# NOAA Weather Satellite Reception & Tracking System

CONTENTS

# NOAA Weather Satellite Reception & Tracking System

## Overview

The satellite tracking system at satellite.stsgym.com provides comprehensive coverage for:

1. **NOAA Weather Satellites** - APT (Automatic Picture Transmission) signal reception on 137 MHz
2. **ADS-B Aircraft Tracking** - Real-time aircraft positions on 1090 MHz
3. **Pass Prediction** - Orbital calculations using TLE data
4. **Web Interface** - Live map visualization

## NOAA Weather Satellites

### Active Satellites

| Satellite | Frequency | Status | Image Type |
|---|---|---|---|
| NOAA-15 | 137.620 MHz | Active | APT |
| NOAA-18 | 137.9125 MHz | Active | APT |
| NOAA-19 | 137.100 MHz | Active | APT |

### APT Signal Details

- **Modulation**: AM (Amplitude Modulation)
- **Bandwidth**: ~40 kHz
- **Image Format**: 2 channels (Visible + Infrared)
- **Resolution**: 4 km/line
- **Scan Rate**: 2 lines/second
- **Signal Type**: Analog (APT - Automatic Picture Transmission)

### RTL-SDR Reception Setup

## Hardware Configuration

### RTL-SDR Setup on the SDR host

```
        # List RTL-SDR devices
        rtl_eeprom -d 0

        # Test reception
        rtl_fm -d 0 -f 137620000 -s 48000 -g 40 - | play -r 480
raw -e s16 -c 1 -V1 -

        # Record NOAA pass
        noaa-apt -d 0 -f 137620000 -g 40 -o recording.wav
```

### Antenna Recommendations

| Antenna Type | Gain | Best For |
|---|---|---|
| V-Dipole | 3-6 dB | Simple setup, good all-around |
| Turnstile | 6-8 dB | Better polar coverage |
| QFH | 8-12 dB | Best for LEO satellites |
| Quadrifilar Helix | 10-15 dB | Optimal for NOAA |

## Software Configuration

### noaa-recorder Service

```
        # /etc/systemd/system/noaa-recorder.service
        [Unit]
        Description=NOAA Satellite Recorder
        After=network.target

        [Service]
        Type=simple
        User=wez
        WorkingDirectory=/home/wez/scripts
        ExecStart=/usr/bin/python3 /home/wez/scripts/noaa_recor
        Restart=on-failure
        RestartSec=30
```

```
[Install]
WantedBy=multi-user.target
```

**noaa_recorder.py**

```
#!/usr/bin/env python3
"""
NOAA Automatic Recorder
Predicts passes and records NOAA APT signals
"""

import subprocess
from datetime import datetime, timedelta
from orbit_predictor.sources import EtcTLESource
from orbit_predictor.locations import Location

# Ground station location
GROUND_STATION = Location(
    latitude=38.0,  # Configure for your location
    longitude=-97.0,
    elevation_m=300
)

# Active NOAA satellites
NOAA_SATELLITES = {
    'NOAA-15': {'norad_id': 25338, 'frequency': 1376200
    'NOAA-18': {'norad_id': 28654, 'frequency': 1379125
    'NOAA-19': {'norad_id': 33591, 'frequency': 1371000
}

def predict_next_pass(satellite_name, hours=24):
    """Predict next satellite pass"""
    source = EtcTLESource.from_url('https://celestrak.o
NORAD/elements/weather.txt')
    predictor = source.get_predictor(satellite_name)

    passes = predictor.passes_over(GROUND_STATION,
limit=hours*60)
    return passes
```

```
        def record_pass(satellite_name, frequency, duration_sec
output_file):
            """Record NOAA APT signal"""
            cmd = [
                'rtl_fm',
                '-d', '0',           # Device index
                '-f', str(frequency), # Frequency in Hz
                '-s', '48000',        # Sample rate
                '-g', '40',           # Gain
                '-p', '1',            # PPM correction
                output_file
            ]

            subprocess.run(cmd, timeout=duration_seconds + 30)

        def decode_apt(wav_file, output_image):
            """Decode APT signal to image"""
            subprocess.run(['wxtoimg', '-t', 'auto', wav_file,
output_image])
```

# Database Models

## Satellite Model

```
        class Satellite(db.Model):
            """Satellite TLE data and metadata"""
            __tablename__ = 'satellites'

            id = db.Column(db.Integer, primary_key=True)
            norad_id = db.Column(db.Integer, unique=True,
nullable=False, index=True)
            name = db.Column(db.String(100), nullable=False)
            tle_line1 = db.Column(db.Text, nullable=False)
            tle_line2 = db.Column(db.Text, nullable=False)
            satellite_type = db.Column(db.String(50))  # 'weath
'ham', 'gps'
            frequency = db.Column(db.Float)  # MHz
            active = db.Column(db.Boolean, default=True)
```

```
            tle_updated_at = db.Column(db.DateTime,
default=datetime.utcnow)
```

### SatellitePass Model

```
        class SatellitePass(db.Model):
            """Predicted satellite pass over ground station"""
            __tablename__ = 'satellite_passes'

            id = db.Column(db.Integer, primary_key=True)
            satellite_id = db.Column(db.Integer,
db.ForeignKey('satellites.id'))
            aos_time = db.Column(db.DateTime, nullable=False)  #
Acquisition of signal
            los_time = db.Column(db.DateTime, nullable=False)
# Loss of signal
            max_elevation = db.Column(db.Float, nullable=False)
Degrees
            aos_azimuth = db.Column(db.Float)  # Rise direction
            los_azimuth = db.Column(db.Float)  # Set direction
            duration_minutes = db.Column(db.Float)
            pass_type = db.Column(db.String(20))  # 'ascending'
'descending'
```

### SatelliteReception Model

```
        class SatelliteReception(db.Model):
            """Recorded satellite reception"""
            __tablename__ = 'satellite_receptions'

            id = db.Column(db.Integer, primary_key=True)
            satellite_id = db.Column(db.Integer,
db.ForeignKey('satellites.id'))
            pass_id = db.Column(db.Integer,
db.ForeignKey('satellite_passes.id'))
            user_id = db.Column(db.Integer,
db.ForeignKey('users.id'))
```

```
    # Reception details
    start_time = db.Column(db.DateTime, nullable=False)
    end_time = db.Column(db.DateTime)
    frequency = db.Column(db.Float)  # MHz
    sample_rate = db.Column(db.Integer)  # Hz
    gain = db.Column(db.Float)  # dB

    # File paths
    raw_file = db.Column(db.String(255))  # .wav or .ra
    image_file = db.Column(db.String(255))  # .png or .

    # Processing status
    decoded = db.Column(db.Boolean, default=False)
    uploaded = db.Column(db.Boolean, default=False)

    # Quality metrics
    snr = db.Column(db.Float)  # Signal-to-noise ratio
    max_signal = db.Column(db.Float)  # Peak signal str
```

# API Endpoints

## Satellites

| Method | Endpoint | Description |
|---|---|---|
| GET | `/api/satellites/` | List all satellites |
| GET | `/api/satellites/<id>` | Get satellite details |
| GET | `/api/satellites/<norad_id>` | Get by NORAD ID |

## Passes

| Method | Endpoint | Description |
|---|---|---|
| GET | `/api/satellites/passes` | Get upcoming passes |
| GET | `/api/satellites/passes/predict` | Predict passes for satellite |
| GET | `/api/satellites/passes/current` | Get currently visible passes |

## Reception

| Method | Endpoint | Description |
|---|---|---|
| GET | | List receptions |

| Method | Endpoint | Description |
|---|---|---|
| | `/api/satellites/receptions` | |
| POST | `/api/satellites/receptions` | Upload new reception |
| GET | `/api/satellites/receptions/<id>` | Get reception details |
| GET | `/api/satellites/receptions/<id>/image` | Download decoded image |

### Example: Get Upcoming Passes

```
GET /api/satellites/passes?hours=24

{
  "passes": [
    {
      "id": 123,
      "satellite": "NOAA-19",
      "norad_id": 33591,
      "frequency": 137.1,
      "aos": "2024-03-15T08:30:00Z",
      "los": "2024-03-15T08:42:00Z",
      "max_elevation": 67.5,
      "duration_minutes": 12,
      "aos_azimuth": 45.2,
      "los_azimuth": 180.5,
      "is_visible": false
    }
  ]
}
```

### Example: Record Pass

```python
# Record NOAA-19 pass
import requests

# Schedule recording
response = requests.post(
    'https://satellite.stsgym.com/api/satellites/recept
```

```
        json={
            'satellite_id': 1,  # NOAA-19
            'pass_id': 123,
            'auto_record': True
        }
    )

    print(response.json())
    # {
    #   "reception_id": 456,
    #   "status": "scheduled",
    #   "start_time": "2024-03-15T08:30:00Z"
    # }
```

# Web Interface

## Live Map Features

- **Real-time satellite positions** - Updates every second
- **Ground track visualization** - Shows satellite path
- **Footprint display** - Coverage area
- **Pass prediction** - Upcoming passes table
- **Reception gallery** - Decoded NOAA images

## Pages

| Path | Description |
|---|---|
| `/` | Live satellite map |
| `/passes` | Pass prediction table |
| `/receptions` | Reception gallery |
| `/satellites` | Satellite list |
| `/settings` | Ground station config |

# Cron Jobs

## TLE Updates

```
    # Update TLE data daily at 3 AM UTC
    0 3 * * * /usr/bin/python3 /home/wez/scripts/update_tle
```

### Pass Prediction

```
        # Predict passes every 5 minutes
        */5 * * * * /usr/bin/python3 /home/wez/scripts/
predict_passes.py
```

### Temperature Monitoring

```
        # Record temperature every 5 minutes
        */5 * * * * /usr/bin/python3 /home/wez/scripts/
temp_monitor.py --record --alert

        # Sync temperature data every 10 minutes
        */10 * * * * /usr/bin/python3 /home/wez/scripts/
temp_monitor.py --sync --hours 1

        # Daily temperature report at 5 AM
        0 5 * * * /usr/bin/python3 /home/wez/scripts/temp_monit
--report
```

# Troubleshooting

## No Satellite Reception

1. **Check RTL-SDR device**

   ```
           lsusb | grep RTL
           rtl_test -t
   ```

2. **Verify frequency**

   ```
           rtl_fm -d 0 -f 137620000 -s 48000 -g 40 - | play -r
   48000 -t raw -e s16 -c 1 -
   ```

3. **Check antenna connection**

   - Outdoor antenna recommended
   - 137 MHz tuned (not generic wideband)

4. **Check pass timing**

```
python3 /home/wez/scripts/noaa_recorder.py --predic
```

### Weak Signal

- Increase RTL-SDR gain (try 40-50 dB)
- Use LNA (Low Noise Amplifier)
- Improve antenna placement
- Check coax cable for damage

### Image Quality Issues

- Ensure proper Doppler correction
- Use higher sample rate (48000 Hz minimum)
- Check for interference sources
- Verify antenna polarization

## Performance Optimization

### Database Indexes

```
-- Speed up pass queries
CREATE INDEX idx_passes_time ON satellite_passes (aos_t
los_time);

-- Speed up reception queries
CREATE INDEX idx_receptions_satellite
ON satellite_receptions (satellite_id, start_time);
```

### Caching

```
# Cache pass predictions
from functools import lru_cache
from datetime import datetime, timedelta

@lru_cache(maxsize=128)
def get_cached_passes(satellite_id: int, date: str):
    # Cache passes for 1 hour
    return SatellitePass.query.filter(
        SatellitePass.satellite_id == satellite_id,
```

```
                SatellitePass.aos_time > datetime.utcnow()
            ).all()
```

# References

- NOAA Satellite Information
- APT Signal Format
- wxtoimg Documentation
- noaa-apt Project
- Orbit Predictor

# Changelog

| Date | Version | Changes |
|---|---|---|
| 2024-03-01 | 1.0.0 | Initial satellite tracking |
| 2024-03-05 | 1.1.0 | Added NOAA APT reception |
| 2024-03-10 | 1.2.0 | Integrated with stsgym.com |
| 2024-03-15 | 1.3.0 | Added pass prediction |
| 2024-03-20 | 1.4.0 | Added reception gallery |
| 2024-03-22 | 2.0.0 | Documentation release |