

Санкт-Петербургский политехнический университет Петра Великого  
Институт металлургии, машиностроения и транспорта  
Высшая школа автоматизации и робототехники

## Курсовая работа

Дисциплина: Программирование на языках высокого уровня  
Тема: создание ПО для наземной станции управления

Выполнил студент гр. 3331506/10401

Машинец Е.О.

Преподаватель

Ананьевский М.С.

«\_\_\_»\_\_\_\_\_2024 г.

Санкт-Петербург  
2024

## Содержание

1. Цель.....	3
2. Программные инструменты .....	3
3. Структура проекта.....	4
4. Реализация работы с HamLib (Windows).....	17
5. Настройка работы Gpredict .....	25
6. Заключение .....	39

## **1. Цель**

Задача заключается в создании ПО для поворотного устройства на базе raspberry pi и Arduino uno. Одноплатный компьютер и микроконтроллер (далее мк) связаны между собой через последовательный порт (Com-порт), из периферии у мк есть 2 индуктивных концевых датчика, работающих на эффекте Холла. Необходимо используя библиотеку HamLib и протокол связи Easyscomm написать ПО для мк, который будет получать данные об азимуте и элевации спутника от ПО Gpredict, обрабатывать эти данные в шаги шагового двигателя и управлять поворотным устройством.

## **2. Программные инструменты**

1. Hamlib - это библиотека программного обеспечения, которая предоставляет стандартизированный интерфейс для управления радиоаппаратурой. Это позволяет разработчикам программного обеспечения создавать приложения для управления радиоаппаратурой без необходимости написания кода, специфичного для каждого устройства. Hamlib поддерживает множество различных радиоаппаратур и протоколов связи, что делает его популярным инструментом среди радиолюбителей и профессиональных операторов.

2. Easyscomm - это протокол связи, используемый для управления радиоаппаратурой в радиолюбительских приложениях. Он предоставляет простой способ для программного обеспечения управлять настройками и параметрами радиоаппаратуры через последовательный порт компьютера. Easyscomm обычно используется в связке с программным обеспечением для управления трансиверами и другими радиоустройствами.

3. Gpredict - это программное обеспечение для прогнозирования орбит и отслеживания искусственных спутников Земли. Оно предоставляет пользовательский интерфейс для просмотра орбитальных параметров спутников, их текущего положения на небе и предполагаемого времени прохождения над вашим местоположением. Gpredict обычно используется радиолюбителями и спутниковыми операторами для планирования связей с искусственными спутниками и для управления антеннами во время прохождения спутников.

### 3. Структура проекта



Рисунок 1.1 – Рэндер сборочной единицы в SolidWorks2022

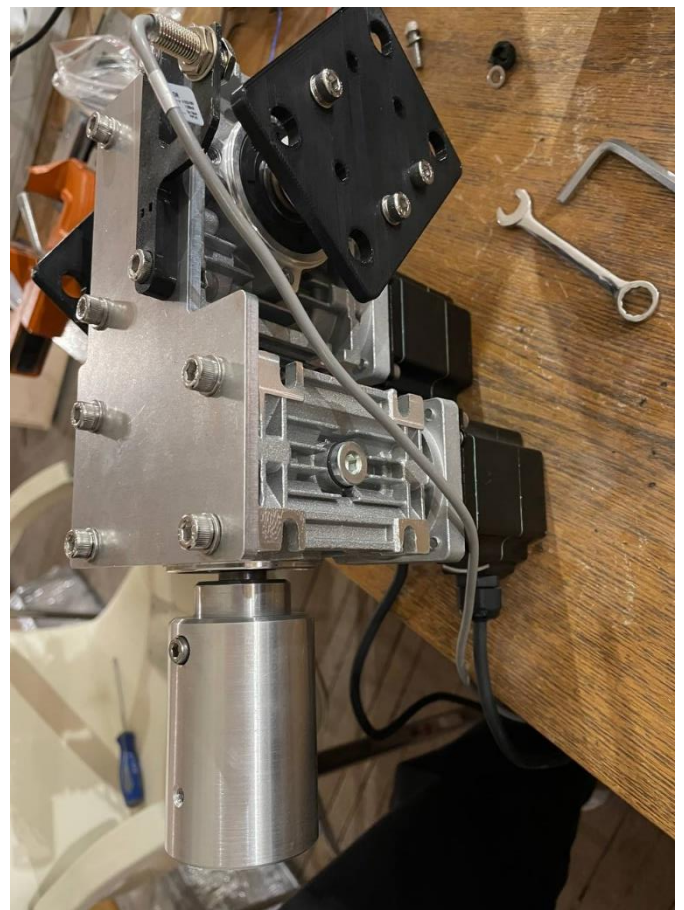
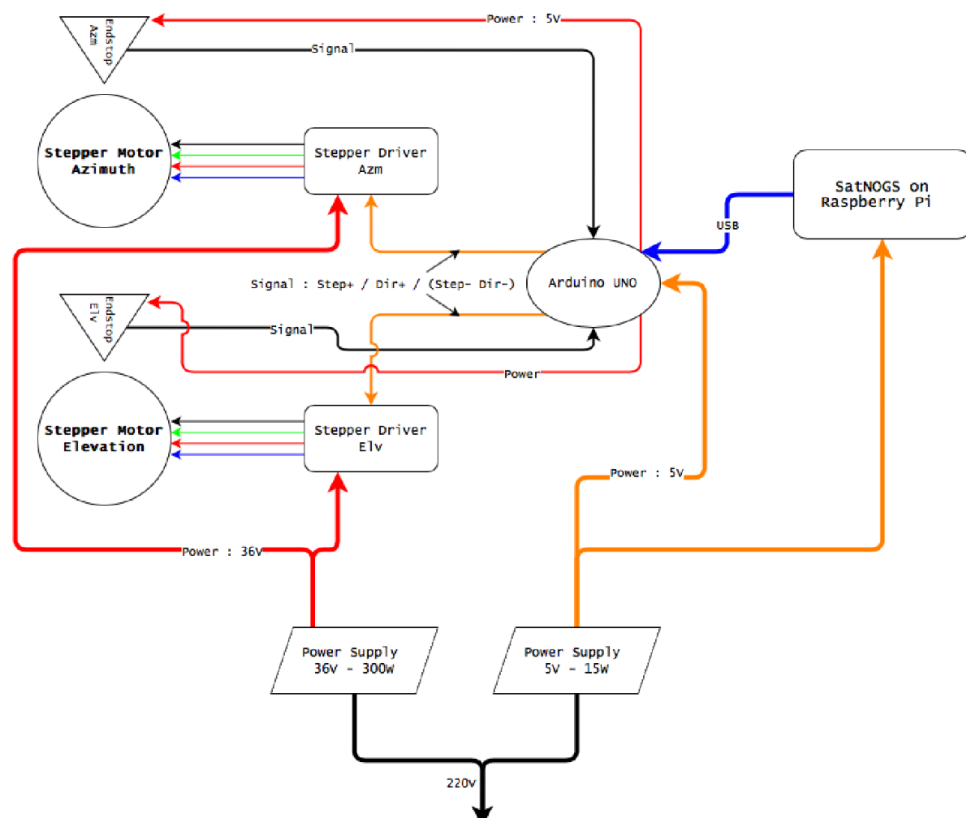


Рисунок 1.2 – Реальное изображение поворотного устройства

Электрическая схема проекта:



## Easycomm implementation:

### Easycomm implementation

- AZ, Azimuth, number - 1 decimal place [deg]
- EL, Elevation, number - 1 decimal place [deg]
- SA, Stop azimuth moving
- SE, Stop elevation moving
- RESET, Move to home position
- PARK, Move to park position
- IP, Read an input, number
  - Temperature = 0
  - SW1 = 1
  - SW2 = 2
  - Encoder1 = 3
  - Encoder2 = 4
  - Load of M1/AZ = 5
  - Load of M2/EL = 6
  - Speed of M1/AZ (DPS) = 7
  - Speed of M2/EL (DPS) = 8
- VE, Request Version
- GS, Get status register, number
  - idle = 1
  - moving = 2
  - pointing = 4
  - error = 8
- GE, Get error register, number
  - no\_error = 1
  - sensor\_error = 2
  - homing\_error = 4
  - motor\_error = 8
  - over\_temperature = 12
  - wdt\_error = 16
- VL, Velocity Left ,number [mdeg/s]
- VR, Velocity Right, number [mdeg/s]
- VU, Velocity Up, number [mdeg/s]
- VD, Velocity Down, number [mdeg/s]

## Конфигурация пинов:

```
#ifndef ROTATOR_PINS_H_
#define ROTATOR_PINS_H_
// #define M1IN1 10 ///< Motor 1 PWM pin
#define M1IN1 2 ///< Motor 1 PWM pin
#define M1IN2 5 ///< Motor 1 PWM pin
#define M2IN1 3 ///< Motor 2 PWM pin
#define M2IN2 6 ///< Motor 2 PWM pin
#define SW1 11 ///< Digital input, to read the status of end-stop for motor 1
#define SW2 9 ///< Digital input, to read the status of end-stop for motor 2
#define RS485_DIR 2 ///< Digital output, to set the direction of RS485 communication
#define SDA_PIN 3 ///< I2C data pin
#define SCL_PIN 4 ///< I2C clock pin
#endif /* ROTATOR_PINS_H_ */
```

## Код основной программы разделенной по блокам:

### 1. Инициализация

```
#define SAMPLE_TIME      0.1    ///< Control loop in s
#define RATIO            80     ///< Gear ratio of rotator gear box
#define MICROSTEP        2     ///< Set Microstep
#define MIN_PULSE_WIDTH  20     ///< In microsecond for AccelStepper
#define MAX_SPEED        1600   ///< In steps/s, consider the microstep
#define MAX_ACCELERATION 1600   ///< In steps/s^2, consider the microstep
#define SPR              400L   ///< Step Per Revolution, consider the microstep
#define MIN_M1_ANGLE     0      ///< Minimum angle of azimuth
#define MAX_M1_ANGLE     360    ///< Maximum angle of azimuth
#define MIN_M2_ANGLE     0      ///< Minimum angle of elevation
#define MAX_M2_ANGLE     180    ///< Maximum angle of elevation
#define DEFAULT_HOME_STATE HIGH ///< Change to LOW according to Home sensor
#define HOME_DELAY       12000  ///< Time for homing Deceleration in millisecond

#include <AccelStepper.h>
#include <Wire.h>
#include "easycomm.h"
#include "rotator_pins.h"
#include "endstop.h"
```

2. Создаются объекты шаговых двигателей и концевых выключателей.

```
uint32_t t_run = 0; // run time of uC
easycomm comm;
AccelStepper stepper_az(1, 3, 2);
AccelStepper stepper_el(1, 5, 4);
endstop switch_az(SW1, DEFAULT_HOME_STATE), switch_el(SW2, DEFAULT_HOME_STATE);
//wdt_timer wdt;

enum _rotator_error homing(int32_t seek_az, int32_t seek_el);
int32_t deg2step(float deg);
float step2deg(int32_t step);
```

3. Функция setup() :

Используются функции библиотеки AccelStepper, easycomm и функция самописной библиотеки endstop.h : init(). За функцией init() скрывается стандартная функция pinMode() библиотеки <Arduino.h>.

```
void setup() {
    // Homing switch
    switch_az.init();
    switch_el.init();

    // Serial Communication
    comm.easycomm_init();

    // Stepper Motor setup
    stepper_az.setEnablePin(MOTOR_EN);
    stepper_az.setPinsInverted(false, false, true);
    stepper_az.enableOutputs();
    stepper_az.setMaxSpeed(MAX_SPEED);
    stepper_az.setAcceleration(MAX_ACCELERATION);
    stepper_az.setMinPulseWidth(MIN_PULSE_WIDTH);
    stepper_el.setPinsInverted(false, false, true);
    stepper_el.enableOutputs();
    stepper_el.setMaxSpeed(MAX_SPEED);
    stepper_el.setAcceleration(MAX_ACCELERATION);
    stepper_el.setMinPulseWidth(MIN_PULSE_WIDTH);

    // Initialize WDT
    // wdt.watchdog_init();
}
```



```
void init() {
    pinMode(_pin, INPUT_PULLUP);
}
```

Фрагмент кода основной программы

```
void loop() {
    rotator.switch_az = switch_az.get_state();
    rotator.switch_el = switch_el.get_state();

    // Run easycomm implementation
    comm.easycomm_proc();

    // Get position of both axis
    control_az.input = step2deg(stepper_az.currentPosition());
    control_el.input = step2deg(stepper_el.currentPosition());
    // Check rotator status
    if (rotator.rotator_status != error) {
        if (rotator.homing_flag == false) {
            // Check home flag
            rotator.control_mode = position;
            // Homing
            rotator.rotator_error = homing(deg2step(-MAX M1 ANGLE),
                                           deg2step(-MAX M2 ANGLE));
            if (rotator.rotator_error == no_error) {
                // No error
                rotator.rotator_status = idle;
                rotator.homing_flag = true;
            } else {
                // Error
                rotator.rotator_status = error;
                rotator.rotator_error = homing_error;
            }
        } else {
            // Control Loop
            stepper_az.moveTo(deg2step(control_az.setpoint));
            stepper_el.moveTo(deg2step(control_el.setpoint));
            rotator.rotator_status = pointing;
            // Move azimuth and elevation motors
            stepper_az.run();
            stepper_el.run();
            // Idle rotator
            if (stepper_az.distanceToGo() == 0 && stepper_el.distanceToGo() == 0) {
                rotator.rotator_status = idle;
            }
        }
    }
}
```

```

    } else {
        // Error handler, stop motors and disable the motor driver
        stepper_az.stop();
        stepper_az.disableOutputs();
        stepper_el.stop();
        stepper_el.disableOutputs();
        if (rotator.rotator_error != homing_error) {
            // Reset error according to error value
            rotator.rotator_error = no_error;
            rotator.rotator_status = idle;
        }
    }
}

```

1. Получает состояние концевых выключателей для осей азимута и элевации.
2. Обрабатывает команды протокола связи Easyscomm.
3. Получает текущие позиции обеих осей и проверяет статус поворотного устройства.
4. Если статус поворотки не в ошибки, он либо запускает процесс "домашнего" позиционирования, либо входит в цикл управления для перемещения шаговиков.
5. Если произошла ошибка, останавливает шаговые двигатели и отключает их
6. Если ошибка не является ошибкой позиционирования, сбрасывает ошибку и статус поворотки в бездействие.

Этот участок кода отвечает за процесс "домашнего" позиционирования поворотки, то есть нахождение начальной позиции, используя концевые выключатели

```
enum _rotator_error homing(int32_t seek_az, int32_t seek_el) {
    bool isHome_az = false;
    bool isHome_el = false;

    // Move motors to "seek" position
    stepper_az.moveTo(seek_az);
    stepper_el.moveTo(seek_el);

    // Homing loop
    while (isHome_az == false || isHome_el == false) {
        // Update WDT
        // wdt.watchdog_reset();
        if (switch_az.get_state() == true && !isHome_az) {
            // Find azimuth home
            stepper_az.moveTo(stepper_az.currentPosition());
            isHome_az = true;
        }
        if (switch_el.get_state() == true && !isHome_el) {
            // Find elevation home
            stepper_el.moveTo(stepper_el.currentPosition());
            isHome_el = true;
        }
        // Check if the rotator goes out of limits or something goes wrong (in
        // mechanical)
        if ((stepper_az.distanceToGo() == 0 && !isHome_az) ||
            (stepper_el.distanceToGo() == 0 && !isHome_el)){
            return homing_error;
        }
        // Move motors to "seek" position
        stepper_az.run();
        stepper_el.run();
    }
}
```

```
// Delay to Decelerate and homing, to complete the movements
uint32_t time = millis();
while (millis() - time < HOME_DELAY) {
    // wdt.watchdog_reset();
    stepper_az.run();
    stepper_el.run();
}
// Set the home position and reset all critical control variables
stepper_az.setCurrentPosition(0);
stepper_el.setCurrentPosition(0);
control_az.setpoint = 0;
control_el.setpoint = 0;

return no_error;
```

Вот алгоритм процесса

### 1. Начало функции `homing()`:

- Принять количество шагов `seek_az` и `seek_el` для осей азимута и элевации соответственно.
- Инициализировать переменные `isHome_az` и `isHome_el` как `false`.

### 2. Перемещение шаговиков в "seek" позицию:

- Переместить моторы осей азимута и угла места в указанные позиции `seek_az` и `seek_el`.

### 3. Цикл "домашнего" позиционирования:

- Пока домашняя позиция для одной из осей не будет обнаружена (`isHome_az == false` или `isHome_el == false``):
- Проверить состояние концевых выключателей:
- Если концевик для оси азимута активирован и домашняя позиция для оси азимута еще не была обнаружена (`isHome_az == false``):
- Переместить шаговик оси азимута в его текущую позицию, считая это домашней позицией.
- Установить `isHome_az` в `true`.
- Если концевик для оси элевации активирован и домашняя позиция для оси угла места еще не была обнаружена (`isHome_el == false``):
- Переместить шаговик оси элевации в его текущую позицию, считая это домашней позицией.
- Установить `isHome_el` в `true`.
- Проверить, не вышла ли поворотная установка за пределы или не произошла ли механическая ошибка.
- Продолжить движение моторов в направлении "seek" позиции.

### 4. Задержка и сброс переменных:

- После обнаружения домашней позиции для обеих осей:
- Установить задержку.
- Установить текущую позицию обеих моторов в 0.
- Сбросить целевую позицию управления на 0.

### 5. Возвращение ошибки:

- Вернуть значение, указывающее на наличие или отсутствие ошибок в процессе позиционирования.

Этот участок кода представляет две функции: `deg2step()` и `step2deg()`, которые выполняют конвертацию между градусами и шагами для двигателей шагового типа.

**Функция `deg2step()`** принимает значение в градусах и конвертирует его в количество шагов, необходимых для прохождения указанного количества градусов.

Принимает значение в градусах в формате `float`. Возвращает количество шагов типа `int32_t`.

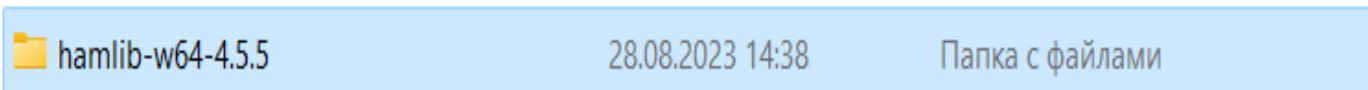
**Функция `step2deg()`** . действует наоборот

```
int32_t deg2step(float deg) {
    return (RATIO * SPR * deg / 360);
}


















float step2deg(int32_t step) {
    return (360.00 * step / (SPR * RATIO));
}
```

## 4. Реализация работы с HamLib (Windows)

Устанавливаем саму библиотеку



В папке bin лежат два файла с расширением .exe: rotctld.exe, ritctld.exe. Первый из них устанавливает общение с самим поворотным устройством, а второй устанавливает связь с антенной. Для наших нужд потребуется только rotctld.

 ampctl.exe	05.04.2023 15:36	Приложение	42 КБ
 ampctld.exe	05.04.2023 15:36	Приложение	50 КБ
 libgcc_s_sjlj-1.dll	12.06.2021 19:54	Расширение при...	1 158 КБ
 libhamlib-4.dll	05.04.2023 15:36	Расширение при...	9 963 КБ
 libusb-1.0.dll	11.12.2020 12:00	Расширение при...	278 КБ
 libwinpthread-1.dll	11.11.2018 22:47	Расширение при...	579 КБ
 pause	23.01.2024 22:07	Файл	1 КБ
 rigctl.exe	05.04.2023 15:36	Приложение	182 КБ
 rigctlcom.exe	05.04.2023 15:36	Приложение	193 КБ
 rigctld.exe	05.04.2023 15:36	Приложение	196 КБ
 rigmem.exe	05.04.2023 15:36	Приложение	34 КБ
 rigsmtr.exe	05.04.2023 15:36	Приложение	26 КБ
 rigswr.exe	05.04.2023 15:36	Приложение	26 КБ
 rigtestlibusb.exe	05.04.2023 15:36	Приложение	22 КБ
 rotctl.exe	05.04.2023 15:36	Приложение	55 КБ
 rotctld.bat	12.03.2024 19:32	Пакетный файл ...	1 КБ
 rotctld.exe	05.04.2023 15:36	Приложение	64 КБ

Далее необходимо написать исполняемый файл (batch)с текстовыми командами который будет задавать параметры соединения и устанавливать связь.

Создаем .bat файл с текстом

```
(rotctld -m 202 -r COM7 -s 9600 -T 127.0.0.1 -t 4533 -C timeout=500 -C retry=0 > pause)
```

- m название установки
- r ком порт
- s baudрейд
- T универсальный айпишник, выбирает айпи твоего компа
- t номер

Затем необходимо запустить его в командной строке с правами администратора, перед этим зайти в каталог с ним, пример : "C:\Program Files\hamlib-w64-4.5.5\bin\rotctld.bat").

Либо же другой вариант: .exe файл пкм – свойства-отправить-рабочий стол-пкм-свойства- и в графу объект, где путь, написать команду без названия исполняемого файла.

## 5. Настройка работы Gpredict

Скачайте и установите GPredict. Далее нам понадобится добавить поворотное устройство.

gpredict-win32-2.3.37

06.03.2024 16:52

Папка с файлами

Перейдите в меню Edit > Preferences > Interfaces > Rotators > Add New. Дайте ему имя. Например, Arduino. Порт - localhost 4533

Satellite	Az	El	Dir	Range	Next Event	Alt	Orbit
AO-73	229,64°	-35,41°	↑	8312	AOS: 2024/05/06 00:17:40	584	56473
AO-85	73,92°	-0,14°	↓	3056	AOS: 2024/05/05 18:48:54	686	37313
AO-91	215,79°	-72,87°	↓	12922	AOS: 2024/05/05 19:43:06	743	34968
FO-29	92,25°	-57,17°	↓	11946	AOS: 2024/05/05 22:05:48	1073	36873
ISS	188,92°	-43,43°	↑	9293	AOS: 2024/05/05 20:47:45	398	45212
SO-50	317,43°	-42,05°	↑	9480	AOS: 2024/05/05 17:37:31	650	15041

Это все для настройки. Далее мы можем протестировать систему.

Для теста нам нужно сделать несколько шагов:

Запустите .bat файл. В GPredict откройте модуль Антенна и нажмите "Включить". Вы должны увидеть, что пакеты команд отправляются на Arduino. Далее проверяем, что и Азимут, и Элевация работают правильно. Вручную управляем шаговыми двигателями через приложение. Но иногда это может не сработать с первого раза. В таком случае требуется просто перезапустить hamlib и Gpredict.

## **6. Заключение**

В ходе выполнения курсовой работы было успешно разработано программное обеспечение для поворотного устройства на базе Raspberry Pi и Arduino Uno. Еще было изучено много нового, приобретены навыки программирования микроконтроллеров, работы с различными типами датчиков, также освоена библиотека HamLib и протокол связи Easyscomm. Мы также научились разрабатывать и интегрировать различные компоненты в единое программное обеспечение для эффективного управления устройством.

Благодаря данной работе мы расширили свои знания в области электроники, программирования и автоматизации процессов, что позволит нам успешно применять и развивать эти навыки в будущих проектах.

В будущем проект будет дорабатываться, ближайшая задача – это избавление от персонального компьютера и “пересаживание” всего проекта на Debian (дистрибутив Linux).