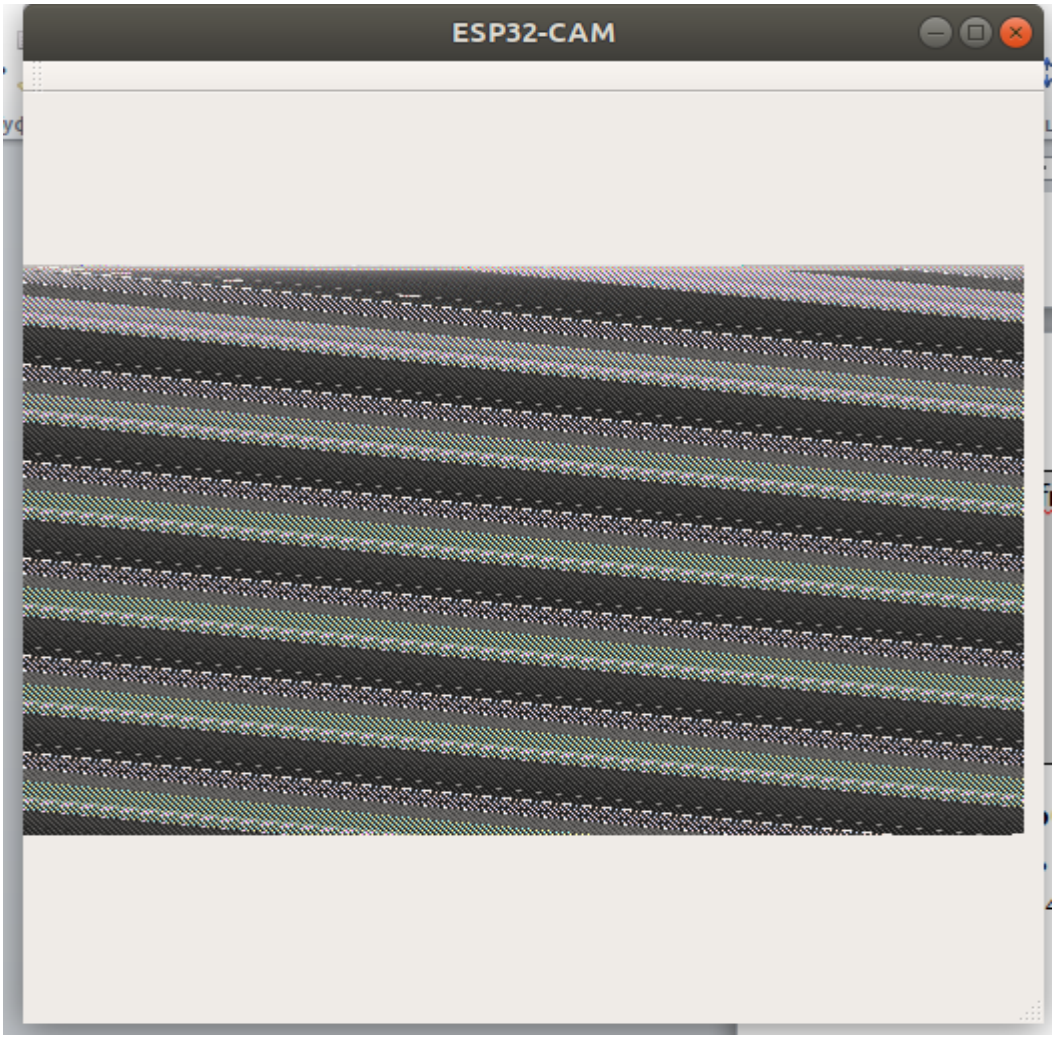

Рис. 9. Результат работы программы

Так как камера повреждена, то вместо картинки имеем произвольные артефакты. Листинг программы приведён ниже:

MainWindow.h:

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QTcpServer>
#include <QTcpSocket>
#include <QNetworkInterface>

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
```

```
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();
 public slots:
    void getImage();
    void newCamera();
private:
    Ui::MainWindow *ui;
    QTcpServer* server;
    QTcpSocket* cameraSocket;
    QPixmap map;
    QString data;
    QImage *image;
};

#endif // MAINWINDOW_H
```

MainWindow.cpp:

```
#include "mainwindow.h"
#include "ui_mainwindow.h"

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    server = new QTcpServer(this);
    QString ipAddress;
        QList<QHostAddress> ipAddressesList = QNetworkInter-
face::allAddresses();
        // use the first non-localhost IPv4 address
        for (int i = 0; i < ipAddressesList.size(); ++i) {
            if (ipAddressesList.at(i) != QHostAddress::LocalHost &&
                ipAddressesList.at(i).toIPv4Address()) {
                ipAddress = ipAddressesList.at(i).toString();
                break;
            }
        }
        // if we did not find one, use IPv4 localhost
        if (ipAddress.isEmpty())
            ipAddress = QHostAddress(QHostAddress::LocalHost).toString();
```

```
        qDebug()<<ipAddress;
        server->listen(QHostAddress(ipAddress),2323);
        con-
nect(server,&QTcpServer::newConnection,this,&MainWindow::newCamera);
        ui->setupUi(this);
    }

    MainWindow::~MainWindow()
    {
        delete ui;
    }
    void MainWindow::newCamera()
    {
        cameraSocket = new QTcpSocket(this);
        cameraSocket = server->nextPendingConnection();
        image = new QImage( 640, 480, QImage::Format_RGB16);
        ui->label->setGeometry(0,0,640,480);
       con-
nect(cameraSocket,&QAbstractSocket::disconnected,cameraSocket,&QAbstractS
ocket::deleteLater);
       con-
nect(cameraSocket,&QAbstractSocket::readyRead,this,&MainWindow::getImage
);
       qDebug()<<"Connected!";
    }
    void MainWindow::getImage()
    {
        QByteArray array = cameraSocket->readAll();

        image->loadFromData((const                    uchar*)array.data(),
QImage::Format_RGB16);

        QImage          image1((const          uchar*)array.data(),500,300,
QImage::Format_RGBA8888);
         ui->label->setPixmap(QPixmap::fromImage(image1));

    //    data += cameraSocket->readAll();
    //    if(map.loadFromData(data.toUtf8(),"BMP"))
    //    {
    //    ui->label->setPixmap(map);
    //    data.clear();
    //    }
    }
```

А в графическом интерфейсе необходимо создать Qlablel, в котором будет выводиться картинка (рис. 9). На случай, возникновения каких-либо трудностей, то в папке с лабораторной работой находится сама программа.

6. ПРОЕКТИРОВАНИЕ АППАРАТНОЙ ЧАСТИ

Помимо программирования в рамках данной дисциплины нужно освоить монтаж элементов на печатной плате. А прежде, чем приступать к монтажу, необходимо спроектировать плату. А для этого используются специальные САПР. В рамках ЭМС была рассмотрена САПР «Altium Designer», но она довольно сложна для понимания, поэтому начнём с чего-то попроще, а в конце каждый выберет программу по своему усмотрению.

Самой простой программой для проектирования печатных плат является «Sprint Layout». Программа очень проста, что обеспечило ей популярность среди радиолюбителей. Также она очень стара, что позволяет ей спокойно работать даже на самых слабых компьютерах и под ОС «Windows XP». Интерфейс программы показан на рис. 10.

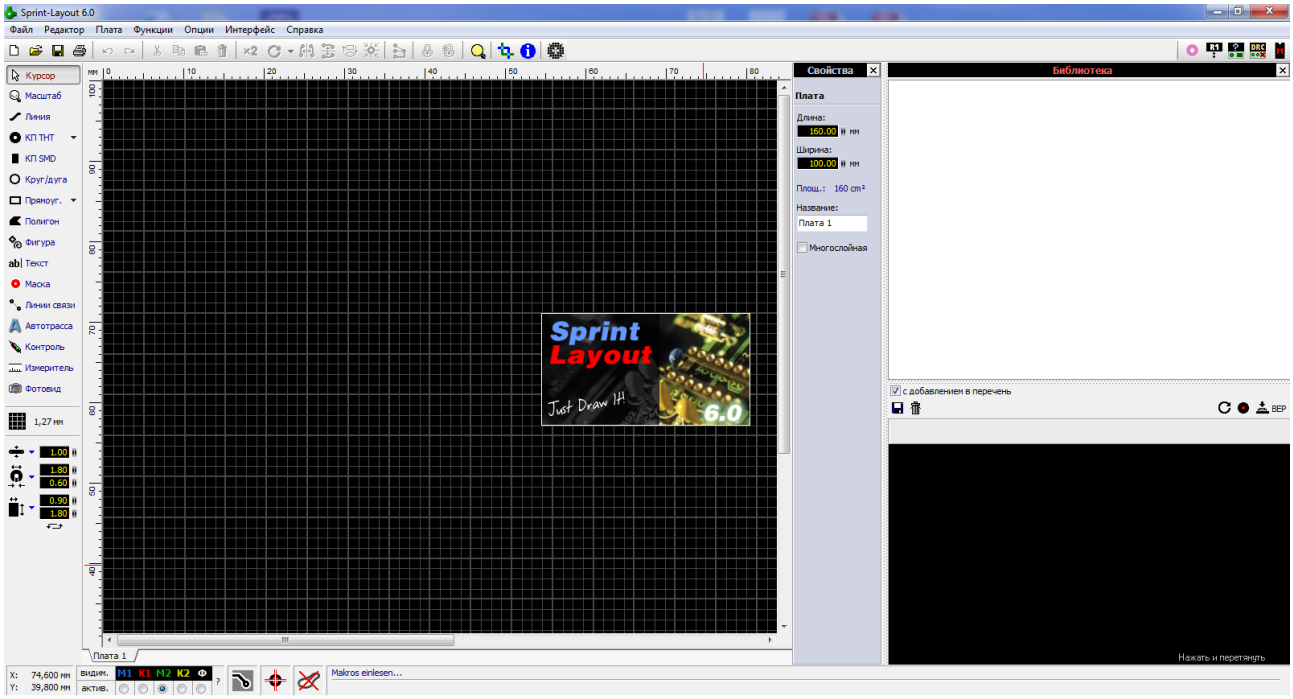

Рис. 10. Главный интерфейс программы

Главный интерфейс разбит на 3 столбца: (слева направо) панель инструментов, рабочее поле, панель свойств, библиотеки компонентов.

Внешний вид библиотеки компонентов показан на рис. 11.

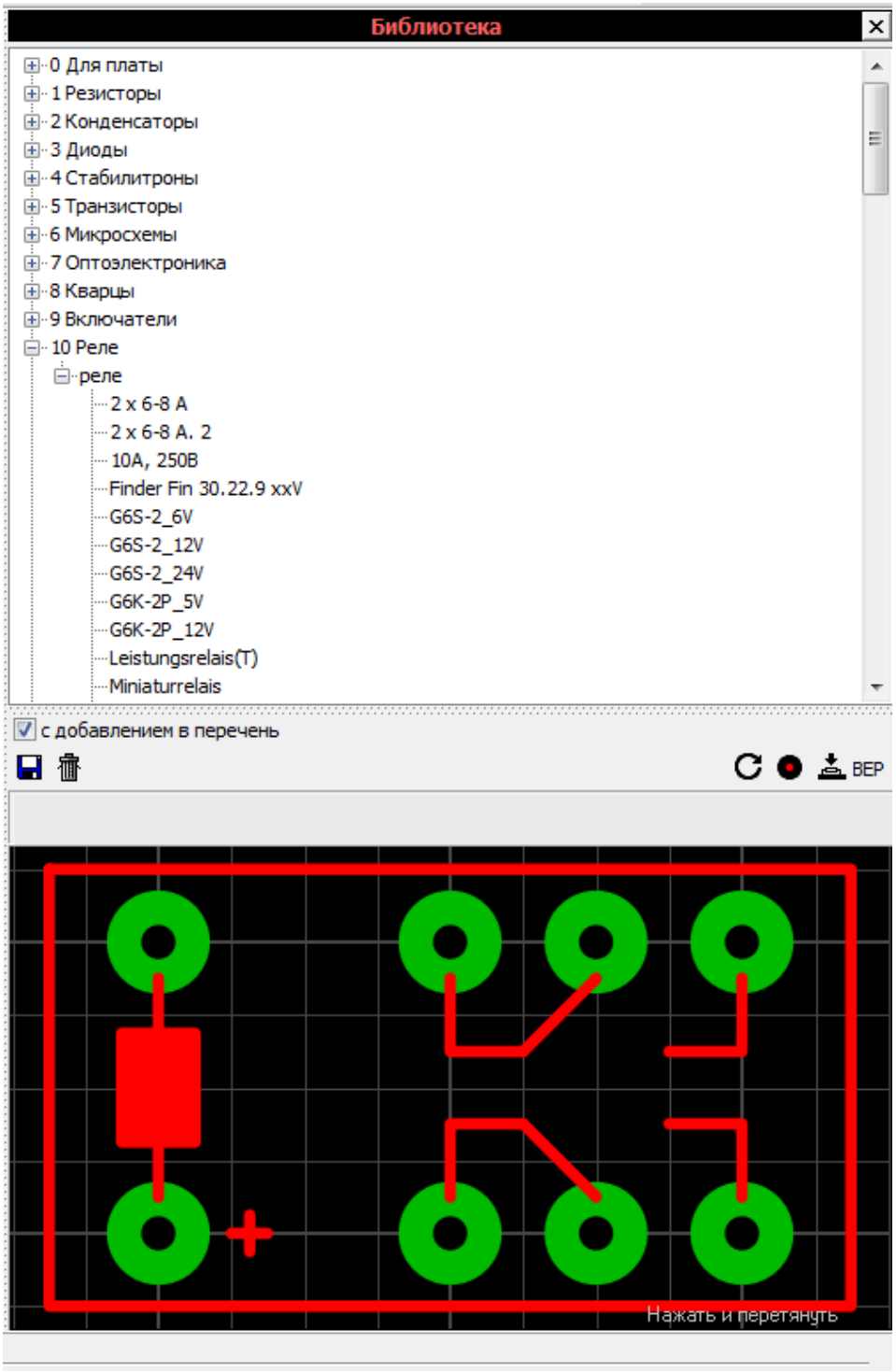

Рис. 11. Внешний вид библиотеки компонентов

Выбор компонента производится перетаскиванием его на рабочее поле (рис. 12).

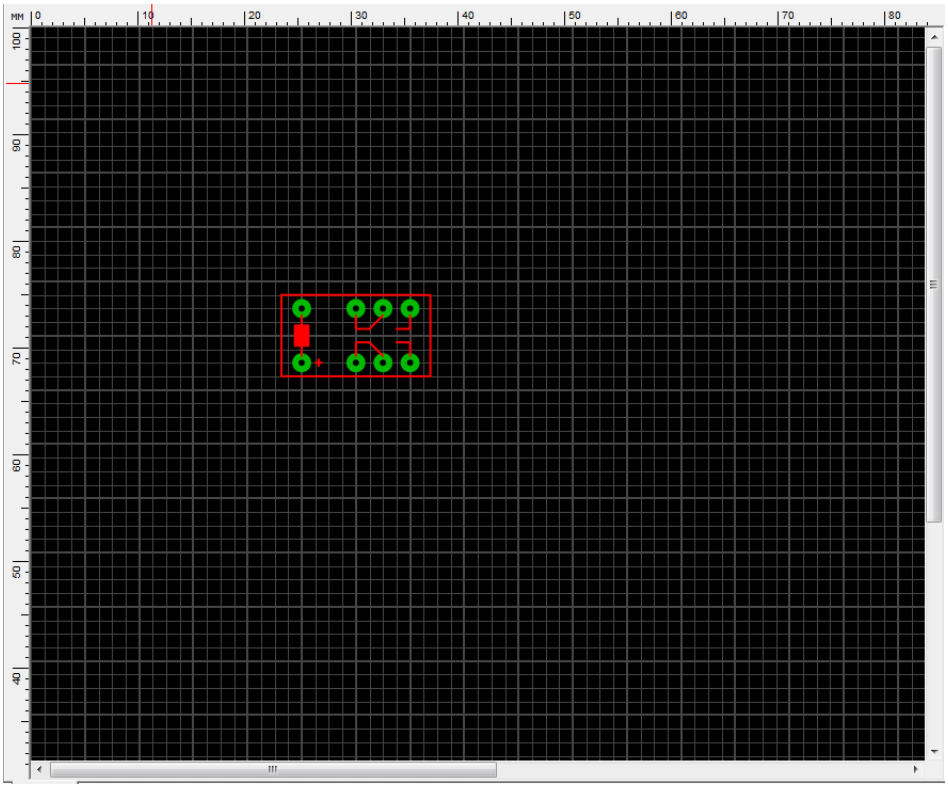

Рис. 12. Компонент на рабочем поле

С помощью панели инструментов можно выбрать инструмент «Линия» и соединить компоненты (рис. 13).

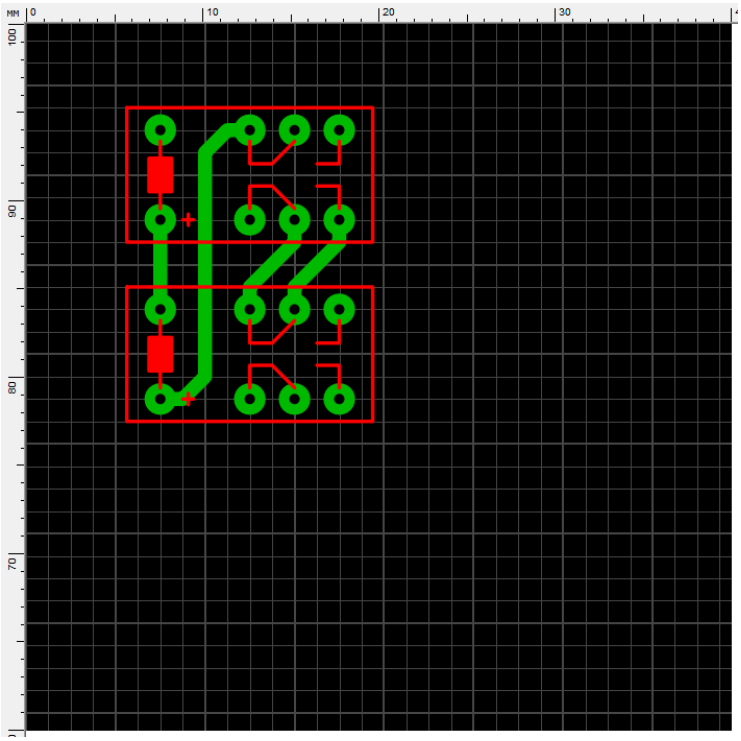

Рис. 13. Соединение компонентов дорожками

А теперь посмотрим подробнее на принцип проектирования платы: всего в программе имеется 4 слоя (рис. 14).

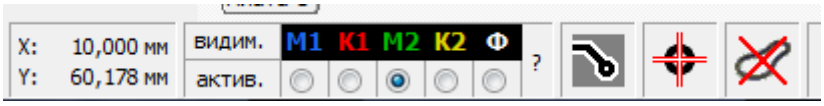

Рис. 14. Слои платы

M1 – Верхний слой;
K1 – Слой верхней шелкографии;
M2 – Нижний слой;
K2 – Слой нижней шелкографии;
Ф – Слой контура платы.

Контур платы необходим при формировании Gerber файлов перед производством на заводе, а также для создания вырезов в плате.

Полученную плату можно распечатать для дальнейшего производства методом ЛУТ, а также для сравнения расположения и размеров компонентов (рис. 15).

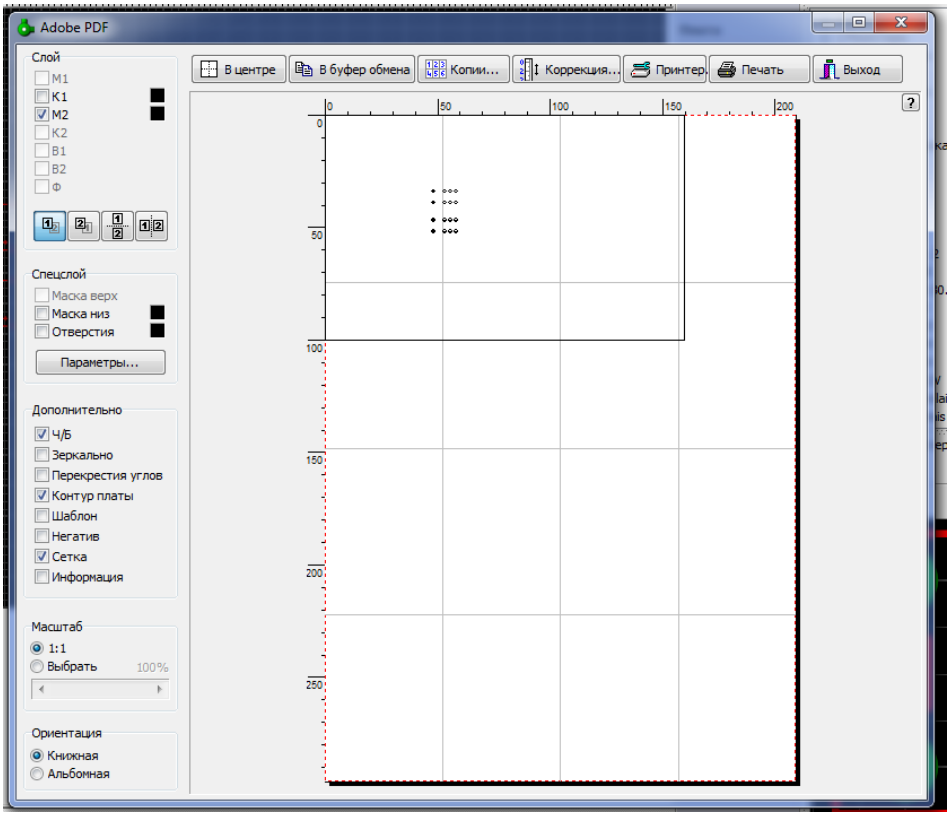

Рис. 15. Интерфейс печати платы

Теперь об основных правилах трассировки плат (очень кратко):

1. Дорожки должны быть как можно короче;
2. Дорожки должны проходить группой, а не расплетаться в паутину по всей плате;

3. Дорожки должны быть по возможности как можно шире (Исключением являются линии передачи);
4. Под антеннами радио модулей дорожки не должны проходить!

Всё остальное делается согласно стандартам ЭМС.

7. ПРОЕКТИРОВАНИЕ КОРПУСА УСТРОЙСТВА

Помимо печатной платы требуется обычно и корпус устройства, в случае использования готового корпуса, то здесь всё достаточно просто: берётся разработанная плата, готовый корпус и всё это собирается воедино. Но когда требуется создать либо полностью новый корпус, либо значительно доработать существующий, то приходится вначале всё смоделировать на компьютере. Для этого используются также специальные программы, к примеру, «КОМПАС 3D» (рис. 16).

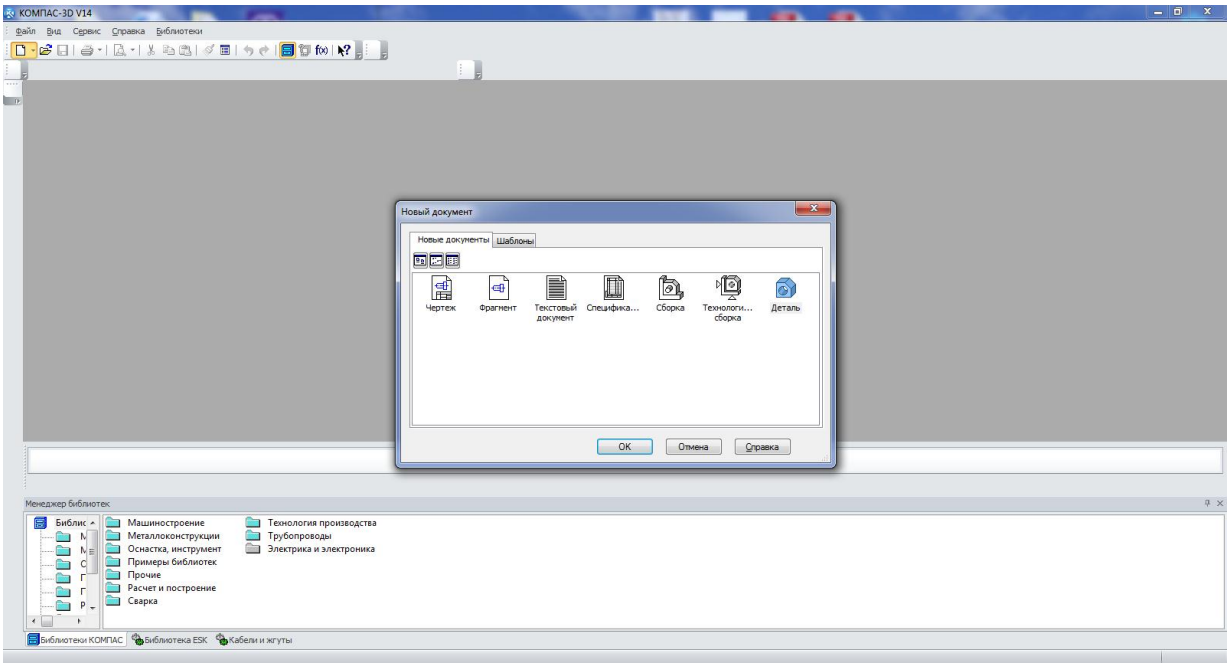

Рис. 16. Интерфейс программы «КОМПАС 3D»

Для работы с 3D моделями нужно выбрать режим «Деталь». Интерфейс работы с 3D моделями показан на рис. 17.

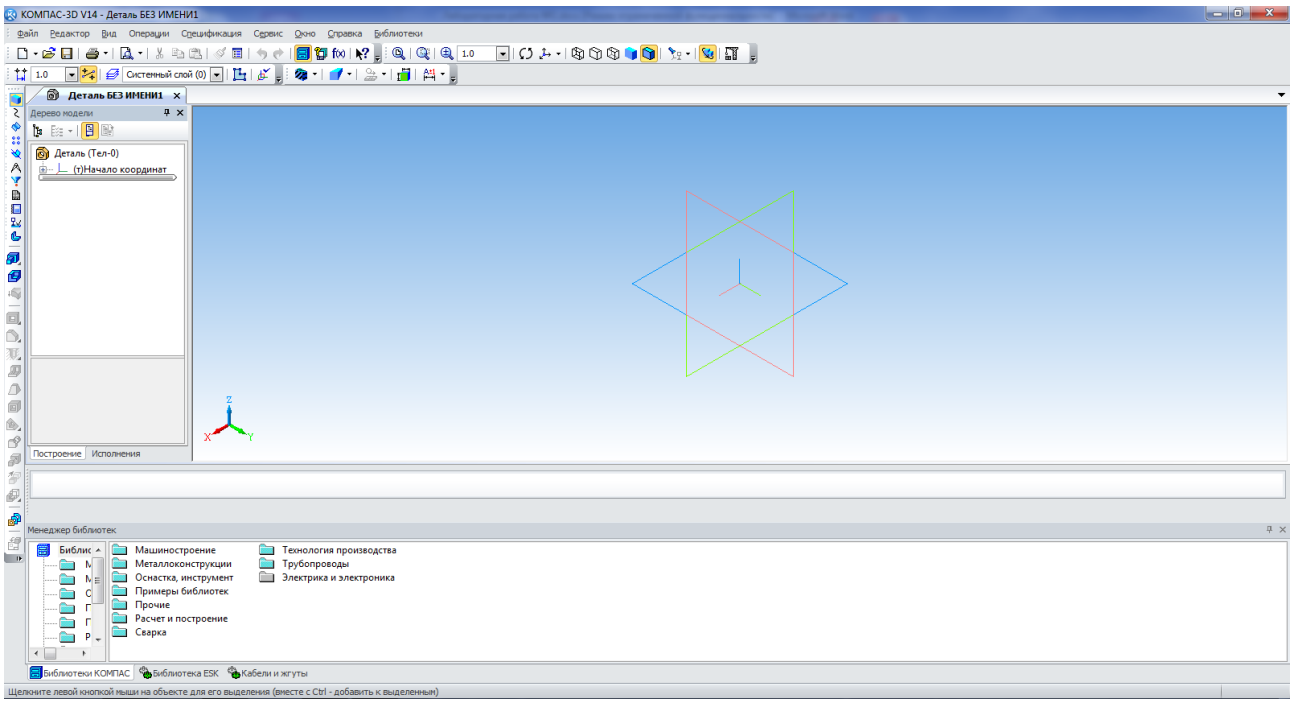

Рис. 17. Интерфейс работы с 3D моделями

Для создания 3D модели нужно выбрать одну из плоскостей и нажать кнопку «Эскиз» (рис. 18).

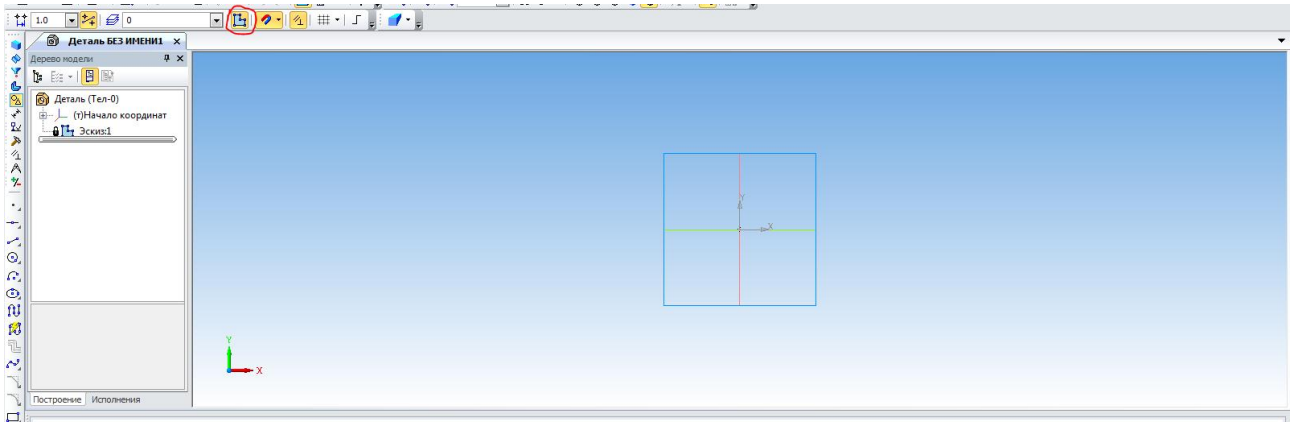

Рис. 18. Режим «Эскиз»

В данном режиме мы можем с помощью простых (или не очень) фигур построить 3D модель в одной из плоскостей. В качестве примера построим 3D модель трубы. Для этого в качестве эскиза построим 2 окружности (рис. 19).

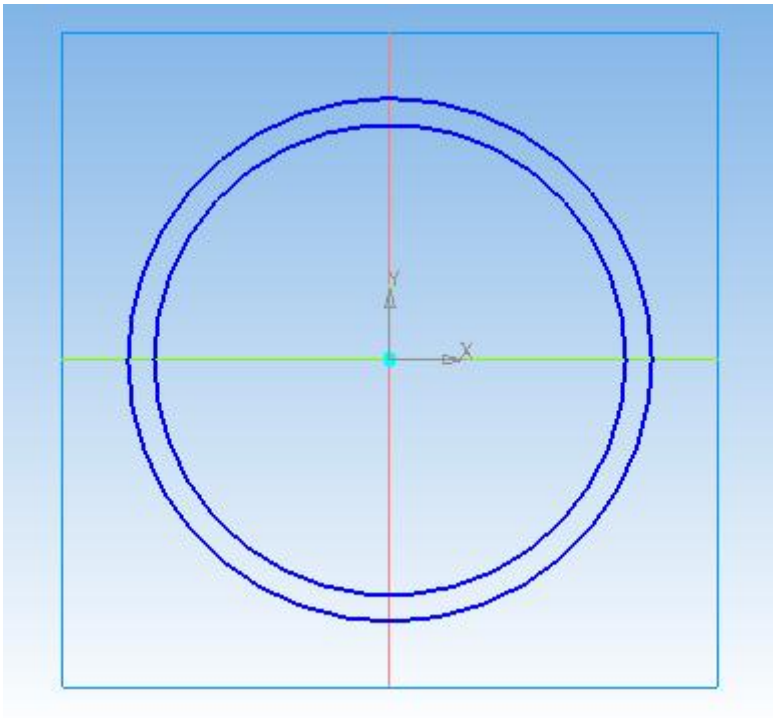

Рис. 19. Две окружности для построения 3D модели трубы

Дальше необходимо выйти из режима работы с эскизами. И «Выдавить» модель в пространство (рис. 20).

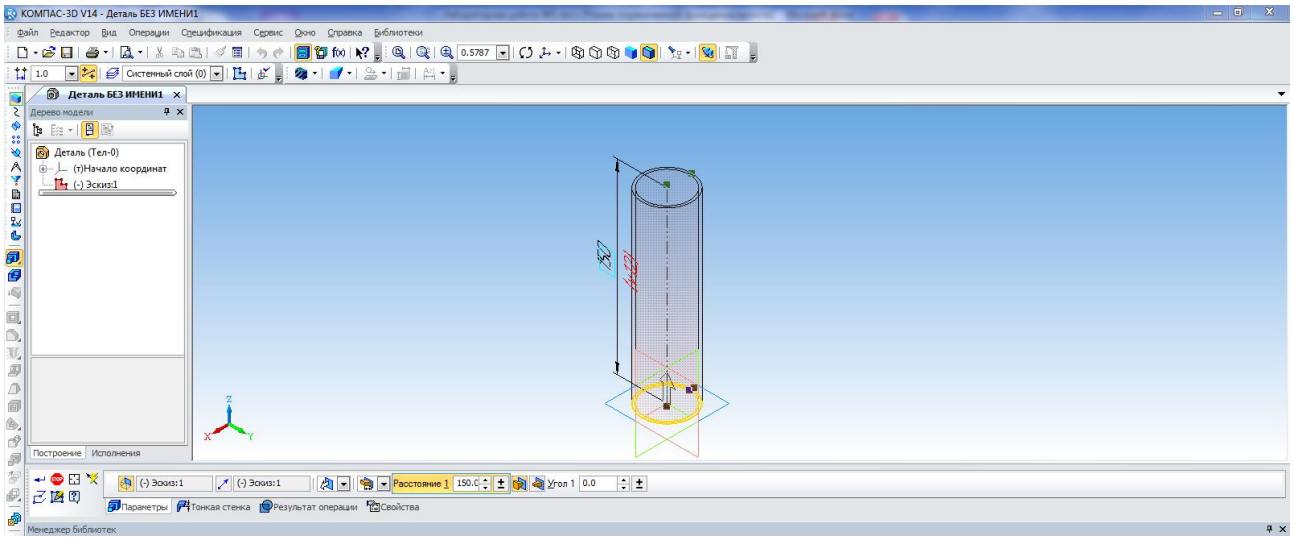

Рис. 20. Выдавливание эскиза в пространство

Итоговый результат показан на рисунке 21.

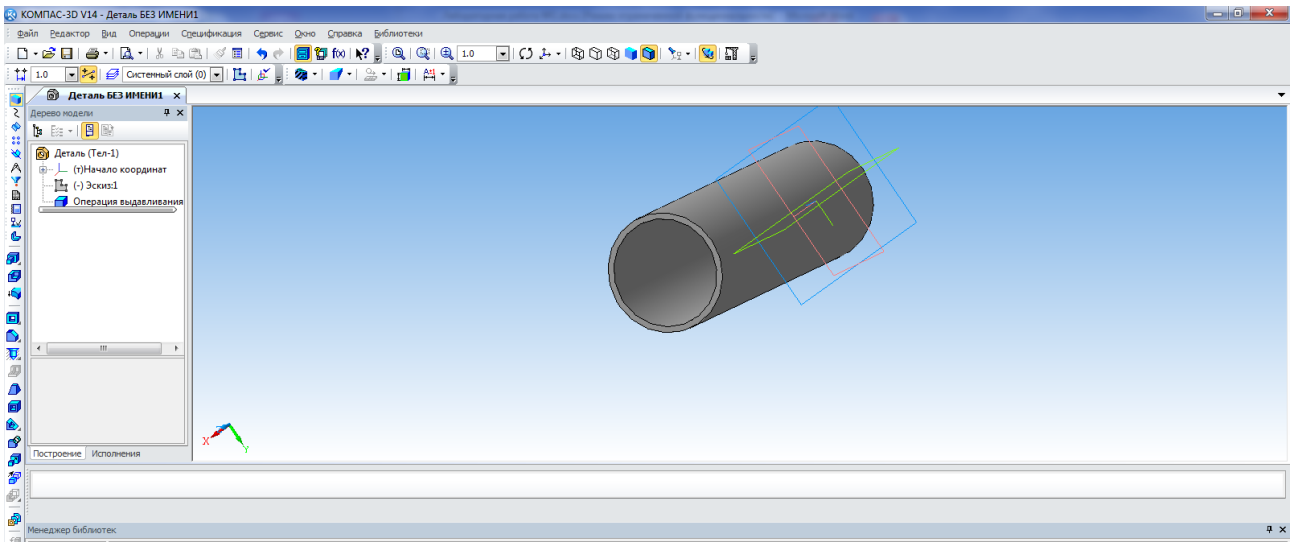

Рис. 21. Итоговая модель трубы

Теперь сделаем вырез в середине. Для этого нужно взять опять одну из плоскостей (создать эскиз на цилиндре нельзя) и создать параллельную ей плоскость (рис. 22).

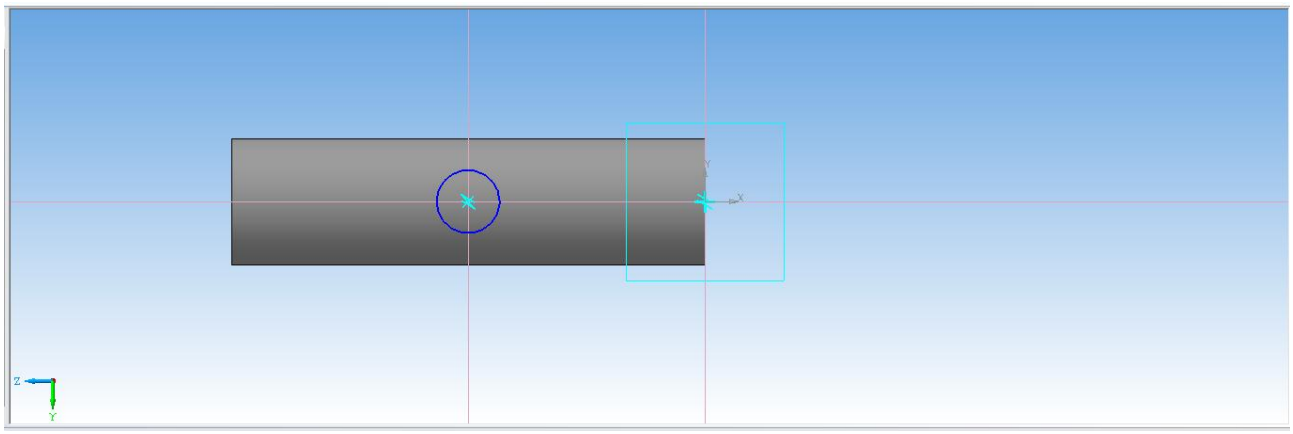

Рис. 22. Эскиз среза в трубе

Далее нужно выполнить операцию вырезания выдавливанием (рис. 23).

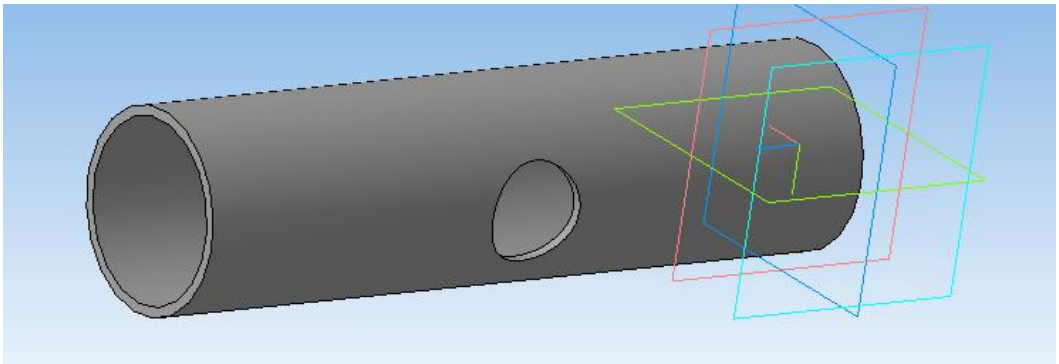

Рис. 23. Внешний вид трубы с вырезом

Последним этапом снимем фаску с одной из сторон трубы (рис. 24).

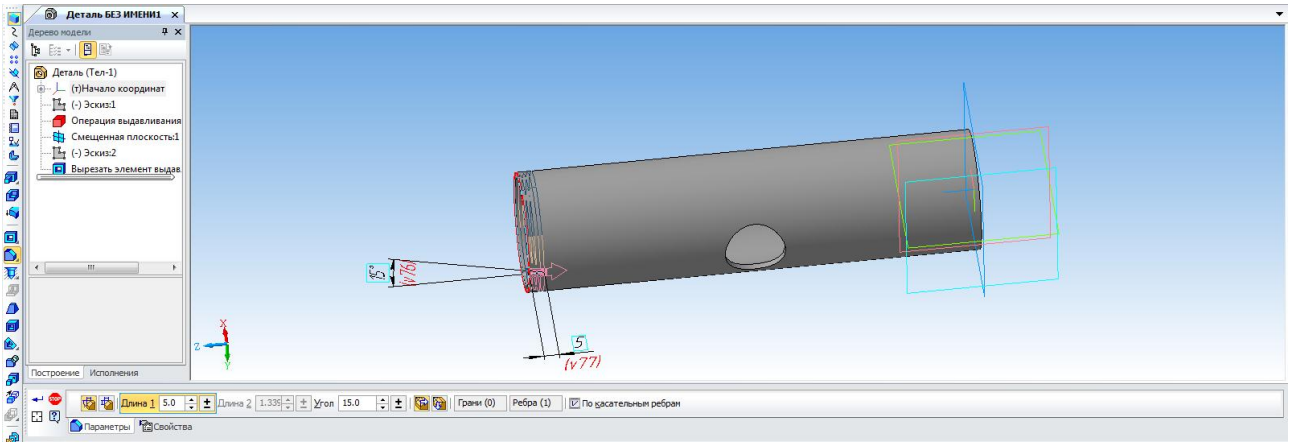

Рис. 24. Снятие фаски

Итоговый результат показан на рис. 25.

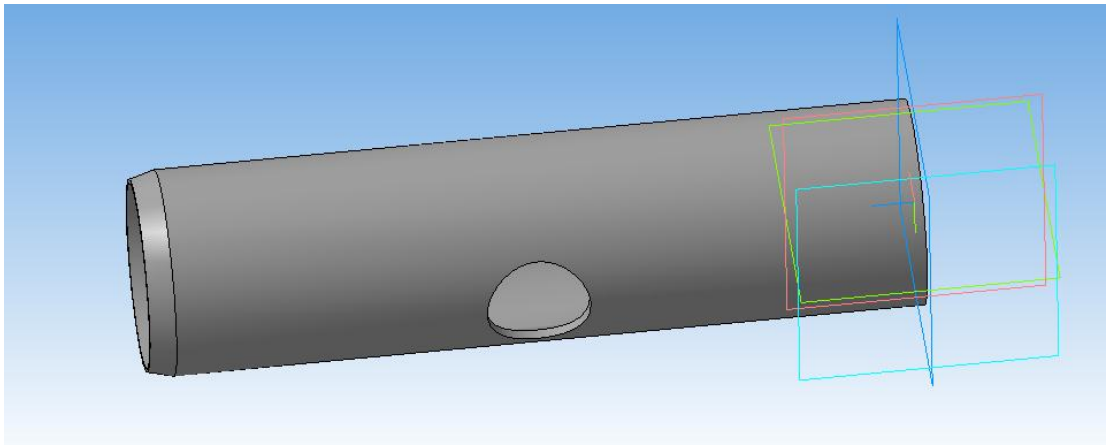

Рис. 25. Итоговый вариант трубы

Для 3D печати перед экспортом модели в формате .stl в КОМПАСЕ версии 14 нужно зайти во вкладку «Сервис» - «Параметры» - «Точность отрисовки МЦХ», где выставить все ползунки на максимум (рис. 26).

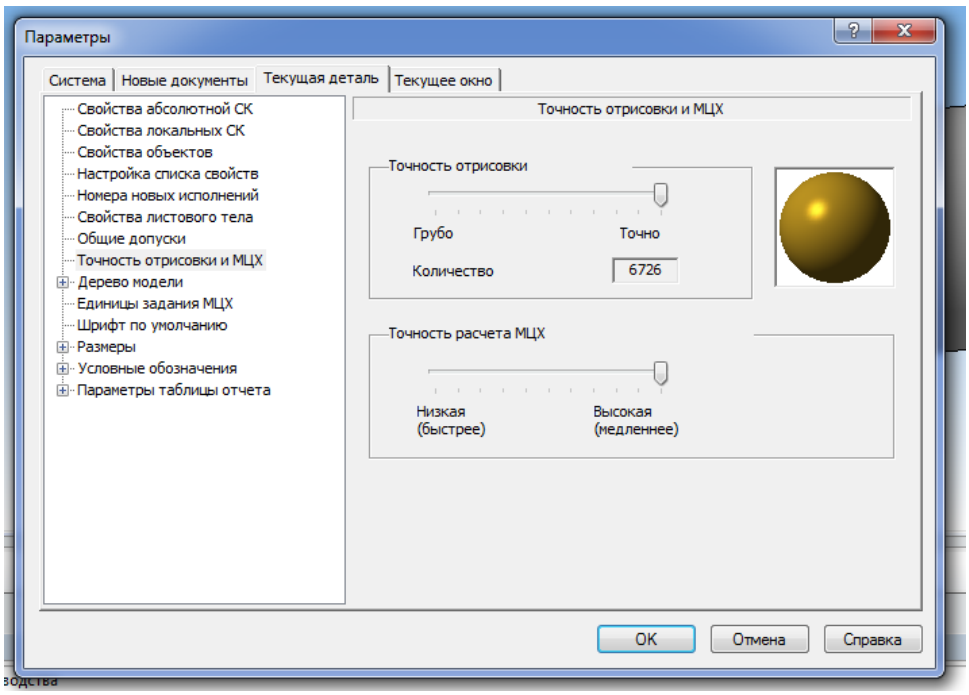

Рис. 26. Настройка точности отрисовки МЦХ

Далее полученную модель можно экспортировать в любую программу для 3D печати и «распечатать» эту модель физически.

Если же печать модели не предполагается, то можно экспортировать её в чертёж, по которому вырезать необходимые элементы в ручную. Для этого нужно создать отдельный документ «Чертёж» и в нём выполнить «Импорт модели» (рис. 27).

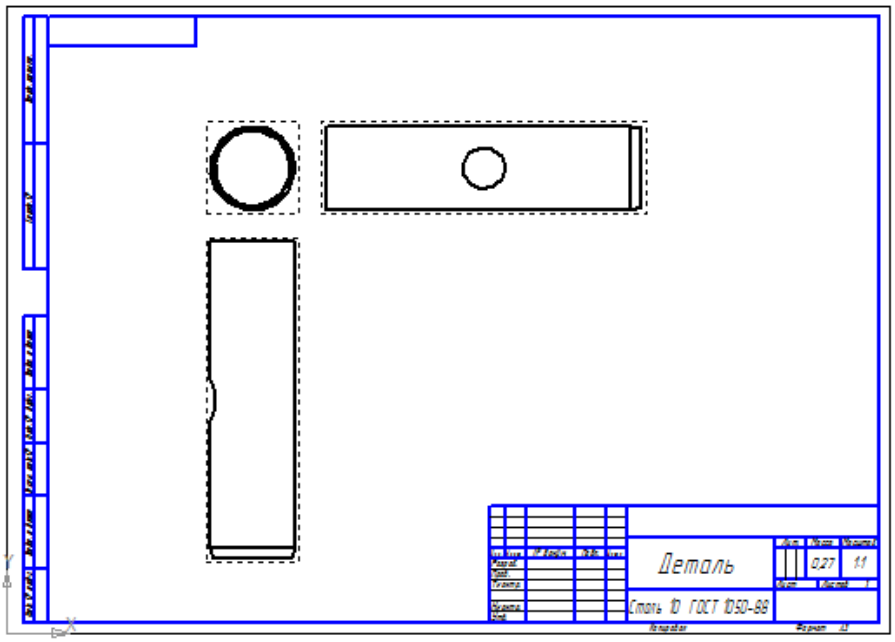

Рис. 27. Экспорт модели в чертёж

8. ИНДИВИДУАЛЬНЫЕ ЗАДАНИЯ ПО ВАРИАНТАМ

Задание №1 – Разработка печатной платы устройства (в любой программе) согласно варианту задания.

Таблица 2

Индивидуальные задания для выполнения

№ группы	РП-**1		РП-**2		РП-**3		РП-**4	
Подгруппа	I	II	I	II	I	II	I	II
№ Задания	1	2	1	2	1	2	1	2

Схемы устройств представлены на рис. 28 и 29.

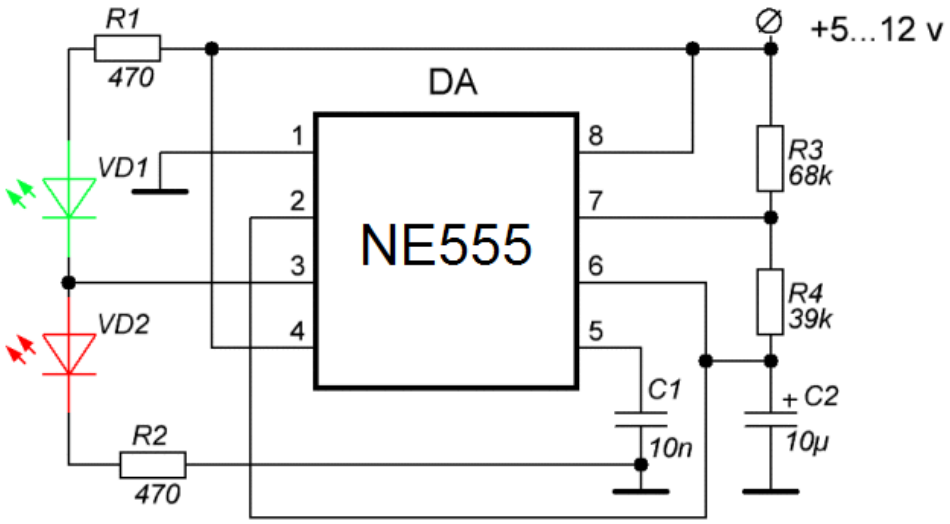

Рис. 28. Схема устройства (вариант 1)

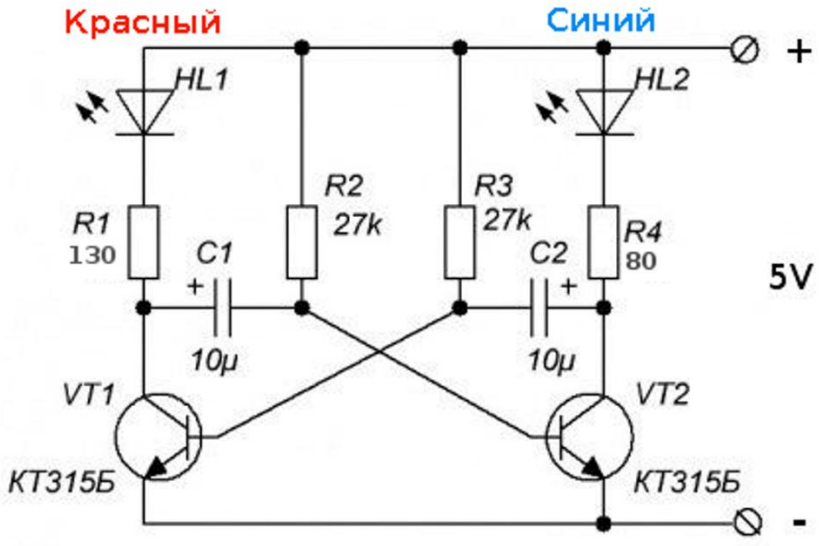

Рис. 29. Схема устройства (вариант 2)

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Бродин В.Б. Системы на микроконтроллерах и БИС программируемой логики / В.Б. Бродин, А.В. Калинин – М.: Издательство ЭКОМ, 2002. – 400 с.
2. Угрюмов Е.П. Цифровая схемотехника / Е.П. Угрюмов – 3-е изд. – СПб: БВХ-Петербург, 2010. – 816 с.
3. Партин А.С. Введение в цифровую схемотехнику / А.С. Партин, В.Г. Борисов – М.: Радио и связь, 1987. – 64 с.
4. Белов А.В. Конструирование устройств на микроконтроллерах / А.В. Белов – СПб: Наука и Техника, 2005. – 256 с.
5. Баранов В.Н. Применение микроконтроллеров AVR: схемы, алгоритмы, программы / В.Н. Баранов – 2-е изд. испр. – М.: Издательский дом «Додэка-XXI», 2006. – 288 с.
6. Голубцов М.С. Микроконтроллеры AVR: от простого к сложному / М.С. Голубцов – М.: СОЛОН-Пресс, 2003. – 288 с.

ОГЛАВЛЕНИЕ

ОСНОВЫ ПРОЕКТИРОВАНИЯ ПРОГРАММНО-АППАРАТНЫХ КОМПЛЕКСОВ И СИСТЕМ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к выполнению лабораторной работы №2
для студентов специальности 11.05.01
«Радиоэлектронные системы и комплексы»
очной формы обучения

Составитель
Сукачев Александр Игоревич

Издается в авторской редакции

Подписано к изданию 19.03.2024.
Уч.-изд. л. 1,5.

ФГБОУ ВО «Воронежский государственный технический университет»
394006 Воронеж, ул. 20-летия Октября, 84