National Centre for Radio Astrophysics

Internal Technical Report

Web Page Development for servo configuration parameters

Debasmita Roy

imdebasmita81@gmail.com

Objective : To develop a web interface for accessing servo configuration parameters from the local antennas as per the user requirement.

Document History:

Revision	Date	Modification / Change
Ver . 1	09.05.2024	Initial release
Ver 1.1	16.05.2024	UI Enhancements
Ver 1.2	23.05.2024	Feature Expansion
Ver 1.3	03.06.2024	Document and Testing
Ver 2	10.06.2024	User Feedback Integration

PROJECT OVERVIEW

The project aims to develop a comprehensive web interface for accessing and configuring servo parameters. This web application will provide faculty members with a user-friendly platform to manage and modify servo settings, ensuring enhanced usability, security, and real-time feedback.

Development phases:

Phase 1 : Initial development focusing on basic UI and backend API for viewing and modifying servo parameters.

Phase 2 : Enhancements to UI for improved user experience and additional features like real-time feedback and user authentication.

Phase 3 : Performance optimizations, security enhancements, and comprehensive logging.

Phase 4 : Implementation of advanced features such as undo functionality and exporting/importing settings.

Phase 5 : Extensive testing, documentation, and final deployment.

Benefits:

1. Streamlined process for configuring servo parameters, reducing the time and effort required by faculty members.
2. Real-time feedback and validation ensure accurate configuration changes.
3. Strong authentication and authorization mechanisms protect sensitive data.

ACKNOWLEDGEMENTS

I would like to express my profound gratitude to **Thiyagarajan B** Sir for giving me the opportunity to work under the aegis of GMRT Observatory. I would like to thank him for the consent support and co – operation provided by him during the entire project. His constant encouragement enabled me to carry on the project quite successfully.

I would also like to thank my mentor **Ms. Priyanka Priyadarshini** for always supporting and encouraging me and the constant belief in me that I can carry out my project here and also helping me to get this opportunity.

I would also like to thank the entire GMRT staff who have directly or indirectly helped me in completing the project and for their cooperation provided to me without which it would have been difficult to get accustomed to the conditions here.

(**DEBASMITA ROY**)

2024(SP20240328144152848)

SUIIT – BURLA, ODISHA

Table of Contents

INTRODUCTION

With advancements in technology, modern antennas often employ servos for precise alignment and configuration. Managing these servos, especially in remote or distributed systems, requires a user-friendly and efficient interface. This report outlines the development of a web-based platform for accessing and configuring servo parameters associated with antennas.

In the realm of modern communication and broadcasting, antennas play a critical role in ensuring effective signal transmission and reception. The precise alignment and configuration of these antennas are crucial for optimal performance. Servos, which are specialized motors capable of exact positioning, are often employed to adjust and control antenna parameters such as azimuth and elevation.

This project involves developing a web page for accessing servo configuration parameters from the local antennas as per the user requirements . The interface prioritizes responsiveness across devices , minimizing latency for seamless configuration changes .

Traditionally, configuring these servos has required direct physical interaction or the use of complex and specialized software. However, with the advent of web technologies, there is a growing trend towards developing web-based interfaces that allow remote access and configuration of these systems.

This initiative not only enhances the efficiency of antenna management but also reduces the need for physical presence and specialized equipment, thereby cutting costs and increasing operational flexibility. As the demand for more sophisticated and accessible control systems grows, the development of web interfaces for servo configuration represents a significant step forward in the field of antenna technology.

OBJECTIVES

The development of a web page for accessing and configuring servo parameters on antennas aims to achieve the following key objectives:

1. **Remote Access and Control:**
   Enable users to access and adjust servo configurations from any location with internet connectivity, eliminating the need for physical presence at the antenna site.

2. **User – Friendly Interface:**
   Design an intuitive and easy-to-use web interface that simplifies the process of configuring servo parameters, making it accessible to users with varying levels of technical expertise.

3. **Real – Time Monitoring:**
   Provide real-time feedback on servo status and antenna parameters, allowing users to monitor and adjust settings dynamically for optimal performance.

4. **Enhanced Security:**
   Implement robust security measures, including user authentication, authorization, and data encryption, to protect the system from unauthorized access and potential cyber threats.

5. **Scalability and Flexibility:**
   Develop a scalable solution that can handle multiple antennas and servo systems, with the flexibility to integrate with other existing systems and technologies.

6. **Cross – Platform Compatibility:**
   Ensure the web interface is compatible with various devices and browsers, providing a consistent user experience across desktops, tablets, and smartphones.

7. **Performance Optimization:**
   Optimize the web application for fast loading times and minimal latency, ensuring a responsive and smooth user experience even in real-time operations.

8. **Support for Future Enhancements:**
   Design the system with future enhancements in mind, allowing for easy updates and the addition of new features as technology and user needs evolve.

These objectives aim to create a comprehensive, secure, and efficient web-based solution for managing and configuring servo parameters on antennas, significantly improving operational capabilities and user experience.

REQUIREMENT ANALYSIS

FUNCTIONAL REQUIREMENTS:

1. User Authentication and Authorization :

Authentication – Implement secure login mechanisms (e. g.., username / password , two factor authentication).

Authorization – Role based access control (RBAC) to restrict access and actions based on user roles.

2. Servo Configuration Interface:

View Current Parameters – Display current servo settings such as position , speed , and operational status.

Modify Parameters - Provide forms and controls to update servo configurations , including position adjustments , speed settings , and calibration options .

3. Real – Time Monitoring:

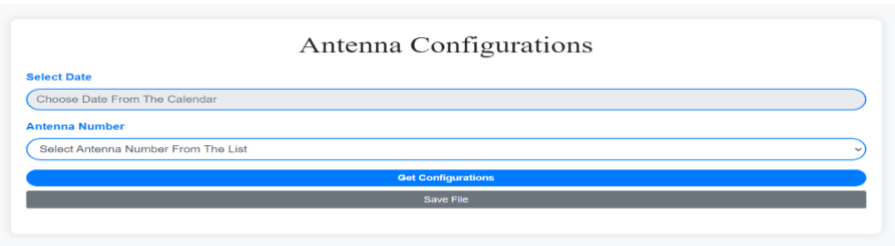
Status Updates – Display real time status of servos and antennas , including alerts for any operational issues.

Live Feedback – Provide instant feedback on parameter changes and servo movements.

4. User Interface:

Dashboard - Centralized view of all connected antennas and their statuses.

Detailed View – Pages or sections dedicated to individual antennas and their servo configuration.



NON – FUNCTIONAL REQUIREMENTS:

1. Performance:

Responsive Design – Ensure the web application performs well across various devices and screen sizes.

Speed – Optimize for fast loading times and low latency , especially for real – time updates.

2. Scalability:

Horizontal Scaling – Ability to handle an increasing number of antennas and servo configurations without performance degradation.

Database Scalability – Efficient handling of large volumes of configuration data and logs.

3. Usability:

Intuitive Interface – Design with a focus on ease of use , ensuring users can easily navigate and perform tasks without extensive training.

Accessibility – Comply with accessibility standards to ensure the application is usable by individuals with disabilities.

4. Reliability:

High Availability – Ensure the system is robust and can operate with minimal downtime.

Error Handling – Graceful handling of errors with informative messages to guide the user in troubleshooting.

5. Maintainability:

Modular Design – Use modular coding practices to facilitate easy updates and maintenance.

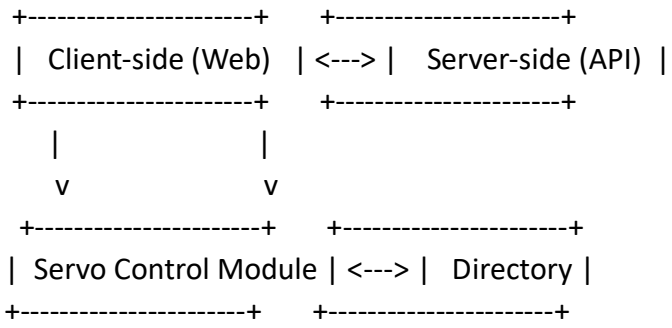
Documentation – Comprehensive documentation for both users and developers.

SYSTEM DESIGN

The system design for the web page development to access and configure servo parameters of antennas involves defining the architecture , data flow , user interface , and integration points .

1. Architecture:

The architecture comprises three main components : the client – side , the server – side , and the servo control module.



2. Component Breakdown:

Client side (Web Interface) –

HTML 5, CSS3 , JavaScript , React or angular.

Displays an overview of all connected antennas and their configuration.

Interface for viewing and modifying parameters.

View and generate reports on configuration changes and system performance

Server side (API) –

Python with Flask.

Secure login and role – based access control.

Restful services for client – server communication.

Processing and validation of configuration changes.

3. Data Flow:

Users interact with the web interface to view or change servo configurations.

Client – side sends HTTP requests to the server – side API.

Server – side processes the requests , performs business logic , and interacts with the database.

Server – side communicates with the Servo Control Module to apply changes or retrieve status.

Server – side sends responses back to the client – side , which updates the UI accordingly .

4. User Interface Design :

Overview of all antennas and their current statuses.
Visual indicators for operational status
Quick access to detailed views of individual antennas.
Real time feedback on changes.
Save and cancel buttons to commit or revert changes.
Table view of historical changes with filtering and sorting options.

5. Save button :

A button is displayed on the website.
Files for the specific dates can be saved in the .txt format.
Any user can store the files according to his comfortability.

6. Faulty Cases:

If any of the 5 configurations are missing then it will display as “File not found”.
If any of the specific date and antenna number directories are not present then it will show as “ The configurations for the specific date or antenna number is not found”.

By thoroughly designing the system architecture , user interface , data flow and component breakdown , the development of a web page for accessing and configuring servo parameters on antennas can be efficiently executed , ensuring a robust , secure , and user – friendly application.

IMPLEMENTATION

The implementation of the web page is setting up the development environment , developing the front – end and back – end components , integrating with servo control module , and ensuring security and performance optimization .

Node.js for JavaScript development .
Python and pip for Python back – end .
Git for version control.

pip install -r requirements.txt

Flask==2.1.2
SQL Alchemy==1.4.31
Flask-SQL Alchemy==2.5.1
requests==2.27.1
Jinja2==3.1.1
Werk zeug==2.1.2
Gun icorn==20.1.0
Flask-JWT-Extended==4.4.2
Flask-Cors==3.0.10

pip list

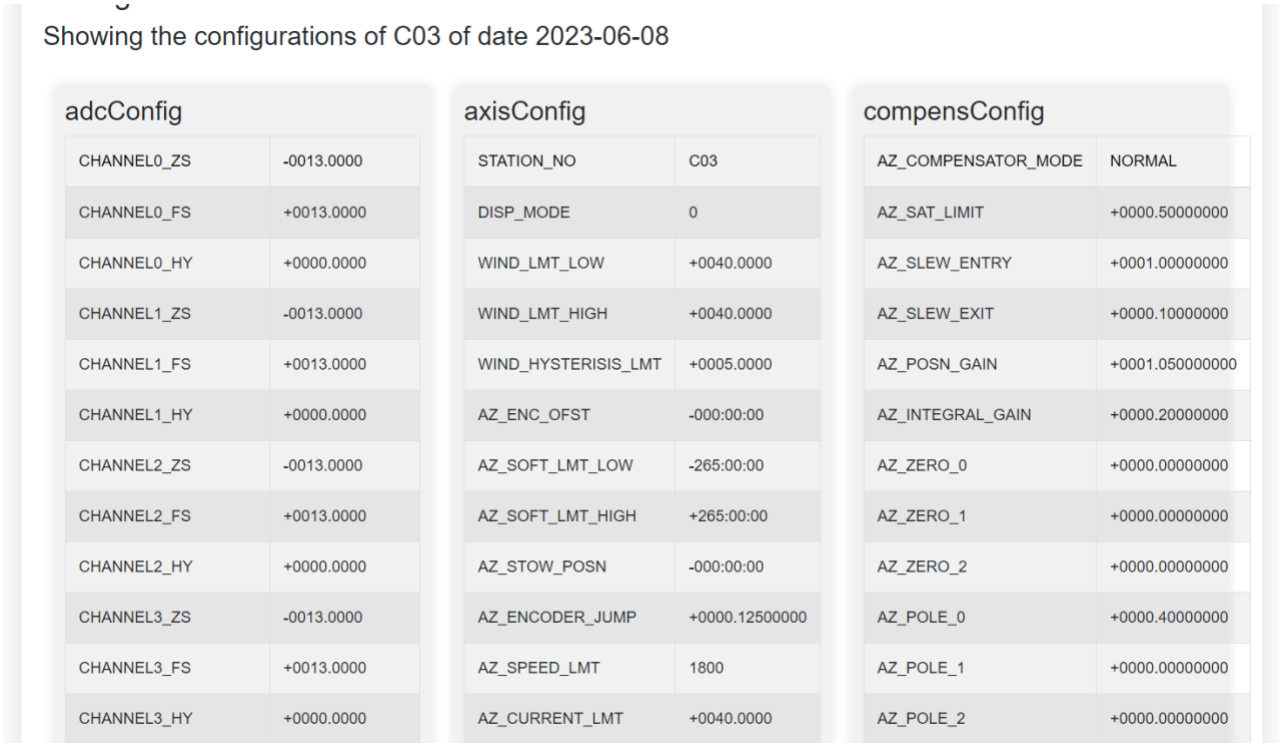
Creating directories for the front – end , back – end and configuration files. There will be a display for an overview of all connected antennas and provide forms to view and modify servo parameters . There are table views for historical data and logs . It is provided with set up routing for navigating between different pages.

import React, { use State } from 'react';
import './AntennaConfig.css';

const Antenna Config = () => {
  const [servo Position, set Servo Position] = use State(0);
  const [servo Speed, set Servo Speed] = use State(0);
  const [status, set Status] = use State('Idle');

```
const handle Position Change = (event) => {
  set Servo Position(event .target .value);
};

const handle Speed Change = (event) => {
  set Servo Speed(event. target. value);
};

const handle Submit = (event) => {
  event. Prevent Default();
  // Here you would normally make an API call to update the servo settings
  Set Status('Updating...');
  Set Timeout(() => set Status('Idle'), 2000); // Simulate a network request
};

return (
  <div class Name="antenna-config">
    <h1>Antenna Configuration</h1>
    <form on Submit={handle Submit}>
      <div class Name="form-group">
        <label html For="position">Servo Position:</label>
        <input
          type="number"
          id="position"
          value={servo Position}
          on Change={handle Position Change}
          min="0"
          max="180"
        />
      </div>
      <div class Name="form-group">
        <label html For="speed">Servo Speed:</label>
        <input
          type="number"
          id="speed"
          value={servo Speed}
          on Change={handle Speed Change}
          min="0"
          max="100"
        />
      </div>
```

```
      <button type="submit">Update Configuration</button>                    10
    </form>
    <p>Status: {status}</p>
   </div>
  );
};

export default Antenna Config;
```



Showing the configurations of C03 of date 2023-06-08

adcConfig

CHANNEL0_ZS	-0013.0000
CHANNEL0_FS	+0013.0000
CHANNEL0_HY	+0000.0000
CHANNEL1_ZS	-0013.0000
CHANNEL1_FS	+0013.0000
CHANNEL1_HY	+0000.0000
CHANNEL2_ZS	-0013.0000
CHANNEL2_FS	+0013.0000
CHANNEL2_HY	+0000.0000
CHANNEL3_ZS	-0013.0000
CHANNEL3_FS	+0013.0000
CHANNEL3_HY	+0000.0000

axisConfig

STATION_NO	C03
DISP_MODE	0
WIND_LMT_LOW	+0040.0000
WIND_LMT_HIGH	+0040.0000
WIND_HYSTERISIS_LMT	+0005.0000
AZ_ENC_OFST	-000:00:00
AZ_SOFT_LMT_LOW	-265:00:00
AZ_SOFT_LMT_HIGH	+265:00:00
AZ_STOW_POSN	-000:00:00
AZ_ENCODER_JUMP	+0000.12500000
AZ_SPEED_LMT	1800
AZ_CURRENT_LMT	+0040.0000

compensConfig

AZ_COMPENSATOR_MODE	NORMAL
AZ_SAT_LIMIT	+0000.50000000
AZ_SLEW_ENTRY	+0001.00000000
AZ_SLEW_EXIT	+0000.10000000
AZ_POSN_GAIN	+0001.050000000
AZ_INTEGRAL_GAIN	+0000.20000000
AZ_ZERO_0	+0000.00000000
AZ_ZERO_1	+0000.00000000
AZ_ZERO_2	+0000.00000000
AZ_POLE_0	+0000.40000000
AZ_POLE_1	+0000.00000000
AZ_POLE_2	+0000.00000000

APPENDIX

```
# const express = require('express');
const helmet = require('helmet');
const rate Limit = require('express-rate-limit');
const jwt = require('json web token');

const app = express();

// Middleware for security headers
app. use(helmet());

// Rate limiting
const limiter = rate Limit({
  window Ms: 15 * 60 * 1000, // 15 minutes
  max: 100, // limit each IP to 100 requests per windowMs
});
app. use(limiter);

// Input validation example (using express-validator)
const { body, validation Result } = require('express-validator');

app. post('/update-config', [
  body('position').is Int({ min: 0, max: 180 }).with Message('Position must be
between 0 and 180'),
  body('speed').is Int({ min: 0, max: 100 }).with Message('Speed must be
between 0 and 100'),
], (req, res) => {
  const errors = validation Result(req);
  if (!errors.is Empty()) {
    return res. status(400).json({ errors: errors. array() });
  }
  // Handle the valid input
```

```
});

// JWT Authentication
const authenticate JWT = (req, res, next) => {
  const token = req. header('Authorization').replace('Bearer ', '');
  if (!token) {
    return res. status(401).send('Access Denied');
  }
  try {
    const verified = jwt. verify(token, process. env. JWT_SECRET);
    req. user = verified;
    next();
  } catch (err) {
    res. status(400).send('Invalid Token');
  }
};

app. use('/secure-endpoint', authenticate JWT, (req, res) => {
  res. send('This is a secure endpoint');
});

app. listen(3000, () => {
  console.log('Server is running on port 3000');
});
```

Ensuring security is paramount when developing a web page that interacts with servo configuration on antennas. Use of multi – factor authentication to ensure that only authorized users can access the system by defining roles and permission to ensure users can only access functions and data they are authorized to . Encryption is done between all data transmitted between the client and server using SSL/TLS by using HTTPS to secure HTTP requests. Implementing session timeouts and automatic logouts after periods of inactivity.

By incorporating these security considerations and practices , we can significantly enhance the security posture of the web application , ensuring that the antenna configuration system is protected against various threats and vulnerabilities.

FUTURE ENHANCEMENTS

As technology evolves and user needs grow , there are several future enhancements that can be made . These enhancements aim to improve usability , functionality , security , and overall system performance.

Improved Mobile Experience - Further optimize the interface for mobile devices to ensure ease of use on smaller screens.

Adaptive Layouts – Implement adaptive layouts that adjust based on user preferences and device capabilities.

Interactive Dashboard – Creating interactive dashboards that allow users to customize the data they want to see.

Scalable Architecture – Design the system architecture to handle increased loads and scale efficiently with demand.

Efficient Data Handling – Optimize the handling and transmission of real – time data to reduce latency and improve performance.

Multi – User Collaboration – Enable real time collaboration features where multiple users can work on servo configurations simultaneously.

Version Control – Implement version control for configurations settings , allowing users to track changes , revert to previous states , and audit modifications.

CONCLUSION

Web page development for accessing servo configuration on antenna parameters is a critical endeavor in the realm of modern technology, particularly for applications requiring precise and remote control over antenna systems. This development combines user-friendly interfaces, robust backend systems, and stringent security measures to ensure reliable and efficient operation. The core objectives of such a web page are to facilitate ease of access, precise control, and real-time monitoring of servo configurations, thereby enhancing the functionality and usability of antenna systems. Through careful requirement analysis, detailed system design, and meticulous implementation, developers can create a platform that meets the needs of users and stakeholders. Security considerations are paramount, ensuring that the system is protected against unauthorized access, data breaches, and other threats. Implementing strong authentication, authorization, encryption, and regular security audits are essential practices to safeguard the system.

Testing and validation are integral to the development process, ensuring that the system operates as intended and meets all specified requirements. By adopting comprehensive testing strategies, developers can identify and resolve issues early, ensuring a robust and reliable application. Looking forward, future enhancements such as advanced UI improvements, real-time capabilities, enhanced security measures, integration with IoT and edge computing, advanced data analytics, and improved collaboration tools will further augment the system's capabilities. These enhancements will not only improve the user experience but also provide more powerful and flexible control over antenna configurations.

REFERENCES

1. https://exploringjs.com/impatient-js/.

2. Haver beke , Marijn. "Eloquent JavaScript: A Modern Introduction to Programming." 3rd Edition. Accessed June 23, 2022. https://eloquentjavascript.net/.

3. "Using Flask with AJAX." Real Python. Accessed June 4, 2023. https://realpython.com/flask-by-example-implementing-a-redis-task-queue/.

4. "Flask Documentation (2.1.x)." Flask. Accessed July 4, 2021. https://flask.palletsprojects.com/en/2.1.x/.

5. "Web Development Basics." MDN Web Docs. Accessed Jan 3, 2024. https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web.

6. "Main Concepts." React. Accessed April 4, 2024. https://reactjs.org/docs/getting-started.html.