CENTRO PARA LA INSTRUMENTACIÓN ASTRONÓMICA
UNIVERSIDAD DE CONCEPCIÓN

Universidad de Concepción

DOCUMENTO INFORMATIVO.

Implementación de sensor PT100 para el proyecto Cargas de Calibración para radiotelescopio LLAMA.

Preparado por:	Rol	Fecha
Vicente Montesinos Z.	Practicante	29/01/24
Revisado por:		
Aprobado por:		

5 de febrero de 2024

Índice

1. Introducción

El Proyecto Cargas de Calibración que plantea el desarrollo de 3 cargas de calibración para el radiotelescopio LLAMA, las que tienen como objetivos elementales caracterizar el receptor de potencia total y calibrar los datos obtenidos por el radiotelescopio.

Uno de sus objetivos generales es diseñar y fabricar tres cargas de calibración con temperaturas físicas diferentes, que operen simultáneamente con las cuales sea posible observar con el receptor del radiotelescopio un promedio de temperatura 60 ∘C.

2. Objetivo

El objetivo principal de esta implementación es proporcionar un monitoreo preciso de la temperatura ambiente y el amplificador LNA que compone el sistema. Esto permitirá calcular la potencia asociada a la temperatura para poder calcular un TRX preciso el cual indicaría el ruido del receptor.

3. Componentes

Sensor de Temperatura PT100 Un sensor PT100 es un dispositivo de temperatura que utiliza un resistor de platino (Pt) para medir cambios en la temperatura. La resistencia del platino varía de manera predecible con la temperatura, permitiendo mediciones precisas y amplio rango de operación.

FPGA con PMOD AD1 Module Una FPGA (Field-Programmable Gate Array) con un ADC (Convertidor Analógico-Digital) integrado es un dispositivo electrónico que combina la versatilidad de la lógica programable con la capacidad de convertir señales analógicas a digitales. Esto permite implementar funciones digitales complejas y procesar señales analógicas directamente en el chip FPGA.

Circuito de acondicionamiento Este circuito amplifica y ajusta la débil señal de resistencia de la PT100, convirtiéndola en una señal de voltaje adecuada para su procesamiento en una tarjeta electrónica.

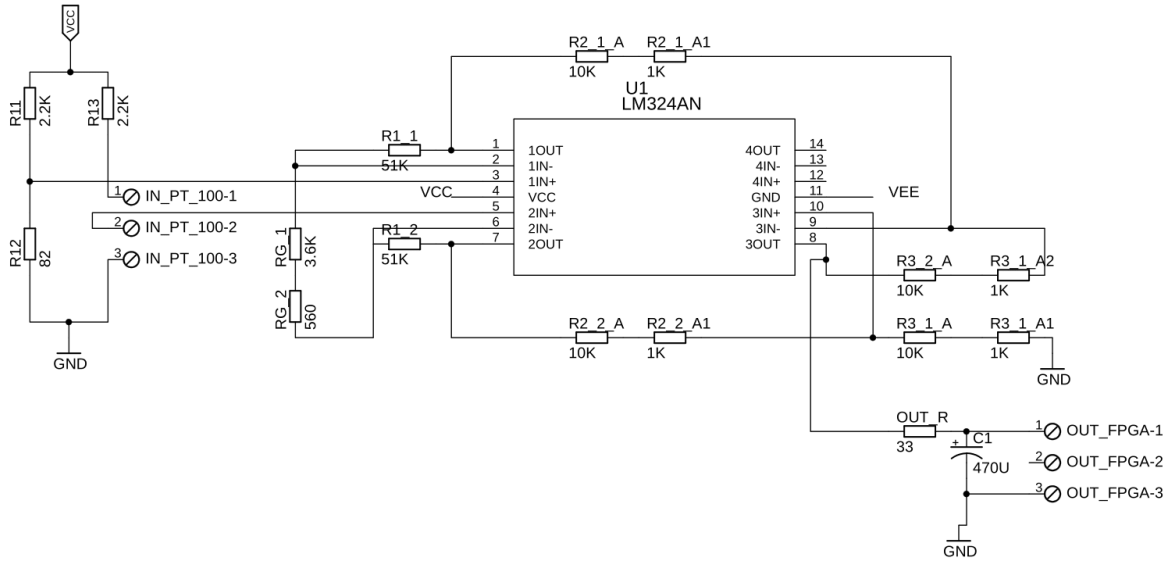


Figura 1: Circuito de Acondicionamiento.

4. Conexiones

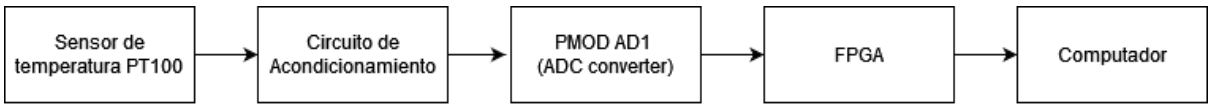


Figura 2: Diagrama de conexiones general.

4.1. Sensor de Temperatura PT100 - Circuito de Acondicionamiento

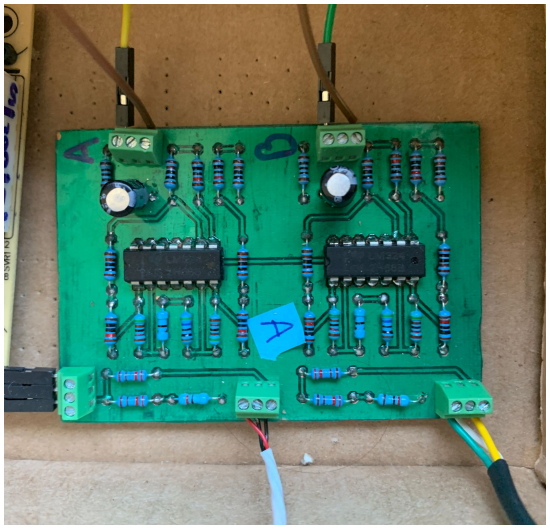


Figura 3: Circuito de Acondicionamiento en el sistema.

En este circuito, se encuentran dos puertos disponibles que permiten la conexión simultánea de dos sensores de temperatura PT100. En la parte inferior, se observan dos sensores ya enlazados al circuito.

4.2. Circuito de Acondicionamiento - PMOD AD1

En la Figura 3, en la parte superior se localizan los puertos designados para la conexión del módulo PMOD AD1.

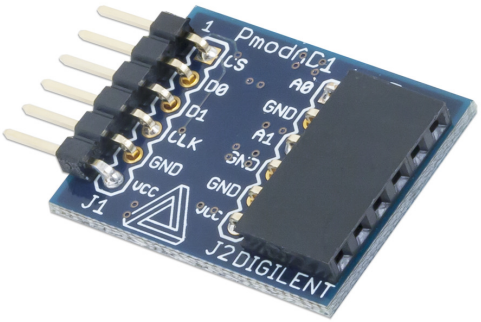


Figura 4: PMOD AD1.

La conexión al PMOD AD1 se realiza a sus puertos A0, A1 y GND. La diferenciación de sensores se verá luego en el código de recepción de datos de la FPGA.

4.3. PMOD AD1 - FPGA

La conexión del PMOD AD1 a la FPGA se lleva a cabo a través de los puertos A0, A1 y GND. La distinción entre los sensores se establecerá más adelante en el código de recepción de datos de la FPGA.

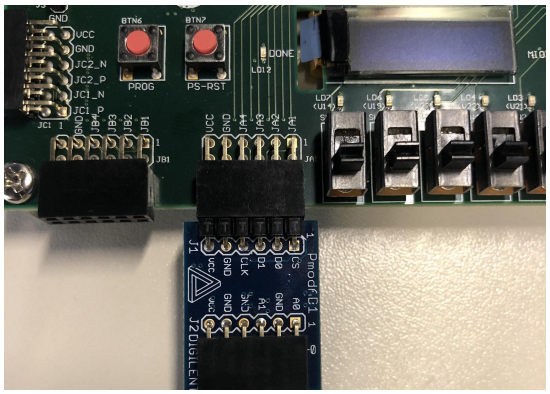


Figura 5: PMOD AD1 conectado a un PMOD Host.

4.4. FPGA - Computador

La programación de la FPGA y su adquisición de datos se llevará acabo mediante el puerto micro USB integrado en la FPGA.

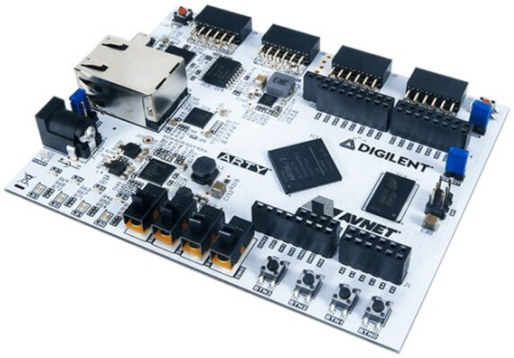


Figura 6: FPGA Arty A7 35T ocupada para desarollar el sistema.

5. Programación FPGA

El proyecto fue diseñado en Vivado 2016.1 y se encuentra disponible en Github.

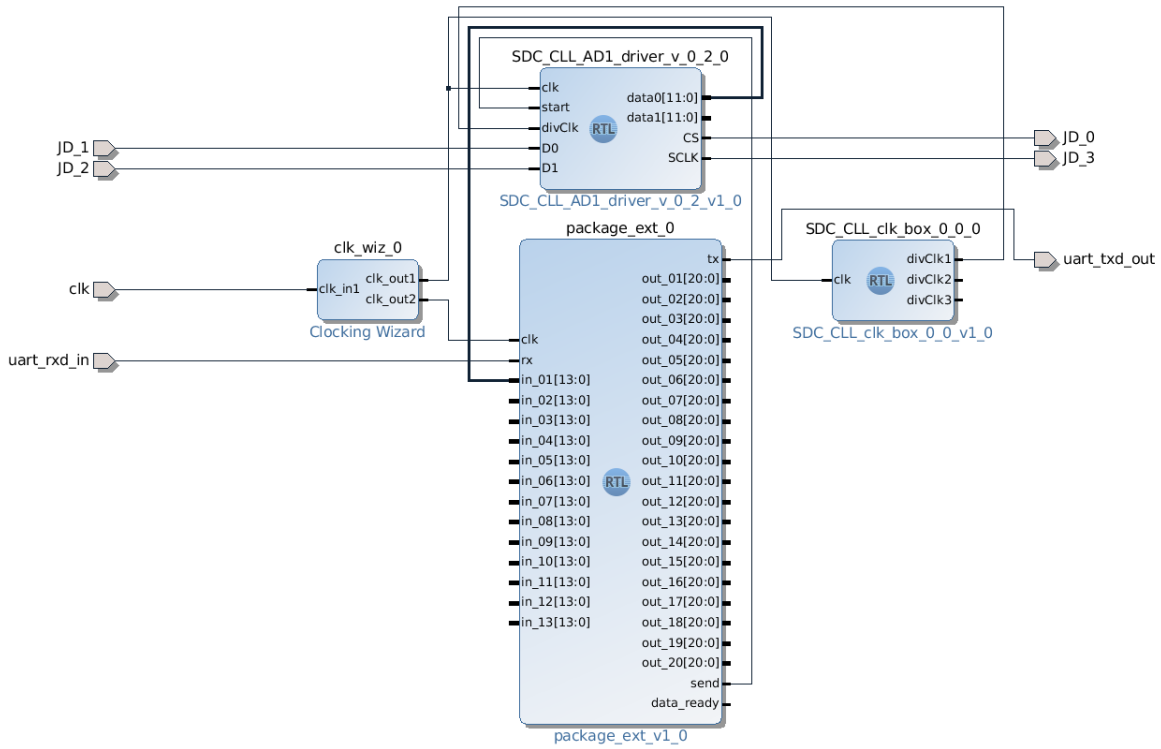


Figura 7: Arquitectura FPGA.

Mediante el controlador para el PMOD AD1 (SDC_CLL_AD1_driver), se especifica que la conexión se realiza en los puertos JD de la FPGA. Los datos obtenidos con el AD1 se transmitirán a la entrada ïn_01"del "package_ext", facilitando la comunicación con la computadora al simular un puerto COM.

6. Calibración del sensor PT100

Los datos adquiridos con la FPGA serán interpretados con Python. El código para interpretar los datos se encuentra en Github.
Para poder hacer la lectura de los datos entregados por la FPGA, se ejecuta la siguiente función:

```python
def lectura():
    global arr, last_time
    rcv = comms.read(1)  # Lee un byte desde la FPGA y lo almacena en la variable rcv.

    if rcv == b'\xc8':  # Comprueba si el byte es igual a b'\xc8' (200).
        rcv = comms.read(1)  # Lee un byte adicional.
        char_rcv = chr(rcv[0])  # Convierte el byte a su representación de carácter.
        unicode_rcv = ord(char_rcv)  # Convierte el carácter a su valor Unicode.
        arr.append(unicode_rcv)  # Añade el valor Unicode a la lista arr.

    elif rcv == b'\xc9':  # Comprueba si el byte es igual a b'\xc9' (201).
        rcv = comms.read(1)  # Lee un byte adicional.
        char_rcv = chr(rcv[0])  # Convierte el byte a su representación de carácter.
        unicode_rcv = ord(char_rcv)  # Convierte el carácter a su valor Unicode.
        arr.append(unicode_rcv)  # Añade el valor Unicode a la lista arr.

    elif rcv == b'\xca':  # Comprueba si el byte es igual a b'\xca' (202).
        arr = []  # Vacía la lista arr.

    if len(arr) > 1:  # Si la longitud de arr es mayor que 1.
        output = F_FtoFF(arr[0], arr[1])  # Ejecuta la función F_FtoFF con los dos primeros elementos de arr.
        print(output)  # Imprime resultados.

```

Figura 8: Función lectura.

Con el objetivo de calibrar los datos provenientes del sensor PT100, se conectaron dos sensores: uno vinculado a un Lakeshore 366 y otro al circuito de acondicionamiento. Ambos sensores fueron expuestos a un horno infrarrojo (T-962) que elevaba la temperatura desde 20 grados Celsius. Los datos registrados se almacenaron en el formato bits, temperatura. Posteriormente, se invirtieron los sensores y se repitió la recopilación de datos.

Con los datos ya recopilados se calculó una regresión lineal entre los bits entregados por la FPGA y la temperatura indicada por el Lakeshore, posterior a este cálculo, se calculó el Máximo Error Absoluto (Max Err Abs), el Valor Cuadrático Medio (RMS) y la Desviación Estándar (STD) de cada toma. Con la finalidad de conservar la toma que tuviese un RMS menor.

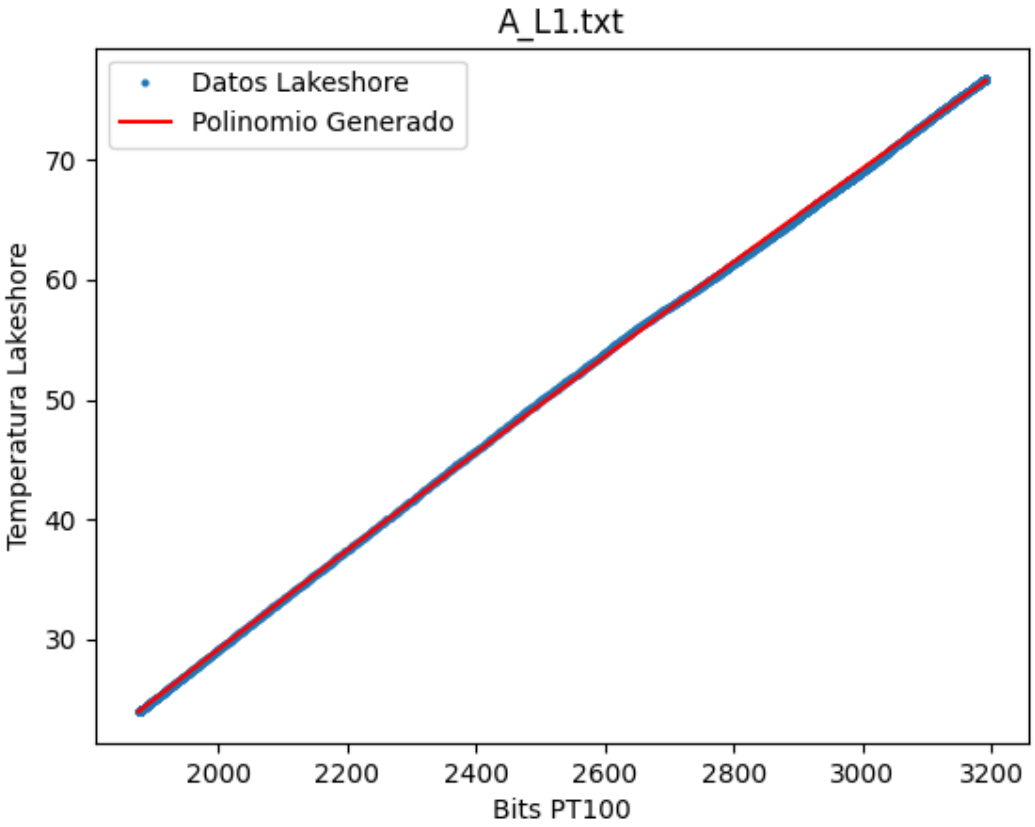


Figura 9: Puerto A PCB, Puerto 1 Lakeshore

Max Err Abs	RMS	Desv Está
±0.40946148613166855°C	0.1598448134698206	0.08967686163636691

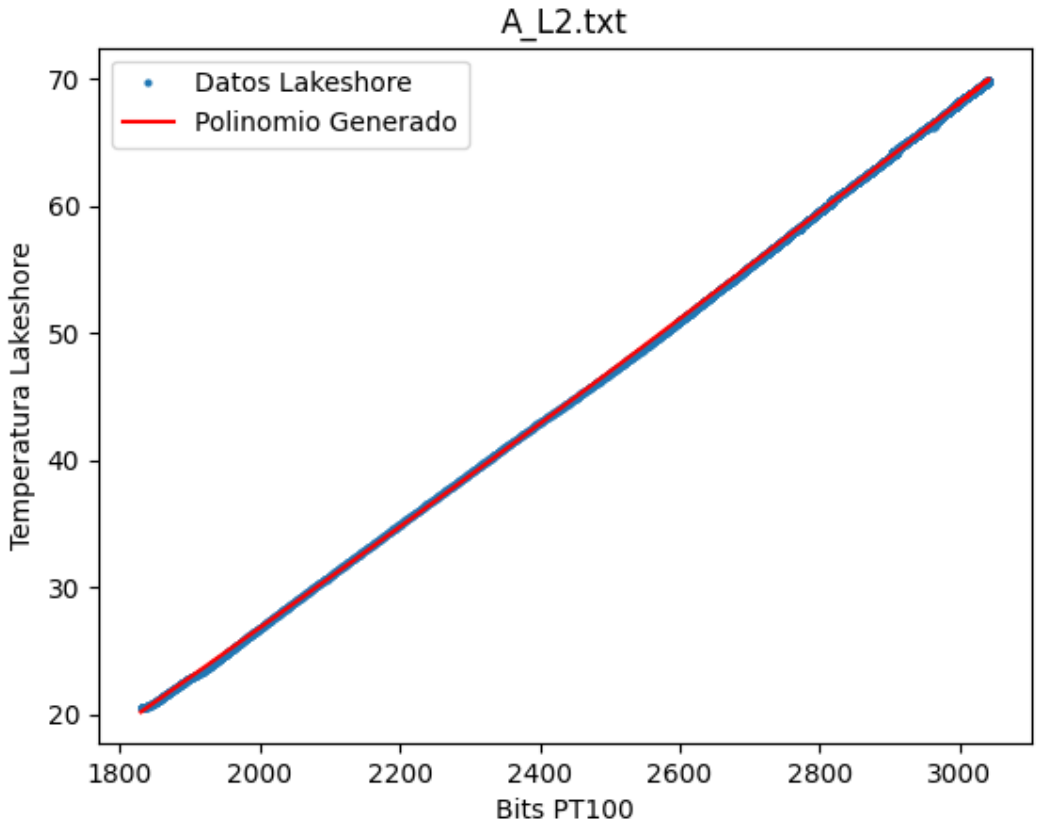


Figura 10: Puerto A PCB, Puerto 2 Lakeshore

Max Err Abs	RMS	Desv Está
±0.3634140815462956°C	0.08398227013497993	0.05598040978327

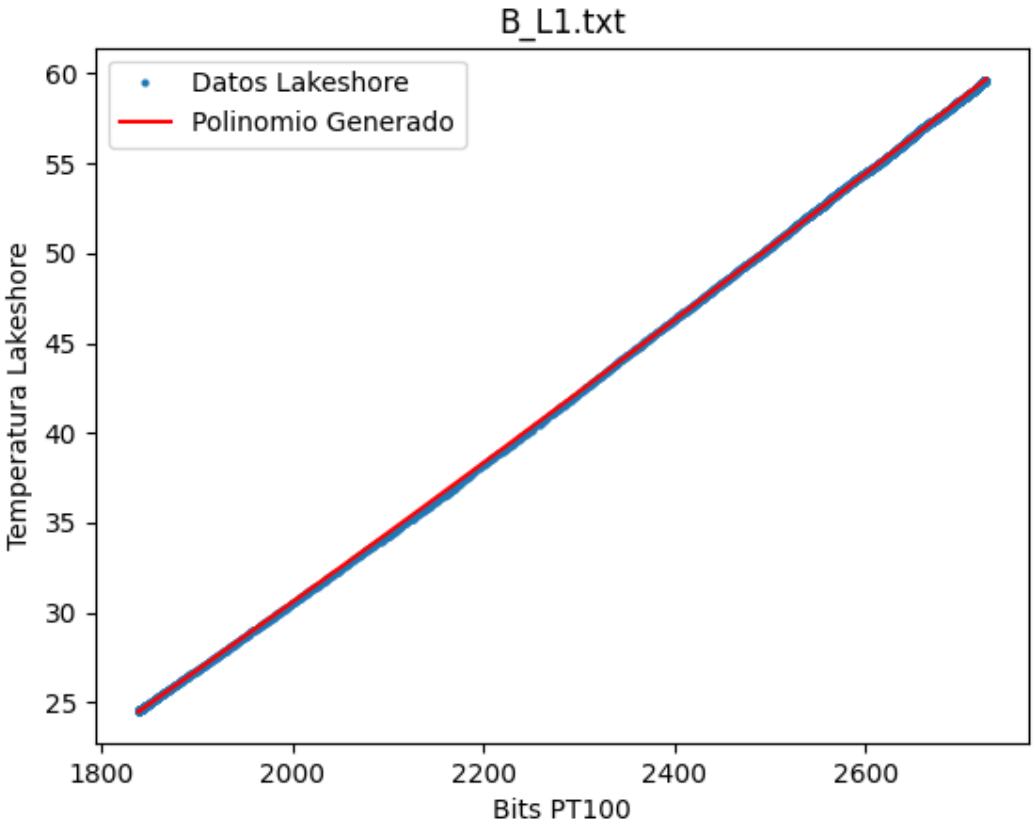


Figura 11: Puerto B PCB, Puerto 1 Lakeshore

Max Err Abs	RMS	Desv Está
±0.2243056072669063°C	0.06603126662828604	0.0394606460466

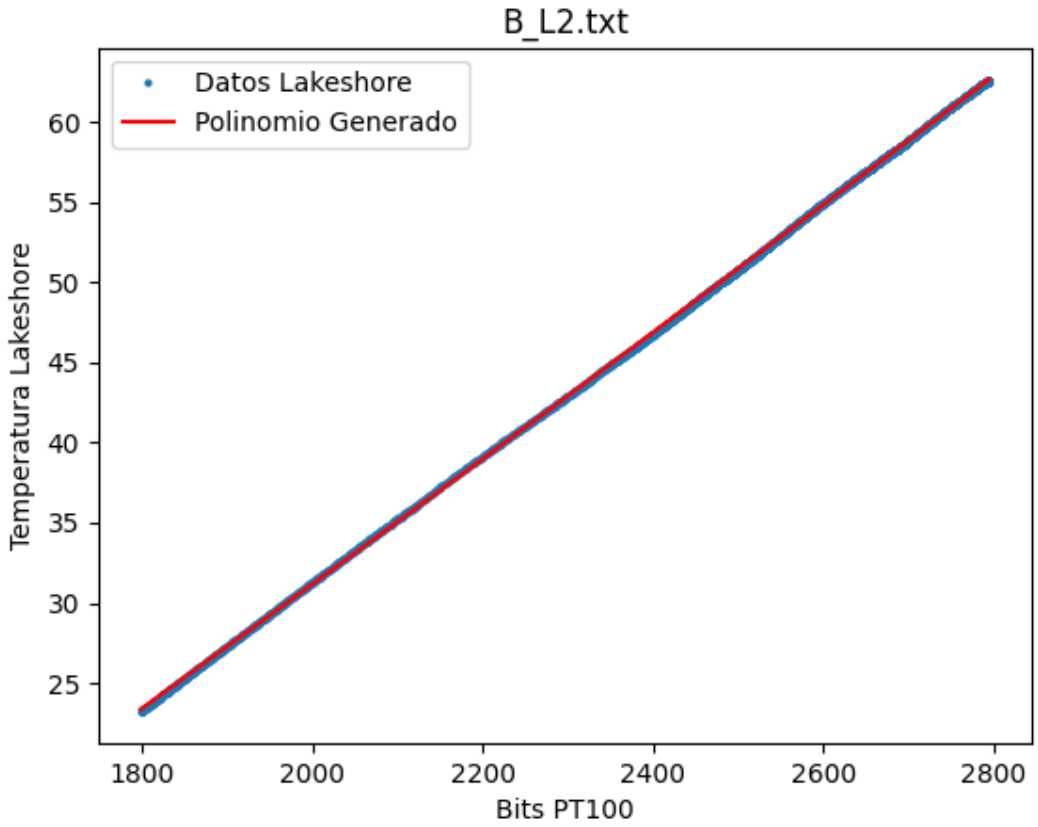


Figura 12: Puerto B PCB, Puerto 2 Lakeshore

Max Err Abs	RMS	Desv Está
±0.2404416766996036°C	0.07803436560186841	0.0457674715527

En resumen, las tomas fueron las siguientes:

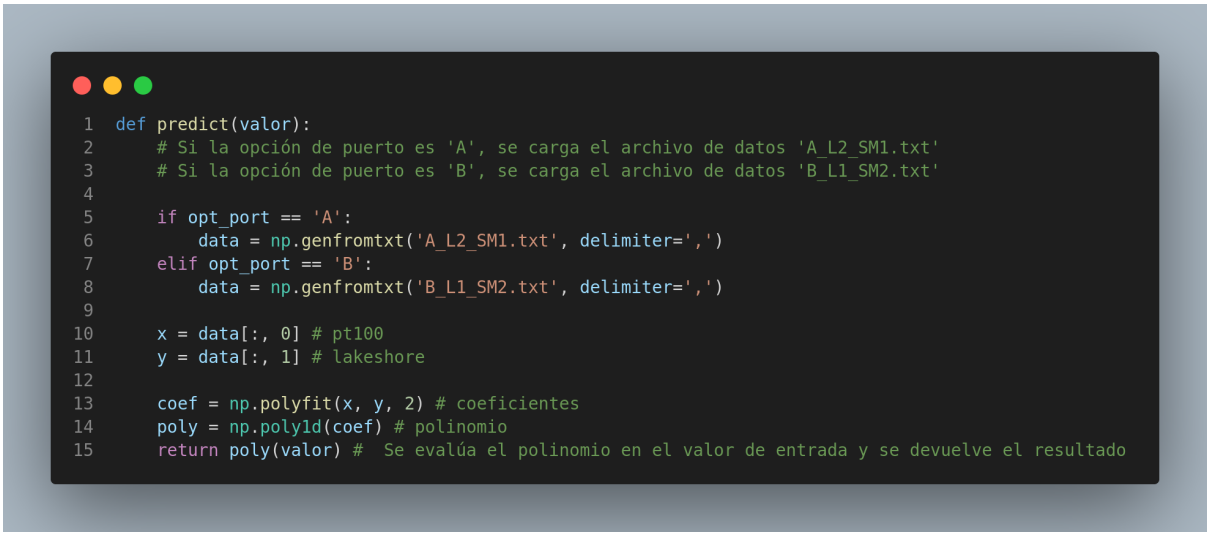
Toma	Max Err Abs	RMS	Desv Está
A_L1	±0.40946148613166855°C	0.1598448134698206	0.08967686163636691
A_L2	±0.3634140815462956°C	0.08398227013497993	0.05598040978327
B_L1	±0.2243056072669063°C	0.06603126662828604	0.0394606460466
B_L2	±0.2404416766996036°C	0.07803436560186841	0.0457674715527

Los RMS menores de cada puerto (A y B) son A_L2 y B_L1, los polinomios de cada uno son:

$$\text{A\_L2: } (2{,}033e^{-6}x^2 + 0{,}03114x - 43{,}57)$$
$$\text{B\_L1: } (2{,}511e^{-6}x^2 + 0{,}02819x - 35{,}83)$$

7. Resultados

Ahora aplicamos una función que interprete la salida de la FPGA (en bits) a nuestra función principal de lectura.



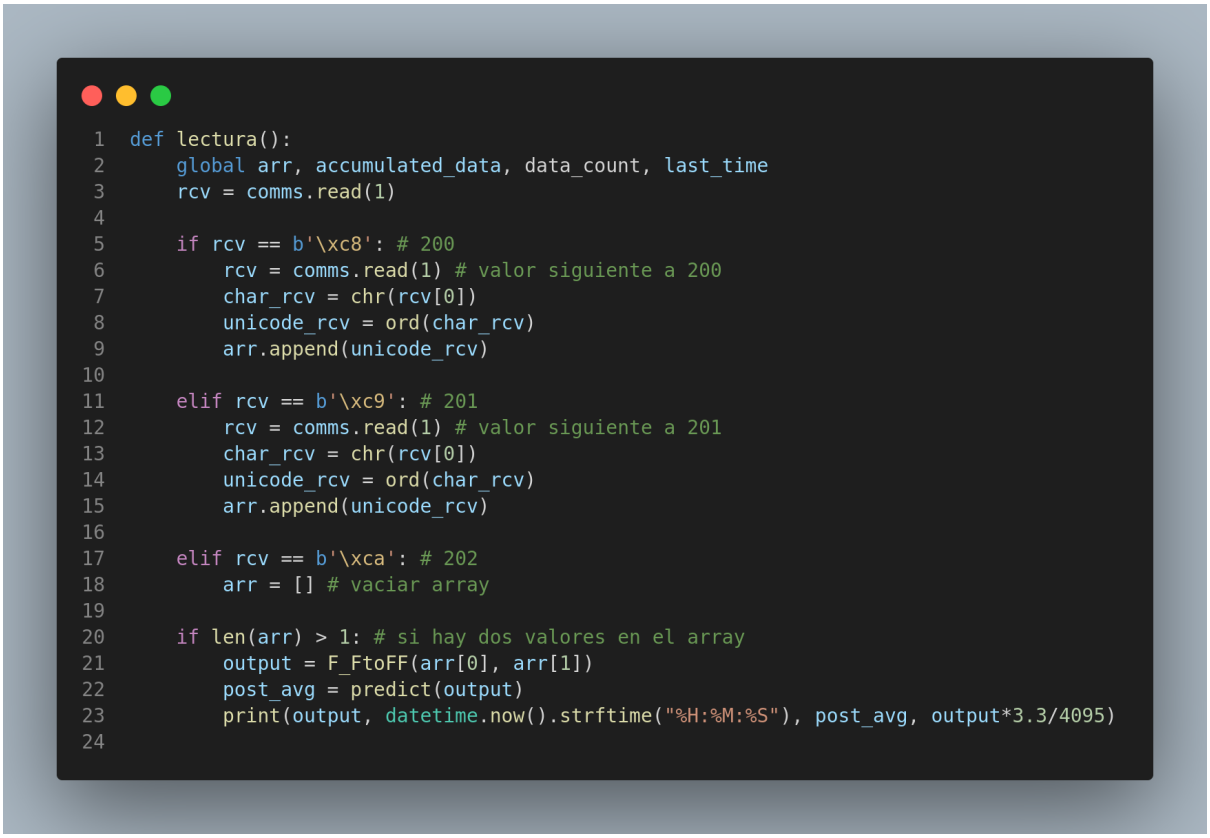
```python
def predict(valor):
    # Si la opción de puerto es 'A', se carga el archivo de datos 'A_L2_SM1.txt'
    # Si la opción de puerto es 'B', se carga el archivo de datos 'B_L1_SM2.txt'

    if opt_port == 'A':
        data = np.genfromtxt('A_L2_SM1.txt', delimiter=',')
    elif opt_port == 'B':
        data = np.genfromtxt('B_L1_SM2.txt', delimiter=',')

    x = data[:, 0] # pt100
    y = data[:, 1] # lakeshore

    coef = np.polyfit(x, y, 2) # coeficientes
    poly = np.poly1d(coef) # polinomio
    return poly(valor) #  Se evalúa el polinomio en el valor de entrada y se devuelve el resultado
```

Figura 13: Función predict.



```python
def lectura():
    global arr, accumulated_data, data_count, last_time
    rcv = comms.read(1)

    if rcv == b'\xc8': # 200
        rcv = comms.read(1) # valor siguiente a 200
        char_rcv = chr(rcv[0])
        unicode_rcv = ord(char_rcv)
        arr.append(unicode_rcv)

    elif rcv == b'\xc9': # 201
        rcv = comms.read(1) # valor siguiente a 201
        char_rcv = chr(rcv[0])
        unicode_rcv = ord(char_rcv)
        arr.append(unicode_rcv)

    elif rcv == b'\xca': # 202
        arr = [] # vaciar array

    if len(arr) > 1: # si hay dos valores en el array
        output = F_FtoFF(arr[0], arr[1])
        post_avg = predict(output)
        print(output, datetime.now().strftime("%H:%M:%S"), post_avg, output*3.3/4095)

```

Figura 14: Función lectura agregando función predict.

7.1. Prueba con temperatura ambiente.

Se realizó una prueba tratando de tener una temperatura constante.

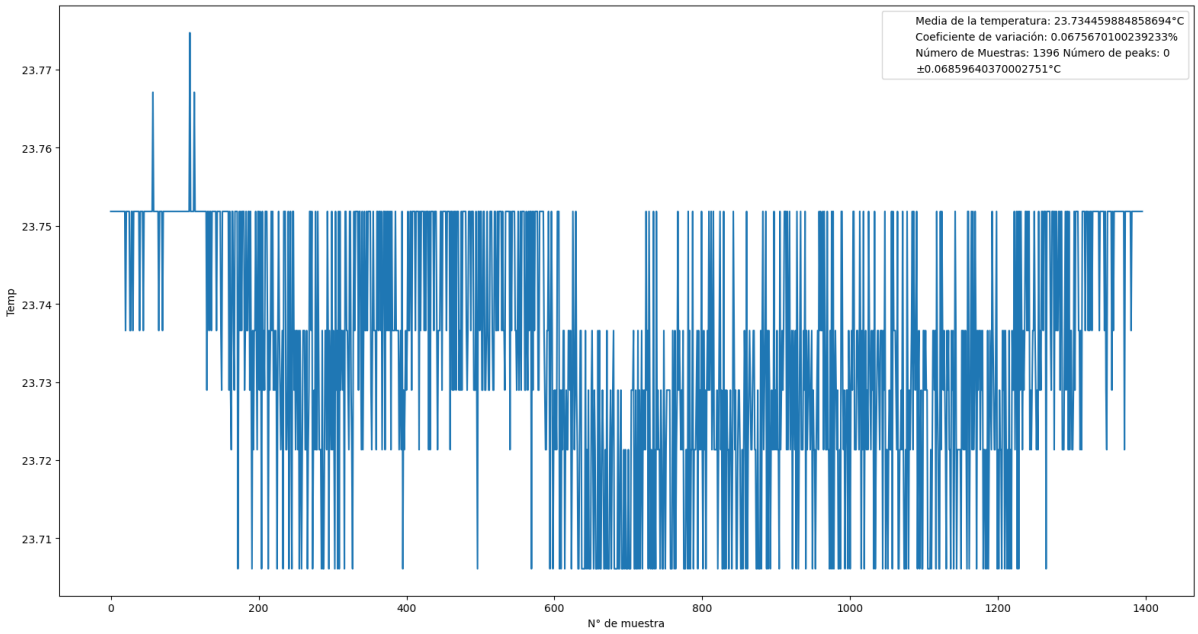


Figura 15: Resultado temperatura ambiente.

Notamos que las medidas varían un %0,06 respecto a la media. Lo que es un resultado aceptable dentro de los requerimientos de este sensor.

7.2. Prueba con variaciones

Para descartar errores con variaciones bruscas en la temperatura y que no ocurran peaks no esperados, se realizó una prueba sumergiendo los sensores en agua caliente.

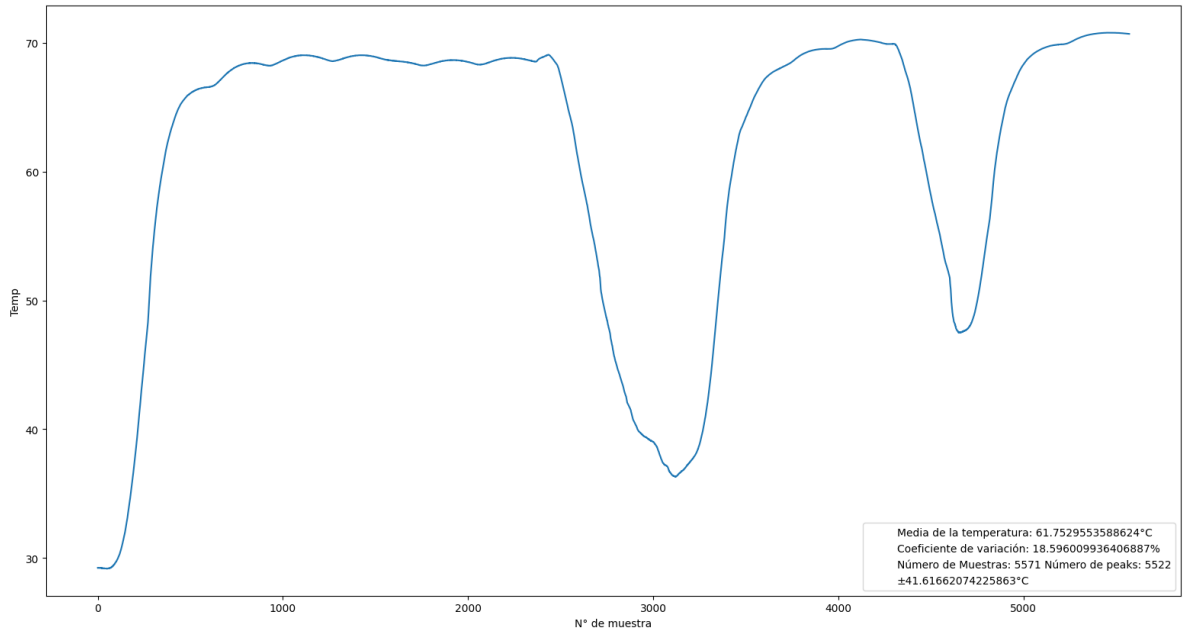


Figura 16: Resultado con variaciones de temperatura.

Notamos que no hay ningún peak que pueda ser interpretado como ruido o un error en el sistema.

8. Conclusión

En conclusión, este sistema ocupando los sensores PT100 junto a una regresión lineal, tiene una precisión de un %0,06 y en las pruebas realizadas soportó hasta temperaturas de 80°C. Satisfaciendo los requerimientos para el proyecto Cargas de Calibración para radiotelescopio LLAMA.