**Proposal - Google summer of code 2023**

GRC : standalone applications and pluggable workflows.

By

Rahul Balaji



[**Overview 3**](#_rrar1dgps27e)

[**The project milestones 4**](#_4ve0pdidvxav)

[**How do I plan to achieve the milestones ? 6**](#_ksmxnrgo4jdp)

[1) Move GRC to a separate repository and remove dependence on GNU Radio: 6](#_buwz1tcz7y35)

[2) Modularize options block: 7](#_iiyvsw8bq3pk)

[3) Modularize Templates: 7](#_pq417jrh05c9)

[4) Allowing templating with jinja: 8](#_visvug6fqgsi)

[5) Testing: 8](#_p41mqisv1z0l)

[6) Modularize gr-modtool templates(extended goal): 9](#_lkrqv4b633ae)

[7) Support multiple domain workflows(extended goal): 9](#_aucsauh43mfb)

[**Plan of action: Timeline 9**](#_c95y22dj050p)

[4th april to 4th may 9](#_t0qpf5nuc2jq)

[4th may to 28th may 9](#_futauot9nzcs)

[29th may to 4th june 9](#_o2mc3qb93x7n)

[5th june to 18th june 10](#_obijqxzz3gq)

[19th june to 2nd july 10](#_5qy3dpxxaa5)

[3rd july to 9th july 10](#_bozw24cu1cn9)

[10th july to 16th july 10](#_3x1owadmsvcb)

[17th july to 23rd july 10](#_ypdkp1j0xhzb)

[24th july to Aug 6th 10](#_w7hufymgq33p)

[Aug 7th to aug 13th 11](#_mvi78v4o0g6o)

[Aug 14th to aug 20th 11](#_fxubwyu9nnf4)

[Aug 21st to aug 28th 11](#_o7tvupkfqgul)

[Apart from the project contribution I will 11](#_rfs20qbon0h)

[**What city and country will I reside in during the summer ? 12**](#_moj1q5efxmq9)

[**What are my expectations from my mentor ? 12**](#_lyr5cmdm1xzz)

[**Why did I choose this project ? 12**](#_wsxxhanbjkwl)

[**What is my experience with the open-source world as a user and as a contributor ? 13**](#_v9ows28pag99)

[**What are my other commitments ? 14**](#_5fm4251h30w4)

[**GSoC and University 14**](#_msnj6dtx9dj4)

[**The license under which I will code 14**](#_6eru77ci6jri)

[**What’s my plan after the GSoC period is over ? 14**](#_i7q55cuoty2a)

[**Acknowledgement on the “three strikes rule”: 15**](#_lxwrzzbepu27)

[**About me 15**](#_r9lzaw7upsih)

**Rahul Balaji**

Email address : [rahulbalaji78@gmail.com](mailto:rahulbalaji78@gmail.com)

GNU Radio id : rahul\_balaji

GitHub Id : [haru-02](https://github.com/haru-02)

LinkedIn : [Rahul Balaji](https://www.linkedin.com/in/rahul-balaji-b1aa491a6/)

“Cyberspectrum is the best spectrum.”

**GRC : Standalone application and pluggable workflows.**

**March 31, 2023**

# Mentor

Håkon Vågsether, Josh Mormon

# Overview

GNU Radio is a free and open-source software toolkit used for developing digital signal processing (DSP) applications. It provides a framework for designing and implementing signal processing algorithms, as well as a set of pre-built signal processing blocks for building complex radio systems. It is used in a wide variety of applications from software defined radios (SDRs) to radar and wireless communication systems.

In GRC, or GNU Radio Companion, is the GUI tool paired with GR for drawing and visually representing the flowgraphs in GNU Radio. The drag and drop interface makes things easier for users to build and test their radio systems. This project has mainly two objectives regarding GRC that needs to be accomplished.

Overtime, GRC has become its own standalone application with uses outside GR. With different workflows available and also different orgs forking and maintaining their own versions.

So the objective is to extract GRC from the GNU Radio software framework and create a more flexible workflow by making it modular. With an overview of steps already given in the ideas page, I plan to set them as sub goals of a milestone and follow them to accomplish the task. In this proposal, I have explained in detail on how I plan to achieve each step to the best of my knowledge.

If time permits, I also plan to modularize gr-modtool templates as well per [GREP 0026](https://github.com/gnuradio/greps/blob/main/grep-0026-modtool-template-rework.md), as well as work on supporting multiple domain’s workflows.

# The project milestones

The milestones are as given below.

1. **Move GRC to a separate repository and remove dependence on GNU Radio :**
2. Move GRC to a separate repo.
3. Record the dependencies of GRC on GR.
4. Remove the GR imports.
5. Replace parts of code that use GR import functions.
6. Clean up code.
7. **Modularize options block:**
8. Understand and record the imports under “templates” in options.block.yml
9. Replace imports with feature to use other plugins instead of GR.
10. **Modularize Templates:**
11. Record and understand the core/generator/cpp\_templates folder in GRC.
12. Overwrite code under Namespace gr.
13. **Allow templating with jinja as well:**
14. Understand Jinja code for templating.
15. Write a script for transforming a jinja code to grc.tree.yml template file.
16. Feed the yml file to grc to generate files in that format.
17. **UI improvements :**
18. Improve UI to accommodate for new changes to GRC, try giving users ability to choose workflow and add plugins from ui itself.
19. **Testing and bug fixes :**

Testing is required to make sure that all modules are running correctly and function the way they have to. Once all the other milestones have been reached, extensive testing will be conducted to ensure all features work as intended.

1. **Modularize gr-modtool templates(extended goal):**
2. Remove cruft, make the template folder more structured.
3. Replace all the template.py files with a more modular .mako files.
4. Install the new files kept in a directory along with gr-newmod.
5. Use Mako's TemplateLookup functionality to directly look up templates in that directory.
6. Enable in tree testing.
7. **Support multiple Domain workflows(extended goal):**
8. Understand the different workflows and the templates required.
9. Provide a dropdown menu on the tool bar where users can choose what workflow they need.
10. Then get the info from the front-end and allow GRC to choose from a list of templates for output based on the value chosen from the dropdown menu.
11. Need to figure out if this feature ties in with jinja template support feature.

The above two extended features have been somewhat discussed prior in the ideas page and in [GREP 0026](https://github.com/gnuradio/greps/blob/main/grep-0026-modtool-template-rework.md). However, most features planned are explored in detail over the next section.

# 

# 

# 

# 

# How do I plan to achieve the milestones ?

Each milestone are made up of numerous sub-goals that achieve it. The following section explains how I plan to achieve each sub-goal and my understanding of the work involved.

## Move GRC to a separate repository and remove dependence on GNU Radio:

* **Move GRC to a separate repository:**

It is vital that we maintain the git history while transferring the GRC to a separate repo so that the original authors of code are credited for their work. It also helps us have additional context. The process is as described below with reference from this [article](https://www.johno.com/move-directory-between-repos-with-git-history).

The steps involved are:

* + 1. Create a new repo.
    2. Clone the source repo.
    3. Filter all the commits aside from the directory you care about in the source repository.
    4. Add a local remote to the source repository.
    5. Then fetch the local remote.
    6. Then create a branch from the source’s master.
    7. Then merge in the source repo.
    8. Lastly, clean up the source repo.
    9. And commit.

While this may seem easy, the GRC is heavily integrated with GR, and it will take some work to determine if we need any external files also to be extracted that is outside the GRC folder and to ensure that all the required files are present, I will also perform a manual check apart from following these steps.

* **Record the dependencies of GRC on GR:**

In this step I will try and record all the files that are dependent on imports of modules form GR. This will help me understand the overall architecture of GRC on a deeper level, as well as help me describe the function of each file involved, with which I can determine what part of the code and files I should replace and remove, and what I should implement.

* **Replace parts of the code that use import from gr:**

The next issue to tackle is to actually remove those imports from those files and then find an alternative option/plugin that can be placed so that users can choose their own workflows. Basically to remove the gr imports and then replace the functions that use modules imported from GR with something else.

* **Clean up the code:**

To wrap up and make sure the changes I have made so far do not break the system entirely and it is at least in a state where it can compile without major issues.

## Modularize options block:

* This milestone involves a review over the imports under “templates” in options.block.yml, I will try and understand the use of all the modules, make a record and then replace the modules that import and use gr.

## Modularize Templates:

* The core/generator/cpp\_templates folder in GRC is one of the main components in GRC, and some files in the folder use namespace gr. This will also go through a similar process like options block. It boils down to,
  + 1. Identify the functionalities of code files.
    2. Group related components together into modules.
    3. Define the inputs and outputs of each module.
    4. Ensure that each module turns out with a specific purpose.
    5. Avoid unnecessary dependencies, especially remove dependencies on GR.
    6. Document each function/module.

## Allowing templating with jinja:

* Jinja is a templating engine that is quite popular for python that used to define templates which act like placeholders for dynamic data, which is populated during runtime.
* So the first thing we need to accomplish is to figure out what the output of the jinja template should be, wether xml, JSON or plain text, as this is the output file that we will later process.
* [Jinja documentation](https://jinja.palletsprojects.com/en/3.1.x/api/#basics) for reference.
* The grc.tree.yml file which decides the template of the GRC file should be dynamically generated from this Jinja template, to create a script for this function is the next sub-goal. The Jinja templates can produce yaml output, so it can be easier to generate the grc.tree.yml file.
* This dynamically generated file should then be used as the new template for grc file by GR.

## Testing:

* As an end product is designed and implemented, it is also necessary to make sure that it works as intended. To port all the GRC test cases and change them as necessary is also one of the milestones to be achieved in this project. We need to port the test cases also when we are separating GRC from GR repository while maintaining the commit history. This overs running all previously designed GRC test cases that are applicable and also generating new test cases where required.

## 

## Modularize gr-modtool templates(extended goal):

* The plan of action is to just follow the steps provided in [GREP 0026](https://github.com/gnuradio/greps/blob/main/grep-0026-modtool-template-rework.md). Removing cruft, using features of mako to make the templates more modular will be the goal of this phase.
* Learn mako.

## Support multiple domain workflows(extended goal):

* Providing the choice of workflow using a dropdown menu on the toolbar is the main idea here.
* Wire the choice to a set of templates that GRC has access to in runtime.
* Learn about different workflow templates.

# Plan of action: Timeline

## 4th april to 4th may

1. Getting familiar with the codebase.
2. Fixing bugs related to GRC.
3. Resolve issues in repository.

## 4th may to 28th may

1. Community bonding.
2. Discuss other project approaches with the mentors.
3. Make a blog explaining the plan of action to maintain transparency.
4. Try resolving any documentation issue if present while following and learning tutorials.
5. Improve documentation of both the Github Repository and the Wiki Page for enhancing contribution workflow.

## 29th may to 4th june

1. Targeting milestone 1, move GRC to a separate repository.
2. Record and learn more about GRC code architecture and its dependencies.
3. Clear all imports of gr.

## 5th june to 18th june

1. Replace parts of code that use gr imports.
2. Resolve dependency issues.

## 19th june to 2nd july

1. Learn and record options block modules.
2. Modularize options block.

## 3rd july to 9th july

1. Understand files in templates folder.
2. Modularize the templates block.
3. Start writing report for midterm evaluation.

## 10th july to 16th july

1. Finish modularizing templates block.
2. Submit midterm evaluation report by 14th july.
3. Learn about jinja.
4. Survey the documentation changes required for GRC.

## 17th july to 23rd july

1. Implement documentation changes for grc.
2. Continue learning on jinja.
3. Test GRC, with emphasis on modularized options and template blocks.

## 24th july to Aug 6th

1. checks on GRC extraction.
2. Testing GRC, continued.
3. Work on implementing Jinja template support.

## Aug 7th to aug 13th

1. Finish work on Jinja templates.
2. Manual test for Jinja templates.

## Aug 14th to aug 20th

1. Wrap up of necessary milestones.
2. Final code cleanup, bug\_fixes.
3. Work on extended goals.
4. Final GSoC report submission.
5. Testing, building test cases.

## Aug 21st to aug 28th

1. Extra period of time to improvise on any unexpected bottlenecks in plan.

## Apart from the project contribution I will

* Fix bugs and build features.
* Improve the documentation of the project where necessary.
* Write blogs to show the progress of the project we are building.
* Try to make a tutorial video.
* Weekly updates on project status in the mailing list.

The timeline might change according to planning or difficulty of the feature implementation or bug resolution. The prime focus would be to keep everything under the assigned timeline and add the extras if time permits. If in case I am not able to finish a particular task during the assigned timeline, I will definitely continue working on it after the GSoC period ends.

# 

# What are my expectations from my mentor ?

Guide me to understand the existing codebase of GNU Radio and GRC, and the associated terminologies, and other APIs of GNU Radio wherever I am incapable of doing so on my own. Suggest resources to have a clearer view of how things are done ideally. Help me come to a decision when I have more than one way of doing things and tell me why that is the best decision. Ultimately, take time to review my work and provide their timely insight while helping me in optimizing the code with respect to Time, Space, and CPU utilization complexities.

# Why did I choose this project ?

I was seeking to do a challenging project this summer, the Google Summer of Code 2023 provides the perfect opportunity. I have contributed to other open-source organizations before however after reviewing project ideas, I decided that I should focus on the GNU Radio projects. Once this decision was made and after reaching out to the community and Håkon Vågsether sir, it proved to be a good fit for reaching my project and learning goals.

I just recently got into the rabbit hole of radio hobby and i started searching for how we can make our own radios when I came across SDR systems, and how GNU Radio is one of the most useful tools for implementing such systems. Being an avid supporter of open-source projects, and a big fan of linux and the GNU project, I thought of getting in contact with the community and start contributing to the project.

The community was very welcoming to newcomers and they helped me quickly get started on learning more about the codebase and setting up the dev environment. The supportive and the friendly nature of the community really motivated me to work more for the cause and to learn more from them. Being part of the GNU Radio community is not only till GSOC but for future as well. I will continue to contribute to the community. Being very well versed in the technical stack that is required for the projects of GNU Radio uses while making code contributions to the existing codebase gave me a head start in the learning curve. It also allows for a more immediate impact on the overall GNU Radio project mission.

# What is my experience with the open-source world as a user and as a contributor ?

Contributing and learning from open-source projects is a never ending journey that I began almost two years ago on on my sophomore year in my university. I am a core-member in Google Student-Developer Club of my university, and also a core-team member of OpenCode, a technical club that is driven to promote and contribute to open-source projects. I have contributed to small projects before like our institute’s counseling services app, as well as big organizations like libre-office and kyverno, where I have made some minor bug fixes and documentation contributions. Since I only recently started to make contributions to GNU Radio, I have made a few PRs for minor bug fixes and clean up works on GRC. Learning and contributing from many organizations has made me pick up many important transferable skills like knowledge on software development techniques, tools like git and github, as well as programming knowledge. Most importantly, I believe that it has improved my ability to learn and pick up any tools required to code at a much faster pace and adapt to the environment.

Some PRs that I have posted at GNU Radio : [PR #6611](https://github.com/gnuradio/gnuradio/pull/6611) , [PR #6609](https://github.com/gnuradio/gnuradio/pull/6609).

Winter of Code, DSC NIT Rourkela : [project-avocado](https://github.com/haru-02/Proposal-for-DSC_NITR-project-avocado/blob/main/rahul-balaji.md).

Some repos that showcase my basic [C++](https://github.com/haru-02/competitive-coding) and [python](https://github.com/haru-02/Space-Invaders) skills.

# 

# What are my other commitments ?

My university end-semester exams and final evaluations occur from April 20th to May 5th at most. So **I have no commitments in the summer other than GSoC.** I will be available for at least 45 hours a week through online platforms and I am ready to extend whenever needed. I would be working full-time for GSoC. If I do get any commitments, I would accept them only after consulting the mentors. **The project will remain my top priority over this time period. I aim to work at-least for a period of 350 hours on this project.**

# GSoC and University

While I am a computer science student and the coding culture in my university actively encourages us to pursue such programs, I do not receive any extra credits for participating in this program. I pursue this endeavor solely of my own interest.

# The license under which I will code

**All my contributions will be under the default GNU Radio License of GPLv3.**

# What’s my plan after the GSoC period is over ?

Unquestionably, I would like to keep contributing to GNU Radio even after GSoC and will be available and enthusiastic to keep developing the platform. Even if I am not selected this year, I would like to help this project by resolving issues, suggesting new ideas, and generally participating in general discussions along with making code contributions. I usually help out people with code and documentation. In conclusion, I would like to keep contributing to the GNU Radio organization and do whatever I can to help.

# 

# 

# Acknowledgement on the “three strikes rule”:

I have thoroughly read the [rules of conduct](https://wiki.gnuradio.org/index.php?title=GSoCManifest) for GNU Radio, and I acknowledge the three strikes rule.

# About me

I am Rahul Balaji, a final year undergrad student from the National Institute of Technology, Rourkela, India pursuing a Bachelor of Technology in Computer Science and Engineering. I will graduate by the end of may. I love building Web applications and software solutions that solve problems which can be accessed by people anywhere in the world. I am a tech enthusiast and love open-source development. In my sophomore year, I joined two most prominent tech societies of our campus, Google student developer club and OpenCode. I also participate regularly in hackathons. I was also a part of the organizing team of HackNITR, which is one of the biggest hackathons hosted recently in India. Beyond open-source work, I also have experience working as a software Intern at John Deere, where I helped develope internal tools for audit using frameworks and tools like React.JS, Java - SpringBoot, Swagger docs etc. that increased the overall productivity. I am proficient with programming languages like Python and C++ and javascript. The resume below summarizes my overall skills. Beyond computers and code, I like to read books and play video-games during my free time.