#### Министерство науки и высшего образования Российской Федерации Федеральное государственное автономное образовательное учреждение высшего образования

### P5#y1«Пермский национальный исследовательский политехнический университет»

#### Электротехнический факультет

#### Кафедра «Информационные технологии и автоматизированные системы» направление подготовки: 15.03.06 Мехатроника и робототехника

**О Т Ч Е Т**

**по учебно-исследовательской работе**

#### Выполнил студент гр. ИПР-22-1б Гребенщиков Кирилл Владимирович

(фамилия, имя, отчество)

(подпись)

#### Проверил:

#### ст. преподаватель Д.А. Карлов

(должность, Ф.И.О. руководителя по практической подготовке от кафедры)

(оценка) (подпись)

(дата)

#### Министерство науки и высшего образования Российской Федерации Федеральное государственное автономное образовательное учреждение высшего образования

### P56#y1«Пермский национальный исследовательский политехнический университет»

#### Электротехнический факультет

#### Кафедра «Информационные технологии и автоматизированные системы» направление подготовки: 15.03.06 Мехатроника и робототехника

**О Т Ч Е Т**

**по учебно-исследовательской работе**

#### Выполнил студент гр. ИПР-22-1б

#### Мисюрев Михаил Валерьевич

(фамилия, имя, отчество)

(подпись)

#### Проверил:

#### ст. преподаватель Д.А. Карлов

(должность, Ф.И.О. руководителя по практической подготовке от кафедры)

(оценка) (подпись)

(дата)

Пермь 2024

**РЕФЕРАТ**

## Лодка на радиоуправлении c водомётом для тушения пожаров с воды. Объём отчёта 16 страниц.

## Количество иллюстраций 13.

## Количество таблиц 1.

## Количество используемых источников 4. Объектом исследования является лодка на радиоуправлении.

## Ключевые слова: лодка, водомёт, электронные компоненты, радиомо- дуль, NRF24L01, L298N, сервопривод, MR996R, схемы.

## Предметом исследования является оборудование для тушения пожаров, установленное на лодке.

## Целью исследования является создание лодки на радиоуправлении для пожаротушения.

## В результате проведенного исследования будет разработана лодка на радиоуправлении.

**Содержание**

[Введение 4](#_TOC_250003)

1. [Необходимые материалы 5](#_TOC_250002)
2. [Организационно-штатная структура 12](#_TOC_250001)

[Заключение 13](#_TOC_250000)

Список литературы 14

# ВВЕДЕНИЕ

## Лодка на радиоуправлении которая будет носить название "Титаник". Лодка будет работать на базе микроконтроллера Arduino Nano с сервоприводом и управляться при помощи пульта дистанционного управления, по воде лодка будет передвигаться за счёт гребного винта, соединённого с электродвигателем. А также лодка будет способна поворачивать за счёт рулей, которые будут расположены с двух сторон от гребного винта, соединёнными с сервоприводом при помощи рычага. Так же с помощью помпы лодка сможет забирать воду из под себя и выстреливать в нужном направлении, за счёт сервопривода будет изменятся направление выстрела воды.

## Цель исследования: спроектировать и запрограммировать лодку на радио- управлении на базе микроконтроллера Arduino Nano.

## Задачи исследования:

## Составить список комплектующих.

## Составить схему подключения электронных компонентов устройства.

## Собрать схему электронных компонентов.

## Создать модель корпуса лодки и распечатать её.

## Установить аппаратную часть в корпус.

## Написать код в Arduino IDE для передвижения лодки.

# Необходимые материалы

## Мы составили список комплектующих, которые нам понадобятся для создания лодки на ду, и купили их.

## Список комплектующих:

## Плата Arduino nano (2шт) (рис.1)

## Сервопривод (2шт) (рис.2)

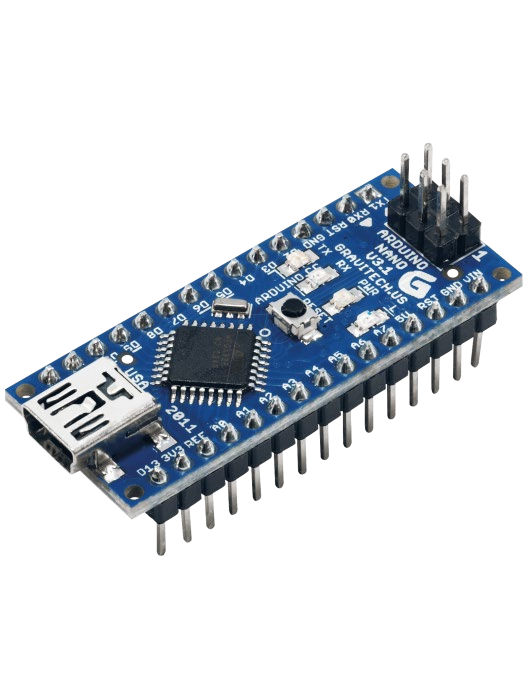
## Электромотор (5В) (2шт) (рис.3)

## Аккумулятор (9В) (рис.4)

## Н-мост (L298В) (рис.5)

## Радиомодуль (2шт) (рис.6)

## Модуль Raspberry Pi Camera Module V2



## Рисунок №1 – Arduino Nano[1]



## Рисунок №2 – Сервопривод[2]



## P300#yIS1Рисунок №3 – Электромотор[3]

## P302#yIS1Рисунок №4 – Аккумулятор[4]

## Рисунок №5 – Драйвер L298N[5]

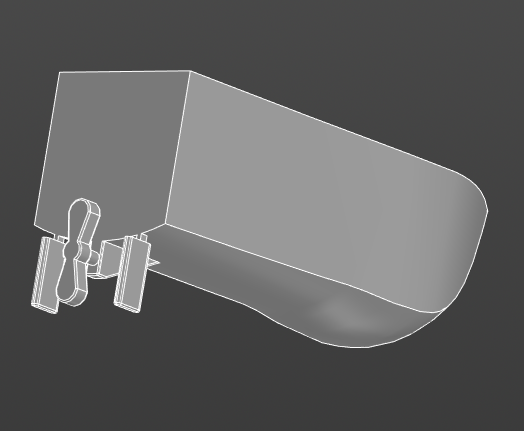


## Рисунок №6 – радиомодуль NRF24L01[6]

## Изображение выглядит как Электронная техника, Электронный компонент, схема, Компонент схемы Контент, сгенерированный ИИ, может содержать ошибки.В ходе работы, мы использовали следующую схему (рис.7)

## Рисунок 7 – электрическая схема устройства лодки[7]

Был разработан корпус лодки (рис. 8-10).



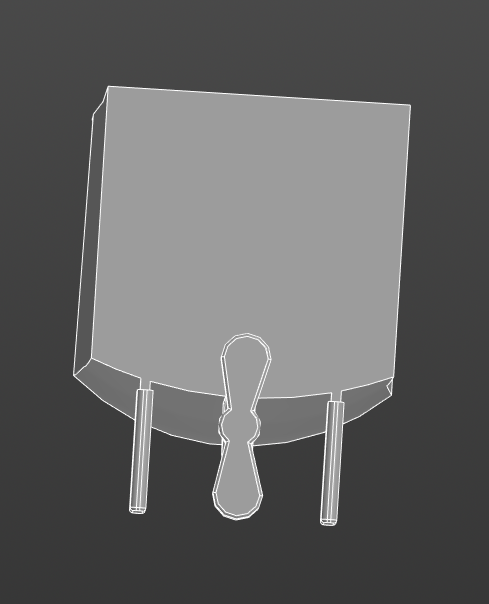


Рисунок №8 - Вид сбоку[8]

Рисунок №9 – Вид с сзади[9]

## 

## 

Рисунок №10 – Вид спереди[10]

## Рисунок №11 – Модуль Raspberry Pi Camera Module V2

## Для того, чтобы схема правильно работала, в среде программирования Ar- duino IDE был написан следующий код для приёмника:

#include <Wire.h>

#include <Adafruit\_PWMServoDriver.h>

Adafruit\_PWMServoDriver pwm1 = Adafruit\_PWMServoDriver(0x40);

#define SERVOMIN 150 // изменять значения пока не откалибруете сервопривод на по- ложение в 0 градусов

#define SERVOMAX 600 // изменять значения пока не откалибруете сервопривод на по- ложение в требуемое кол-во градусов

#define SERVO\_1 0

#include <SPI.h> // Подключаем библиотеку для работы с SPI-интерфейсом #include <nRF24L01.h> // Подключаем файл конфигурации из библиотеки RF24 #include <RF24.h> // Подключаем библиотеку для работа для работы с модулем NRF24L01

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| #define | IN1 | 8 | // | Input1 | подключен | к | выводу | 8 |
| #define | IN2 | 7 | // | Input2 | подключен | к | выводу | 7 |
| #define | IN3 | 6 | // | Input3 | подключен | к | выводу | 6 |
| #define | IN4 | 2 | // | Input4 | подключен | к | выводу | 2 |
| #define | EN1 | 3 | // | Input4 | подключен | к | выводу | 3 |
| #define | EN2 | 5 | // | Input4 | подключен | к | выводу | 5 |

#define CE 10 // Номер пина Arduino, к которому подключен вывод CE радиомодуля #define CSN 9 // Номер пина Arduino, к которому подключен вывод CSN радиомодуля RF24 radio(CE,CSN); // Создаём объект radio с указанием выводов CE и CSN

byte recieved\_data[2]; // массив принятых данных

byte vint = 3; // EN1 на 3 цифровом

byte address[][6] = {"1Node", "2Node", "3Node", "4Node", "5Node", "6Node"}; //воз- можные номера труб

void setup() { Serial.begin(9600); pinMode (EN1, OUTPUT); pinMode (IN1, OUTPUT); pinMode (IN2, OUTPUT); pinMode (EN2, OUTPUT); pinMode (IN4, OUTPUT); pinMode (IN3, OUTPUT);

pwm1.begin(); // активируем PCA9685

pwm1.setPWMFreq(50); // Задаём частоту подаваемую на сервопривод

radio.begin(); //активировать модуль

radio.setAutoAck(1); // режим подтверждения приёма, 1 вкл 0 выкл radio.setRetries(0, 15); // (время между попыткой достучаться, число попыток) radio.enableAckPayload(); // разрешить отсылку данных в ответ на входящий сиг-

нал

radio.setPayloadSize(32); // размер пакета, в байтах

radio.openReadingPipe(1, address[0]); // хотим слушать трубу 0 radio.setChannel(0x60); // выбираем канал (в котором нет шумов!)

radio.setPALevel (RF24\_PA\_MAX); // уровень мощности передатчика. На выбор RF24\_PA\_MIN, RF24\_PA\_LOW, RF24\_PA\_HIGH, RF24\_PA\_MAX

radio.setDataRate (RF24\_250KBPS); // скорость обмена. На выбор RF24\_2MBPS, RF24\_1MBPS, RF24\_250KBPS

//должна быть одинакова на приёмнике и передатчике!

//при самой низкой скорости имеем самую высокую чувствительность и дальность!!

radio.powerUp(); // начать работу

radio.startListening(); // начинаем слушать эфир, мы приёмный модуль

}

void loop() { byte pipeNo;

while ( radio.available(&pipeNo)) { // есть входящие данные

// чиатем входящий сигнал

radio.read(&recieved\_data, sizeof(recieved\_data));

// подать на драйвер сигнал с 0 места массива

analogWrite(vint, recieved\_data[0]);

// повернуть серво на угол 0..180

// значение получено с 1 элемента массива

pwm1.Write(recieved\_data[1]);

}

}

## Для передатчика:

#include <SPI.h> // Подключаем библиотеку для работы с SPI-интерфейсом #include <nRF24L01.h> // Подключаем файл конфигурации из библиотеки RF24 #include <RF24.h> // Подключаем библиотеку для работа для работы с модулем NRF24L01

#define CE 10 // Номер пина Arduino, к которому подключен вывод CE радиомодуля #define CSN 9 // Номер пина Arduino, к которому подключен вывод CSN радиомодуля RF24 radio(CE,CSN); // Создаём объект radio с указанием выводов CE и CSN

byte address[][6] = {"1Node", "2Node", "3Node", "4Node", "5Node", "6Node"}; //воз- можные номера труб

byte OX = 4; // Ось Х джостика на 4 аналоговом

byte OY = 5; // Ось Y джостика на 5 аналоговом

byte transmit\_data[2]; // массив, хранящий передаваемые данные

byte latest\_data[2]; // массив, хранящий последние переданные данные

boolean flag; // флажок отправки данных

void setup() {

Serial.begin(9600); //открываем порт для связи с ПК

radio.begin(); // активировать модуль

radio.setAutoAck(1); // режим подтверждения приёма, 1 вкл 0 выкл radio.setRetries(0, 15); // (время между попыткой достучаться, число попыток) radio.enableAckPayload(); // разрешить отсылку данных в ответ на входящий сиг-

нал

radio.setPayloadSize(32); // размер пакета, в байтах

radio.openWritingPipe(address[0]); // мы - труба 0, открываем канал для пере- дачи данных

radio.setChannel(0x60); // выбираем канал (в котором нет шумов!)

radio.setPALevel (RF24\_PA\_MAX); // уровень мощности передатчика. На выбор RF24\_PA\_MIN, RF24\_PA\_LOW, RF24\_PA\_HIGH, RF24\_PA\_MAX

radio.setDataRate (RF24\_250KBPS); // скорость обмена. На выбор RF24\_2MBPS, RF24\_1MBPS, RF24\_250KBPS

//должна быть одинакова на приёмнике и передатчике!

//при самой низкой скорости имеем самую высокую чувствительность и дальность!!

radio.powerUp(); //начать работу

radio.stopListening(); //не слушаем радиоэфир, мы передатчик

}

void loop() {

transmit\_data[0] = map(analogRead(OX), 509, 1023, 0, 255 ); // получить значение

// в диапазоне 0..1023, перевести в 0..255, и записать на 1 место в массиве

transmit\_data[1] = map(analogRead(OY), 509, 1023, 0, 255);

for (int i = 0; i < 2; i++) { // в цикле от 0 до числа каналов

if (transmit\_data[i] != latest\_data[i]) { // если есть изменения в trans-

mit\_data

flag = 1; // поднять флаг отправки по радио

latest\_data[i] = transmit\_data[i]; // запомнить последнее изменение

}

}

if (flag == 1) {

radio.powerUp(); // включить передатчик

radio.write(&transmit\_data, sizeof(transmit\_data)); // отправить по радио

flag = 0; //опустить флаг

radio.powerDown(); // выключить передатчик

}

}

Для сервопривода и моторчиков:

#include <Servo.h> // подключаем библиотеку для сервопривода

Servo servo; // объявляем переменную servo типа "servo"

int val;

int motorPin1 = 12;

int motorPin2 = 11; // освобождаем память в контроллере для переменой

void setup() {

Serial.begin(9600);

servo.attach(13);

pinMode(12, OUTPUT);

pinMode(11, OUTPUT); // привязываем сервопривод к выходу 11

}

void loop() {

// проверяем, поступают ли какие-то команды

if (Serial.available()) {

val = Serial.read(); // переменная val равна полученной команде

if (val == 'd') { servo.write(135); delay(1000); servo.write(90); } // при 1 поворачиваем серво на 10

if (val == 'a') { servo.write(45); delay(1000); servo.write(90); } // при 2 поворачиваем серво на 45

if (val == 'w') {digitalWrite(motorPin1, HIGH);digitalWrite(motorPin2, LOW); delay(1000);digitalWrite(motorPin1, LOW);digitalWrite(motorPin2, LOW);}

if (val == 's') {digitalWrite(motorPin1, LOW);digitalWrite(motorPin2, HIGH); delay(1000);digitalWrite(motorPin1, LOW);digitalWrite(motorPin2, LOW);}

}

}

Для Raspberry Pi (Python):

import cv2  
import serial  
import time  
import numpy as np  
# Настройки последовательного порта (скорость и порт могут отличаться)  
SERIAL\_PORT = "/dev/ttyACM0" # Для Arduino на USB  
SERIAL\_BAUDRATE = 9600  
# Настройки камеры  
CAMERA\_WIDTH = 640  
CAMERA\_HEIGHT = 480  
try:  
ser = serial.Serial(SERIAL\_PORT, SERIAL\_BAUDRATE, timeout=1)  
print("Подключено к Arduino")  
except serial.SerialException as e:  
print(f"Ошибка подключения к Arduino: {e}")  
exit(1)  
camera = cv2.VideoCapture(0) # Открыть камеру Raspberry Pi  
camera.set(cv2.CAP\_PROP\_FRAME\_WIDTH, CAMERA\_WIDTH)  
camera.set(cv2.CAP\_PROP\_FRAME\_HEIGHT, CAMERA\_HEIGHT)  
def process\_frame(frame):  
# 1. Предобработка изображения (уменьшение шума, изменение цветовой модели)  
gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY) # Конвертируем в черно-белый  
blurred = cv2.GaussianBlur(gray, (5,5), 0) # Размытие для уменьшения шума  
# 2. Обнаружение объектов (простой пример: обнаружение контуров)  
edges = cv2.Canny(blurred, 50, 150) # Обнаружение границ  
contours, \_ = cv2.findContours(edges, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)  
# Находим самый большой контур (предположительно препятствие)  
if contours:  
biggest\_contour = max(contours, key = cv2.contourArea)  
M = cv2.moments(biggest\_contour)  
if M["m00"] > 0:  
cX = int(M["m10"] / M["m00"])  
cY = int(M["m01"] / M["m00"])  
[cv2.circle](https://vk.com/away.php?to=http%3A%2F%2Fcv2.circle&cc_key=)(frame, (cX, cY), 7, (0, 0, 255), -1) # Рисуем кружок на центре  
# 3. Принимаем решение на основе положения препятствия  
if cX < CAMERA\_WIDTH/3:  
command = "L" # Поворот влево  
elif cX > CAMERA\_WIDTH\*2/3:  
command = "R" # Поворот вправо  
else:  
command = "F" # Плыть прямо  
else:  
command = "F"  
else:  
command = "F" # Если нет объектов - плыть прямо  
return command, frame  
while True:  
ret, frame = [camera.read](https://vk.com/away.php?to=http%3A%2F%2Fcamera.read&cc_key=)()  
if not ret:  
print("Не удалось получить кадр с камеры.")  
break  
command, frame = process\_frame(frame)  
cv2.imshow("Camera feed", frame) # Показ кадров с камеры (для отладки)  
try:  
ser.write(command.encode()) # Отправляем команду на Arduino  
print(f"Отправлена команда: {command}")  
except serial.SerialException as e:  
print(f"Ошибка отправки команды на Arduino: {e}")  
time.sleep(1)  
key = cv2.waitKey(1) & 0xFF  
if key == ord('q'): # Выход по нажатию 'q'  
break  
camera.release()  
cv2.destroyAllWindows()  
ser.close()

2. Arduino (C++):  
[#include](https://vk.com/im?sel=302293251&st=%23include) <Servo.h>  
[#define](https://vk.com/im?sel=302293251&st=%23define) LEFT\_MOTOR\_PIN 2 // Пин левого мотора  
[#define](https://vk.com/im?sel=302293251&st=%23define) RIGHT\_MOTOR\_PIN 3 // Пин правого мотора  
Servo left\_motor;  
Servo right\_motor;  
const int STRAIGHT\_SPEED = 1500; // Скорость прямо  
const int TURN\_SPEED = 1300; // Скорость при повороте  
void setup() {  
Serial.begin(9600); // Инициализация последовательного порта  
left\_motor.attach(LEFT\_MOTOR\_PIN); // Присоединяем мотор к пину  
right\_motor.attach(RIGHT\_MOTOR\_PIN); // Присоединяем мотор к пину  
}  
void loop() {  
if (Serial.available() > 0) {  
char command = [Serial.read](https://vk.com/away.php?to=http%3A%2F%2FSerial.read&cc_key=)(); // Получаем команду с Raspberry Pi  
if (command == 'L') {  
turnLeft();  
} else if (command == 'R') {  
turnRight();  
} else if (command == 'F') {  
goForward();  
}  
}  
}  
void goForward(){  
left\_motor.writeMicroseconds(STRAIGHT\_SPEED);  
right\_motor.writeMicroseconds(STRAIGHT\_SPEED);  
}  
void turnLeft(){  
left\_motor.writeMicroseconds(TURN\_SPEED); // Уменьшаем скорость левого  
right\_motor.writeMicroseconds(STRAIGHT\_SPEED); // Держать правою впереди  
delay(200);  
goForward();  
}  
void turnRight(){  
left\_motor.writeMicroseconds(STRAIGHT\_SPEED); // Держать левую впереди  
right\_motor.writeMicroseconds(TURN\_SPEED); // Уменьшаем скорость правого  
delay(200);  
goForward();  
}

# Организационно-штатная структура

## Гребенщиков Кирилл Владимирович - программный инженер группы.

## Мисюрев Михаил Валерьевич – моделирование и сборка.

# ЗАКЛЮЧЕНИЕ

## Нам удалось выполнить все поставленные задачи:

## Составлен список комплектующих.

## Купить необходимые комплектующие.

## Разработана схема подключения электронных компонентов.

## Собрана схему для передвижения лодки.

## Смоделировать корпус лодки.

## Написана программу в Arduino IDE для передвижения лодки. В ходе работы над поставленными задачами были выявлены следующие проблемы:

## Не оптимизированный код.

## Много работы с установкой компонентов в готовый корпус.

## При дальнейшей работе с данным проектом это проблемы можно решить так:

## Оптимизировать код.

## Распечатать корпус на 3D принтере.

## Собрать всё в корпусе.

**СПИСОК ИСПОЛЬЗОВАНЫХ ИСТОЧНИКОВ**

## PNGWING URL: https://[www.pngwing.com/ru/free-png-xzcpk](http://www.pngwing.com/ru/free-png-xzcpk) (дата обра- щения: 06.07.2023). - Arduino Nano

## Сервопривод // PNGWING URL: https://[www.pngwing.com/ru/free-png-](http://www.pngwing.com/ru/free-png-) xvdhm (дата обращения: 06.07.2023).

## Электромотор // RcDrive URL: https://rcdrive.ru/unit.php?unit=24180&pPage=11 (дата обращения: 06.07.2023).

## Аккумулятор // Ozon URL: https://[www.ozon.ru/product/2-sht-](http://www.ozon.ru/product/2-sht-) akkumulyator-9v-500mah-6f22-466858374/?sh=6oYUDYFd0Q (дата обра- щения: 06.07.2023).

## Драйвер L298N // PNGWING URL: https://[www.pngwing.com/ru/free-png-](http://www.pngwing.com/ru/free-png-) zdljf (дата обращения: 06.07.2023).

## Радиомодуль NRF24L01 // Вольтик URL: https://voltiq.ru/shop/nrf24l01- pa-lna/ (дата обращения: 06.07.2023).

## Datasheet драйвера L298N // Arduino Master URL: https://arduinomas- ter.ru/uroki-arduino/shema-raboty-n-mosta-dlya-upravleniya-dvigatelyami/ (дата обращения: 06.07.2023).

## Datasheet Arduino Nano // Arduino Nano datasheed URL: [http://arduino-](http://arduino-/) kid.ru/arduino\_nano\_datasheet (дата обращения: 06.07.2023).

## Datasheet сервопривода // Амперка URL: http://wiki.amperka.ru/articles:servo (дата обращния: 06.07.2023).