

Apéndice B. Descripción de bloques en GNU Radio de la arquitectura STERT1

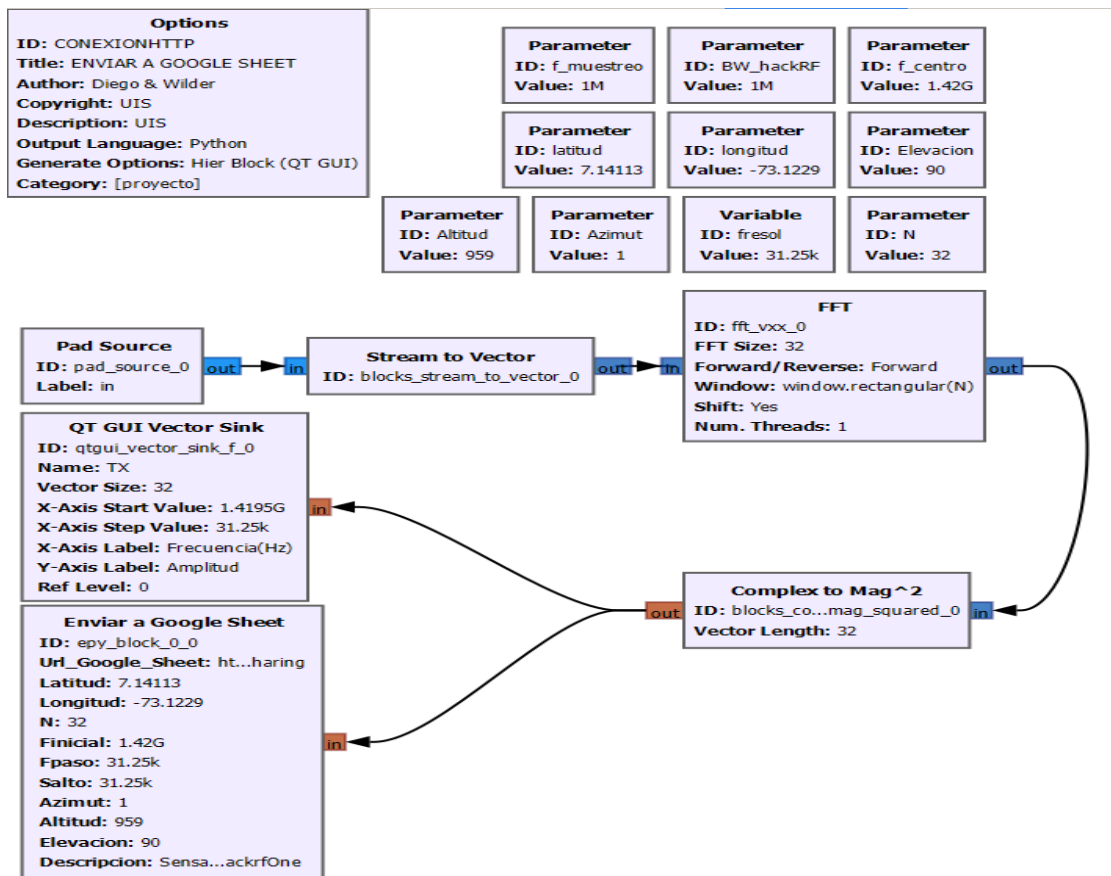
La fase de medición abarca todo el procesamiento de la señal espectral, desde su captura y tratamiento en GNU Radio hasta un bloque Python encargado de transmitirla a Internet. Es realizada por el Sistema de Transmisión de Espectro Radioeléctrico en Tiempo Real (STERT1) y está representada en el flujograma de GNU Radio

1. Bloque ENVIAR A GOOGLE SHEET

Este flujograma pertenece al bloque jerárquico creado, su función es agrupar varios bloques con sus respectivas conexiones internas en un mismo sistema para poder presentarla en otro sistema como un solo bloque, es decir, agrupamos todos estos bloques y lo transformamos en uno para ser utilizado en otros sistemas.

Figura 1

Configuración del flujograma para la creación del bloque jerárquico.



Nota. cómo ya se mencionó la razón de esta configuración es para agrupar varios bloques en uno solo que pueda ser utilizado en varios sistemas, esto es muy útil ya que no limitamos la configuración a un solo proyecto.

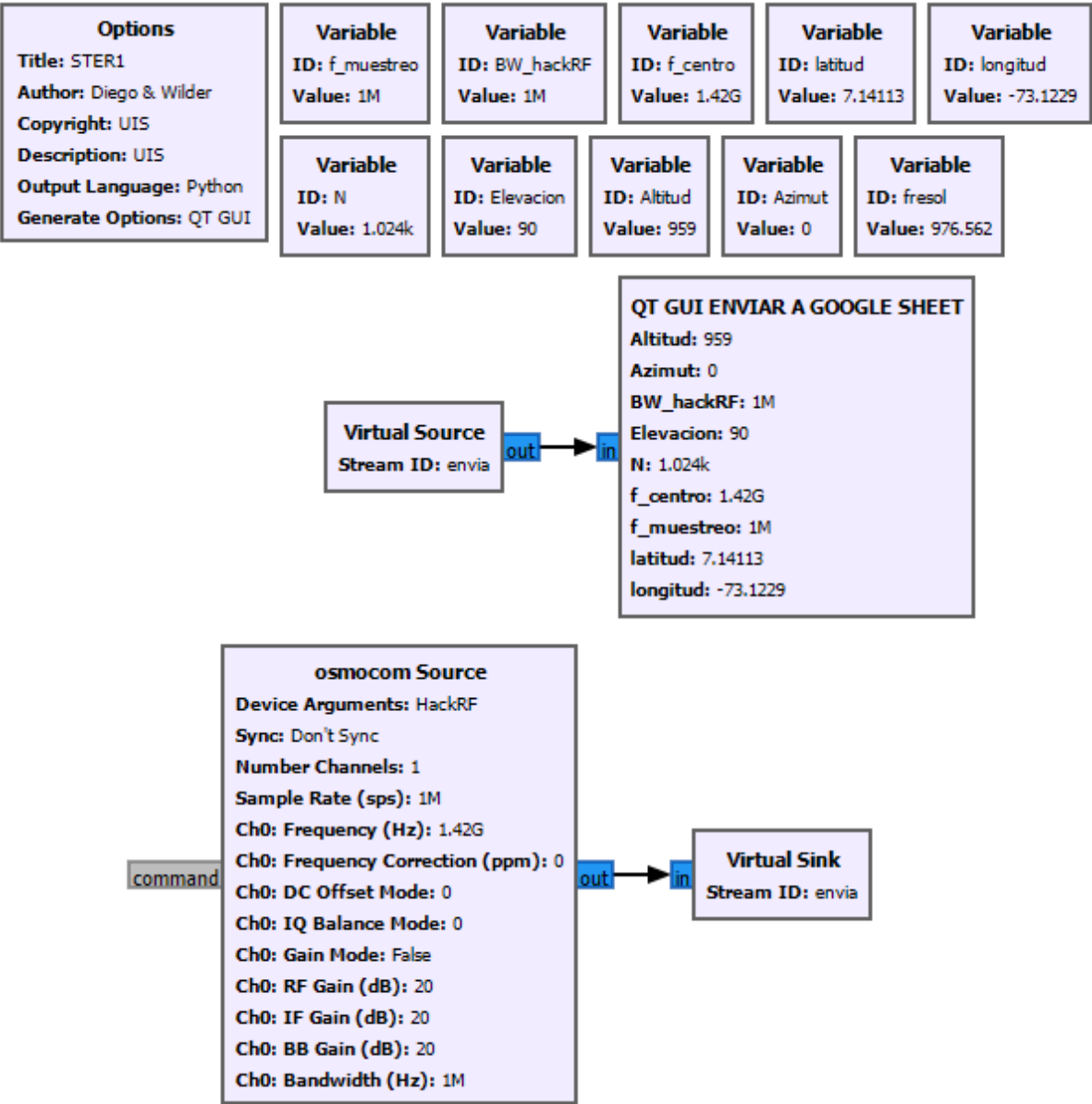
1.1 Bloque externo para la arquitectura STERT1

El flujograma STERT1 contiene al bloque jerárquico creado anteriormente, este bloque lo conecto a una entrada que se necesite, por ejemplo, una señal Signal source(coseno, seno, cuadrada,etc), señales de audio(Wav File Source) o el SDR(Osmocom Source) de esta manera simplificamos la arquitectura del sistema para que se pueda solo cambiar el valor de las variables

y la entrada del sistema. El virtual Source está reemplazando los bloques mencionados anteriormente (Signal source, Wav File Source, Osmocom Source).

Figura 2

Flujograma STERT1.



Nota. Este bloque permite la conexión y comunicación a internet registrando datos en la hoja de

cálculo, el Virtual Source/Sink en la imagen se debe reemplazar con algún bloque que permita el ingreso de datos o señales.

1.1.1 Parameters/Variable

Son los valores que la persona puede variar en el sistema con el fin de tener control en todo momento, garantizando en todo momento el control. La diferencia entre Parameters y Variable es que el bloque Variable solo puede modificar bloques de un flujograma y el bloque Parameters después de encapsularse en un bloque jerárquico puede ser modificado externamente por sistema que utilice el bloque jerárquico creado.

1.1.2 FFT

El bloque FFT recibe un vector de N muestras en el dominio del tiempo y calcula su transformada rápida de Fourier, entregando un vector complejo de N puntos en el dominio de la frecuencia. Cada elemento del resultado describe la magnitud y fase de una componente espectral concreta dentro de la ventana analizada.

1.1.3 Stream to vector

toma los valores de las muestras que llegan una a una y, cada vez que reúne N valores consecutivos, los empaqueta en un solo vector.

1.1.4 Complex to mag ^2

toma los coeficientes complejos generados por la FFT y los transforma en potencia para cada componente de frecuencia.

1.1.5 QT GUI Vector Sink

Recibe el vector y lo dibuja como una serie de puntos conectados. Cada nueva actualización reemplaza por completo la gráfica anterior, de modo que el usuario ve siempre el último vector recibido.

1.1.6 Virtual Sink/Source

el virtual source es un cable invisible, únicamente estetico para no cruzar cables

1.2 Enviar a Google Sheet

Figura 3

Código para el Envío de Datos Parte A.

```
1  import numpy as np
2  from gnuradio import gr
3  import json
4  import requests
5  from datetime import datetime
6
7  class blk(gr.sync_block):
8      def __init__(self,
9          url_google_sheet='https://docs.google.com/spreadsheets/d/12wzyVj
10         latitud=7.141879, longitud=-73.122193, N=32,
11         finicial=1417000000, fpaso=1421000000, salto=40000,
12         Azimut=0, Altitud=0, Elevacion=0, Descripcion='Lugar aqui'):
13
14         gr.sync_block.__init__(self,
15             name="Enviar a Google Sheet",
16             in_sig=[(np.float32, N)], # Recibe un vector de tamaño N
17             out_sig=None) # No tiene salida, solo envía datos
18
19         # URL del servicio web de Google Apps Script
20         self.url_web_service = 'https://script.google.com/macros/s/AKfycbws09mGt3'
```

Nota. Esta imagen pertenece a la parte A del código que permite el envío de datos a internet, lo más importante que resalta el código es el envío de constantes a la dirección del Web Services y registro a la hoja de cálculo (Google Sheet).

Figura 4

Código para el Envío de Datos Parte B.

```
21
22     # Parámetros de configuración
23     self.url_google_sheet = url_google_sheet
24     self.latitud = latitud
25     self.longitud = longitud
26     self.Azimut = Azimut
27     self.Elevacion = Elevacion
28     self.Altitud = Altitud
29     self.Descripcion = Descripcion
30     self.finicial = finicial
31     self.fpasso = fpasso
32     self.salto = salto
33     self.N = N
34
35     # Contador de entradas procesadas
36     self.entradas_procesadas = 0
37
38     def registrarDatos(self, data):
39         """
40         Función que organiza el JSON y realiza el envío de los datos.
41         """
42         # Convertir el array de datos a una cadena separada por comas
43         data_str = ','.join(map(str, data))
44
45         # Obtener la fecha y hora actuales
46         fecha_actual = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
```

Nota. Esta imagen pertenece a la parte B del código que permite el envío de datos a internet, lo más importante que resalta el código es el envío de constantes a la dirección del Web Services y registro a la hoja de cálculo (Google Sheet).

Figura 5

Código para el Envío de Datos Parte C.

```
47
48     # Crear el JSON con los datos
49     json_data = {
50         "ordentipo": "crear",
51         "url": self.url_google_sheet,
52         "numeroHoja": 0,
53         "filaencabezados": 1,
54         "columnaId": 1,
55         "datos": [
56             fecha_actual,      # Fecha y hora actual
57             self.latitud,      # Latitud
58             self.longitud,     # Longitud
59             self.Azimut,       # Azimut
60             self.Elevacion,    # Elevación
61             self.Altitud,      # Altitud
62             self.Descripcion,  # Descripción
63             self.finicial,     # Frecuencia inicial
64             self.fpasso,       # Paso de frecuencia
65             self.N,            # Número de muestras
66             data_str            # Datos espectrales como string
67         ]
68     }
69
70     # Enviar el JSON como solicitud POST al servicio web
```

Nota. Esta imagen pertenece a la parte C del código que permite el envío de datos a internet, lo más importante que resalta el código es el envío de constantes a la dirección del Web Services y registro a la hoja de cálculo (Google Sheet).

Figura 6

Código para el Envío de Datos Parte D.

```

71         try:
72             response = requests.post(self.url_web_service, json=json_data)
73             response.raise_for_status()
74             print(f"Datos enviados exitosamente: {response.text}")
75         except requests.exceptions.RequestException as e:
76             print(f"Error enviando datos: {e}")
77
78     def work(self, input_items, output_items):
79         """
80         Procesa los datos de entrada y los envía cuando se cumple la condición.
81         """
82         # Incrementar el contador de entradas procesadas
83         self.entradas_procesadas += len(input_items[0])
84
85         # Enviar datos cuando el número de entradas procesadas alcanza el "salto"
86         if self.entradas_procesadas >= self.salto:
87             self.entradas_procesadas = 0 # Resetear el contador
88
89             # Obtener los datos de entrada
90             data = input_items[0][0] # Datos recibidos como vector de tamaño N
91
92             # Enviar los datos al Google Sheet
93             self.registrarDatos(data)
94
95         return len(input_items[0]) # Indica que se procesaron todas las muestras de entrada

```

Nota. Esta imagen pertenece a la parte D del código que permite el envío de datos a internet, lo más importante que resalta el código es el envío de constantes a la dirección del Web Services y registro a la hoja de cálculo (Google Sheet).

1.2.1 ¿Qué es este bloque y para qué sirve?.

Este bloque es responsable de conectar GNU Radio con internet, es decir, este bloque funciona como un puente para enviar información desde GNU Radio al webservice y guardarla en la nube, para ser más precisos en una hoja de cálculo de Google Sheets.

1.2.2 Bibliotecas que se cargan al principio.

El archivo importa cinco librerías:

numpy maneja los arreglos de números que llegan desde GNU Radio, gr permite definir el bloque dentro del framework de GNU Radio, json prepara el mensaje que viajará por Internet,

requests realiza la conexión HTTP hacia Google Apps Script, datetime genera la fecha y la hora exactas de cada captura.

1.2.3 Funciones de las líneas de código.

Para dar un mejor entendimiento del código podemos explicar la razón de esta manera:

`class blk(gr.sync_block)`. Aquí definimos un nuevo bloque de GNU Radio llamado blk. Al heredar de `gr.sync_block`, el bloque trabaja en “modo síncrono”, es decir, recibe y procesa un lote de datos a la vez.

`def __init__(...)` Este método es el constructor del bloque. Se ejecuta una sola vez al iniciar el programa y sirve para fijar parámetros como la dirección de la hoja de cálculo, las coordenadas, el número de muestras y otros detalles.

`url_google_sheet=...`: dirección de tu hoja de cálculo en Google

`latitud, longitud, N, finicial, fpaso, salto, Azimut, Altitud, Elevacion, Descripcion:` valores que describen el lugar, la cantidad de datos y los ajustes de frecuencia y posición.

`gr.sync_block.__init__(...)` Dentro del constructor, llamamos al constructor padre para dar nombre al bloque (“Enviar a Google Sheet”), definir que reciba vectores de N números (`in_sig`) y que no emita nada (`out_sig=None`), ya que solo envía datos a Internet.

`self.url_web_service = 'https://script.google.com/.../exec'` Guardamos la dirección del servicio web que procesará los datos en Google Apps Script.

`self.url_google_sheet = url_google_sheet` Almacenamos localmente la dirección de la hoja de cálculo que recibirá los datos.

`self.latitud = latitud` hasta `self.N = N` Estas líneas guardan cada uno de los parámetros recibidos (coordenadas, número de muestras, frecuencias, descripciones) en variables del bloque, para usarlas an más adelante.

`self.entradas_procesadas = 0` Inicializamos un contador que nos ayudará a decidir cuándo enviar los datos, según la cantidad de muestras acumuladas.

`def registrarDatos(self, data)` Esta función organiza la información y la envía al servicio web. La llamamos cuando queremos subir un lote de datos a la hoja de cálculo.

`data_str = ','.join(map(str, data))` Convierte la lista de números en una sola cadena donde cada valor queda separado por comas, para poder incluirlos fácilmente en el mensaje JSON.

`fecha_actual = datetime.now().strftime("%Y-%m-%d %H:%M:%S")` Obtiene la fecha y hora exactas en que se hace el envío, y las convierte a texto con el formato año-mes-día hora:minuto:segundo.

`json_data = { ... }` Construye un diccionario con toda la información que se desea enviar:

Tipo de operación ("crear")

La hoja de cálculo y su configuración (hoja, encabezados, columna de identificación)

Los datos en sí: fecha, coordenadas, frecuencias, número de muestras y la cadena con todos los valores del espectro.

`response = requests.post(self.url_web_service, json=json_data)` Envía ese diccionario en formato JSON al servicio web mediante una llamada HTTP POST.

`response.raise_for_status()` Chequea si la petición tuvo éxito; si hubo un error (por ejemplo, falta de conexión o URL equivocada), detiene la ejecución y lanza una excepción.

`print(f'Datos enviados exitosamente: {response.text}')` Si todo va bien, imprime un mensaje en pantalla con la respuesta que devolvió el servidor.

`except requests.exceptions.RequestException as e:` En caso de problemas al enviar la petición, se captura el error y se muestra un mensaje con la causa.

`def work(self, input_items, output_items)` Este método se llama cada vez que llega un nuevo bloque de datos al bloque en GNU Radio. Aquí decidimos si enviamos la información o la seguimos acumulando.

`self.entradas_procesadas += len(input_items[0])` Sumamos al contador la cantidad de muestras recibidas en este lote.

`if self.entradas_procesadas >= self.salto` Comprobamos si hemos alcanzado el límite definido (salto), es decir, si ya tenemos suficientes muestras para enviar.

`self.entradas_procesadas = 0` Reiniciamos el contador para empezar a acumular desde cero.

`data = input_items[0][0]` Tomamos el primer vector de tamaño N que llegó, que contiene la forma de onda ya procesada.

`self.registrarDatos(data)` Llamamos a la función que arma el JSON y manda los datos a la hoja de cálculo.

`return len(input_items[0])` Indicamos a GNU Radio que hemos consumido todas las muestras que llegaron en este paso.

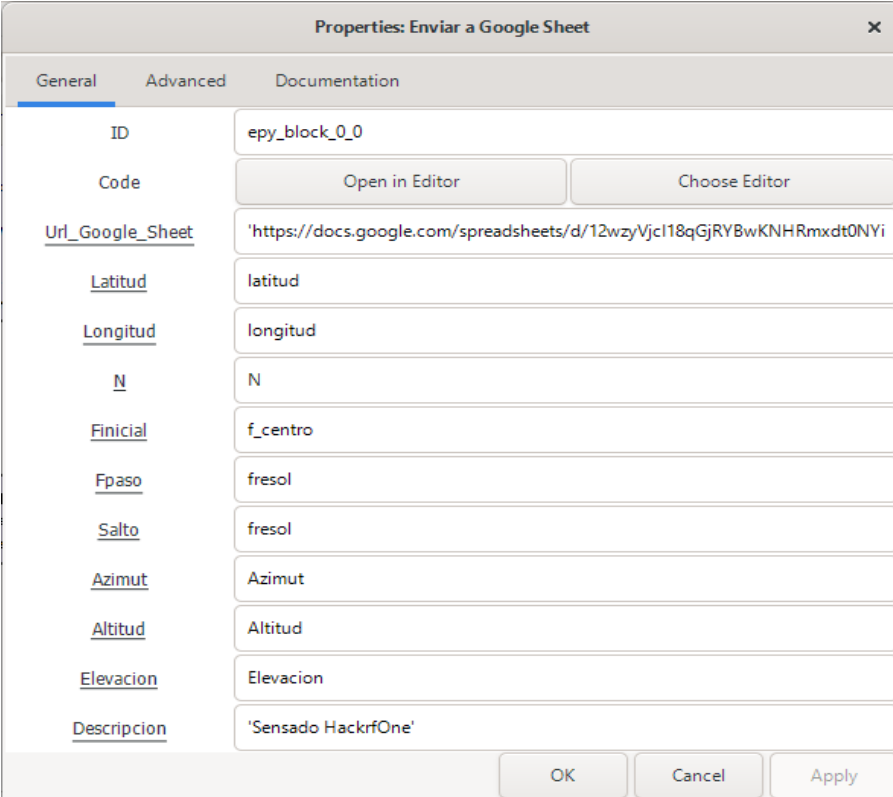
Se crea la clase del bloque (blk) Esta clase representa el bloque que añadirás en GNU Radio. Al iniciarse recibe valores como la dirección de tu hoja de cálculo, la posición geográfica, el tamaño del paquete (N) y cada cuántas mediciones quiere enviar los datos (salto).

También se fija la dirección del servicio web de Google Apps Script que guardará los datos en la hoja.

El bloque está configurado para recibir paquetes de exactamente N números y no devolver nada al diagrama; su misión es solamente subir información a Internet.

Figura 7

Configuración de las constantes.



Properties: Enviar a Google Sheet		
General	Advanced	Documentation
ID	epy_block_0_0	
Code	<div>Open in Editor</div> <div>Choose Editor</div>	
Url_Google_Sheet	'https://docs.google.com/spreadsheets/d/12wzyVjcl18qGjRYBwKNHRmxd0NYi'	
Latitud	latitud	
Longitud	longitud	
N	N	
Finicial	f_centro	
Fpaso	fresol	
Salto	fresol	
Azimut	Azimut	
Altitud	Altitud	
Elevacion	Elevacion	
Descripcion	'Sensado HackrfOne'	

OK

Cancel

Apply

Nota. Las constantes que se presentan en la imagen se controlan con el bloque Variable en GNU Radio.

Se guarda un contador de paquetes `self.entradas_procesadas` empieza en 0 y servirá para saber cuántos paquetes han pasado desde el último envío.

Función `registrarDatos` convierte el paquete de números en una sola línea de texto separada por comas, toma la fecha y hora del momento, reúne esa fecha, las coordenadas, los ángulos de la antena, los datos de frecuencia y el texto con números dentro de un solo mensaje (formato JSON), envía el mensaje al servicio web con una petición POST. Si hay un fallo de red, muestra el error y continúa.

Función `work` es que cada vez que GNU Radio le pasa un nuevo lote de paquetes, suma la cantidad recibida al contador, cuando el contador alcanza el número indicado en salto, lo pone de nuevo a 0 y toma el primer paquete del lote como muestra, llama a `registrarDatos` para que ese paquete se suba a la hoja de cálculo, devuelve a GNU Radio el número de paquetes que procesó para que el programa siga sin interrupciones.

3. Resultado.

De forma periódica, según el valor de salto, el bloque envía una respuesta del espectro acompañada de hora y lugar al servicio web, que la guarda como una nueva fila en Google Sheets. Así obtienes un registro en la nube que puedes consultar o graficar cuando se requiera.

Figura 8

Flujograma STERT1.

	A	B	C	D	E	F	G	H	I	J	K
1	Fecha	latitud	longitud	Azimut	Elevacion	Altitud	Descripcion	finicial	fpaso	N	Datos
2	25-05-20 16:59:	7,141126	-73,12289	0	90	959	Sensado HackrfOne	1420000000	4882,8125	1024	7.0491806e-06
3											
4											

Nota. En la hoja de cálculo se registran los datos enviados desde GNU Radio, donde el usuario puede ver la fecha en la que se registran los datos.

El sistema está diseñado para que registre respuesta en una misma fila, por lo tanto constantemente se va a sobrescribir la fila 2 con datos nuevos, esto se hizo con el fin de evitar un problema cómo lo es el llenado de memoria de Google cuando se registran muchos datos en varias filas de la hoja calculo.