Observatorio de Yebes
Apartado 148, 19080 Guadalajara, SPAIN

DDS AD9850: TESTS AND CHARACTERIZATION

Alberto Río Martínez, Adrián Alonso, Pablo García, Carlos Almendros

IT-CDT 2021-12

Observatorio de Yebes

Apdo. 148 19080 Guadalajara

SPAIN

Phone: +34 949 29 03 11

Fax: +34 949 29 00 63

REVISION HISTORY

Version	Date	Author	Updates
1.0	24/06/2021	Alberto Río Martínez	First Version

INDEX

1. Introduction

HC-SR08 is a signal Generator based on complete DDS Synthesizer AD9850, 125MHz clock frequency (cause of the cristal of the module) and frequency range 0-40MHz.

AD9850 is an integrated device that uses advanced DDS technology coupled with a D/A converter and comparator, to form a programmable frequency synthesizer and clock generator function. There are 40 bits of data: 32 bits for frequency information, 2 bits for control, 1 bit for Power-Down control, and 5 bits for phase information. This induces on a frequency resolution of 0.029Hz for CLKIN=125MHz; and a phase resolution of 11.25 degrees.

Pin No.	Mnemonic	Function
4–1, 28–25	D0–D7	8-Bit Data Input. This is the 8-bit data port for iteratively loading the 32-bit frequency and 8-bit phase/ control word. D7 = MSB; D0 = LSB. D7 (Pin 25) also serves as the input pin for the 40-bit serial data word.
5, 24	DGND	Digital Ground. These are the ground return leads for the digital circuitry.
6, 23	DVDD	Supply Voltage Leads for digital circuitry.
7	W_CLK	Word Load Clock. This clock is used to load the parallel or serial frequency/phase/control words.
8	FQ_UD	Frequency Update. On the rising edge of this clock, the DDS will update to the frequency (or phase) loaded in the data input register, it then resets the pointer to Word 0.
9	CLKIN	Reference Clock Input. This may be a continuous CMOS-level pulse train or sine input biased at 1/2 V supply. The rising edge of this clock initiates operation.
10, 19	AGND	Analog Ground. These leads are the ground return for the analog circuitry (DAC and comparator).
11, 18	AVDD	Supply Voltage for the analog circuitry (DAC and comparator).
12	R_{SET}	This is the DAC's external R_{SET} connection. This resistor value sets the DAC full-scale output current. For normal applications (F_S I_{OUT} = 10 mA), the value for R_{SET} is 3.9 kΩ connected to ground. The R_{SET}/I_{OUT} relationship is: I_{OUT} = 32 (1.248 V/R_{SET}).
13	QOUTB	Output Complement. This is the comparator's complement output.
14	QOUT	Output True. This is the comparator's true output.
15	VINN	Inverting Voltage Input. This is the comparator's negative input.
16	VINP	Noninverting Voltage Input. This is the comparator's positive input.
17	DACBL (NC)	DAC Baseline. This is the DAC baseline voltage reference; this lead is internally bypassed and should normally be considered a "no connect" for optimum performance.
20	IOUTB	The Complementary Analog Output of the DAC.
21	IOUT	Analog Current Output of the DAC.
22	RESET	Reset. This is the master reset function; when set high it clears all registers (except the input register) and the DAC output will go to Cosine 0 after additional clock cycles—see Figure 19.

Figure 1: Pins from AD9850.

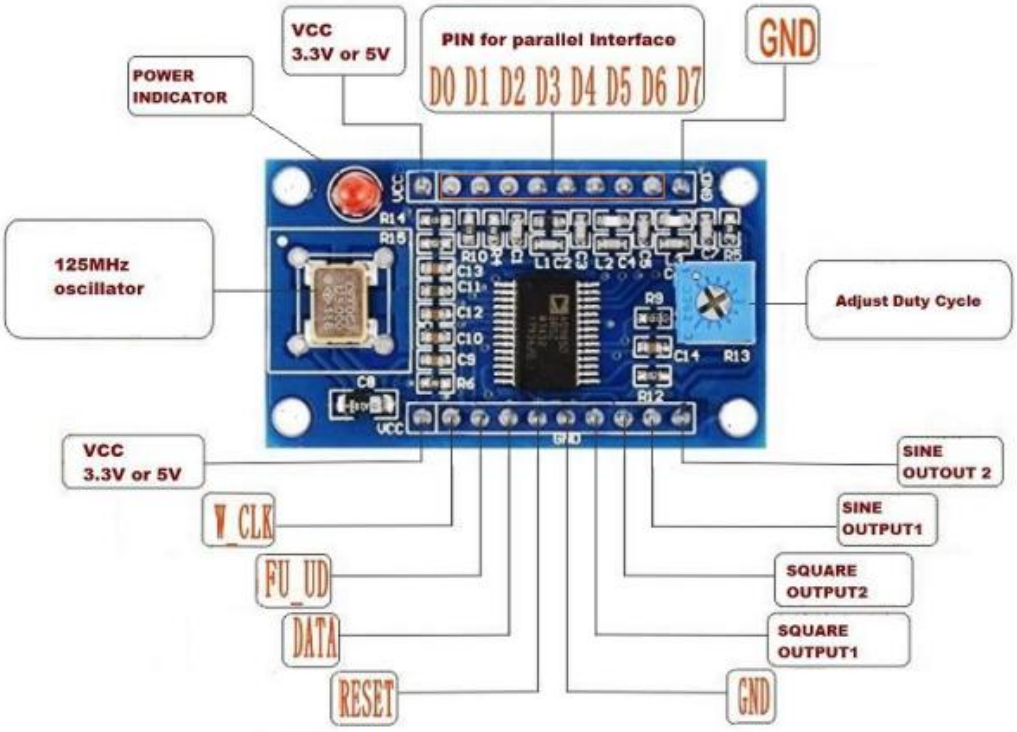

Figure 2: HC-SR08 diagram.

AD9850 can load the data content on serial or parallel load. Parallel load consists of five iterative loads of an 8-bit control word (byte). Serial load Serial load consists of 40 loads of an 1-bit control Word by a single pin (pin DATA on HC-SR08). After these two loadings FQ_UD goes up one cycle to reset pointer to first Word.

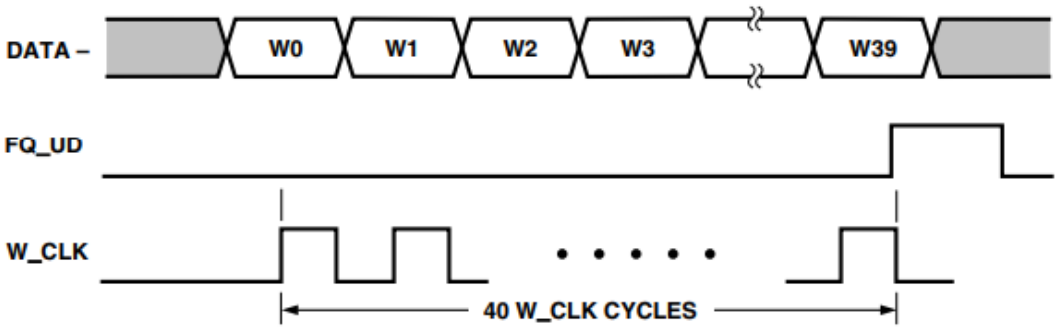

Figure 3: Serial load configuration.

W0	Freq-b0 (LSB)	W14	Freq-b14	W28	Freq-b28
W1	Freq-b1	W15	Freq-b15	W29	Freq-b29
W2	Freq-b2	W16	Freq-b16	W30	Freq-b30
W3	Freq-b3	W17	Freq-b17	W31	Freq-b31 (MSB)
W4	Freq-b4	W18	Freq-b18	W32	Control
W5	Freq-b5	W19	Freq-b19	W33	Control
W6	Freq-b6	W20	Freq-b20	W34	Power-Down
W7	Freq-b7	W21	Freq-b21	W35	Phase-b0 (LSB)
W8	Freq-b8	W22	Freq-b22	W36	Phase-b1
W9	Freq-b9	W23	Freq-b23	W37	Phase-b2
W10	Freq-b10	W24	Freq-b24	W38	Phase-b3
W11	Freq-b11	W25	Freq-b25	W39	Phase-b4 (MSB)
W12	Freq-b12	W26	Freq-b26		
W13	Freq-b13	W27	Freq-b27		

Figure 4: 40-Bit Serial Load Word Function Assignment.

AD9850 has one analog output and a clock output. Our output is the analog, that is the output of the 10-bit DAC, so we obtain a stepped signal, which need a DC-Block to filter harmonics. The analog output frequency follow the folllowing $f_{OUT} = (\Delta\ Phase \times CLKIN)/2^{32}$ expression: that is also used on the control code.

2. Python code description

2.1. Functions definition

```
# Declaracion de pines GPIO
W_CLK = 15
FQ_UD = 16
DATA = 18
RESET = 22

# Declaracion de los pines como salidas
GPIO.setup(W_CLK, GPIO.OUT)
GPIO.setup(FQ_UD, GPIO.OUT)
GPIO.setup(DATA, GPIO.OUT)
GPIO.setup(RESET, GPIO.OUT)

# Inicializar todas las variables a cero
GPIO.output(W_CLK, False)
GPIO.output(FQ_UD, False)
GPIO.output(DATA, False)
GPIO.output(RESET, False)
```

Figure 5: Pin configuration as outputs, HC-SR08 port selection for the pins and initialization as low level of the pins.

```
# Funcion para mandar nivel alto a un pin
def nivelAlto(pin):
        GPIO.output(pin, True)
        GPIO.output(pin, True)
        GPIO.output(pin, False)
        return
```

Figure 6: Function "nivelAlto" that puts up level on a pin.

```
# Funcion para mandar un bit a data
def mandar_bit(data):
                GPIO.output(DATA, data & 0x01) #coge el
                nivelAlto(W_CLK) #pone W_CLK a nivel al
```

Figure 7: Function"mandar_bit" that loads the LSB bit on data and put up level on pin W_CLK (W_CLK should be at up level each time one bit is loaded in serial loading as is shown in Figure 1) with nivelAlto function.

```
# Funcion para enviar frecuencia y fase al AD9850 asumiendo el C
def enviarFrecuenciayFase(frecuencia, fase):
        frec=int(frecuencia*4294967296/125000000) #frecuencia qu
        fas=int(fase*32/360) #fase multiplicada por 3^5 de los 5
        for b in range (0,34): #mandamos 35 palabras de un bit c
                mandar_bit(frec) #mandamos un bit de frec
                frec=frec>>1 #desplazamos 1 posicion pra mandar
        mandar_bit(0x00) #dejamos de mandar informacion
        for a in range (0,4): #
                mandar_bit(fas)
                fas=fas>>1
        mandar_bit(0x00)
        nivelAlto(FQ_UD) #ponemos a nivel alto FQ despues de los
        return
```

Figure 8: Function "enviarFrecuenciayFase" that sends frequency and phase by the correct assigned words shown in Figure 2, in the case the clock is 125MHz. For sending frequency and phase is needed to multiply the variables by the inverse of the tuning resolution (tuning resolution is the maximum variable valor divided by 2 raised to number of bits of resolution, 32 in frequency and 5 in phase). After that, the 35 bits of frequency and control are sent by a 35 loop that uses mandar_bit function and do a right shift to send the next bit in the following iteration. The 5 bits that contains phase information are sent like the frequency ones. For ending, a bit of 0 value is sent for ending the operation and FQ_UD is set to up level to reset the word pointer.

2.2. Main

```
#main
def main():
        frecu = float(input("frecuencia en MHz=")) #introducimos frecuencia en MHz por la terminal para su posterior uso
        frecuencia=frecu*1000000 #multiplicamos la variable introducida en MHz por un millon para obtener variable en Hz
        fase=float(input("Fase en grados="))
        nivelAlto(RESET) #inicializamosn los pines
        nivelAlto(W_CLK)
        nivelAlto(FQ_UD)
        enviarFrecuenciayFase(frecuencia,fase) #enviamos frecuencia al AD9850

#Inicializa main
if __name__=="__main__":
        main()
```

Figure 9: Main that allows to introduce frequency (MHz) and phase (degrees) variables by terminal. After that RESET, W_CLK and FQ_UD are initialized and the introduced frequency and phase variables are sent to AD9850bby enviarFrecuenciayFase function.

3. Experimental results

For obtaining the results we only need to put the wanted frequency and phase on terminal like in Figure 10, and it works as explained previously. Figures 11 and 13 show the output of AD9850 for instructions 2 and 3 from Figure 10, respectively. Cause of the D/A converter, the output signals are stepped and, by FFT from oscilloscope, is possible to see the harmonics. It can be appreciated that there are harmonic components after the fundamental harmonic, so, if they are filtered as in Figure 12 (where are filtered with a low pass from oscilloscope with a cut frequency of 20MHz) is obtained a clean output signal with the wanted frequency and phase.

```
pi@raspberrypi:~/software/RPi_RFSigGen-master $ python3 rpi_rfsiggenerator.py
frecuencia en MHz=5.4
Fase en grados=180
pi@raspberrypi:~/software/RPi_RFSigGen-master $ python3 rpi_rfsiggenerator.py
frecuencia en MHz=5
Fase en grados=180
pi@raspberrypi:~/software/RPi_RFSigGen-master $ python3 rpi_rfsiggenerator.py
frecuencia en MHz=10
Fase en grados=90
```

Figure 10: Code execution with different values of frequency and phase.

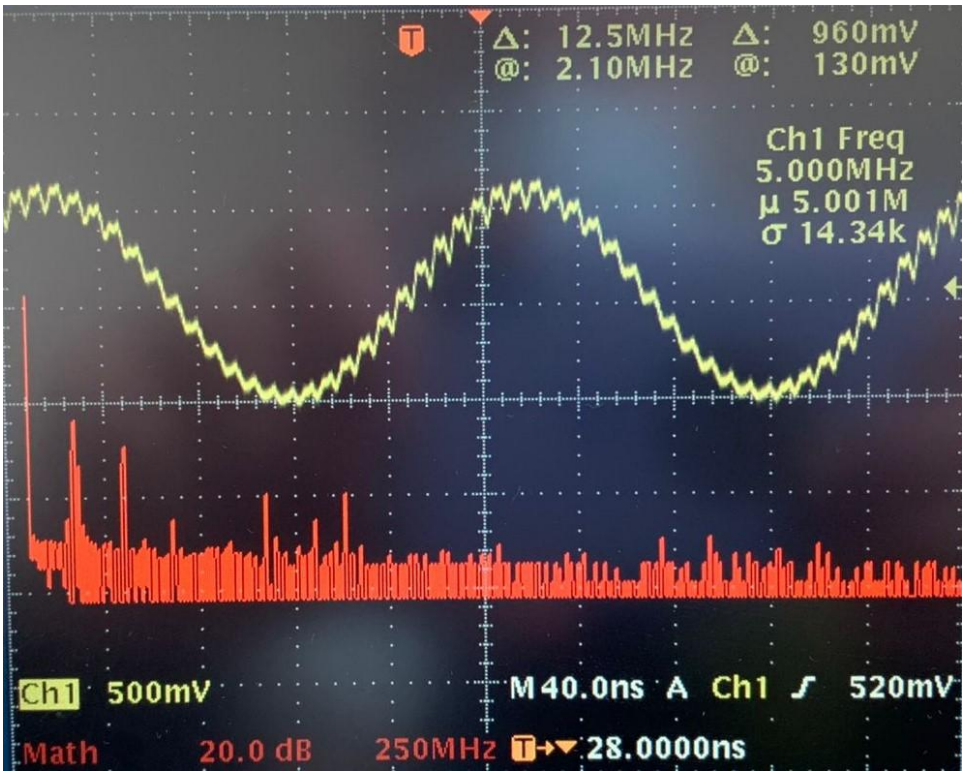

Figure 11: Analog output and FFT analysis for 5MHz frequency and 180 degrees phase.

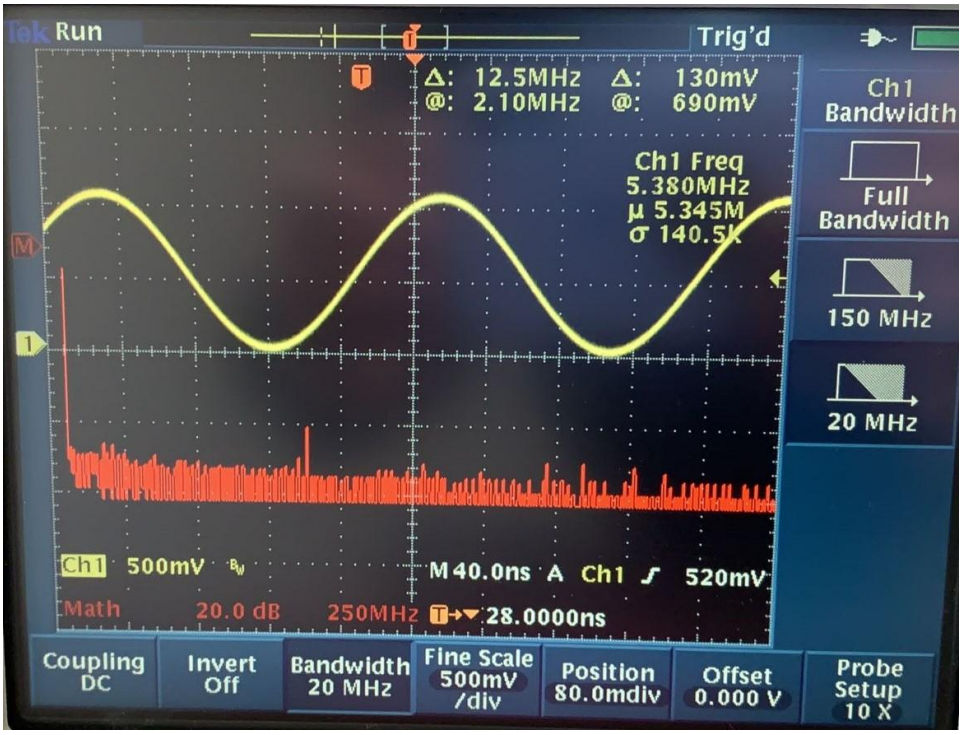

Figure 12: Analog output and FFT analysis for 5.4MHz frequency and 180 degrees phase.

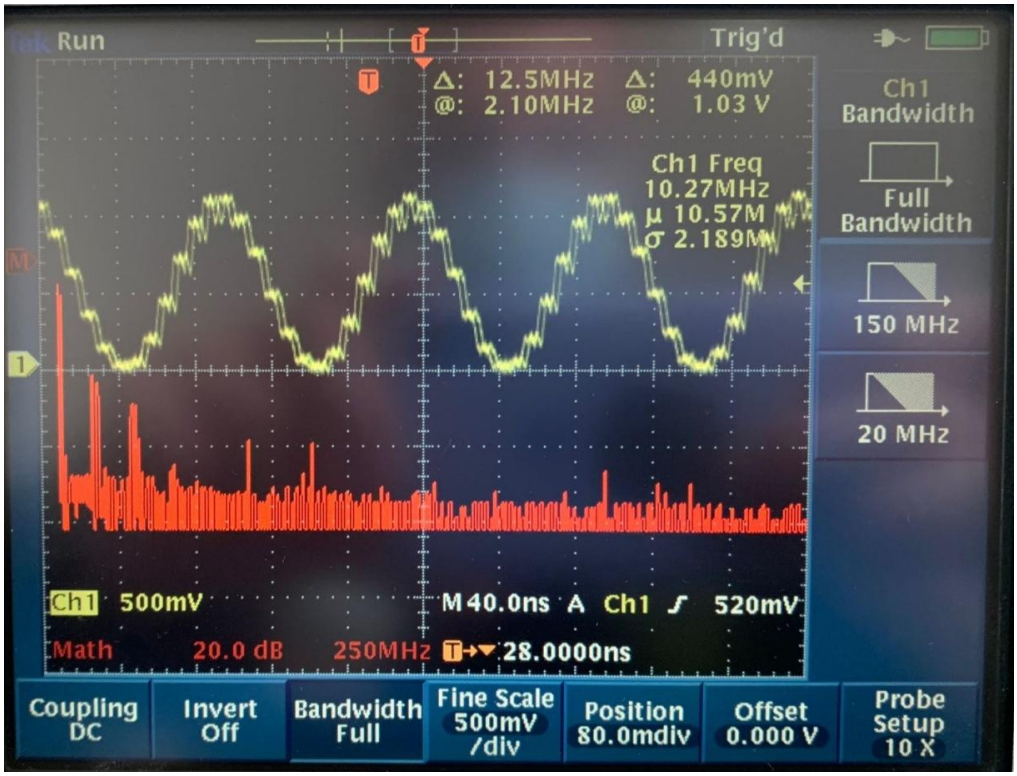

Figure 13: Analog output and FFT analysis for 10MHz frequency and 90 degrees phase.

The main goal of the system is to introduce very stable phase changes into Phase Cal Ground Unit, so twice clocks should be synchronized. For obtaining the synchronization is needed to change 125MHz HC-SR08 clock by the external clock from Phase Cal. External clock is introduced by a BNC wire connected to HC-SR08. In Figure 14 is shown the experimental setup, and in Figure 15 the signal introduced with 10dBm by an analog signal generator. As is shown in Figure 16, the experimetal results are the same as when they have been made with the default clock from HC-SR08 board.

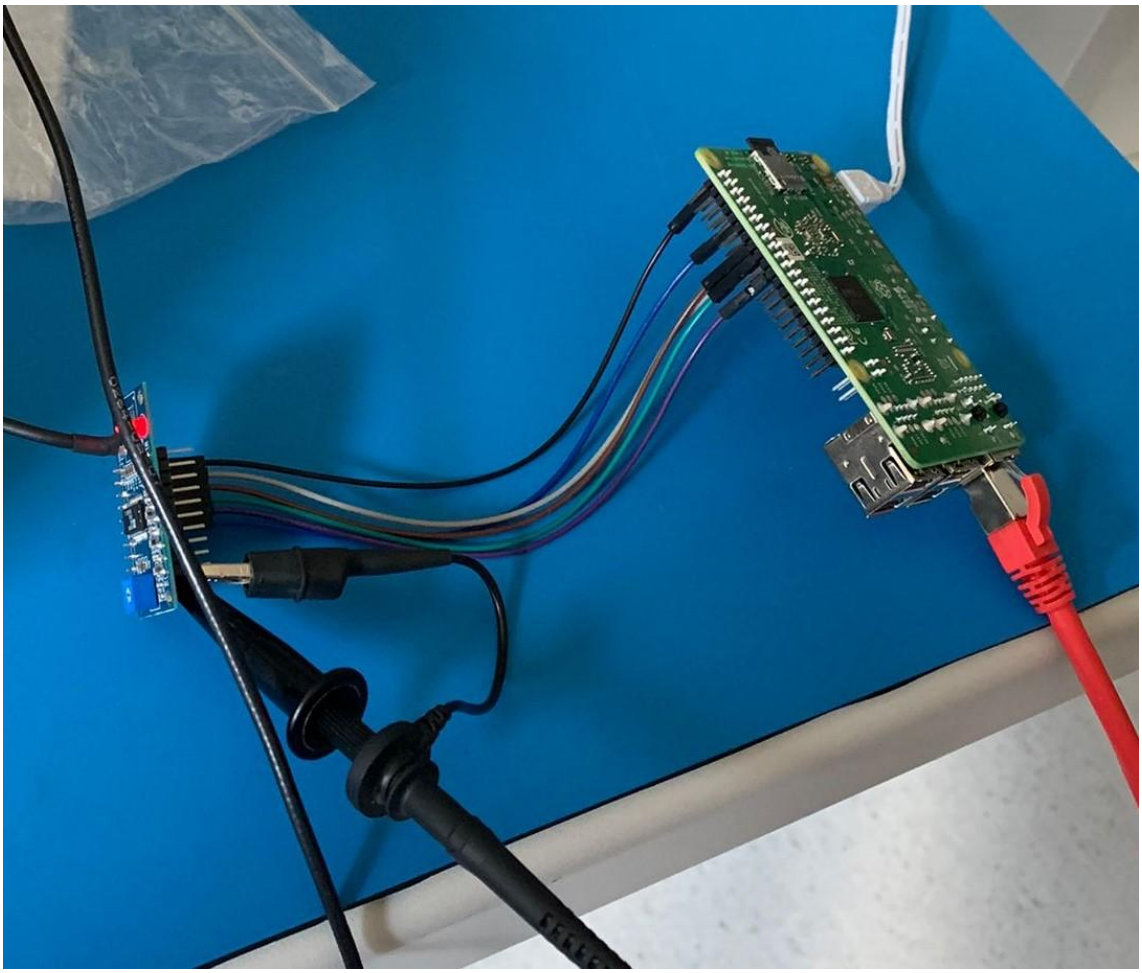

Figure 14: Experimental setup.

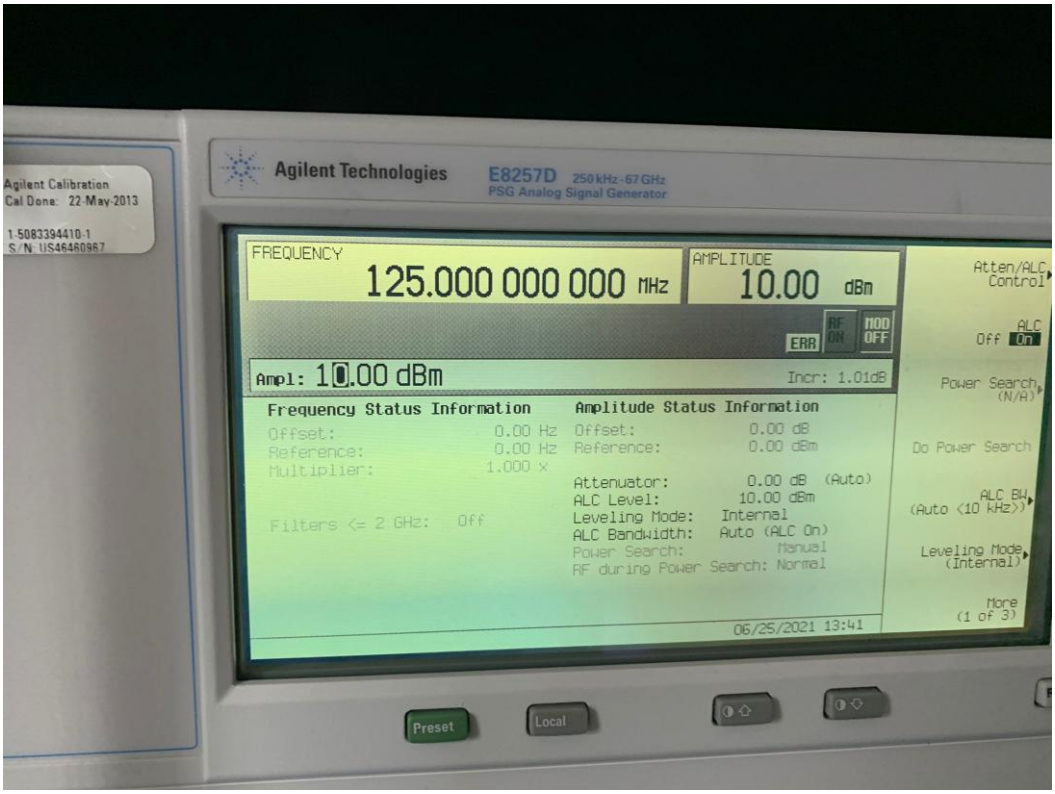

Figure 15: Analog Signal Generator that introduces clock signal.

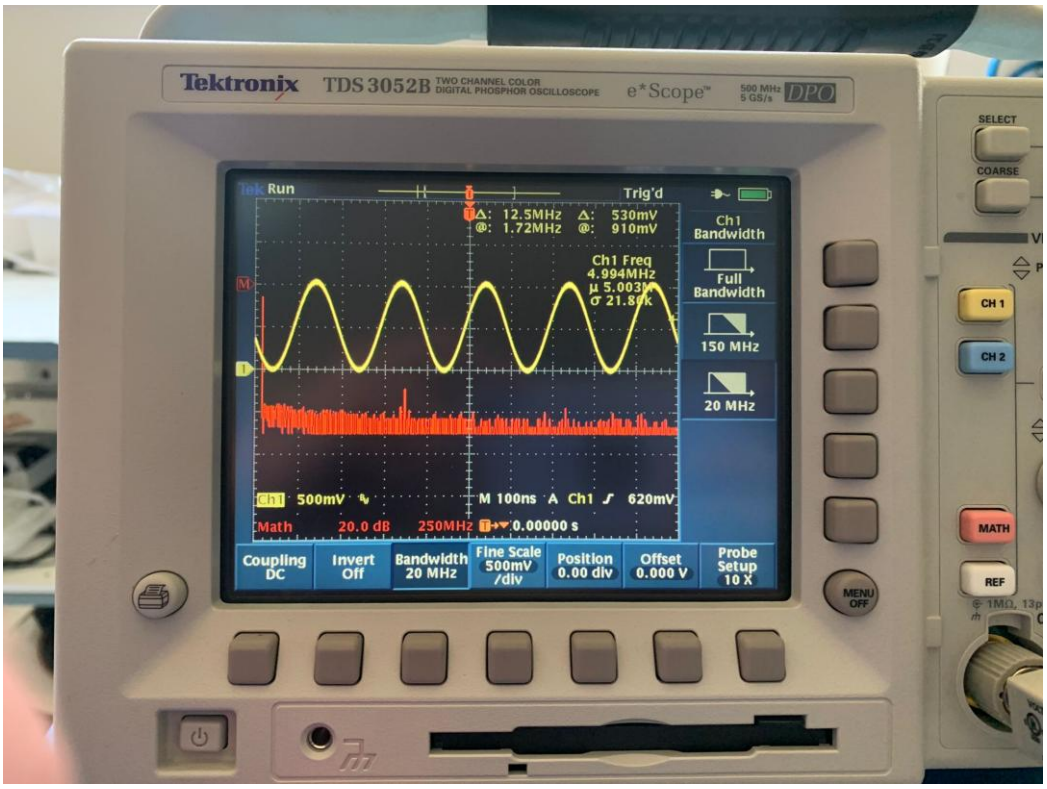

Figure 16: Results.

For ending, Phase cal clock is 10MHz and DDS clock should have a value near to 100MHz. For clock synchronization is used a 10MHz to 100MHz very stable conversion board (CRYSTEK CMOD225-10-100) between DDS and module clock (in this way we have twice connected and sinchronized by the same clock), whose noise figure is studied below; in Figure 20 we can see the phase variations of the board between 9:44am and 10:04am of July 27, 2021. There are low variations and typical phase deviation (obtained by writing values from measurement on a matlab .txt and plottting them with it) are very low too, so we can deduce that the noise figure of the board is fine and it can be use don the system.

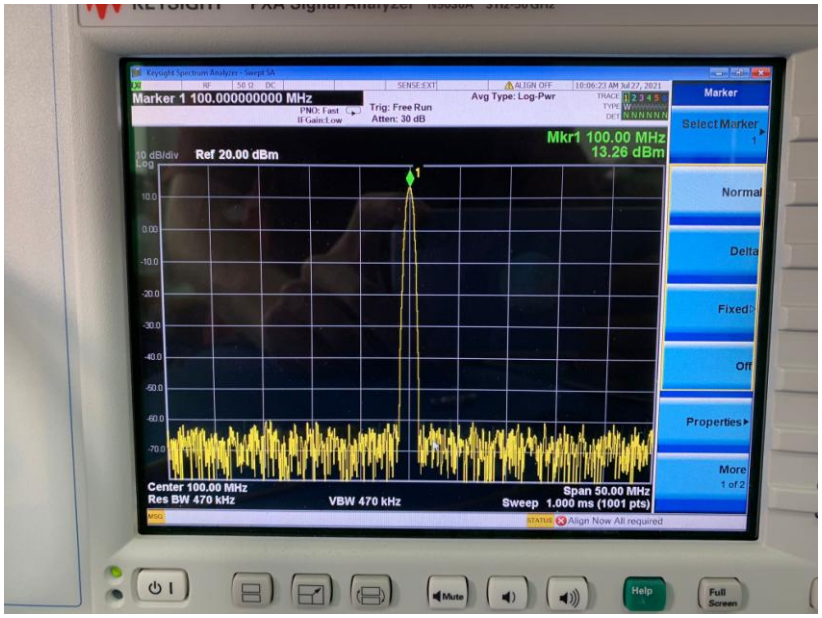

Figure 17: Frequency figure from spectrum analyzer.

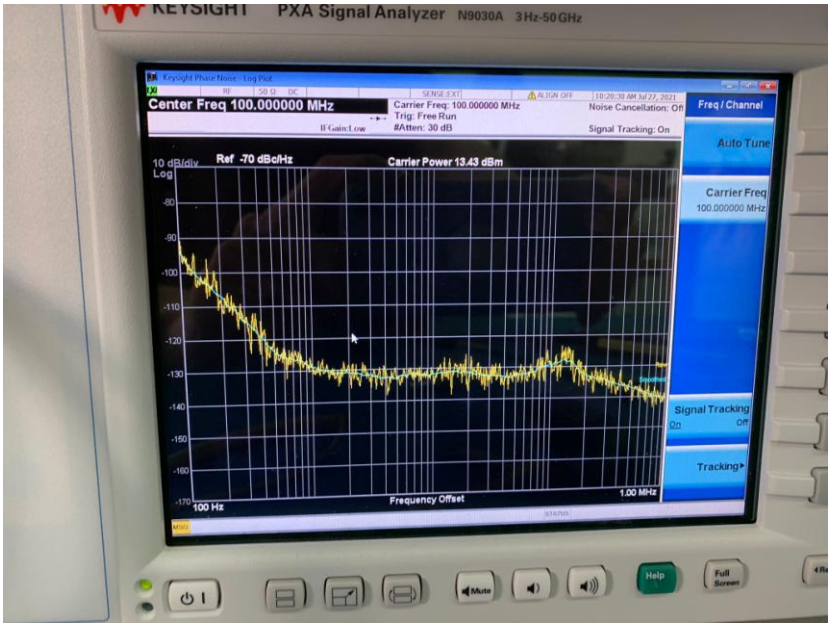

Figure 18: Phase figure from spectrum analyzer.

```
>> std(phase)

ans =

    0.1683
```

Figure 19: Typical phase deviation.

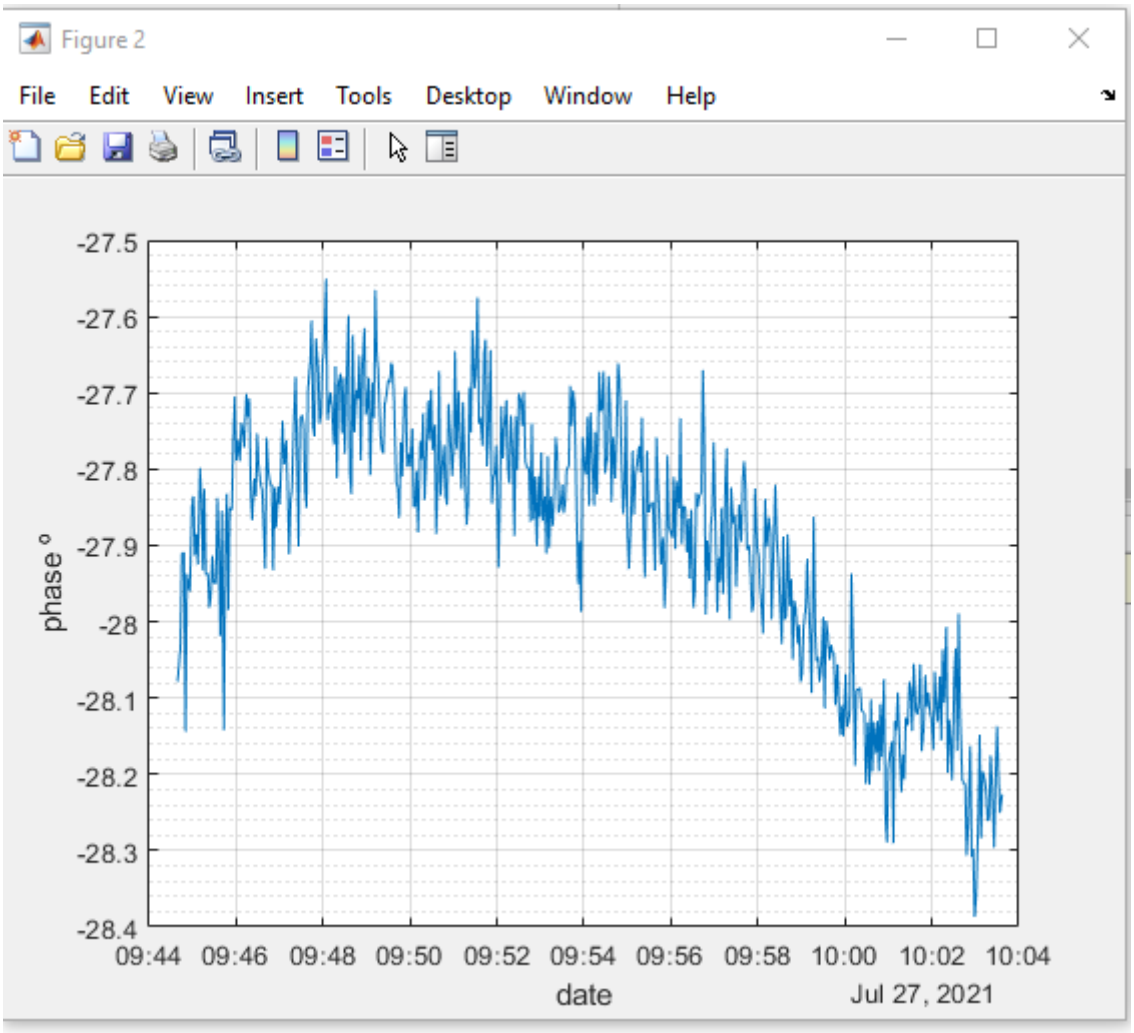

Figure 20: Phase Noise Figure.

4. References

- CMOS, 125 MHz Complete DDS Synthesizer AD9850 datasheet - ANALOG DEVICES

- HC-SR08: ADS9850 Signal Generator Module datasheet

5. Appendix: SSH connection to Raspberry Pi 3 Model B+

The terminology to make ssh connection to Raspberry pi is: ssh –X –l pi (IP address). To end connection is important to execute: sudo poweroff.