Design and Construction of a Multi-element Phased Array Radio Interferometer

Juan Sebastián Hincapié Tarquino

Universidad Nacional de Colombia
Facultad de Ciencias
Observatorio Astronómico Nacional
Bogotá, Colombia
2023

Design and Construction of a Multi-element Phased Array Radio Interferometer

Juan Sebastián Hincapié Tarquino

Tesis presentada como requisito parcial para optar al título de:
Magister en Ciencias - Astronomía

Director:
Prof. Benjamín Calvo Mozo
Observatorio Astronómico Nacional
Facultad de Ciencias

Co-Director:
Juan Carlos Martínez Oliveros, PhD.
Space Sciences Laboratory
University of California - Berkeley

Línea de Investigación:
Astrofísica Solar
Grupo de Investigación:
Group of Solar Astrophysics (GoSA)
Grupo de Astronomía, Astrofísica y Cosmología

Universidad Nacional de Colombia
Facultad de Ciencias
Observatorio Astronómico Nacional
Bogotá, Colombia
2023

Do not go gentle into that good night,
Old age should burn and rave at close of day;
Rage, rage against the dying of the light.

Though wise men at their end know dark is right,
Because their words had forked no lightning they
Do not go gentle into that good night.

Good men, the last wave by, crying how bright
Their frail deeds might have danced in a green bay,
Rage, rage against the dying of the light...

Dylan Thomas

To my mother and my grandmother...

Acknowledgements

First and foremost, to my mother and my grandmother, whose advice, love and constant support from day one gave me the strength and hunger to take all the gigantic steps that led me to where I am today, so this thesis, as well as everything I do, will always be for them, with the utmost love I can muster.

To my supervisors. Professor Benjamín, for welcoming me with open arms since I was a wee little engineering student, for your friendship, guide and advice on a daily basis, for being my mentor and, for lack of a better term, my academic father. To Juan Carlos, also for your guide and support, for helping me identify my strengths and for trusting that I had what it took to pull off this behemoth of a project. Also my total appreciation and gratitude to Professor Javier Araque, for accepting the supervision of my Bachelor's thesis and for being a part of this project since its early stages.

To Fernanda, for over a decade of being my best friend, advisor, right hand, and extended family. All my love to you. To many more birthdays, art and magnets in our fridges!

To Islena, for always being there with your big smile and even bigger heart, for trying to help me think on how to do things better, for being the head, heart and soul of everything, and for making absolutely everything possible (even if you don't want to admit it). Thank you. Always.

To Javier, for helping every day, either with the most ridiculous or the most complicated things. Without your selflessness and your support, none of this would have been done. Also to Santiago and Jimmy for helping in the early concept, focusing the design and finding the way to get all things necessary to build it all. To Professors Santiago, Leonardo and Giovanny, also for your help and encouragement throughout this road. Also a special mention to Jack Hickish for his collaboration, troubleshooting and helping me understand why things didn't work, and how they should work.

To my dearest Catalina, for all the poetry and fantastic food, for your laughter, love, encouragement and everything in-between. To all my closest friends, Alexandra, Sebastián, Nancy, Brian, Saida, Tatiana, Érika, Valeria, Wilmar, Jose Iván, because somehow you all also have a hand on this. Also a mention of gratitude to the Group of Solar Astrophysics, past and present.

To anyone I may have forgot to mention (because there is no more room left), thank you all.

Abstract

This document presents the implementation and optimization of a novel design for a three-element radio interferometer with a time correlation system, to observe solar radio emissions in a bandwidth centered on the Neutral Hydrogen line, or HI line, at 1.42 GHz.

For the commissioning phase, is proposed to conduct astronomical observations along with a series of measurements in order to determine the sectors in the sky that present lower radio interference, as well as to verify the radio interferometer's proper operation.

Also, this novel design, opens the door for conducting new radio emission studies at the Observatorio Astronómico Nacional, with locally collected data, being a scientific research tool that can be extended beyond Solar Astrophysics to other cosmic radio sources, thus contributing to the development of a growing discipline in Colombia. Additionally, is intended that the instrument is another teaching resource and contributes to interdisciplinary collaborations in the country.

Keywords: Radio Astronomy, Interferometry, Correlator, Instrumentation, Solar Radio Bursts.

Resumen

Título: Diseño y construcción de un radio interferómetro de varios elementos como arreglo en fase.

Este trabajo presenta la implementación y optimización de un diseño novedoso de un radio interferómetro de tres elementos con un sistema de correlación temporal para observar emisiones solares en la banda de radio, en un ancho de banda alrededor de la línea de Hidrogeno neutro o línea HI a 1.42GHz.

Para la parte de implementación y puesta a prueba se plantea realizar observaciones astronómicas en conjunto con series de mediciones para determinar los sectores del cielo con menor interferencia así como verficar el correcto funcionamiento del radio interferómetro.

Además, es un diseño novedoso de instrumento, que abre la puerta para realizar en el Observatorio Astronómico Nacional nuevos estudios de emisión en radio, con datos recolectados localmente, empleándose como herramienta de investigación científica cuyo uso pueda extenderse más allá de la Astrofísica Solar hacia otras fuentes cósmicas de radio, contribuyendo al desarrollo de una disciplina que está surgiendo en Colombia. También, se pretende que el instrumento sea una forma de apoyo pedagógico y de fomentar colaboraciones interdisciplinares en el país.

Palabras clave: Radio Astronomía, Interferometría, Correlador, Instrumentación, Estallidos Solares de Radio.

Table of Contents

1. Introduction

Motivation

For years, Scientists have been pursuing different ways to perform astronomical observations beyond the limited range of optical wavelengths, in light of broadening the kind of information available, so they can have a better understanding of physical processes in and around astrophysical objects. Historically, Radio astronomy has been an important branch of astronomy, enabling diverse studies such as mapping the structure of our galaxy, or in our case of interest, seeing different features on our Sun, with "another set of eyes".

Radio astronomy is a field of astronomy that has not seen significant development in Colombia, but is gradually growing thanks to some projects and ideas aiming, not only to shift the scales to a more viable way of performing astronomical studies in a country whose climate conditions are predominantly cloudy, but also to contribute to scientific and technological advances for future generations of Colombian astronomers, scientists, and engineers, in addition to the benefit it presumes for the nation and the society. With this in mind, a radio astronomy project was carried out as a Bachelor's Thesis in Electronic Engineering at Universidad Nacional de Colombia in 2016, and that is considered as the founding stage and corner stone for this current project as a logical continuation in this research field.

Main Goal

In this context, the main goal proposed for this project is to design and implement a multi-element radio interferometer, each one with collector dish, for observations at 1.42 GHz.

2. Background

2.1. Solar Radio Emission

For a better understanding of Solar Radio emission, is essential to know the different solar phenomena studied through optical methods. Solar radio emission can be divided in two types: Undisturbed and Disturbed. Undisturbed Radio Emission is that where average physical conditions, such as temperature or density, fairly describe the state of the solar atmosphere in any given time. On the other hand, Disturbed Radio Emission origins in discrete regions that differ significantly from average conditions, and are commonly associated to Active Regions, Sunspots, faculae, filaments, transitory flares, prominences and other disturbances associated to Radio Bursts. All variable phenomena in the Sun can be considered as a part of Solar Center of Activity (CA), which extends to all layers of the Solar Atmosphere, from the photosphere to the corona [14].

2.1.1. Overview of Radio Emission Processes in the Solar Atmosphere

Radio emission processes in the Sun and solar wind span over more than 6 decades in frequency, from below 30 kHz (typical plasma frequency at 1 Astronomical Unit [1]) to just over 30 GHz. Over this spectral range, most of the diverse and complex emission mechanisms can be somewhat explained in terms of three characteristic frequencies, depending in physical parameters such as electron density (n_e), electron temperature (T_e) or magnetic field (B). These three frequencies are the *plasma frequency* (Eq. 2-1), the *electron gyrofrequency* (Eq. 2-2) and the frequency at which the *free-free (Bremsstrahlung) emission reaches optical depth unity* (Eq. 2-3) [7].

$$\nu_P \approx\ 8.98 \times 10^3\sqrt{n_e}\ [\text{Hz}] \tag{2-1}$$

$$\nu_B =\ 2.8 \times 10^6 B\ [\text{Hz}] \tag{2-2}$$

$$\nu_{\tau_{ff}=1} \approx\ 0.5 n_e T_e^{-3/4} L^{1/2}\ [\text{Hz}] \tag{2-3}$$

[1] 1 AU $\approx 1.5\times10^{11}$m, is the mean Earth-Sun distance.

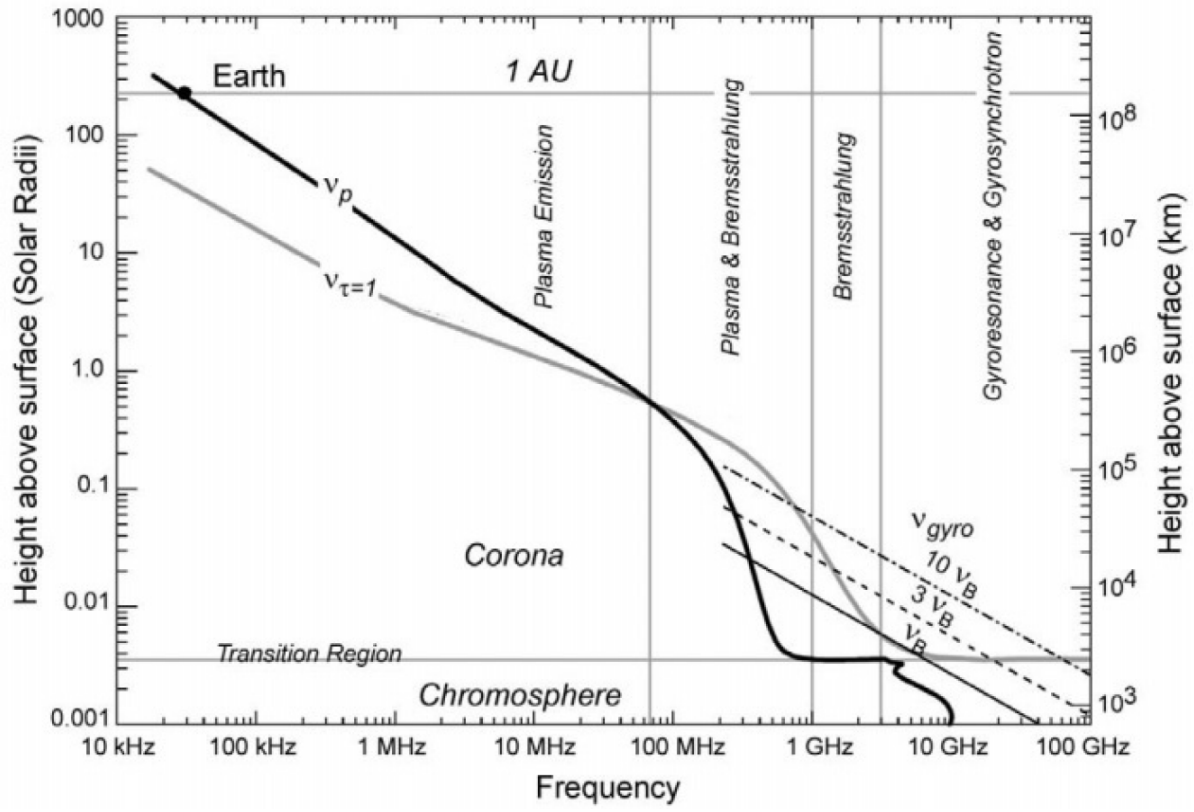

Figure 2-1.: Characteristic radio frequencies for the Solar Atmosphere. Taken from [7].

where L is the relevant scale length in cm for free-free emission (the density scale height in the case of an isothermal corona). The relation of each of these frequencies to height above the photosphere in the solar atmosphere is shown in figure **2-1**, where the frequency at a higher level is the frequency determined by the emission mechanism, according to the plasma conditions. From 30 kHz to ∼100 MHz the dominant mechanism is plasma emission, at heights greater than $0.2R_{\odot}$ above the photosphere. For an in-depth discussion on this topic, see [7].

The lines at the lower right corner represent the gyrofrequency and its harmonics. Emission occurs typically at the third harmonic ($\nu = 3\nu_B$), which lies above the absorption frequency given at the $\tau_{ff} = 1$ level, between 1 - 2 GHz up to 20 GHz. During bursts, gyroemission is usually at $\nu = 10\nu_B$ and can extend to 800–900 MHz (Decimetric Range) [7].

2.1.2. Types of Radio Emission

Radio emission from the sun is present in various forms at different frequencies, each varying slightly from another depending on the source. It consists on Quiet Sun Emission, Slowly Varying Component (or S-Component) and Solar Radio Bursts, which were classified by Kundu in merely taxonomical categories, according to their corresponding shape in radio spectrum [1].

The following list describes some of these radio sources :

1. **Quiet Sun:** A low level of solar activity, below which radio emission never falls for long periods of time, with little or none visible sunspots. Also, part of this emission is related to thermal continuum emission during Solar Minimum [1, 14].

2. **The Slowly Varying Component:** (Or S-Component) It is a component of daily variation, correlated with the transit on the solar disk of structures such as active regions or streamers. Its emission is a thermal emission associated to high density and magnetic field regions, in the vicinity of sunspots and chromospheric plages. Can be observed in the range of 500MHz to 10 GHz (λ = 3 - 60 cm) [14, 17].

3. **Type I Radio Bursts:** A radio storm consisting on several short bursts, each spanning from tenths of a second up to 15 seconds, with variable intensity. This storm may last several hours or even days. The origin of this type of emission is largely unknown but short bursts appear to be related to plasma emission, and the Type I Radio Storm could be related to continuous reconnection above an active region [1, 6].

4. **Type II Radio Bursts:** Caused by the perturbation on the solar atmosphere of a shock wave travelling upwards from a large flare, propagating at velocities of 500 - 2000 km/s outward into the corona and interplanetary medium, and it can be caused by plasma ejected by the flare. This type of radio bursts have a narrow band emission and a slow drift from higher to lower frequencies, spanning several minutes, usually seen at the fundamental and second harmonic of plasma emission frequency [1, 6].

5. **Type III Radio Bursts:** Also narrow band bursts, associated with transient phenomena on or around an Active Region on the Sun's surface. Type III radio bursts differ from Type II bursts, since their drift in frequency is in the order of seconds. Type III bursts are caused by high speed electron streams (or beams), propagating from the lower parts of the Corona into the interplanetary medium [1, 6].

6. **Type IV Radio Bursts:** Broadband Continuum emissions with a maximum peak just after a flare has erupted on the solar surface, due to either plasma emission or gyrosynchrotron emission. Can last for several hours. There's also a Moving Type IV radio burst, which has a similar cause but is entrained in a CME or an expanding arch [1, 6].

7. **"U" Radio Bursts:** Also usually called "*Castelli-U*". This are fast bursts with a typical duration of seconds, and a variation on wavelength, increasing and decreasing. Associated to flares and transient phenomena near active regions, with an origin similar to Type III bursts [1].

This classification of Radio bursts is summarized in table **2-1** and can be seen in figure **2-2**.

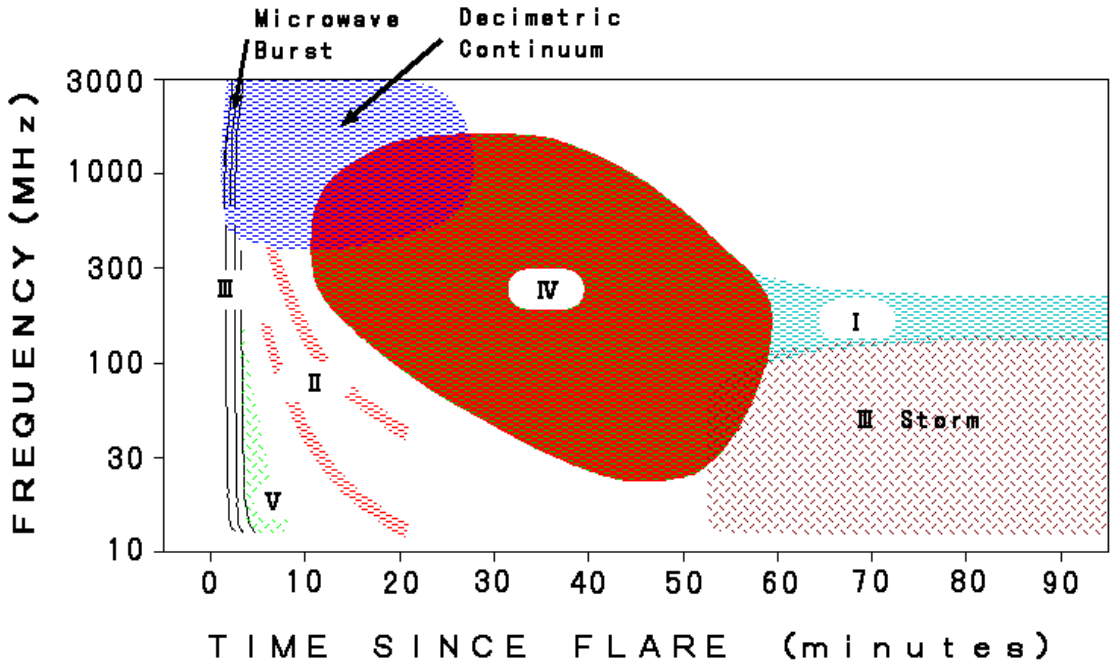

Figure 2-2.: Schematic diagram showing the Classification of Solar Radio Bursts. Taken from NICT (http://sunbase.nict.go.jp/solar/denpa/hiras/types.html)

2.2. 21 cm Radio Emission

2.2.1. The HI Line

Hydrogen is the most abundant element in the Interstellar Medium (ISM), comprising about 75 % of all baryonic matter. The structure of the interstellar medium is irregular, with some large low-density regions, where Neutral hydrogen (HI) atoms are abundant, and some more complex clouds with additional elements present, such as dust grains or different molecular compounds. Additionally, this medium is not in a state of equilibrium, since the physical state varies considerably between regions, due to the gas temperature dependence on either local heating or cooling processes [3], [25].

These neutral hydrogen atoms are detectable in the HI spectral line, which is the transition between the hyperfine structure levels $1^2S_{1/2}$, F=0 and F=1 of neutral hydrogen. This arises from the fact that the electron and proton have a particular direction of spin, and the energy of the configuration where the relative spins are in parallel (or aligned) is slightly higher from when the spins are in antiparallel (or in opposite directions). When the atom changes from one state to the other, a photon is emitted with a wavelength λ=21.1 cm (ν_{10}=1420.405 MHz) [3], [5], [8], [25].

In this case (a gas of neutral atoms), a selection rule suppresses transitions where the change in

Table 2-1.: Classification of Solar Radio Bursts. Taken from Space Weather Services, Australian Bureau of Meteorology (http://www.sws.bom.gov.au/World_Data_Centre/1/9/3)

Type	Characteristics	Duration	Frequency Range	Associated Phenomena
I	Short, narrow-bandwidth bursts. Usually occur in large numbers with underlying continuum.	Single Burst: $\sim$ 1s Storm: Hours - Days	80-200 MHz	Active Regions, Flares, Eruptive prominences
II	Slow frequency drift bursts. Accompanied by a (usually) stronger second harmonic	3 - 30 min	20-150 MHz (Fundamental)	Flares, Proton emission, MHD shock waves
III	Fast frequency drift bursts. Occur singularly, in groups or storms. Accompanied by a second harmonic	Single Burst: 1 - 3 s Groups: 1 - 5 min Storm: min - Hours	10 kHz -1 GHz	Active Regions, Flares
IV	Stationary Type IV. Broadband Continuum with fine structure	hours- days	20 MHz - 2 GHz	Flares, Proton emission
	Moving Type IV. Broadband, slow frequency drift with smooth continuum	30 min - 2 hours	20 - 400 MHz	Eruptive prominences, MHD shock waves
	Flare Continua. Broadband, smooth Continuum	3 - 45 min	25 - 200 MHz	Flares, Proton emission
V	Smooth, short-lived continuum. Follows some type III bursts.	1 - 3 min	10 - 200 MHz	Active Regions, Flares

orbital angular momentum is null, and since both F=1 and F=0 states have zero angular momentum, the F=1 state cannot be excited by ordinary radiation [8]. However, the F=1 state can be excited by collisions and the return to the F=0 ground state can be detected [8]. The probability for this forbidden transition is $A_{10} = 2.868 \times 10^{-15}$ s^{-1}, which is about 10^{23} times smaller than the probability for an allowed optical transition [25].

With this line is possible to know the density distribution of HI in the ISM, to study the structure, temperature and motion of clouds containing hydrogen as well as the rotation of the Milky Way or the recession of other astrophysical objects [5], [8], [25].

2.2.2. 21cm Solar Radio Emission

At wavelengths close to $\lambda = 21$ cm, solar radio emission consists mainly of two components, both of thermal origin; one is thermal radiation from the Quiet Sun, which can be used to extract important information such as temperatures or electron densities from layers difficult to observe in the optical range of the spectrum. The second component is The Slowly Varying Component, which, as previously mentioned, is a radio emission closely correlated with active regions, plages or other structures localized in the solar atmosphere [15].

The study of the chromosphere is of direct interest for Solar Radio Astronomy, since it can be directly observed in radio waves in the mm and cm wavelength regions. For our case of study, at 1.4 GHz (21 cm) the radio emission originates from somewhere in the upper chromosphere [13], [10]. Christiansen, Warburton & Davies (1957) conducted a detailed study of bright regions at 21cm with a grating interferometer, showing that these regions lay in the solar atmosphere at an average height of 25000 km above the photosphere, with sizes between 3' and 10' and temperatures close to 2×10^6 K. They discovered that the radio sources were always associated with plages and, when resolved, had a similar size. Additionally, Christiansen *et. al.* (1960) showed that at $\lambda = 21$ cm the size and shape of a radio bright region (or "radio plage", as they call it) corresponds to the size and shape of the associated chromospheric plage [12].

3. State of the Art

3.1. Other HI Radio Observatories

In the first part of the project, we considered other instruments around the world, dedicated to observe the 21 cm radio emission, either solar, stellar, galactic or some other type of astrophysical object.

3.1.1. Brazilian Decimetric Array (BDA)

The Brazilian Decimetric Array (BDA) is a radio interferometer developed by the National Institute for Space Research (INPE, Brazil), located at the INPE campus in Cachoeira Paulista, Sao Paulo, Brazil. Each antenna element consists of a 4 meter diameter parabolic dish in an alt-azimuth mount [4], [19]. The BDA provides solar radio images, analyzed using spectral tomography techniques to be used for space weather forecasting. Furthermore, the BDA is also useful for galactic and extra-galactic research of the southern sky that is not accesible to other instruments such as the Karl Jansky Very Large Array (VLA) [4]. The project is developed in three phases:

- **Phase I:** Installation of an East-West five-element array with dual polarized log-dipole feeds [19], operating in the frequency range of 1.2 - 1.7 GHz. The array was laid out over a total 216 m distance, getting a spatial resolution of $\sim$3 arcmin at 1.5 GHz [4].
- **Phase II:** 21 additional antennas laid out over 252 meters in the East-West direction, and 9 laid out over 166 meters in southern direction, forming a T-array. Frequency range increased to 2.8 and 5.6 GHz [4].
- **Phase III:** Finally, 8 additional antennas will be laid out in East-West direction and 4 in South direction, baselines will increase to 2.5 km and 1.25 km in each respective direction. Spatial resolution increases to $\sim$4.5 arcsec at 5.6 GHz [4].

3.1.2. Karl G. Jansky Very Large Array (VLA)

The Karl G. Jansky Very Large Array (VLA) is a 27–element interferometric array, arranged along the arms of an upside-down Y. The VLA is located at an elevation of 2100 meters on the Plains of San Agustin, New Mexico, and is managed from the Pete V. Domenici Science Operations Center (DSOC) in Socorro, New Mexico. All 28 antennas (27 active and 1 spare) have a 25m diameter

and are equipped with a set of 8 receivers covering a range between 1-50GHz as follows: 1–2 GHz (L-band), 2–4 GHz (S-band), 4–8 GHz (C-band), 8–12 GHz (X-band), 12–18 GHz (Ku-band), 18–26.5 GHz (K-band), 26.5–40 GHz (Ka-band), and 40–50 GHz (Q-band) [16].

There are four main antenna configurations, denoted D, C, B and A from smallest to largest, as well as an additional BnA configuration reserved exclusively for VLA Sky Survey observations. These configurations are performed by moving the component antennas over railroad tracks along each arm. The VLA completes one full cycle through all four configurations in approximately 16 months. The VLA's resolution is generally diffraction-limited, and thus is set by the array configuration and the observing frequency. The instrument is sensitive only to structures on a range of angular scales between the diffraction limit (the smallest angular scale detectable) and a "Largest Angular Scale"(which depends on the fringe spacing formed by the shortest baselines in the configuration) [16]. Table **3-1** shows the resolution and the scale at which severe attenuation of large-scale structure occurs, the maximum (B_{Max}) and minimum (B_{Min}) antenna separations, the approximate synthesized beam size (full width at half-power) and the largest angular scale of detectable emission for the central frequency of the band of interest (L-band).

Table 3-1.: Configuration Properties for the Karl G. Jansky Very Large Array at L-Band (1.5 GHz). Taken from NRAO: VLA Resolution (https://science.nrao.edu/facilities/vla/docs/manuals/oss/performance/resolution).

Configuration	**A**	**B**	**C**	**D**
B_{Max} [km]	36.4	11.1	3.4	1.03
B_{Min} [km]	0.68	0.21	0.0355	0.035
Band	**Synthesized Beamwidth θ_{HPBW} [arcsec]**			
1.5 GHz (L)	1.3	4.3	14	46
	Largest Angular Scale θ_{LAS} [arcsec]			
1.5 GHz (L)	36	120	970	970

3.1.3. KAT-7

KAT-7 is a seven dish radio interferometer located in Northern Cape, South Africa. It was built as a precursor to the MeerKAT radio telescope array as an intent to demonstrate South Africa's ability to host the Square Kilometer Array (SKA); However, it has become a pioneering scientific instrument by itself [20]. It consists on seven dishes with 12 meters in diameter, made out of fibre glass, with baselines ranging from 26 m to 185 m. The receivers operate in a frequency range between 1.2 and 1.95 GHz, and are cryogenically cooled to about 70 K in order to reduce radio noise and increase the system's sensitivity [20], [22]. KAT-7 has 5 correlator modes, as shown on table **3-2**.

Table 3-2.: Correlator Modes for the KAT-7 radio interferometer at Northern Cape, South Africa. Taken from [22]

Mode	# Bands	Band Bandwidth	Channel Bandwidth
Wideband	1	256 MHz	390.625 kHz [1]
8k Wideband	1	256 MHz	48.8 kHz [2]
HI Galaxy Clusters	1	400/16 = 25 MHz	25/4096 = 6.1 kHz
HI Galaxies / Maser Search	1	400/64 = 6.25 MHz	6.25/4096 = 1.525 kHz
Wideband	1	400/256 = 1.5625 MHz	1.5625/4096 = 381.4697 Hz

1. The channel bandwidth is obtained by dividing the IF Bandwidth (400 MHz) by the total number of channels (1024). Only 256 MHz of the IF Bandwidth is useable.

2. IF bandwidth (400 MHz) is divided by 8192 channels.

3.1.4. MeerKAT

MeerKAT consists of 64 antennas in an Offset Gregorian dish configuration, where the diameter for the main reflector is 13.5 m and for the sub-reflector is 3.8 m. The main reflector is made up of 40 aluminum panels mounted on a steel support framework. MeerKAT is a stand-alone world-class instrument, but also serves as a precursor for the Square Kilometer Array (SKA), and will be merged into the SKA1-Mid array. [23], [21].

MeerKAT's 64 dishes are distributed as follows:

- 48 of the antennas (75 %) are concentrated in the dense core area, where the shortest baseline is 29 m and the longest baseline is 1 km.
- An outer component containing the remaining 16 antennas (25 %), where the maximum baseline is 8 km.

For the specific case of the L-band receiver, it operates in the range between 900 and 1670 MHz. A digitizer is mounted on the receiver indexer, close to the L-band receiver, with a sample frequency of 1712 MHz and a 10 bit resolution [23].

4. Interferometer Design and Implementation

4.1. Feeding System: Yagi-Uda Antenna

The interferometric array will use a novel design of Yagi-Uda antennas manufactured in FR-4 Printed Circuit Board (PCB) designed specifically to operate in a bandwidth around the center frequency of 1.42 GHz (λ = 21 cm), consisting on a series of linear dipoles around the energized element, acting as parasitic radiators (4 as directors and 1 as a reflector). The dimensions for these elements, and the separation between them, are scaled by a factor multiple of the wavelength of interest, λ = 21 cm. The feeding system for the antenna is a modified Grounded Coplanar Waveguide with a balun transition connected to a folded dipole, in order to obtain a 50Ω input impedance and prevent mismatching with the transmission line. The resulting antenna is linearly polarized, according to the Axial Ratio measurement yielded in the simulation. The full antenna design is shown in figure **4-1**, and the complete design steps are found in [11] [1].

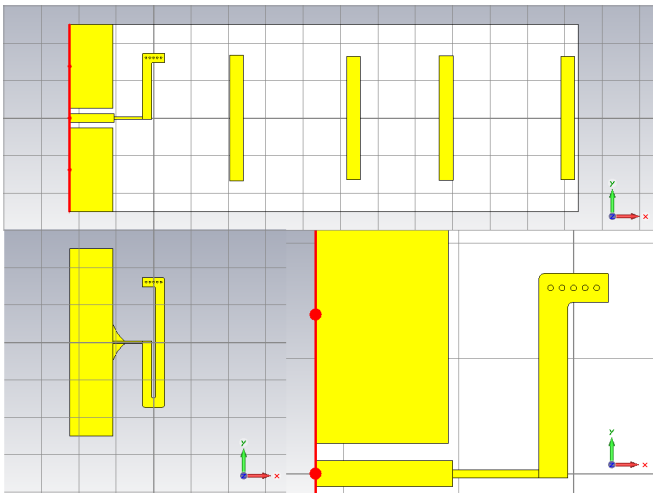

Figure 4-1.: Yagi-Uda Antenna design with Folded Dipole and Grounded Coplanar Waveguide feed. *Top*: Top Layer View; *Bottom, Left*: Bottom Layer View; *Bottom, Right*: Detail of Via-Holes on folded dipole. Simulation performed with CST Microwave Studio [11].

[1] A copy of the document can be accessed at: https://bit.ly/JSHT-BEng2016

(a)

(b)

Figure 4-2.: (**a**) Printed Yagi-Uda Antenna, fabricated with photolithography for radio interferometer implementation. *Top*: Top Layer View; *Bottom*: Bottom Layer View.
(**b**) Validation of Input Reflection Coefficient (S_{11} parameter) and Bandwidth for manufactured Yagi-Uda Antenna at the Electronics Engineering Laboratory - Universidad Nacional de Colombia [11].

Following the manufacturing process, the antenna operation was characterized with a Vector Network Analyzer (VNA) in the Electronics Engineering Laboratory at Universidad Nacional de Colombia (as shown on figure **4-2**) and in the Anechoic Chamber at Universidad de los Andes.

This novel antenna design was previously used to implement a two-element drift (or transit) solar radio interferometer, where each of the elements consists on a 2×2 array. This array configuration was implemented in order to increase the electrical length of the antenna without actually changing the dimensions of the elements that comprise it (directors, reflector and feed element), as well as increasing the directive characteristics of the antenna, such as realized gain and received power [11].

For each element, the 2×2 configuration, between Yagi-Uda antennas the frontal separation is of 0.75λ=15.83 cm and lateral separation of λ=21.1cm, as described in Hincapie Tarquino, 2016 [11], and is shown in figure **4-3**.

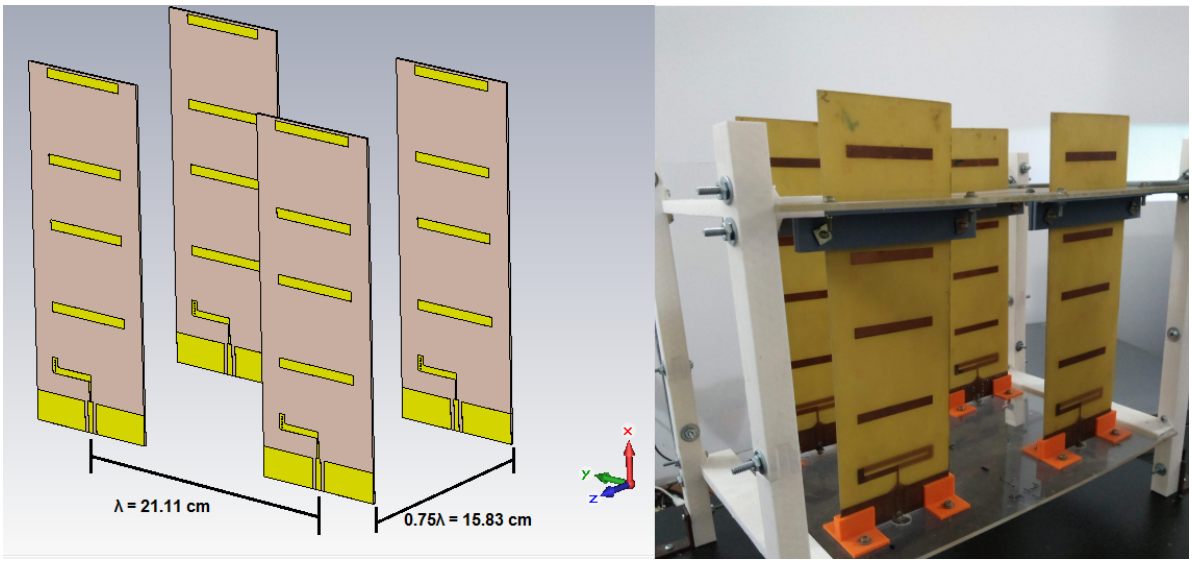

Figure 4-3.: Yagi-Uda antenna array configuration. *Left*: 2×2 schematic designed in simulation. Simulation performed with CST Microwave Studio; *Right*: Implemented 2×2 array element. Adapted from [11].

4.2. Antenna Element: Yagi-Uda Antenna and Reflector Dish

The concept of antenna element for this instrument is based on an offset-fed paraboloidal reflector dish, with a 1.2m diameter, where the feeding system is the previously described Yagi-Uda antenna. The reason for including the reflector dish is the enhancement in the directive characteristics of the antenna element, such as directivity and gain, as well as the narrowing of the main

lobe (Half-Power Beamwidth) in the radiation pattern. Comparison between the configurations for the previously described 2-element drift interferometer (isolated Yagi-Uda antenna and 2×2 antenna array) and the current design are shown in table **4-1**.

In the design stage we also considered the option of placing the 2×2 Yagi-Uda array as a feeding network. This was ultimately discarded since the reflector dish was sub-utilized given that this feeding network is highly-directive, considerably higher than the isolated antenna and with a much narrower beam, leading to a loss in directivity for the element. Simulation results for the Antenna element configuration of the 1.2m Reflector Dish with Printed Yagi-Uda antenna are shown in figure **4-4**.

Table 4-1.: Simulation Result Summary for Single Yagi-Uda Antenna and 2×2 Antenna Array with Offset-Fed Reflector Dish

PARAMETER	**Isolated Antenna without Reflector Dish** [11]	**2×2 Antenna Array without Reflector Dish** [11]	**Single Antenna with Offset-Fed Reflector Dish**	**2×2 Antenna Array with Offset-Fed Reflector Dish**
Input Reflection Coefficient (S11 Parameter)	-34.3583 dB (at 1.422 GHz)	-34.3583 dB (at 1.422 GHz)	-24.3591 dB (at 1.422 GHz)	-24.3591 dB (at 1.422 GHz)
Half-Power (-3 dB) Bandwidth	317.6 MHz	317.6 MHz	321 MHz	321 MHz
Directivity	10.99 dBi	15.56 dBi	21.96 dBi	20.07 dBi
Gain (IEEE)	9.397 dB	13.97 dB	20.18 dB	18.29 dB
Realized Gain	9.394 dB	13.96 dB	20.16 dB	18.28 dB
Half-Power Beamwidth (HPBW)	48.8 deg	31.1 deg	13.2 deg	9.9 deg
Front-to-Back Ratio	18.928 dB	18.924 dB	24.583 dB	22.002 dB

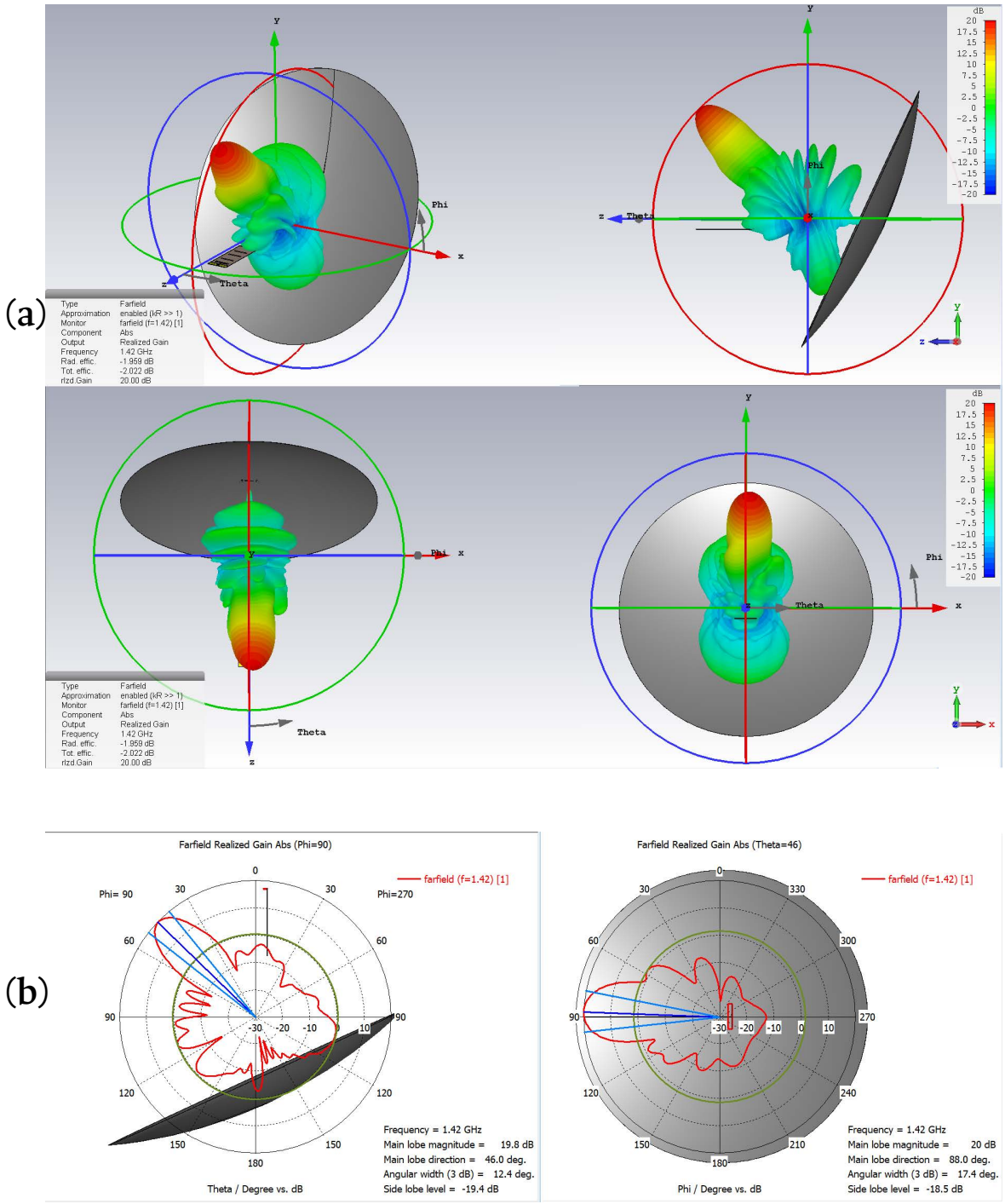

Figure 4-4.: Far-field radiation pattern for Antenna with 1.2m Offset-fed reflector dish. Realized Gain. (**a**) Full View. (**b**) Polar View (Side and front of dish). Simulation performed with CST Microwave Studio.

4.2.1. Pier design and Motor Assembly

In order to have a fully steerable antenna element we designed a pier support with 2 coupled stepper motors that enable pointing the antenna in any direction, one for azimutal motion and the other for elevation. For each motion we coupled a NEMA 23 Stepper Motor with a reducer gearbox, so we could obtain a higher resolution in motion without reducing the Pulse-Width Modulation (PWM), and therefore not losing torque in the motors.

The motor control stage for steering the antenna consists of a Master-Replica RS485 Half-Duplex protocol on Arduino, and a DM542T Digital Stepper Driver for each motor, as shown on the block diagram in figure **4-5**. There, the "replica" arduinos, digital drivers and regulated power supply are located in isolated boxes on the antenna pier supports, and the Master arduino is located next to the Control PC inside the building's observation dome control room.

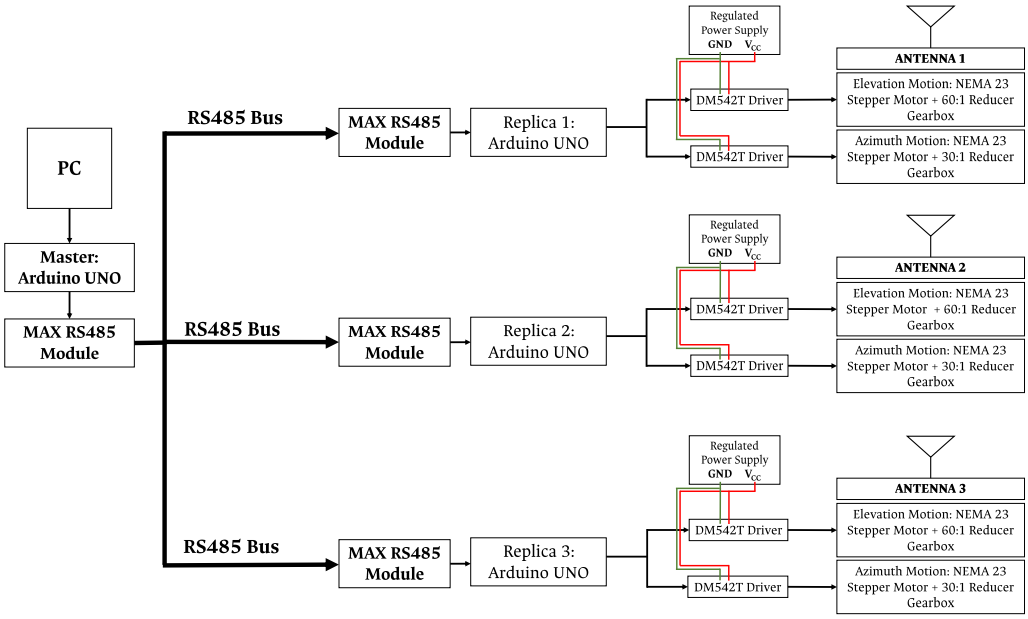

Figure 4-5.: Block Schematics for Antenna Motion Control Stage.

First, on the pier support was placed the stepper motor and gearbox for azimuth motion, where the extended axis in the gearbox is fixed, located in the center of the pier upper plate, so the motor moves on azimuth along with the upper section of the mechanism in a full 360° range.

A system of plates was submitted to custom machining for enclosure of the second gear box and placement of the dish support. Following this, the stepper motor was attached to the gearbox and placed on top of the first motor, in order to guarantee the elevation motion for the reflector dish, in a range between 24° and 114°, so that the main lobe for the offset-fed reflector dish

covers between 0° and 90° in elevation, according to the simulation design shown on figure **4-6**, showing some side and perspective views of the antenna element with the reflector dish in different positions, ensuring that we have the full range of motion for each antenna element. The process for assembling the motor system is shown in figure **4-7**, and the full installed antenna element is shown in figure **4-8**.

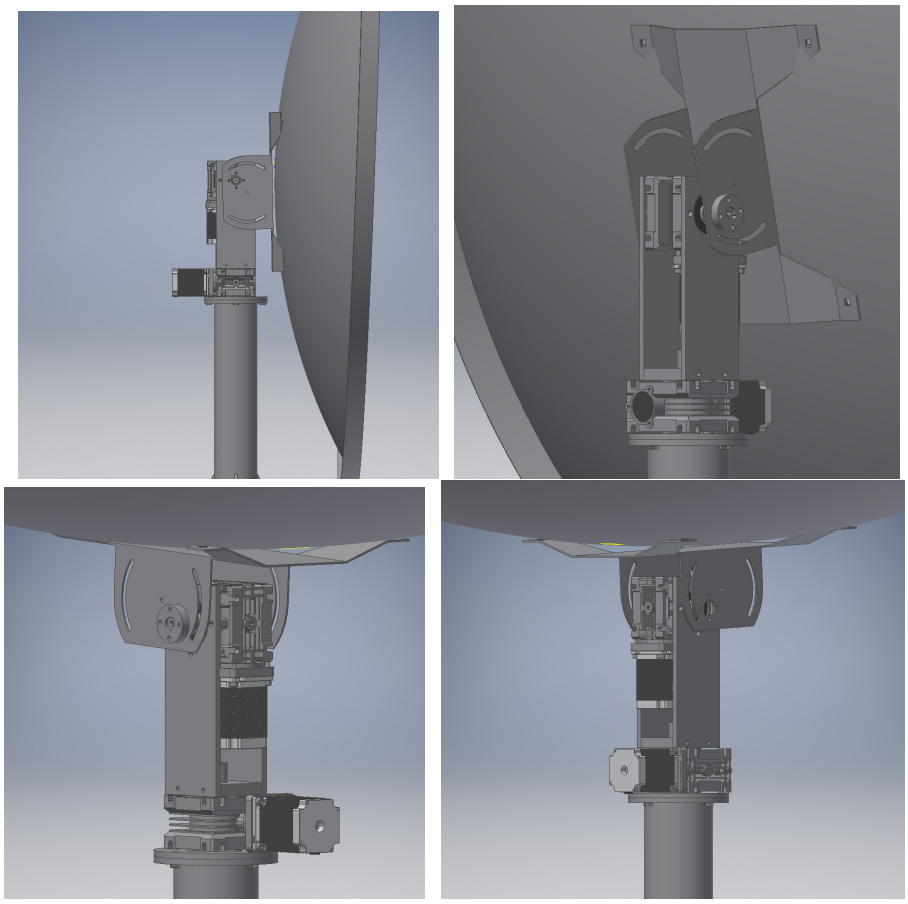

Figure 4-6.: Simulation design for Motor assembly, in order to verify the full motion range for each antenna element. Side and perspective views with the reflector dish pointing to the horizon (***Top, Left*** & ***Top, Right***) and to the zenith (***Bottom, Left*** & ***Bottom, Right***).

Figure **4-9** shows the implemented Master and Replica modules for the control stage. This set-up provides enough motion resolution for the antenna elements to appropriately track the astronomical source within their Half-Power Beamwidth (HPBW). Therefore, with full-step operation for the stepper-motors and the reducer gear-boxes, we have a minimal motion of 216 arcseconds per step on the Azimuth axis, and 108 arcseconds per step on the Altitude axis. Additionally, the Master-Arduino module has two "enable" switches, one to enable the manual control for the system, and the other to block the verification signal coming from the "replicas". The corresponding control scripts are shown in Appendix A.

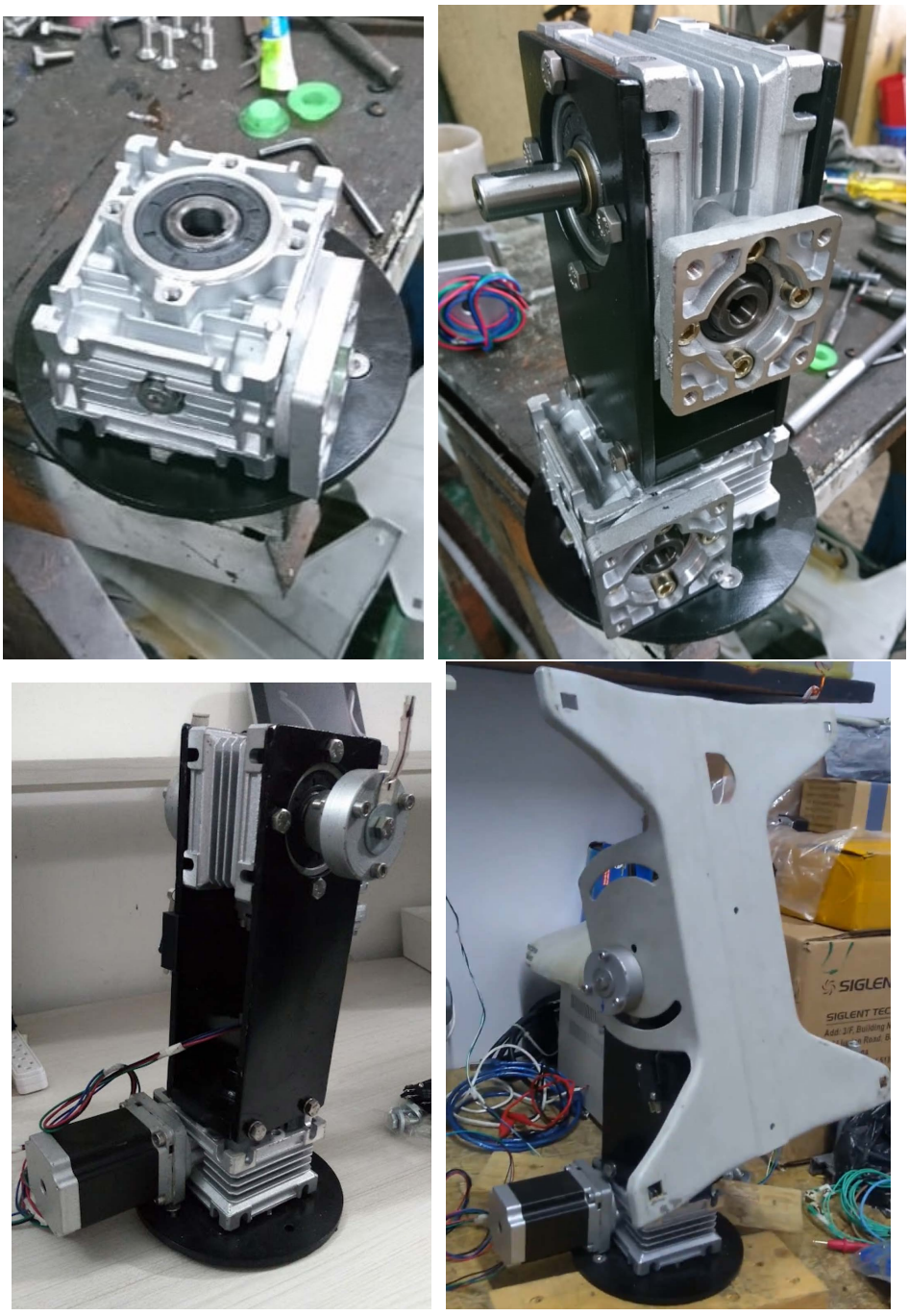

Figure 4-7.: Motor assembly process for antenna element. ***Top, Left:*** Gearbox for Azimuth motion. ***Top, Right:*** Assembled gearboxes and metal enclosures. ***Bottom, Left:*** Stepper motors attached to gearbox system. ***Bottom, Right:*** Assembled motor system with dish support plate.

Figure 4-8.: Installation process for each antenna element

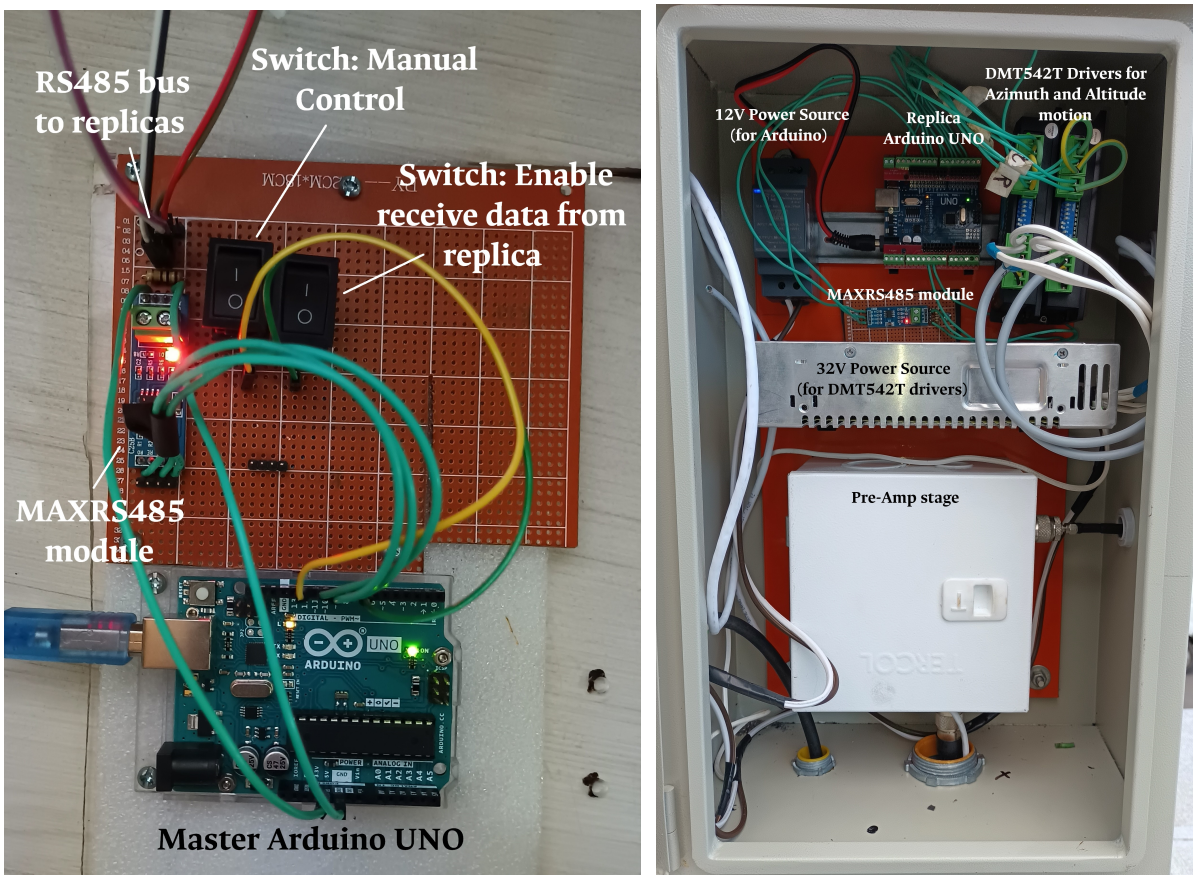

Figure 4-9.: Control system for antenna motion control stage. ***Left:*** Master Arduino UNO with MAXRS485 module and bus to all replicas; ***Right:*** Isolated box with Replica Arduino UNO, MAXRS485 module and DMT542T drivers for each antenna element

4.3. Array Implementation

The interferometric array consists of 3 steerable elements. For the implementation we considered the space available in the roof of the Observatorio Astronómico Nacional - Campus building, in a way that it would not interfere with other instruments installed there. The selected positions for the antenna elements are shown in figure **4-10**, with their corresponding baselines of 27, 33 and 17.5 meters. A fourth antenna element is added to perform calibration, pointing to a different region of the Sky. Since this element will not perform source tracking at any moment, it does not have the previously described stepper motor and reducer gearbox system.

Through a topographic field survey, using an optical level and a levelling rod, we measured the height differences between the three selected points in order to determine the total height of each pier support, so that the bases of the parabolic dishes were on the same plane. This with the purpose of minimizing the geometric delay for the signals received in each antenna.

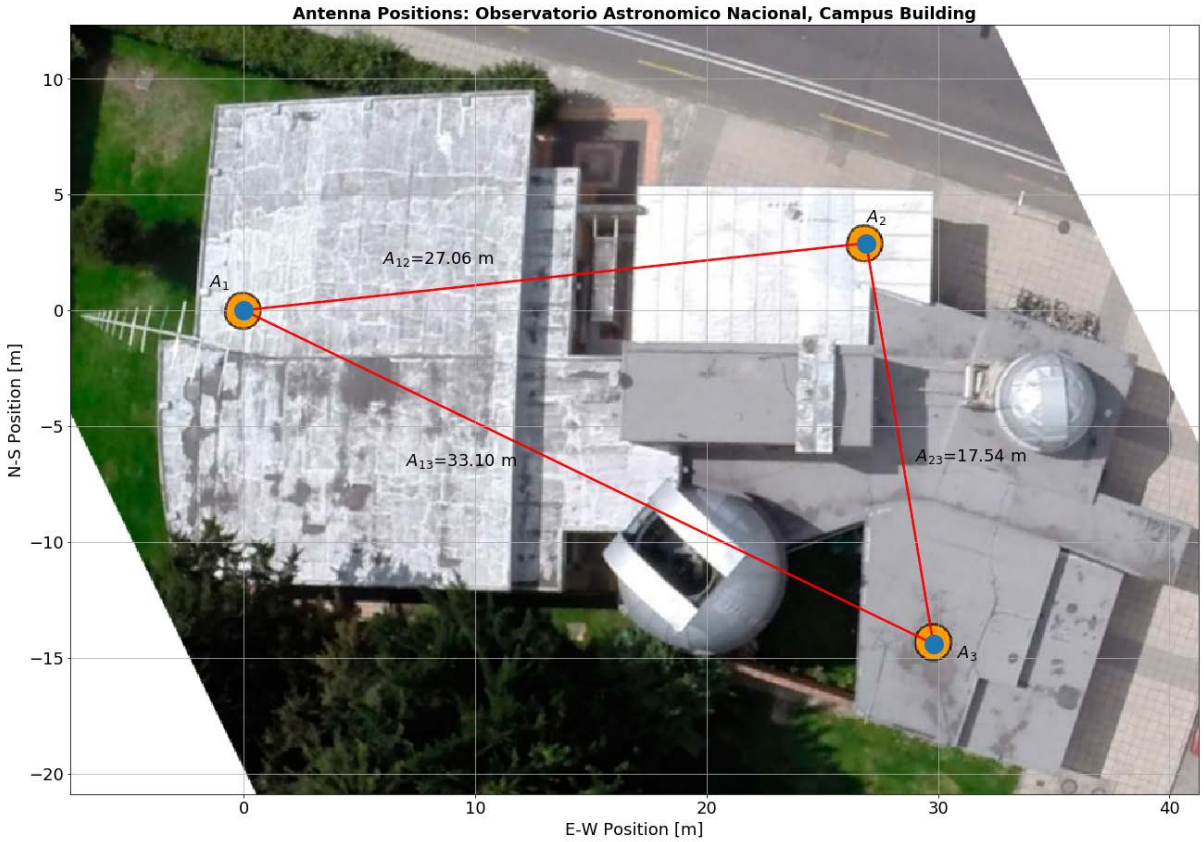

Figure 4-10.: Array location at Observatorio Astronómico Nacional, Campus Building - Universidad Nacional de Colombia.

With these baselines, we can estimate the some important parameters or spatial scales for the interferometric array, such as the angular resolution, as well as the Largest Angular Scale (LAS)

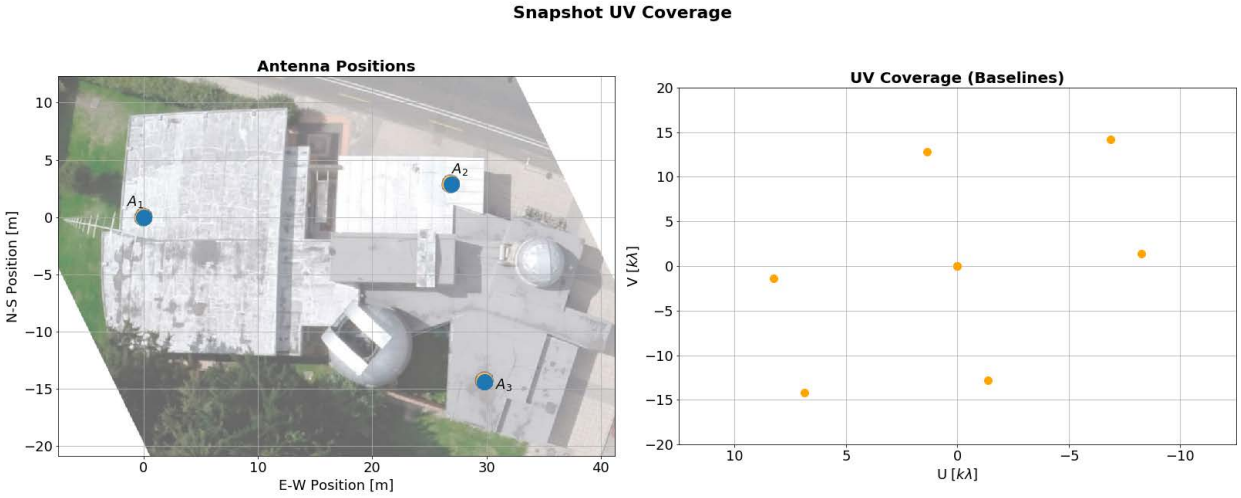

Figure 4-11.: Snapshot UV Coverage for the Interferometric Array.

structure). The angular resolution of the array depends on the observed wavelength (λ) and the maximum baseline, or separation between a pair of antennas. For our case, the interferometer's angular resolution is given by:

$$\theta_{AR} = \frac{\lambda}{B_{max}} = \frac{0{,}2111 \text{ m}}{33{,}1 \text{ m}} = 0.006377 \text{ rad} \approx 0.3654 \text{ deg} \tag{4-1}$$

The largest angular scale (LAS) structure is the structure which can be imaged reasonably well in full synthesis observations by an interferometer, and is given by the shortest baseline, as follows

$$\theta_{LAS} = \frac{\lambda}{B_{min}} = \frac{0{,}2111 \text{ m}}{17{,}54 \text{ m}} = 0.012035 \text{ rad} \approx 0.6896 \text{ deg} \tag{4-2}$$

These two parameters described in equations 4-1 and 4-2 determine the range of angular sizes to which out instrument is sensitive to: 0.3654 deg <θ <0.6896 deg.

Figure **4-11** shows the intereferometer's UV coverage during a snapshot (a single sample observation). This plane is a geometric plane defined for the analysis of interferometric observations, and it is tangent to the celestial sphere at the position of the astronomical object of interest. The axes u and v are set in the east-west and north-south directions, respectively. The axes are scaled using the observation wavelength as their unit of distance. This plane is constructed in order to represent the Fourier transform of the signal received from the source structure (visibility), from each projected baseline [3], [24], [25].

4.3.1. Interferometer Front-End: Pre-Amp Stage

The radiation coming from celestial sources is, in comparison, much weaker than man made signals (Radio Frequency Interference, RFI). Therefore, it is necessary to amplify the desired radio signal, to later "subtract" the noise from the signal, in order to have only our astronomical signal. This signal to noise ratio (SNR) allows the instrument to resolve radio emission coming from a given astrophysical object.

The design for the pre-amp stage shown on figure **4-12** is based on the amplification stage used for several systems of the e-Callisto Solar Spectrometer Network, such as the receiver at Reeve Observatory (Anchorage, Alaska) [18]. Additionally, since a similar pre-amplification stage is already being used for FiCoRI, an interferometer dedicated to solar observations at the Observatorio Astronómico Nacional - Campus Building [9], we can ensure that the pre-amp stage is stable and suitable for the purposes of our instrument.

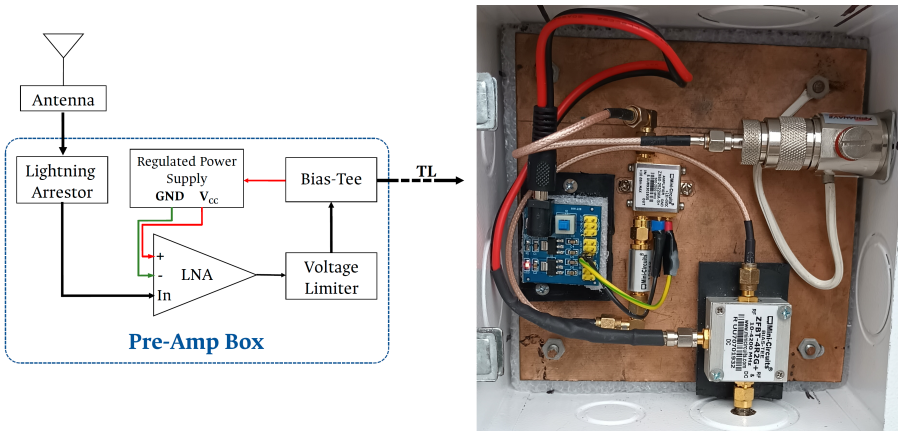

Figure 4-12.: ***Left:*** Block Schematics for Pre-Amp box for each antenna element. ***Right:*** Implemented Pre-Amp stage.

4.3.2. Interferometer Front-End: Downconversion Stage

For the correlation stage we used a Reconfigurable Open Architecture Computing Hardware (ROACH) board, which is a standalone FPGA processing board developed by the Collaboration for Astronomy Signal Processing and Electronics Research (CASPER) group[2]. It consists basically of a FPGA as a centerpiece, and a PowerPC subsystem running Linux to control the board, i.e., programming the FPGA and allow interfacing between the FPGA software registers and any external device. This ROACH-1 includes 2 Analog-to-Digital Converters (ADC's), with two channels each.

[2]ROACH - CASPER: https://github.com/casper-astro/casper-hardware/tree/master/FPGA_Hosts/ROACH

The ADC's in the ROACH-1 board used for the correlation stage have a maximum usable sample rate of 450 Msps for each of their inputs, 4 in total. Due to Nyquist sampling theorem, this gives us a maximum usable frequency of 225 MHz for each of the 4 inputs. Given these constraints of the ADC's inputs, we need to perform a downconversion of the signal received once is acquired, converting it from 1.42 GHz to a lower frequency. This process preserves all the information from the original signal at the band of interest, but simplifies its processing due to its lower frequency. The downconversion stage is implemented using the "Valon 2100 Downconverter for the 21cm band", a downconverter specifically design for operation at a band centered around 1.42 GHz, our frequency band of interest, and outputs a signal between 10-30 MHz. This downconverter consists of a dual-mixing stage (one at 1275 MHz and one at 125 MHz), each one with a local oscillator, feed with a reference signal provided by the "Valon 5009 Frequency Synthesizer" appropiately tuned for this purpose. Each output of the frequency synthesizer is connected to special 4-1 power splitters before the downconverter's oscillator inputs, with each of the first three being used for the observing antennas, and the fourth oscillator output feeds the signal coming from the calibration antenna. Block schematics for the design of the downconversion stage is shown in figure **4-13**.

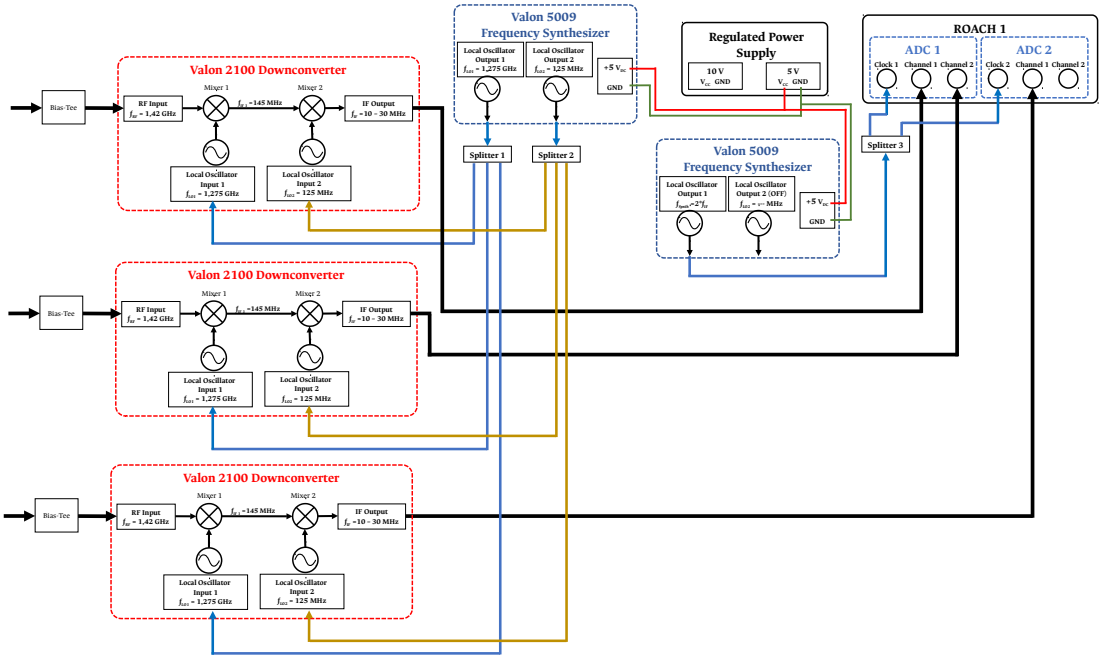

Figure 4-13.: Block Schematics for Downconversion stage.

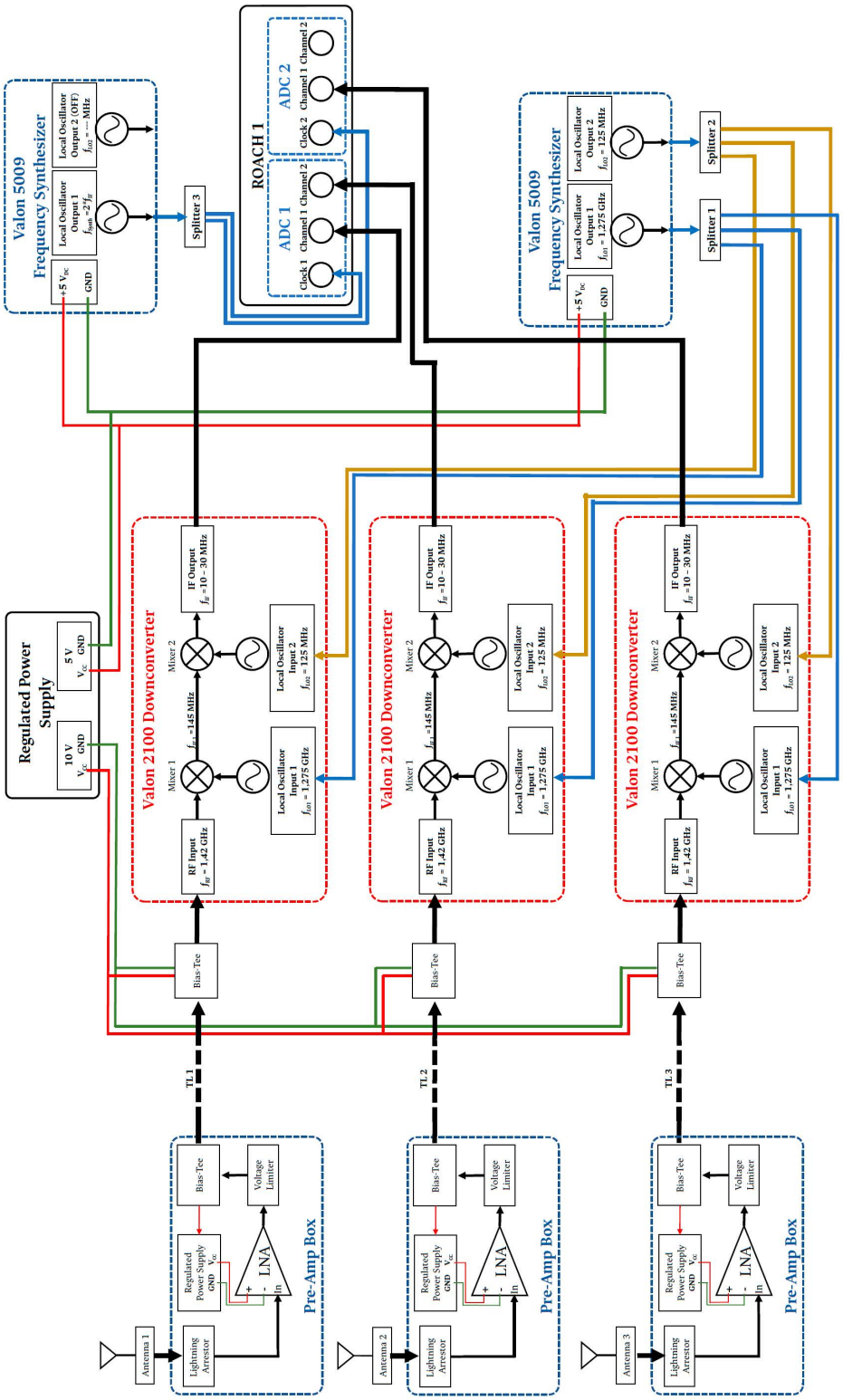

Figure 4-14.: Block Schematics for full Interferometric Array.

4.3.3. Interferometer Back-End: Data Acquisition and Full-Correlation implementation

The correlation stage, as previously mentioned, is implemented using a ROACH-1 board, whose centerpiece is a Xilinx Virtex 5 SX95t FPGA board, and also includes 2 ADC's, with two channels each. The appropriate programming of the FPGA is done using the toolflow developed by the CASPER (Collaboration for Astronomy Signal Processing and Electronics Research) group for several platforms using Matlab R2013b/Simulink and Xilinx ISE design suite. In particular, our design is based on their tutorial for a Wideband Pocket Correlator [3], using 4 antennas with 4 channels, 2 per ADC, but modified to suit our specific needs, particularly the frequency of the input signal coming from the downconversion stage. Such design can be envisioned either as a 2-input full Stokes correlator, or a 4-input single polarization correlator. For us it's the latter, since the Yagi-Uda antenna designed is a linearly-polarized antenna.

Even though the operational Half-power bandwidth for the implemented Yagi-Uda antenna is fairly big (close to 300 MHz around the central frequency) as shown on table **4-1**, to define the operational bandwidth of the instrument, we must also consider the sampling rate of the data. Specifically, this is the width of our frequency spectrum, which in turn will define the frequency resolution for the radio interferometer.

Considering the highest frequency coming out of the downconversion stage and into the ADC's (of 30MHz), we make our reference signal generated by our frequency synthesizer the same frequency of 30 MHz to sample out input signal. In that order, we can say that our operational bandwidth is effectively of 30 MHz. This bandwidth is split into a number of different channels during the correlation process, giving us the width of each frequency bin. This is the frequency resolution of the instrument, i.e., how precise you can measure a given frequency in your bandwidth. The FFT block outputs 1024 values in total, 512 cosine values (odd) and 512 sine values, so we are using only 512 channels for the sine values. So, the frequency resolution of our instrument is given by:

$$\Delta\nu = \frac{BW}{\text{No. of Channels}} = \frac{30\text{ MHz}}{512} = 58.6\text{ kHz} \tag{4-3}$$

Additionally, one should consider the time resolution of the instrument, which is is simply the spectral dump rate, and it depends on the science case behind it. If one is interested on studying short timescale events, the time resolution should be high, but if one interested in larger scale events or even deep-field surveys, a lower time resolution is enough. In the case of this project, we were able to compile a correlator file design that covered the specific bandwidth

[3] ROACH Tutorial 4: Wideband Pocket Correlator:
https://casper-toolflow.readthedocs.io/projects/tutorials/en/latest/tutorials/roach/tut_corr.html

and frequency resolution defined in equation 4-3, with a time resolution of 30 seconds for the instrument. This enables us to look at events in a relatively short time-scale. Since our case of interest is mainly solar, with such time resolution we are able to observe not only the continuum radiation from both the Quiet Sun and the effects from the S-Component, but we could also be able to observe the characteristic signature of Type III radio bursts, which are due to accelerated particles associated to eruptive events such as flares or coronal mass ejections (see table **2-1**). For the cases of observations beyond solar radio emission, this time resolution cannot be used to study transients with very short timescales, such as Fast Radio Bursts (FRBs), but is more than enough to perform 21cm Intensity Mapping, in order to characterize the underlying density of Hydrogen in a given region of space, such as our own galaxy.

The heart of the correlator design is a block which implements the fourier transform along with a polyphase filterbank. The Polyphase filter bank (PFB) techinque is a mechanism designed to compensate for two significant drawbacks from applying a fourier transform to an input signal, namely leakage (an input tone appearing in more than one frequency bin) and scalloping loss (loss in energy between frequency bin centres). The full detail for the implementation of such technique is given by Jayanth Chennamangalam on a dedicated memo available from CASPER (4).

Another important component of the design is the block that performs the multiplicate and accumulate (MAC) operation, which multiplies directly the antenna signals in pairs, producing the auto-correlation for each antenna and the cross-correlations for each of the baselines, storing them temporarily in the ROACH's RAM, and then is read and transferred to the computer's hard-drive (or some sort of additional external storage unit). Additionally the design includes a couple of blocks that remove the bitgrowth introduced by the PFB-FFT blocks, accessing and controlling some parameters via-software, and detecting and reporting possible signal saturation. The full correlator block-design implemented in Matlab/Simulink using CASPER's toolflow is shown in figure **4-15**.

Once finished, this block diagram is compiled and synthesized into an FPGA bitstream to be implemented physically in the FPGA within the ROACH-1 board. To upload the bitstream file and appropriately program the FPGA, as well as to later being able to communicate via software with it, we use the python KATCP wrapper library. The design starts by itself once the FPGA is programmed, so we use a python script to initialize the correlator according to our design, and another one to store in the computer the data produced by the correlator for each auto- and cross-correlation operation. A description of the python scripts used are shown in Appendix B.

4 CASPER Memo 41 - The Polyphase Filter Bank Technique:
https://casper.astro.berkeley.edu/wiki/images/2/24/Casper_memo_pfb.pdf

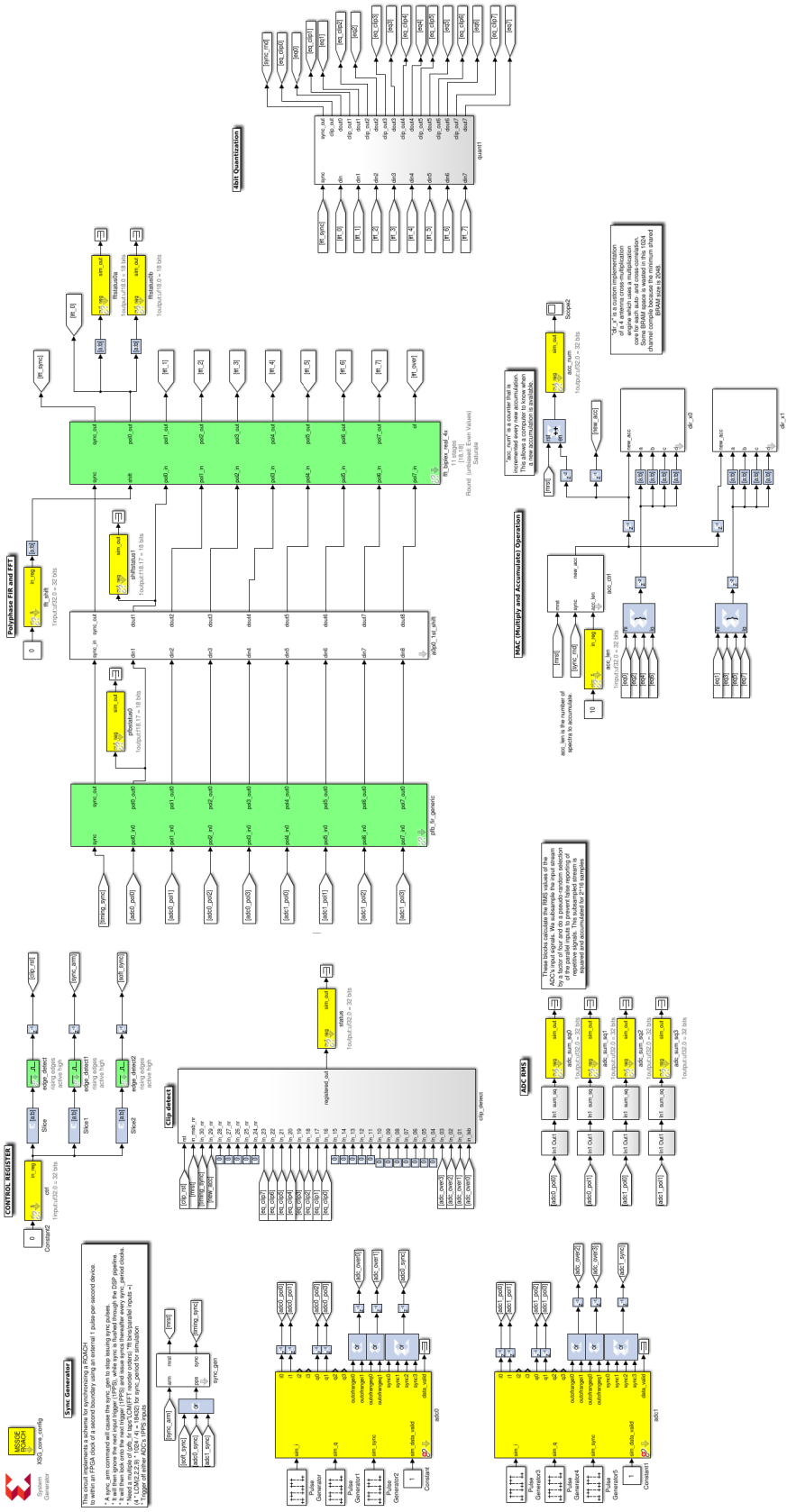

Figure 4-15.: Block schematics for PhAraON's correlator, implemented in MATLAB/Simulink.

Figure 4-16.: Implemented Correlation and Down-conversion stages, along with the control system for the instrument.

Figure 4-17.: Panoramic of the installed interferometer on the roof of Observatorio Astronómico Nacional - Campus building, Universidad Nacional de Colombia.

5. Interferometer Results

5.1. Single Dish Results

For the first validation stage of the prototype, an antenna element was implemented on a fully steerable pier, in order to verify not only the realized gain and received power for the antenna element itself, but also the physical characteristics such as the weight of the feed antenna applied to the parabolic reflector dish, for what would be the final design for the pier support. This first pier design has a manual control for motion on azimuth with a stepper motor and reducer gearbox, and on altitude with a dc actuator.

Tests were performed by pointing directly to the Sun and variating the azimuth for different values of altitude around the source, as a simple scan. In the case of the image seen in figure **5-2**, is the first reported attempt to construct a radio image of the Sun at 1.42 GHz in Colombia with a single dish. Since the motion rate in both azimuth an altitude was not uniform, given that the control is performed manually through a joystick, and the dc actuator does not have an uniform motion during the full range, the results presented in figure **5-2** are labeled with arbitrary units.

These results, as well as the evolving design and implementation stages have been presented as a Poster in three international conferences, "FReSWeD: Towards Future Research in Space Weather Drivers" held in San Juan, Argentina in 2019, "FASR 2021 Workshop" held online, and the "XIII COLAGE - Latin American Conference on Space Geophysics" held in Sao José dos Campos, Brazil, where it was recognized by the Latin American Association of Space Geophysics (ALAGE) with the Roberto Manzano Award, as the best student contribution for the Space Weather session.

Additionally, these validation stage enabled us to see the problems with such pier implementation if we wanted to get an appropriate source tracking system. Since at that point there was not much control in motion to do the source scanning process, mainly because the motion in altitude/elevation exerted by the DC actuator is not uniform throughout the entire motion range, is one of the reasons for the artifacts and additional distortion in the results shown on figure **5-2**. Additionally, at this point the correlator was not yet fully implemented, so the data acquisition was made using a RTLSDR dongle, specifically designed for amateur radio astronomy applications, as our ADC. With the pier configuration described in Chapter 4 of this document, and shown on figures **4-6** to **4-8**, we were able to conduct the tests for the full interferometric array.

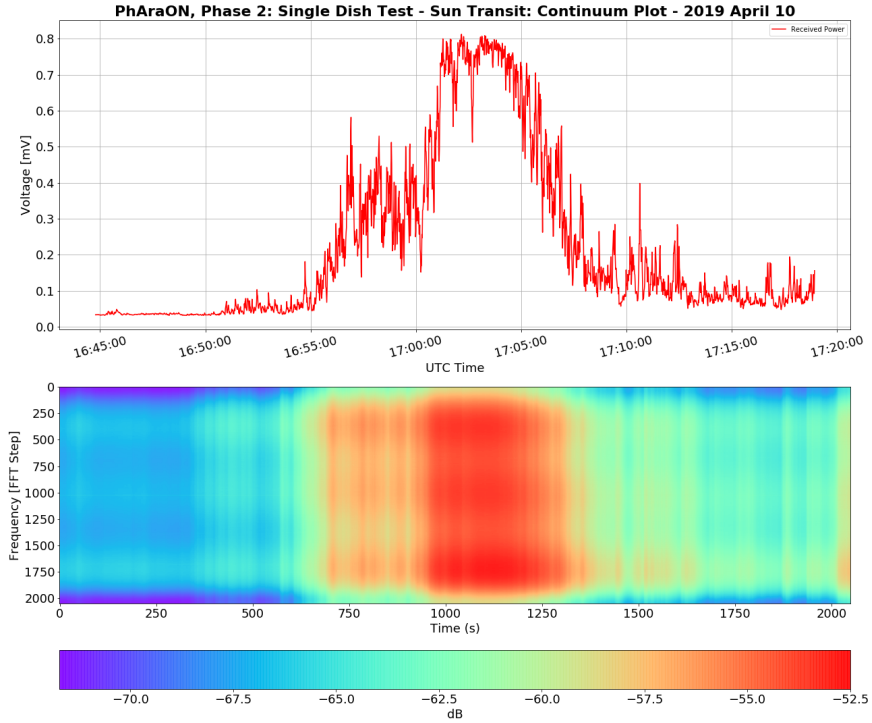

Figure 5-1.: Single dish results, scanning the Sun by variating the azimuth for a fixed altitude.

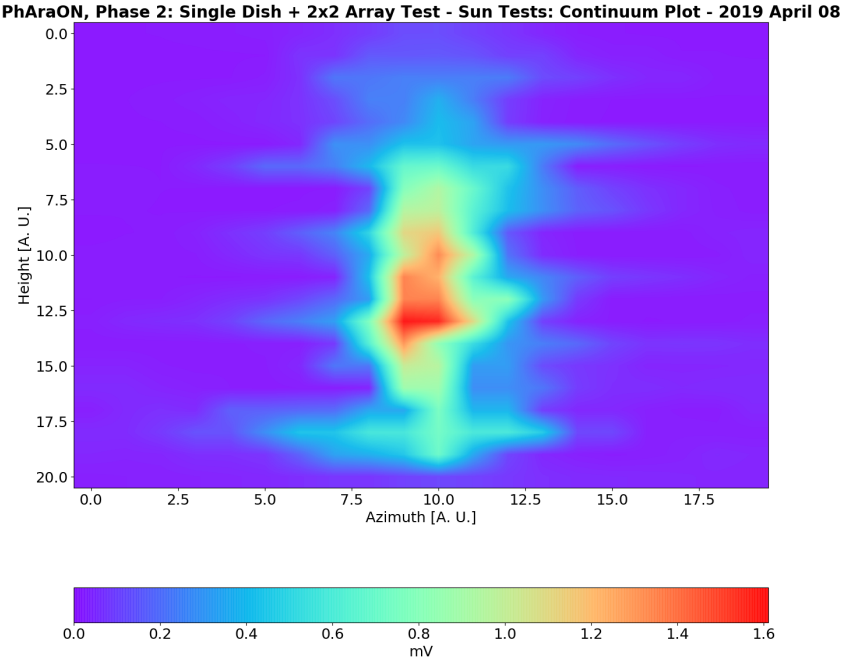

Figure 5-2.: Single dish results, scanning the Sun by variating both azimuth and altitude, to acquire a preliminary image.

5.2. Full Interferometer Results

With the final system design installed in place, the so-called "commissioning stage" was executed, where several tests were performed to verify the appropriate operation of the instrument, and evaluate if any additional corrections should be made. We performed a series of tests with and without any source tracking.

5.2.1. No-source tracking: Source transit

The first series of tests were performed without any source tracking, just pointing the instrument to a given region of the sky and letting the source pass by (transit). Since our source of interest is the Sun, we choose to point at the upper culmination of the Sun and observe for several days at a time, so we could also detect the level of background radiation even when the Sun was not in the field of view of the instrument. The three steerable elements, denoted here as **Antenna A, Antenna B and Antenna C**, were moved to point the region of the sky where we could detect the upper culmination of the Sun. Antenna D, however, since is the element used for calibration and does not have the stepper-motor pier system, is always pointing to a different region of the Sky. Some of those results are presented in figure **5-3** for the auto-correlation of the antennas, and for the cross-correlations tests from figure **5-4** to **5-6** . The results shown cover from sunset of January 26, 2023 until just about after noon of January 27, 2023, local time (GMT -5); Since the data is expressed in UTC time, the data shown appears from 00:00 to 18:00 of January 27, 2023.

In these results, we can see the effects of source passing by during the day, as well as the decrease in received power by each antenna during nighttime. The dark frame at the end of the dataset corresponding to antenna A (figure **5-3**) is due to a fluctuation in the power line that feeds the pre-amplification stage, which resulted in the voltage regulator and LNA being temporarily turned off. Consequently, this effect can also be seen in the upper panels for the cross-correlation plots (figures **5-4** and **5-5**). In antenna D we see no significant increase, since it is pointing off-source. Additionally, the difference in power levels is due to a slight difference in the signal amplification provided by each LNA.

5.2.2. Source tracking

An additional test was made to verify the detection and tracking of the source by the instrument. This test was performed by pointing the instrument to a specific region of the sky, and allowing the Sun to transit over the main beam of the antenna. Then, the instrument main lobe was pointed again at the Sun. This was done in order to verify and confirm the detection of the Sun transiting over the main lobe in all antennas, as shown on figures **5-3** through **5-6**. Later, we pointed the radio interferometer to a different region of the sky, above the horizon but with an

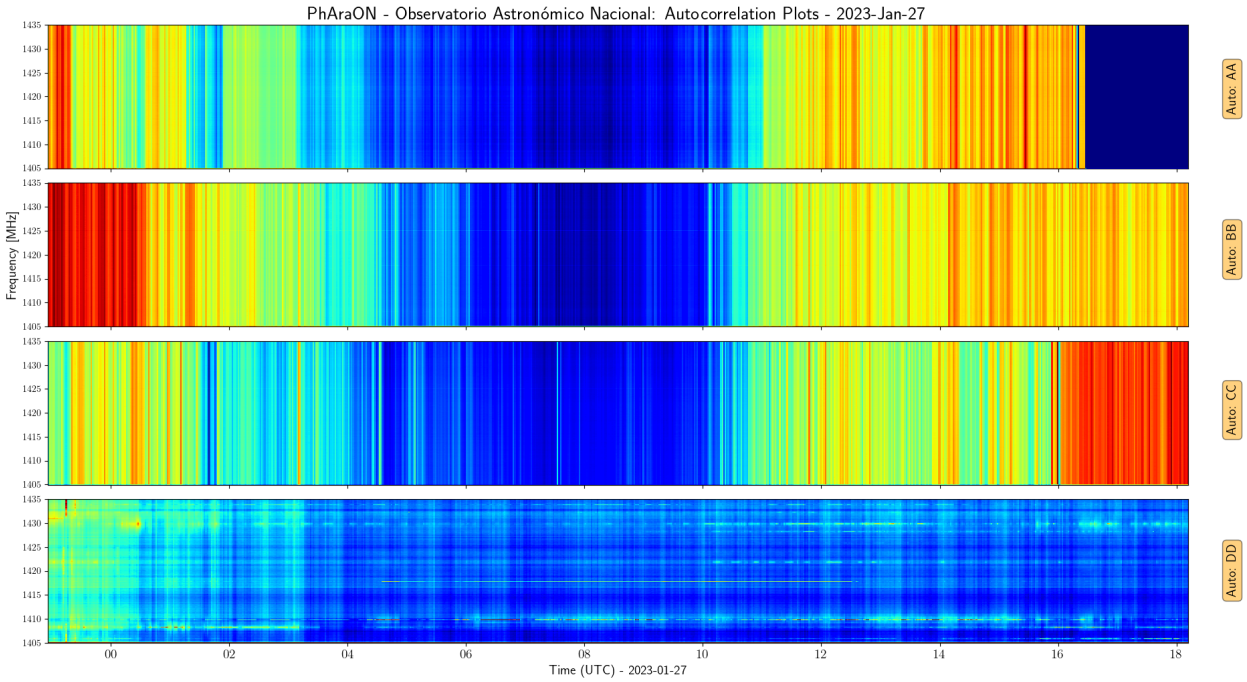

Figure 5-3.: PhAraON results: Auto-correlation tests. January 26-27, 2023. Antenna D is pointing off-source

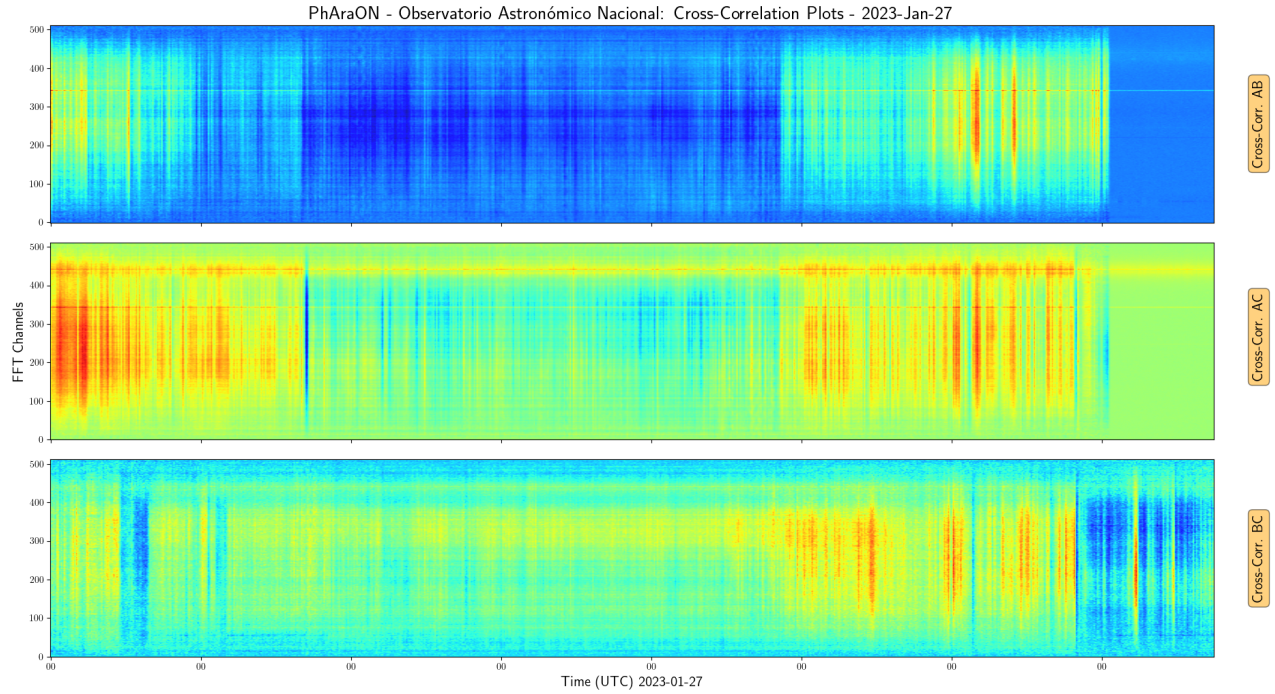

Figure 5-4.: PhAraON results: Cross-correlation tests between the antennas observing the source, i.e., Antennas A, B & C. January 26-27, 2023.

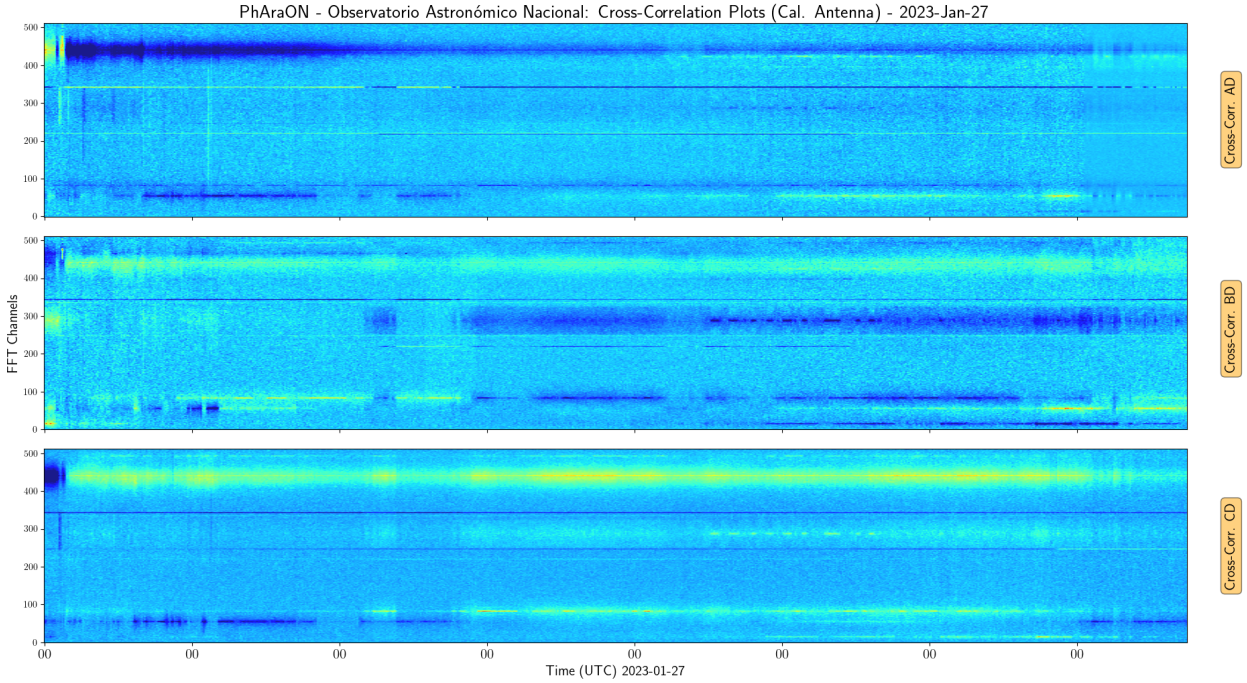

Figure 5-5.: PhAraON results: Cross-correlation tests between each of the steerable antennas (i.e., Antennas A, B & C) with the Calibration antenna (D) pointing off-source. January 26-27, 2023.

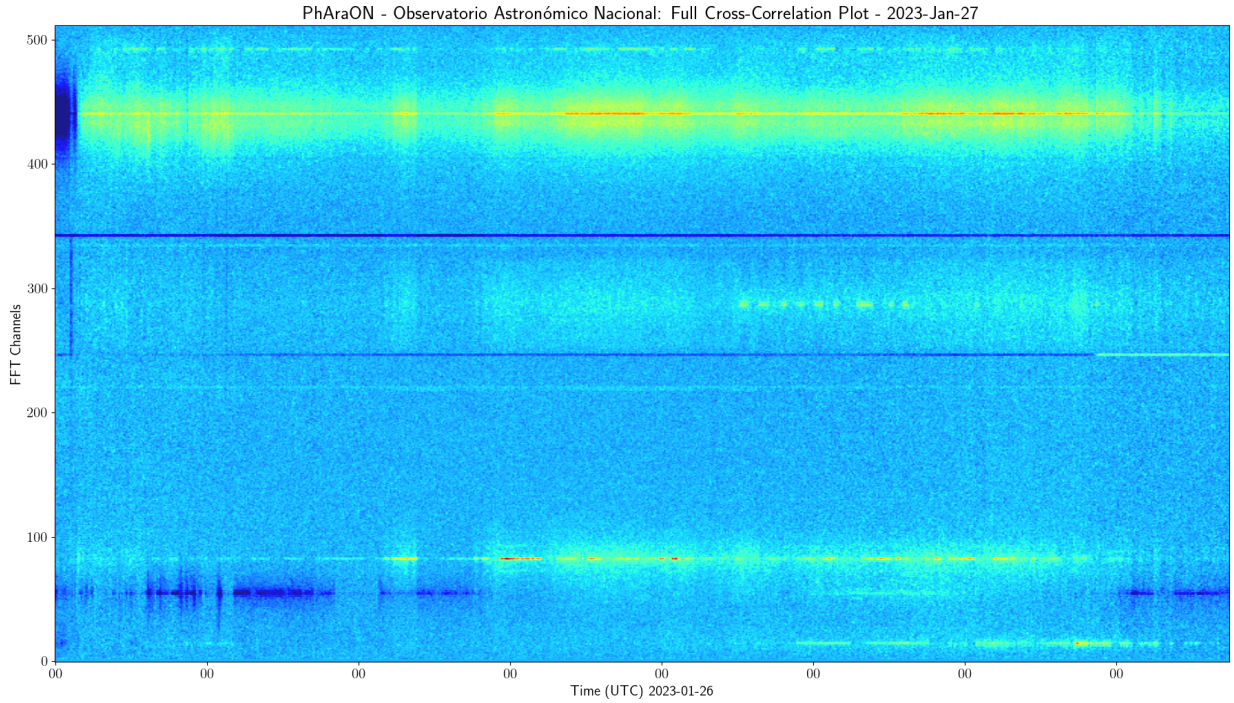

Figure 5-6.: PhAraON results: Full Cross-correlation tests. January 26-27, 2023.

opposite azimuth angle, and therefore clearly observe the contribution from the sky background.

The first solar transit test was made locating the main lobe at the coordinates Az=241°, Alt=42,1°, as reported by Stellarium Astronomy Software [1] [26]. The Sun transited through the center of the main beam of all the antennas at 20:00 UTC (15:00 GMT -5), which means the increase in power would be detected between 19:20 and 20:30 UTC, given and antenna beamwidth of ≈12 degrees, as described in table **4-1** and figure **4-4**. A screenshot of the interferometer pointing through Stellarium is shown on figure **5-7**.

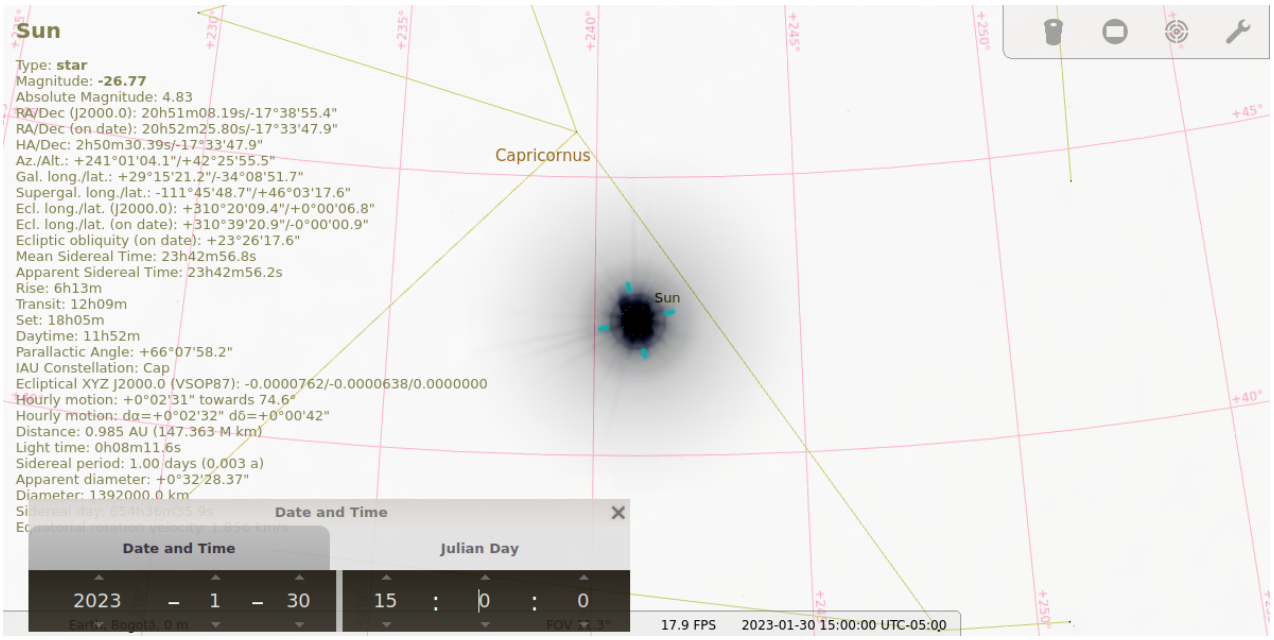

Figure 5-7.: Interferometer orientation control through Stellarium. Source tracking/drifting Stage 1: Azimuth=241°, Altitude=42.1°. January 30, 2023.

The instrument was left static pointing at these coordinates until around 23:07 UTC (18:07 GMT -5, local sunset), when it was steered to point at the coordinates Az=0°, Alt=30°. This off-pointing was performed over night, observing a decrease in the received power with respect to the power emitted by the Sun. The next day, the radio interferometer was pointed again to a region in the sky where the Sun would transit the center of the main beam and drift. At these point a increase of received power was observed again, characteristic of a solar detection. The chosen coordinates for this stage were Az=156°, Alt=65°, therefore, the central detection of the Sun's transit occurred at 16:30 UTC (11:30 GMT -5). A screenshot of the interferometer pointing through Stellarium for this second set of coordinates is shown on figure **5-8**.

[1] Stellarium Astronomy Software: https://stellarium.org/

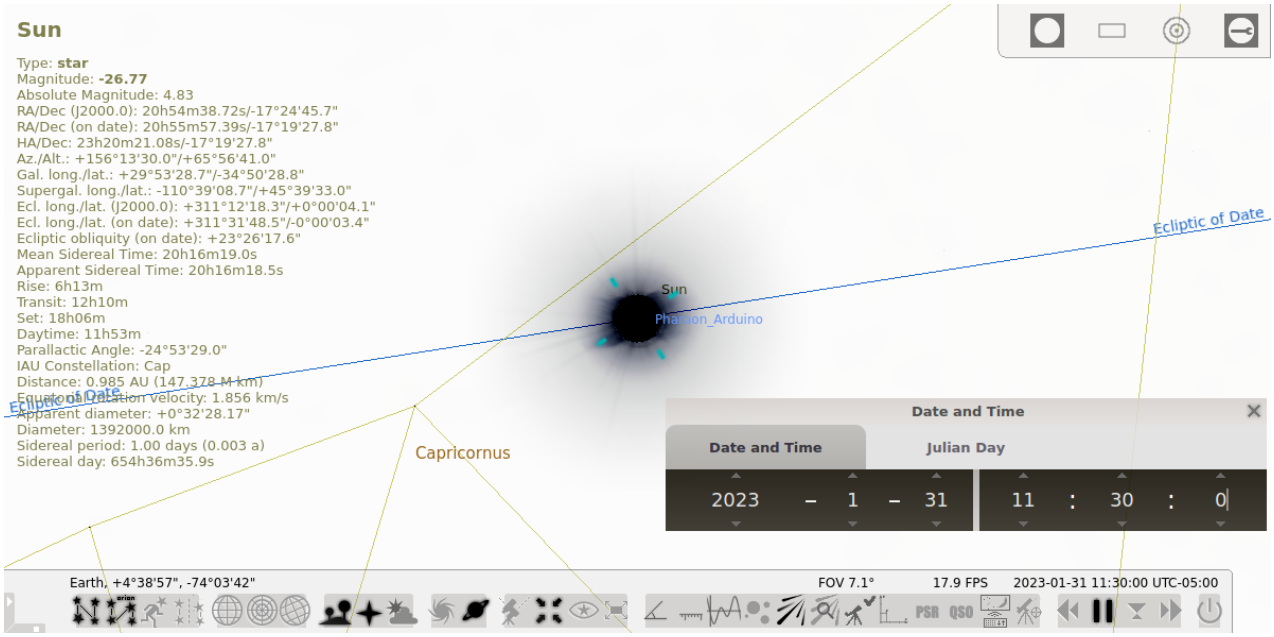

Figure 5-8.: Interferometer orientation control through Stellarium. Source tracking/drifting Stage 3: Azimuth=156°, Altitude=65°. January 31, 2023.

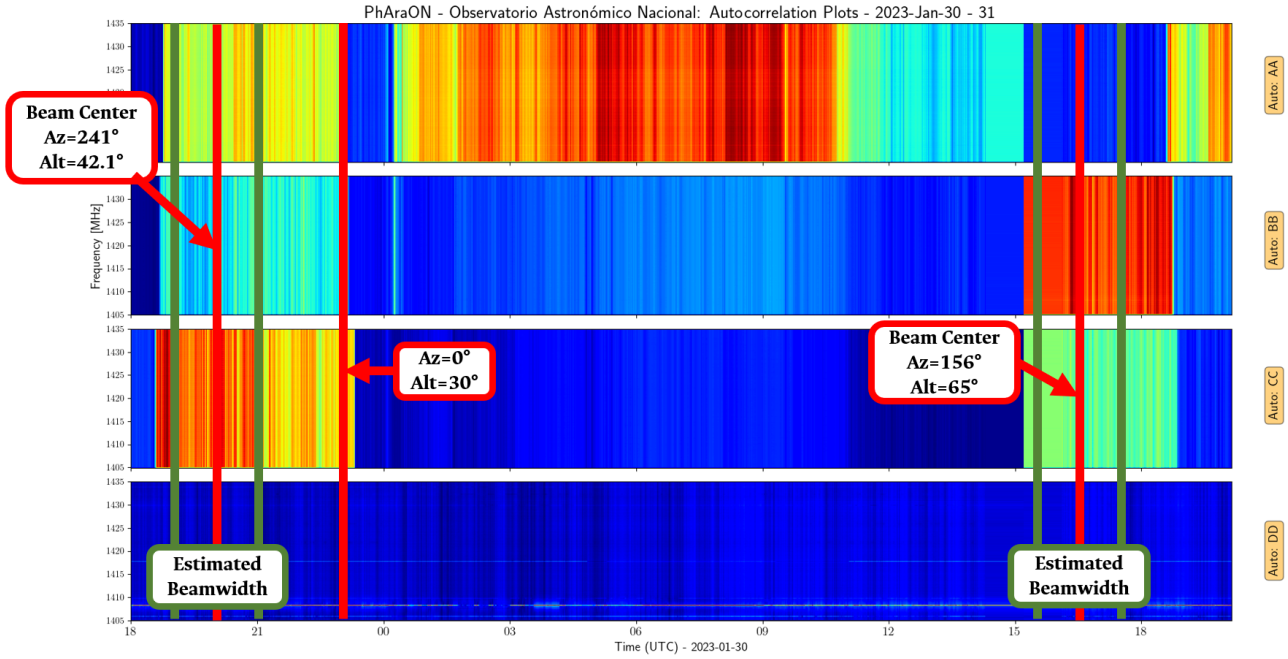

Figure 5-9.: PhAraON results: Auto-correlation plots. January 30-31, 2023. The instrument was steered to track the Sun between 19:30 - 20:30 UTC, January 30th, at Az=241°, Alt=42.1°, and then was parked so the sky transited over the beam of the interferometer.

For this test, we pointed the interferometer to the source to track it, parked it so the sky transited over the beam, and then pointed again to track. In this way we were able to detect the solar transit in the interferometer beam for both previously specified times and coordinates, albeit, the result was not quite as expected. When plotted directly from the raw files generated by the ROACH-1 correlator, we noticed that the values for the received power in each of the steered antennas was "inverted", i.e., for the times of detection of the Sun's pass, the received power seemed to be lower in comparison to the power received through the night. However, the plotted auto-correlation values did evidence a significant change in the received power, simultaneously for all steered antennas, and around the expected time with the specified coordinates.

Subtracting the full arrays from an averaged maximum value allowed us to see the full dynamic range of the change in received power, as well as proportionally "inverting" such power levels; this so we could see more clearly where those changes in the received power occurred during the observation. The obtained results for this 3-stage source tracking and drifting test are shown in figure **5-9**.

This is by no means a proper, science-level result expected from the developed instrument, and must be thoroughly double-checked and fixed, so we can proceed with a proper calibration and use the instrument for actual scientific observations carried at Observatorio Astronómico Nacional, working with our own data. Despite this, is clearly shown in figure **5-10** that, when comparing these results (blue line) with the local elevation angle for the Sun during January 30th and 31st, 2023 (red line), we can state that we have an actual detection of the Sun with our instrument. To highlight this, we have also plotted with vertical lines, the corresponding times for local sunset, sunrise and meridian pass of the Sun during the observation time.

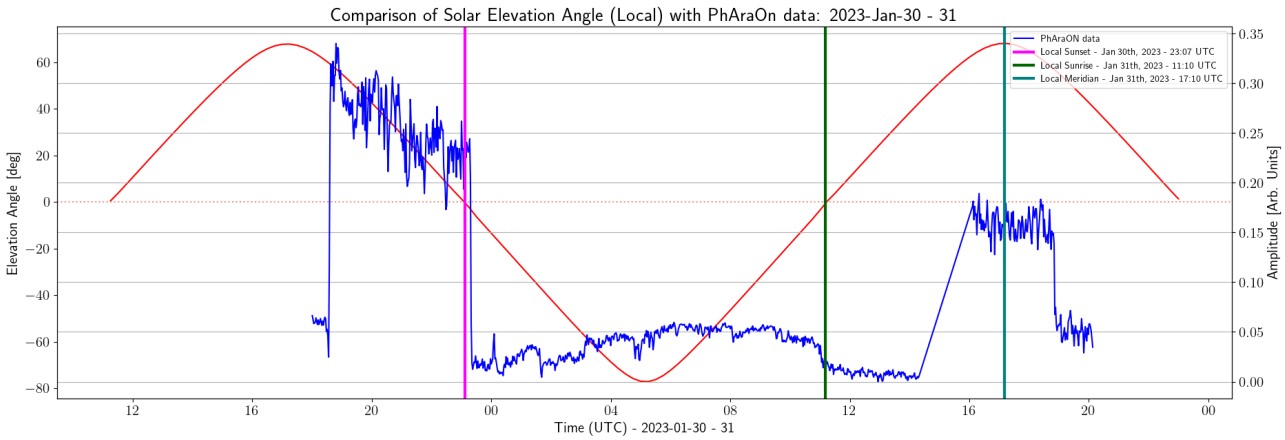

Figure 5-10.: Comparison of local solar elevation angle with obtained PhAraON results. January 30-31, 2023.

Additionally, one possible cause for this kind of discrepancy in the results is related to the initialization of the correlator, particularly with the software gain value introduced when executing the initialization script. As was described in Chapter 4, the synthesized bitstream is uploaded for the implementation on the FPGA within the ROACH-1 board through a python initialization script, where one also can establish a gain coefficient to scale the signal received from each antenna. This signal scaling process occurrs within the 4-bit quantizer block in the correlator diagram (figure **4-15**). A detail of this 4-bit quantizer block is shown in figure **5-11**.

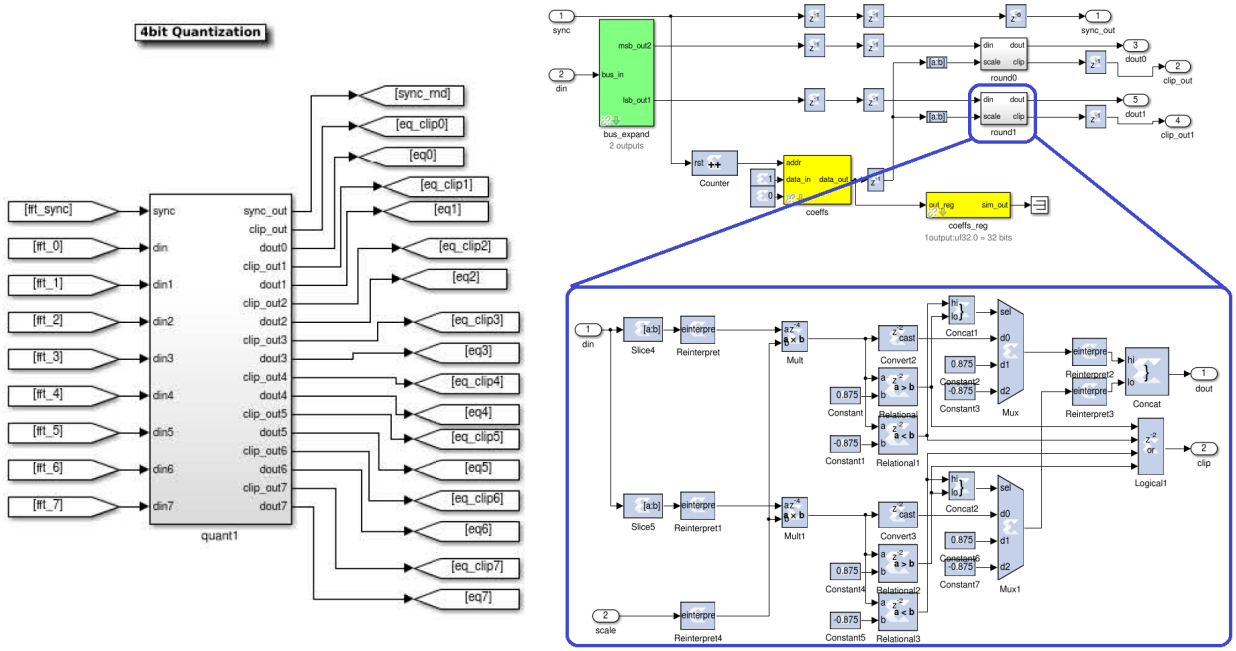

Figure 5-11.: 4-bit quantizer block implemented in MATLAB/Simulink for the correlator. ***Left:*** Top Block. ***Right:*** Block detail for each antenna branch and the corresponding "*round*" block in charge of the scaling process.

In CASPER's tutorials for the "Wideband Spectrometer" [2] and "Wideband Pocket Correlator", which are the basis for our correlator's design, describes the 4-bit quantizer block as a subsystem in charge of removing the bitgrowth introduced by the PFB and FFT blocks, as well as reporting signal saturation to the "Clip-detect" block. Additionally, the previously mentioned gain control is added through the software register labeled as "coeffs" in figure **5-11**. The highlighted "round" block is the one in charge of carrying this operation within the quantizer subsystem.

Initial tests for the implementation of the correlator revealed that typical gain values (between 1 and 1000) introduced by the python initialization script through the software register, were insufficient for this scaling process, which ended in output files with null-data for each sampling.

[2]ROACH Tutorial 3: Wideband Spectrometer:
https://casper-toolflow.readthedocs.io/projects/tutorials/en/latest/tutorials/roach/tut_spec.html

This means that the received values at the input of the *round* block were somehow read as too low, and the block effectively writes them to zero. Subsequent implementation tests resulted in a range of gain values that would output legible data collected by the instrument; such values were found to be above a gain coefficient of 25000. For the case of the observations whose results are shown in figure **5-9**, the input gain value used is 150000. This could be effectively read later on as a value high enough that eventually saturates the signal for the subsequent sampling cycle. This could result not only in a saturation of the signal, but perhaps in a gain roll-off effect, and therefore in a erroneous setting of the values to be written into the Multiplicate and Accumulate (MAC) operation of the correlator, hence to the output data files.

6. Conclusions and Discussion

As a logical continuation of the instrumentation project developed for a 2-element drift solar radio interferometer laid out on a E-W baseline [11], the most important result is the successful design, development, implementation and testing of a multi-element radio interferometer, to conduct astronomical observations at 1.42 GHz. This also stands out as an adaptation of an implemented prototype for an antenna feed, whose manufacturing is relatively easy and low-cost. Furthermore, we can assert that the scientific and technological goals proposed for this Master's Thesis project were successfully achieved.

The implementation of the correlator using a ROACH-1 board makes the system versatile and relatively easy to modify and update, according to the scientific and technical needs of a future observation campaign, through the use of the toolflow developed by CASPER collaboration, and also following the tutorials they have kindly made available. We should note that the current implementation of the correlator is of slightly more careful consideration, given that the implementation of the downconversion stage depended on the technical specifications and availability of commercial downconverters specifically tuned for our band of interest. Even if the main idea of a "Pocket Correlator" based on those aforementioned tutorials remained, such a drastic downconversion stage (from 1.42 GHz o 30 MHz) implied heavy modifications and full redesign of several blocks, to be able to compile and satisfy the technical needs for our current research case.

Albeit the presented results are far from what we consider to be science-quality data to conduct professional observations for research in astrophysics, such results verify the operation of the radio interferometer by detecting the pass of an extended source over its field of view. With the revision and correction of the drawbacks presented during the commissioning stages, not only we will have better observations of both the Sun and other celestial radio sources with emission at 1.42 GHz, but is highly possible that the instrument can be properly calibrated in the short term to obtain good quality data in physical units of spectral flux density, whether it is Solar Flux Units (SFU), W/m^2, or Janskys (Jy).

Additionally, even though the band between 1400 and 1427 MHz is specifically reserved for radio astronomy applications, according to *Agencia Nacional del Espectro (ANE)* [1], entity in charge of

[1] Agencia Nacional del Espectro.
https://portalespectro.ane.gov.co/Style %20Library/ane_master/cnabf-tecnico.aspx

supervising and controlling the radio spectrum in Colombia, we can see clearly in figure **5-3** that there is a semi-continuous emission around 1408 MHz, perhaps related to some radiolocation applications, which introduces some undesired RFI to our observations.

Finally, even though at first the instrument was envisioned to operate as a fully functional phased array, as the title of the project suggests, different time, logistics and cost constraints didn't make it possible to reach such point. However, the current stage of operation of the instrument is a enormous advance, since is very close to a fully implemented passive phased array. The final step in the implementation of a full active phased array is just evaluating how the appropriate transmitter and phase shifters would eventually be integrated. The current Yagi-Uda antenna used, besides being a novel prototype, can be effectively used for such phase-shifting endeavours.

A phased array, as shown on figure **6-1**, is a series of antennas whose major radiation can be (electronically) oriented in any direction, by controlling the phase excitation between the elements, to form a scanning array [2]. Such systems can be used to form multiple beams. The phase and delay of the signal from each antenna element can be adjusted so that the signals from a given direction in the sky combine in phase, maximizing the sensitivity (phase shifting) [24]. In a correlator array, as is the instrument described in this document, the effect of an antenna phase shift can be simulated by changing the measured phases when combining the correlator outputs, i.e., combining measured cross-correlations with an appropriate phase variation to achieve beam-scanning [24].

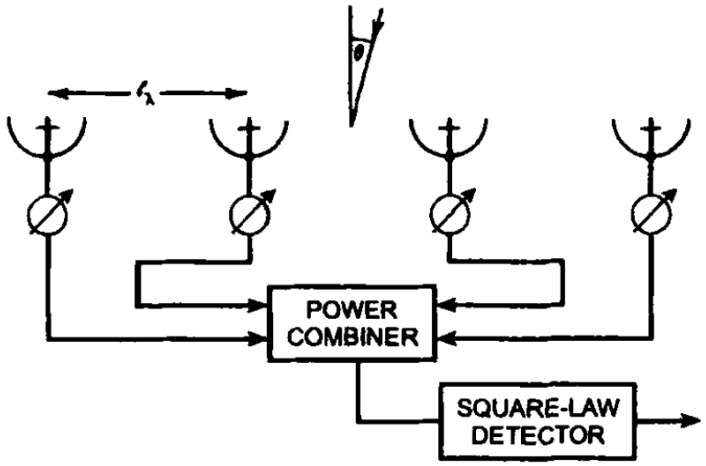

Figure 6-1.: Simple four-element linear interferometer connected as a phased array, with a phase shifter in the output of each antenna. Taken from [24].

A. Appendix A: Arduino codes for the control and operation of the Interferometer

The following are the corresponding codes for the control and operation of the radio interferometer to conduct astronomical observations. First,we present the code for the Master Arduino, which communicates directly with Stellarium, and sends the instructions to the replicas.

```
#include "RS485_protocol.h"
#include <SoftwareSerial.h>
#include <stdlib.h>
#include <math.h>

const byte ENABLE_PIN = 8;
SoftwareSerial rs485 (9, 10); // receive pin, transmit pin
const int baudRate = 9600;

const int buttonAlign = 2;     // Arduino Pin enables Star Alignment (Arduino UNO
   )
const int Vel = 1;
const int Az_resol = 216;     // Minimum resolution for Azimuth motion (in
   arcseconds) at full_step = 216 arcsec
const int h_resol = 108;     // Minimum resolution for Height motion (in
   arcseconds) at full_step = 108 arcsec


int buttonState = 0;           // variable in which arduino will read the logical
    state of buttonAlign
char input[20];                // char array - Stores input data
char txAR[10];                 // char array - Stores output data
char txDEC[11];                // char array - Stores output data
long ARtel, DECtel;            // Telescope coordinates, expressed in seconds
long ARtarget, DECtarget;      // Target coordinates, expressed in seconds
long A_diff = 0, h_diff = 0;         // Difference between target coordinates
   and telescope coordinates
```

```
long arHH, arMM, arSS;         // Right Ascension (AR) Coordinate of the
   telescope to be transmitted to Stellarium
char SIGNtel;                  // Declination (DEC) sign: 43 = + 45 = -
long decDEG, decMM, decSS;     // Declination (DEC) Coordinate of the telescope
   to be transmitted to Stellarium
boolean enablegoto_master;
long latitude, TSL, A_teltx, A_tel, h_tel, H_tel, H_target, h_target, A_target;
const float pigreco = 3.141593;
float phi, delta_tel, H_telRAD, h_telRAD, A_telRAD, cos_phi, sin_phi;
float delta_target, H_targetRAD, h_targetRAD, A_targetRAD;
float numerator1, denominator1, numerator2, denominator2;

/*-----------------------------------------------------------*/
// data to be sent and received
//==============================
const int Az_Right = 2;
const int Az_Left = 3;
const int Alt_Up = 4;
const int Alt_Down = 5;
int pin_move_right = 0;
int pin_move_left = 0;
int pin_move_up = 0;
int pin_move_down = 0;
//==============================
int valAz = 0;
int valAlt = 0;
byte moveAz = 0;
byte moveAlt = 0;
byte brake = 1;
int brakeval = 1;

const int brakeswitch_az = 11;
int brakevalue_az = 0;

const int manual_switch = 12;
int manual_enable = 0;

const int receive_button = 7;
int receive_enable = 0;

unsigned long prevUpdateTime = 0;
unsigned long updateInterval = 33;
```

```
bool newTxData = false;
bool newTxData_manual = false;
bool newRxData = false;

int delayrw = 0;

// Callback Routines - RS485
void fWrite (const byte what) // Writes data to RS485 bus
  {
  rs485.write (what);
  }

int fAvailable () // Checks if RS485 bus is available
  {
  return rs485.available ();
  }

int fRead () // Reads data from RS485 bus
  {
  return rs485.read ();
  }

/*------------------------------*/
void setup() {

  pinMode(buttonAlign, INPUT);
  rs485.begin (baudRate);
  Serial.begin(baudRate);
  Serial.println("¡RS485 Half Duplex TX listo!");
  pinMode (ENABLE_PIN, OUTPUT); // driver output enable
//  pinMode (LED_PIN, OUTPUT); // built-in LED
  pinMode(brakeswitch_az, INPUT);
  pinMode(Az_Left, INPUT_PULLUP);
  pinMode(Az_Right, INPUT_PULLUP);
  pinMode(Alt_Up, INPUT_PULLUP);
  pinMode(Alt_Down, INPUT_PULLUP);
  pinMode(manual_switch, INPUT_PULLUP);
  pinMode(receive_button, INPUT_PULLUP);

  A_target = A_tel = 0;       // Azimuth (A) Initial coordinate (in arcseconds)
     to display the telescope reticle
```

```
  h_target = h_tel = 16702;       // Height (h) Initial coordinate (in arcseconds)
       to display the telescope reticle
  latitude = 16702;               // Latitude of Observation Site expressed in
      arcseconds (at OAN Campus Building) = 16702

  phi = (float(latitude)/3600.0)*pigreco/180.0; // Latitude in Radians
  cos_phi = cos(phi);
  sin_phi = sin(phi);
  enablegoto_master = false;
  TSL = 0;                          // (in this Sketch there is no sidereal clock,
      therefore TSL is fixed at zero and must be expressed in seconds)

}
byte old_brake = 1;
byte old_level_az = 0;
byte old_level_alt = 0;

void loop() {
  brakevalue_az = digitalRead(brakeswitch_az);
  pin_move_right = digitalRead(Az_Right);
  pin_move_left = digitalRead(Az_Left);
  pin_move_up = digitalRead(Alt_Up);
  pin_move_down = digitalRead(Alt_Down);
  manual_enable = digitalRead(manual_switch);
  receive_enable = digitalRead(receive_button);

  if(Serial.available()>0){
      communication();
  }


  A_diff = A_target - A_tel;
  h_diff = h_target - h_tel;

  old_level_az = 0;
  old_level_alt = 0;

  if (receive_enable == LOW){
    receiveData();
  }

  if (manual_enable == LOW) {
```

```
    manual_control();
  }
  else{
    automatic_control();
    }
}
//---------------------------------
void receiveData() {
  byte msg[] = {
    1,
    moveAz,
    moveAlt,
    brake
    };
  byte received = recvMsg (fAvailable, fRead, msg, sizeof (msg));
  if (received){
    brake = msg[3];
    return brake;
    }
    old_brake = brake;
    if (old_brake == 0){
      enablegoto_master = false;
      }
}
//---------------------------------
void automatic_control(){
    if((brakevalue_az == HIGH) && (brake == 0)){
      enablegoto_master = false;
      brake = 0;
      valAz = 0;
      valAlt = 0;
      updateDataToSend_auto();
      transmitData();
      }
    if (old_brake == 1) {
      if(enablegoto_master == true){
          updateDataToSend_auto();
          transmitData();
          valAz = 0;
          valAlt = 0;
          old_level_az = 0;
          old_level_alt = 0;
```

```
      }
    }
  }
//---------------------------------
void manual_control(){

    if((brakevalue_az == HIGH) && (brake == 0)){
      enablegoto_master = false;
      brake = 0;
      valAz = 0;
      valAlt = 0;
      updateDataToSend_manual();
      transmitData();
    }
   if (old_brake == 1) {
    if(enablegoto_master == true){
       updateDataToSend_manual();
       transmitData();
       valAz = 0;
       valAlt = 0;
       old_level_az = 0;
       old_level_alt = 0;
      }
    }
  }
//---------------------------------
void updateDataToSend_manual() {
   if (millis() - prevUpdateTime >= updateInterval) {
      prevUpdateTime = millis();
      if (enablegoto_master == true){
        if (newTxData_manual == false) { // ensure previous message has been sent
          goto_object_manual();
          }
        }
   }
}
//---------------------------------
void updateDataToSend_auto() {
   if (millis() - prevUpdateTime >= updateInterval) {
      prevUpdateTime = millis();
      if (enablegoto_master == true){
        if (newTxData == false) { // ensure previous message has been sent
```

```
          goto_object_master();
          }
       }
   }
}
//---------------------------------
void transmitData() {
moveAz = valAz;
moveAlt = valAlt;

   // assemble message
   byte msg [] = {
      1,   // device 1
      moveAz,
      moveAlt,
      brake
   };

   if (newTxData == true) {
       digitalWrite (ENABLE_PIN, HIGH); // enable sending
       sendMsg (fWrite, msg, sizeof msg);
       digitalWrite (ENABLE_PIN, LOW); // disable sending

       newTxData = false;
       delayMicroseconds(delayrw);
       old_level_az = moveAz;
       old_level_alt = moveAlt;
   }

   if (newTxData_manual == true) {
       digitalWrite (ENABLE_PIN, HIGH); // enable sending
       sendMsg (fWrite, msg, sizeof msg);
       digitalWrite (ENABLE_PIN, LOW); // disable sending

       newTxData_manual = false;
       delayMicroseconds(delayrw);
       old_level_az = moveAz;
       old_level_alt = moveAlt;
   }
}
//---------------------------------
void communication(){
```

```
int i=0;
input[i++] = Serial.read();
delay(5);
while((input[i++] = Serial.read()) != '#'){
  delay(5);
}
input[i]='\0';

if(input[1]==':' && input[2]=='G' && input[3]=='R' && input[4]=='#'){ // With
    the #:GR# command, Stellarium asks to send the Right Ascension (AR)
    coordinate
  transmitAR();
}

if(input[1]==':' && input[2]=='G' && input[3]=='D' && input[4]=='#'){ // With
    the #:GD# command, Stellarium asks to send the Declination (DEC) coordinate
  transmitDEC();
}

if(input[1]==':' && input[2]=='Q' && input[3]=='#'){                  // With
    the #:Q# command, Stellarium asks to stop the motors
// Send Stop to motors.
}

if(input[0]==':' && input[1]=='S' && input[2]=='r'){                  // With
    the :Sr command Stellarium sends the Right Ascension (AR) Coordinate
  getAR();
}

if(input[0]==':' && input[1]=='S' && input[2]=='d'){                  // With
    the :Sd command Stellarium sends the Declination (DEC) Coordinate
  getDEC();
}

if(input[0]==':' && input[1]=='M' && input[2]=='S' && input[3]=='#'){ // With
    the :MS# Command, Stellarium asks if Rotation is possible
  Serial.print("0");
  buttonState = digitalRead(buttonAlign);                             // Reads
     whether the Switch for Initial Alignment is ON or OFF
  if (buttonState == HIGH) {                                          // If its
      HIGH updates the coordinates, otherwise continues with GOTO
```

```
     ARtel = ARtarget;
     DECtel = DECtarget;
   }
   enablegoto_master = true;                                          //
      Enables GOTO
  }
}
//-----------------------------
void transmitAR(){                                                    // Transmits
   Right Ascension data in the format HH:MM:SS#
    convert_AZ_EQ();
    arHH = ARtel/3600;                                                // Obtains
       the HOUR in the Telescope's AR Coordinate
    arMM = (ARtel-arHH*3600)/60;                                      // Obtains
       the MINUTE in the Telescope's AR Coordinate
    arSS = (ARtel-arHH*3600)-arMM*60;                                 // Obtains
       the SECOND in the Telescope's AR Coordinate
    sprintf(txAR, "%02d:%02d:%02d#", int(arHH), int(arMM), int(arSS));
    Serial.print(txAR);
   }
//-----------------------------
void transmitDEC(){                                                   // Transmits
   Declination data in the Format sDDßMM:SS# (where s is the + or - sign)
    convert_AZ_EQ();
    (DECtel < 0) ? SIGNtel = 45: SIGNtel = 43;                        // Checks the
        Sign in the Telescope's DEC Coordinate
    decDEG = abs(DECtel)/3600;                                        // Obtains
       the DEGREES in the Telescope's DEC Coordinate
    decMM = (abs(DECtel) - decDEG*3600)/60;                           // Obtains
       the MINUTES in the Telescope's DEC Coordinate
    decSS = (abs(DECtel) - decDEG*3600) - decMM*60;                   // Obtains
       the SECONDS in the Telescope's DEC Coordinate
    sprintf(txDEC, "%c%02d%c%02d:%02d#", SIGNtel, int(decDEG), 223, int(decMM),
        int(decSS));
    Serial.print(txDEC);
   }
//-----------------------------
void getAR(){                                                         // Receives
   the Target's AR Coordinate in the format :Sr HH:MM:SS#
    Serial.print("1");
    ARtarget = (atol(input+3))*3600 + (atol(input+6))*60 + atol(input+9); //
       Converts to seconds the Target's AR Coordinate
```

```
    enablegoto_master = false;                                                    //
       Keeps GOTO disabled until the arrival of DEC Coordinate
   }
//------------------------------
void getDEC(){                                                                  // Receives
   the Target's DEC Coordinate in the format :Sd +DDßMM:SS#
   Serial.print("1");
   DECtarget = (atol(input+4))*3600 + (atol(input+7))*60 + atol(input+10); //
      Converts to seconds the Target's DEC Coordinate
   if (input[3] == '-'){                                                        // For a
      correct conversion, we keep only the numeric part
    DECtarget *=(-1);                                                           // (i.e.
       without the sign) Therefore, DEC coordinate is negative
    }                                                                           // After
       conversion to seconds, it is multiplied by -1
    enablegoto_master = false;
    convert_EQ_AZ();
  }
//-------------------------------------------------------------------------------------

void convert_EQ_AZ(){
  H_target = TSL - ARtarget;
  if (H_target < 0){
    H_target = H_target + 86400;
                                                                  // Hour Angle is always
       positive
  }
  H_targetRAD = (float(H_target)*15.0/3600.0)*pigreco/180.0;
                            // Target's Hour Angle expressed in Radians
  delta_target = (float(DECtarget)/3600.0)*pigreco/180.0;
                               // Target's Declination expressed in Radians

  h_targetRAD = asin(sin_phi*sin(delta_target) + cos_phi*cos(delta_target)*cos(
     H_targetRAD));
  h_targetRAD = (h_targetRAD*180.0/pigreco)*3600.0;
                                    // Target's Height (h) expressed in
     arcseconds
  h_target = long(h_targetRAD);

  numerator1 = sin(H_targetRAD);
  denominator1 = (cos(H_targetRAD)*sin_phi - tan(delta_target)*cos_phi);
  A_targetRAD = atan(numerator1/denominator1);
```

```
  A_targetRAD = (A_targetRAD*3600.0)*180.0/pigreco;
                                      // A_targetRAD expressed in arcseconds
  A_target = long(A_targetRAD);

  if ((numerator1 >= 0) && (denominator1 >=0)){
                                         // Adding the Offset. Also, includes the
     correction to have the Azimuth's origin at North
    A_target = A_target + 648000;
                                                              // by adding another 180°
        (expressed in arcseconds)
  }

     //
  if(denominator1 < 0){
    A_target = A_target + 1296000;
  }
  if ((numerator1 < 0) && (denominator1 >=0)){
    A_target = A_target + 1944000;
  }

  if (A_target >= 1296000){
                                                                 // If Azimuth
     exceedes 360°, subtract 360°
    A_target = A_target-1296000;
  }
}
//---------------------------
void convert_AZ_EQ(){
  A_teltx = A_tel;
  A_teltx = A_teltx - 648000;
                                                             // Restoring Azimuth's
     Origin to the South by subtracting 180°
  if(A_teltx < 0){
                                                                        // and if it
     's negative, add 360°
    A_teltx = A_teltx + 1296000;
  }

  A_telRAD = (float(A_teltx)/3600.0)*pigreco/180.0;
                                      // Telescope's Azimuth (A) expressed in
     radians
```

```
  h_telRAD = (float(h_tel)/3600.0)*pigreco/180.0;
                                        // Telescope's Height (h) expressed in
     radians

  delta_tel = asin(sin_phi*sin(h_telRAD) - cos_phi*cos(h_telRAD)*cos(A_telRAD));
       // Telescope's Declination (DEC) expressed in radians
  delta_tel = (delta_tel*180.0/pigreco)*3600.0;
                                          // Telescope's Declination (DEC)
     expressed in arcseconds
  DECtel = long(delta_tel);

  numerator2 = sin(A_telRAD);
  denominator2 = (cos(A_telRAD)*sin_phi + tan(h_telRAD)*cos_phi);
  H_telRAD = atan(numerator2/denominator2);
  H_telRAD = (H_telRAD*180.0/(pigreco))*(3600.0/15.0);
                                    // H_telRAD expressed in seconds
  H_tel = long(H_telRAD);

 if(denominator2 < 0){
                                                                  // Adding the
    Offset
    H_tel = H_tel + 43200;
  }
  if ((numerator2 < 0) && (denominator2 >=0)){
    H_tel = H_tel + 86400;
  }

  if (H_tel >= 86400){
    H_tel = H_tel - 86400;
  }

  ARtel = TSL - H_tel;
  if (ARtel < 0){
                                                                      // Right
     Ascension is always positive
    ARtel = ARtel + 86400;
  }
}
//---------------------------
void goto_object_master(){
  if ((A_diff >0 && A_diff <= 648000) || (A_diff <=(-648000))) { //Compares the
     AR coordinates and chooses the shortest way to reach target
```

```
    increment_A_tel(); /* Move Right */
  }

  if ((A_diff > 648000) || (A_diff <0 && A_diff > (-648000))) {
    decrement_A_tel(); /* Move Left */
  }

  if (h_target > h_tel){
    go_up();
  }
  if (h_target < h_tel){
    go_down();
  }
  if (h_target == h_tel){
    stop_movement();
  }
}

//------------------------
void goto_object_manual(){
  if ( ((pin_move_right == LOW) && (pin_move_left == HIGH)) ){ //Compares the AR
      coordinates and chooses the shortest way to reach target
    increment_A_tel(); /* Move Right */
  }

  if ( ((pin_move_right == HIGH) && (pin_move_left == LOW))){
    decrement_A_tel(); /* Move Left */
  }

  if ( ((pin_move_up == LOW) && (pin_move_down == HIGH)) ){
    go_up();
  }

  if ( ((pin_move_up == HIGH) && (pin_move_down == LOW)) ){
    go_down();
  }

  if ( ((pin_move_left == HIGH) && (pin_move_right == HIGH)) || ((pin_move_up ==
      HIGH) && (pin_move_down == HIGH)) || (((pin_move_left == LOW) && (
      pin_move_right == LOW)) || ((pin_move_up == LOW) && (pin_move_down == LOW)))
       )
  {
```

```
    stop_movement();
  }

}

//-------------------------
void increment_A_tel(){
  int Az_count = 0;
  while(Az_count < Az_resol){
    Az_count++;
    A_tel++;
    valAz = 0; /* "Stop/Don't Move Az */
    newTxData = true;
    newTxData_manual = true;
    }
  if(Az_count = Az_resol){
    valAz = 2; /* 2 = "Move Az_Right */
    newTxData = true;
    newTxData_manual = true;

    Az_count == 0;
    A_tel++;
    }
  if(A_tel >= 1296000){
    A_tel = A_tel - 1296000;
    }

  if(A_tel >= 504000 && A_tel <= 792000){
    stop_movement();
    enablegoto_master = false;
    }
}

//---------------------------
void decrement_A_tel(){
  int Az_count = 0;
  while(Az_count < Az_resol){
    Az_count++;
    A_tel--;
    valAz = 0; /* "Stop/Don't Move Az */
    newTxData = true;
    newTxData_manual = true;
```

```
    }
  if(Az_count = Az_resol){
    valAz = 3; /* 3 = "Move Az_Left */
    newTxData = true;
    newTxData_manual = true;

    Az_count == 0;
    A_tel--;
    }
  if(A_tel < 0){
    A_tel = A_tel + 1296000;
  }
  if(A_tel >= 504000 && A_tel <= 792000){
    stop_movement();
    enablegoto_master = false;
    }
}
//--------------------------
void go_up(){
  int Alt_count = 0;
  while(Alt_count < h_resol){
    Alt_count++;
    h_tel++;
    valAlt = 0; /* "Stop/Don't Move Alt */
    newTxData = true;
    newTxData_manual = true;
    }
  if(Alt_count = h_resol){
    valAlt = 4; /* 4 = "Move Alt_Up */
    newTxData = true;
    newTxData_manual = true;

    Alt_count == 0;
    h_tel++;
    }
}
//-----------------------------
void go_down(){
  int Alt_count = 0;
  while(Alt_count < h_resol){
    Alt_count++;
    h_tel--;
```

```
    valAlt = 0; /* "Stop/Don't Move Alt */
    newTxData = true;
    newTxData_manual = true;
    }
  if(Alt_count = h_resol){
    valAlt = 5; /* 5 = "Move Alt_Down */
    newTxData = true;
    newTxData_manual = true;

    Alt_count == 0;
    h_tel--;
    }
}
//==============================
void stop_movement(){
  brake = 0;
  brakeval = 0;
  valAz = 0;
  valAlt = 0;
  newTxData = true;
  newTxData_manual = true;
}
//==============================
```

And the following is the code for all of the Replica Arduinos, which receives the instructions from the Master Arduino and send said instructions to the stepper motors for the motion of each antenna element.

```
#include <SoftwareSerial.h>
#include "RS485_protocol.h"

SoftwareSerial rs485 (9, 10); // receive pin, transmit pin
const byte ENABLE_PIN = 8;
const int baudRate = 9600;

byte msg [4];
int valAz = 0;
int valAlt = 0;

byte moveAz = 0;
byte moveAlt = 0;
```

```
byte brake=1;
bool newRxData = false;

unsigned long prevUpdateTime = 0;
unsigned long updateInterval = 33;
bool newTxData = false;

int delayrw = 0;
int EN_Az = 2;
int DIR_Az = 4;
int PUL_Az = 3;
int EN_Alt = 5;
int DIR_Alt = 7;
int PUL_Alt = 6;

const int brakeswitch_az = 11;
int brakevalue_az = 0;
const int brakeswitch_el = 12;
int brakevalue_el = 0;
int brakeval = 0;
byte old_brake = 1;

int delay_Alt = 0;
boolean enablegoto = false;


void fWrite (const byte what)
{
  rs485.write (what);
}

int fAvailable ()
{
  return rs485.available ();
}

int fRead ()
{
  return rs485.read ();
}

void setup()
```

```
{
  rs485.begin (baudRate);
  Serial.begin(baudRate);
  Serial.println("¡RS485 Half Duplex RX listo!");
  pinMode (ENABLE_PIN, OUTPUT); // driver output enable
  pinMode(brakeswitch_az, INPUT);
  pinMode(brakeswitch_el, INPUT);

  pinMode(EN_Az, OUTPUT);
  pinMode(DIR_Az, OUTPUT);
  pinMode(PUL_Az, OUTPUT);

  digitalWrite(EN_Az, HIGH);
  digitalWrite(DIR_Az, LOW);
  digitalWrite(PUL_Az, LOW);

  pinMode(EN_Alt, OUTPUT);
  pinMode(DIR_Alt, OUTPUT);
  pinMode(PUL_Alt, OUTPUT);

  digitalWrite(EN_Alt, HIGH);
  digitalWrite(DIR_Alt, LOW);
  digitalWrite(PUL_Alt, LOW);

}

void loop()
{
  brakevalue_az = digitalRead(brakeswitch_az); //digital read of brake pin
  brakevalue_el = digitalRead(brakeswitch_el);
  receiveData();
  if ((brakevalue_az == HIGH)) {
    brake = 0;
    valAz = 0;
    valAlt = 0;
    enablegoto = false;
    Serial.print("BRAKE! Output BrakeVal: ");
    Serial.println(brake);
  }

  if ((brakevalue_az == LOW)) {
    brake = 1;
```

```
    enablegoto = true;

    Serial.print("Output BrakeVal: ");
    Serial.println(brake);
  }

  updateDataToSend();
  transmitData();


} // end of loop

//=============
void receiveData(){
  newRxData == true;
  byte msg[] = {
    1,
    moveAz,
    moveAlt,
    brake
    };
  byte received = recvMsg (fAvailable, fRead, msg, sizeof (msg));

  brake = msg[3];
  valAz = msg[1];
  valAlt = msg[2];
  Serial.print("Received: Dev=");
  Serial.print(msg[0]);
  Serial.print(" ; moveAz=");
  Serial.print(valAz);
  Serial.print(" ; moveAlt=");
  Serial.print(valAlt);
  Serial.print(" - Brake=");
  Serial.println(brake);
  newRxData = false;
  return valAz, valAlt, brake;
  }
//==============================
void updateDataToSend() {
    if (millis() - prevUpdateTime >= updateInterval) {
        prevUpdateTime = millis();
        if (newTxData == false) { // ensure previous message has been sent
```

```
        if(enablegoto == true){
          goto_object();
          }
        if(enablegoto == false){
          stop_movement(EN_Az);
          stop_movement(EN_Alt);
          }
      }
   }
}
//============================
void transmitData() {
  moveAz = valAz;
  moveAlt = valAlt;
  brake;
  newTxData == true;
    // assemble message
    byte msg [] = {
       1,   // device 1
       moveAz,
       moveAlt,
       brake
    };
  if (newTxData == true) {
       // send to slave
       digitalWrite (ENABLE_PIN, HIGH); // enable sending
       sendMsg (fWrite, msg, sizeof msg);

       Serial.print("TX: Dev = ");
       Serial.print(msg[0]);
       Serial.print(" ; moveAz = ");
       Serial.print(msg[1]);
       Serial.print(" ; moveAlt=");
       Serial.print(msg[2]);
       Serial.print("; Brake = ");
       Serial.println(msg[3]);
       Serial.println("");

       digitalWrite (ENABLE_PIN, LOW); // disable sending

       newTxData = false;
   }
```

```
}
//=============

//=============
void goto_object() {
  if (valAz == 0){
    stop_movement(EN_Az);
  }
  if (valAz == 2) {
    increment_A_tel();
  }

  if (valAz == 3) {
    decrement_A_tel();
  }

  if (valAlt == 0) {
    stop_movement(EN_Alt);
  }

  if (valAlt == 4) {
    go_up();
  }

  if (valAlt == 5) {
    go_down();
  }

}
//=============
void increment_A_tel() {
  Serial.println("Move Right");
  digitalWrite(EN_Az, LOW);
  digitalWrite(DIR_Az, HIGH);
  move_step(PUL_Az);
  newTxData = true;
}
//============
void decrement_A_tel() {
  Serial.println("Move Left");
  digitalWrite(EN_Az, LOW);
  digitalWrite(DIR_Az, LOW);
```

```
  move_step(PUL_Az);
  newTxData = true;
}
//===========
void go_up() {
  Serial.println("Move Up");
  digitalWrite(EN_Alt, LOW);
  digitalWrite(DIR_Alt, LOW);
  move_step(PUL_Alt);
  newTxData = true;
}
//============
void go_down() {
  Serial.println("Move Down");
  digitalWrite(EN_Alt, LOW);
  digitalWrite(DIR_Alt, HIGH);
  move_step(PUL_Alt);
  newTxData = true;
}
//============
void stop_movement(int axis) {
  Serial.println("STOP!");
  newTxData = true;
  digitalWrite(axis, HIGH);
}
//============
void move_step(int axis){
  digitalWrite(axis, HIGH);
  delay(delay_Alt);
  digitalWrite(axis, LOW);
  delay(delay_Alt);
}
```

B. Appendix B: Python scripts for initialization and communication with the correlator

In this section we present the python scripts for initializing the correlator, as well as communicating with it to start the data acquisition and storage process. These scripts are based in the original scripts designed for the CASPER tutorials, written by Jason Manley and its corresponding modification for FiCoRI, made by Juan Camilo Guevara [9].

- Correlator Initialization.

```
#!/usr/bin/env python
'''
This script demonstrates programming an FPGA and configuring a wideband
    Pocket correlator using the Python KATCP library along with the
    katcp_wrapper distributed in the corr package. Designed for use with
    CASPER workshop Tutorial 4.
\n\n
Author: Jason Manley, August 2010.
Modified: May 2012, Medicina.
Modified: Aug 2012, Nie Jun
Modified for FiCoRI by Juan C. Guevara Gomez on February 2017
Modified for PhAraON by Juan Sebastián Hincapié Tarquino on October 2022
'''
#TODO: add support for coarse delay change
#TODO: add support for ADC histogram plotting.
#TODO: add support for determining ADC input level

import corr,time,numpy,struct,sys,logging,pylab
katcp_port=7147
def exit_fail():
    print 'FAILURE DETECTED. Log entries:\n',lh.printMessages()
    try:
        fpga.stop()
    except: pass
    raise
```

```
        exit()

    def exit_clean():
        try:
            fpga.stop()
        except: pass
        exit()

    if __name__ == '__main__':
        from optparse import OptionParser

        p = OptionParser()
        p.set_usage('correlator_init.py <ROACH_HOSTNAME_or_IP> [options]')
        p.set_description(__doc__)
        p.add_option('-l', '--acc_len', dest='acc_len', type='int',default
            =5*(2**28)/1024,
            help='Set the number of vectors to accumulate between dumps. default
                is (2^28)/1024.')
        p.add_option('-g', '--gain', dest='gain', type='int',default=25000,
            help='Set the digital gain (4bit quantisation scalar). default is
                160.')
        p.add_option('-s', '--skip', dest='skip', action='store_true',
            help='Skip reprogramming the FPGA and configuring EQ.')
        p.add_option('-b', '--bof', dest='boffile', type='str', default='',
            help='Specify the bof file to load')
        opts, args = p.parse_args(sys.argv[1:])

        if args==[]:
            print 'Please specify a ROACH board. \nExiting.'
            exit()
        else:
            roach = args[0]

        if opts.boffile != '':
            boffile = opts.boffile
        else:
            boffile = 'tut04_2022nov10_32bit_2022_Nov_10_1220.bof'

    try:
        loggers = []
        lh=corr.log_handlers.DebugLogHandler()
        logger = logging.getLogger(roach)
```

```
logger.addHandler(lh)
logger.setLevel(10)

print('Connecting to server %s on port %i... '%(roach,katcp_port)),
fpga = corr.katcp_wrapper.FpgaClient(roach, katcp_port, timeout=10,logger
    =logger)
time.sleep(1)

if fpga.is_connected():
    print 'ok\n'
else:
    print 'ERROR connecting to server %s on port %i.\n'%(roach,
        katcp_port)
    exit_fail()

print '------------------------'
print 'Programming FPGA...',
if not opts.skip:
    fpga.progdev(boffile)
    print 'done'
else:
    print 'Skipped.'

print 'Configuring fft_shift...',
fpga.write_int('fft_shift',(2**11)-1)
print 'done'

print 'Configuring accumulation period...',
fpga.write_int('acc_len',opts.acc_len)
print 'done'

print 'Resetting board, software triggering and resetting error counters
    ...',
fpga.write_int('ctrl',0)
fpga.write_int('ctrl',1<<17) #arm
fpga.write_int('ctrl',0)
fpga.write_int('ctrl',1<<18) #software trigger
fpga.write_int('ctrl',0)
fpga.write_int('ctrl',1<<18) #issue a second trigger
print 'done'

N_DEPTH = 1024
```

```
    N_WIDTH = 32
#Previous  N_WIDTH = 32

    if N_WIDTH == 32:
        format = "I" # 32-bit unsigned integer. See the struct documentation
    elif N_WIDTH == 16:
        format = "H" # 16-bit unsigned int
    elif N_WIDTH == 8:
        format = "B" # 8-bit unsigned
    else:
        print("unsupported width!")
        exit()

    coeffs = [opts.gain] * N_DEPTH
    # convert to a binary string
    coeffs_str = struct.pack(">%d%s" % (N_DEPTH, format), *coeffs)
    # write the string to the FPGA
    fpga.write('quant1_coeffs', coeffs_str)
    fpga.write('quant1_coeffs1', coeffs_str)
    fpga.write('quant1_coeffs2', coeffs_str)
    fpga.write('quant1_coeffs3', coeffs_str)

    #EQ SCALING!
    # writes only occur when the addr line changes value.
    # write blindly - don't bother checking if write was successful. Trust in
        TCP!
    print 'done'

    print "ok, all set up. Try using poco_adc_amplitudes.py or poco_plot_adc.
        py to determine the " + \
          "adc input level, then try plotting using correlator_plot_auto.py
              or correlator_plot_cross.py"

except KeyboardInterrupt:
    exit_clean()
except:
    exit_fail()

exit_clean()
```

- Acquiring and Storing the data

```
#!/usr/bin/env python
```

```
'''
Script is for saving data, designed for the Pharaon. This script is based on
    the script uses on CASPER workshop Tutorial 4 written by Jason Manley,
   August 2009, and Ficori script modified by Juan C. Guevara Gomez, 2016.
\n\n
Modified: Juan Sebastian Hincapie Tarquino, 2022.
'''
#TODO: add support for coarse delay change
#TODO: add support for ADC histogram plotting.
#TODO: add support for determining ADC input level
from __future__ import print_function
import corr,time,numpy,struct,sys,logging,pylab
import datetime
from datetime import timedelta

katcp_port=7147
UTadd=timedelta(hours=5)
def exit_fail():
    print('FAILURE DETECTED. Log entries:\n',lh.printMessages())
    try:
        fpga.stop()
    except: pass
    raise
    exit()

def exit_clean():
    try:
        fpga.stop()
    except: pass
    exit()

def get_data(baseline):

    aa_0=struct.unpack('>256l',fpga.read('dir_x0_aa_real',1024,0))
    bb_0=struct.unpack('>256l',fpga.read('dir_x0_bb_real',1024,0))
    cc_0=struct.unpack('>256l',fpga.read('dir_x0_cc_real',1024,0))
    dd_0=struct.unpack('>256l',fpga.read('dir_x0_dd_real',1024,0))

    aa_1=struct.unpack('>256l',fpga.read('dir_x1_aa_real',1024,0))
    bb_1=struct.unpack('>256l',fpga.read('dir_x1_bb_real',1024,0))
    cc_1=struct.unpack('>256l',fpga.read('dir_x1_cc_real',1024,0))
    dd_1=struct.unpack('>256l',fpga.read('dir_x1_dd_real',1024,0))
```

```
#get the data_ crosscorr...

#FULL

x0_i=struct.unpack('>256l',fpga.read('dir_x0_%s_imag'%baseline,1024,0))
x0_r=struct.unpack('>256l',fpga.read('dir_x0_%s_real'%baseline,1024,0))
x1_i=struct.unpack('>256l',fpga.read('dir_x1_%s_imag'%baseline,1024,0))
x1_r=struct.unpack('>256l',fpga.read('dir_x1_%s_real'%baseline,1024,0))

#Per Baseline
ab_x0_i=struct.unpack('>256l',fpga.read('dir_x0_ab_imag',1024,0))
ab_x0_r=struct.unpack('>256l',fpga.read('dir_x0_ab_real',1024,0))
ac_x0_i=struct.unpack('>256l',fpga.read('dir_x0_ac_imag',1024,0))
ac_x0_r=struct.unpack('>256l',fpga.read('dir_x0_ac_real',1024,0))
ad_x0_i=struct.unpack('>256l',fpga.read('dir_x0_ad_imag',1024,0))
ad_x0_r=struct.unpack('>256l',fpga.read('dir_x0_ad_real',1024,0))
bc_x0_i=struct.unpack('>256l',fpga.read('dir_x0_bc_imag',1024,0))
bc_x0_r=struct.unpack('>256l',fpga.read('dir_x0_bc_real',1024,0))
bd_x0_i=struct.unpack('>256l',fpga.read('dir_x0_bd_imag',1024,0))
bd_x0_r=struct.unpack('>256l',fpga.read('dir_x0_bd_real',1024,0))
cd_x0_i=struct.unpack('>256l',fpga.read('dir_x0_cd_imag',1024,0))
cd_x0_r=struct.unpack('>256l',fpga.read('dir_x0_cd_real',1024,0))

ab_x1_i=struct.unpack('>256l',fpga.read('dir_x1_ab_imag',1024,0))
ab_x1_r=struct.unpack('>256l',fpga.read('dir_x1_ab_real',1024,0))
ac_x1_i=struct.unpack('>256l',fpga.read('dir_x1_ac_imag',1024,0))
ac_x1_r=struct.unpack('>256l',fpga.read('dir_x1_ac_real',1024,0))
ad_x1_i=struct.unpack('>256l',fpga.read('dir_x1_ad_imag',1024,0))
ad_x1_r=struct.unpack('>256l',fpga.read('dir_x1_ad_real',1024,0))
bc_x1_i=struct.unpack('>256l',fpga.read('dir_x1_bc_imag',1024,0))
bc_x1_r=struct.unpack('>256l',fpga.read('dir_x1_bc_real',1024,0))
bd_x1_i=struct.unpack('>256l',fpga.read('dir_x1_bd_imag',1024,0))
bd_x1_r=struct.unpack('>256l',fpga.read('dir_x1_bd_real',1024,0))
cd_x1_i=struct.unpack('>256l',fpga.read('dir_x1_cd_imag',1024,0))
cd_x1_r=struct.unpack('>256l',fpga.read('dir_x1_cd_real',1024,0))

cross_corr_ab=[]
cross_corr_ac=[]
cross_corr_ad=[]
cross_corr_bc=[]
cross_corr_bd=[]
```

```
cross_corr_cd=[]
cross_corr_full=[]
auto_corr_a=[]
auto_corr_b=[]
auto_corr_c=[]
auto_corr_d=[]

for i in range(256):
    cross_corr_ab.append(complex(ab_x0_i[i], ab_x0_r[i]))
    cross_corr_ab.append(complex(ab_x1_i[i], ab_x1_r[i]))

    cross_corr_ac.append(complex(ac_x0_i[i], ac_x0_r[i]))
    cross_corr_ac.append(complex(ac_x1_i[i], ac_x1_r[i]))

    cross_corr_ad.append(complex(ad_x0_i[i], ad_x0_r[i]))
    cross_corr_ad.append(complex(ad_x1_i[i], ad_x1_r[i]))

    cross_corr_bc.append(complex(bc_x0_i[i], bc_x0_r[i]))
    cross_corr_bc.append(complex(bc_x1_i[i], bc_x1_r[i]))

    cross_corr_bd.append(complex(bd_x0_i[i], bd_x0_r[i]))
    cross_corr_bd.append(complex(bd_x1_i[i], bd_x1_r[i]))

    cross_corr_cd.append(complex(cd_x0_i[i], cd_x0_r[i]))
    cross_corr_cd.append(complex(cd_x1_i[i], cd_x1_r[i]))

    cross_corr_full.append(complex(x0_i[i], x0_r[i]))
    cross_corr_full.append(complex(x1_i[i], x1_r[i]))

    auto_corr_a.append(aa_0[i])
    auto_corr_a.append(aa_1[i])

    auto_corr_b.append(bb_0[i])
    auto_corr_b.append(bb_1[i])

    auto_corr_c.append(cc_0[i])
    auto_corr_c.append(cc_1[i])

    auto_corr_d.append(dd_0[i])
    auto_corr_d.append(dd_1[i])

return auto_corr_a, auto_corr_b, auto_corr_c, auto_corr_d, cross_corr_ab,
```

```
         cross_corr_ac, cross_corr_ad, cross_corr_bc, cross_corr_bd,
        cross_corr_cd, cross_corr_full


#START OF MAIN:

if __name__ == '__main__':
    from optparse import OptionParser

    p = OptionParser()
    p.set_usage('correlator_plot_cross.py <ROACH_HOSTNAME_or_IP> [options]')
    p.set_description(__doc__)
    p.add_option('-c', '--cross', dest='cross', type='str',default='ab',
        help='Plot this cross correlation magnitude and phase. default: ab')
    p.add_option('-C','--channel',dest='ch',action='store_true',
        help='Set plot with channel number or frequency.')
    p.add_option('-f','--frequency',dest='fr',type='float',default=1420.0,
        help='Set plot max frequency.(If -c sets to False)')
    opts, args = p.parse_args(sys.argv[1:])

    if args==[]:
        print('Please specify a ROACH board. \nExiting.')
        exit()
    else:
        roach = args[0]

    if opts.ch !=None:
        ifch = opts.ch
    else:
        ifch = False

    if ifch == False:
        if opts.fr != '':
            maxfr = opts.fr
        else:
            maxfr = 1420.0
        xaxis = numpy.arange(0.0, maxfr, maxfr*1./1024)

    baseline=opts.cross

try:
    loggers = []
```

```
lh=corr.log_handlers.DebugLogHandler()
logger = logging.getLogger(roach)
logger.addHandler(lh)
logger.setLevel(10)

print('Connecting to server %s on port %i... '%(roach,katcp_port))
fpga = corr.katcp_wrapper.FpgaClient(roach, katcp_port, timeout=40,logger
    =logger)
time.sleep(1)

if fpga.is_connected():
    print('ok\n')
else:
    print('ERROR connecting to server %s on port %i.\n'%(roach,
        katcp_port))
    exit_fail()


# start the process
print('Creating files with data')
print('Clock FPGA',fpga.est_brd_clk())

time_init=datetime.datetime.now()

dma_start = datetime.datetime(time_init.year,time_init.month,time_init.
    day,time_init.hour,time_init.minute,0)

fa = open('/home/pharaon/PharaonData/Data_Auto_Ant_A_%s.dat'%(datetime.
    datetime.now()).strftime('%Y%m%d'),'w')
fb = open('/home/pharaon/PharaonData/Data_Auto_Ant_B_%s.dat'%(datetime.
    datetime.now()).strftime('%Y%m%d'),'w')
fc = open('/home/pharaon/PharaonData/Data_Auto_Ant_C_%s.dat'%(datetime.
    datetime.now()).strftime('%Y%m%d'),'w')
fd = open('/home/pharaon/PharaonData/Data_Auto_Ant_D_%s.dat'%(datetime.
    datetime.now()).strftime('%Y%m%d'),'w')

fab = open('/home/pharaon/PharaonData/Data_Cross_phase_Ant_AB_%s.dat'%(
    datetime.datetime.now()).strftime('%Y%m%d'),'w')
fac = open('/home/pharaon/PharaonData/Data_Cross_phase_Ant_AC_%s.dat'%(
    datetime.datetime.now()).strftime('%Y%m%d'),'w')
fad = open('/home/pharaon/PharaonData/Data_Cross_phase_Ant_AD_%s.dat'%(
    datetime.datetime.now()).strftime('%Y%m%d'),'w')
```

```
    fbc = open('/home/pharaon/PharaonData/Data_Cross_phase_Ant_BC_%s.dat'%(
        datetime.datetime.now()).strftime('%Y%m%d'),'w')
    fbd = open('/home/pharaon/PharaonData/Data_Cross_phase_Ant_BD_%s.dat'%(
        datetime.datetime.now()).strftime('%Y%m%d'),'w')
    fcd = open('/home/pharaon/PharaonData/Data_Cross_phase_Ant_CD_%s.dat'%(
        datetime.datetime.now()).strftime('%Y%m%d'),'w')
    ffull = open('/home/pharaon/PharaonData/Data_Cross_phase_Ant_Full_%s.dat
        '%(datetime.datetime.now()).strftime('%Y%m%d'),'w')

    DT = timedelta(days=1)
    while counter > 0:
#             print('Integration number %d'%counter)
              time_init=datetime.datetime.now()
              dma_now = datetime.datetime(time_init.year,time_init.month,
                 time_init.day,time_init.hour,time_init.minute,0)
              if dma_now-dma_start < DT:
                    if fpga.read_uint('acc_num') == counter+1:
                          auto_corr_a, auto_corr_b, auto_corr_c,
                             auto_corr_d, cross_corr_ab, cross_corr_ac,
                             cross_corr_ad, cross_corr_bc, cross_corr_bd,
                             cross_corr_cd, cross_corr_full = get_data(
                             baseline)
                          print(str(datetime.datetime.now()+UTadd),*
                             auto_corr_a,sep=' ',file=fa)
                          print(str(datetime.datetime.now()+UTadd),*
                             auto_corr_b,sep=' ',file=fb)
                          print(str(datetime.datetime.now()+UTadd),*
                             auto_corr_c,sep=' ',file=fc)
                          print(str(datetime.datetime.now()+UTadd),*
                             auto_corr_d,sep=' ',file=fd)
                          print(str(datetime.datetime.now()+UTadd),*
                             cross_corr_ab,sep=' ',file=fab)
                          print(str(datetime.datetime.now()+UTadd),*
                             cross_corr_ac,sep=' ',file=fac)
                          print(str(datetime.datetime.now()+UTadd),*
                             cross_corr_ad,sep=' ',file=fad)
                          print(str(datetime.datetime.now()+UTadd),*
                             cross_corr_bc,sep=' ',file=fbc)
                          print(str(datetime.datetime.now()+UTadd),*
                             cross_corr_bd,sep=' ',file=fbd)
                          print(str(datetime.datetime.now()+UTadd),*
                             cross_corr_cd,sep=' ',file=fcd)
```

```
                    print(str(datetime.datetime.now()+UTadd),*
                        cross_corr_full,sep=' ',file=ffull)
                    counter = counter+1
                    print('Integration number %d'%counter)

        else:
            fa.close()
            fb.close()
            fc.close()
            fd.close()
            fab.close()
            fac.close()
            fad.close()
            fbc.close()
            fbd.close()
            fcd.close()
            ffull.close()
            dma_start = dma_now
            fa = open('/home/pharaon/PharaonData/Data_Auto_Ant_A_%s
                .dat'%(datetime.datetime.now()).strftime('%Y%m%d')
                ,'w')
            fb = open('/home/pharaon/PharaonData/Data_Auto_Ant_B_%s
                .dat'%(datetime.datetime.now()).strftime('%Y%m%d')
                ,'w')
            fc = open('/home/pharaon/PharaonData/Data_Auto_Ant_C_%s
                .dat'%(datetime.datetime.now()).strftime('%Y%m%d')
                ,'w')
            fd = open('/home/pharaon/PharaonData/Data_Auto_Ant_D_%s
                .dat'%(datetime.datetime.now()).strftime('%Y%m%d')
                ,'w')

            fab = open('/home/pharaon/PharaonData/
                Data_Cross_phase_Ant_AB_%s.dat'%(datetime.datetime.
                now()).strftime('%Y%m%d'),'w')
            fac = open('/home/pharaon/PharaonData/
                Data_Cross_phase_Ant_AC_%s.dat'%(datetime.datetime.
                now()).strftime('%Y%m%d'),'w')
            fad = open('/home/pharaon/PharaonData/
                Data_Cross_phase_Ant_AD_%s.dat'%(datetime.datetime.
                now()).strftime('%Y%m%d'),'w')
            fbc = open('/home/pharaon/PharaonData/
                Data_Cross_phase_Ant_BC_%s.dat'%(datetime.datetime.
```

```
                        now()).strftime('%Y%m%d'),'w')
                    fbd = open('/home/pharaon/PharaonData/
                        Data_Cross_phase_Ant_BD_%s.dat'%(datetime.datetime.
                        now()).strftime('%Y%m%d'),'w')
                    fcd = open('/home/pharaon/PharaonData/
                        Data_Cross_phase_Ant_CD_%s.dat'%(datetime.datetime.
                        now()).strftime('%Y%m%d'),'w')
                    ffull = open('/home/pharaon/PharaonData/
                        Data_Cross_phase_Ant_Full_%s.dat'%(datetime.
                        datetime.now()).strftime('%Y%m%d'),'w')

#                   print('Integration number %d'%counter)

#    print('Integration number %d'%counter)
    print('Saving complete. Exiting...')
except AttributeError:
    pass
except KeyboardInterrupt:
    exit_clean()
except:
    exit_fail()
exit_clean()
```

References

[1] Arnold, Steven: *Getting Started in Radio Astronomy. Beginner Projects for the Amateur*. The Patrick Moore Practical Astronomy Series, Springer Verlag, 1988

[2] Balanis, Constantin.: *Antenna Theory: Analysis and Design*. 3rd Edition. Interscience Publishers - John Wiley & Sons, 2005

[3] Condon, James J. ; Ransom, Scott M.: *Essential Radio Astronomy: Chapter 7*. 2016

[4] Faria, C. ; Stephany, S. ; Sawant, H.S. ; Cecatto, J.R. ; Fernandes, F.C.R.: Brazilian Decimetric Array (BDA) project - Phase II. In: *Solar and Stellar Variability: Impact on Earth and Planets, Proceedings International Astronomical Union* S264 (2009), p. 493–495

[5] Fielding, John: *Amateur Radio Astronomy*. 1st Edition, Reprinted. Radio Society of Great Britain, 2008

[6] Gary, Dale E.: *Physics 728, Radio Astronomy Course: Lecture # 11 - Solar Radio Emission II*. https://web.njit.edu/~gary/728/Lecture11.html. 2014. – [Web; Accessed on Apr-16-2018]

[7] Gary, Dale E. ; Hurford, Gordon J.: Radio Spectral Diagnostics. In: GARY, Dale E. (Ed.) ; KELLER, Christoph U. (Ed.): *Solar and Space Weather Radiophysics: Current Status and Future Developments*. Springer Science + Business Media, 2005, Chapter 4

[8] Gasiorowicz, Stephen: *Quantum Physics*. 3rd Edition. John Wiley & Sons, 2003

[9] Guevara Gómez, Juan C.: *Design and Development of a Solar Radio Interferometer of Two Elements*, Observatorio Astronómico Nacional, Facultad de Ciencias. Universidad Nacional de Colombia, M.Sc. Thesis, 2017

[10] Hanslmeier, Arnold: *The Sun and Space Weather*. 2nd Edition. Springer, 2007

[11] Hincapié Tarquino, Juan S.: *Diseño y Construcción de un Radiointerferómetro Solar de dos elementos*, Departamento de Ingeniería Eléctrica y Electrónica, Facultad de Ingeniería. Universidad Nacional de Colombia, Bachelor in Engineering Thesis, 2016

[12] Krishnan, T. ; Labrum, N.R.: The Radio Brightness on the Sun at 21 cm from Combined Eclipse and Pencil-beam Observations. In: *Australian Journal of Physics* 14 (3) (1961), p. 403–419

[13] Krüger, Albert: *Introduction to Solar Radio Astronomy and Radio Physics*. D. Reidel Publishing Company, 1979

[14] Kundu, Mukul R.: *Solar Radio Astronomy*. Interscience Publishers - John Wiley & Sons, 1965

[15] Labrum, N.R.: The Radio Brightness of the Quiet Sun at 21 cm Wavelength near Sunspot Maximum. In: *Australian Journal of Physics* 13 (4) (1960), p. 700–711

[16] National Radio Astronomy Observatory (NRAO): *The Karl G. Jansky Very Large Array: Introduction*. https://science.nrao.edu/facilities/vla/docs/manuals/oss/intro. 2020. – [Web; Accessed on Apr-21-2020]

[17] Pick, Monique: Overview of Solar Radio Physics and Interplanetary Disturbances. In: GARY, Dale E. (Ed.) ; KELLER, Christoph U. (Ed.): *Solar and Space Weather Radiophysics: Current Status and Future Developments*. Springer Science + Business Media, 2005, Chapter 2

[18] Reeve, Whitham D.: *Tower Mounted Amplifier Instruction Manual*. http://www.reeve.com/Documents/CALLISTO/TMAInstructionManual.pdf. 2016. – [Web; Accessed on Jun-16-2020]

[19] Sawant, H.S. ; Ramesh, R. ; Cecatto, J.R. et a.: Brazilian Decimetric Array (Phase I). In: *Solar Physics* 242 (2007), p. 213–220

[20] South African Radio Astronomy Observatory (SARAO): *KAT-7. Seven-dish MeerKAT precursor array*. https://www.sarao.ac.za/science-engineering/kat-7/. 2019. – [Web; Accessed on May-07-2020]

[21] South African Radio Astronomy Observatory (SARAO): *MeerKAT Radio Telescope*. https://www.sarao.ac.za/science-engineering/meerkat/about-meerkat/. 2019. – [Web; Accessed on May-07-2020]

[22] Square Kilometer Array (SKA) South Africa: *SKA South Africa Public science/engineering site: KAT-7*. http://public.ska.ac.za/kat-7. 2012. – [Web; Accessed on May-07-2020]

[23] Square Kilometer Array (SKA) South Africa: *SKA South Africa Public science/engineering site: MeerKAT*. http://public.ska.ac.za/meerkat. 2012. – [Web; Accessed on May-07-2020]

[24] Thompson, J.M.; Swenson Jr G.: *Interferometry and Synthesis in Radio Astronomy*. 3rd Edition. Springer, 2017

[25] Wilson, T. L. ; Rohlfs, K. ; Hüttemeister, S: *Tools of Radio Astronomy*. 6th Edition. Springer Verlag, 2013

[26] Zotti, Georg ; Hoffmann, Susanne M. ; Wolf, Alexander ; Chéreau, Fabien ; Chéreau, Guillaume: The Simulated Sky: Stellarium for Cultural Astronomy Research. In: *Journal of Skyscape Archaeology* 6 (2021), Mar., Nr. 2, p. 221–258