Министерство образования и науки Российской Федерации

федеральное государственное автономное образовательное учреждение

высшего образования

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ

ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Инженерная школа неразрушающего контроля и безопасности

Отделение электронной инженерии

Направление подготовки «Биотехнические системы и технологии»

Курсовой проект

«Машинка на радиоуправлении»

по дисциплине «Основы микропроцессорной техники»

Выполнил:

студент гр. 1Д51 Ухов С. А.

Принял:

ассистент

отделения электронной инженерии Мусоров И.С.

Томск – 2018

Содержание

Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Отделение электронной инженерии

Зав. отделением _____ П.Ф. Баранов

(подпись, дата)

ЗАДАНИЕ

на выполнение курсового проекта

по дисциплине «Основы микропроцессорной техники»

Студенту гр. 1Д51

1 Тема курсового проекта Машинка на радиоуправлении

2 Срок сдачи студентом готовой работы 20.12.2018

3 Исходные данные к работе

- Напряжение питания – 9В.
- Радиомодуль NRF21L01.
- Предусмотреть кнопку сброса.
- При проектировании принципиальной схемы предусмотреть разъем для программирования микроконтроллера.
- Режим измерения – непрерывный.

4 Содержание пояснительной записки

Введение

Выбор и обоснование структурной схемы

Разработка принципиальной схемы

Алгоритм программы

Код программы

Экспериментальные результаты

Заключение

5 Дата выдачи задания на выполнение курсового проекта 27.09.2018

Руководитель

_____ И.С. Мусоров

Задание принял к исполнению

Введение

Дети с помощью игрушек познают мир, выражают свои чувства, увлекательно проводят досуг. Детские машинки, самолеты, танки, вертолеты на пульте управления дарят малышу незабываемые эмоции, развивают интеллектуальные, умственные и физические навыки.

С радиоуправляемыми моделями дети всех возрастов получают яркие эмоции, прививаются примитивные технические понятия, которые в будущем могут поспособствовать профориентации ребенка.

Что касается интеллектуального развития, то специалисты выделяют несколько основоположных факторов:

- расширение кругозора;
- развитие внимательности и догадливости;
- совершенствование мелкой моторики;
- развитие фантазии;
- тренировка логического мышления.

С помощью радиоуправляемых моделей можно заменить ребенку компьютер, создать достойные условия для проведения досуга на улице, а не в душном помещении. И главное, вдохновить детей на более близкое общение друг с другом.

1 Выбор и обоснование структурной схемы

Для реализации данного проекта был выбран радиомодуль NRF24L01. Модуль nRF24L01 — это цифровой приемник и передатчик, заключенный в одной маленькой микросхеме. Размер платы, на которой размещается микросхема составляет всего 15 х 29 мм. При помощи данного радиомодуля будет осуществлять передача от передатчика (пульта управления) к приемнику (машинке).

Описание модуля NRF24L01

Нельзя создать по-настоящему интересный проект, не дав возможность создания коммуникаций между различными элементами системы. Поэтому так важно выбрать правильную платформу для организации связи между модулями. NRF24l01 отлично подходит

NRF24l01 – это высокоинтегрированная микросхема с пониженным потреблением энергии (ULP) 2Мбит/с для диапазона 2,4 ГГц. При помощи модуля можно связать несколько устройств для передачи данных по радиоканалу. Можно объединить до семи приборов в одну общую радиосеть на частоте 2,4 ГГц, один из модулей будет выступать в роли ведущего, остальные – ведомые. Радиомодуль NRF24l01 стоит дешево, поэтому его можно встретить в самых разных проектах – от умного дома до различных самодельных роботов.

Характеристики nrf24l01

- Низкие затраты энергии;
- Наличие усовершенствованного ускорителя аппаратного протокола ShockBurst;
- Операционная система ISM;

- Скорость передачи данных 250 Кбит/с, 1 Мбит/с и 2 Мбит/с;
- Полная совместимость со всеми стандартными сериями nRF24L Nordic, а также сериями nRF24E и nRF240;
- Напряжение питания 3,3В;
- Рабочие температуры от -40С до 85С, температуры хранения от -40С до 125С;
- Дальность связи до 100 м.

Основой модуля служит nRF24L01+ производства компании Nordic Semiconductor. На микросхеме расположены все необходимые элементы и вилка разъема. По интерфейсу SPI можно произвести настройку протокола, установить выходную мощность и наладить каналы обмена данных.

Распиновка NRF24L01

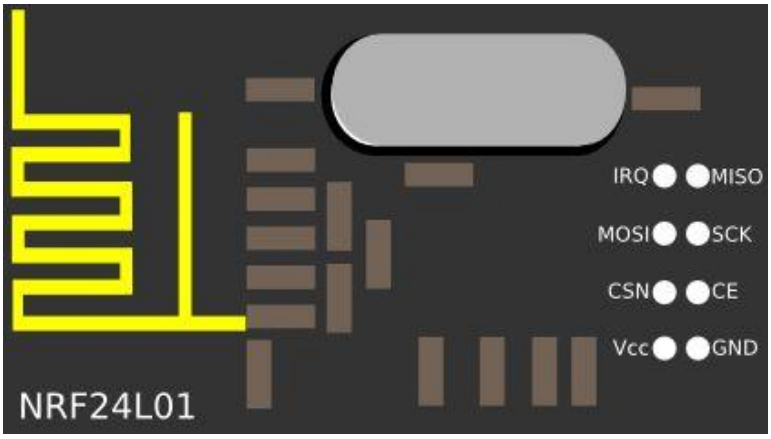


Рисунок 1 – nRF24L01+

Помимо выходов питания линии сигналов могут подключаться к контактам с питающим напряжением 5 В. Вход устройства, которое подключается к плате, должен потреблять ток не выше 10 мА.

Микросхема содержит следующие выходы:

- GND – земля;

- VCC – напряжение питания 3,3В
- CE – высокий уровень микросхемы;
- CSN – включение низкого уровня микросхемы. В этом случае устройство реагирует на SPI команды;
- SCK – такт SPI, максимальное значение 10 МГц;
- MOSI – передача информации от контроллера;
- MISO – прием данных в контроллер;
- IRQ – сигнал для аппаратного прерывания.

Организация питания nrf24l01

Во время запуска микроконтроллера могут возникнуть проблемы, которые связаны с тем, что не предусмотрена нужная сила тока в модуле питания 3,3 В. Из-за этого могут возникнуть помехи, мешающие стабильной работе. Обычно подобные трудности появляются, когда используются платы Arduino Uno, Nano, Mega, то есть в тех, в которых не хватает мощности. Для приведенных видов плат на пины подается небольшой ток 50 мА.

Существует несколько методов решения этой проблемы:

- Подключение конденсатора к микросхеме на 3, 3 В(+) и землю GND (-). Емкость лучше выбирать 10 мкФ и более.
- Дополнительный источник напряжения на 3,3 В.
- Разработка отдельной платы, установка на нее модуля nRF24L01 и добавление конденсаторов на 1 и 10 мкФ.
- Применение YourDuinoRobo1, который обладает дополнительным регулятором на 3,3 В.

Для согласованной работы радиомодуля NRF24L01 и машинки на радиоуправлении используются микроконтроллеры STM8S. Данные микроконтроллеры будут как в приёмнике, так и в передатчике.

Описание отладочной платы TM8SVLDISCOVERY

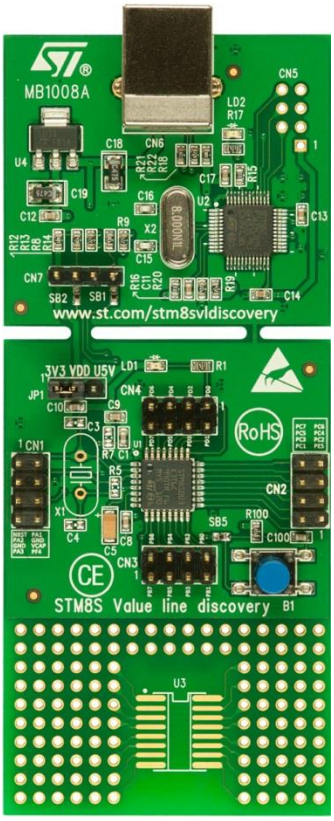


Рисунок 3 - Отладочная плата STM8SVLDISCOVERY.

STM8SVLDISCOVERY — недорогой отладочный комплект для начала работы с новой линейкой микроконтроллеров STM8S «Value Line» от STMicroelectronics. Отличительная особенность комплекта — полноценный инструментарий, состоящий из двух частей — целевого микроконтроллера **STM8S003K3T6** с необходимой для работы обвязкой и встроенный программатор-отладчик ST-Link.

Для начала работы разработчику нужно только подключить STM8SVLDISCOVERY к USB-порту персонального компьютера. В дальнейшем, комплект может использоваться в качестве программатора-отладчика для собственных разработок.

Ключевые особенности STM8SVLDISCOVERY:

- Встроенный ST-Link, интерфейс SWIM
- Микроконтроллер STM8S003K3T6, 8 Кб Flash, 1 Кб SRAM, 128 байт EEPROM
- Выбор источника питания — 5 В или 3.3 В
- Пользовательская кнопка B1
- Два светодиода LD1 и LD2
- Выводы микроконтроллера выведены на разъемы
- Контактная область для запайки компонентов

Для работы управления и привода используем микросхема L293D имеет DIP корпус с 16-ю выводами.

Описание микросхемы L293D.

Схема выводов ниже.

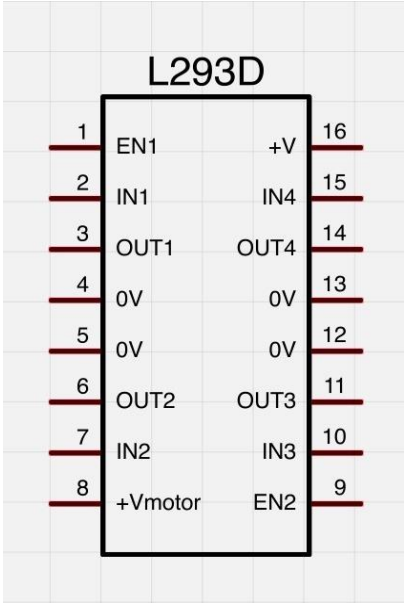


Рисунок 4 - Микросхема L293D имеет DIP корпус с 16-ю выводами

Помним, что отсчет выводов ведется против часовой стрелки и начинается от выемки в корпусе микросхемы.

- +V — питание микросхема, 5В;
- +Vmotor — питание двигателей, до 36В;
- 0V — земля;
- En1, En2 — выводы включения/выключения H-мостов;
- In1, In2 — управляющие выводы первого H-моста;
- Out1, Out2 — выводы для подключения первого двигателя;
- In3, In4 — управляющие выводы второго H-моста;
- Out3, Out4 — выводы для подключения второго двигателя.

Выводы En1 и En2 служат для отключения или включения мостов. Если мы подаем 0 на En, соответствующий мост полностью выключается и двигатель перестает вращаться. Эти сигналы пригодятся нам для управления тягой двигателя при помощи ШИМ сигнала.

В качестве кнопок используем VE059

Описание VE059

Тип: PBS-33B

Размер отверстия: 12 мм

Максимальное напряжение: 12 В

Номинальный тепловой ток: 1 (А).

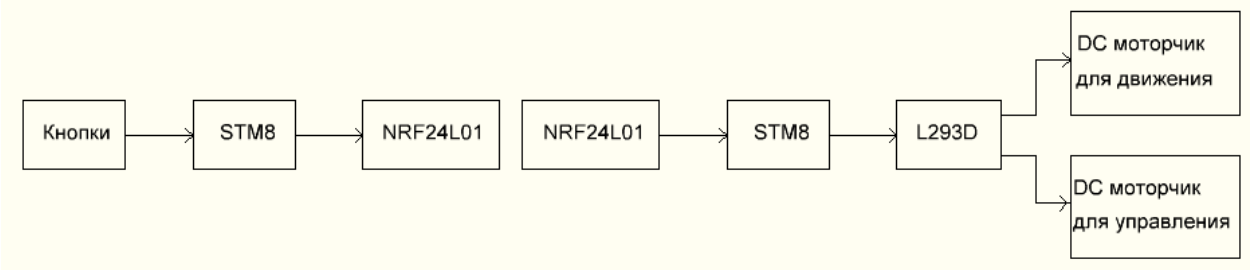


Рисунок 5 - Структурная схема радиоуправляемой машинки.

Расчет принципиальной схемы

Расчет резисторов:

1) Резистор R1 – резистор на входе сброса микрокооллера, его рекомендуемое сопротивление 55 кОм.

Примем из ряда E24: R1 = 56 кОм ± 5%.

Мощность рассеивания:

$$Pr1 = \frac{U\text{пит}^2}{R1} = \frac{5^2}{56000} = 0{,}00044 \text{ Вт}$$

Выберем R1 из справочника: CF-25 (С1-4) 0,25 Вт, 56 кОм, 5%

Расчет конденсаторов:

3) Резисторы R2-R5 – постоянные токоограничивающие резисторы, их рекомендуемое сопротивление 2 кОм.

Мощность рассеивания:

$$Pr = \frac{U\text{пит}^2}{R} = \frac{5^2}{2000} = 0{,}0125 \text{ Вт}$$

Выберем R2-R5 из справочника: CF-25 (C1-4) 0,25 Вт, 2 кОм, 5%.

Расчет конденсаторов:

1) Конденсатор C1 – конденсатор на входе сброса микроконтроллера, его рекомендуемая емкость 0,1 мкФ.

Выберем C1 из справочника: К73-17, 0,1 мкФ, 63 В, 5%.

2) Конденсаторы C2, C3 – конденсаторы в схеме включения стабилизатора.

Конденсатор C2 на входе необходим для ликвидации высокочастотных помех при подаче входного напряжения, его рекомендуемая емкость 0,33 мкФ. Конденсатор C3 на выходе стабилизатора обеспечивает стабильность блока питания при резком изменении тока нагрузки, а также уменьшает степень пульсаций, его рекомендуемая емкость 0,1 мкФ.

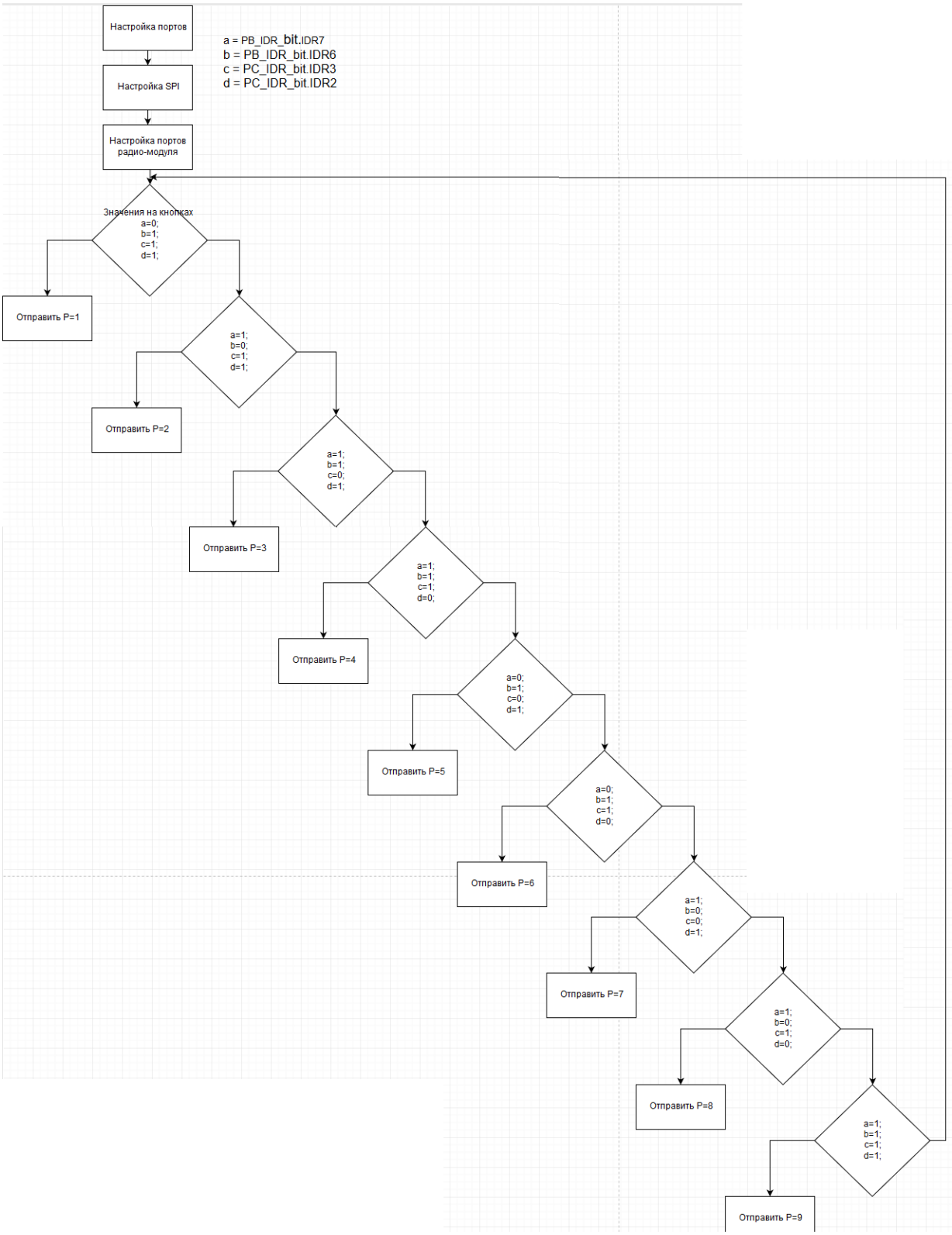
Выберем C2 и C3 из справочника: К73-17, 0.33 мкФ, 63 В, 5% и К73-17, 0.1 мкФ, 63 В, 5% соответственно.

3) C4 – конденсатор на входе питания микроконтроллера, его рекомендуемая емкость 0,1-2,2 мкФ.

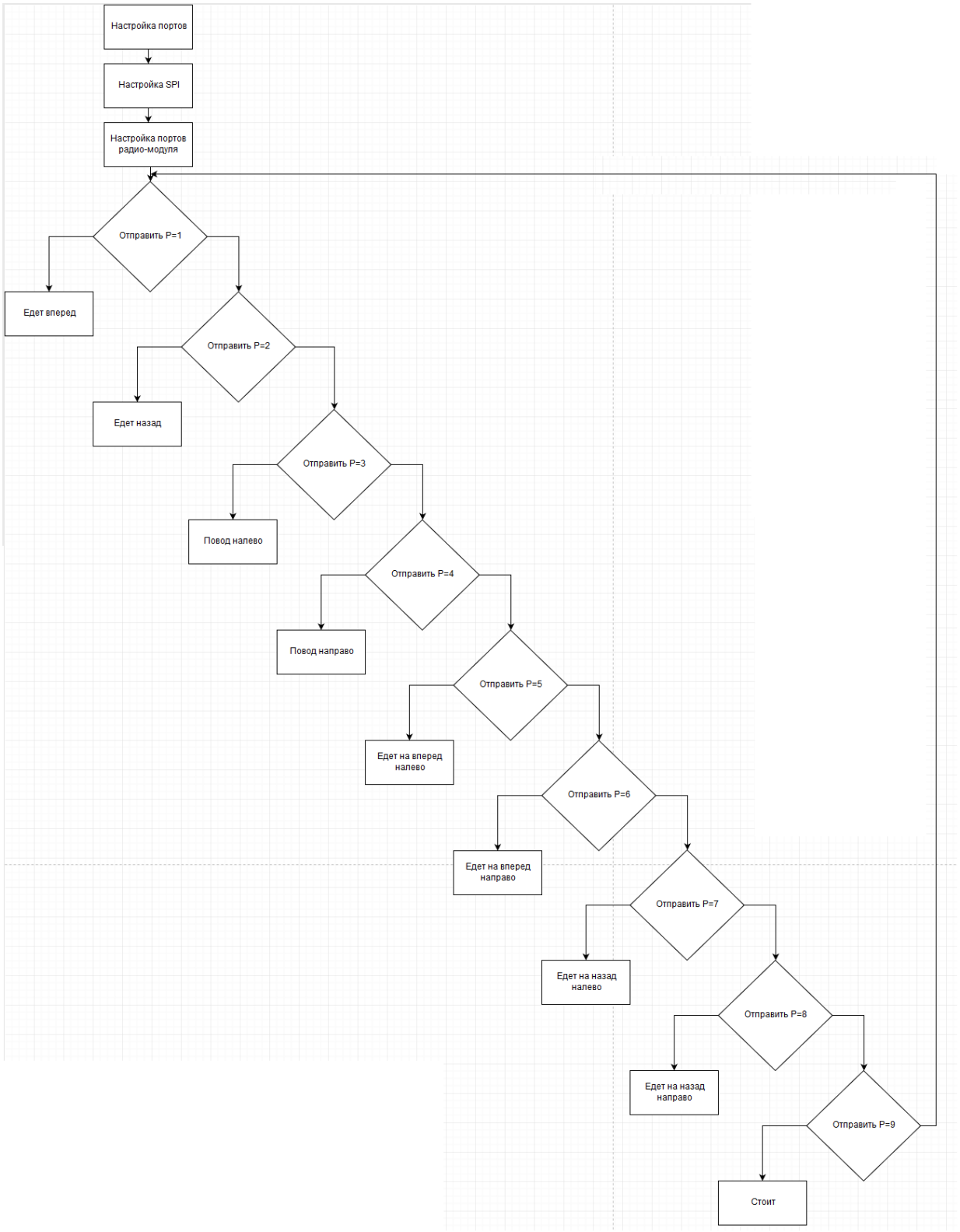
Выберем C4 из справочника: К73-17, 0.33 мкФ, 63 В, 5%.

Алгоритм программы.

Алгоритм передатчика:



Алгоритм приемника:



Код программы

Код передатчика:

```
#include "iostm8s003k3.h" // передатчик
#include "nRF24L01.h"

void interruptSend_init(void);  //объявление подпрограммы настройки //прерываний

#pragma vector=0x06
__interrupt void EXTI_PC7(void); //имя вектора внешнего прерывания порта D
int i, j,k,ret, a, b=5, c1, c2, c3, c4, c5, d1, d2, d3, d4, d5, P,K;
int spi_send (int data);
int main( void )                //основная программа
{
   // настройка портов
   PC_DDR_bit.DDR7 = 0;     //вход
  PC_CR1_bit.C17 = 0;  //дифференциальный вход
  PC_CR2_bit.C27 = 0;  //запретить внешние прерывани¤

  PC_DDR_bit.DDR6 = 1;       //выход
  PC_CR1_bit.C16 = 1;  //дифференциальный выход
  PC_CR2_bit.C26 = 1;  //запретить внешние прерывани¤
```

```
PC_DDR_bit.DDR5 = 1;       //выход

PC_CR1_bit.C15 = 1;  //дифференциальный выход

PC_CR2_bit.C25 = 1;  //запретить внешние прерывани¤


PC_DDR_bit.DDR4 = 1;       //выход CE // разрешает начало команды

PC_CR1_bit.C14 = 1;  //дифференциальный выход

PC_CR2_bit.C24 = 1;  //запретить внешние прерывани¤

PC_ODR_bit.ODR4 = 0; //CE


PE_DDR_bit.DDR5 = 1;       //режим ввода(0) или вывода(1)

PE_CR1_bit.C15 = 1;  // DDR=0: 0 - плавающий; 1 - с нат¤гиванием; DDR=1:
0 - открытый стой; 1 - двунаправленное управление;

PE_CR2_bit.C25 = 1;  // DDR=0: 0 - прер откл; 1 - прер вкл; DDR=1: 0 - 2ћvц;
1 - 10ћvц;

PE_ODR_bit.ODR5 = 1; //NSS 1


PD_DDR = 0xFF;       //выход

PD_CR1 = 0xFF;       //дифференциальный выход

PD_CR2 = 0xFF;       //запретить внешние прерывани¤


SPI_CR2 = 0x00;

SPI_ICR = 0x00;

SPI_SR = 0x00;
```

```
SPI_CRCPR = 0x07;
SPI_CR1_bit.BR = 2;
SPI_CR2_bit.SSM=1;
SPI_CR2_bit.SSI=1;
SPI_CR1_bit.MSTR=1;
for(i=0;i<90;i++);
SPI_CR1_bit.SPE=1;

interruptSend_init();

//настройка регистров радио-модуля

PE_ODR_bit.ODR5 = 0;
spi_send (nRF24L01_EN_AA_REG | 0x20); //W_r
spi_send (0x00);
PE_ODR_bit.ODR5 = 1;

PE_ODR_bit.ODR5 = 0;
spi_send (nRF24L01_EN_RXADDR_REG | 0x20); //W_r
spi_send (0x01);
PE_ODR_bit.ODR5 = 1;

PE_ODR_bit.ODR5 = 0;
spi_send (nRF24L01_SETUP_AW_REG | 0x20); //W_r
```

```
spi_send (0x03);
PE_ODR_bit.ODR5 = 1;

PE_ODR_bit.ODR5 = 0;
spi_send (nRF24L01_SETUP_RETR_REG | 0x20); //W_r
spi_send (0x00);
PE_ODR_bit.ODR5 = 1;

PE_ODR_bit.ODR5 = 0;
spi_send (nRF24L01_RF_CH_REG | 0x20); //W_r
spi_send (0x40);
PE_ODR_bit.ODR5 = 1;

PE_ODR_bit.ODR5 = 0;
spi_send (nRF24L01_RF_SETUP_REG | 0x20); //W_r
spi_send (0x0F);
PE_ODR_bit.ODR5 = 1;

PE_ODR_bit.ODR5 = 0;
spi_send (nRF24L01_RX_PW_P0_REG | 0x20); //W_r
spi_send (0x01);
PE_ODR_bit.ODR5 = 1;

PE_ODR_bit.ODR5 = 0;
```

```
  spi_send (nRF24L01_CONFIG_REG | 0x20); //W_r
  spi_send (0x03); // дл¤ передатчика 02, дл¤ приемника 03
  PE_ODR_bit.ODR5 = 1;
  for(j=0; j<800; j++); //>5мс

  //конец настройки регистров радио-модуля
  for(j=0; j<30000; j++); //>>5мс
  PC_ODR_bit.ODR4 = 1; //CE

  while (1)
  {
  }
}
int spi_send (int data)
{
  SPI_DR=data;
  while (SPI_SR_bit.TXE==0);//ждать, пока данные DR отправ¤тс¤
  while (SPI_SR_bit.RXNE==0);//ждать окончани¤ передачи
  ret=SPI_DR;
  return ret;
}

//подпрограмма настройки прерываний
void interruptSend_init(void)
```

```
{
  EXTI_CR1|=0x80;
  asm("rim");         //глобальное разрешение прерываний
  PB_DDR_bit.DDR4 = 0;        //вход
  PB_CR1_bit.C14 = 0;   //дифференциальный вход
  PB_CR2_bit.C24 = 1;   //разрешить внешние прерывани¤
}

//обработчик прерывания
__interrupt void EXTI_PC7(void)
{
   PE_ODR_bit.ODR5 = 0;
   spi_send (0x07); //R_r
   b=spi_send (0xFF);
   PE_ODR_bit.ODR5 = 1;

    PC_ODR_bit.ODR4 = 0; //CE

    PE_ODR_bit.ODR5 = 0;
    spi_send (0x61); //считать полученный пакет
    P = spi_send (0xff);
    PE_ODR_bit.ODR5 = 1;
    PC_ODR_bit.ODR4 = 1; //CE
```

```
 PE_ODR_bit.ODR5 = 0;
spi_send (0x07| 0x20); //обнулить флаг приема
spi_send (0x40);
PE_ODR_bit.ODR5 = 1;

switch (P)
{
case 1:
   K=0x20;
   PD_ODR=K;
   break;
case 2:
   K=0x10;
   PD_ODR=K;
   break;
case 3:
   K=0x04;
   PD_ODR=K;
   break;
case 4:
   K=0x08;
   PD_ODR=K;
   break;
case 5:
```

```
        K=0x24;
        PD_ODR=K;
        break;
    case 6:
        K=0x28;
        PD_ODR=K;
        break;
    case 7:
        K=0x18;
        PD_ODR=K;
        break;
    case 8:
        K=0x00;
        PD_ODR=K;
        break;
    case 15:
        K=0x00;
        PD_ODR=K;
        break;
     }
}
```

Код приемника:

```
#include "iostm8s003k3.h" // передатчик

#include "nRF24L01.h"

void interruptSend_init(void);  //объявление подпрограммы настройки //прерываний


#pragma vector=0x06

__interrupt void EXTI_PC7(void);

        //имя вектора внешнего прерывания //порта D


int i, j,k,ret, a, b=5,s,z1,z2,z3,z4,z5,y1,y2,y3,y4,y5;

int spi_send (int data);

int main( void )                    //основная программа

{

  // настройка портов

   PC_DDR_bit.DDR7 = 0;       //вход

  PC_CR1_bit.C17 = 0;  //дифференциальный вход

  PC_CR2_bit.C27 = 0;  //запретить внешние прерывания


  PC_DDR_bit.DDR6 = 1;       //выход

  PC_CR1_bit.C16 = 1;  //дифференциальный выход

  PC_CR2_bit.C26 = 1;  //запретить внешние прерывания


  PC_DDR_bit.DDR5 = 1;       //выход

  PC_CR1_bit.C15 = 1;  //дифференциальный выход
```

```
PC_CR2_bit.C25 = 1;  //запретить внешние прерывания


PC_DDR_bit.DDR4 = 1;       //выход CE // разрешает начало команды

PC_CR1_bit.C14 = 1;  //дифференциальный выход

PC_CR2_bit.C24 = 1;  //запретить внешние прерывания


PE_DDR_bit.DDR5 = 1;       //режим ввода(0) или вывода(1)

PE_CR1_bit.C15 = 1;  // DDR=0: 0 - плавающий; 1 - с натягиванием; DDR=1:
0 - открытый стой; 1 - двунаправленное управление;

PE_CR2_bit.C25 = 1;  // DDR=0: 0 - прер откл; 1 - прер вкл; DDR=1: 0 - 2ћvц;
1 - 10ћvц;

PE_ODR_bit.ODR5 = 1; //NSS 1


SPI_CR2 = 0x00;

SPI_ICR = 0x00;

SPI_SR = 0x00;

SPI_CRCPR = 0x07;

SPI_CR1_bit.BR = 2;

SPI_CR2_bit.SSM=1;

SPI_CR2_bit.SSI=1;

SPI_CR1_bit.MSTR=1;

for(i=0;i<90;i++);

SPI_CR1_bit.SPE=1;
```

```
interruptSend_init();
//настройка регистров радио-модуля
 PE_ODR_bit.ODR5 = 0;
 spi_send (nRF24L01_CONFIG_REG | 0x20); //W_r
 spi_send (0x02); // дл¤ передатчика 02, дл¤ приемника
 PE_ODR_bit.ODR5 = 1;
 for(j=0; j<800; j++); //>5мс

 PE_ODR_bit.ODR5 = 0;
 spi_send (nRF24L01_EN_AA_REG | 0x20); //W_r
 spi_send (0x00);
 PE_ODR_bit.ODR5 = 1;

 PE_ODR_bit.ODR5 = 0;
 spi_send (nRF24L01_EN_RXADDR_REG | 0x20); //W_r
 spi_send (0x01);
 PE_ODR_bit.ODR5 = 1;

 PE_ODR_bit.ODR5 = 0;
 spi_send (nRF24L01_SETUP_AW_REG | 0x20); //W_r
 spi_send (0x03);
 PE_ODR_bit.ODR5 = 1;
```

```
PE_ODR_bit.ODR5 = 0;
spi_send (nRF24L01_SETUP_RETR_REG | 0x20); //W_r
spi_send (0x00);
PE_ODR_bit.ODR5 = 1;

PE_ODR_bit.ODR5 = 0;
spi_send (nRF24L01_RF_CH_REG | 0x20); //W_r
spi_send (0x40);
PE_ODR_bit.ODR5 = 1;

PE_ODR_bit.ODR5 = 0;
spi_send (nRF24L01_RF_SETUP_REG | 0x20); //W_r
spi_send (0x0F);
PE_ODR_bit.ODR5 = 1;


PE_ODR_bit.ODR5 = 0;
spi_send (nRF24L01_RX_PW_P0_REG | 0x20); //W_r
spi_send (0x01);
PE_ODR_bit.ODR5 = 1;
//конец настройки регистров радио-модуля

PE_ODR_bit.ODR5 = 0; //считать флаг прерывания
spi_send (0x07); //R_r
```

```
a=spi_send (0xff); //такты, во врем¤ которых идет считывание из NRF

PE_ODR_bit.ODR5 = 1;


//проверка мифов

PE_ODR_bit.ODR5 = 0;

spi_send (nRF24L01_RX_ADDR_P0_REG); //W_r

z1=spi_send (0xff);

z2=spi_send (0xff);

z3=spi_send (0xff);

z4=spi_send (0xff);

z5=spi_send (0xff);

PE_ODR_bit.ODR5 = 1;


PE_ODR_bit.ODR5 = 0;

spi_send (nRF24L01_TX_ADDR_REG); //W_r

y1=spi_send (0xff);

y2=spi_send (0xff);

y3=spi_send (0xff);

y4=spi_send (0xff);

y5=spi_send (0xff);

PE_ODR_bit.ODR5 = 1;


while (1)

{
```

```
 /*
for(j=0; j<800; j++); //>5мс

 PE_ODR_bit.ODR5 = 0;
spi_send (0x20); //W_r

spi_send (0x18); //что записать
PE_ODR_bit.ODR5 = 1;

 PE_ODR_bit.ODR5 = 0;
spi_send (0x00); //R_r

j=spi_send (0xff); //такты, во врем¤ которых идет считывание из NRF
PE_ODR_bit.ODR5 = 1;*/
//Привод
 // Вперед
 if ((PB_IDR_bit.IDR7 == 0) && (PB_IDR_bit.IDR6 == 1) && (PC_IDR_bit.IDR3 == 1) && (PC_IDR_bit.IDR2 == 1))
 {
   PE_ODR_bit.ODR5 = 0;
   spi_send (nRF24L01_W_TX_PAYLOAD); //R_для отправки
   spi_send (0x01); // то, что мы отправляем приемнику!!!!!!!!!!!!!!!!!!!!!!!!!!!!
   PE_ODR_bit.ODR5 = 1;
   PC_ODR_bit.ODR4 = 1;
```

```
    for (j=0;j<700;j++);
   /*  PC_ODR_bit.ODR4 = 1;
    for(j=0; j<800; j++);*/
    PC_ODR_bit.ODR4 = 0;
   }
  // Назад
   else if ((PB_IDR_bit.IDR7 == 1) && (PB_IDR_bit.IDR6 == 0) && (PC_IDR_bit.IDR3 == 1) && (PC_IDR_bit.IDR2 == 1))
   {
    PE_ODR_bit.ODR5 = 0;
    spi_send (nRF24L01_W_TX_PAYLOAD); //R_для отправки
    spi_send (0x02); // то, что мы отправляем приемнику!!!!!!!!!!!!!!!!!!!!!!!!!!
    PE_ODR_bit.ODR5 = 1;
    PC_ODR_bit.ODR4 = 1;
    for(j=0; j<700; j++);
    PC_ODR_bit.ODR4 = 0;
   }
   //лево
   else if ((PB_IDR_bit.IDR7 == 1) && (PB_IDR_bit.IDR6 == 1) && (PC_IDR_bit.IDR3 == 0) && (PC_IDR_bit.IDR2 == 1))
   {
    PE_ODR_bit.ODR5 = 0;
    spi_send (nRF24L01_W_TX_PAYLOAD); //R_для отправки
```

```
    spi_send (0x03); // то, что мы отправляем приемнику!!!!!!!!!!!!!!!!!!!!!!!!!!!!
    PE_ODR_bit.ODR5 = 1;
    PC_ODR_bit.ODR4 = 1;
    for(j=0; j<800; j++);
    PC_ODR_bit.ODR4 = 0;
  }
  // право
  else if ((PB_IDR_bit.IDR7 == 1) && (PB_IDR_bit.IDR6 == 1) && (PC_IDR_bit.IDR3 == 1) && (PC_IDR_bit.IDR2 == 0))
  {
    PE_ODR_bit.ODR5 = 0;
    spi_send (nRF24L01_W_TX_PAYLOAD); //R_для отправки
    spi_send (0x04); // то, что мы отправляем приемнику!!!!!!!!!!!!!!!!!!!!!!!!!!!!
    PE_ODR_bit.ODR5 = 1;
    PC_ODR_bit.ODR4 = 1;
    for(j=0; j<800; j++);
    PC_ODR_bit.ODR4 = 0;
  }
  //вперед лево
  else if ((PB_IDR_bit.IDR7 == 0) && (PB_IDR_bit.IDR6 == 1) && (PC_IDR_bit.IDR3 == 0) && (PC_IDR_bit.IDR2 == 1))
  {
    PE_ODR_bit.ODR5 = 0;
```

```
    spi_send (nRF24L01_W_TX_PAYLOAD); //R_для отправки

    spi_send (0x05); // то, что мы отправляем приемнику!!!!!!!!!!!!!!!!!!!!!!!!!!!

    PE_ODR_bit.ODR5 = 1;

    PC_ODR_bit.ODR4 = 1;

    for(j=0; j<800; j++);

    PC_ODR_bit.ODR4 = 0;

  }

  // вперед право

  else if ((PB_IDR_bit.IDR7 == 0) && (PB_IDR_bit.IDR6 == 1) && (PC_IDR_bit.IDR3 == 1) && (PC_IDR_bit.IDR2 == 0))

  {

    PE_ODR_bit.ODR5 = 0;

    spi_send (nRF24L01_W_TX_PAYLOAD); //R_для отправки

    spi_send (0x06); // то, что мы отправляем приемнику!!!!!!!!!!!!!!!!!!!!!!!!!!!!

    PE_ODR_bit.ODR5 = 1;

    PC_ODR_bit.ODR4 = 1;

    for(j=0; j<800; j++);

    PC_ODR_bit.ODR4 = 0;

  }

  //назад лево

  else if ((PB_IDR_bit.IDR7 == 1) && (PB_IDR_bit.IDR6 == 0) && (PC_IDR_bit.IDR3 == 0) && (PC_IDR_bit.IDR2 == 1))

  {
```

```
    PE_ODR_bit.ODR5 = 0;

    spi_send (nRF24L01_W_TX_PAYLOAD); //R_для отправки

    spi_send (0x07); // то, что мы отправляем приемнику!!!!!!!!!!!!!!!!!!!!!!!!!!!

    PE_ODR_bit.ODR5 = 1;

    PC_ODR_bit.ODR4 = 1;

    for(j=0; j<800; j++);

    PC_ODR_bit.ODR4 = 0;

  }

  //назад право

  else if ((PB_IDR_bit.IDR7 == 1) && (PB_IDR_bit.IDR6 == 0) && (PC_IDR_bit.IDR3 == 1) && (PC_IDR_bit.IDR2 == 0))

  {

    PE_ODR_bit.ODR5 = 0;

    spi_send (nRF24L01_W_TX_PAYLOAD); //R_для отправки

    spi_send (0x08); // то, что мы отправляем приемнику!!!!!!!!!!!!!!!!!!!!!!!!!!!

    PE_ODR_bit.ODR5 = 1;

    PC_ODR_bit.ODR4 = 1;

    for(j=0; j<800; j++);

    PC_ODR_bit.ODR4 = 0;

  }

  // просто стоит

 else if ((PB_IDR_bit.IDR7 == 1) && (PB_IDR_bit.IDR6 == 1) && (PC_IDR_bit.IDR3 == 1) && (PC_IDR_bit.IDR2 == 1))
```

```
   {
     PE_ODR_bit.ODR5 = 0;
     spi_send (nRF24L01_W_TX_PAYLOAD); //R_для отправки
     spi_send (0x0F); // то, что мы отправляем приемнику!!!!!!!!!!!!!!!!!!!!!!!!!!!
     PE_ODR_bit.ODR5 = 1;
     PC_ODR_bit.ODR4 = 1;
     for(j=0; j<800; j++);
     PC_ODR_bit.ODR4 = 0;
   }
  }
}

int spi_send (int data)
{
  SPI_DR=data;
  while (SPI_SR_bit.TXE==0);//ждать, пока данные DR отправ¤тс¤
  while (SPI_SR_bit.RXNE==0);//ждать окончани¤ передачи
  ret=SPI_DR;
  return ret;
}

//подпрограмма настройки прерываний
void interruptSend_init(void)
```

```
{
  EXTI_CR1|=0x80;
  asm("rim");        //глобальное разрешение прерываний
  PB_DDR_bit.DDR4 = 0;        //вход
  PB_CR1_bit.C14 = 0;  //дифференциальный вход
  PB_CR2_bit.C24 = 1;  //разрешить внешние прерывани¤
}


//обработчик прерывания
__interrupt void EXTI_PC7(void)
{
  PC_ODR_bit.ODR4 = 0;
  PE_ODR_bit.ODR5 = 0; //обнулить флаг прерывания
  spi_send (0x27); //W_r
  spi_send (0x2E); //запись пред значения регистра для обнуления флага завершения отправки
  PE_ODR_bit.ODR5 = 1;


  PE_ODR_bit.ODR5 = 0; //считать флаг прерывания
  spi_send (0x07); //R_r
  a=spi_send (0xff); //такты, во врем¤ которых идет считывание из NRF
  PE_ODR_bit.ODR5 = 1;
}
```

Заключение

В результате выполнения данного курсового проекта была изучена работа микроконтроллера STM8S003K3, его работа с периферийными устройствами такими как DC моторчики, радио-модулем nRF24L01+ и драйвером L293D.

Изучены периферические устройства DC моторчики, радио-модулем nRF24L01+ и драйвером L293D, их архитектура, настройка и принципы работы.

В результате была сделана машинка на пульте управления, которая работает на основе STM8S003K3, и передача сигнала между передатчиком и приемником осуществляется при помощи nRF24L01+. Разработан принцип заботы, радиоуправления – в зависимости от комбинаций сигналов на портах подключенных к кнопкам управления машинка двигается в определенном направлении.