Observations of pulsars with RAEGE 13.2m antenna in Yebes

Óscar Zamora García de Marina, Cristina García Miró
oscarzamoragarcia@gmail.com

Technical Report IT-CDT 2025-6

GOBIERNO DE ESPAÑA
MINISTERIO DE TRANSPORTES Y MOVILIDAD SOSTENIBLE
INSTITUTO GEOGRÁFICO NACIONAL
cnig
YEBES

Revision history

Version	Date	Author	Updates
1.0	8-08-2025	O. Zamora	First version
2.0	31-08-2025	O. Zamora, C. García	
3.0	17-11-2025	O. Zamora	Final version

List of acronyms:

SDR Software Defined Radio

FPGA Field Programable Gate Array

ADC Analog to Digital Converter

FFT Fast Fourier Transform

BW bandwidth

RF Radio-Frequency

RFI Radio-Frequency Interference

OS Operative System

DSP Digital Signal Processing

PRESTO PulsaR Exploration and Search TOolkit

RAEGE Red Atlantica de Estaciones Geodinamicas y Espaciales, Atlantic Network of Geodynamic and Space Stations

UAH Universidad de Alcalá de Henares, Alcala de Henares's University

UEM Universidad Europea de Madrid,European University in Madrid

UPNA Universidad Pública de Navarra,Public University of Navarra

RX Receive eXchange

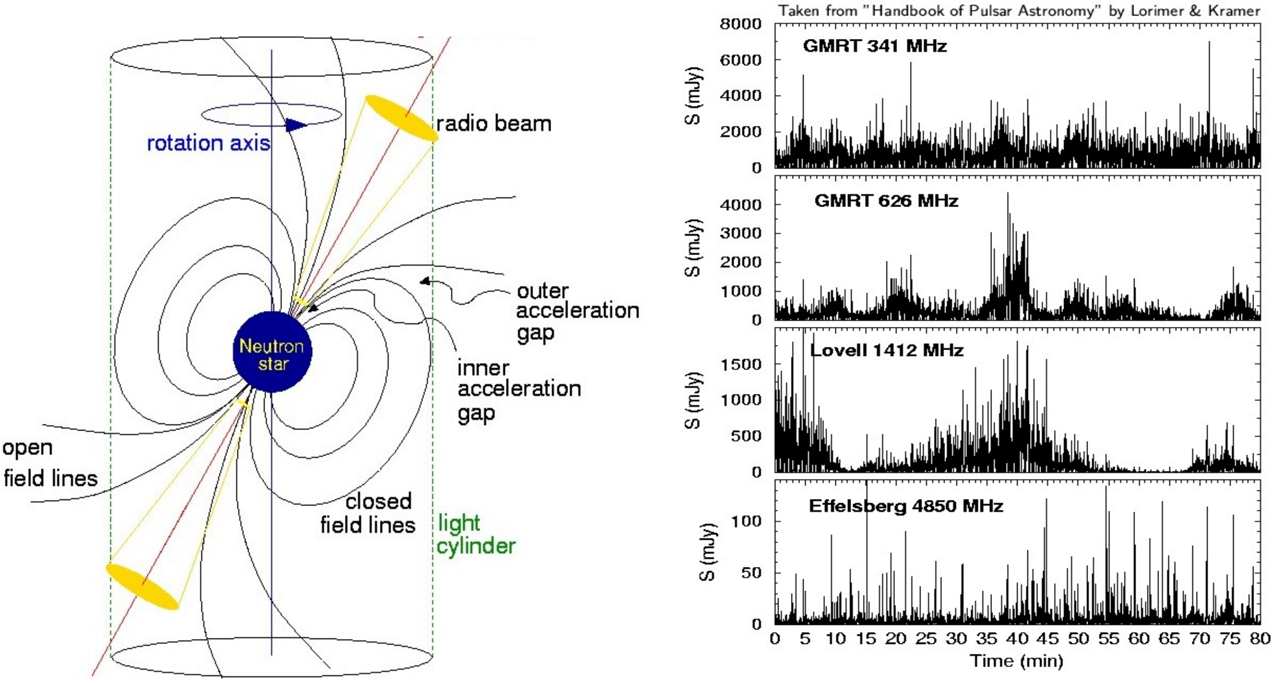
LO Local Oscillator

DM Dispersion Measure

Contents

1 Introduction

Pulsars are strong magnetized neutron stars spinning to a high velocity. Broadband electromagnetic waves are emitted in relationship with their spin period [Fig. 1a]. Seen by an observer these objects are very coherent and precise periodic sources. However, the theoretical background is incomplete and there is no complete formal model to explain this phenomenon [Lorimer et al. 2005].



(a) Model of a pulsar, rotational radio beam can be seen as a cosmic lighthouse.

(b) Flux density of PSR B1133+16 at different frequencies, the higher the frequency the less the flux density (S).

Figure 1: Characteristics of pulsars, Lorimer et al. 2005.

A key fact is that the flux density (S) of a pulsar depends strongly on the frequency of observation, at higher frequencies less power is emitted [Fig. 1b]. Although for low frequencies even a Yagi antenna can be used for detection, for the microwave region a parabolic set-up is needed. Cutting-edge cryogenic low noise amplifiers and a significant aperture should also be used to detect new faint objects or study a substantial amount of pulsars. So, the RAEGE 13.2m antenna at 3 GHz is a perfect instrument to employ in a pulsar survey at Yebes Observatory, or at any other of the RAEGE Network antennas.

Once collected, the signal is processed by a filter bank spectrometer and then recorded in a digital format used for pulsar studies (.fil). After observing, mathematical software can be used to perform a pulsar search. This report presents the workflow for pulsar detection used by Zamora Garcia de Marina 2025a applied to the Yebes RAEGE 13m antenna. Basically, it consists of an initial configuration and calibration of the antenna, a software spectrometer

(Pulsar_filterbank) that runs on a Software Defined Radio (SDR) and a modified version of PulsaR Exploration and Search TOolkit (PRESTO) software to search of pulsars offline once the observation has ended.

Using the RAEGE antenna for this type of observations can open a new research line at the Yebes Observatory, and can be used for educational purposes (Astroyebes, Universidad de Alcalá de Henares (UAH) and nearby universities internships, etc.). As was previously mentioned, more work needs to be done in the field of pulsars, so any new information would be relevant and useful. Furthermore, neutron stars are one of the most hot topics in modern astrophysics.

The presented work is part of a Bachelor's thesis [Zamora Garcia de Marina 2025a] presented at Universidad Europea de Madrid (UEM) performed in collaboration with the Universidad Pública de Navarra (UPNA). The latter has kindly lend us the SDR unit.

2 Pulsar receiver: SDR

A SDR is a very versatile electronic device built for Radio-Frequency (RF) digitization and signal manipulation (baseband conversion). SDRs also allow to use custom software for signal operations using a computer (digital signal processing). As RAEGE receiver frequency range stars at 3 GHz, after a superconducting filter installation to suppress Radio-Frequency Interference (RFI) at S-band, the chosen SDR was the Blade RF micro XA4 model [Table 4]. This SDR is operative between 47 MHz and 6 GHz, and it is also compatible with YEBES40M CX receiver (frequency range of 4.5-9 GHz).

Type	Cyclone V Field Programable Gate Array (FPGA)
Memory	3383 kbits
Transceiver type	Analog Devices AD9361
Receptors and sources	2x2 MIMO
Bandwidth	47-6000 MHz
Sample rate	61.44 MHz
Instantaneous bandwidth	56 MHz
Operational temperature	0-70 °C
Software supported	GNU Radio, Math Lab

Table 1: *BladeRF micro XA4 features, Nuand 2022.*

The BladeRF micro XA4 [Fig. 2] is composed of different parts. The first part is the RF section, where the RF signal is received by the Receive eXchange (RX) bus and then switched by a "SPDT" conmutator. The SPDT drives the RF signal to the most appropriate channel of the radio transceiver according to the RF frequency. Now in the radio transceiver [Fig. 3], the signal is amplified and then converted to baseband, in other words, it is shifted to a lower frequency for better handling. Down conversion is made using a Local Oscillator (LO), which has 3 different sets of frequencies depending on the RF value. Later, the filtered and amplified signal is digitalized using an Analog to Digital Converter (ADC). Lastly, the SRD FPGA [Fig.

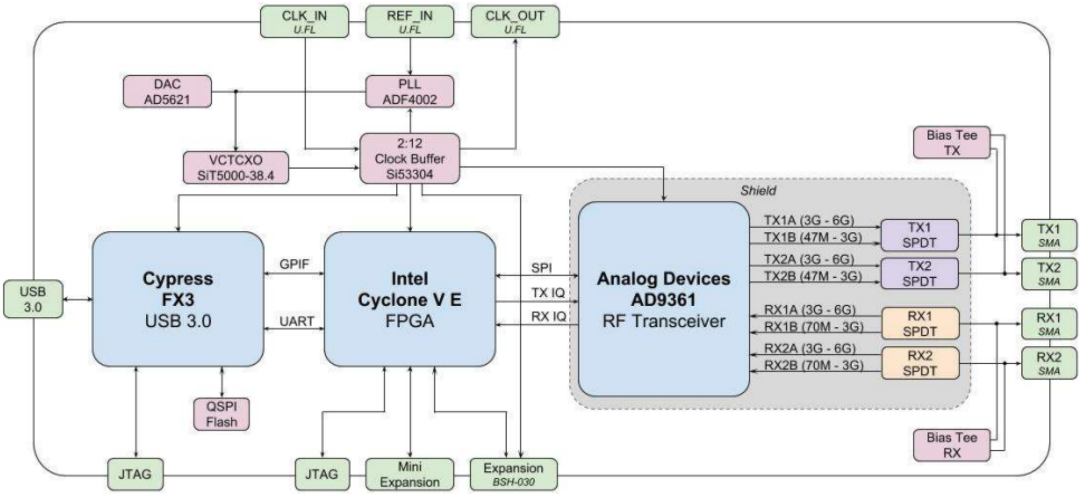


Figure 2: BladeRF micro XA4 block diagram, Nuand 2022.

2] will receive the ADC data stream and cast them to a computer using a USB bus. In the computer, digital signal operations will be performed at software level. The FPGA has different firmware sets, but for pulsar detection it will only be used as the SDR operative brain.

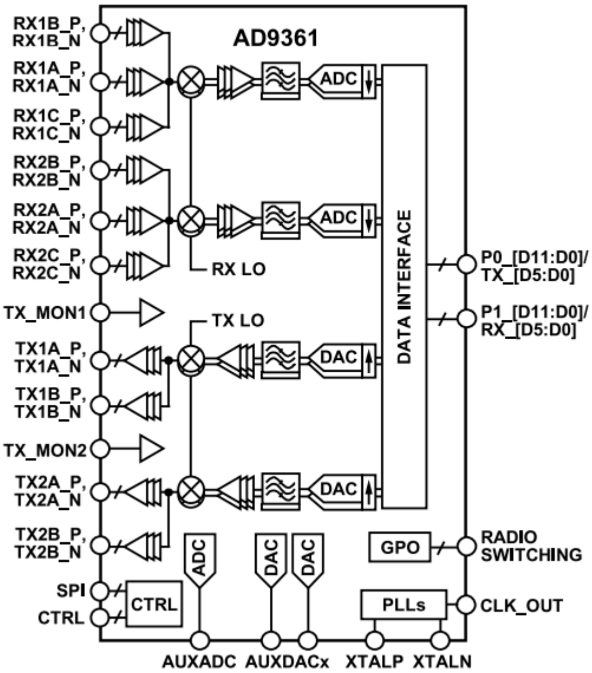


Figure 3: BladeRF micro XA4 RF transceiver block diagram, Analog Devices 2024.

Before using a brand new BladeRF, it is necessary to flash an FPGA version in the SDR unit, according to Nuand 2025, using the “Easy installation for Ubuntu:The bladeRF PPA” instructions. Following the same reference, install the BladeRF library (LibbladeRF) and a FPGA updater in a Ubuntu computer. If the SDR device will be used on GNU Radio, LibbladeRF and Gr-osmosdr libraries must also be installed in the computer according to Stolnikov 2025. PRESTO will be required as well, so the application needs to be installed in Ubuntu, version 22.04.5 LTS (recommended). BladeRF also works on virtual machines (Oracle Virtual Box), but USB 3.0 mode must be enabled. The BladeRF has 2 reception (RX) channels, but for pulsars only one channel is used (dual polarisation observations will effectively double the integration time). By default, that channel is RX1, as used in this work.

It is important to note that baseband conversion is done to satisfy the Nyquist Theorem of sampling with cheaper digitizers. For an adequate sampling of a signal, the sample rate must be at least twice the highest frequency of a sampled signal. In baseband, the 0 frequency corresponds to the central RF frequency in complex representation, so the sample rate must be at least twice the bandwidth.

Another important fact is that the original real and analogue RF signal, after conversion to baseband, is now an I/Q complex signal, with I and Q the real and imaginary parts. The signal is then represented as a phasor, to avoid ambiguities when tensorial products are performed on the signal (more than one subspace is feasible).

3 Software spectrometer

SDR and Digital Signal Processing (DSP) technologies had made possible software spectrometers. These devices perform FFTs using mathematical code, and then compute the squared magnitudes of the resulting data (only the magnitude of the signal is useful instead of complex numbers). Operations are coded using Python on the GNU Radio software, a free software development toolkit that provides signal processing blocks to implement software-defined radios and signal processing systems.

To study pulsars using professional software (e.g. PRESTO toolkit), an advanced software spectrometer is needed, such as the Pulsar_filterbank code by Leech 2020. In this study, an upgraded version developed by Zamora Garcia de Marina 2025b compatible with Python 3 (newer versions of GNU Radio, v.3.10) is used. All modifications are detailed in Bachelor's thesis Zamora Garcia de Marina 2025a, see also a lite version of the GNU flow diagram on Appendix A, [Fig. 11]. The Pulsar_filterbank code is composed of 3 main stages: A first Fast Fourier Transform (FFT) of the data, subsequent filtering and RFI removing, and lastly, data formatting (.fil file output) for PRESTO compatibility (instead of raw I/Q or Wav audio files used by common SDR software).

To install the spectrometer software in Ubuntu, go to the YEBES branch [Zamora Garcia de Marina 2025b] and clone the GitHub repository, i.e. using bash console:

```
#!/bin/bash
git clone -b YEBES https://github.com/oscarzamora98N/
   pulsar_filterbank.git
```

Use the YEBES branch (-b YEBES), that has been updated for the RAEGE network.

3.1 FFT stage

First, the SDR settings are defined in the Osmocom source block: sample rate, bandwidth (BW), central frequency, and gain. Then is performed an spectral inversion on Raw I/Q format data due to PRESTO requirements (it stars from high to low frequency). After that, the total length of the data vector is defined, so total observation time can be set depending on the sample rate. At the end of this stage, the FFT and squared modulus are calculated. The FFT size (or filterbank size) is determined automatically by the Fb_helper.py code depending on the pulsar parameters. The equation $N_{chans} \sim \Delta t(f)/resol_{time}(PW50)$ is a good approximation, while the desired resolution time is lower than PW50 time, where $\Delta t(f)$ and PW50 are the difference in arrival times of a pulse at different frequencies [Eq. 2] and the pulse width at 3 dB. By changing the resolution parameter, the size can be adjusted to the user preferences (recommended). It is important to note that the pulsar parameters must be changed if a different pulsar than PSR B0329+54 is observed. All parameters can be found at the bottom of the flow diagram on GNU Radio.

In principle, the Blade RF can perform observations only using the RF stage of RAEGE, but the best approach is to employ the IF stage, where the signal level can be adjusted setting the IF attenuation at the ACS antenna controller (Object Explorer). The RF (or sky) frequency can be set on the freq block (labeled as Obs. Frequency). On the contrary, if the SDR is connected to the IF stage, you should calculate the equivalent IF frequency of your RF frequency of observation. The IF frequency can be defined on the IF parameter block and will be the tuned frequency at the SDR "RF" stage. You can calculate the equivalent IF using this equation $IF = RF - LO$.

When using a not powerful enough server, data calculation delays can occur on the GNU Radio code resulting in data loss at the file building stage. At the copying issues block, set the parameter to 1 for powerful computers to restore the original data concatenation code and 2 for less powerful ones. In the later case, the user needs to concatenate the .filtemp data to the .fil file by running the code below.

```python
#Python code, run it using Python + name.py
import shutil
outfile=input()
dataname = outfile.replace(".fil", ".filtmp")
inf = open(dataname, "rb")
outf = open(outfile, "ab")
shutil.copyfileobj(inf, outf)
inf.close()
outf.close()
```

3.2 Filtering stage

At the end of the FFT calculation, the sample rate is reduced by integration or decimation. Either of them can be selected using 1 or 0 for the integrator parameter value. The best option is to use decimation (0), because the RFI will be detected better by the PRESTO cleaner. RFI is not white Gaussian noise, therefore averaging is not a useful approach. On the other hand, for environments less affected by RFI, integration will improve signal level. This function averages all samples over a specific time to reduce the effective sample rate. Decimation or integration rate (fbrate) will be determined automatically depending on the pulsar parameters (half width of the pulse beam). The decimation will adjust the efficiency on the time resolution (sample rate). Using much higher resolution than necessary will be computational expensive for data saving and processing for the same output.

In the next step, an RFI detector in the frequency domain will detect and erase harmful signals according to a user defined threshold. The default value is 2.5, so if the sample power is 2.5 times or greater than the mean power, its value will be replaced by the mean. Additionally, a roll-off correction of the SDR bandpass will be calculated and applied. That operation will correct both ends of the bandpass using the central mean value. In order to fill the RFI mask erased samples, a fast noise source block provides random samples that become FFT averaged samples. It is recommended to use the default value for RFI masking to avoid removing the weak signal of the pulsar. If an RFI source is known, it can be added to the rfilist to be flagged.

Optionally, at the end of this stage, a DC block (high pass filter) can be applied by setting HP parameter value equal to 1. This operation will reduce the power of steady interferences detected in the previous Single Pole IIR filter. The power subtraction can be adjusted by the HP pseudo gain parameter, then the data will be reduced n times its value ($newvalue[i] = value[i] * (1 - g)$). This operation will also flatten some power peaks in the data (constant background noise), avoiding channels to saturate or having high differences of power.

3.3 File building stage

Data is written in Sigproc filterbank format ($.fil$), a compatible file format with PRESTO. Filterbank files are composed of a header and data (in power per channel format). The header contains information about the data gathering process (metadata), see Table 2.

HEADER_START	Flag for a starting header
Rawdatafile	Name of the original data file
Datatype	Data type (1=filterbank)
Telescope id	Name of radio telescope used (RAEGYEB=67, RAEGSMAR=68)
Machine id	Backend used (66=YEBES-SDR)
Source name	Pulsar name
Obs Date String	UTC date of observation (Year+Month+Hour)
RA J2000	RA coordinate (Hour:Min:Sec), in GNU Radio put as Decimal Hour.
Dec J2000	Dec coordinate (Deg:ArcMin:ArcSec), in GNU Radio put as Decimal Degree.
Nchans	Number of filterbank channels
Foff	Filterbank channel bandwidth (MHz)
Nbits	Number of bits per time sample
Beam	Number of radio telescope beams (RAEGE has only 1 beam)
Nifs	Number of separate IF channels
...	...
HEADER_END	Flag for a ending header

Table 2: *Header of a Sigproc filterbank file (.fil)*

The first step to build the output data file is to convert the observational data to characters using utf-8. While observing, the data is appended in a temporal file named “psr-+yr+m+h(utc).filtmp”. The temporal and final data path can be set in the prefix parameter. Upper and lower branches in the flow diagram produce other format files: 16-bit and de-dispersed files respectively. But only the middle brach must be used (8-bit files), the others should be disabled (wide and profile values = 0). At this point, any other operation will be performed in Python using the fb_helper code. When the observation ends, the header parameters and the data of the temporal file are written by the update_header code in a final .fil output file.

4 Data processing: PRESTO

PRESTO is a professional Pulsar Searching software for Linux platforms developed by Scott Ransom 2025. Apart from pulsar candidates search and known pulsars detection, this application also implements other functionalities such as RFI detection and cleaning, FFT and data visualization, data conversion, time of arrivals (TOAs) calculation, single pulses and fast radio bursts (FRBs) search. As mentioned before, PRESTO 5.0.3 works with standard file formats (fil, psrfits) and time series in text file format. The following sections detail the workflow for searching and observing known pulsars.

4.1 Workflow and theoretical considerations

The methodology for processing pulsar data in PRESTO is as follows:

1. RFI is searched and erased in the time domain.

2. FFT and time series are analysed at 0 DM.
3. RFI is searched and erased in the Fourier domain (a.k.a. Birds).
4. De-Dispersion.
5. Search for periodic signals using Fourier acceleration.
6. Search for pulsar candidates among the periodic signals found.
7. Fold the time series of the selected candidates.

Before going any further, check Appendix C, for the installation procedure of PRESTO. Make sure that your built application runs the sanity checks. For further details on PRESTO functions and their usage, check Ransom 2012b.

4.1.1 RFI in the time domain

Using the rfifind command, PRESTO searches for short and strong radio pulses (human made RFI) and statistical anomalies in the time domain Fig. 4. Despite of their periodicity, pulsars are very weak sources in comparison to interferences. The RFI is searched by mean and standard deviation comparison in each bin (time fragment) with the expected parameters for that filterbank at 0 DM (a signal emitted from Earth or not dispersed). If a bin is affected by RFI, the software clips it (ignore that temporal fragment) and sets it as a bad interval. The fraction of bad intervals should be below 30% to consider it as a good observation. Also, the receiver gain should be adjusted to ensure that the data power is below the red dash line named “Max power”. If PRESTO is showing an empty graph, probably the SDR detector is saturated by overpowered data or the data is not strong enough.

4.1.2 FFT analysis at DM=0: RFI visualization

Due to the fact that the data is a temporal power spectra (three-dimensional: power vs time and frequency), before going any further it is needed to do a single de-dispersion at DM=0 (terrestrial signals) to have all the frequencies (channels) together as function of time (two-dimensional: power vs time). The FFT can be analysed by using realfft and explorefft functions. Performing the FFT twice (first in the filterbank and the second now) will provide the same result but scaled by a number and rotated 180 degrees. Therefore, a power vs frequency representation can be obtained from a power vs time representation. At Fourier domain, the RFI can be seen as strong and narrow peaks. Fig. 5 shows an example of AC RFI around 50 Hz. If a peak is larger than 60 (normalized power) at low frequency, it is considered RFI and should be removed.

4.1.3 RFI in Fourier domain: "Birds"

To remove RFI in PRESTO the user needs to provide a Birds list in a specific format. Then, the accelsearch function with z=0 can be employed for searching and characterizing periodic RFI. Only the Birds that peak above 60 of normalized power need to be included. The birds file must have the “.birds” extension and the format specified in Fig. 6.

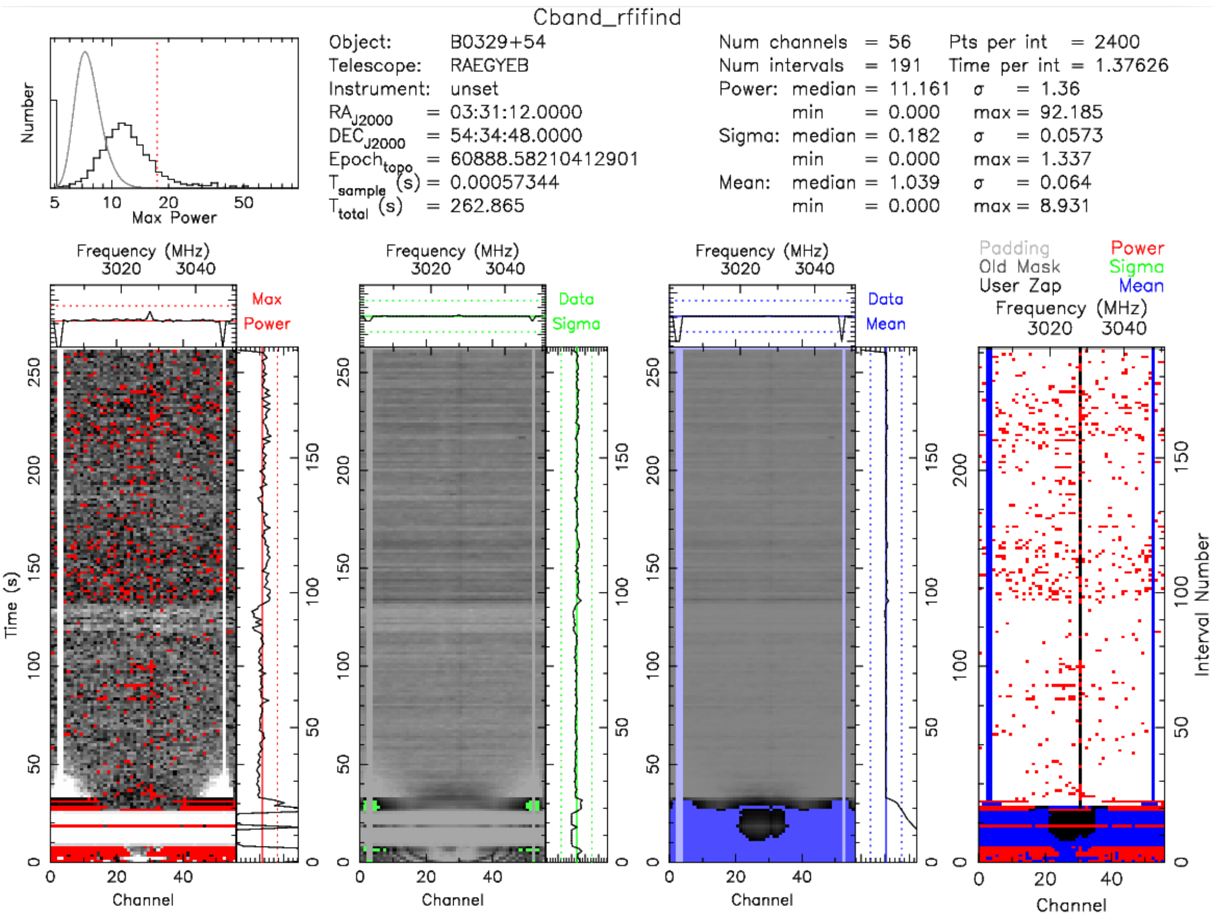
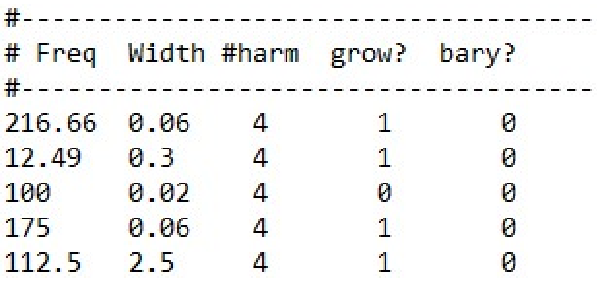


Figure 4: Data collected in a observation of pulsar PSR 0329+54 that can be considered as "good". The power is below the red line and almost constant, and RFI level (red dots) is moderate. Sometimes at the start of the integration (< 1min), the combined SDR and filterbank system is not stable, so data power will oscillate (even in separate channels). PRESTO detects these anomalies (blue and green dots) and trims them.



```
#-------------------------------------
# Freq   Width #harm  grow?  bary?
#-------------------------------------
216.66  0.06    4       1       0
12.49   0.3     4       1       0
100     0.02    4       0       0
175     0.06    4       1       0
112.5   2.5     4       1       0
```

Figure 6: Birds file format

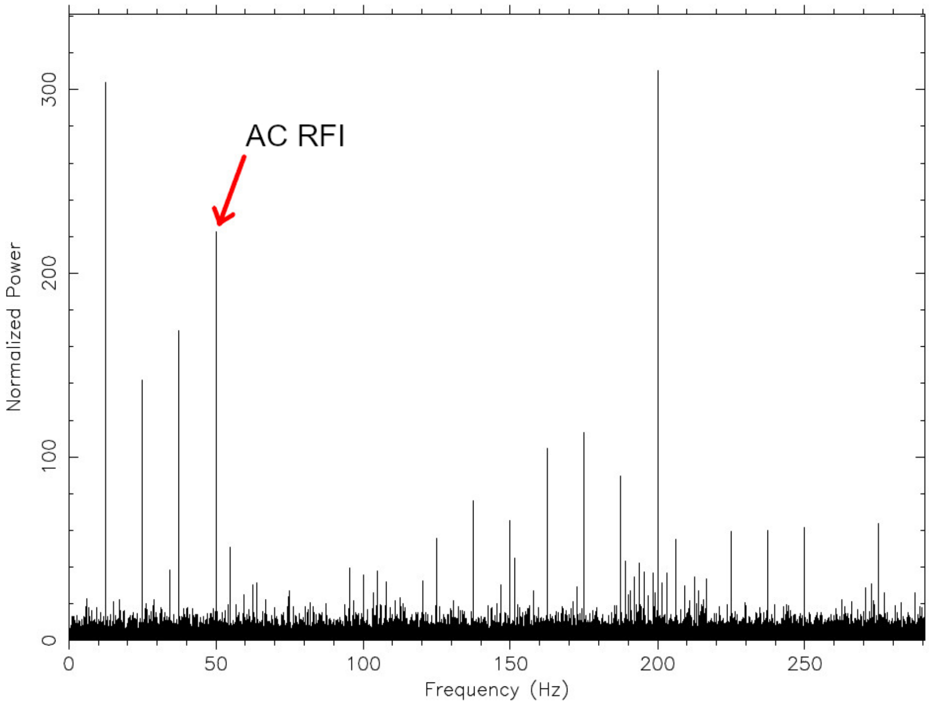


Figure 5: Example of alternating electric current RFI at 50 Hz

4.1.4 Incoherent de-Dispersion

The free electrons of the interstellar medium disperse the traveling electromagnetic waves from a pulsar. That dispersion will introduce a delay as a function of the frequency, so the pulse will become wider in time (a pulsar emits in broadband). Incoherent de-dispersion consists in reversing the dispersion process by aligning the different frequencies of a pulse in time (Fig. 7). To perform the de-dispersion is needed to know the Dispersion Measure (DM) value of the observed pulsar and the equations that are shown below:

$$DM = \int_0^d n_e dl \tag{1}$$

$$\Delta t \simeq \frac{e^2}{2\pi m_e c}(f_{ref}^{-2} - f_{channel}^{-2})DM \tag{2}$$

where f_{ref} parameter is the reference frequency to consider, usually the central frequency of the observed band, and $f_{channel}$ is the central frequency of the channel to be de-dispersed. A key fact is that every channel has a width (is not infinitesimal), so incoherent de-dispersion correction is limited by that value. As a result incoherent de-dispersion does not correct inside the channels (t_{DM}), so the resolution is limited by that dispersion time:

$$t_{DM}[\mu s] \simeq 8.2976 * 10^9 * DM * \Delta f * f^{-3} \tag{3}$$

with Δf and f the width and central frequency of the channel, in MHz. The intra-channel dispersion time, should be significantly lower than the pulse period (P_0) to have enough resolution in a pulsar detection. At low frequencies, a small Δf ($\sim 0.1MHz$) should be used in order to mitigate intra-channel dispersion and maintain sensitivity to detect millisecond and/or high DM pulsars. Usually, t_{DM} is larger than the native resolution of a channel ($t = \frac{1}{\Delta f}$), so it is not considered. Using an observing frequency of 3.5 GHz for RAEGE antennas, and 1 MHz of bandwidth for the filterbank channels, $t_{DM} \simeq 0.2\mu s/DM$. For example, for PSR B0329+54 $t_{DM} \simeq 5.4\mu s \ll P_0 = 0.7s$, so 1 MHz resolution is more than enough.

Taken from "Handbook of Pulsar Astronomy" by Lorimer & Kramer

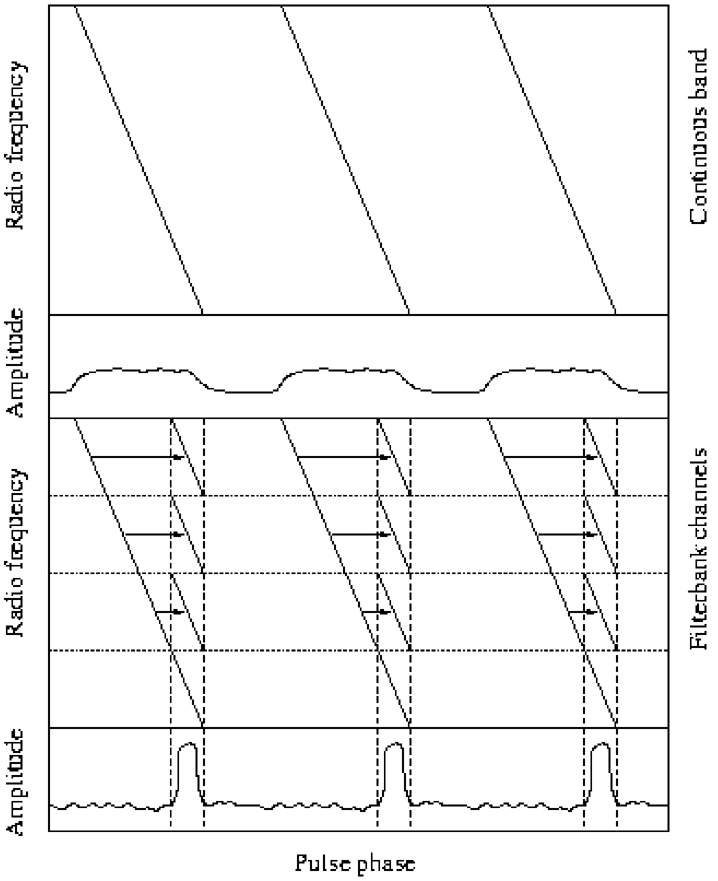


Figure 7: Incoherent de-dispersion process

When the de-dispersion is calculated, PRESTO also corrects for the Doppler effect in the observed signal using the TEMPO application, that contains a database of radio telescopes and its coordinates.

4.1.5 Periodic signals search

Isolated pulsars like PSR B0329+54, have a frequency profile in the form of a peak or modified Dirac delta function [Ransom 2012a]. PRESTO accelsearch function was developed to detect binary pulsars using Fourier acceleration, but it can also be used to detect isolated pulsars [Ransom et al. 2002]. With the optional parameter zmax=0, accelsearch will search for individual objects using Fourier interpolation (Fourier series). Fourier acceleration consists in correcting frequency o phase drifts produced by a rotational binary system, using the Fourier domain. In PRESTO, the acceleration is approximated as a constant, because $T_{obs} \ll T_{orbital}$. Accelsearch will use the f dot ($\dot{f}$) plane, where each point of this space has a z value, and it represents the number of bins the frequency has been shifted (Hz/s) from the reference one (Fig. 8). Using zmax=0 will fix the frequency to the reference.

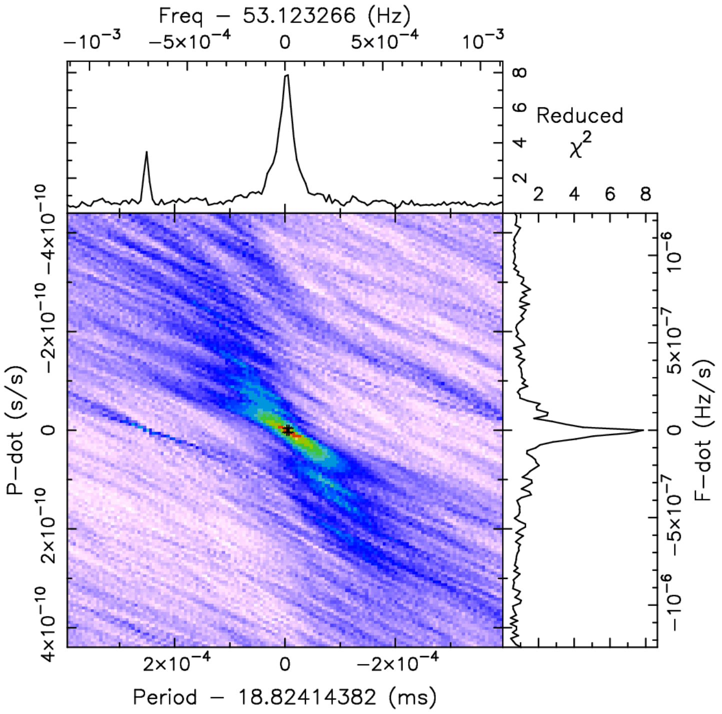


Figure 8: F dot plane example for an isolated pulsar

4.1.6 Pulsar candidates search

By using ACCEL shift script, PRESTO will search and classify pulsar candidates among the periodic signals found before. The classification process will remove candidates with DM issues, duplications or candidates that are harmonics of other candidates.

4.1.7 Candidates folding

Folding consists in adding together all the individual pulses of the detected pulsar in order to get a better signal to noise ratio. Folded pulse profiles have a constant phase in function of frequency, as the dispersion was already corrected.

4.2 Bypassing the workflow

The presented workflow can be bypassed using only the rfifind and prepfold functions.

1. Use rfifind to check RFI levels, data quality and creating an RFI mask.

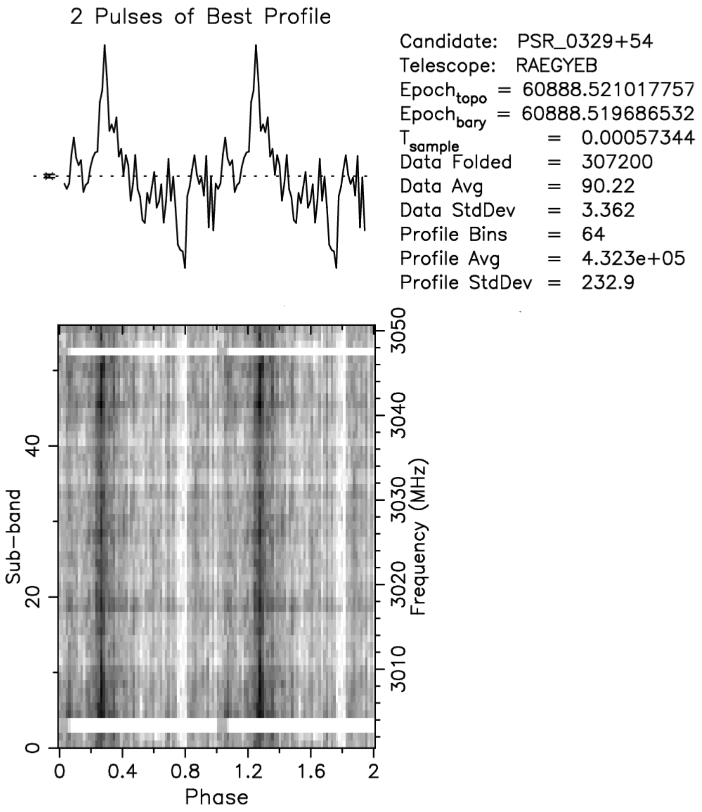


Figure 9: Example of individual pulses and folded data of PSR B0329+54 taken with RAEGYEB, on August 1st 2025

2. Use prepfold function using the pulsar name (-psr [name]) and the RFI mask (-mask [name of the file]). For the pulsar name use only the coordinates, for example PSR B0329+54 will be -PSR B0329+54.

Also in prepfold, use the same number of subbands as channels and be aware of the DM that is being used/found by prepfold. For online help and explanation of the different parameters, type ”prepfold” on the bash shell. Be careful when employing this shortcut, because fine tuning could be required to achieve good results (or even to be able to detect an object). To achieve proficiency in PRESTO is needed a long learning curve.

4.3 Single pulse search: Giant pulses studies

PRESTO has a single pulse search mode, that can be employed for giant pulses detection. Run the script “single_pulse_search.py *.dat” with “–nobadblocks” to perform a giant pulses search. This feature could be explored in detail in a possible future technical report on giant pulses observations with RAEGE antennas, that will include calibration details and a sensitivity calculator based on MURMUR [Natali 2025].

5 Detection of pulsars with Yebes RAEGE 13m

The first step in pulsar observation is to set up a reception system, composed of a parabolic antenna, an SDR and a server running Linux with GNU Radio and PRESTO installed.

The antenna used for this work is the Yebes RAEGE antenna, a ring-focus radio telescope with 13.2m of aperture and a cryogenic receiver covering 3 to 14 GHz effective bandwidth. This development can be exported to any other RAEGE or VGOS antenna. Table 3 shows the characteristics of the operational RAEGE antennas.

Table 3: RAEGE antennas characteristics

Parameter	RAEGYEB	RAEGSMAR
Coordinates	40°31’24.5"N, 3°05’18.6"W	36°59’7.15"N, 25°7’33.27"W
AZ range	540° aprox.	
EL range	0-100°	
Main reflector diameter	13.2m	
Subreflector diameter	1.55m	
System f/D	0.4	
Surface RMS	< 200 microns	
AZ max. speed	12°/s	
EL max. speed	6°/s	
VGOS receiver band a	RF 2972.4-3472.4[a] MHz –> IF 500-1000 MHz (LO 2472.4 MHz)	
Trec (K)	18 (both lin. pol.)	
Tcal (K)	5-7	3-5
DPFU (K/Jy)	0.038, 0.038	0.035, 0.035
Aperture efficiency[b]	77%	71%
Tsys (K)	50-70	
SEFD (Jy)	2000	
Gain curve EL POL	1.0	

[a] Use <3.4 GHz for RAEGSMAR to avoid 5G RFI.

[b] Aperture efficiency is calculated as $(2k\,DPFU)/A_{geom}$.

The most appropriate frequency to observe pulsars is the lower end of the RAEGE frequency band a, from 3 to 3.5 GHz (< 3.4 GHz for RAEGSMAR). The SDR is then connected through its input port RX1 to the Up-down converter UDC_1 located in the Yebes 40m back-end room, using a coaxial SMA cable connected to the IF R/V monitor output. At the UDC_1 the RF signal is downconverted to 500-1000 MHz IF frequency.

For configuration and control, the SDR is connected through the USB 3.0 type B port to a windows laptop running a virtual machine with Ubuntu Jellyfish Linux distribution, where the GNU Radio and PRESTO applications are installed.

5.1 Detection threshold

To know if a pulsar could be detected on a certain antenna, the modified radiometer equation for pulsars should be employed [Lorimer et al. 2005]:

$$S_{min} = \beta \frac{(S/N_{min})T_{sys}}{G\sqrt{n_p t_{int} \Delta f}} \sqrt{\frac{W}{P-W}} \tag{4}$$

where β is a correction factor (for an 8 bit filterbank is almost equal to 1), n_p the number of polarizations detected, t_{int} the integration time, Δf the observation bandwidth, P and W the period and width of the pulse. S_{min} represents the minimum detectable flux density of a pulsar for that particular setup, so if the spectral flux density (S) of a pulsar exceeds that value, it will be detectable (as we have enough sensitivity).

Using RAEGYEB and PSR B0329+54 parameters:

1. $T_{sys} \sim 50K$
2. $SEFD = S_{sys} = 2000Jy$
3. $DPFU = G_{max} = 0.038K/Jy$
4. $\beta \simeq 1$ (a 8 bit filterbank is employed)
5. $\Delta f = 50 * 10^6 Hz$ (we assume a 50 MHz bandwidth)
6. $n_p = 1$ (we only use 1 polarization)
7. $P = 0.71s$
8. $W = 6.6 * 10^{-3}s$ (we used W50)
9. $S/N_{min} = 20log(\sigma_{min}) = 14dB$ (we assume 5σ)
10. $t_{int} = x$ s
11. B0329+54: $S_{3000} \simeq 0.02Jy$ [Kramer et al. 2003]

$$t_{int} = \left(\frac{SNR_{min} * T_{sys}}{G * S_{min}}\right)^2 \frac{W}{(P-W)\Delta f} \simeq 160s \sim 2.7min \tag{5}$$

If you use 10 MHz of bandwidth, $t_{int} \simeq 13.3min$ using the same setup, a longer time is needed. In a typical 20 minutes integration, using the previous setup and pulsar you will get a significant amount of flux:

$$S_{min}(t_{int} = 20min) \simeq 7mJy \tag{6}$$

5.2 Antenna configuration

For pulsar observations, the 13m antenna is configured and calibrated as for a VGOS observation, following:

1. Apply pointing model from RAEGE> pcorrections()

2. From Field System fs13m> proc=point (pulsar PSR B0329+54 has been added as psr0329)
3. fs13m> setupvgos (to configure UDCs, LOs and backend channels as per VGOS observations)
4. fs13m> calbb (to configure fivept and onoff parameters)
5. fs13m> taurusa (to send the antenna to a nearby calibrator with flux tabulated in /usr2/control/flux.ctl file)
6. fs13m> fivept (to correct pointing offsets)
7. fs13m> onoff (to calibrate in amplitude using rxg_file and flux.ctl information)
8. fs13m> calnoise=off (to turn off the noise diode modulation during the pulsar integration)
9. fs13m> pcaloff (to turn off the phase calibration signal during the pulsar integration)
10. fs13m> psr0329 (to track the pulsar and start the integration[1])
11. Adjust the IF attenuation for both linear polarizations using the ACS Object Explorer, at the DownConvBroadBand_1 component [2]
12. Repeat fivept and onoff between integrations

5.3 Spectrometer configuration

To start the software spectrometer, open the GNU Radio application using this code on bash console:

```
#!/bin/bash
gnuradio-companion
```

Also you can inset that in a text file with .sh ending to run it as program (in properties, permissions put a mark on Execute to allow that). Once you are in GNU Radio, open the provided flow diagram on Zamora Garcia de Marina 2025b, using a mouse click on file:open (at the left corner of the window) and select the file which ends on“.grc” in the pulsar filterbank folder (at root/home directory). Before starting an observation, check that you are using the parameters shown below (on the flow diagram you will find these at the bottom).

- Sample rate: srate = 50 MHz (is the bandwidth value too)
- IF: IF = 525 MHz (central IF frequency)
- RF: freq = 2997.4 MHz (equivalent RF frequency)
- Observation length: runtime = xx min

[1] The ACS only generates predicts for 20 minutes, use timed directives for longer integrations, e.g. psr0329@2025.213.16:00:00, and check the execution queue with Field System command ’ti’.

[2] For short integrations 13 dB IF attenuation and 4-3 dB gain at the spectrometer works fine. For longer integrations the gain needs to be increased.

- Number of filterbank channels: fbsize = 56 (Use ∼ bandwidth value)
- Number of channels adjust: resolution = 14
- RF gain: rfgain = 4-3 dB
- IF gain and Baseband gain = 10 (do not change these values)
- Signal integrator On/Off: Integrator = 0
- Band pass filter, edges correction On/Off: rolloff = 1
- High pass filter On/Off: hp = 1
- High pass filter gain: hpgain = 0.6
- Decimation On/Off: decimation = 1
- RFI mask generator threshold: thresh = 2.5 (Use a very high value to disable this function)
- Detector scaling : dscale = 1
- Copying issues case: test = 1 (Use 2 if you are losing observation time, don't forget to copy the .filtemp file on .fil file manually as it was explained before.)
- PSR parameters: ra, dec, p0, pw50, dm and source (name).
- Set-up parameters: backend = 66 (YEBES-SRD), antenna = 67 (RAEGYEB). Use 68 for RAEGSMAR and 66 for YEBES40m antennas.

To start an observation on the software spectrometer, hit the play button on the top graphical menu of GNU Radio window. Once done, a bash window will appear with Blade RF SDR information. If the window is closed by itself instantaneously, check if you have plugged the SRD to the computer. If everything is okey, the window will be closed when the observation has ended. Occasionally, if you have low gain a 0/0 message will be displayed as an error, this is a bug on the original code of RFI masker. If you get the error repeatedly you should disable the RFI mask.

If you are using test = 2 run this code when the observation has ended.

```python
#Python code, run it using Python + name.py
import shutil
outfile=input()
dataname = outfile.replace(".fil", ".filtmp")
inf = open(dataname, "rb")
outf = open(outfile, "ab")
shutil.copyfileobj(inf, outf)
inf.close()
outf.close()
```

5.4 Data processing commands: Bypass

Whenever the integration has finished, the raw data can be inspected with the PRESTO command rfifind, and de-disperse and folded with the command prepfold, for quick checks (Bypass approach). For the general workflow refer to Appendix D.

Execute the following steps before starting using PRESTO:

```
#Everytime you start the computer, use that commands to active PRESTO
cd presto
source presto-env/bin/activate
#Put the name of your folder, in my case is a_YEBES
cd a_YEBES
```

Copy the data of the observed pulsar (.fil file), which is inside pulsar_filterbank folder, to the PRESTO folder. And use the name of your PSR data instead of psr-202508011230.fil on the commands that are shown below.

```
#First run rfifind, replacing psr-202508011230.fil by the name of
   your pulsar data.
rfifind -time 1.0 -o Sband psr-202508011230.fil
#Second run prepfold doing the same as before
prepfold -nsub 56 -dm 26 -dmstep 1 -nodmsearch -mask Sband_rfifind.
   mask -psr 0329+54 psr-202508011230.fil
```

1. -nusub = number of subands to fold, use the same number as channels you have
2. -mask = The RFI mask generated by rfifind, used to erase RFI.
3. -psr = Pulsar name, using this PRESTO will use as target the expected parameters of that pulsar (in theory).

   ↓ Use only if you need ↓

4. -dm = Target DM, use the DM of the observed pulsar.
5. -dmstep = Number of steps on DM search, use 1 as the lowest number of steps.
6. -nodmsearch = No search DM, only will be used the DM value set.

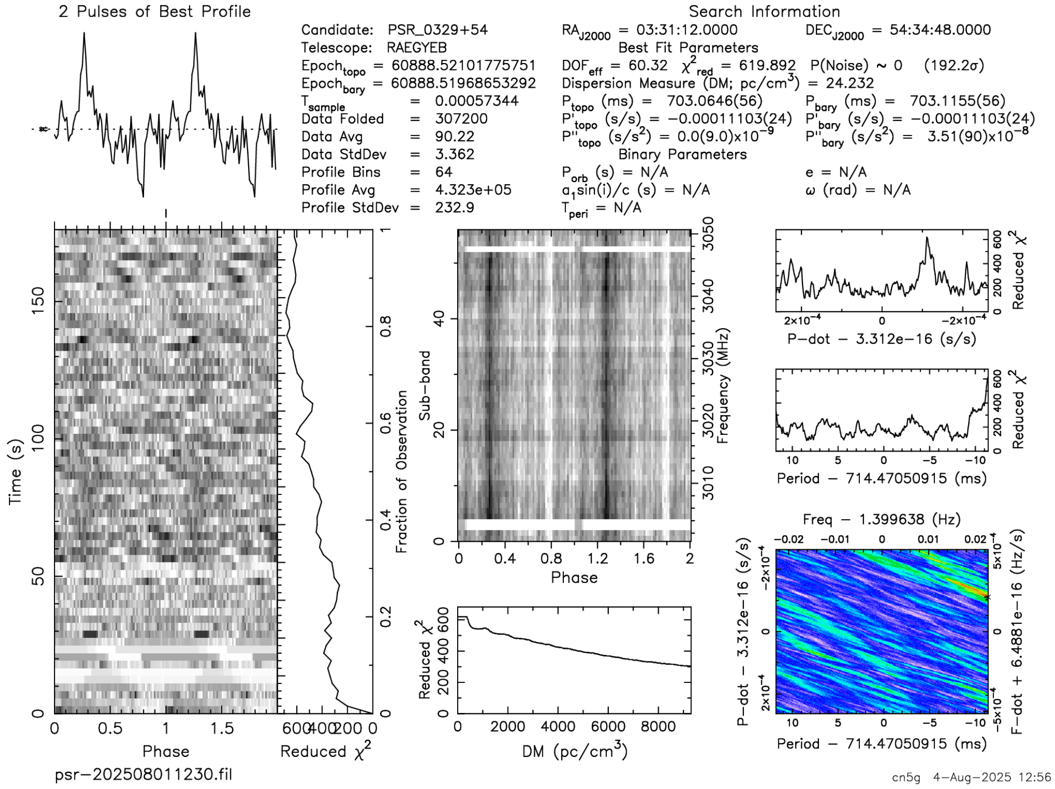
If, for unknown reasons, PRESTO is not aiming at the expected DM value, use parameters -dm, -dmstep and -nodmsearch. If you need more optional parameters or you need help, type only “prepfold” to invoke the documentation of prepfold (this also works for other PRESTO commands). Use this example as a model, processing data on PRESTO is not usually easy.

If you need a quick check on observation time, final sample rate (the SDR sample rate was decimated) or any other information, you can read the .fil file using the code shown below.

```
#To read a .fil file
readfile name.fil
```

6 Results and Conclusions

In a 20 minute observation of the pulsar PSR B0329+54 conducted the day 01-08-2025 at 12:30 UTC+2, we obtained the following plot shown below (effective integration time was 175 s).



```
                                                    Search Information
Candidate:  PSR_0329+54          RA_J2000 = 03:31:12.0000                DEC_J2000 = 54:34:48.0000
Telescope:  RAEGYEB                        Best Fit Parameters
Epoch_topo = 60888.52101775751   DOF_eff = 60.32  χ²_red = 619.892  P(Noise) ~ 0   (192.2σ)
Epoch_bary = 60888.51968653292   Dispersion Measure (DM; pc/cm³) = 24.232
T_sample        =  0.00057344    P_topo (ms) =  703.0646(56)         P_bary (ms) =  703.1155(56)
Data Folded     =  307200        P'_topo (s/s) = −0.00011103(24)     P'_bary (s/s) = −0.00011103(24)
Data Avg        =  90.22         P''_topo (s/s²) = 0.0(9.0)x10⁻⁹     P''_bary (s/s²) =  3.51(90)x10⁻⁸
Data StdDev     =  3.362                   Binary Parameters
Profile Bins    =  64            P_orb (s) = N/A                     e = N/A
Profile Avg     =  4.323e+05     a₁sin(i)/c (s) = N/A                ω (rad) = N/A
Profile StdDev  =  232.9         T_peri = N/A
```

Figure 10: Pulsar B0329+54 folded profile and f dot graph across 175s of processed time in a 20min observation

Therefore, using RAEGYEB antenna and the setup detailed in this report, the pulsar B0329+54 was detected with 5σ confidence. Note that the $\dot{f}$ graph presents a spectral drift (the maximum is not centered) as it was not possible to synchronize the SDR clock with a more accurate clock (GNSS or hydrogen maser). The Blade RF does not provide this capability and its crystal oscillator is not as accurate as needed. For precise timing studies an SDR that allows synchronisation with a GNSS or an atomic clock would be required.

In conclusion, this study has shown the viability of pulsar studies using RAEGE antennas.

7 Acknowledgments

O. Zamora acknowledges the helpful guidance on PRESTO usage and problem solving provided by S. Ransom (NRAO), the willingness of M. Natali to include tabulated data for 3 GHz pulsar emission into the MURMUR sensitivity calculator, and the advisory of Raquel Gómez Medina (UEM), Rafael Ruiz Feliú (UPNA), Silvia Díaz Lucas (UPNA) and Cristina García Miró (OAN) in the completion of the bachelor's thesis.

The Yebes Observatory acknowledges UPNA for lending the SDR unit used during the observing tests.

References

Analog Devices, Inc. (2024). *AD9361 Datasheet*. URL: https://www.analog.com/en/products/ad9361.html.

Kramer et al. (2003). "Simultaneous single-pulse observations of radio pulsars: Fig 3". In: *Astronomy and Astrophysics*.

Leech, Marcus (2020). *Pulsar_filterbank*. URL: https://github.com/ccera-astro/pulsar_filterbank.

Lorimer, Duncan and Michael Kramer (2005). *Handbook of pulsar astronomy*. 1st ed. Cambridge: Cambridge university press.

Natali, Mario (2025). *MURMUR*. URL: https://i0naa.altervista.org/index.php/downloads.

Nuand (2022). *bladeRF 2.0 micro schematics*. URL: https://www.nuand.com/product/bladerf-xa4/.

– (2025). *Blade RF Getting Started: Linux*. URL: https://github.com/Nuand/bladeRF/wiki/Getting-Started%3A-Linux#user-content-Easy_installation_for_Ubuntu_The_bladeRF_PPA.

Ransom, Scott (2012a). *Accelerating Acceleration Searches for Pulsars*. URL: https://www.cv.nrao.edu/~sransom/accel_accel_searches.pdf.

– (2012b). *Searching for Pulsars with PRESTO*. URL: https://www.cv.nrao.edu/~sransom/PRESTO_search_tutorial.pdf.

– (2025). *PRESTO (V5.0.3)*. URL: www.cv.nrao.edu/~sransom/presto/.

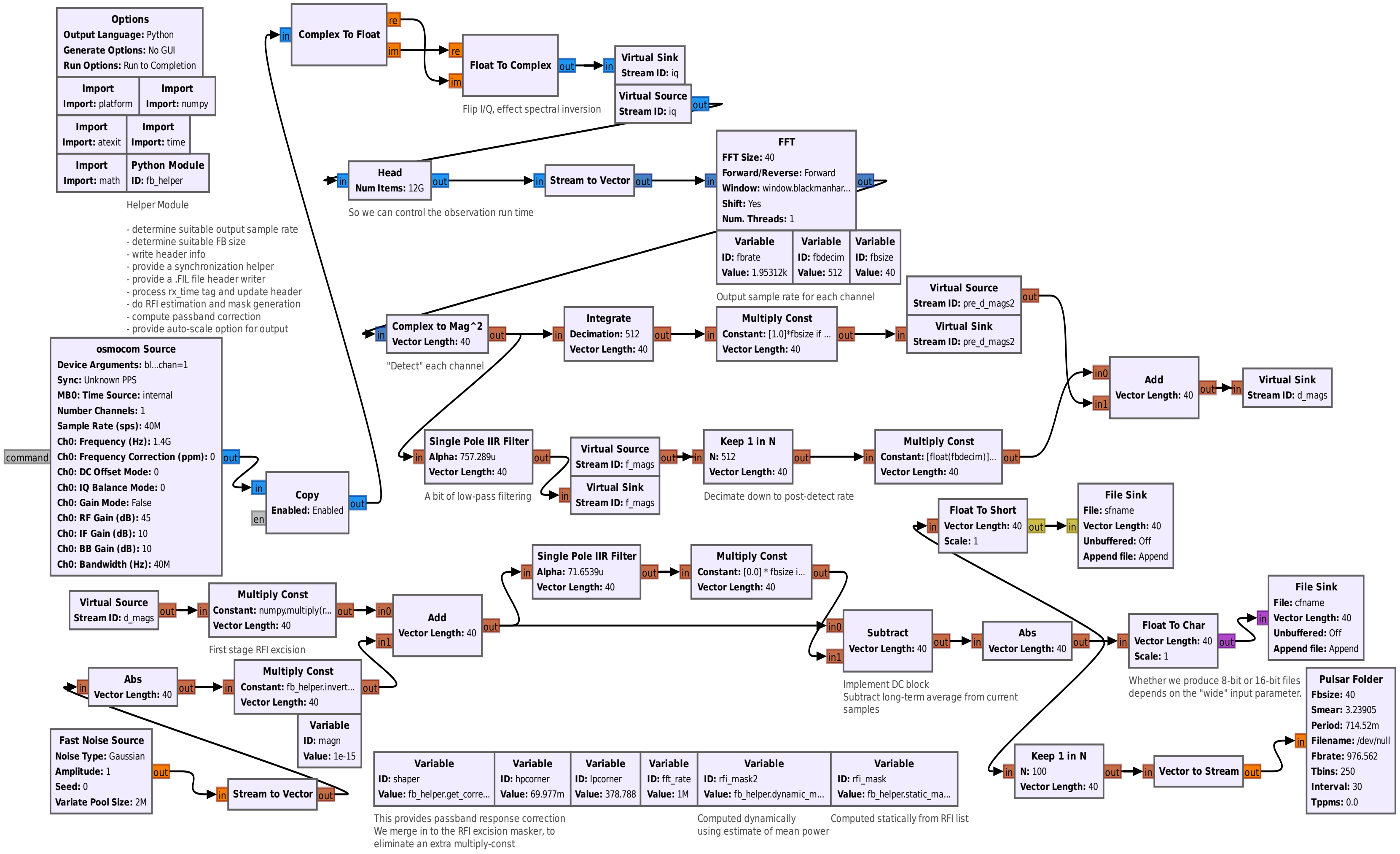
Ransom, Scott M., Stephen S. Eikenberry, and John Middleditch (2002). "Fourier Techniques for Very Long Astrophysical Time-Series Analysis". In: *The Astronomical Journal* 124.3, pp. 1788–1809. ISSN: 1538-3881. DOI: 10.1086/342285. URL: http://dx.doi.org/10.1086/342285.

Stolnikov, Dimitri (2025). *Wiki:Build process*. URL: https://osmocom.org/projects/gr-osmosdr/wiki.

Zamora Garcia de Marina, Óscar (2025a). "Detección de hidrógeno neutro en la vía láctea". Bachelor's Thesis. UEM.

– (2025b). *Pulsar_filterbank (Bare minimun Python 3 fix)*. URL: https://github.com/oscarzamora98N/pulsar_filterbank.

A Pulsar Filterbank GNU Radio flow diagram



Options
Output Language: Python
Generate Options: No GUI
Run Options: Run to Completion

Import
Import: platform

Import
Import: numpy

Import
Import: atexit

Import
Import: time

Import
Import: math

Python Module
ID: fb_helper

Helper Module

- determine suitable output sample rate
- determine suitable FB size
- write header info
- provide a synchronization helper
- provide a .FIL file header writer
- process rx_time tag and update header
- do RFI estimation and mask generation
- compute passband correction
- provide auto-scale option for output

osmocom Source
Device Arguments: bl...chan=1
Sync: Unknown PPS
MB0: Time Source: internal
Number Channels: 1
Sample Rate (sps): 40M
Ch0: Frequency (Hz): 1.4G
Ch0: Frequency Correction (ppm): 0
Ch0: DC Offset Mode: 0
Ch0: IQ Balance Mode: 0
Ch0: Gain Mode: False
Ch0: RF Gain (dB): 45
Ch0: IF Gain (dB): 10
Ch0: BB Gain (dB): 10
Ch0: Bandwidth (Hz): 40M

Copy
Enabled: Enabled

Complex To Float

Float To Complex

Flip I/Q, effect spectral inversion

Virtual Sink
Stream ID: iq

Virtual Source
Stream ID: iq

Head
Num Items: 12G

So we can control the observation run time

Stream to Vector

FFT
FFT Size: 40
Forward/Reverse: Forward
Window: window.blackmanhar...
Shift: Yes
Num. Threads: 1

Variable
ID: fbrate
Value: 1.95312k

Variable
ID: fbdecim
Value: 512

Variable
ID: fbsize
Value: 40

Output sample rate for each channel

Complex to Mag^2
Vector Length: 40

"Detect" each channel

Integrate
Decimation: 512
Vector Length: 40

Multiply Const
Constant: [1.0]*fbsize if ...
Vector Length: 40

Virtual Source
Stream ID: pre_d_mags2

Virtual Sink
Stream ID: pre_d_mags2

Add
Vector Length: 40

Virtual Sink
Stream ID: d_mags

Single Pole IIR Filter
Alpha: 757.289u
Vector Length: 40

A bit of low-pass filtering

Virtual Source
Stream ID: f_mags

Virtual Sink
Stream ID: f_mags

Keep 1 in N
N: 512
Vector Length: 40

Decimate down to post-detect rate

Multiply Const
Constant: [float(fbdecim)]...
Vector Length: 40

Float To Short
Vector Length: 40
Scale: 1

File Sink
File: sfname
Vector Length: 40
Unbuffered: Off
Append file: Append

Virtual Source
Stream ID: d_mags

Multiply Const
Constant: numpy.multiply(r...
Vector Length: 40

First stage RFI excision

Add
Vector Length: 40

Single Pole IIR Filter
Alpha: 71.6539u
Vector Length: 40

Multiply Const
Constant: [0.0] * fbsize i...
Vector Length: 40

Subtract
Vector Length: 40

Implement DC block
Subtract long-term average from current samples

Abs
Vector Length: 40

Float To Char
Vector Length: 40
Scale: 1

File Sink
File: cfname
Vector Length: 40
Unbuffered: Off
Append file: Append

Whether we produce 8-bit or 16-bit files depends on the "wide" input parameter.

Abs
Vector Length: 40

Multiply Const
Constant: fb_helper.invert...
Vector Length: 40

Variable
ID: magn
Value: 1e-15

Fast Noise Source
Noise Type: Gaussian
Amplitude: 1
Seed: 0
Variate Pool Size: 2M

Stream to Vector

Variable
ID: shaper
Value: fb_helper.get_corre...

Variable
ID: hpcorner
Value: 69.977m

Variable
ID: lpcorner
Value: 378.788

Variable
ID: fft_rate
Value: 1M

Variable
ID: rfi_mask2
Value: fb_helper.dynamic_m...

Variable
ID: rfi_mask
Value: fb_helper.static_ma...

This provides passband response correction
We merge in to the RFI excision masker, to eliminate an extra multiply-const

Computed dynamically using estimate of mean power

Computed statically from RFI list

Keep 1 in N
N: 100
Vector Length: 40

Vector to Stream

Pulsar Folder
Fbsize: 40
Smear: 3.23905
Period: 714.52m
Filename: /dev/null
Fbrate: 976.562
Tbins: 250
Interval: 30
Tppms: 0.0

Figure 11: Pulsar_filterbank´s lite version of GNU Radio flow diagram, Zamora Garcia de Marina 2025a.

B Commercial SDR comparative

An SDR to be usable for Pulsar Observations must be compatible with the pulsar_filterbank script (GNU Radio) and with RAEGE's frequency range (3-3.5 GHz). Specifically, the SDR must be compatible with the GNU Radio gr-osmosdr library, listed on Stolnikov 2025. Also, a significative bandwidth is required (al least 30 MHz), because the detected power is proportional to the bandwidth. For smaller bandwidths, larger integration times are required for detection (up to several hours). Table 4 shows the characteristics of the most common SDRs compatible with pulsar observing.

SDR	F_{max}	Bandwidth	Cost	Comments
Blade RF micro xA4	6 GHz	61MHz	540€	Used for this work
HackRF ONE	6 GHz	20 MHz	300€	Small bandwidth
LIME SDR	3.8 GHz	61 MHz	450€	-
USRP (Ettus family)	6 GHz	56 MHz	1400€	GPS can be added

Table 4: *SDRs compatible with pulsar observing system, and comparative among them*

It is important to note that the two SDR available in Yebes Observatory have limitations: the ADALM-Pluto is not compatible with the OSCOM library, and the Air Spy SRD operates up to 1.7 GHz that is compatible with the IF frequency range for band a but it has a very limited BW of 10 MHz.

C Installation of PRESTO

C.1 Installation procedure for Ubuntu and PRESTO 5.0.3

An installation manual can be found in PRESTO's GitHub, but for a non Linux user it can be tough to use. Therefore, this section provides the translated to English version of the original manual of Zamora Garcia de Marina 2025a.

Fist, make sure that repository sites and repositories are updated and then clone PRESTO's repository.

```
#!/bin/bash
sudo apt-get update
sudo apt update
sudo apt upgrade
git clone https://github.com/scottransom/presto.git
```

Now, a virtual environment of Python 3 will be installed to avoid software incompatibility issues between the Operative System (OS) and PRESTO. If it is not installed, the OS automatic upgrades could broke your PRESTO application. Then, some packages will be installed inside the virtual environment.

```
#Virtual environment install python3.10-venv
cd presto
python3 -m venv presto-env
source presto-env/bin/activate
sudo apt install cmake
#This package is needed for the next steps
sudo apt-get install libopenmpi-dev
sudo apt install openmpi-bin openmpi-common libopenmpi-dev

#Opennmpi's environment variables set
export C_INCLUDE_PATH=/usr/lib/x86_64-linux-gnu/openmpi/include:$
   C_INCLUDE_PATH
export LIBRARY_PATH=/usr/lib/x86_64-linux-gnu/openmpi/lib:$
   LIBRARY_PATH

#Installaion of packages needed by PRESTO inside the virtual
   environment
apt install git build-essential libfftw3-bin libfftw3-dev pgplot5
   libglib2.0-dev libcfitsio-bin libcfitsio-dev libpng-dev latex2html
    gfortran tcsh autoconf libx11-dev python3-dev python3-numpy
   python3-pip
pip install meson meson-python ninja
pip install --upgrade pip
#Now out of PRESTO, meson is built
cd $PRESTO
meson setup build --prefix=$VIRTUAL_ENV
```

Now, TEMPO should be installed outside of the PRESTO folder and using the virtual environment.

```
git clone git://git.code.sf.net/p/tempo/tempo
cd tempo
./prepare
./configure
make
```

Before building PRESTO, the environment variables should be set, for make findable the installed software or packages by OS. REPLACE “cn5g” name with your system/user name.

```
echo 'export PRESTO=/home/cn5g/presto' >> ~/.bashrc
source ~/.bashrc

echo 'export TEMPO=/home/cn5g/tempo' >> ~/.bashrc
source ~/.bashrc

echo 'export LIBRARY_PATH=/home/cn5g/presto/presto-env/lib/
   x86_64-linux-gnu:$LIBRARY_PATH' >> ~/.bashrc
```

```
source ~/.bashrc

echo 'export LIBRARY_PATH=/home/cn5g/presto/presto-env/lib/
   x86_64-linux-gnu:$LD_LIBRARY_PATH' >> ~/.bashrc
#Bash's settings file is reloaded
source ~/.bashrc

echo 'export LD_LIBRARY_PATH=/home/cn5g/presto/presto-env/lib/
   x86_64-linux-gnu:$LD_LIBRARY_PATH' >> ~/.bashrc
source ~/.bashrc

#####At the end of bashrc should be something like this####

export PRESTO=/home/cn5g/presto
export TEMPO=/home/cn5g/tempo
export LIBRARY_PATH=/home/cn5g/presto/presto-env/lib/x86_64-linux-gnu
   :$LIBRARY_PATH
export LIBRARY_PATH=/home/cn5g/presto/presto-env/lib/x86_64-linux-gnu
   :$LD_LIBRARY_PATH
export LD_LIBRARY_PATH=/home/cn5g/presto/presto-env/lib/
   x86_64-linux-gnu:$LD_LIBRARY_PATH

#########################################################
```

Now check that everything is okay for the meson building stage with “check_meson_build.py”. This script will check if you have defined well the environment variables and if the software was properly installed. If an okey check is given, you can go to the next step. If that is not the case, you can ask to ChatGPT (a dangerous deal!).

```
python check_meson_build.py

####If mpi package makes an error, use:######

meson setup build --prefix=/usr --libdir=/usr/lib --pkg-config-path=/
   usr/lib/x86_64-linux-gnu/pkgconfig
export LDFLAGS="-L/usr/lib/x86_64-linux-gnu"
export LIBS="-lmpi"

#######################################################
```

Finally, PRESTO is compiled and installed. Don’t forget to run ”makewisdom”, to get faster FFTs.

```
meson compile -C build
meson install -C build
makewisdom
```

```
##makewisdom installation can fail due to v-environment, use:##

gcc -I./include src/makewisdom.c -o makewisdom -lfftw3f
mkdir -p lib
./makewisdom > lib/fftw_wisdom.txt

################################################################

#Now you can test you PRESTO build using:

python tests/test_presto_python.py

python examplescripts/ffdot_example.py

python python/fftfit_src/test_fftfit.py

#If you get errors running these, use the comple path of the script
```

PRESTO is now operative, while you have active the virtual environment and you are inside the PRESTO folder.

```
#Everytime you start the computer, use that commands to active PRESTO
cd presto
source presto-env/bin/activate
#Put the name of your folder, in my case is a_YEBES
cd a_YEBES
```

C.2 Troubleshooting

If “accelshifty” funtion gives an error, modify the file “sifting.py” on“:/home/cn5g/presto/presto-env/lib/python3.10/site-packages/presto/sifting.py” and replace the code of “def candlist_from_candfile:” up to the first line commentary by:

```
def candlist_from_candfile(filename, trackbad=False, trackdupes=False
   ):
    candfile = open(filename, 'r', encoding="latin-1", errors="
       replace")
    # First identify the length of the observation searched
```

It is recommended to invoke that function using python3 instead of python.

C.2.1 Ddplan -w

Ddplan optional parameter -w has a bug and you will get an error if you run it. Add or replace the necessary dedisp_.py file parts to look like this (not exactly like this, because it depends on your de-dispersion and data parameters). Remember that you are working on S-band (2 − $4GHz$) instead of L-band, so the mask file will be named as ”Sband_rfifind.mask”.

```
from builtins import zip
from builtins import range
import os
import numpy as np

def myexecute(cmd):
    print("’%s’"%cmd)
    os.system(cmd)

# By default, do not output subbands
outsubs = False

nsub = 32

basename = ’Lband’
rawfiles = ’psr-202504071014.fil’

# dDM steps from DDplan.py
dDMs        = [3.0, 5.0]
# dsubDM steps
dsubDMs     = [72.0, 120.0]
# downsample factors
downsamps   = [np.int64(4), np.int64(8)]
# number of calls per set of subbands
subcalls    = [5, 2]
# The low DM for each set of DMs
startDMs    = [0.0, 360.0]
# DMs/call
dmspercalls = [24, 24]


# Loop over the DDplan plans
for dDM, dsubDM, dmspercall, downsamp, subcall, startDM in zip(dDMs,
   dsubDMs, dmspercalls, downsamps, subcalls, startDMs):
    # Loop over the number of calls
    for ii in range(subcall):
        subDM = startDM + (ii+0.5)*dsubDM
        loDM = startDM + ii*dsubDM
        if outsubs:
            # Get our downsampling right
            subdownsamp = downsamp // 2
            datdownsamp = 2
            if downsamp < 2: subdownsamp = datdownsamp = 1
            # First create the subbands
            myexecute("prepsubband -sub -subdm %.2f -nsub %d
               -downsamp %d -mask Lband_rfifind.mask -o %s %s" %
                      (subDM, nsub, subdownsamp, basename, rawfiles))
            # And now create the time series
            subnames = basename+"_DM%.2f.sub[0-9]*"%subDM
            myexecute("prepsubband -lodm %.2f -dmstep %.2f -numdms %d
                -downsamp %d -mask Lband_rfifind.mask -o %s %s" %
                      (loDM, dDM, dmspercall, datdownsamp, basename,
                         subnames))
        else:
            myexecute("prepsubband -nsub %d -lodm %.2f -dmstep %.2f
               -numdms %d -downsamp %d -mask Lband_rfifind.mask -o %s
                %s" %
                      (nsub, loDM, dDM, dmspercall, downsamp,
                         basename, rawfiles))
```

C.3 RAEGE antennas addition in PRESTO

RAEGE 13m at Yebes and Santa Maria sites (RAEGYEB and RAEGSMAR) were requested to be officially included in PRESTO and TEMPO. While the requests are being reviewed, you can use a modified local version. You will need to modify all following files to upgrade your local version.

- In TEMPO: TEMPO/obsys.dat
- In PRESTO: PRESTO/SRC/misc_utils.c
- In PRESTO: PRESTO/SRC/polycos.c
- In PRESTO: PRESTO/SRC/sigproc_fb.c
- In PRESTO: PRESTO/bin/get_TOAs.py
- In PRESTO: PRESTO/python/presto/polycos.py
- In PRESTO: PRESTO/python/presto/sigproc.py

Scott Ramson, the PRESTO software creator, detailed some of the antenna adding steps on PRESTO's FAQ, but here we will be modifying more files.

C.3.1 TEMPO antennas database:obsys.dat

TEMPO will provide the correction polynomials (Doppler effect, Relativistic effects, etc..) in function of the antenna coordinates, so make sure that the coordinates are correct. At the end of .dat file add the text that is shown below and make sure that the one letter code (l,j and p) is not occupied by other antenna. If it is necessary, change the code to another letter or use no letter(use "-"). If TEMPO add the shown modifications, you MUST NOT change these.

```
 4848761.7579   -261484.0570      4123085.1343    1   YEBES40M        l   YS
```

```
4848831.021   -261629.388     4122976.576    1  RAEGYEB       j  YJ
4618524.302   -2166020.720    3816270.345    1  RAEGSMAR      p  SA
```

If in the future it is needed to include a new site (like RAEGE at Canary Islands), use a flag value of 1 for XYZ coordinates and put the official names of the antennas and their two digit code (Don’t use the tabulator key or copy paste because you will get a nice error). Remember these, because you will need them for PRESTO modding.

C.3.2 PRESTO: misc_utils.c

Search “void telescope_to_tempocode” in the code, and go to the end of that observatories list. Put this between the last antenna and the geocenter:

```
//Here is the last antenna added (KAT-7 in my case)
    } else if (strcmp(scope, "yebes40m") == 0 ) {
      strcpy(obscode, "YS");
      strcpy(outname, "YEBES40M");
    } else if (strcmp(scope, "raegyeb") == 0 ) {
      strcpy(obscode, "YJ");
      strcpy(outname, "RAEGYEB");
    } else if (strcmp(scope, "raegsmar") == 0 ) {
      strcpy(obscode, "SA");
      strcpy(outname, "RAEGSMAR");
//Here is the geogenter code (Geocenter)
```

C.3.3 PRESTO:polycos.c

Search “/* Write tz.in */” in the code, and go to the end of that observatories list. Put this between the last antenna and the geocenter:

```
//Here is the last antenna added (KAT-7 in my case)
     } else if (strcmp(idata->telescope, "YEBES40M") == 0) {
        scopechar = ’l’;
        tracklen = 12;
    } else if (strcmp(idata->telescope, "RAEGYEB") == 0) {
      scopechar = ’j’;
      tracklen = ’12’;
    } else if (strcmp(idata->telescope, "RAEGSMAR") == 0) {
      scopechar = ’p’;
      tracklen = ’12’;
//Here is the geogenter code (Geocenter)
```

The parameter tracklen refers to the antenna tracking capabilities, use 12 for full tracking capabilities (12 = 12h of tracking).

C.3.4 PRESTO:sigproc_fb.c

Search “void get_telescope_name” in the code, and go to the end of the observatory list. Use this code as a model to do the same as before:

```
//Here is the last antenna added (KAT-7 in my case)
    case 66:
        strcpy(s->telescope, "YEBES40M");
        s->beam_FWHM = 2.0/ 3600.0 * beam_halfwidth(s->fctr, 40) ;
        break;
    case 67:
      strcpy(s->telescope, "RAEGYEB");
        s->beam_FWHM= 2.0/ 3600.0 * beam_halfwidth(s->fctr, 13.2);
        break;
    case 68:
      strcpy(s->telescope, "RAEGSMAR");
        s->beam_FWHM= 2.0/ 3600.0 * beam_halfwidth(s->fctr, 13.2);
        break;
    default:
        strcpy(s->telescope, "Unknown");
        s->beam_FWHM = default_beam;
        break;
    }
}

void get_backend_name(int machine_id, struct spectra_info *s)
{
// ... other backends
 case 66:
      strcpy(string, "YEBES-SDR");
      break;
//Here is the Unknown backend code
```

For the case number, use any unused number of your choice and use the telescope aperture in meters at (s->fctr, value) for FWHM beam calculation. Our backend is named “YEBES-SRD” and is the case 66, so this number should appear on the filterbank code (in the backend parameter block).

C.3.5 PRESTO:get_TOAs.py

Search “scopes = {” in the code, and go to the end of the observatory list. Use this code as a model to do the same as before:

```
//Here is the last antenna added (MeerKAT in my case)
          'YEBES40M': 'l',
          'RAEGYEB': 'j',
          'RAEGSMAR': 'p',
          'Geocenter': 'o'}
```

```
scopes2 = {’GBT’:’gbt’,
// ... other antennas code
          ’YEBES40M’: ’ys’,
          ’RAEGYEB’: ’yj’,
          ’RAEGSMAR’: ’sa’,
          ’Geocenter’: ’coe’}
```

Make sure that you have used lower case letters for the antenna code.

C.3.6 PRESTO:polycos.py

Search “#telescope_to_id = {” in the code, and go to the end of the observatory list. Use this code as a model to do the same as before:

```
# Telescope name to TEMPO observatory code conversion
//Here is the last antenna added (CHIME in my case)
                   "YEBES40M": ’1’, \
                   "RAEGYEB": ’j’, \
                   "RAEGSMAR": ’p’, \
                   "Geocenter": ’0’, \
                   "Barycenter": ’@’}

# TEMPO observatory code to Telescope name conversion
id_to_telescope = {’1’: "GBT", \
// ... other antennas
                   ’1’: "YEBES40M", \
                   ’j’: "RAEGYEB", \
                   ’p’: "RAEGSMAR", \
                   ’0’: "Geocenter", \
                   ’@’: "Barycenter"}

# Telescope name to track length (max hour angle) conversion
telescope_to_maxha = {"GBT": 12, \
// ... other antennas
                   "YEBES40M": 12, \
                   "RAEGYEB": 12, \
                   "RAEGSMAR": 12, \
// Geocenter code
```

C.3.7 PRESTO:sigproc.py

Search “telescope_ids = {” in the code, and go to the end of the observatory dictionary. Use this code as a model to do the same as before:

```
telescope_ids = {"Fake": 0, "Arecibo": 1, "ARECIBO 305m": 1,
                 "Ooty": 2, "Nancay": 3, "Parkes": 4, "Jodrell": 5,
                    "GBT": 6, "GMRT": 7, "Effelsberg": 8, "ATA": 9,
                    "SRT": 10, "LOFAR": 11, "VLA": 12, "CHIME": 20,
                    "FAST": 21, "MWA": 30, "MeerKAT": 64, "KAT-7": 65, "
                       YEBES40M": 66, "RAEGYEB": 67, "RAEGSMAR": 68}
ids_to_telescope = dict(list(zip(list(telescope_ids.values()), list(
   telescope_ids.keys()))))

machine_ids = {"FAKE": 0, "PSPM": 1, "Wapp": 2, "WAPP": 2, "AOFTM":
   3,
                "BCPM1": 4, "BPP": 4, "OOTY": 5, "SCAMP": 6,
                "GBT Pulsar Spigot": 7, "SPIGOT": 7, "BG/P": 11,
                "PDEV": 12, "CHIME+PSR": 20, "MWA-VCS": 30,
                "MWAX-VCS": 31, "MWAX-RTB": 32, "KAT": 64, "KAT-DC2":
                  65, "YEBES-SDR": 66}
```

As value, use the number of "case" that was defined previously in the sigproc_fb.c file [Section C.3.4].

C.3.8 Rebuilding PRESTO

The last step is to rebuild PRESTO. To do that, start by running “meson setup build –prefix=$VIRTUAL_ENV” You can also wipe the previous setup build of meson, for further details run twice the last command and a help text will be provided. Now jump to ”python check_meson_build.py” [Section C.1] and follow all the given instructions.

D PRESTO workflow commands: General approach

In this section the necessary commands for the complete workflow will be detailed. You will find a very similar command history in [Ransom 2012b].

```
#Read the .fil file, and keep the sample rate value, number of
   channels, frequency and bandwidth.

readfile dataname.fil

#Now run rfifind, replacing dataname.fil by the name of your pulsar
   data.

rfifind -time 1.0 -o Sband dataname.fil

#Then use prepdata
prepdata -nobary -o Sband_topo_DM0.00 -dm 0.0 -mask Sband_rfifind.
   mask dataname.fil

#Use explore data to see the time series
```

```
exploredat Sband_topo_DM0.00.dat

#Use realfft perform a FFT and then see it using explorefft

realfft Sband_topo_DM0.00.dat
explorefft Sband_topo_DM0.00.fft

#Use accelsearch to search and characterize RFI

accelsearch -numharm 4 -zmax 0 Sband_topo_DM0.00.dat
```

Now make a .birds file as it was discussed before [Section 4.1.3], name it as“Sband.birds”. You can use a text editor to make that file using the content of “Sband_DM0.00_ACCEL_0” (text) file. Use bary value (a barycenter frequency) as 0 define it as topocentric (RFI is on the Earth). Note that #harm=Num Harm , and grow=1 is that RFI is growing in z (FFT ’r’ (bin)) in function of harmonics. For width use the FFT ’r’ (bin) value of the first harmonic (or if the frequency don’t grow, use the biggest value). Use as a model the .birds file shown below.

```
#-------------------------------------
# Freq   Width #harm  grow?  bary?
#-------------------------------------
22.7     0.1     2       0       0
50.1     0.2     4       1       0
```

Put that file on the PRESTO’s folder. DO NOT SKIP if you have seen strong low frequency noise on the FFT.

```
#Erase and convert the birds using simple_zapbirds, use in cn5g =
   your user name

python3 /home/cn5g/presto/bin/simple_zapbirds.py Sband.birds
   Sband_topo_DM0.00.fft
```

Now, make a de-dispersion plan using the commented command below as a model. Note that your sample rate time could be different that the defined on the software spectrometer, so use the value that appeared when you had used the command readfile. The r parameter is reduction factor on effective resolution time to speed up the search. Use -w to write an automatic de-dispersion script and then modify it according to [C.2.1].

```
#Make a de-dispersion plan using the commented command as model
#DDplan.py -d [max DM value to correct] -n [number of channels] -b [
   bandwidth in MHz] -t [sample rate time, use the intruction given
   on readfile] -f [frequency of observation (RF)] -s [number of
   subbands, use = number of channels] -r 0.5

python3 /home/cn5g/presto/bin/DDplan.py -d 100.0 -n 56 -b 50 -t
   0.000573 -f 3025.9 -s 56 -r 0.5 -w de-dispersion_plan
```

```
#Now clean the directory, make the FFTs and zaps the brids on each
   one.

mkdir subbands
mv *.sub* subbands
rm -f Sband*topo* tmp*
realfft *.dat
python3 /home/cn5g/presto/bin/simple_zapbirds Sband.birds *.fft

#Run accelsearch to search periodic signals on all files
ls *.fft | xargs -n 1 accelsearch -zmax 0

#Run accelsift to search pulsar candidates, and go to cands.txt to
   see them.

python3 /home/cn5g/presto/examplescripts/ACCEL_sift.py > cands.txt
```

Usually you will see on the candidates file a value similar to the expected one on DM and period of the pulsar. Due to observing/processing errors, so use that value to fold the data. If you don't see your pulsar or even you are seeing something that is not tabulated, the RFI could be "comfortable" at your receiver. So, clean well the birdies and avoid low elevations to make a deal with RFI.

```
#Run prepfold using the commented command as model to fold the time
   series (coarse approach)
#prepfold -accelcand [number of the candidate (see cands.text)]
   -accelfile Sband_DM[put here the DM of that candidate]_ACCEL_0.
   cand Sband_DM[put here the DM of that candidate].dat

prepfold -accelcand 1 -accelfile Sband_DM26.00_ACCEL_0.cand
   Sband_DM26.00.dat

#Finally, you can fold your raw data as it was shown on the shortcut.

prepfold -n 45 -nsub 56 -mask Sband_rfifind.mask -p 0.714 -dm 26.0
   yourdata.fil
```

Using prepfold command without -psr parameter, will require to define dm and period to target (use the found value on cands.txt) before folding data. n parameter is the number of bins in the profile, use a value lower than nsub to avoid artifacts (a warning will be displayed if the value is similar).

-Giant pulses:

```
#Run single_pulse_search to search single pulses (use -b to search
   also big pulses)
```

```
/home/cn5g/presto/bin/single_pulse_search.py -b *.dat
```