Montaje y caracterización de un detector de potencia en la banda de 50MHz a 2.7GHz

D. Cordobés, J.A. López-Pérez, C. Almendros

Informe Técnico IT - OAN 2006 - 07

CONTENIDO

** No disponible en PDF*

I. Características técnicas del detector

En este informe se analiza un detector de potencia, basado en el circuito integrado AD8362 de Analog Devices, que presenta las siguientes características:

- Banda de frecuencias de operación: 50MHz – 2.7GHz
- Rango dinámico: 50 dB (desde –50 dBm hasta 0 dBm de potencia de entrada)
- Razón de conversión tensión de salida / potencia de entrada: 50 mV / dB

La hoja de características del AD8362 se puede consultar en el Apéndice IV.

En la Figura 1 se muestran varias medidas del detector de potencia que han sido realizadas empleando el sistema que se describe en el punto 2 de este informe.

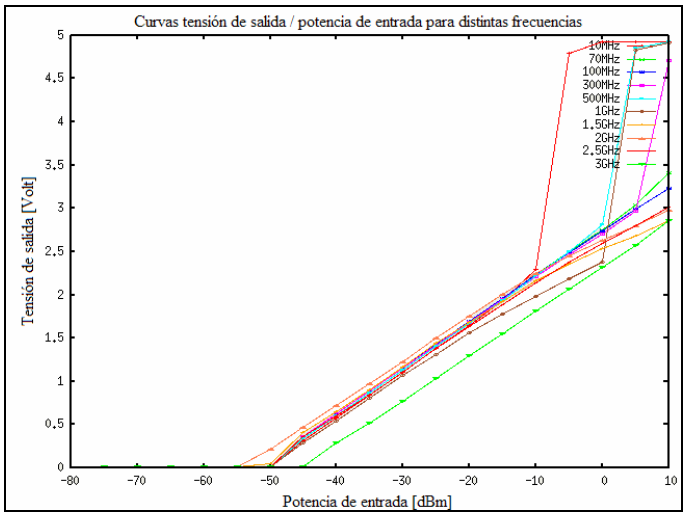

Figura 1.- Características medidas del detector de potencia

A la vista de estas especificaciones, las posibles aplicaciones futuras del detector son las siguientes:

- Detección de la potencia del módulo de frecuencia intermedia del receptor de holografía con propósitos de apuntado y monitorización de la señal de radiobaliza emitida por el satélite.

- Monitorización del estado de las etapas de frecuencia intermedia de cualquier otro receptor de radioastronomía.

II. Diseño e implementación del prototipo del detector de potencia

Para caracterizar el detector de potencia se ha implementado el circuito de la Figura 2. El aspecto final de la placa montada se presenta en la Figura 3. El *layout* se muestra en el Apéndice I.

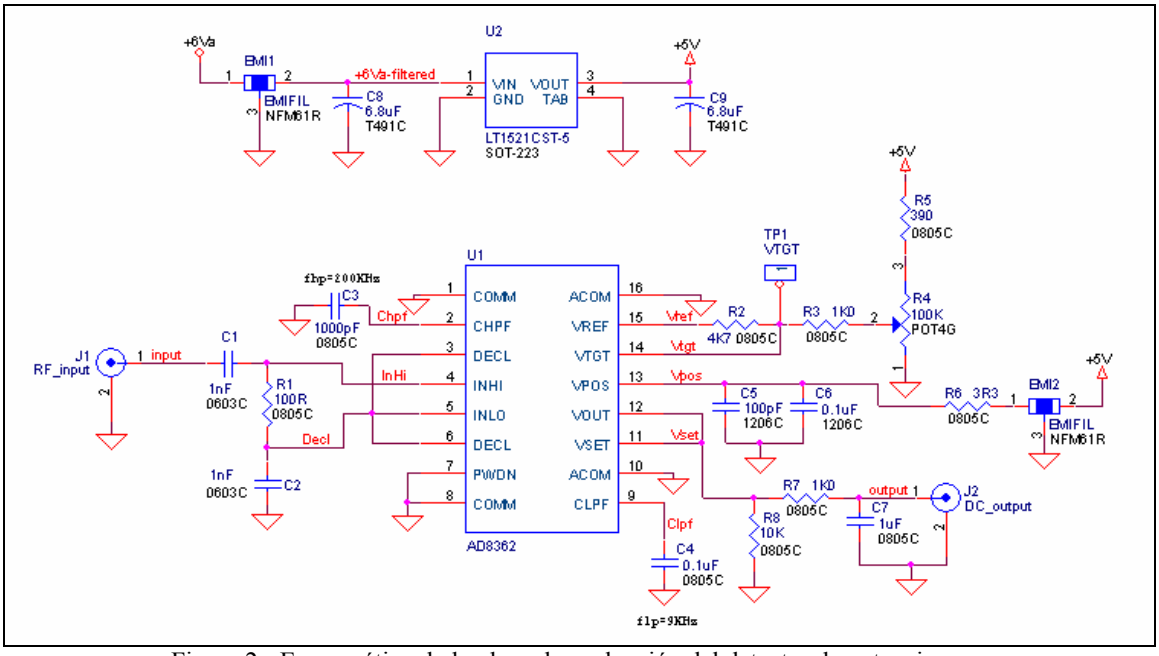

Figura 2.- Esquemático de la placa de evaluación del detector de potencia

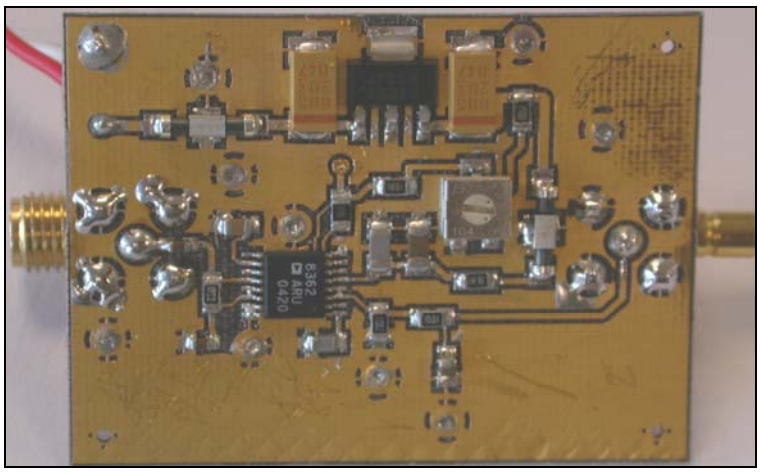

Figura 3.- Prototipo realizado del detector de potencia

2.1 Polarización del detector de potencia AD8362

A continuación se describe cómo se ha polarizado el detector de potencia AD8362:

- Pines 2 y 9 (CHPF y CLPF): CHPF sirve para establecer el ancho de banda del filtro de promediado que emplea internamente el detector de potencia. Los condensadores C3 y C4 se han elegido para conseguir una frecuencia de corte superior de 200KHz y una inferior de 9KHz, respectivamente.

- Pines 3 y 6 (DECL): Son terminales de desacoplo y se han puesto a tierra a través del condensador C2.

- Pines 4 y 5 (INHI y INLO): Son los pines a través de los cuales se excita el detector de potencia en configuración de entrada diferencial. En nuestro caso, hemos empleado una entrada unipolar ó *Single Ended*, motivo por el cual INLO se ha puesto a masa a través de C2. La resistencia interna del detector de potencia entre estos dos pines es de 100Ω, por lo que se ha empleado una resistencia R1 de otros 100Ω para hacer que la impedancia de entrada que presente el circuito sea de 50Ω. En principio, el fabricante del AD8362 establece que el detector puede operar en el rango de frecuencias de 50Hz a 2.7GHz, pero en la práctica, la presencia de la red de entrada formada por C1, R1 y C2 provoca que la frecuencia mínima de señal a aplicar al detector sea de unos 50MHz, haciendo que el rango de operación efectivo del detector sea de 50MHz a 2.7GHz.

- Pines 11 y 12 (VSET y VOUT): VSET se ha conectado a VOUT para que el filtro de promediado sea el establecido según CHPF y CLPF. En VOUT se ha implementado un filtro paso bajo RC de 200Hz.

- Pin 13 (VPOS): Este pin sirve para alimentar el detector a través de un regulador de tensión y filtros EMI.

- Pines 14 y 15 (VTGT y VREF): VTGT sirve para establecer el valor de sensibilidad del detector (potencia mínima que se ha de introducir en el detector para que este la detecte correctamente). Mediante el circuito formado por R2, R3, R4 y R5 se consigue variar la sensibilidad. VREF proporciona una tensión de 1.25 Volt.

- Pin 7 (PWDN): Sirve para habilitar / deshabilitar el AD8362.

- Pines 1, 8, 10, 16 (COMM, ACOM): Han de ser conectados a tierra.

III. Descripción del sistema de medida del detector de potencia

El sistema de medida que se ha utilizado para caracterizar el detector de potencia se muestra en la Figura 4 y está formado por tres bloques básicos:

- Generador de señal IFR 2042 (9KHz – 5.4GHz)
- Multímetro Keithley 2701
- Software de medida

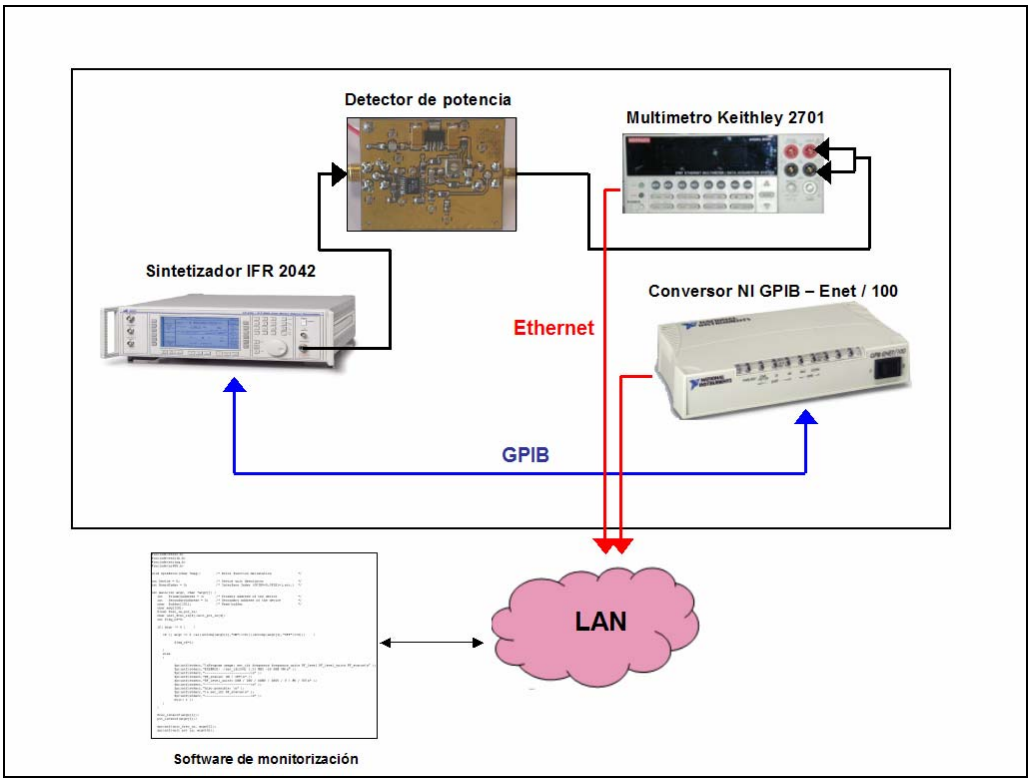

Figura 4.- Sistema de medida empleado para caracterizar el detector de potencia

La función de cada uno de los elementos del sistema de medida se detalla a continuación:

1) Generador de señal IFR 2042: Sirve para excitar al detector de potencia con la potencia y frecuencia deseadas. Ha sido comandado a través de Ethernet gracias al conversor NI Ethernet - GPIB / 100 [1].

2) Multímetro Keithley 2701: Se ha empleado para medir la tensión de salida del detector de potencia. Ha sido comandado a través de Ethernet por medio de Sockets TCP.

3) Software de medida: Es el encargado de configurar los dos elementos anteriores para realizar un barrido en potencia y frecuencia, con el fin de leer los correspondientes valores de tensión a la salida del detector, procesar y mostrar los resultados. Reúne las siguientes funcionalidades:

   - Recepción de los parámetros de ejecución desde línea de comandos.
   - Realiza una integración, del tiempo especificado por el usuario, de los datos recibidos del Keithley 2701.
   - Obtiene los coeficientes de regresión lineal que mejor ajustan los puntos obtenidos a la salida del detector.
   - Calcula el error RMS del ajuste.
   - Guarda los resultados en un fichero.

   El software se realizó en el lenguaje C [2] y se muestra en el Apéndice II.

IV. Resultados

A la hora de caracterizar el detector de potencia se han realizado estudios de linealidad y de estabilidad:

- Los de linealidad han consistido en la realización de barridos en potencia de entrada para obtener así la tensión de salida del detector en el rango de frecuencias de funcionamiento del mismo.
- Para analizar la estabilidad del detector se han hecho medidas, durante una hora, a las potencias de interés.

4.1 Linealidad del detector

Para las medidas de linealidad, se han empleado dos tipos de fuentes de excitación:

- Generador IFR 2042
- Generador de ruido blanco gaussiano

4.1.1 Generador IFR 2042

En la Figura 5 se presentan las curvas obtenidas de la tensión de salida en función de la potencia de entrada en el rango de frecuencias desde 10MHz hasta 2.7GHz con una tensión VTGT (ver punto 2.1) de 1.5 Volt. Esta tensión fue con la que el comportamiento del detector más se aproximó al comportamiento deseado, por lo que fue la que finalmente elegimos. Se hicieron también medidas a otras tensiones VTGT, que se muestran en el Apéndice III.

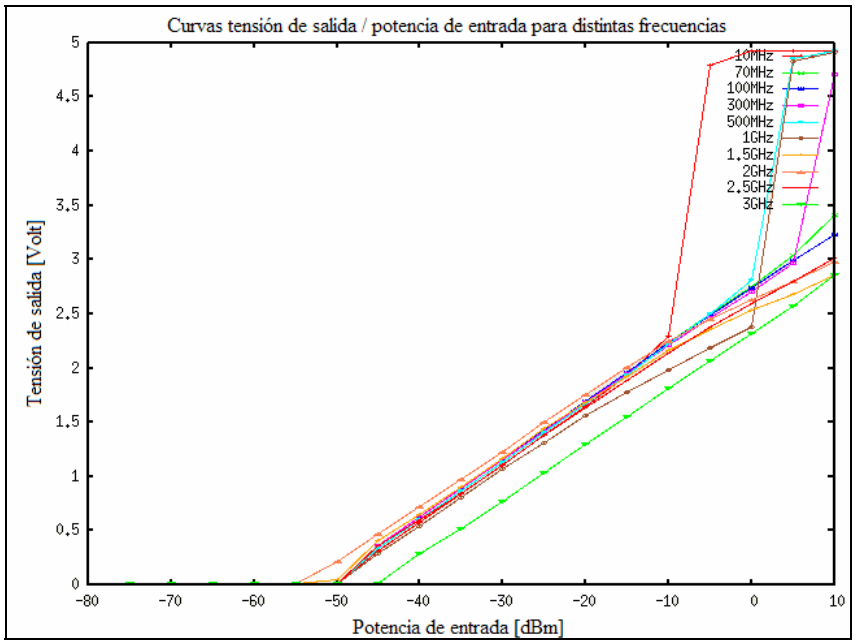

Figura 5.- Curvas medidas de Tensión de salida – Potencia de entrada (paso de 5dB) desde 10MHz hasta 2.7GHz con VTGT = 1.5 Volt.

A la vista de las curvas de la Figura 5, se puede resolver que el sensor es bastante lineal en el margen de potencias de entrada desde –50dBm hasta 0dBm. Las medidas a 10MHz y 3GHz presentan gran error ya que, en el primer caso la red capacitiva – resistiva de entrada a la placa de evaluación del detector (ver punto 2.1) fija su frecuencia inferior de trabajo en unos 50MHz y en el segundo caso se sobrepasa la frecuencia máxima de operación indicada por el fabricante (2.7GHz).

Tras realizarse barridos de potencia de entrada se miden los valores de tensión (V_i) a la salida del detector y se interpolan a través de una recta de regresión. En las Figuras 6 a 13 se muestran las curvas de regresión lineal obtenidas (F_M) para varias frecuencias, así como el error de linealidad [dBV] (definido como $20 \cdot \log\left(F_{M_i} - V_i\right)$). En las Tablas 1 a 9 se presentan las rectas de regresión estimadas, el error RMS y el error de linealidad obtenido.

4.1.1.1 70MHz

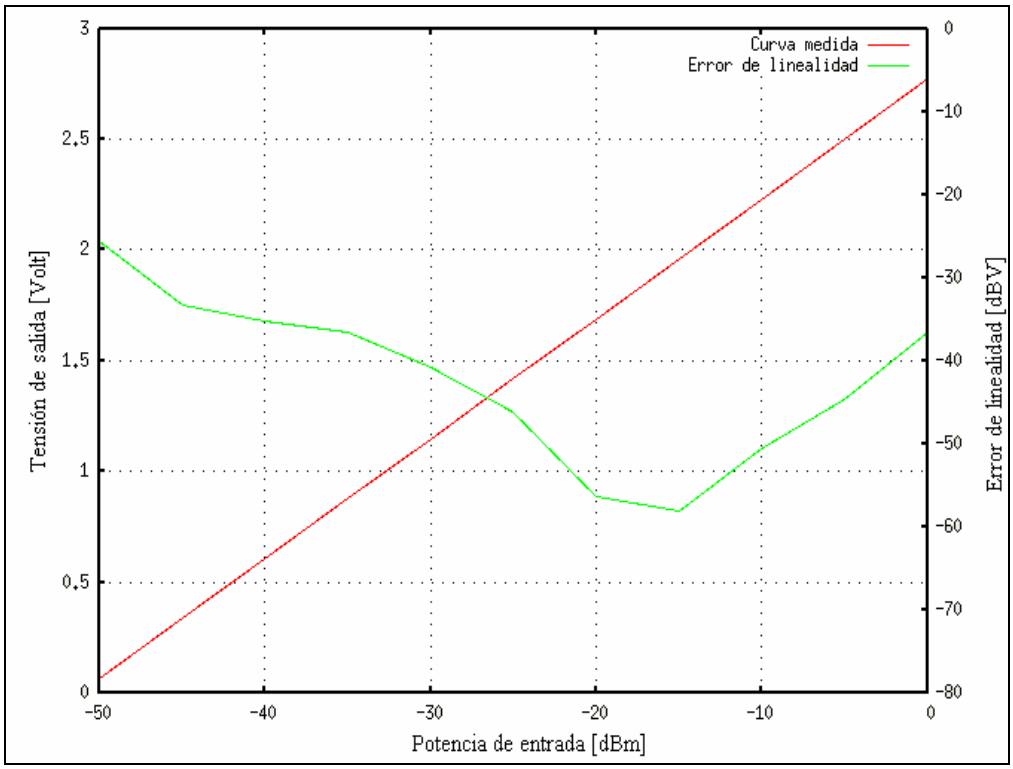

Figura 6.- Curva de conversión a 70MHz

Recta de regresión obtenida (F_M)	**Error RMS** $\sqrt{\frac{\sum_{i=1}^{N}(V_i - F_{M_i})^2}{N}}$	**Error de linealidad pico a pico** $MAX(V_i - F_{M_i}) - MIN(V_i - F_{M_i})$, i = 1..N
$F_M[Volt] = 0.054 \cdot P_{in}[dBm] + 2.772$	0.019 V	0.074 V

Tabla 1.- Recta de regresión, error RMS y error de linealidad a 70MHz

4.1.1.2 100MHz

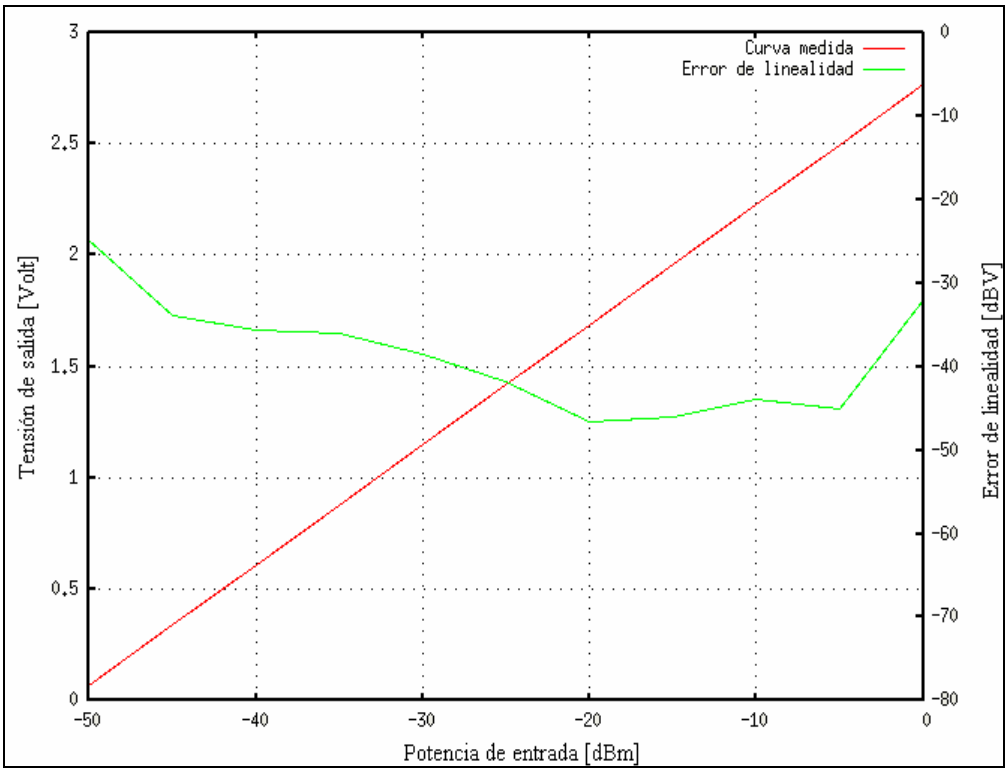

Figura 7.- Curva de conversión a 100MHz

Recta de regresión obtenida (F_M)	**Error RMS** $\sqrt{\frac{\sum_{i=1}^{N}(V_i - F_{M_i})^2}{N}}$	**Error de linealidad pico a pico [Volt]** $MAX(V_i - F_{M_i}) - MIN(V_i - F_{M_i})$, i = 1..N
$F_M[Volt] = 0.054 \cdot P_{in}[dBm] + 2.765$	0.0217 V	0.077 V

Tabla 2.- Recta de regresión, error RMS y error de linealidad a 100MHz

4.1.1.3 300MHz

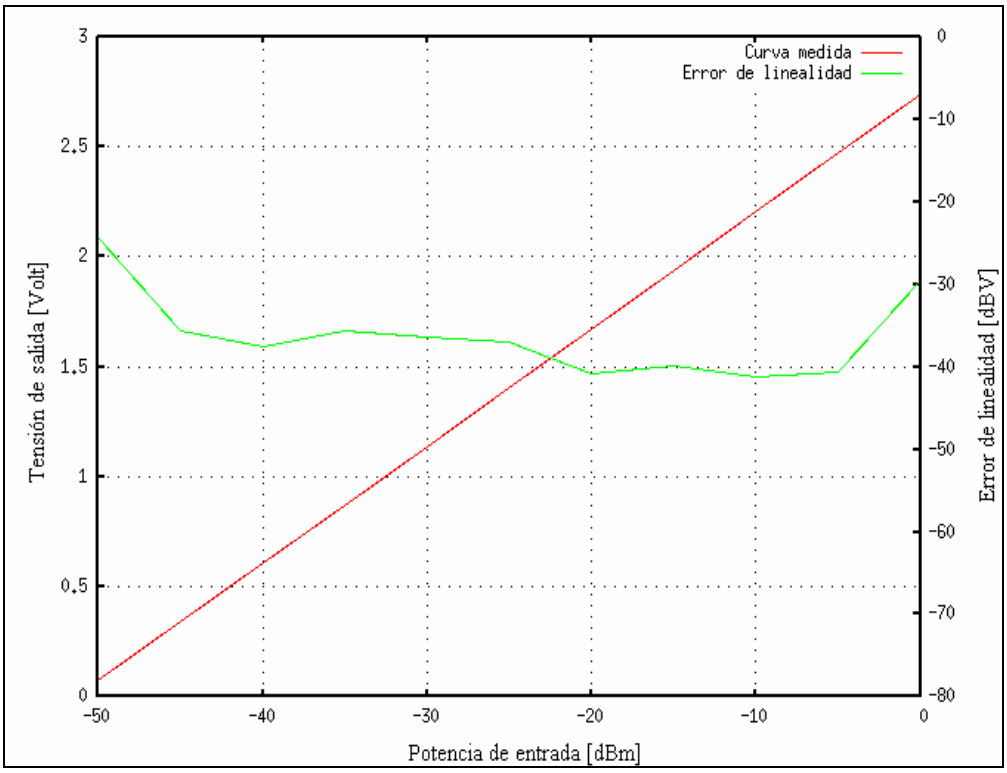

Figura 8.- Curva de conversión a 300MHz

Recta de regresión obtenida (F_M)	**Error RMS** $\sqrt{\frac{\sum_{i=1}^{N}(V_i - F_{M_i})^2}{N}}$	**Error de linealidad pico a pico [Volt]** $MAX(V_i - F_{M_i}) - MIN(V_i - F_{M_i})$, i = 1..N
$F_M[Volt] = 0.053 \cdot P_{in}[dBm] + 2.738$	0.024 V	0.077 V

Tabla 3.- Recta de regresión, error RMS y error de linealidad a 300MHz

4.1.1.4 500MHz

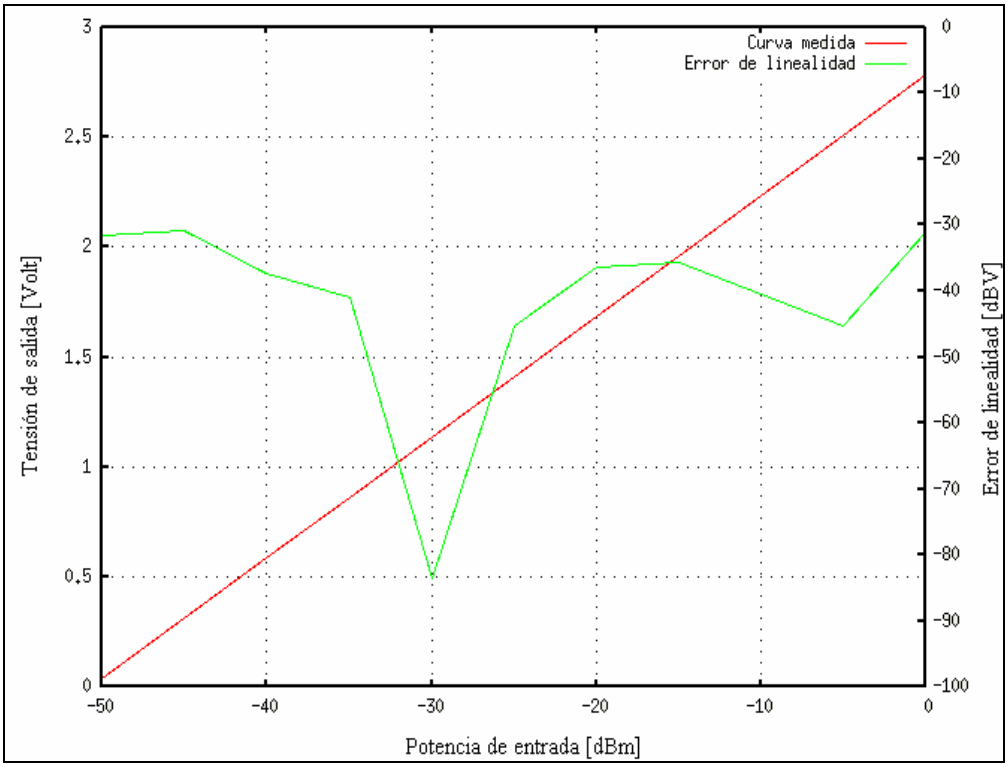

Figura 9.- Curva de conversión a 500MHz

Recta de regresión obtenida (F_M)	**Error RMS** $\sqrt{\frac{\sum_{i=1}^{N}(V_i - F_{M_i})^2}{N}}$	**Error de linealidad pico a pico [Volt]** $MAX(V_i - F_{M_i}) - MIN(V_i - F_{M_i})$, i = 1..N
$F_M[Volt] = 0.055 \cdot P_{in}[dBm] + 2.779$	0.017 V	0.055 V

Tabla 4.- Recta de regresión, error RMS y error de linealidad a 500MHz

4.1.1.5 1GHz

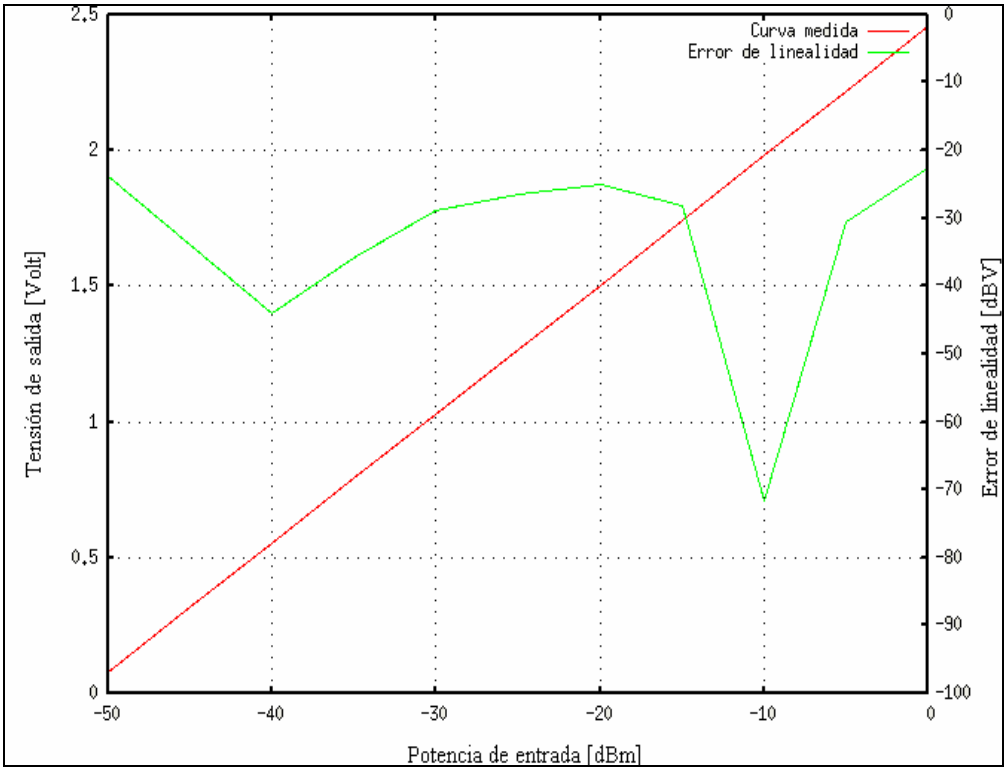

Figura 10.- Curva de conversión a 1GHz

Recta de regresión obtenida (F_M)	**Error RMS** $\sqrt{\frac{\sum_{i=1}^{N}(V_i - F_{M_i})^2}{N}}$	**Error de linealidad pico a pico [Volt]** $MAX(V_i - F_{M_i}) - MIN(V_i - F_{M_i})$, i = 1..N
$F_M[Volt] = 0.048 \cdot P_{in}[dBm] + 2.455$	0.042 V	0.13 V

Tabla 5.- Recta de regresión, error RMS y error de linealidad a 1GHz

4.1.1.6 1.5GHz

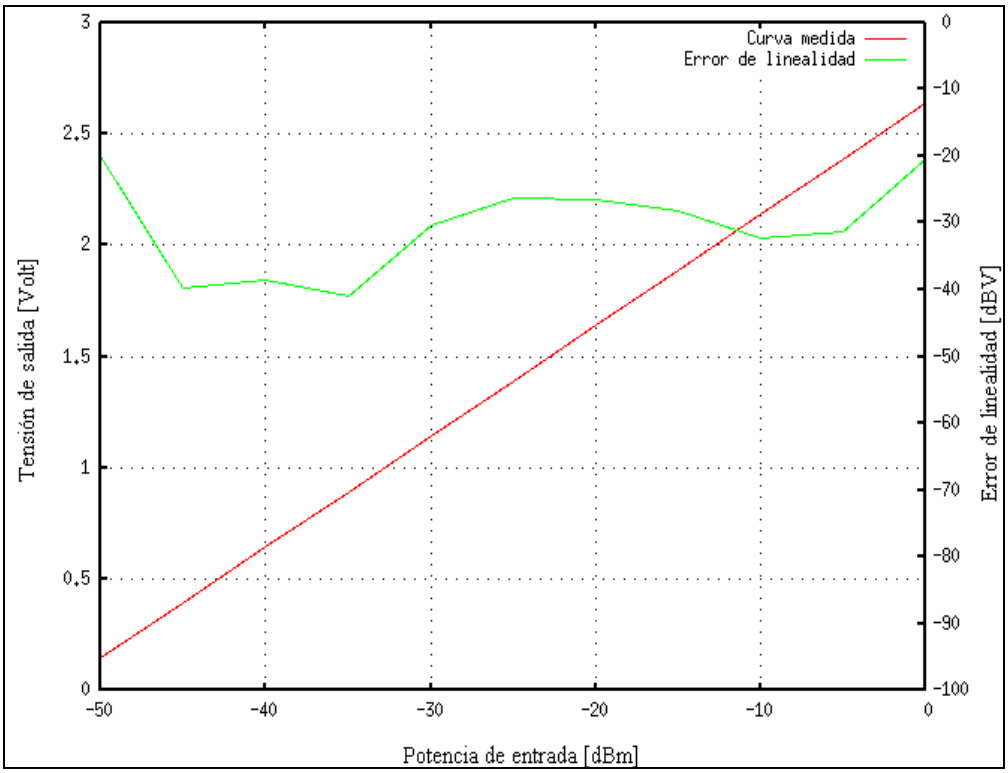

Figura 11.- Curva de conversión a 1.5GHz

Recta de regresión obtenida (F_M)	**Error RMS** $\sqrt{\frac{\sum_{i=1}^{N}(V_i - F_{M_i})^2}{N}}$	**Error de linealidad pico a pico [Volt]** $MAX(V_i - F_{M_i}) - MIN(V_i - F_{M_i})$, i = 1..N
$F_M[Volt] = 0.05 \cdot P_{in}[dBm] + 2.634$	0.05 V	0.149 V

Tabla 6.- Recta de regresión, error RMS y error de linealidad a 1.5GHz

4.1.1.7 2GHz

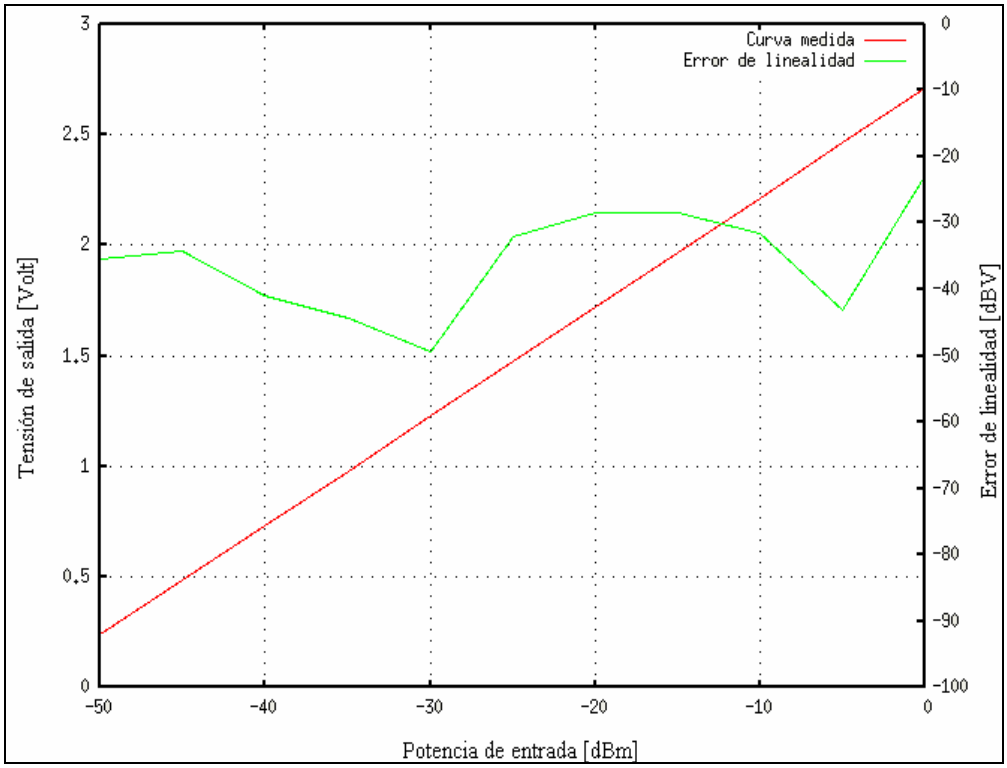

Figura 12.- Curva de conversión a 2GHz

Recta de regresión obtenida (F_M)	**Error RMS** $\sqrt{\frac{\sum_{i=1}^{N}(V_i - F_{M_i})^2}{N}}$	**Error de linealidad pico a pico [Volt]** $MAX(V_i - F_{M_i}) - MIN(V_i - F_{M_i})$, i = 1..N
$F_M[Volt] = 0.049 \cdot P_{in}[dBm] + 2.709$	0.03 V	0.111 V

Tabla 7.- Recta de regresión, error RMS y error de linealidad a 2GHz

4.1.1.8 2.5GHz

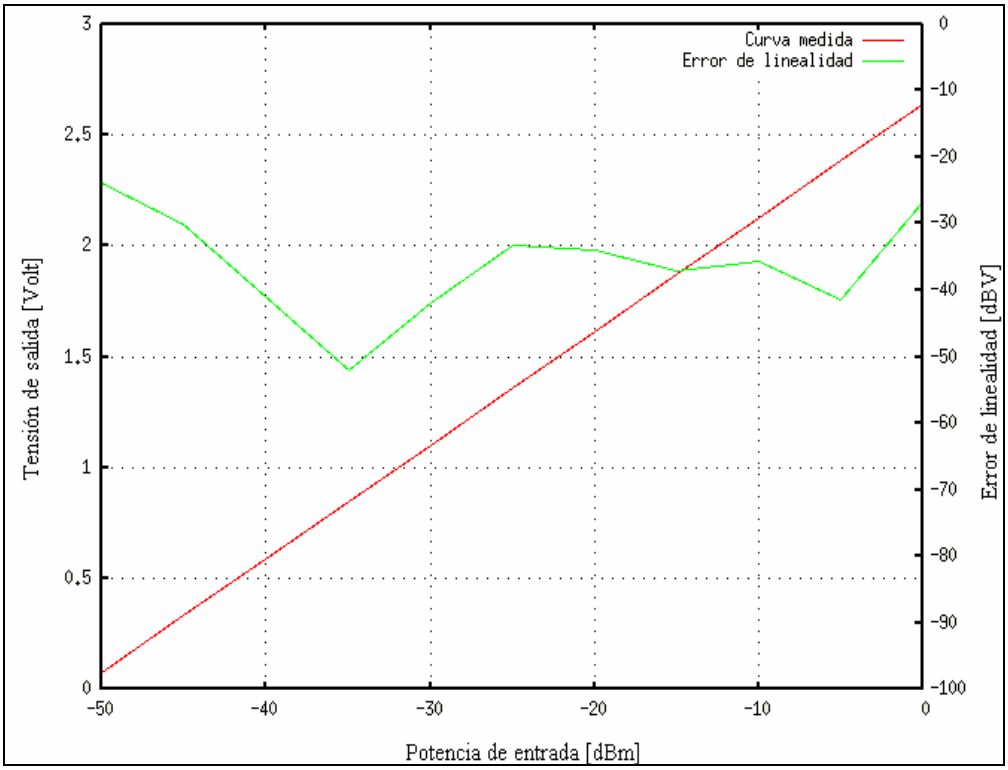

Figura 13.- Curva de conversión a 2.5GHz

Recta de regresión obtenida (F_M)	**Error RMS** $\sqrt{\frac{\sum_{i=1}^{N}(V_i - F_{M_i})^2}{N}}$	**Error de linealidad pico a pico [Volt]** $MAX(V_i - F_{M_i}) - MIN(V_i - F_{M_i})$, i = 1..N
$F_M[Volt] = 0.052 \cdot P_{in}[dBm] + 2.639$	0.028 V	0.095 V

Tabla 8.- Recta de regresión, error RMS y error de linealidad a 2.5GHz

4.1.1.9 Tabla resumen

Frecuencia	Recta de regresión	Error RMS	Error de linealidad pico a pico
70MHz	$F_M[Volt] = 0.054 \cdot P_{in}[dBm] + 2.772$	0.019 V	0.074 V
100MHz	$F_M[Volt] = 0.054 \cdot P_{in}[dBm] + 2.765$	0.022 V	0.077 V
300MHz	$F_M[Volt] = 0.053 \cdot P_{in}[dBm] + 2.738$	0.024 V	0.077 V
500MHz	$F_M[Volt] = 0.055 \cdot P_{in}[dBm] + 2.779$	0.017 V	0.055 V
1GHz	$F_M[Volt] = 0.048 \cdot P_{in}[dBm] + 2.455$	0.042 V	0.13 V
1.5GHz	$F_M[Volt] = 0.05 \cdot P_{in}[dBm] + 2.634$	0.05 V	0.149 V
2GHz	$F_M[Volt] = 0.049 \cdot P_{in}[dBm] + 2.709$	0.03 V	0.111 V
2.5GHz	$F_M[Volt] = 0.052 \cdot P_{in}[dBm] + 2.639$	0.028 V	0.095 V

Tabla 9.- Recta de regresión, error RMS y error de linealidad desde 70MHZ hasta 2.5GHz

4.1.2 Generador de ruido blanco gaussiano

Para este caso se ha empleado un generador de ruido blanco gaussiano que trabaja en la banda desde 10MHz hasta 1.1GHz. El resultado de la medida se presenta en la Figura 14.

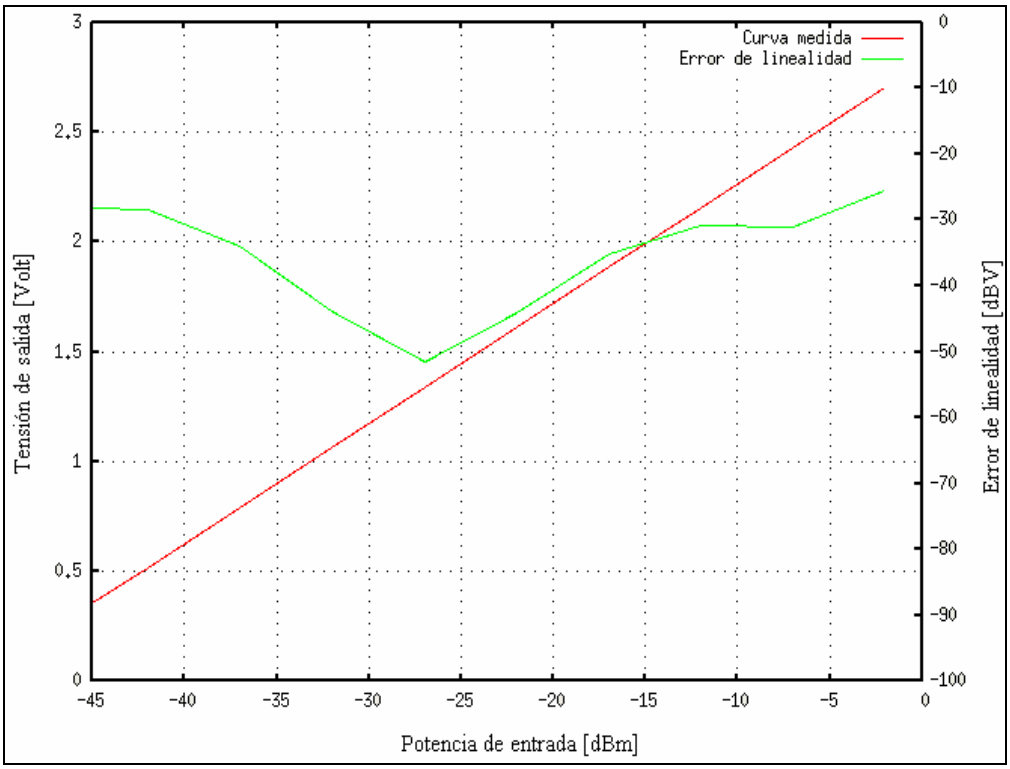

Figura 14.- Curva de conversión con la fuente de ruido blanco gaussiano

Recta de regresión obtenida (F_M)	**Error RMS** $\sqrt{\frac{\sum_{i=1}^{N}(V_i - F_{M_i})^2}{N}}$	**Error de linealidad pico a pico [Volt]** $MAX(V_i - F_M) - MIN(V_i - F_M)$, i = 1..N
$F_M[Volt] = 0.055 \cdot P_{in}[dBm] + 2.81$	0.0008 V	0.09 V

Tabla 10.- Recta de regresión, error RMS y error de linealidad con la fuente de ruido blanco gaussiano

4.2 Estabilidad del detector

Nuevamente, para las medidas de estabilidad se han empleado dos tipos de fuentes de excitación:

- Generador IFR 2042
- Generador de ruido blanco gaussiano

4.2.1 Generador IFR 2042

4.2.1.1 Potencia de entrada –40dBm

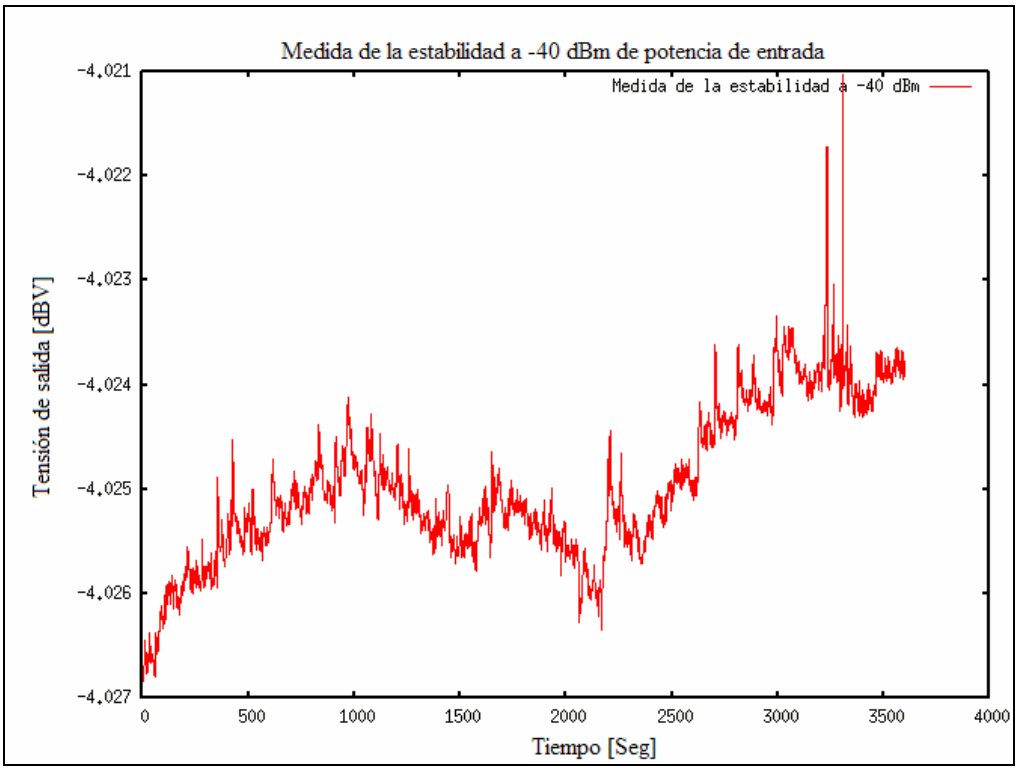

Figura 15.- Medida de la estabilidad durante una hora a la potencia de entrada de –40dBm

Variación pico a pico obtenida [dBV]
0.006

4.2.1.2 Potencia de entrada –20dBm

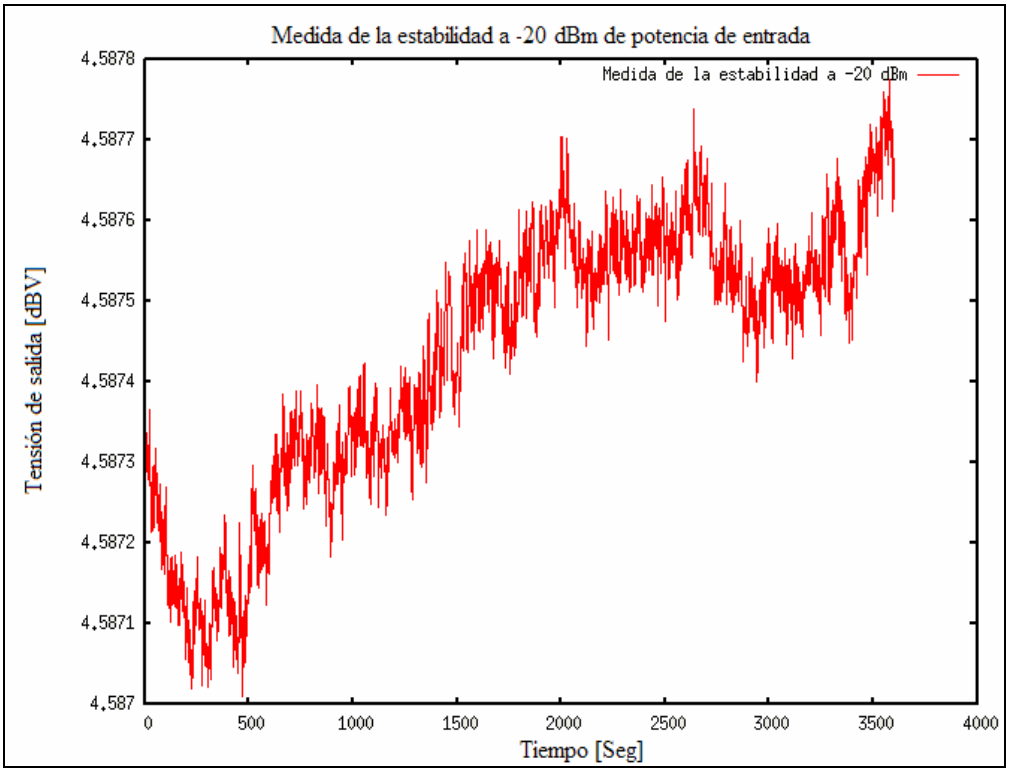

Figura 16.- Medida de la estabilidad durante una hora a la potencia de entrada de –20dBm

Variación pico a pico obtenida [dBV]
0.00077

4.2.1.3 Potencia de entrada 0dBm

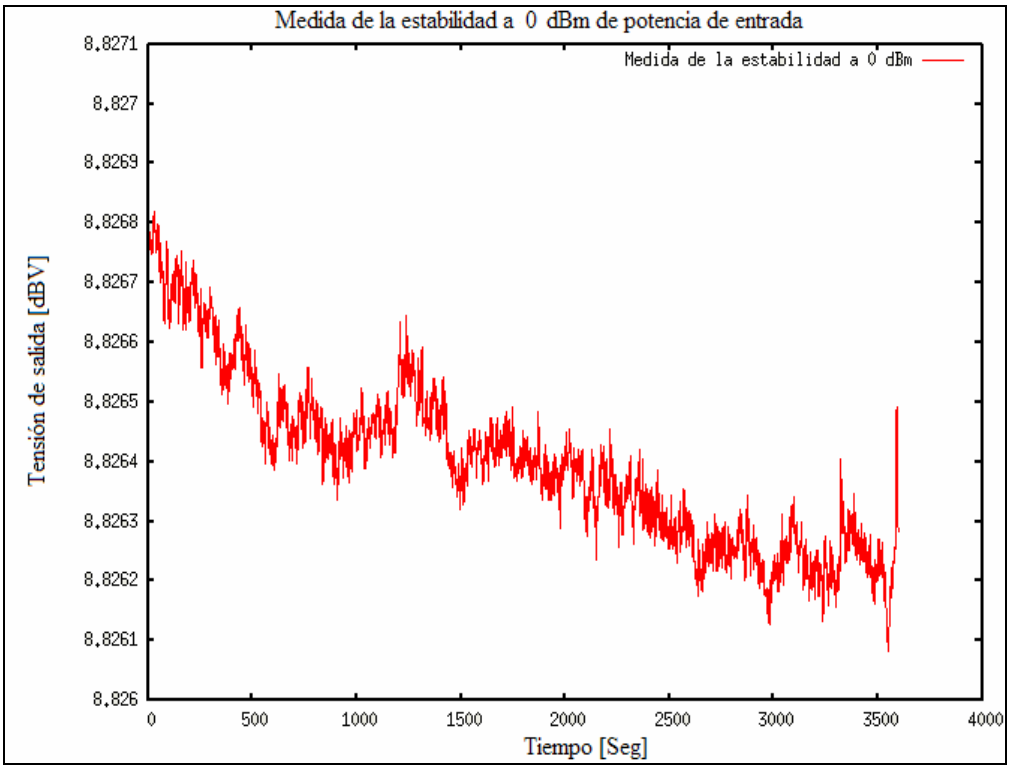

Figura 17.- Medida de la estabilidad durante una hora a la potencia de entrada de 0dBm

Variación pico a pico obtenida [dBV]
0.00071

4.2.2 Generador de ruido blanco gaussiano

4.2.2.1 Potencia de entrada –37dBm

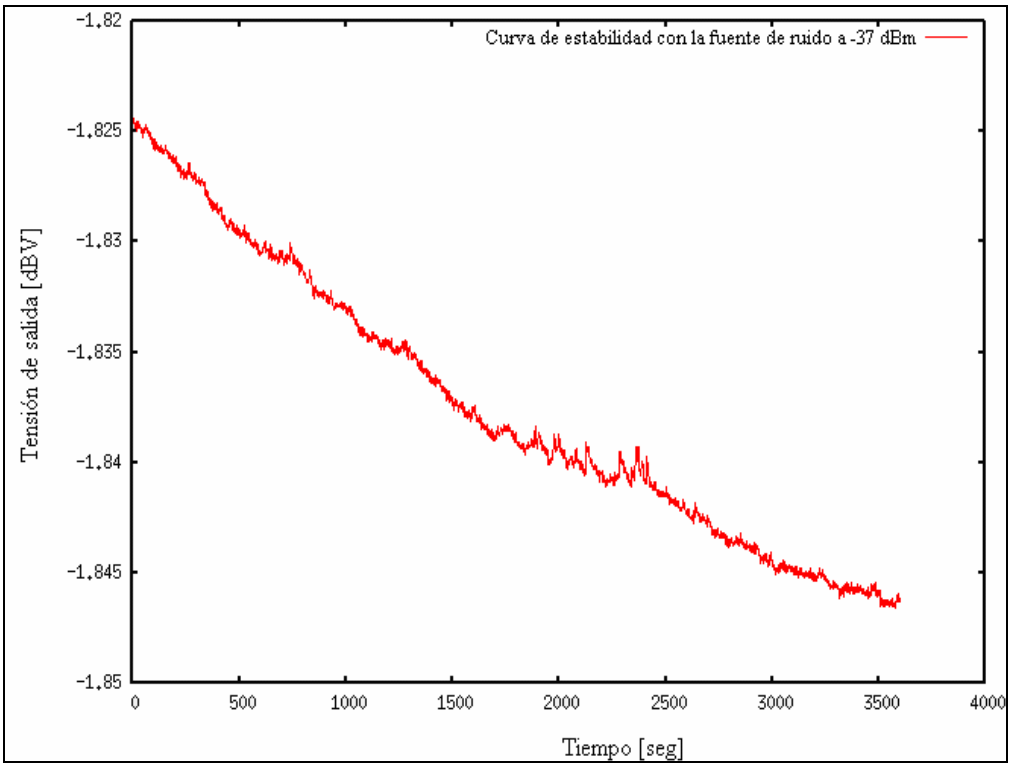

Figura 18.- Medida de la estabilidad durante una hora a la potencia de entrada de –37dBm

Variación pico a pico obtenida [dBV]
0.021

4.2.2.2 Potencia de entrada –17dBm

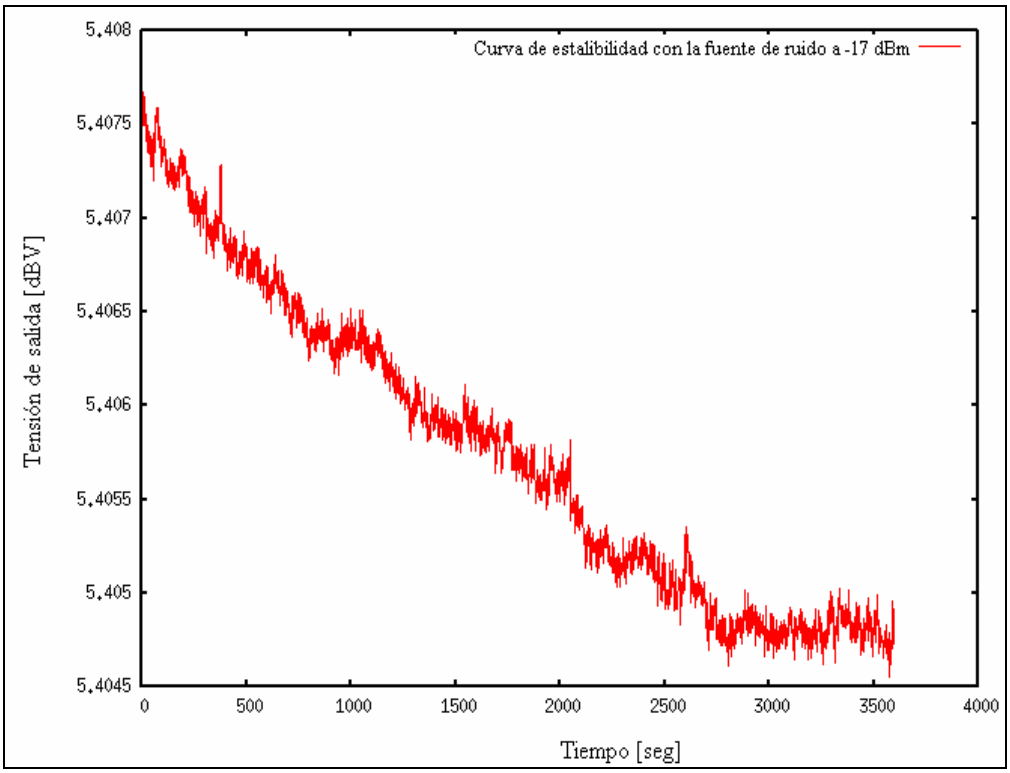

Figura 19.- Medida de la estabilidad durante una hora a la potencia de entrada de –17dBm

Variación pico a pico obtenida [dBV]
0.003

4.2.2.3 Potencia de entrada -2dBm

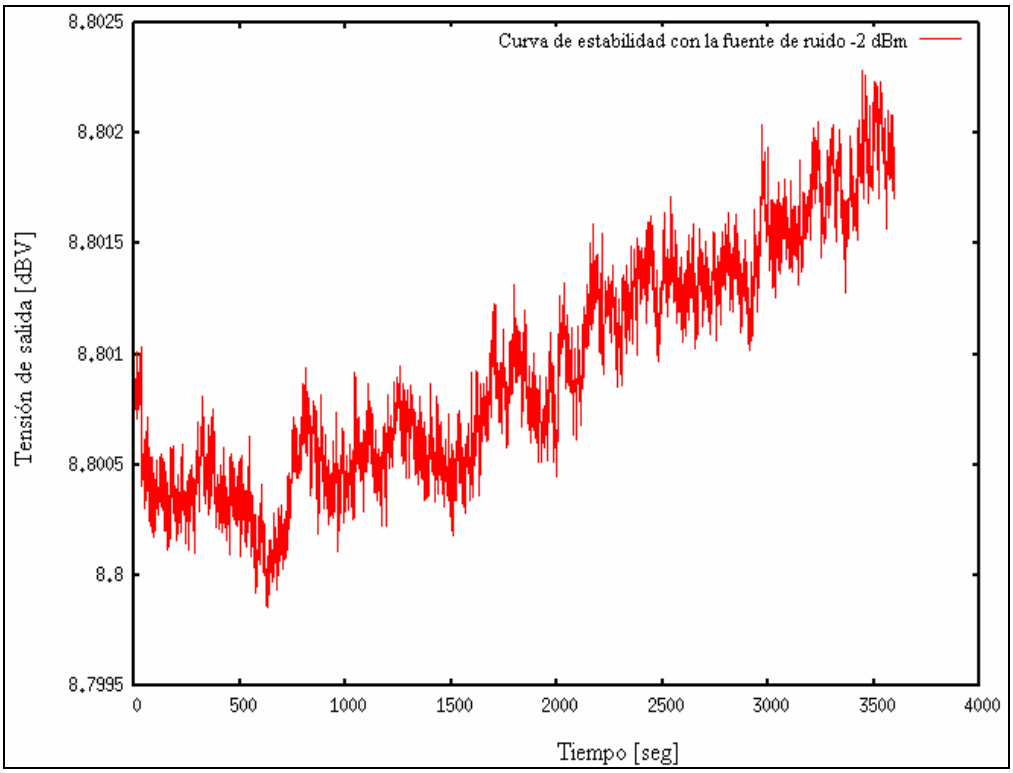

Figura 20.- Medida de la estabilidad durante una hora a la potencia de entrada de –2dBm

Variación pico a pico obtenida [dBV]
0.0022

V. Referencias bibliográficas

[1] José A. López Pérez, D. Cordobés, R. Bolaño: *"Comunicación con dispositivos GPIB a través de Ethernet"*. Informe Técnico OAN 2006-1. Enero 2006.

[2] B. Kernighan, D. Ritchie "El lenguaje de programación C. Segunda Edición". Ed Prentice Hall

Apéndice I: Placas PCB

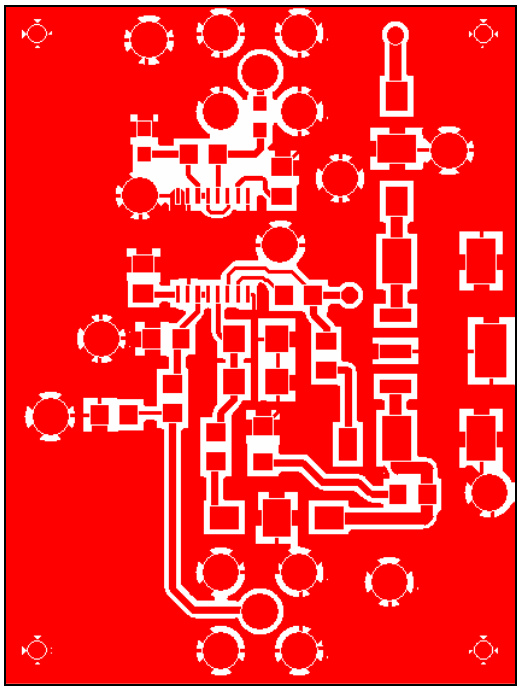

Fig A1.1.- Cara superior de la placa prototipo del detector de potencia

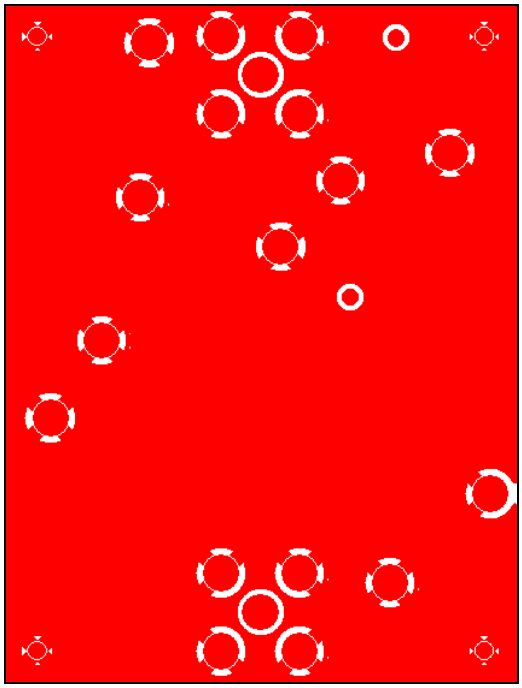

Fig A1.2.- Cara inferior de la placa prototipo del detector de potencia

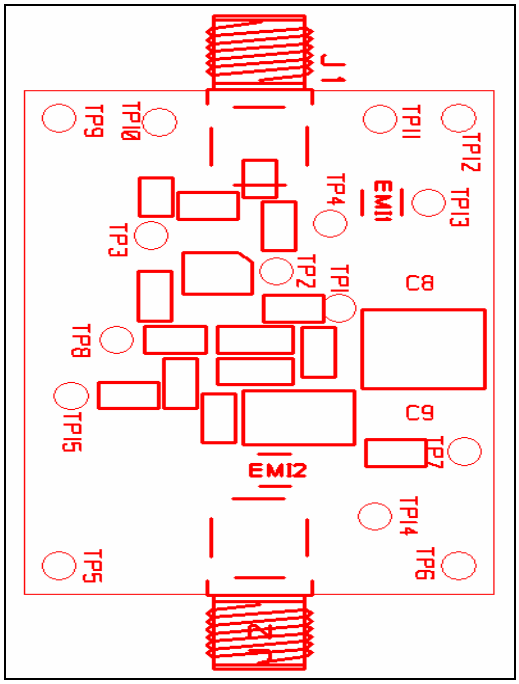

Fig A1.3.- Componentes de la placa de evaluación del detector de potencia

Item	Quantity	Reference	Part	
1	2	C1,C2	1nF	
2	1	C3	1000pF	
3	2	C6,C4	0.1uF	
4	1	C5	100pF	
5	1	C7	1uF	
6	2	C9,C8	6.8uF	
7	2	EMI1,EMI2	EMIFIL	
8	1	J1	RF_input	
9	1	J2	DC_output	
10	1	R1	100R	
11	1	R2	4K7	
12	2	R3,R7	1K0	
13	1	R4	100K	
14	1	R5	390	
15	1	R6	3R3	
16	1	R8	10K	
17	1	TP1	VTGT	
18	11	TP2,TP3,TP4,TP5,TP6,TP7, TP8,TP9,TP10,TP11,TP12		VIA
19	1	U1	AD8362	
20	1	U2	LT1521CST-5	

Fig A1.4.- Descripción de los componentes de la placa prototipo del detector de potencia

Apéndice II: Software

1) Caracterización del detector de potencia

```
/*********************************************************************
* File: set_ifr_2042_pot.c
*********************************************************************/
#include "tcpsocketclient.h"
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<ni488.h>
#include <unistd.h>


void GpibError(char *msg);        /* Error function declaration             */

int Device = 0;                   /* Device unit descriptor                 */
int BoardIndex = 0;               /* Interface Index (GPIB0=0,GPIB1=1,etc.) */

int main(int argc, char *argv[]) {
   int   PrimaryAddress = 1;      /* Primary address of the device          */
   int   SecondaryAddress = 0;    /* Secondary address of the device        */
   char msg[120];
   double frec_in,pot_in,tiempo_inicio,tiempo_aux;
   char unit_frec_in[4],unit_pot_in[4];

   int aux,aux2,cont_aux=0,tam_registro;

   double tope=0,tope_positivo=0,tope_negativo=0,
   sumat_pot=0,sumat_pot2=0,sumat_tension=0;
   double sumat_cruzado=0,coef_a,coef_b,coef_correl,
   num_corr=0,denom_corr1=0,denom_corr2=0;

   double *valor_pot,*valor_fin_tension;

   const char delimit[]=",\r\n";
   char query[128];
   double frec,tension_float,acumul_tension;

   char *tension;
   char *tiempo;

   double tiempo_float[100],t_integ,valor_recta,residuo;
```

```
    char *token;
    FILE *fich;

    int barrido_ini=atoi(argv[3]),
    barrido_fin=atoi(argv[4]),barrido_paso=atoi(argv[5]);


    if( argc != 7 )    {
        fprintf(stderr,"------------------------\n" );
        fprintf(stderr,"ERROR\n" );
        fprintf(stderr,"\nProgram usage: set_ifr frequency frequency_units RF_level
        RF_level_units RF_status barr_ini barr_fin barr_paso power_units\n" );
        fprintf(stderr,"EXAMPLE: ./set_ifr2042_pot 10 KHZ -75 10 5 DBM\n" );
        fprintf(stderr,"------------------------\n" );
        exit( 1 );
    }

    sprintf(msg, "fich_out_");
    strcat(msg, argv[1]);
    strcat(msg, argv[2]);
    strcat(msg, ".dat");

    fich=fopen(msg,"w");

    sprintf(msg, "\n#Fichero de salida de la medida de ");
    strcat(msg, argv[1]);
    strcat(msg, " ");
    strcat(msg, argv[2]);

    fprintf(fich,msg);
    fprintf(fich,"\n#--------------------------------\n");

    t_integ=1.0; //1 seg. de tiempo de integracion


/*****************************************************************************
 * Inicializacion del Keithley 2701
 *****************************************************************************/

    Tcpsocketclient so1 = Tcpsocketclient("193.146.252.64", 1394);
    so1.Connect();
    so1.Write("TRAC:CLE\r\n", strlen("TRAC:CLE\r\n"), 0);
    sleep(2);
    so1.Write("*RST\r\n", strlen("*RST\r\n"), 0);
    sleep(2);
    so1.Write("FUNC 'VOLT'\r\n", strlen("FUNC 'VOLT'\r\n"), 0);
    sleep(2);
```

```
   tam_registro=0;

   for(aux=barrido_ini;aux!=barrido_fin+barrido_paso;aux=aux+barrido_paso){
        tam_registro++;
   }
   valor_fin_tension=(double *)malloc(tam_registro*sizeof(double));
   valor_pot=(double *)malloc(tam_registro*sizeof(double));
   frec_in=atof(argv[1]);
   sprintf(unit_frec_in, argv[2]);
   sprintf(unit_pot_in, argv[6]);

/*****************************************************************************
 * Inicializacion del conversor GPIB-ENET
 *****************************************************************************/

   Device = ibdev(               /* Create a unit descriptor handle         */
         BoardIndex,             /* Board Index (GPIB0 = 0, GPIB1 = 1, ...) */
         PrimaryAddress,         /* Device primary address                  */
         SecondaryAddress,       /* Device secondary address                */
         T10s,                   /* Timeout setting (T10s = 10 seconds)     */
         1,                      /* Assert EOI line at end of write         */
         0);                     /* EOS termination mode                    */
   if (ibsta & ERR) {            /* Check for GPIB Error                    */
      GpibError("ibdev Error");
   }
   ibclr(Device);                /* Clear the device                        */
   if (ibsta & ERR) {
      GpibError("ibclr Error");
   }
/*****************************************************************************
 * Inicializacion del IFR 2042
 *****************************************************************************/
   ibwrt(Device, (void *)"*RST", strlen("*RST"));
   if (ibsta & ERR) {
       GpibError("ibwrt Error");
   }
   ibwrt(Device, (void *)"IMODE NOISE1", strlen("IMODE NOISE1"));
   if (ibsta & ERR) {
      GpibError("ibwrt Error");
   }
   ibwrt(Device, (void *)"MOD:OFF", strlen("MOD:OFF"));
   if (ibsta & ERR) {
      GpibError("ibwrt Error");
   }
   sprintf(msg, "CFRQ:VALUE %f",frec_in);
   strcat(msg,unit_frec_in);
   ibwrt(Device, msg, strlen(msg));
```

```
if (ibsta & ERR) {
    GpibError("ibwrt Error");
}
for(aux=barrido_ini;aux!=barrido_fin+barrido_paso;aux=aux+barrido_paso){
    valor_pot[cont_aux]=aux;
    sprintf(msg, "RFLV:VALUE %d",aux);
    strcat(msg,unit_pot_in);
    strcat(msg,";ON");
    ibwrt(Device, msg, strlen(msg));
    printf("--------------------------------\n");
    printf("Potencia a la salida del IFR: %s\n", msg);
    if (ibsta & ERR) {
           GpibError("ibwrt Error");
    }
    sleep(1);
    acumul_tension=0.0;

    //Lectura del Keithley. Se promedian las medidas durante t_integ

    for(aux2=0;aux2<100000;aux2++){
           for (int i = 0; i < 128; i++)
                  query[i] = '\0';

           so1.Write("READ?\r\n", strlen("READ?\r\n"), 0);
           usleep(100000);
           so1.Read(query, 127, 0);
           printf("Respuesta del Keithley: %s", query);
           token = strtok(query, delimit);
           tension = (char *)malloc(strlen(token)+1);
           strcpy(tension, token);
           printf("CHAR Tension: %s\n",tension);
           tension_float = atof(tension);
           acumul_tension=acumul_tension+tension_float;
           token = strtok(NULL, delimit);
           tiempo = (char *)malloc(strlen(token)+1);
           strcpy(tiempo, token);
           printf("CHAR Tiempo: %s \n",tiempo);
           tiempo_float[aux2] = atof(tiempo);
           if(aux2==0)    tiempo_inicio=tiempo_float[aux2];
           else{
                  tiempo_aux=tiempo_float[aux2]-tiempo_inicio;
                  if(tiempo_aux>t_integ){
                         acumul_tension=acumul_tension-tension_float; //Quito el
                         ultimo
                         valor_fin_tension[cont_aux]=acumul_tension/aux2;  //  Se
                         hace una media de los valores
                         break;
                  }
```

```
                }
                printf("FLOAT         Tension         y         tiempo         %15.13f
                %f\n",tension_float,tiempo_float[aux2]);
        }
        sleep(2);
        printf("Valores obtenidos de potencia [dBm] y tension [VDC], valores tomados
        (AUX2) %f %f %d",valor_pot[cont_aux],valor_fin_tension[cont_aux],aux2);
        printf("--------------------------------\n");
        fprintf(fich,"%15.13f
        %15.13f\n",valor_pot[cont_aux],valor_fin_tension[cont_aux]);
        cont_aux++;
}
        // Ahora en valor_pot y valor_fin_tension tengo los valores de potencia y t
        ension del barrido
        // Estimo recta regresion y coef. correl.
for(aux2=0;aux2<tam_registro;aux2++){
        sumat_pot=sumat_pot+valor_pot[aux2];
        sumat_pot2=sumat_pot2+(valor_pot[aux2]*valor_pot[aux2]);
        sumat_tension=sumat_tension+valor_fin_tension[aux2];
        sumat_cruzado=sumat_cruzado+(valor_fin_tension[aux2]*valor_pot[aux2]);
}
coef_b=((tam_registro*sumat_cruzado)-(
sumat_pot*sumat_tension))/((tam_registro*sumat_pot2)-(sumat_pot*sumat_pot));
coef_a=(sumat_tension/tam_registro)-coef_b*(sumat_pot/tam_registro);

for(aux2=0;aux2<tam_registro;aux2++){
        num_corr=num_corr+(valor_pot[aux2]-
        sumat_pot/tam_registro)*(valor_fin_tension[aux2]-sumat_tension/tam_registro);
        denom_corr1=denom_corr1+(valor_pot[aux2]-
        (sumat_pot/tam_registro))*(valor_pot[aux2]-(sumat_pot/tam_registro));
        denom_corr2=denom_corr2+(valor_fin_tension[aux2]-
        (sumat_tension/tam_registro))*(valor_fin_tension[aux2]-
        (sumat_tension/tam_registro));
}
coef_correl=num_corr/(sqrt(denom_corr1)*sqrt(denom_corr2));
printf("Coeficientes a,b,corr : %f %f %f\n",coef_a,coef_b,coef_correl);
aux2=0;
for(aux=barrido_ini;aux!=barrido_fin+barrido_paso;aux=aux+barrido_paso){
        valor_recta=coef_a+coef_b*aux;
        residuo=valor_fin_tension[aux2]-valor_recta;
        if(residuo>0)
                if(residuo>tope_positivo)
                        tope_positivo=residuo;
        else if(residuo<0)
                if(residuo<tope_negativo)
                        tope_negativo=residuo;
        printf("Residuo %f\n",residuo);
        aux2++;
```

```
    }
        printf("\nMaximo residuo [Voltios] positivo, negativo, RMS %f %f
        %f\n",tope_positivo,tope_negativo,tope_positivo-tope_negativo);
        fprintf(fich,"#---------------------------------\n");
        fprintf(fich,"#Coeficientes de regresión lineal:\n");
        fprintf(fich,"#Coeficiente a:              %8.6f\n",coef_a);
        fprintf(fich,"#Coeficiente b:              %8.6f\n",coef_b);
        fprintf(fich,"#Coeficiente de correlacion: %8.6f\n",coef_correl);
        fprintf(fich,"#RMS  [Voltios]:                        %8.6f\n",tope_positivo-
        tope_negativo);
        fprintf(fich,"#---------------------------------\n");

/*****************************************************************************
 * Cierre del conversor y del programa
 *****************************************************************************/

   ibonl(Device, 0);              /* Take the device offline                */
   if (ibsta & ERR) {
      GpibError("ibonl Error");
   }
   free(tiempo);
   free(tension);
   free(valor_fin_tension);
   free(valor_pot);
   fclose(fich);
   return 0;
}
/*****************************************************************************
 *                      Function GPIBERROR
 * This function will notify you that a NI-488 function failed by
 * printing an error message.  The status variable IBSTA will also be
 * printed in hexadecimal along with the mnemonic meaning of the bit
 * position. The status variable IBERR will be printed in decimal
 * along with the mnemonic meaning of the decimal value.  The status
 * variable IBCNTL will be printed in decimal.
 *
 * The NI-488 function IBONL is called to disable the hardware and
 * software.
 *
 * The EXIT function will terminate this program.
 *****************************************************************************/
void GpibError(char *msg) {
    printf ("%s\n", msg);
    printf ("ibsta = &H%x  <", ibsta);
    if (ibsta & ERR )  printf (" ERR");
    if (ibsta & TIMO)  printf (" TIMO");
    if (ibsta & END )  printf (" END");
    if (ibsta & SRQI)  printf (" SRQI");
```

```
        if (ibsta & RQS )  printf (" RQS");
        if (ibsta & CMPL)  printf (" CMPL");
        if (ibsta & LOK )  printf (" LOK");
        if (ibsta & REM )  printf (" REM");
        if (ibsta & CIC )  printf (" CIC");
        if (ibsta & ATN )  printf (" ATN");
        if (ibsta & TACS)  printf (" TACS");
        if (ibsta & LACS)  printf (" LACS");
        if (ibsta & DTAS)  printf (" DTAS");
        if (ibsta & DCAS)  printf (" DCAS");
        printf (" >\n");
        printf ("iberr = %d", iberr);
        if (iberr == EDVR) printf (" EDVR <DOS Error>\n");
        if (iberr == ECIC) printf (" ECIC <Not Controller-In-Charge>\n");
        if (iberr == ENOL) printf (" ENOL <No Listener>\n");
        if (iberr == EADR) printf (" EADR <Address error>\n");
        if (iberr == EARG) printf (" EARG <Invalid argument>\n");
        if (iberr == ESAC) printf (" ESAC <Not System Controller>\n");
        if (iberr == EABO) printf (" EABO <Operation aborted>\n");
        if (iberr == ENEB) printf (" ENEB <No GPIB board>\n");
        if (iberr == EOIP) printf (" EOIP <Async I/O in progress>\n");
        if (iberr == ECAP) printf (" ECAP <No capability>\n");
        if (iberr == EFSO) printf (" EFSO <File system error>\n");
        if (iberr == EBUS) printf (" EBUS <Command error>\n");
        if (iberr == ESTB) printf (" ESTB <Status byte lost>\n");
        if (iberr == ESRQ) printf (" ESRQ <SRQ stuck on>\n");
        if (iberr == ETAB) printf (" ETAB <Table Overflow>\n");

        printf ("ibcntl = %ld\n", ibcntl);
        printf ("\n");
        /* Call ibonl to take the device and interface offline */
        ibonl (Device,0);
        exit(1);
}
```

Apéndice III: Curvas de conversión a distintas VTGT

1) VTGT = 1 Volt

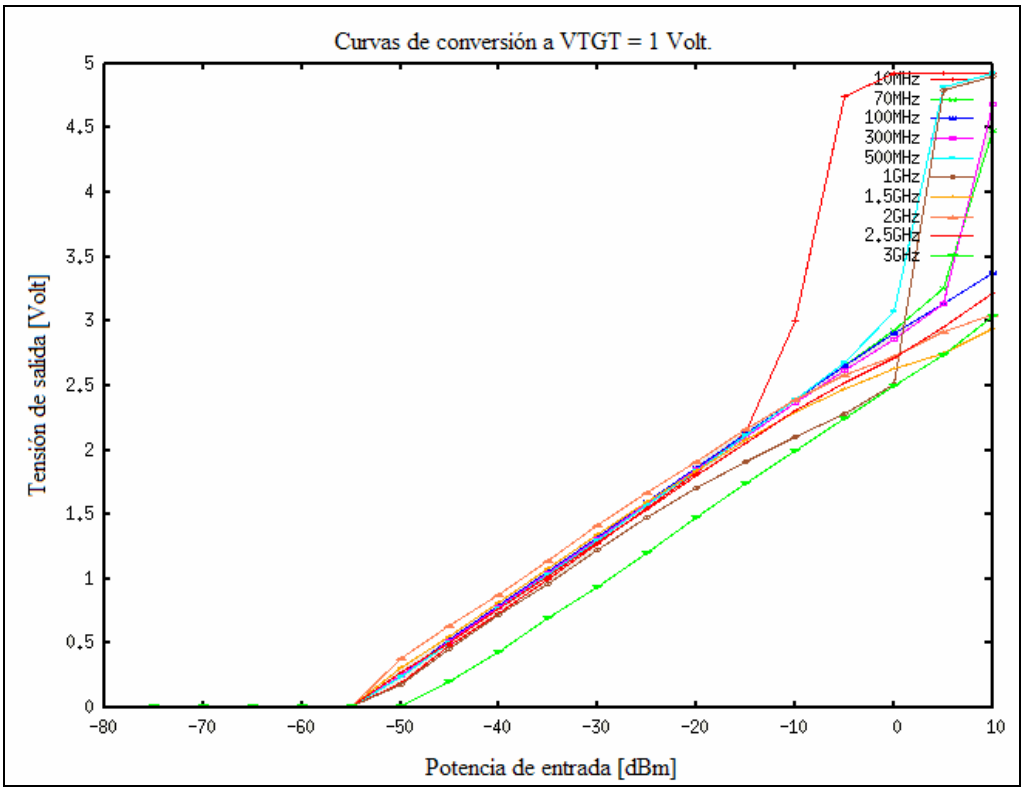

Figura A3.1.- Curvas de conversión a VTGT = 1 Volt

2) VTGT = 2 Volt

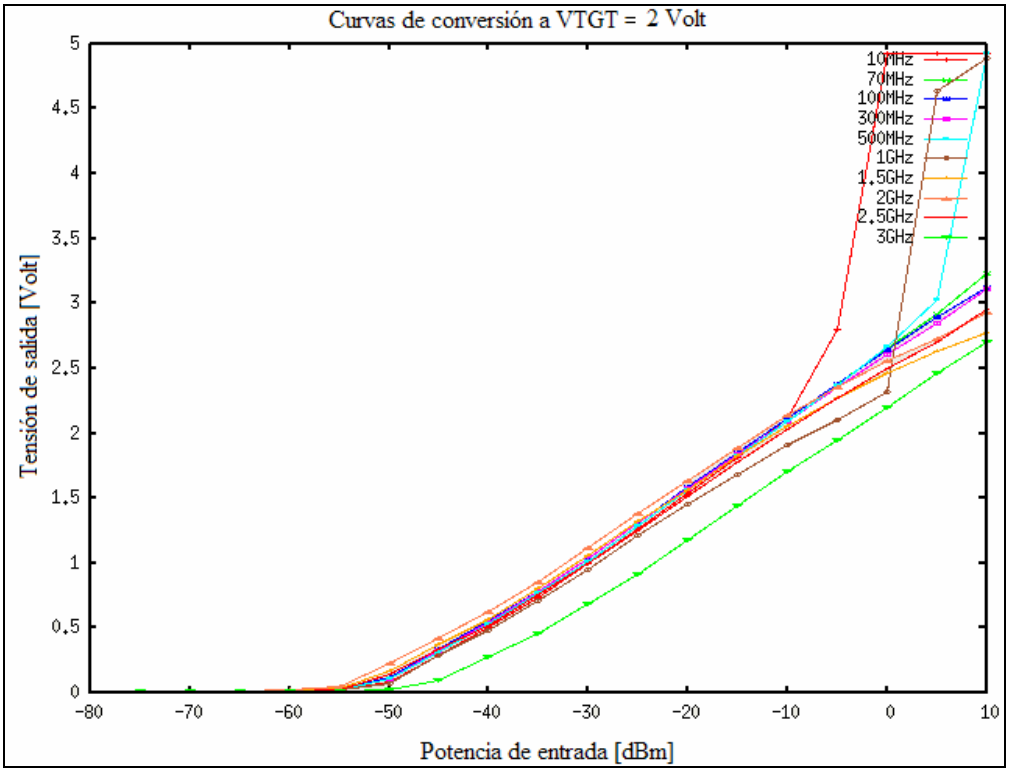

Figura A3.2.- Curvas de conversión a VTGT = 2 Volt

3) VTGT = 2.5 Volt

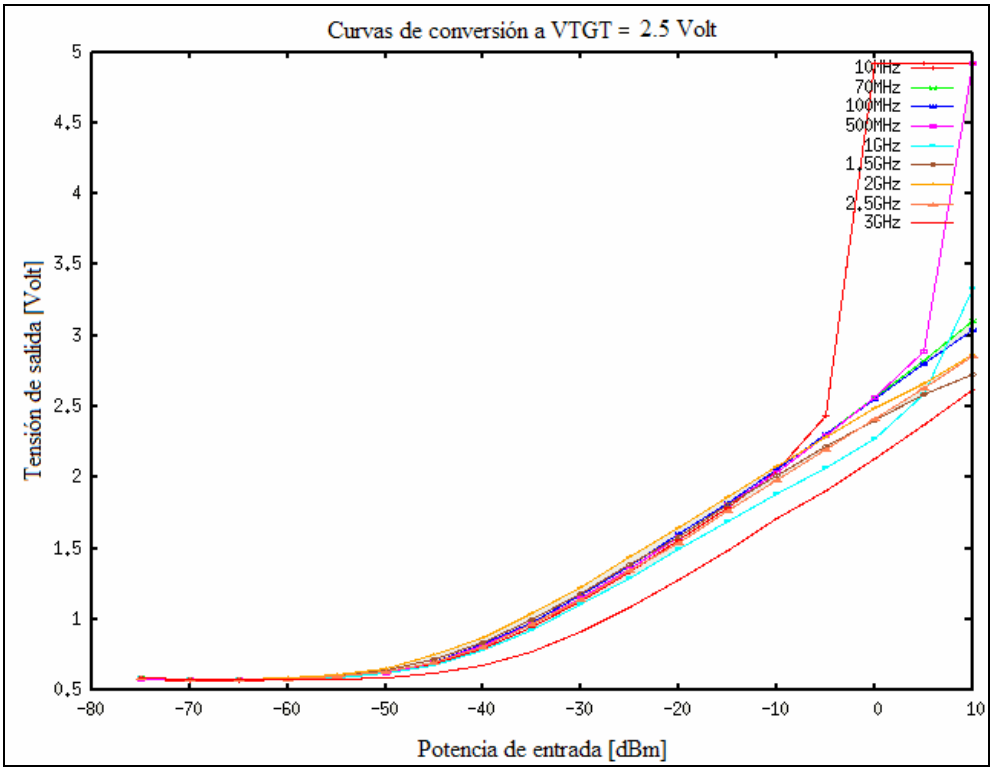

Figura A3.3.- Curvas de conversión a VTGT = 2.5 Volt

4) VTGT = 3 Volt

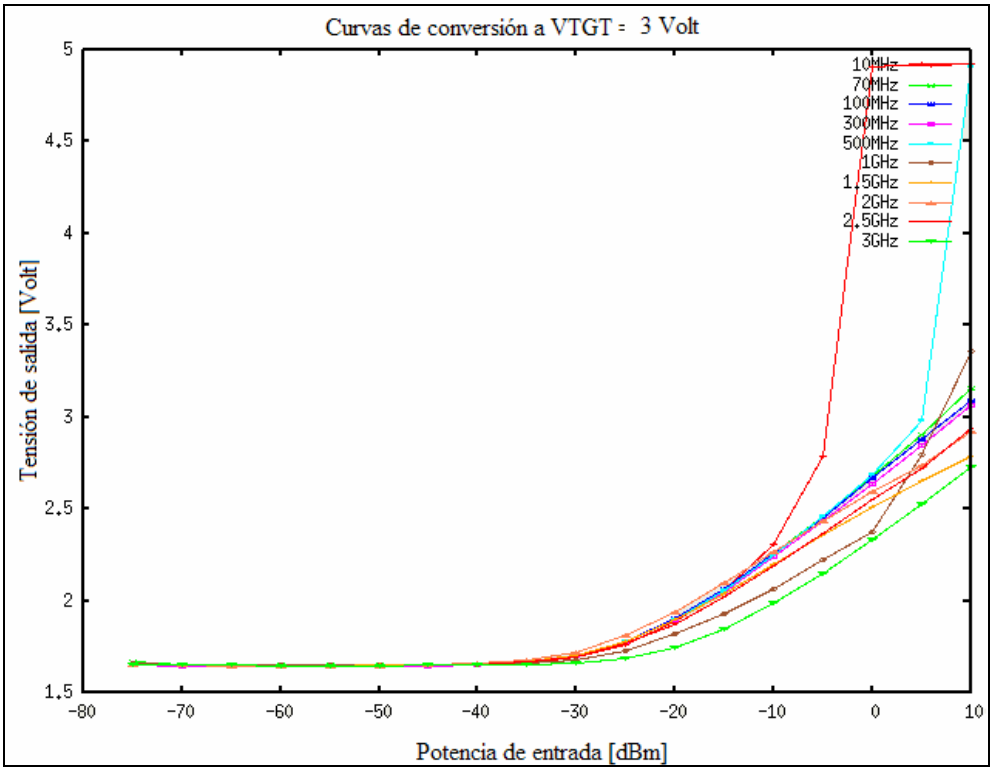

Figura A3.4.- Curvas de conversión a VTGT = 3 Volt

Apéndice IV: Hojas de características