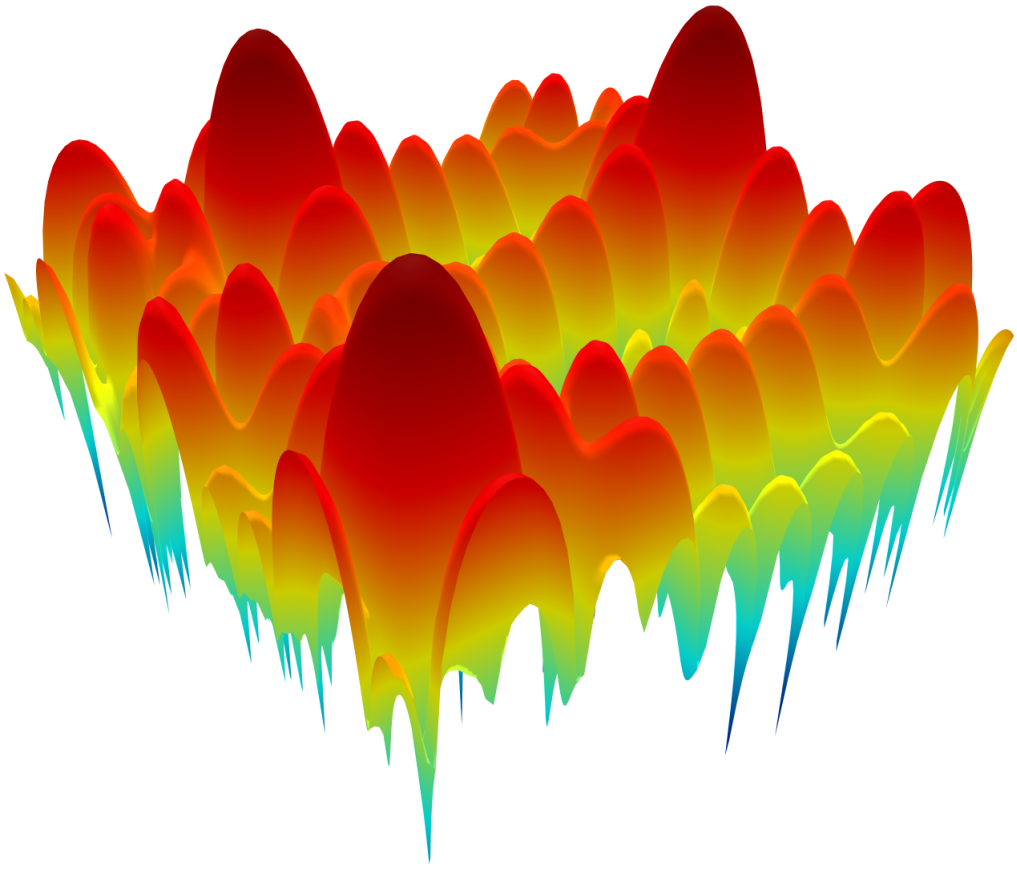
Array Processing ToolBox

gr-array



Author: Rahul Rajeev Pillai

Date: April 7, 2025

Abstract

This is a proposal for a project that seeks to create a tool for designing and analyzing sensor arrays in GNU Radio. The tool will feature a user-friendly GUI to configure array parameters, visualize patterns, analyze performance metrics, and experiment with various built-in array processing algorithms. It will also generate GNU Radio (GR) flow graphs based on the user-defined parameters, enabling seamless transition from design to real-time testing withing the GNU Radio. This project strives to make advanced array processing more accessible, fostering exploration and innovation without relying on proprietary software.

Contents

List of Figures

List of Tables

List of Abbreviations

DoA Direction-of-Arrival

GR GNU Radio

GRC GNU Radio Companion

GUI Graphical User Interface

LCMV Linearly Constrained Minimum Variance

MUSIC MUltiple Signal Classification

MVDR Minimum Variance Distortionless Response

OOT Out-Of-Tree

UCA Uniform Circular Array

UI User Interface

ULA Uniform Linear Array

URA Uniform Rectangular Array

Chapter 1

Introduction

1.1 Project Overview

This project aims to develop an open-source, interactive tool for designing and analyzing sensor arrays within GNU Radio (GR), it is inspired by MATLAB's Phased Array System Toolbox (more specifically: Sensor Array Analyzer). This GUI-based tool should allow users to configure various array parameters, visualize the array-factor in different plots, view performance characteristics and play around with popular Beamforming and Direction-of-Arrival (DoA) estimation algorithms.

Once the design is complete, the tool should generate a compatible GR flow graph, configured with the chosen array settings. This functionality bridges the gap between array design and real-time testing, empowering researchers and engineers to iterate and test their designs seamlessly in GR.

By creating this tool, the project seeks to provide the open-source community with a powerful, accessible alternative to proprietary software, lowering the barrier to entry for advanced array processing research and development.

1.2 Motivation

When I first started as an intern, I was tasked with developing adaptive array algorithms. At the time, my understanding of signal processing was very limited, and working with multi-dimensional signal processing algorithms was really a daunting task. The first major hurdle was grasping the mathematical foundations and intuition behind the algorithms. After extensive effort, I was able to develop a good understanding of most concepts.

The next challenge was implementation and verification. While searching for simulation tools, I found that MATLAB was the primary option, but being paid-license based, I was unable to use it. This left me with no choice but to develop everything from scratch. At that time, my programming experience was limited, making the process even more difficult. I looked online for array processing algorithm implementations to get a sense of how they were implemented. Although there were only a handful of resources available, I managed to find a couple of GitHub repositories and PySDR, which proved to be invaluable. They helped me with signal generation, algorithm implementation, pattern visualization, and even understanding the underlying theory. However, they primarily focused on Uniform Linear Array (ULA), making it quite tricky to do the same for planar arrays. Through extensive study of textbooks and careful adaptation of available

resources, I was eventually able to extend the simulation to planar array.

While I gained a lot from developing everything from scratch, it was also time-intensive and prone to errors. Developing the core algorithm is one aspect, but implementing signal generation, visualization tools and all the other supporting tools, just adds significant overhead. This project aims to simplify these aspects, allowing researchers and engineers to focus on algorithm development rather than spending excessive time on auxiliary tasks.

1.3 Objective

- **OBJ-1**: To develop an interactive graphical tool that enables a user to configure various parameters of a sensor array, visualize the beam pattern of the array using different plotting methods, and analyze its performance metrics.
- **OBJ-2**: The graphical tool should be able to simulate different array processing algorithms for various user-defined array configurations and parameters and provide immediate results.
- **OBJ-3**: The tool should include a mechanism to convert the user-defined configurations into a valid GNU Radio compatible flowgraph.

1.4 Milestones

- **MS-1**: Planning
  - Connect with mentors and the community.
  - Iron out details of the project.
  - Setup work environment.
- **MS-2**: Graphical User Interface (GUI).
  - Set up the GUI framework.
  - Implement user-friendly input controls.
  - Ensure smooth navigation and parameter configuration.
- **MS-3**: Signal Processing Library.
  - Implement a function for steering vector generation based on different parameters for a wide range of array configurations.
  - Develop a library of array processing algorithms in Beamforming and DoA estimation.
  - Create functions for measuring arrays and the algorithms performance.
- **MS-4**: Development of Out-Of-Tree (OOT) modules and flow graph generation.
  - Create a set of customizable GNU Radio Companion (GRC) blocks designed to meet diverse user settings.
  - Implement a functionality within the application window that allows the user to convert their settings to a GRC flowgraph in GNU Radio.

- **MS-5**: Integration.
  - Develop a library for generating various types of plots.
  - Integrate the signal processing library, plotting library, GUI and the flowgraph generator with the application window.
  - Enable the window to be launched directly from within GRC.

Table 1.1: Objective - Milestone mapping.

	MS-1	MS-2	MS-3	MS-4	MS-5
OBJ-1	1	1	1		1
OBJ-2	1		1		1
OBJ-3	1			1	1

1.5 Document Structure

The remainder of this proposal is structured as follows:

- **Chapter 2**: Project Systems
  Describes the high-level system architecture. It details how the project is divided into multiple systems components such as the application window, signal processing library, and OOT module and are structured to work together.
- **Chapter 3**: Application Window
  Explains the graphical interface for the project. It gives an idea on how the User Interface (UI) will look like and explains the different panes of the window. It also goes into the different types of plotting methods that are available.
- **Chapter 4**: Signal Processing Library
  Details the development of core signal processing functions including steering vector generation, beamforming techniques, and DoA estimation algorithms. It also covers performance evaluation tools integrated within the library.
- **Chapter 5**: Out-Of-Tree (OOT) Modules
  Describes the implementation of custom GNU Radio OOT blocks that expose array processing functionalities in the GRC environment. This chapter includes information on how these blocks are designed, parameterized, and linked with the GUI for automated flow graph generation.

Chapter 2

Project Systems

This entire project can be divided into 3 main components.

- Application Window
- Signal Processing Library
- OOT modules

2.1 Application Window

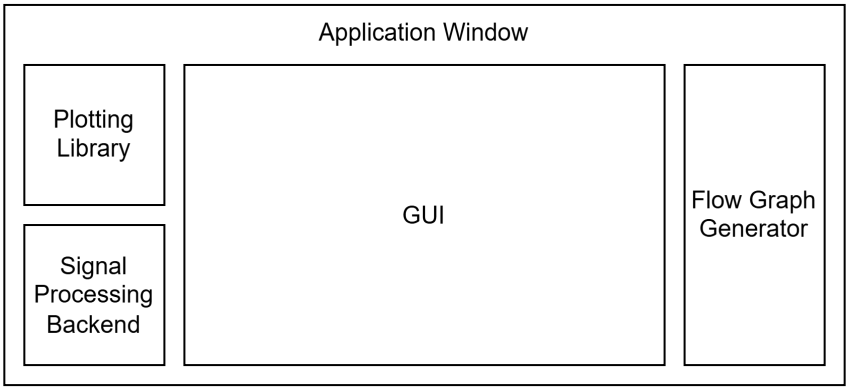


Figure 2.1: Application Window and its sub-systems.

The application window acts as the main interface for the user, bringing together array configuration, visualization, simulation, and flowgraph generation all in a single environment.

It will feature an intuitive GUI with controls like sliders, dropdowns, and text fields to allow users to easily set array parameters, visualize beam patterns etc. The window will also offer simulation capabilities for array processing algorithms and support automatic generation of flow graphs from user-defined settings that can be used in GNU Radio.

To streamline the workflow, the application window can be launched directly from within GRC, ensuring smooth integration with the existing GNU Radio framework.

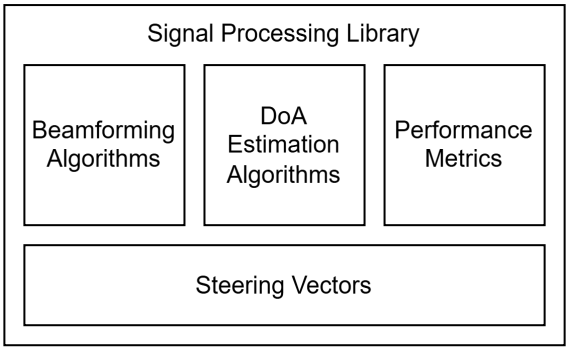


Figure 2.2: Signal Processing Library and its sub-systems.

2.2 Signal Processing Library

The signal processing library forms the core component of the project, providing the computational backbone for array analysis and algorithm simulation.

The library will include functions for steering vector generation, supporting a variety of array geometries and parameter configurations. This ensures flexibility and adaptability across different sensor array setups. Additionally, it will have in-built functions of various Beamforming and DoA estimation, allowing users to evaluate and compare different techniques within the same framework.

To support meaningful analysis, the library will also provide tools to measure performance metrics of both the arrays and the implemented algorithms. These tools will be tightly integrated with the GUI and plotting modules, enabling quick simulation and visualization.

2.3 OOT Modules

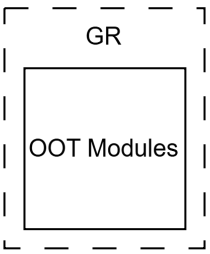


Figure 2.3: OOT Modules.

As part of the system's integration with GNU Radio, a set of custom OOT modules will be developed to extend the platform's native functionality and provide a seamless interface between the graphical tool (application window) and GNU Radio Companion (GRC). These OOT modules are designed to support user-defined array configurations and processing algorithms by using them as reusable blocks within the GRC environment.

Each block will encapsulate a specific array processing function, such as steering vector generation, beamforming, or direction-of-arrival (DoA) estimation, allowing users to incorporate these capabilities directly into their GNU Radio flowgraphs. The blocks will be implemented in C++ or Python, with proper XML code to ensure compatibility with the GRC graphical interface.

This modular design enables users to prototype and validate their array processing pipelines quickly, while maintaining the flexibility to modify and expand the processing chain as needed.

2.4 System Integration

The integration phase brings together all the major components of the project into a functional system. At the heart of this integration lies the seamless interaction between the application window, signal processing library, and the custom OOT modules from GR.

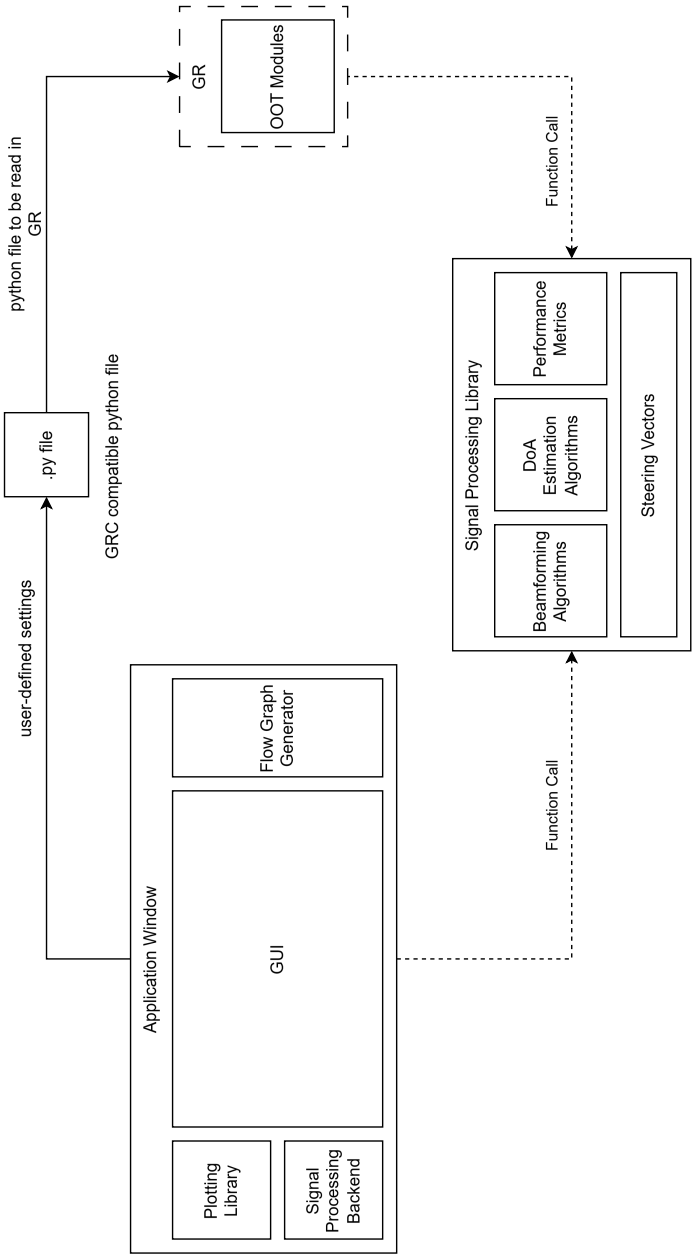


Figure 2.4: System block diagram.

The application window serves as the primary user interface, enabling users to intuitively configure sensor array parameters, select array processing algorithms, and visualize

the resulting beam patterns or estimation outputs. It acts as the front end through which all system functionalities are accessed.

Behind the scenes, the signal processing library performs the core computations. It houses essential algorithms for steering vector generation, Beamforming, and DoA estimation. The library is a separate entity common to the application window and the OOT modules and it is accessed through function calls.

To support real-world deployment and SDR experimentation, the system includes a set of custom OOT modules for GR. The application window can generate GR-compatible flow graphs based on the user's settings, enabling a smooth transition from design and simulation to real-world implementation in GR.

Together, these components form a tightly integrated toolchain, from configuration and simulation to real-time execution and bridging the gap between intuitive design and SDR-based prototyping.

Chapter 3

Application Window

3.1 User Interface

3.2 Visualization Plots

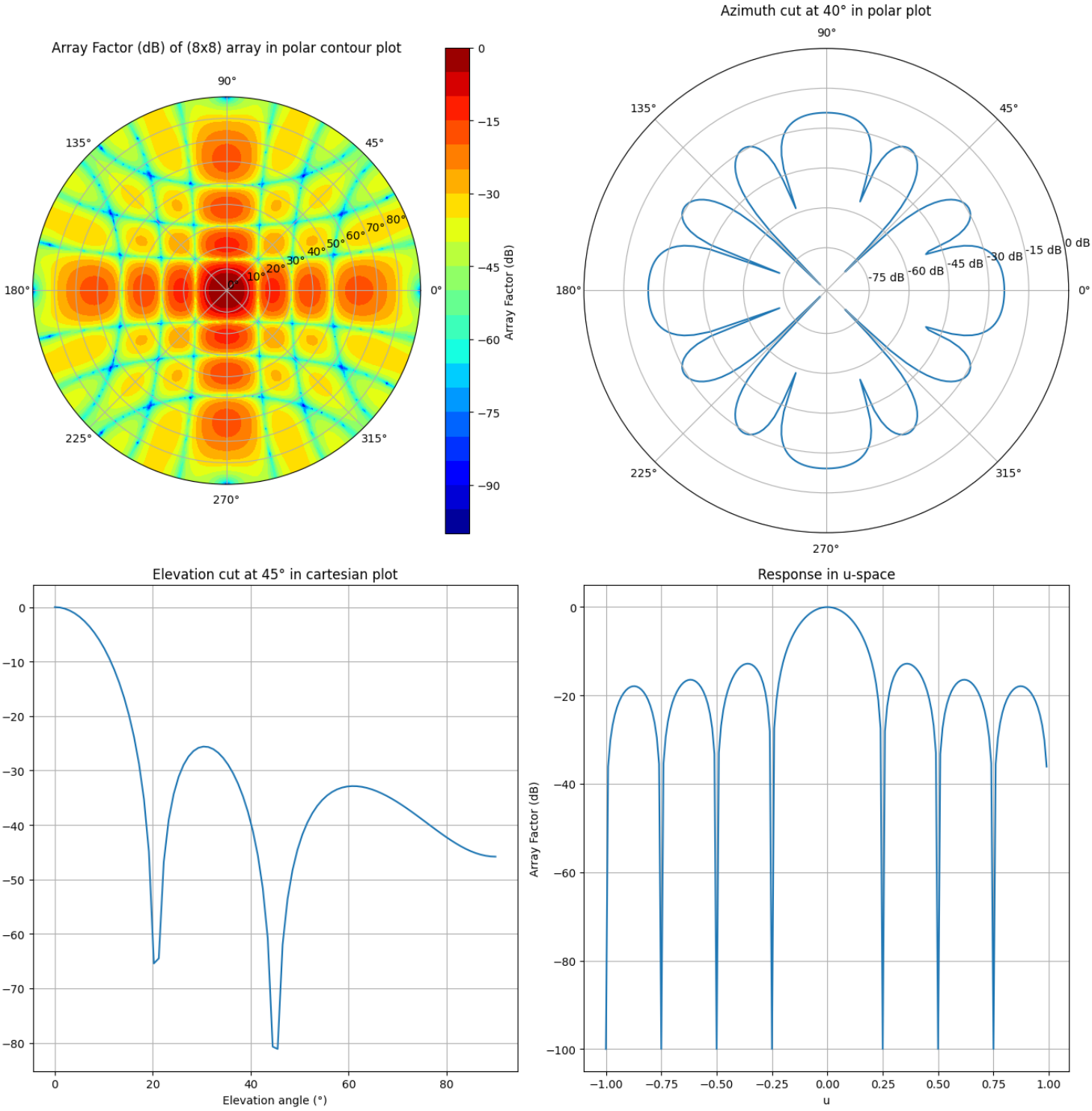


Figure 3.1: Different pattern visualization of an 8×8 array.

Chapter 4

Signal Processing Library

- Array
  - Uniform Linear Array (ULA)
  - Uniform Rectangular Array (URA)
  - Uniform Circular Array (UCA)
- Beamforming
  - Conventional
  - Minimum Variance Distortionless Response (MVDR)
  - Linearly Constrained Minimum Variance (LCMV)
- Direction-of-Arrival (DoA) Estimation
  - Beamscan
  - MVDR
  - MUltiple Signal Classification (MUSIC)

Chapter 5

Out-Of-Tree (OOT) Modules

- *Array Response*: A GR block that receives multiple input signals and outputs N phase-shifted versions of the superimposed signal based on the steering vector.
- *Steering Vector*: A GR block that can be configured to generate phase delays based on the array configurations.
- *Beamforming*: A GR block that takes in N phase-shifted signals and apply popular beamforming algorithms.
- *DoA Estimation*: A GR block popular that takes in N phase-shifted signals and applies DoA estimation algorithms.

Chapter 6

Conclusion

6.1 Deliverables

The deliverables will be the 3 components mentioned in the project systems in chapter 2. Along with this there will be proper documentation each aspect of the project.

6.2 License

All code created during this project will be released under the GNU General Public License v3 (GPLv3).

6.3 Timeline

$$2 \text{ hours} \times 7 \text{ days} \times 22 \text{ weeks} + 1.5 \text{ additional hours on sunday} \times 22 \text{ weeks} = 341 \text{ hours}$$

6.4 About me

- Name: Rahul Rajeev Pillai
- Place of residence: Coimbatore, Tamil Nadu, India
- University: Amrita Vishwa Vidyapeetham
- Academic Background: Bachelors in Electrical and Computer Engineering (2024)
- Work Experience: 1 year
- Work Designation: Jr. Design Engineer
- Field: DSP/Communication
- g-mail: rahulpillairj@gmail.com
- github: https://github.com/Ashborn-SM

6.5 Acknowledgement

I've read and understood the GSoC Student Info, and I agree to follow the rules, including the three-strike policy. I'll communicate regularly with my mentor, stay transparent about my progress, and follow community guidelines throughout the program.

Cyberspectrum is the best spectrum