СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_heading=h.oj1kag9tt7jk)

[1. Теоретические основы решения индивидуального задания 4](#_heading=h.2dev1d5u3to7)

[2. Практическая часть индивидуального задания 5](#_heading=h.3znysh7)

[2.2. Описание применяющегося для решения задачи ПО 6](#_heading=h.od6w019j74fy)

[3. Выполнение общих заданий 29](#_heading=h.4d34og8)

[3.1. Алгоритм решения задачи 29](#_heading=h.2s8eyo1)

[3.2. Описание применяющегося для решения задач ПО 30](#_heading=h.d85yqesdzzk5)

[3.3. Скриншоты с результатами решения задач 31](#_heading=h.26in1rg)

[ВЫВОДЫ 36](#_heading=h.44sinio)

# 

ВВЕДЕНИЕ

Учебная практика по профессиональному модулю ПМ 02 «Осуществление интеграции программных модулей» представляет собой обязательный и важнейший этап подготовки будущих специалистов в сфере информационных технологий. Её основная цель заключается в закреплении и углублении теоретических знаний, полученных в процессе обучения, и формировании практических умений.

Цель практики:

1. Выполнить индивидуальное задание по теме «Изучение инструментов внедрения, поддержки и разработки информационных систем на платформе nanoPi»;
2. Пройти тестирование по олимпиаде «Траектория будущего» и выполнить все задания в адвент - челлендже для программистов на Stepik.

Задачи:

1. Подготовиться и пройти тестирование по олимпиаде «Траектория будущего»;
2. Выполнить все задания в адвент - челлендже для программистов;
3. Изучить учебные пособия, необходимые для работы с nanoPi;
4. Применить полученные знания на практике: подключить к микрокомпьютеру nanoPi различные модули, сервопривод Dynamixel и написать программы для их работы.

# Теоретические основы решения индивидуального задания

## Определения терминов предметной области

В соответствии с выбранным программным проектом был проведен анализ предметной области.

Назначением проектируемого программного средства является работа с микрокомпьютером nanoPi.

**NanoPi** - серия одноплатных компьютеров (SBC) производства компании FriendlyElec. Это компактные устройства, которые интегрируют все основные компоненты компьютера на одной плате, подходят для встраиваемых систем, плат разработки на базе ARM и компактных вычислительных решений.

**Сервопривод Dynamixel** - это устройство, которое совмещает в себе двигатель и контроллер.

**Светодиодный модуль** - это компактное устройство, состоящее из одного или нескольких светодиодов, объединённых на общей плате.

**Модуль трёхцветного светодиода (RGB LED)** - это устройство, в котором в одном корпусе заключены три разноцветных светодиода: красный, зелёный и синий.

**Модуль концевого выключателя** - это электромеханическое устройство, которое размыкает или замыкает цепь в заданных условиях, например, при достижении определённого положения управляющего механизма.

# Изучение инструментов внедрения, поддержки и разработки информационных систем на платформе nanoPi.

## Алгоритм решения задачи

1. Изучить учебное пособие «Одноплатный микрокомпьютер NANOPI - AR»;
2. Установить приложение MobaXterm;
3. Подключить nanoPi к компьютеру;
4. Установить приложение Dynamixel Wizard 2;
5. Узнать ID, номер протокола и скорость передачи у сервопривода;
6. Написать программу на мигание светодиода у сервопривода и его вращение;
7. Изучить учебное пособие «Периферийные функциональные модули. Часть 1»;
8. Узнать ID, номер протокола и скорость передачи у светодиода, трёхцветного светодиода и концевого выключателя;
9. Написать программы на мигание светодиода, мигание трёхцветного светодиода, мигание светодиода при нажатии на концевой выключатель, вращение сервопривода в разные стороны при нажатии на концевой выключатель и вращения сервопривода и свечения светодиод при нажатии на концевой выключатель.

# 

# 2.2. Описание применяющегося для решения задачи ПО

Для работы с nanoPi использовались следующие программы и утилиты:

Терминал в Linux - это программа, с помощью которой пользователь взаимодействует с операционной системой через интерфейс командной строки.

MobaXterm - программа для удалённого доступа и управления вычислительными ресурсами и устройствами.

DYNAMIXEL Wizard - программа для настройки и калибровки сервоприводов Dynamixel.

## Скриншоты с результатами выполнения индивидуального задания

Установили программу MobaXterm (рис. 1).



Рисунок 1 - Установка MobaXterm

Подключились к nanoPi по SSH (рис. 3). Для этого ввели хост 192.168.42.1 и имя пользователя root (рис. 2).

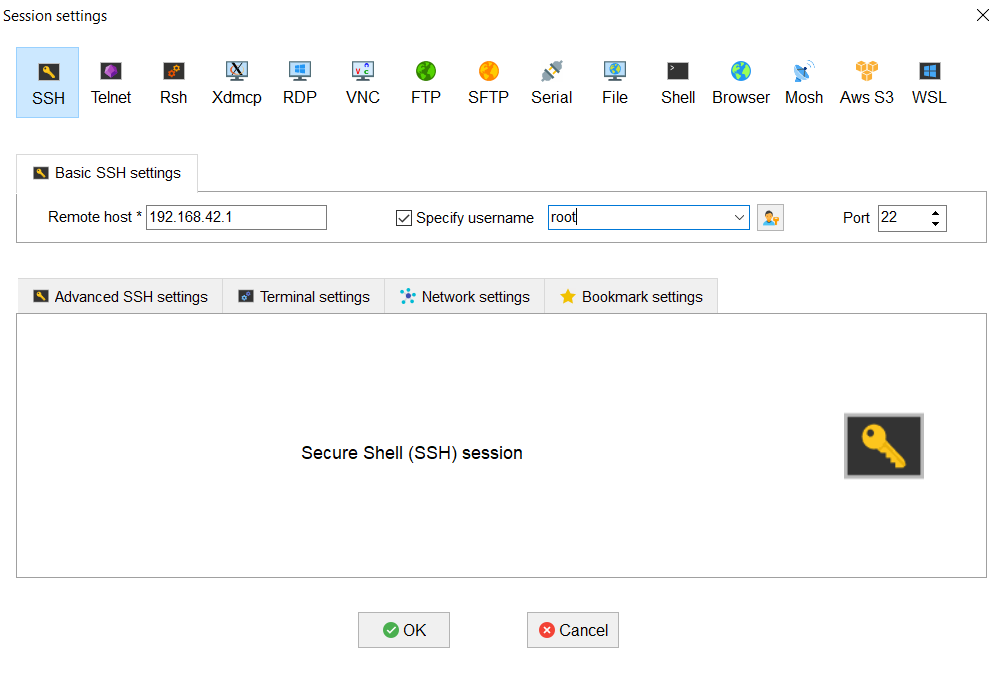


Рисунок 2 - Подключение к nanoPi

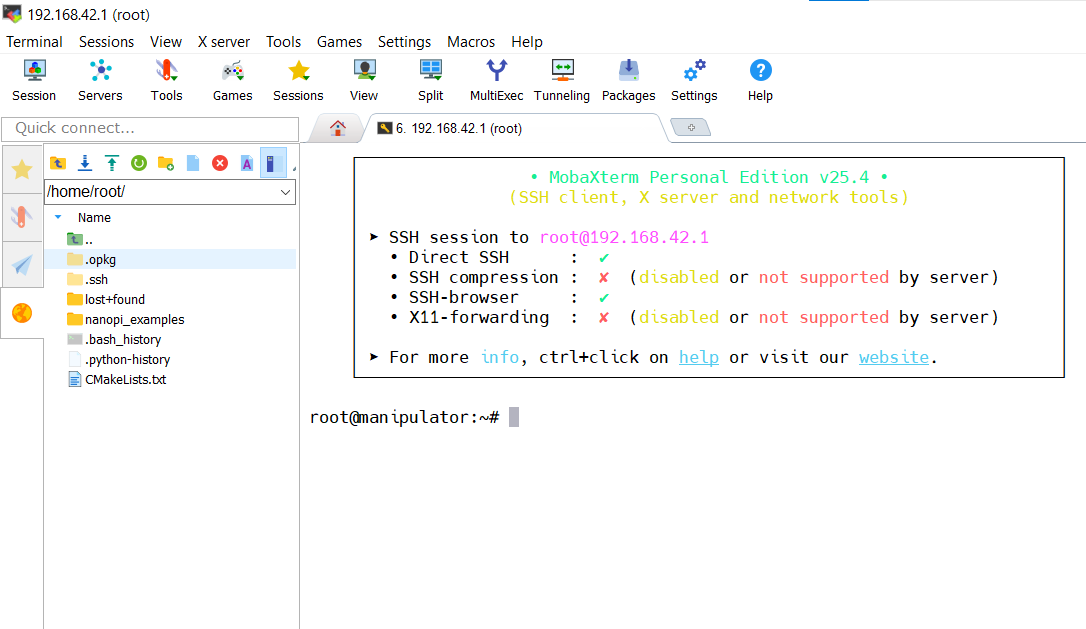


Рисунок 3 - Успешное подключение к nanoPi

Установили программу DYNAMIXEL Wizard, чтобы узнавать ID-адреса устройств DYNAMIXEL, их протоколы и скорость передачи (рис. 4).

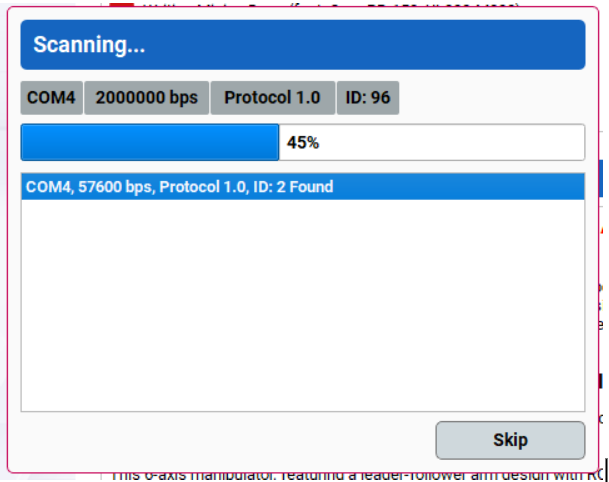


Рисунок 4 - Сканирование сервопривода

Подключили к nanoPi сервопривод Dynamixel и написали программу на мигание светодиода (Рис.5).

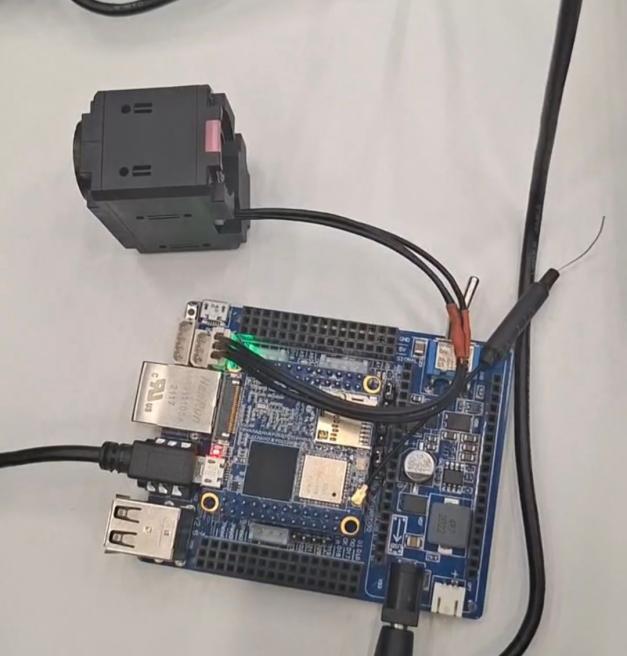


Рисунок 5 - Мигание светодиода на сервоприводе

Чтобы собрать проект, необходимо написать команду cmake (рис. 6).

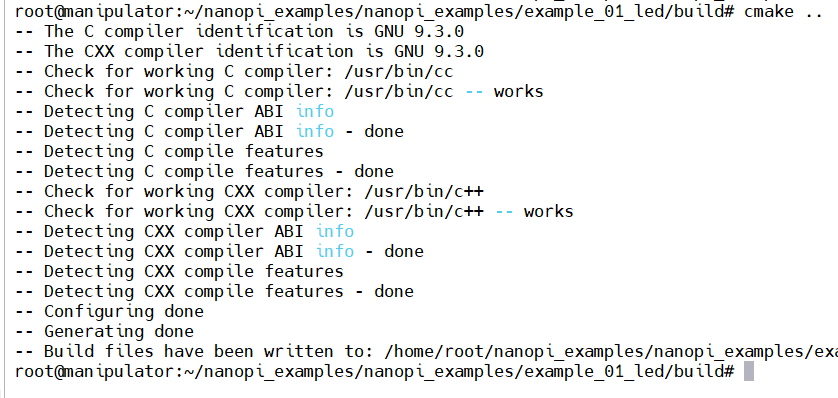


Рисунок 6 - Результат команды cmake

Чтобы скомпилировать проект, нужно написать команду make -j2 (рис. 7).



Рисунок 7 - Результат команды cmake

Написали программу на вращение сервопривода (рис. 8).



Рисунок 8 - Вращение светодиода

Объединили обе программы в одну:

#include <unistd.h>

#include "dynamixel\_sdk.h"

#include "fmt/format.h"

#define DEVICENAME "/dev/ttyS2"

#define PROTOCOL 2.0

#define BAUDRATE 57600

#define LED\_ID 1

dynamixel::PortHandler \*portHandler;

dynamixel::PacketHandler \*packetHandler;

void scanDevices() {

fmt::print("=== Scanning Dynamixel ===\n");

uint8\_t dxl\_error;

uint16\_t model\_number;

for(int id = 0; id < 253; id++) {

if(packetHandler->ping(portHandler, id, &model\_number, &dxl\_error) == COMM\_SUCCESS)

fmt::print("ID: {}, Model: {}\n", id, model\_number);

}

}

void checkRegisters() {

uint8\_t dxl\_error, value;

uint16\_t value16;

fmt::print("\n=== Checking registers ===\n");

struct RegInfo { int addr; const char\* name; bool is2b; };

RegInfo regs[] = {{0, "Model", true}, {2, "Firmware", false}, {7, "ID", false},

{8, "Baud", false}, {11, "Mode", false}, {31, "Temp", false},

{64, "Torque", false}, {65, "LED", false}, {70, "Error", false}};

for(int i = 0; i < sizeof(regs)/sizeof(regs[0]); i++) {

if(regs[i].is2b) {

if(packetHandler->read2ByteTxRx(portHandler, LED\_ID, regs[i].addr, &value16, &dxl\_error) == COMM\_SUCCESS)

fmt::print("{:10} [0x{:02X}]: {}\n", regs[i].name, regs[i].addr, value16);

} else {

if(packetHandler->read1ByteTxRx(portHandler, LED\_ID, regs[i].addr, &value, &dxl\_error) == COMM\_SUCCESS)

fmt::print("{:10} [0x{:02X}]: {}\n", regs[i].name, regs[i].addr, value);

}

}

}

void testLED() {

uint8\_t dxl\_error;

fmt::print("\n=== LED Test ===\n");

packetHandler->write1ByteTxRx(portHandler, LED\_ID, 65, 0, &dxl\_error);

sleep(1);

packetHandler->write1ByteTxRx(portHandler, LED\_ID, 65, 1, &dxl\_error);

sleep(1);

for(int i = 0; i < 10; i++) {

packetHandler->write1ByteTxRx(portHandler, LED\_ID, 65, 1, &dxl\_error);

usleep(250000);

packetHandler->write1ByteTxRx(portHandler, LED\_ID, 65, 0, &dxl\_error);

usleep(250000);

fmt::print(".");

}

fmt::print("\n");

}

void setPositionMode() {

uint8\_t dxl\_error;

packetHandler->write1ByteTxRx(portHandler, LED\_ID, 64, 0, &dxl\_error);

sleep(1);

packetHandler->write1ByteTxRx(portHandler, LED\_ID, 11, 3, &dxl\_error);

packetHandler->write1ByteTxRx(portHandler, LED\_ID, 64, 1, &dxl\_error);

}

void testPosition() {

uint8\_t dxl\_error;

int positions[] = {1024, 2048, 3072, 2048};

for(int i = 0; i < 4; i++) {

packetHandler->write4ByteTxRx(portHandler, LED\_ID, 116, positions[i], &dxl\_error);

sleep(2);

}

}

int main() {

portHandler = dynamixel::PortHandler::getPortHandler(DEVICENAME);

packetHandler = dynamixel::PacketHandler::getPacketHandler(PROTOCOL);

if(!portHandler->openPort() || !portHandler->setBaudRate(BAUDRATE)) {

fmt::print("Port error\n");

return 1;

}

scanDevices();

uint8\_t dxl\_error;

uint16\_t model\_number;

if(packetHandler->ping(portHandler, LED\_ID, &model\_number, &dxl\_error) != COMM\_SUCCESS) {

fmt::print("Ping failed\n");

portHandler->closePort();

return 1;

}

checkRegisters();

testLED();

setPositionMode();

testPosition();

packetHandler->write1ByteTxRx(portHandler, LED\_ID, 64, 0, &dxl\_error);

portHandler->closePort();

return 0;

}

Подключили модуль «Светодиод» к nanoPi и написали программу на мигание светодиода (рис. 9).

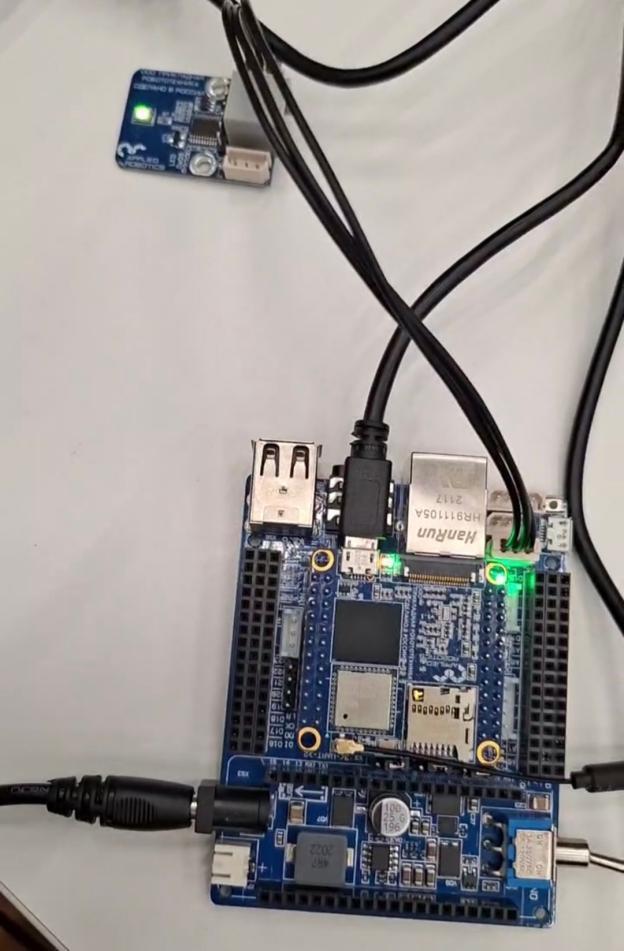


Рисунок 9 - Мигание модуля «Светодиод»

Код:

#include "dynamixel\_sdk.h"

#include <unistd.h>

#include <vector>

#define PROTOCOL\_VERSION 1.0

#define BAUDRATE 57600

#define PORT\_NAME "/dev/ttyS2"

int main() {

PortHandler\* port = PortHandler::getPortHandler(PORT\_NAME);

PacketHandler\* packet = PacketHandler::getPacketHandler(PROTOCOL\_VERSION);

if (!port->openPort() || !port->setBaudRate(BAUDRATE)) {

return -1;

}

usleep(1000000);

std::vector<int> found\_ids;

for (int id = 0; id <= 253; id++) {

uint16\_t model\_number;

uint8\_t error = 0;

if (packet->ping(port, id, &model\_number, &error) == COMM\_SUCCESS) {

found\_ids.push\_back(id);

if (id == 9) {

uint8\_t current\_value;

packet->read1ByteTxRx(port, id, 26, &current\_value, &error);

packet->write1ByteTxRx(port, id, 26, 200, &error);

sleep(2);

packet->write1ByteTxRx(port, id, 26, 0, &error);

sleep(2);

}

}

usleep(10000);

}

port->closePort();

return 0;

}

Подключили модуль «Трёхцветный светодиод» и написали программу на его мигание разными цветами (рис. 10).



Рисунок 10 - Мигание модуля «Трёхцветный светодиод»

Код:

#include "dynamixel\_sdk.h"

#include <unistd.h>

int main() {

dynamixel::PortHandler\* port = dynamixel::PortHandler::getPortHandler("/dev/ttyS2");

dynamixel::PacketHandler\* packet = dynamixel::PacketHandler::getPacketHandler(1.0);

if (!port->openPort() || !port->setBaudRate(57600)) return -1;

uint8\_t error = 0;

while (true) {

packet->write1ByteTxRx(port, 21, 27, 255, &error);

packet->write1ByteTxRx(port, 21, 28, 0, &error);

packet->write1ByteTxRx(port, 21, 29, 0, &error);

sleep(2);

packet->write1ByteTxRx(port, 21, 27, 0, &error);

packet->write1ByteTxRx(port, 21, 28, 255, &error);

packet->write1ByteTxRx(port, 21, 29, 0, &error);

sleep(2);

packet->write1ByteTxRx(port, 21, 27, 0, &error);

packet->write1ByteTxRx(port, 21, 28, 0, &error);

packet->write1ByteTxRx(port, 21, 29, 255, &error);

sleep(2);

packet->write1ByteTxRx(port, 21, 27, 255, &error);

packet->write1ByteTxRx(port, 21, 28, 255, &error);

packet->write1ByteTxRx(port, 21, 29, 255, &error);

sleep(2);

packet->write1ByteTxRx(port, 21, 27, 0, &error);

packet->write1ByteTxRx(port, 21, 28, 0, &error);

packet->write1ByteTxRx(port, 21, 29, 0, &error);

sleep(2);

}

port->closePort();

return 0;

}

Подключили концевой выключатель и сервопривод к nanoPi и написали код, чтобы при нажатии на концевой выключатель сервопривод крутился в разные стороны (рис. 11).



Рисунок 11 - Вращение сервопривода в разные стороны

Код:

#include "dynamixel\_sdk.h"

#include <unistd.h>

int main() {

dynamixel::PortHandler\* port = dynamixel::PortHandler::getPortHandler("/dev/ttyS2");

if (!port->openPort()) return -1;

dynamixel::PacketHandler\* p1 = dynamixel::PacketHandler::getPacketHandler(1.0);

dynamixel::PacketHandler\* p2 = dynamixel::PacketHandler::getPacketHandler(2.0);

port->setBaudRate(9600);

p2->write1ByteTxRx(port, 1, 64, 1, nullptr);

p2->write1ByteTxRx(port, 1, 11, 1, nullptr);

bool dir = true;

bool switch\_state = false;

while (true) {

port->setBaudRate(57600);

usleep(100000);

uint8\_t sw;

if (p1->read1ByteTxRx(port, 23, 27, &sw, nullptr) == COMM\_SUCCESS) {

if (sw == 1 && !switch\_state) {

dir = !dir;

port->setBaudRate(9600);

usleep(100000);

if (dir) p2->write4ByteTxRx(port, 1, 104, 250, nullptr);

else p2->write4ByteTxRx(port, 1, 104, -250, nullptr);

}

switch\_state = (sw == 1);

}

usleep(100000);

}

return 0;

}

Подключили концевой выключатель, сервопривод и два светодиода. Из - за того, что светодиоды имеют одинаковый id (id=9), нужно было поменять id у одного из светодиодов. Для этого нашли программу, потому что в DYNAMIXEL Wizard поменять id у светодиода нельзя (рис. 12). Теперь у светодиодов id 8 и 9.

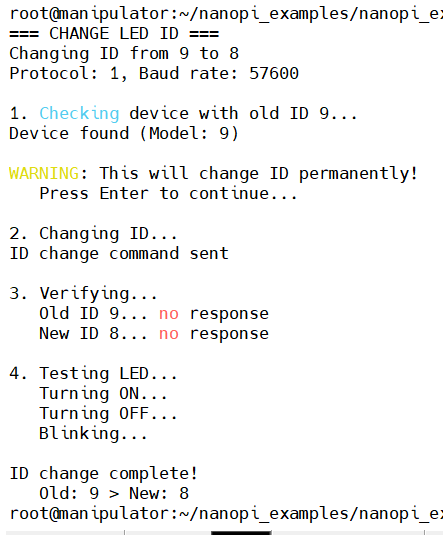


Рисунок 12 - Результат программы

Код:

#include "dynamixel\_sdk.h"

#include <iostream>

#include <unistd.h>

#define PORT\_NAME "/dev/ttyS2"

#define OLD\_ID 9

#define NEW\_ID 8

#define BAUDRATE 57600

#define PROTOCOL 1.0

int main() {

std::cout << "=== CHANGE LED ID ===" << std::endl;

std::cout << "Changing ID from " << OLD\_ID << " to " << NEW\_ID << std::endl;

std::cout << "Protocol: " << PROTOCOL << ", Baud rate: " << BAUDRATE << "\n" << std::endl;

dynamixel::PortHandler\* port = dynamixel::PortHandler::getPortHandler(PORT\_NAME);

dynamixel::PacketHandler\* packet = dynamixel::PacketHandler::getPacketHandler(PROTOCOL);

if (!port->openPort() || !port->setBaudRate(BAUDRATE)) {

std::cout << "Port error!" << std::endl;

return -1;

}

usleep(1000000);

uint8\_t error = 0;

uint16\_t model;

std::cout << "1. Checking device with old ID " << OLD\_ID << "..." << std::endl;

if (packet->ping(port, OLD\_ID, &model, &error) == COMM\_SUCCESS) {

std::cout << "Device found (Model: " << model << ")" << std::endl;

} else {

std::cout << "Device not found!" << std::endl;

port->closePort();

return -1;

}

std::cout << "\nWARNING: This will change ID permanently!" << std::endl;

std::cout << " Press Enter to continue...";

std::cin.ignore();

std::cout << "\n2. Changing ID..." << std::endl;

if (packet->write1ByteTxRx(port, OLD\_ID, 3, NEW\_ID, &error) == COMM\_SUCCESS) {

std::cout << "ID change command sent" << std::endl;

} else {

std::cout << "Failed!" << std::endl;

port->closePort();

return -1;

}

sleep(3);

std::cout << "\n3. Verifying..." << std::endl;

std::cout << " Old ID " << OLD\_ID << "... ";

if (packet->ping(port, OLD\_ID, &model, &error) == COMM\_SUCCESS) {

std::cout << "STILL RESPONDS!" << std::endl;

} else {

std::cout << "no response" << std::endl;

}

std::cout << " New ID " << NEW\_ID << "... ";

if (packet->ping(port, NEW\_ID, &model, &error) == COMM\_SUCCESS) {

std::cout << "RESPONDS!" << std::endl;

std::cout << " Model: " << model << std::endl;

} else {

std::cout << "no response" << std::endl;

}

std::cout << "\n4. Testing LED..." << std::endl;

std::cout << " Turning ON..." << std::endl;

packet->write1ByteTxRx(port, NEW\_ID, 26, 200, &error);

sleep(2);

std::cout << " Turning OFF..." << std::endl;

packet->write1ByteTxRx(port, NEW\_ID, 26, 0, &error);

sleep(1);

std::cout << " Blinking..." << std::endl;

for (int i = 0; i < 3; i++) {

packet->write1ByteTxRx(port, NEW\_ID, 26, 150, &error);

usleep(200000);

packet->write1ByteTxRx(port, NEW\_ID, 26, 0, &error);

usleep(200000);

}

port->closePort();

std::cout << "\nID change complete!" << std::endl;

std::cout << " Old: " << OLD\_ID << " → New: " << NEW\_ID << std::endl;

return 0;

}

Написали программу, чтобы при нажатии на концевой выключатель сервопривод крутился в разные стороны, а светодиоды мигали в зависимости от стороны вращения сервопривода (рис. 13, рис. 14, рис. 15).



Рисунок 13 - Вращение сервопривода в правую сторону



Рисунок 14 - Вращение сервопривода в левую сторону

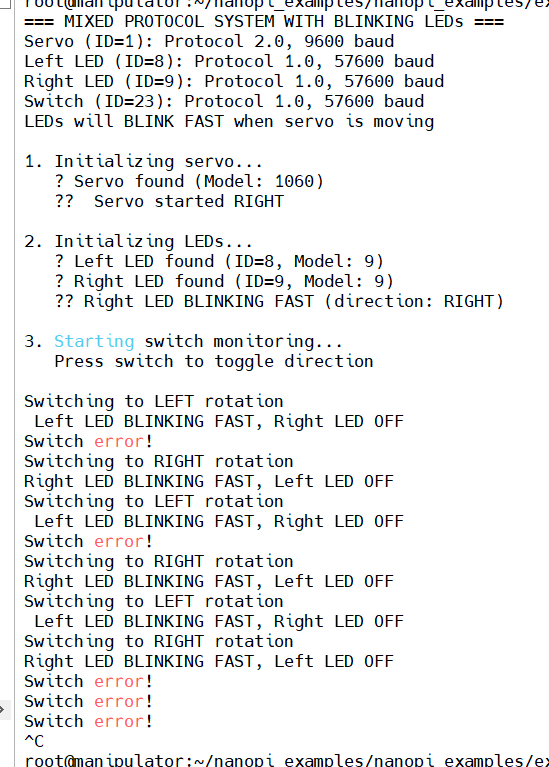


Рисунок 15 - Результат программы

Код:

#include "dynamixel\_sdk.h"

#include <iostream>

#include <unistd.h>

#define PORT\_NAME "/dev/ttyS2"

#define SERVO\_ID 1

#define LED\_LEFT\_ID 8

#define LED\_RIGHT\_ID 9

#define SWITCH\_ID 23

struct BlinkState {

bool left\_on, right\_on;

bool left\_state, right\_state;

unsigned long last\_time;

};

BlinkState blink\_state = {false, false, false, false, 0};

void updateBlinkState() {

unsigned long current\_time = clock() \* 1000000 / CLOCKS\_PER\_SEC;

if (current\_time - blink\_state.last\_time >= 100000) {

if (blink\_state.left\_on) blink\_state.left\_state = !blink\_state.left\_state;

else blink\_state.left\_state = false;

if (blink\_state.right\_on) blink\_state.right\_state = !blink\_state.right\_state;

else blink\_state.right\_state = false;

blink\_state.last\_time = current\_time;

}

}

void applyLEDStates(dynamixel::PacketHandler\* packet, dynamixel::PortHandler\* port) {

uint8\_t error = 0;

uint8\_t left\_val = blink\_state.left\_state ? 200 : 0;

if (packet->ping(port, LED\_LEFT\_ID, nullptr, &error) == COMM\_SUCCESS)

packet->write1ByteTxRx(port, LED\_LEFT\_ID, 26, left\_val, &error);

uint8\_t right\_val = blink\_state.right\_state ? 200 : 0;

if (packet->ping(port, LED\_RIGHT\_ID, nullptr, &error) == COMM\_SUCCESS)

packet->write1ByteTxRx(port, LED\_RIGHT\_ID, 26, right\_val, &error);

}

void setBlinkPattern(bool left, bool right) {

blink\_state.left\_on = left;

blink\_state.right\_on = right;

blink\_state.left\_state = left;

blink\_state.right\_state = right;

blink\_state.last\_time = clock() \* 1000000 / CLOCKS\_PER\_SEC;

}

int main() {

std::cout << "=== MIXED PROTOCOL SYSTEM ===" << std::endl;

bool servo\_right = true;

bool last\_switch = false;

blink\_state.last\_time = clock() \* 1000000 / CLOCKS\_PER\_SEC;

// Инициализация сервопривода

dynamixel::PortHandler\* servo\_port = dynamixel::PortHandler::getPortHandler(PORT\_NAME);

dynamixel::PacketHandler\* servo\_packet = dynamixel::PacketHandler::getPacketHandler(2.0);

if (servo\_port->openPort() && servo\_port->setBaudRate(9600)) {

usleep(2000000);

uint8\_t error = 0;

servo\_packet->write1ByteTxRx(servo\_port, SERVO\_ID, 64, 1, &error);

servo\_packet->write1ByteTxRx(servo\_port, SERVO\_ID, 11, 1, &error);

servo\_packet->write4ByteTxRx(servo\_port, SERVO\_ID, 104, 200, &error);

servo\_port->closePort();

}

// Инициализация светодиодов

setBlinkPattern(false, true);

while (true) {

updateBlinkState();

unsigned long current\_time = clock() \* 1000000 / CLOCKS\_PER\_SEC;

static unsigned long last\_led\_update = 0;

if (current\_time - last\_led\_update >= 20000) {

dynamixel::PortHandler\* update\_port = dynamixel::PortHandler::getPortHandler(PORT\_NAME);

dynamixel::PacketHandler\* update\_packet = dynamixel::PacketHandler::getPacketHandler(1.0);

if (update\_port->openPort() && update\_port->setBaudRate(57600)) {

applyLEDStates(update\_packet, update\_port);

update\_port->closePort();

last\_led\_update = current\_time;

}

}

// Проверка выключателя

dynamixel::PortHandler\* sw\_port = dynamixel::PortHandler::getPortHandler(PORT\_NAME);

dynamixel::PacketHandler\* sw\_packet = dynamixel::PacketHandler::getPacketHandler(1.0);

if (sw\_port->openPort() && sw\_port->setBaudRate(57600)) {

usleep(100000);

uint8\_t state, error = 0;

sw\_packet->read1ByteTxRx(sw\_port, SWITCH\_ID, 27, &state, &error);

sw\_port->closePort();

bool pressed = (state == 1);

if (pressed && !last\_switch) {

servo\_right = !servo\_right;

dynamixel::PortHandler\* s\_port = dynamixel::PortHandler::getPortHandler(PORT\_NAME);

dynamixel::PacketHandler\* s\_packet = dynamixel::PacketHandler::getPacketHandler(2.0);

if (s\_port->openPort() && s\_port->setBaudRate(9600)) {

usleep(100000);

if (servo\_right) {

s\_packet->write4ByteTxRx(s\_port, SERVO\_ID, 104, 200, &error);

setBlinkPattern(false, true);

std::cout << "RIGHT" << std::endl;

} else {

s\_packet->write4ByteTxRx(s\_port, SERVO\_ID, 104, -200, &error);

setBlinkPattern(true, false);

std::cout << "LEFT" << std::endl;

}

s\_port->closePort();

}

}

last\_switch = pressed;

}

usleep(5000);

}

return 0;

}

## Руководство системного программиста

Программа представляет собой систему управления сервоприводом Dynamixel с переключением направления вращения при нажатии концевого выключателя. Система реализует управление устройствами Dynamixel, использующими разные версии протокола через один последовательный порт с динамическим переключением скорости передачи данных.

Система выполняет следующие функции:

1. Управление направлением и скоростью вращения сервопривода Dynamixel;
2. Обработка сигналов концевого выключателя для переключения направления;
3. Динамическое управление скоростью передачи данных для работы с разными протоколами.

Система состоит из трех аппаратных компонентов, подключенных к одной шине Dynamixel:

1. Сервопривод (ID=1) - использует протокол 2.0, оптимальная скорость 9600 бод;
2. Концевой выключатель (ID=23) - использует протокол 1.0, скорость 57600 бод;
3. Последовательный порт /dev/ttyS2 - физический интерфейс связи.

Все устройства подключены к последовательному порту /dev/ttyS2 через общую шину Dynamixel.

Критической особенностью системы является необходимость работы с двумя разными версиями протокола Dynamixel на одном физическом порту:

1. Протокол 1.0 используется для опроса концевого выключателя;
2. Протокол 2.0 используется для управления сервоприводом.

Эта особенность требует динамического переключения скорости порта и смены пакетного обработчика во время выполнения программы, что создает уникальные требования к временным характеристикам системы.

При запуске система выполняет следующую последовательность действий:

1. Открывает последовательный порт /dev/ttyS2 на скорости 9600 бод;
2. Инициализирует два обработчика пакетов: для протокола 1.0 (p1) и протокола 2.0 (p2);
3. Настраивает сервопривод: включает крутящий момент и устанавливает режим управления скоростью;
4. Инициализирует переменные состояния: направление вращения (dir) и состояние выключателя (switch\_state);
5. Устанавливает начальное направление вращения сервопривода.

После инициализации система входит в бесконечный цикл обработки, состоящий из трех основных фаз:

Фаза 1: Подготовка к опросу (каждые 100 мс)

1. Переключение скорости порта на 57600 бод для работы с протоколом 1.0;
2. Пауза 100 мс для стабилизации связи после смены скорости.

Фаза 2: Опрос выключателя

1. Чтение состояния цифрового входа выключателя (адрес 27) с использованием обработчика p1;
2. Обнаружение фронта нажатия: переход из состояния "отпущен" (0) в состояние "нажат" (1);
3. Обновление переменной switch\_state для детектирования только фронтов сигнала.

Фаза 3: Реакция на нажатие

При обнаружении фронта нажатия система выполняет:

1. Инверсию направления вращения (dir = !dir);
2. Переключение скорости порта на 9600 бод для работы с протоколом 2.0;
3. Паузу 100 мс для стабилизации связи;
4. Установку новой скорости сервопривода: +250 для вращения вправо, -250 для вращения влево;
5. Возврат к началу цикла.

Поскольку все устройства используют один физический порт, но разные скорости и протоколы, система реализует следующий подход:

1. Динамическое переключение скорости: Порт постоянно переключается между 9600 и 57600 бод;
2. Разделение обработчиков: Для каждого протокола используется свой экземпляр PacketHandler;
3. Временные задержки: Между сменами скорости вводятся паузы для стабилизации связи;
4. Изоляция операций: Каждая операция выполняется с оптимальными параметрами для конкретного устройства.

Алгоритм обработки выключателя

Система реализует детектирование фронта сигнала для предотвращения множественных срабатываний:

if (sw == 1 && !switch\_state) { *// Обнаружение фронта (0→1)*

*// Обработка нажатия*

switch\_state = true; *// Предотвращение повторной обработки*

}

if (sw == 0 && switch\_state) { *// Обнаружение спада (1→0)*

switch\_state = false; *// Подготовка к следующему нажатию*

}

Сервопривод работает в режиме управления скоростью (Operating Mode = 1):

* Положительные значения: Вращение по часовой стрелке;
* Отрицательные значения: Вращение против часовой стрелки;
* Абсолютное значение 250: Установленная скорость вращения.

Критические регистры сервопривода:

* Адрес 64 (Torque Enable): Включение/выключение крутящего момента;
* Адрес 11 (Operating Mode): Режим работы (1 = контроль скорости);
* Адрес 104 (Goal Velocity): Целевая скорость вращения.

Система использует следующие временные параметры:

1. Основной цикл: ~200 мс на итерацию;
2. Задержка стабилизации: 100 мс после каждой смены скорости;
3. Опрос выключателя: Выполняется каждые 100 мс;
4. Реакция на нажатие: В пределах 200 мс от физического нажатия.

Особенности реализации

Потоковая модель: Система работает в единственном потоке выполнения с последовательной обработкой событий.

Управление ресурсами: Порт открывается один раз при старте и используется повторно со сменой параметров.

Обработка ошибок: Базовая проверка успешности открытия порта, но отсутствует обработка ошибок связи во время выполнения.

Тайминги: Жестко заданные временные задержки обеспечивают предсказуемое поведение, но могут требовать настройки для конкретного оборудования.

Для корректной работы системы необходимо:

1. Все устройства Dynamixel должны быть правильно подключены к шине с уникальными ID;
2. Последовательный порт должен поддерживать скорости 9600 и 57600 бод;
3. Сервопривод должен быть предварительно настроен на работу в режиме управления скоростью;
4. Концевой выключатель должен быть подключен к цифровому входу сервопривода ID 23;
5. Питание шины должно обеспечивать достаточный ток для сервопривода.

Ограничения системы

1. Производительность: Частое переключение скорости может создавать нагрузку на последовательный порт;
2. Надежность: Отсутствие полноценной обработки ошибок может привести к неожиданному поведению;
3. Гибкость: Жестко заданные параметры требуют перекомпиляции для изменения настроек;
4. Расширяемость: Архитектура не поддерживает простое добавление новых устройств.

Для промышленного использования рекомендуется:

1. Добавить полноценную обработку ошибок связи;
2. Реализовать конфигурацию через внешний файл;
3. Добавить логирование операций для отладки;
4. Внедрить механизм плавного изменения скорости;
5. Реализовать безопасную остановку по сигналам ОС.

Данная система представляет собой специализированное решение для управления сервоприводом с простой логикой переключения направления. Ее архитектура отражает компромисс между необходимостью работы с разными протоколами Dynamixel и ограничениями единого физического интерфейса связи.

# Выполнение общих заданий

## Алгоритм решения задачи

Алгоритм для прохождения тестирования по олимпиаде «Траектория будущего»:

1. Выполнить задания для подготовки;
2. Пройти тестирование по всем наминациям.

Алгоритм прохождения адвент-челленджа для программистов.

1. Записаться на курс на Stepik "Поколение Python": квесты, адвенты и конкурсы;
2. Выполнить задания.

## Описание применяющегося для решения задач ПО

Blender (Blender 3D) - бесплатная программа с открытым исходным кодом для 3D-моделирования, анимации, визуализации и создания видеоконтента.

Stepik - российская образовательная платформа и конструктор бесплатных и платных открытых онлайн-курсов и уроков.

Tinkercad - бесплатная онлайн-платформа для 3D-дизайна, симуляции электронных схем и блочного программирования, разработанная компанией Autodesk.

VR Concept - программное обеспечение для работы с 3D-моделями в виртуальной реальности (VR).

## Скриншоты с результатами решения задач

Blender:



Рисунок 16 - Кружка

Tinkercad:

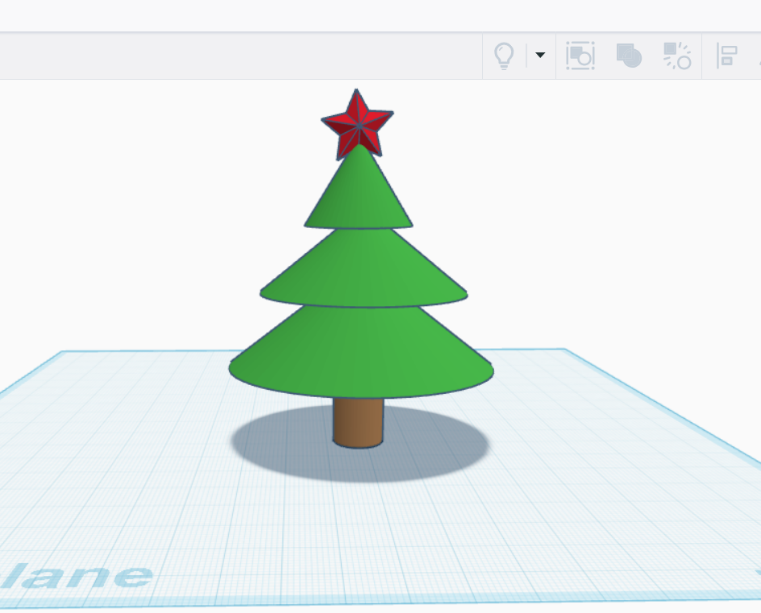


Рисунок 17 - Новогодняя ёлка

Пример задания по номинации «VR-разработка» (рис. 18):

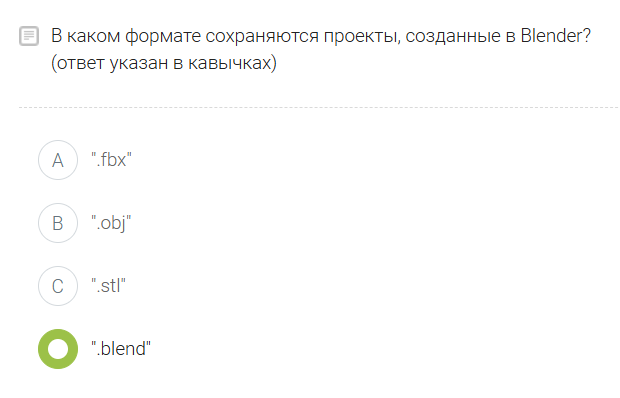


Рисунок 18 - Пример задания

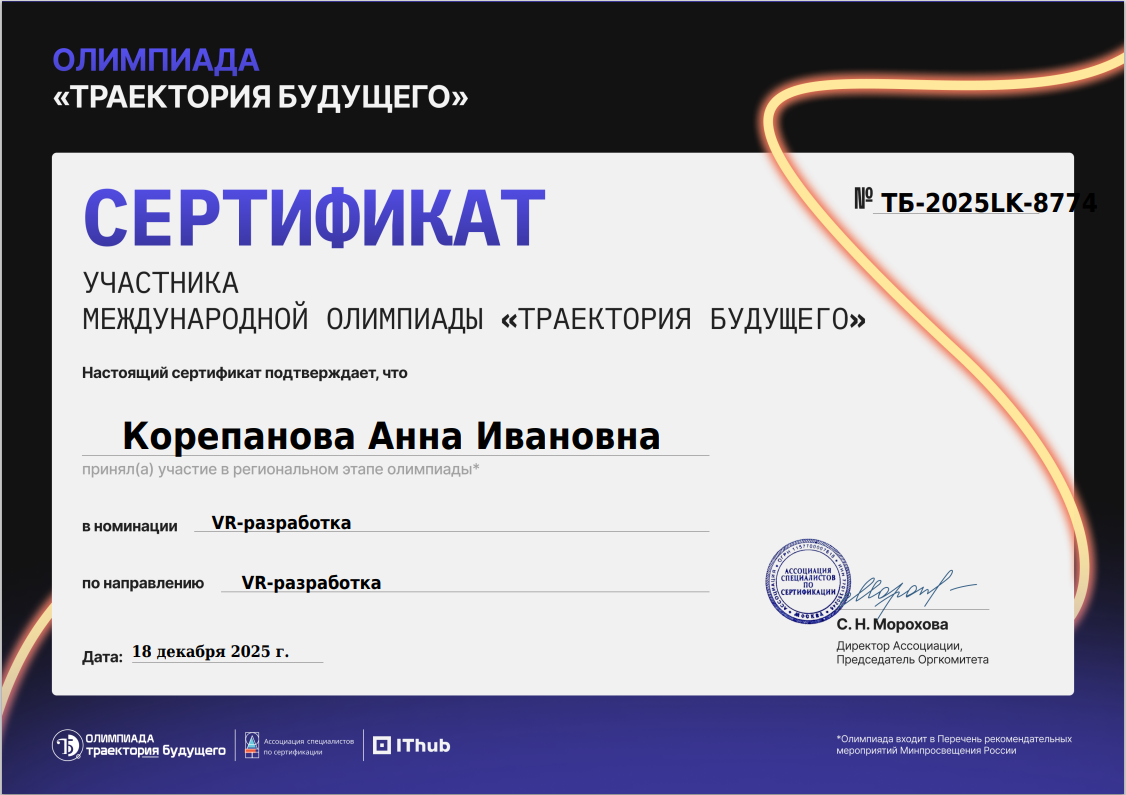


Рисунок 19 – Сертификат

Пример задания по номинации Экосистема «Группы Астра» (рис. 20):

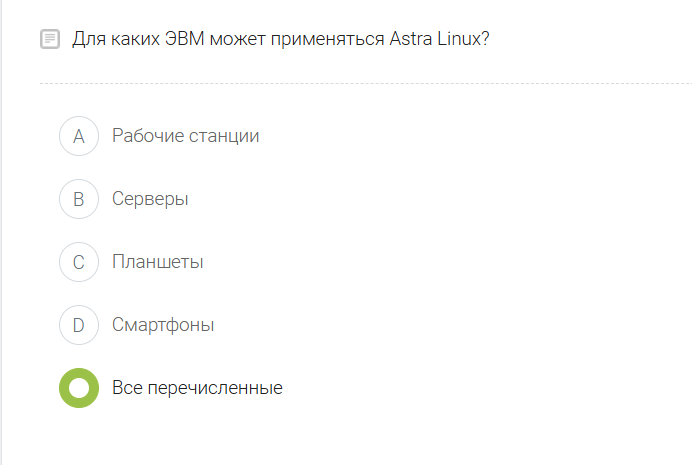


Рисунок 20 - Пример задания

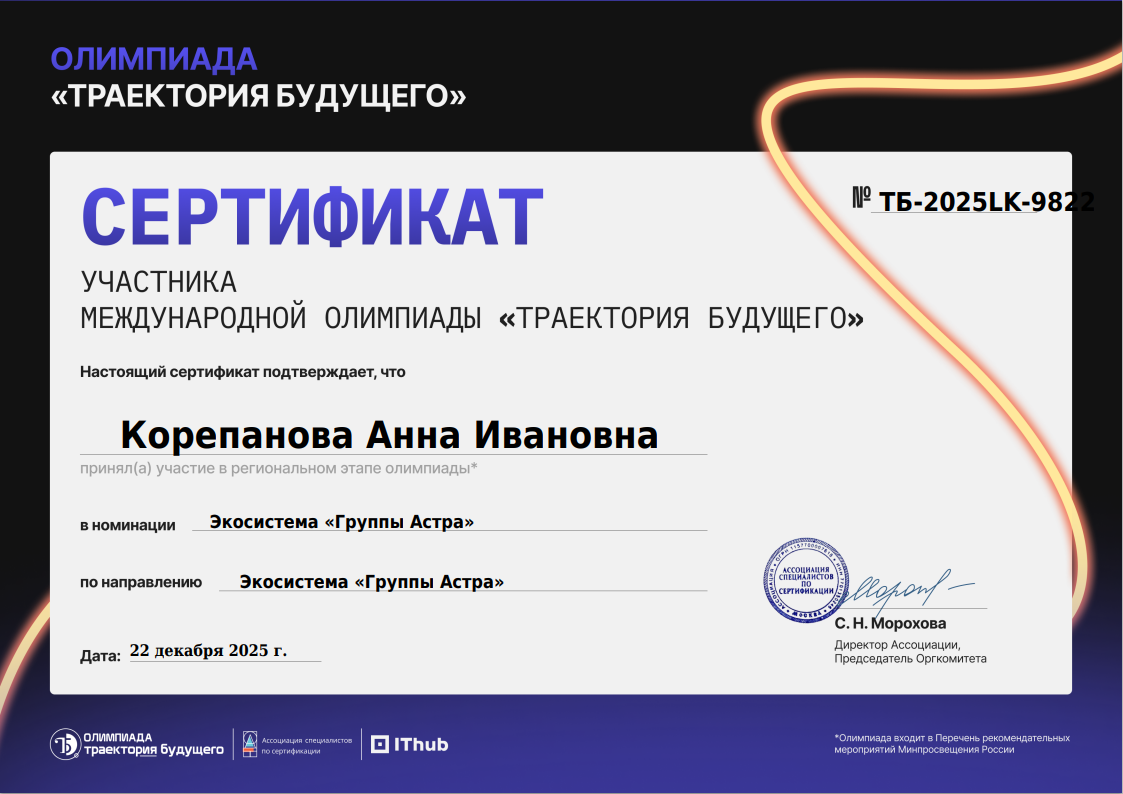


Рисунок 21 - Сертификат

Пример задания из адвент-челленджа для программистов (рис. 22):

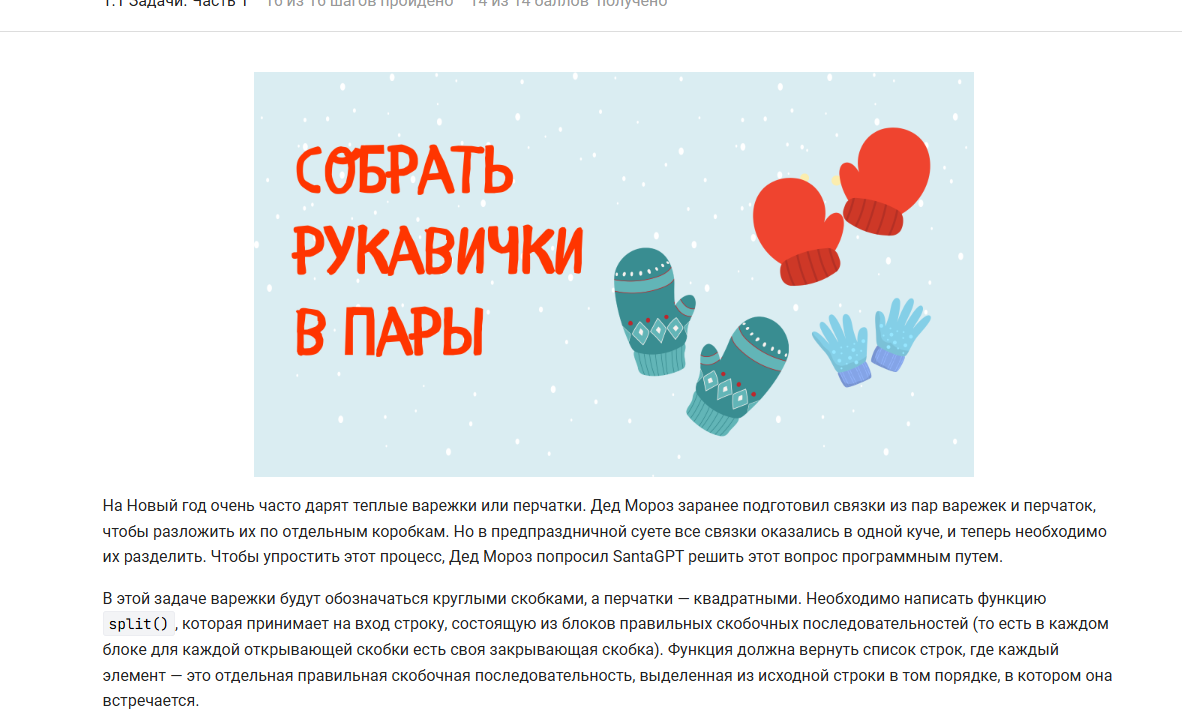


Рисунок 22 - Пример задания

Решение:

import sys

def split(s: str):

res = []

cur = []

stack = []

for c in s:

cur.append(c)

if c in '([':

stack.append(c)

elif c == ')':

stack.pop()

elif c == ']':

stack.pop()

if not stack:

res.append(''.join(cur))

cur.clear()

return res

if \_\_name\_\_ == '\_\_main\_\_':

data = sys.stdin.read().strip()

if data.startswith("print(split('") and data.endswith("'))"):

inner = data[13:-3]

print(split(inner))

else:

print(split(data))



Рисунок 23 - Открытка в честь успешного прохождения адвент-челленджа

# 

# ВЫВОДЫ

Прохождение учебной практики по модулю ПМ 02 «Осуществление интеграции программных модулей» завершено в полном объеме. Все поставленные цели достигнуты, что позволило закрепить профессиональные компетенции в области информационных технологий.

В перечень выполненных работ вошли: участие в олимпиаде «Траектория будущего» и адвент-челлендже, инсталляция среды MobaXterm, а также технические задачи по подключению периферийных модулей и сервоприводов Dynamixel к микрокомпьютеру nanoPi. Это способствовало глубокому освоению навыков интеграции программного обеспечения с аппаратными компонентами.

Особое внимание уделялось анализу информации, поиску оптимальных решений и командному взаимодействию. Опыт работы с технической документацией и современным инструментарием заложил прочный фундамент для дальнейшей профессиональной деятельности и решения реальных производственных задач в IT-сфере