СОДЕРЖАНИЕ

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.

1. Теоретические основы решения индивидуального задания

1.1. 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.

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

2.1. 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.

2.2. Скриншоты с результатами выполнения индивидуального задания

Установили программу MobaXterm (рис. 1).

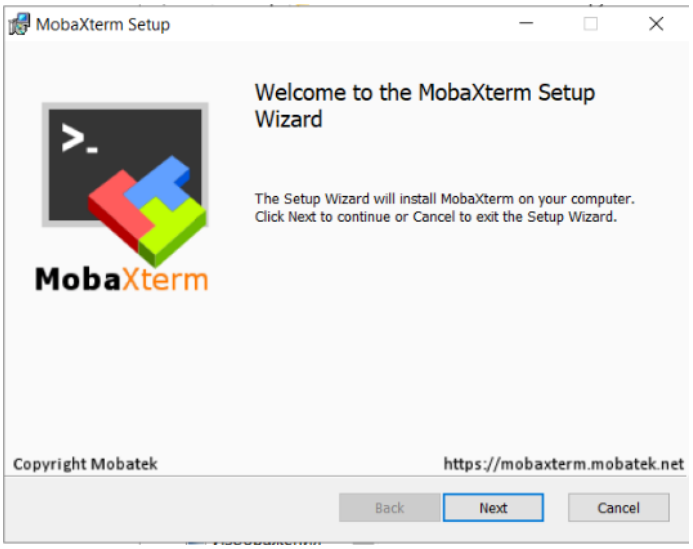


Рисунок 1 - Установка MobaXterm

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

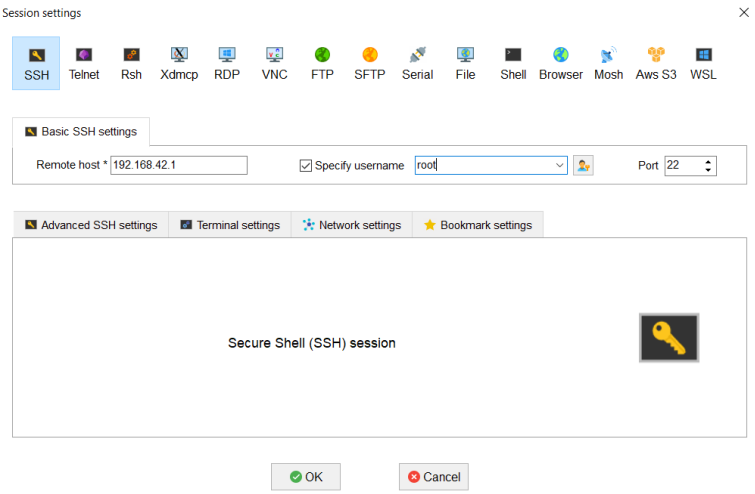


Рисунок 2 - Подключение к nanoPi

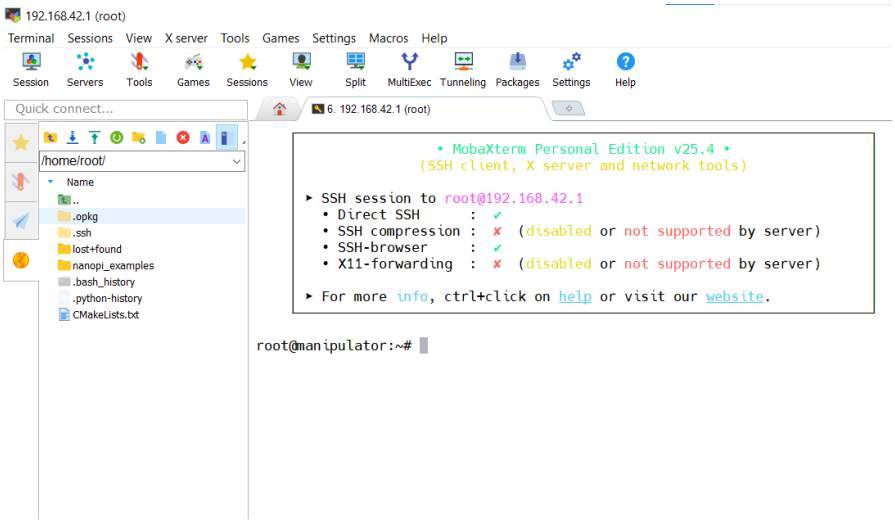


Рисунок 3 - Успешное подключение к nanoPi

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

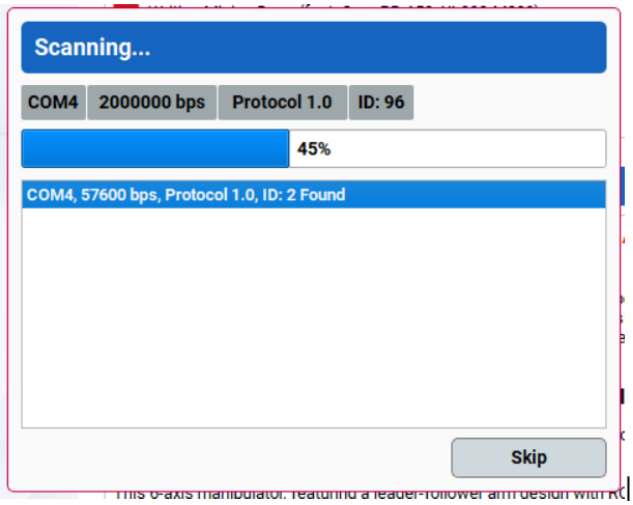


Рисунок 4 - Сканирование сервопривода

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

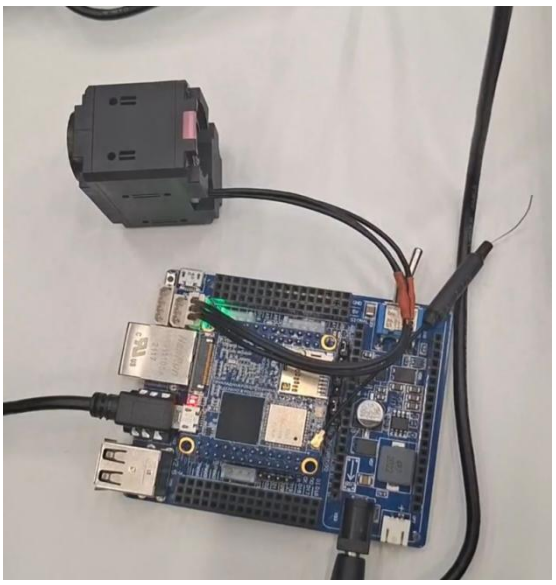


Рисунок 5 - Мигание светодиода на сервоприводе

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

```
root@manipulator:~/nanopi_examples/nanopi_examples/example_01_led/build# cmake ..
-- The C compiler identification is GNU 9.3.0
-- The CXX compiler identification is GNU 9.3.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /home/root/nanopi_examples/nanopi_examples/exa
root@manipulator:~/nanopi_examples/nanopi_examples/example_01_led/build#
```

Рисунок 6 - Результат команды cmake

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

```
root@manipulator:~/nanopi_examples/nanopi_examples/example_01_led/build# make -j2
-- Configuring done
-- Generating done
-- Build files have been written to: /home/root/nanopi_examples/nanopi_examples/example_01_led/build
make[1]: Entering directory '/home/root/nanopi_examples/nanopi_examples/example_01_led/build'
make[2]: Entering directory '/home/root/nanopi_examples/nanopi_examples/example_01_led/build'
Scanning dependencies of target example_01_led
make[2]: Leaving directory '/home/root/nanopi_examples/nanopi_examples/example_01_led/build'
make[2]: Entering directory '/home/root/nanopi_examples/nanopi_examples/example_01_led/build'
[ 14%] Building CXX object CMakeFiles/example_01_led.dir/example_01_led.cpp.o
[ 14%] Building CXX object CMakeFiles/example_01_led.dir/home/root/nanopi_examples/nanopi_examples/src/(
[ 21%] Building CXX object CMakeFiles/example_01_led.dir/home/root/nanopi_examples/nanopi_examples/src/(
[ 28%] Building CXX object CMakeFiles/example_01_led.dir/home/root/nanopi_examples/nanopi_examples/src/(
[ 35%] Building CXX object CMakeFiles/example_01_led.dir/home/root/nanopi_examples/nanopi_examples/src/(
[ 42%] Building CXX object CMakeFiles/example_01_led.dir/home/root/nanopi_examples/nanopi_examples/src/(
[ 50%] Building CXX object CMakeFiles/example_01_led.dir/home/root/nanopi_examples/nanopi_examples/src/(
[ 57%] Building CXX object CMakeFiles/example_01_led.dir/home/root/nanopi_examples/nanopi_examples/src/(
[ 64%] Building CXX object CMakeFiles/example_01_led.dir/home/root/nanopi_examples/nanopi_examples/src/(
[ 71%] Building CXX object CMakeFiles/example_01_led.dir/home/root/nanopi_examples/nanopi_examples/src/(
[ 78%] Building CXX object CMakeFiles/example_01_led.dir/home/root/nanopi_examples/nanopi_examples/src/(
[ 85%] Building CXX object CMakeFiles/example_01_led.dir/home/root/nanopi_examples/nanopi_examples/src/(
[ 92%] Building CXX object CMakeFiles/example_01_led.dir/home/root/nanopi_examples/nanopi_examples/src/(
[100%] Linking CXX executable example_01_led
make[2]: Leaving directory '/home/root/nanopi_examples/nanopi_examples/example_01_led/build'
[100%] Built target example_01_led
make[1]: Leaving directory '/home/root/nanopi_examples/nanopi_examples/example_01_led/build'
```

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).

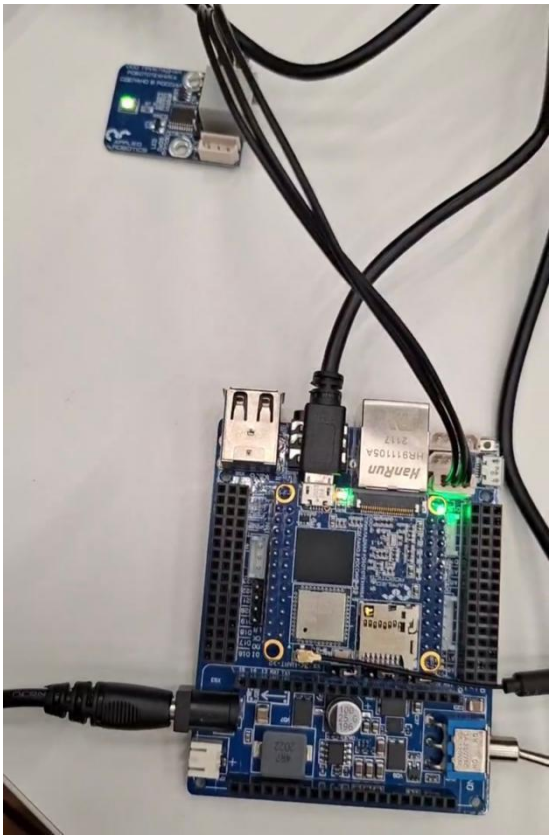


Рисунок 9 - Мигание модуля «Светодиод»

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).

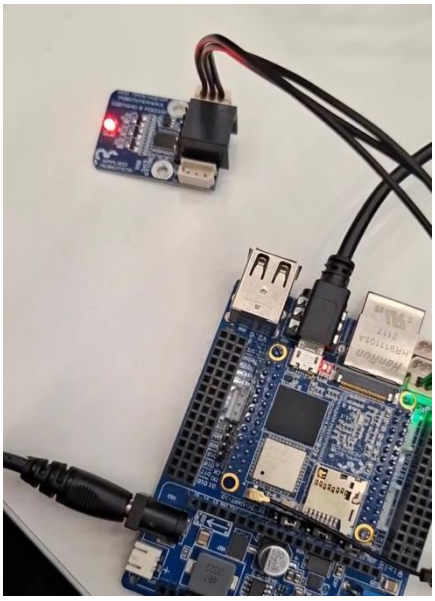


Рисунок 10 - Мигание модуля «Трёхцветный светодиод»

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.

```
root@manipulator:~/nanopi_examples/nanopi_e
=== CHANGE LED ID ===
Changing ID from 9 to 8
Protocol: 1, Baud rate: 57600

1. Checking device with old ID 9...
Device found (Model: 9)

WARNING: This will change ID permanently!
   Press Enter to continue...

2. Changing ID...
ID change command sent

3. Verifying...
   Old ID 9... no response
   New ID 8... no response

4. Testing LED...
   Turning ON...
   Turning OFF...
   Blinking...

ID change complete!
   Old: 9 > New: 8
root@manipulator:~/nanopi_examples/nanopi_e
```

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

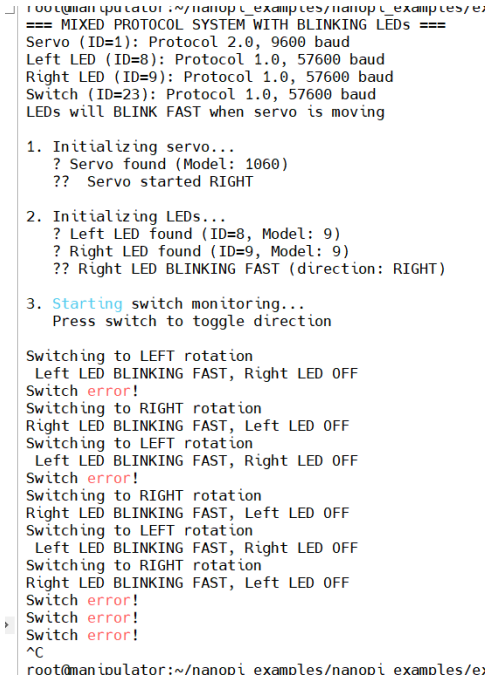


Рисунок 15 - Результат программы

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;
}
```

2.3. 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.

3. Выполнение общих заданий

3.1. 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.

3.2. 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).

3.3. Скриншоты с результатами решения задач

Blender:

Рисунок 16 - Кружка

Tinkercad:

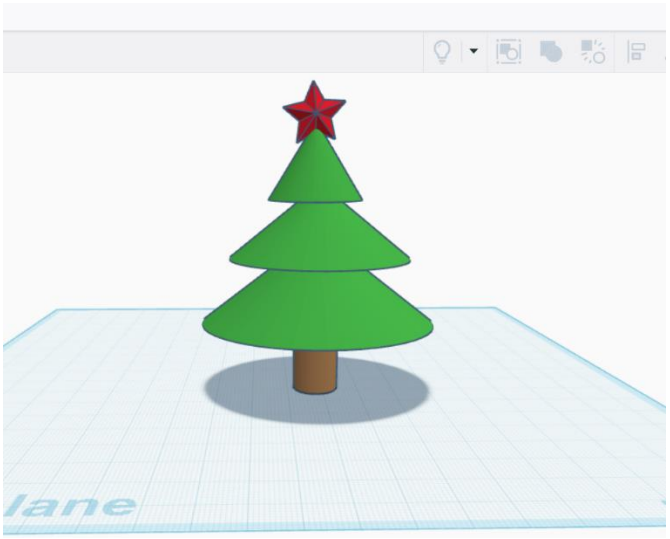


Рисунок 17 - Новогодняя ёлка

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

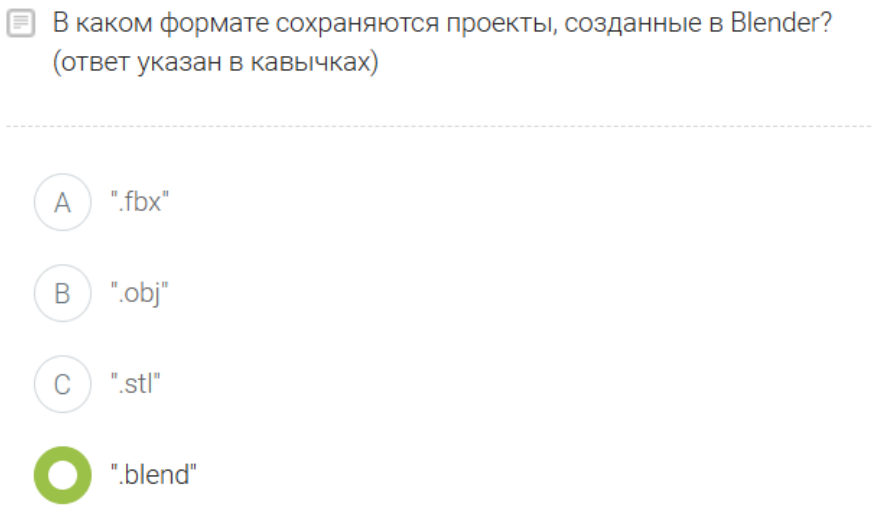


Рисунок 18 - Пример задания

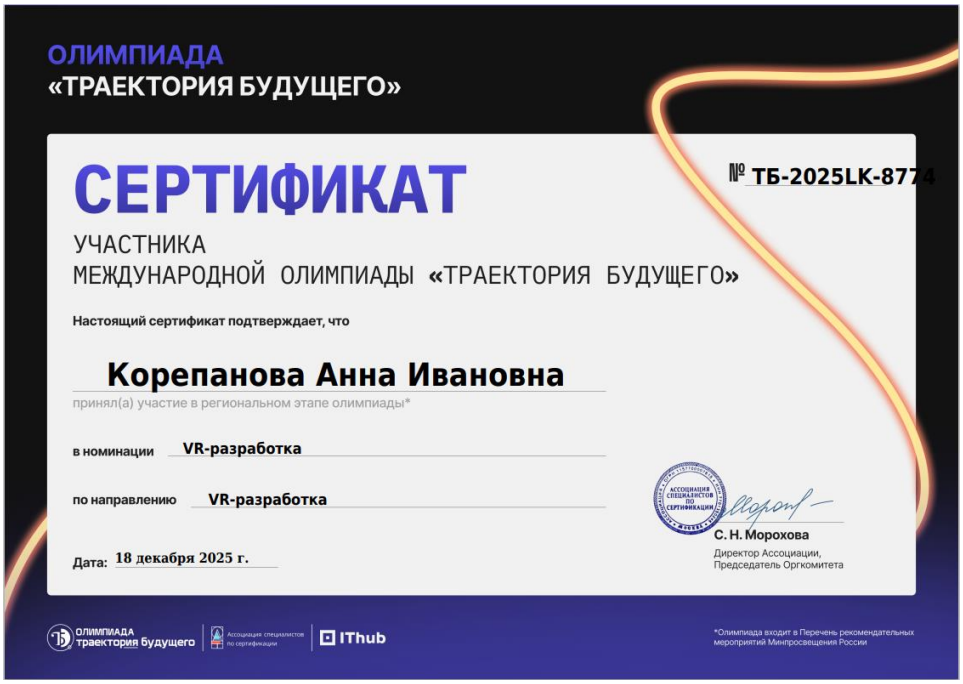


Рисунок 19 – Сертификат

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

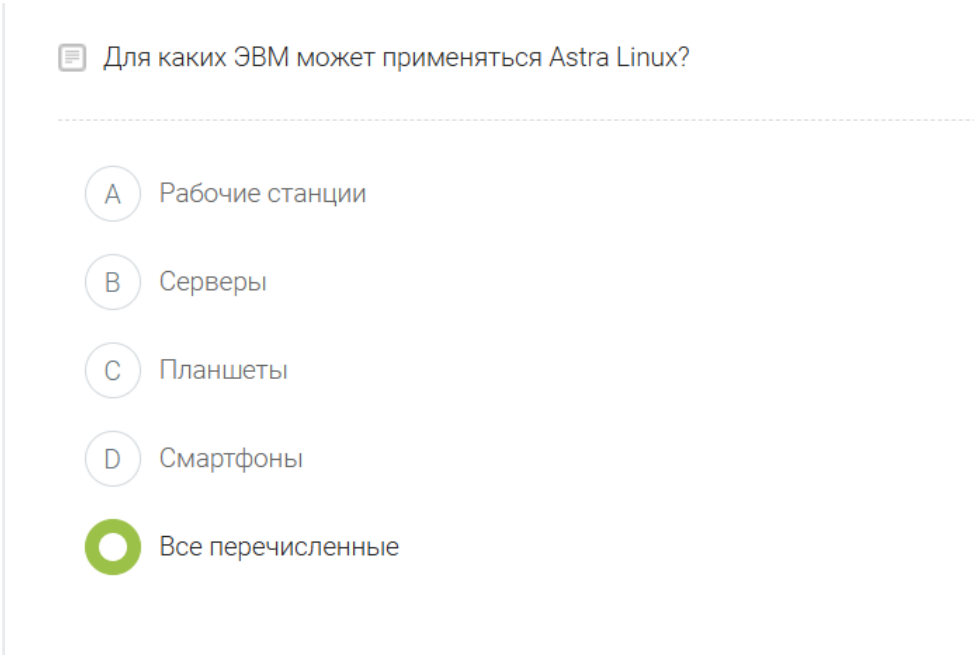


Рисунок 20 - Пример задания

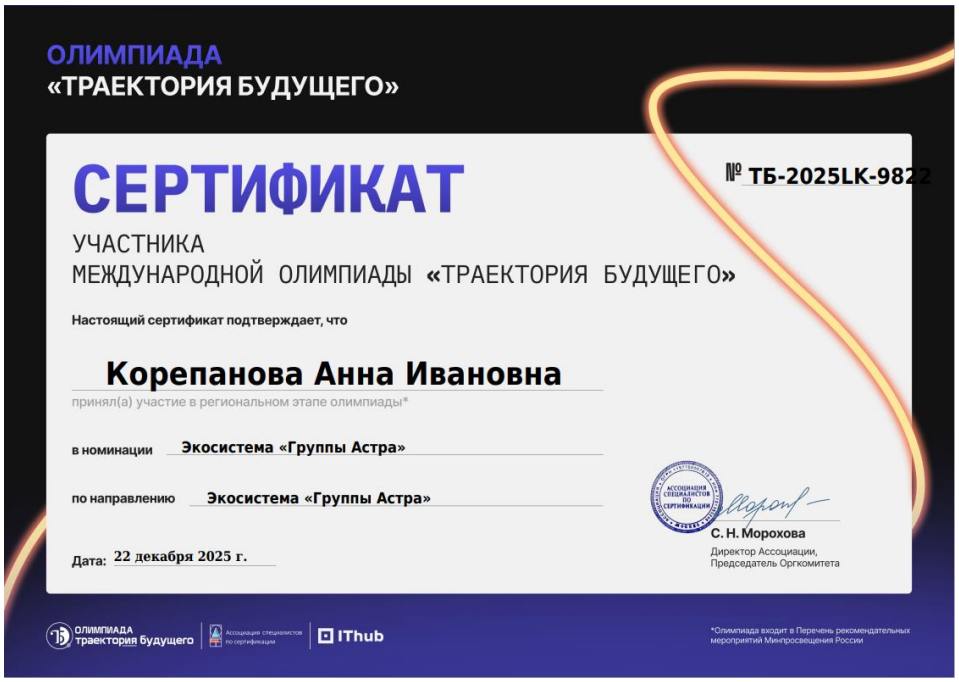
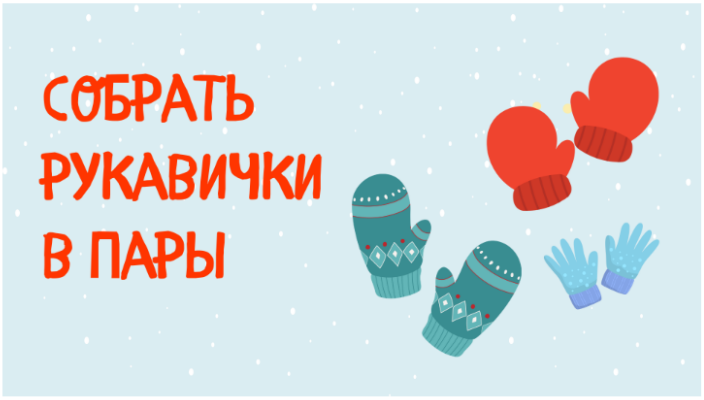


Рисунок 21 - Сертификат

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



На Новый год очень часто дарят теплые варежки или перчатки. Дед Мороз заранее подготовил связки из пар варежек и перчаток, чтобы разложить их по отдельным коробкам. Но в предпраздничной суете все связки оказались в одной куче, и теперь необходимо их разделить. Чтобы упростить этот процесс, Дед Мороз попросил SantaGPT решить этот вопрос программным путем.

В этой задаче варежки будут обозначаться круглыми скобками, а перчатки — квадратными. Необходимо написать функцию `split()`, которая принимает на вход строку, состоящую из блоков правильных скобочных последовательностей (то есть в каждом блоке для каждой открывающей скобки есть своя закрывающая скобка). Функция должна вернуть список строк, где каждый элемент — это отдельная правильная скобочная последовательность, выделенная из исходной строки в том порядке, в котором она встречается.

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