Sterownik Silnika Anteny Radioteleskopu

Autor: Aleks Czarnecki

Spis treści

Wprowadzenie

Sterownik Silnika Anteny Radioteleskopu to kompletny system do sterowania anteną radioteleskopu z wykorzystaniem protokołu SPID. System oferuje zarówno **bibliotekę Python** do bezpośredniego użycia, jak i **API REST Server** z interfejsem webowym.

Główne funkcjonalności

- **Protokół SPID** — natywna obsługa protokołu SPID (Serial Protocol Interface Device)
- **API REST Server** — serwer HTTP z interfejsem webowym do zdalnego sterowania
- **Sterowanie pozycją anteny** — precyzyjne pozycjonowanie w azymutcie i elewacji
- **Kalkulator astronomiczny** — obliczanie pozycji Słońca, Księżyca, planet i gwiazd
- **Śledzenie obiektów** — automatyczne śledzenie obiektów astronomicznych
- **Monitorowanie w czasie rzeczywistym** — ciągłe śledzenie pozycji i stanu anteny
- **Bezpieczeństwo** — automatyczne sprawdzanie limitów mechanicznych i awaryjne zatrzymanie
- **Symulator** — tryb symulacji do testów bez fizycznego sprzętu
- **Interfejs webowy** — nowoczesny panel kontrolny dostępny przez przeglądarkę

Architektura systemu

System składa się z następujących głównych komponentów:

```
┌─────────────────────────────────────┐
│      REST API Server (FastAPI)      │ ← Interfejs HTTP + Web UI
├─────────────────────────────────────┤
│      AstronomicalCalculator         │ ← Obliczenia astronomiczne
├─────────────────────────────────────┤
│        AntennaController            │ ← Główny kontroler
├─────────────────────────────────────┤
```

```
┌─────────────────────────────────────┐
│         MotorDriver                 │ ← Abstrakcja sterownika
├─────────────────────────────────────┤
│  SPIDMotorDriver | SimulatedDriver  │ ← Konkretne implementacje
├─────────────────────────────────────┤
│      Komunikacja szeregowa          │ ← Warstwa fizyczna (SPID)
└─────────────────────────────────────┘
```

Struktura projektu

```
radioteleskop/
├── antenna_controller.py      # Główna biblioteka kontrolera
├── astronomic_calculator.py   # Kalkulator pozycji astronomicznych
├── emergency_stop.py          # System awaryjnego zatrzymania
├── api_server/                # REST API Server
│   ├── main.py               # Główny serwer FastAPI
│   ├── web_interface.html    # Interfejs webowy
│   ├── start_server.py       # Skrypt uruchamiający
│   ├── requirements.txt      # Zależności API
│   └── api_reference.md      # Dokumentacja API
├── examples/                  # Przykłady użycia
│   ├── basic_usage.py       # Podstawowe sterowanie
│   ├── advanced_usage.py    # Zaawansowane funkcje
│   └── api_examples.py      # Przykłady API
├── tests/                     # Testy jednostkowe
│   ├── tests.py            # Główne testy
│   └── test_spid_protocol.py # Testy protokołu
├── requirements.txt        # Zależności projektu
└── readme.md               # Ta dokumentacja
```

Instalacja i konfiguracja

Wymagania systemowe

- Python 3.8+
- Port szeregowy USB/RS232 (dla sprzętu SPID)
- System operacyjny: Linux, Windows, macOS

Instalacja zależności

Biblioteki Python:

```
pip install -r requirements.txt
```

Konfiguracja sprzętu

1. Podłącz kontroler SPID do portu USB/RS232

2. Sprawdź dostępne porty: `python -c "import serial.tools.list_ports; print([p.device for p in serial.tools.list_ports.comports()])"`
3. Skonfiguruj sterownik SPID na 115200 bps

API REST Server

Szybki start

```
# Uruchom serwer
python start_server.py
```

Dostęp do interfejsów

- **Interfejs webowy:** `http://localhost:8000/web_interface.html`
- **Dokumentacja API:** `http://localhost:8000/docs`
- **API Endpoint:** `http://localhost:8000`

Funkcjonalności interfejsu webowego

- Połączenie z anteną (sprzęt/symulator)
- Sterowanie pozycją anteny
- Monitorowanie statusu w czasie rzeczywistym
- Konfiguracja lokalizacji obserwatora
- Śledzenie obiektów astronomicznych
- Logi systemu z auto-scroll
- Awaryjne zatrzymanie (SPACJA lub przycisk)

Podstawowe użycie

Użycie przez bibliotekę Python

```
from antenna_controller import AntennaControllerFactory, Position
from astronomic_calculator import AstronomicalCalculator, ObserverLocation

# Utworzenie kontrolera
factory = AntennaControllerFactory()
controller = factory.create_spid("/dev/ttyUSB0", 115200)

# Inicjalizacja i sterowanie
controller.initialize()
controller.move_to_position(Position(azimuth=180.0, elevation=45.0))

# Pobieranie pozycji
current_pos = controller.current_position
print(f"Pozycja: Az {current_pos.azimuth}°, El {current_pos.elevation}°")
```

Użycie przez API REST

```
# Połączenie z symulatorem
curl -X POST http://localhost:8000/connect \
  -H "Content-Type: application/json" \
  -d '{"use_simulator": true}'

# Ustawienie pozycji
curl -X POST http://localhost:8000/position \
  -H "Content-Type: application/json" \
  -d '{"azimuth": 180, "elevation": 45}'
```

Protokół SPID

System obsługuje natywnie protokół SPID (Serial Protocol Interface Device):

Komendy podstawowe

- **Status:** `^C2` - pobiera aktualną pozycję
- **Ruch:** `PH180 PV045` - ustawia pozycję Az = 180°, El = 45°
- **Stop:** `SA SE` - zatrzymuje wszystkie osie

Konfiguracja komunikacji

- **Prędkość:** 115200 bps
- **Bity danych:** 8
- **Parzystość:** None
- **Bity stop:** 1
- **Kontrola przepływu:** None

Kalkulator astronomiczny

Obsługiwane obiekty

- **Słońce** — pozycja słoneczna
- **Księżyc** — fazy i pozycja księżyca
- **Planety** — Mercury, Venus, Mars, Jupiter, Saturn
- **Gwiazdy** — katalog gwiazd jasnych

Przykład użycia

```
# Konfiguracja obserwatora
location = ObserverLocation(
    latitude=52.40030,   # Poznań
    longitude=16.95508,
    elevation=75,        # metery n.p.m.
    name="Poznań"
)

# Obliczenie pozycji Słońca
```

```
calc = AstronomicalCalculator(location)
sun_pos = calc.calculate_object_position("Sun", ObjectType.SUN)
print(f"Słońce: Az {sun_pos.azimuth:.1f}°, El {sun_pos.elevation:.1f}°")
```

System bezpieczeństwa

Limity mechaniczne

- **Azymut:** 0° - 360° (konfigurowalny)
- **Elewacja:** -90° - +90° (konfigurowalny)
- **Prędkość:** Ograniczenia prędkości ruchu

Awaryjne zatrzymanie

- **Klawisz SPACJA** — w interfejsie webowym
- **Przycisk STOP** — w panelu sterowania
- **Automatyczne** — przy przekroczeniu limitów

Monitoring

- Ciągłe monitorowanie pozycji
- Kontrola komunikacji z kontrolerem
- Automatyczne wykrywanie błędów

Przykłady użycia

1. Podstawowe sterowanie anteną

```
from antenna_controller import AntennaControllerFactory, Position

# Połączenie z anteną
factory = AntennaControllerFactory()
controller = factory.create_spid("/dev/ttyUSB0")

# Ruch do pozycji
controller.move_to_position(Position(180.0, 45.0))

# Oczekiwanie na zakończenie ruchu
controller.wait_for_position()
print("Ruch zakończony")
```

2. Śledzenie Słońca

```
from astronomic_calculator import AstronomicalCalculator, ObserverLocation
from antenna_controller import AntennaControllerFactory

# Konfiguracja
location = ObserverLocation(52.40030, 16.95508, 75, "Poznań")
```

```
calc = AstronomicalCalculator(location)
controller = factory.create_spid("/dev/ttyUSB0")

# Śledzenie Słońca
controller.start_tracking("Sun", calc)
print("Rozpoczęto śledzenie Słońca")
```

3. Użycie symulatora

```
# Symulator do testów bez sprzętu
controller = factory.create_simulated()
controller.move_to_position(Position(90.0, 30.0))
```

Rozwiązywanie problemów

Częste problemy

Brak połączenia z portem szeregowym:

- Sprawdź, czy port jest podłączony
- Użyj `GET /ports` aby zobaczyć dostępne porty
- Sprawdź uprawnienia dostępu do portu (Linux/Mac)

Błąd "Failed to fetch":

- Sprawdź, czy serwer API jest uruchomiony
- Sprawdź adres URL (domyślnie localhost:8000)
- Sprawdź firewall i połączenie sieciowe

Problemy z pozycjonowaniem:

- Sprawdź limity mechaniczne anteny
- Sprawdź kalibrację kontrolera SPID
- Użyj symulatora do testów

Debugging

```
# Włączenie szczegółowych logów
import logging
logging.basicConfig(level=logging.DEBUG)

# Testowanie komunikacji SPID
from antenna_controller import test_spid_communication
test_spid_communication("/dev/ttyUSB0")
```

Logi systemu

Logi są dostępne:

- W konsoli serwera API
- W interfejsie webowym (sekcja "Log Systemu")
- W plikach log (jeśli skonfigurowane)

Projekt: Sterownik Silnika Anteny Radioteleskopu
Autor: Aleks Czarnecki
Protokół: SPID (Serial Protocol Interface Device)
Licencja: MIT