MKC-REM: Úkol č. 3

Filip Paul

Měření intenzity el. pole pomocí spektrálního analyzátoru

27.03.2022

Zadání:

Určete velikost intenzity elektrického pole v místě fázového středu antény na následujících kmitočtech 2400, 2450 a 2500 MHz? Vlastní měření bylo provedeno EMC spektrálním analyzátorem Rohde&Schwarz ESPI7 a na těchto kmitočtech byly naměřeny následující údaje: -130 dBm, 8 dBμV a -92 dBm. Hodnoty anténního faktoru použité antény typu Bi-Log pro výše uvedené kmitočty jsou: 20, 14 a 12 dB/m. Dále byl zjištěn vlastní šum použitého spektrálního analyzátoru a dosahoval následujících hodnot: -23, -20, -18 dBμV.

Vypracování:

Podle datasheetu je možné na spekrálním anaylzátoru Rohde&Schwarz ESPI7 nastavit vstupní impedanci na 50 nebo 75 Ω. Pro výpočty jsem uvažoval vstupní impedanci právě 50 Ω. Vzhledem k oblíbenosti převodů jednotek jsem pro veškeré převody vytvořil python script, pomocí kterého jsou všechny převody prováděny. V následující tabulce jsou již převedené hodnoty. Z druhého sloupce tabulky pro f = 2400 MHz je patrné, že naměřený signál je srovnatelný s vlastním šumem přístroje. Takže hodnoty v tomto sloupci asi nelze brát úplně vážně. V přiloženém python scriptu, který si můžete zobrazit i v mém GITHUB repozitáři zde, je formou komentářů v kódu stručný popis postupu.

f	**2400 MHz**	**2450 MHZ**	**2500 MHz**
Změřeno	0.07 μV	2.51 μV	5.62 μV
ŠUM	0.07 μV	0.10 μV	0.04 μV
SNR	-0.01 dB	28.00 dB	42.99 dB
SIGNAL	-188.51 dBm	-99.34 dBm	-92.06 dBm
INTENZITA	-168.51 dBV/m	-85.34 dBV/m	-80.06 dBV/m

calc.py

```
from UnitConverter import UnitConverter

f = [2400, 2450, 2500] #MHz
spekt_meas = ["-130 dBm" , "8 dBuV", "-92 dBm"]
spekt_noise = ["-23 dBuV" , "-20 dBuV" , "-28 dBuV"]
impedance = 50
AF = ["20 dBm", "14 dBm", "12 dBm"]

E = []
spekt_meas_converted = [] #dBm
spekt_noise_converted = [] #dBm
SNR = []
SIGNAL = []

for i in range(len(spekt_meas)):
     #convert noise and measured signals to uV
     conv_val_meas = UnitConverter(spekt_meas[i],impedance)
     spekt_meas_converted.append(f"{conv_val_meas.uV:.2f} uV")

     conv_val_noise = UnitConverter(spekt_noise[i],impedance)
     spekt_noise_converted.append(f"{conv_val_noise.uV:.2f} uV")

     #SNR = measured[dBm] - noise[dBm] -> dB
     SNR.append(f"{conv_val_meas.dBm - conv_val_noise.dBm :.2f} dB")

     #signal only =  measured[uV] - noise[uV] -> dBm
     conv_val_signal = UnitConverter(f"{conv_val_meas.uV - conv_val_noise.uV} uV",impedance)
     SIGNAL.append(f"{conv_val_signal.dBm:.2f} dBm")

     #E[dBV/m] = AF[dBm] + V[dBV]
     E.append(f"{int(AF[i][:2]) + conv_val_signal.dBm :.2f} dBV/m")

#Save outputs into file for automatic TEX table generator
with open("output2.txt","w") as f:
     f.writelines(f"{[x for x in spekt_meas_converted]}\n")
     f.writelines(f"{[x for x in spekt_noise_converted]}\n")
     f.writelines(f"{[x for x in SNR]}\n")
     f.writelines(f"{[x for x in SIGNAL]}\n")
     f.writelines(f"{[x for x in E]}\n")
```

UnitConverter.py

```
import numpy as np
class UnitConverter:
     #EX: UnitConverter("-130 dBm" ,"dBuV", 50) -> will output 106.98 dBuV
     def __init__(self,convert_from :str,input_impedance :int):
          self.conv_from = convert_from
          self.input_impedance = input_impedance

          self.Power = "not initialize"

          self.kW = "not initialize"
          self.W = "not initialize"
          self.mW = "not initialize"
          self.uW = "not initialize"

          self.dB = "not initialize"
          self.dBm = "not initialize"
          self.dBu = "not initialize"

          self.kV = "not initialize"
          self.V = "not initialize"
          self.mV = "not initialize"
          self.uV = "not initialize"

          self.dBV = "not initialize"
          self.dBmV = "not initialize"
          self.dBuV = "not initialize"

          self.recognize_input_type()
          self.convertAll()

     def convertAll(self):
```

```
32            #POWER TO POWER:
33            self.kW = self.Power*1e-3
34            self.W = self.Power
35            self.mW = self.Power*1e3
36            self.uW = self.Power*1e6
37
38            #POWER TO POWER DB
39            self.dB = 10*np.log10(self.Power)
40            self.dBm = 10*np.log10(self.Power/1e-3)
41            self.dBu = 10*np.log10(self.Power/1e-6)
42
43            #POWER TO VOLTAGE
44            self.V = np.sqrt(self.Power * self.input_impedance)
45            self.kV = self.V*1e-3
46            self.mV = self.V*1e3
47            self.uV = self.V*1e6
48
49            #POWER TO VOLTAGE DB
50            self.dBV = 20*np.log10(self.V)
51            self.dBmV = 20*np.log10(self.V/1e-3)
52            self.dBuV = 20*np.log10(self.V/1e-6)
53
54      def printAll(self):
55            print(f"CONVERTING: {self.conv_from:} at ZIN: {self.input_impedance} to:")
56            print(f"POWER:\n {self.kW: .3f}kW {self.W: .3f}W {self.mW: .3f}mW {self.uW: .3f}uW")
57            print(f"POWER in dB:\n {self.dB: .3f}dB {self.dBm: .3f}dBm {self.dBu: .3f}dBu")
58            print(f"VOLTAGE:\n {self.kV: .3f}kV {self.V: .3f}V {self.mV: .3f}mV {self.uV: .3f}uV")
59            print(f"VOLTAGE in dB:\n {self.dBV: .3f}dBV {self.dBmV: .3f}dBmV {self.dBuV: .3f}dBuV")
60
61      def recognize_input_type(self):
62            #recognizes input type and convert all possible inputs to power in Wats
63            #all folowing conversions will be related to P in WATS
64            val,unit = self.conv_from.split(" ")
65            val = float(val)
66            #POWER in dB
67            if unit == "dB":
68                  self.Power = 10**(val/10)
69            elif unit == "dBm":
70                  self.Power = 10**(val/10)*1e-3
71            elif unit == "dBu":
72                  self.Power = 10**(val/10)*1e-6
73
74            #POWER
75
76            elif unit == "kW":
77                  self.Power = val*1e3
78            elif unit == "W":
79                  self.Power = val
80            elif unit == "mW":
81                  self.Power = val*1e-3
82            elif unit == "uW":
83                  self.Power = val*1e-6
84
85            #VOLTAGE IN DB
86            elif unit == "dBV":
87                  V = 10**(val/20)
88                  self.Power = V**2/self.input_impedance
89            elif unit == "dBmV":
90                  V = 10**(val/20)*1e-3
91                  self.Power = V**2/self.input_impedance
92            elif unit == "dBuV":
93                  V = 10**(val/20)*1e-6
94                  self.Power = V**2/self.input_impedance
95
96            #VOLTAGE
97            elif unit == "kV":
98                  self.Power = (val*1e3)**2/self.input_impedance
99            elif unit == "V":
100                 self.Power = val**2/self.input_impedance
101           elif unit == "mV":
102                 self.Power = (val*1e-3)**2/self.input_impedance
103           elif unit == "uV":
104                 self.Power = (val*1e-6)**2/self.input_impedance
105
106           else:
107                 print("unknown input parameter: supported units: dB dBm dBu kW W mW uW dBV dBmV
```

```
dBuV kV V mV uV")
```