Stefan Nellen

Bitácora

24/11/20

Midiendo la constante de Planck

7:37

Reunión con prof. y ayudante.

-> Discutimos experimento

-> Tenemos material para hacerlo en casa:

- LEDs
- 2 mm 12 metros (Usotronic, Radioshack)
- Resistencias de carbón
- 9V (12xAA)

10:10

-> En el taller

Encontramos artículo brasileño muy completo.

-> Sugiere usar buffer o convertidor de corriente a voltaje.

Usamos C-V, (ping mut. time amp.)

-> Fuente?

LM 7805

Fuente opamp

2 pilas AA en serie, 2 veces.

12:22

↳ Está demasiado bajo.

Fuente opamp → compra vieja

± 12 V.

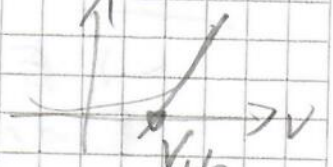
Discutimos sensor de huellas.

→ Presentación profesor y música.

Teoría:

Queremos $I = I_0 e^{-\frac{eV_{th}}{n k_B T}} \left(e^{\frac{eV}{n k_B T}} - 1\right)$

Shockley



S:

Ajustar recta superponiendo que se prende con $V > V_{th}$.

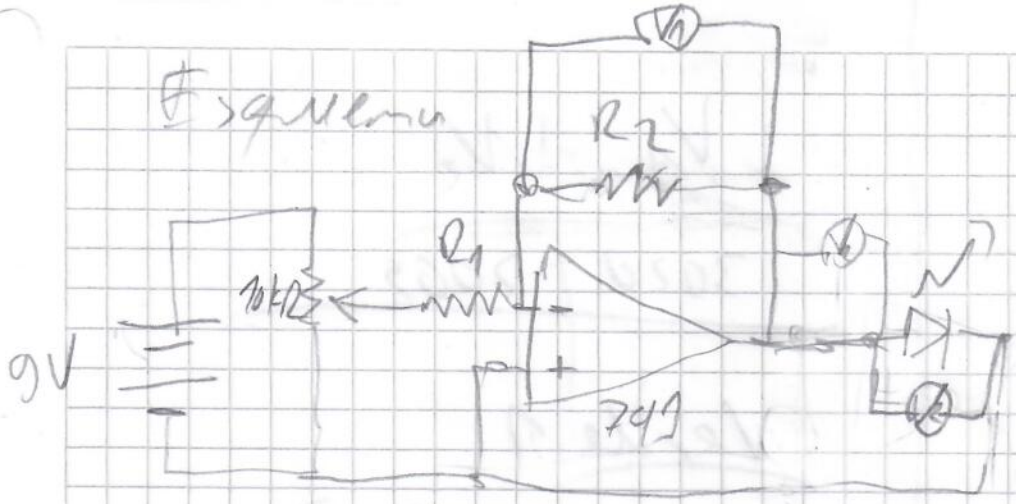
$eV_{th} = E_{emitida} = h\nu \Rightarrow V_{th} = \frac{h}{e}\nu$

⇒ Experimento

Midiendo V_{th} varias veces de ajuste lineal a $m(V - V_{th})$ →. De esta medida sacamos $\frac{h}{e}$. Después (lineal) de ajuste.

25/11/20

Esquema



$R_1 = 102.0 \pm 0.05\ \Omega$

$R_2 = 102.7 \pm 0.05\ \Omega$

Rojo 2

V_1 (mV)	V_2 (mV)
4.4	10.0
9.6	33.6
9.3	228.8
10.5	544
12.4	794
13.0	895
15.0	1255
18.5	1506
22.2	1545

V_1	V_2
29.2	1580
37.1	1602
44.6	1616
64.5	1647
71.2	1648
78.3	1653
81.5	1656
91.3	1663
99.7	1660

V_1	V_2
1777	1675
179.5	1684
191.2	1656
160.8	1698
188	1705
233	1722
287	1776
295	1735
324	1746
396	1761
471	1775
634	1803
515	1783
777	1815
834	1834
920	1843
1165	1873
1430	1904

Se satura

V_1	V_2
3020	2065

Verde 1

V_1 (mV)	V_2 (mV)
4.3	-14.0
8.8	160.2
9.3	277.0
10.5	460
11.5	650
13.7	922
14.2	1130
15.2	1375
16.2	1508
19.1	1652
21.3	1737
23.5	1775
27.4	1768
32.7	1783

V_1	V_2
45.0	1808
50.8	1816
58.7	1825
63.6	1830
69.2	1835
70.7	1841
88.4	1851
55.2	1855
110.5	1865
122.3	1874
137.1	1879
157.6	1886
172.5	1894
193	1903
214	1911
254	1926
361	1950
462	1961
513	1994

V_1	V_2
678	2026
916	2026
2.89	2.924

se satura

2911,20

Hasta un valor.

Se ve el LED amarillo cuando el error es grande se tenga que volver a medir todo

Amarillo

V_1[mV]	V_2[mV]
55.6	53.5
59.6	57.5
63.0	60.7
78.1	76.1
80.8	78.8
55.4	83.6

Rojo 2

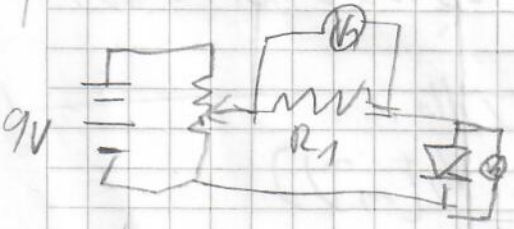
V_1	V_2
92.0	84.1
96.1	94.4
99.5	97.8
108.4	106.0
115.8	114.6

15:35

Error extraño en opamp

Se quemaron dos diodos

Nuevo circuito:



$R_1 = 700.2$

Nota: (El Error fue que la salida del opamp es negativa. No queremos operación inversora de LED).

V_1 [mV]	V_2 [mV]
3.1	1489
1.5	1462
0.3	1457
14.3	1557
18.3	1568
25.6	1552
41.6	1607
50.4	1616
57.4	1621
63.5	1627
68.8	1631
76.2	1637
83.1	1641
96.3	1648
104.0	1653
121.3	1661
141.2	1670
153	1675
148.1	1667

V_1	V_2
232	1707
270	1712
342	1728
438	1748
541	1767
676	1781
843	1812
945	1825
1202	1854
1400	1880
1690	1912
2020	1944
Se satura	
5290	2272
5790	2379

Verde 2

V_1[mV]	V_2[mV]
0.2	1590
2.7	1702
8.7	1751
17.7	1.775
26.6	1796
36.2	1811
46.2	1822
51.6	1828
63.5	1838
72.4	1845
81.9	1852
96.7	1861
105.4	1866
113.2	1870
140.7	1883
164.5	1893
205	1910
260	1928
326	1946

V_1	V_2
455	1979
730	2044
1129	2116
1378	2153
834	2057
595	2009
1167	2123
~~Se satura~~	
1430	2160
1340	2152
Se satura	
5290	2812
4860	2740

Todo son mV

/ /

Anexo 2

V_1	V_2
0.5	1536
1.2	1574
4.4	1626
8.1	1653
12.6	1673
20.0	1694
24.3	1703
34.3	1720
40.4	1728
49.9	1740
60.7	1750
70.2	1758
84.0	1767
91.6	1773
121.2	1790
136.5	1798
157.3	1807
193.6	1817
206	1824
267	1845

V_1	V_2
346	1877
550	1910
666	1932
801	1955
915	1974
1184	2016
1412	2050
1670	2080
Saturación	
4890	2510
5890	2567

Azul

V_1	V_2
0.7	2228
6.5	22124
2.0	2330
3.4	2305
9.0	2441
16.1	2479
22.9	2504
31.6	2528

V_1	V_2
35.2	2541
42.2	2554
52.2	2573
63.2	2592
72.2	2607
77.1	2614
93.5	2637
110.2	2658
122.2	2672
137.4	2688
161.7	2712
214	2759
283	2804
410	2872
450	2890
628	2921
626	2959
716	2984
824	3015
987	3053

V_1	V_2
1114	3080
Se satura	
4560	3543
4700	3493

Roja

V_1	V_2
0.5	879
3.2	897
10.8	955
13.3	976
22.8	989
31.6	1004
51.8	1026
60.1	1033
71.8	1043
89.3	1054
100.7	1057
118.2	1063
147	1073

$\delta \sim 10\%$

V_1	V_2
103.4	1083
222	1088
250	1092
333	1104
434	1113
602	1126
706	1133
828	1139
1206	1159
1408	1167
Se satura	
6120	1243
6730	1.247

Verde 2

V_1	V_2
3.4	2489
9.1	2559
15.0	2599
23.6	2671

V_1	V_2
30.0	2692
40.6	2666
50.8	2683
62.2	2700
69.6	2709
75.1	2720
91.4	2733
103.1	2743
121.8	2758
142.4	2775
168.5	2788

30 a 220
Nuevo diseño

$R_2 = 100.2\,\Omega$

$R_1 = 102.0\,\Omega$

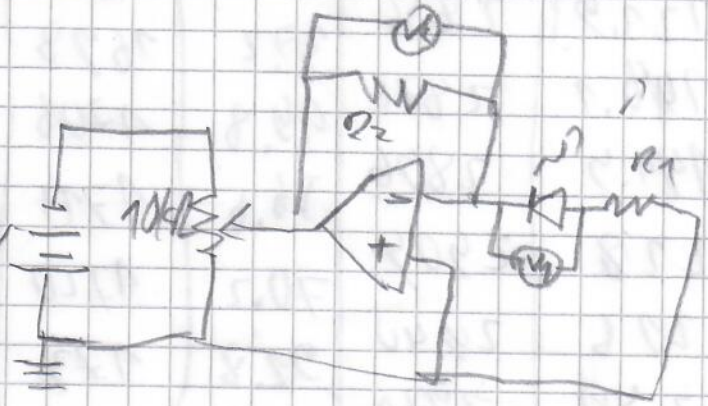
(Experimentar con otros para no saturar mejor)

Verde

V_1	V_2
1808	1984
1845	435

Demasiado sensible

Lo hace mover mediante obstáculo cuando se enciende.



Idea. Invertir circuito.

0 siempre min escala

R=100.2

Regresando a

AZUL 2

V_2 [mV]	V_1 [mV]
~~07~~	
~~191~~	0.
2510	1.7
2670	9.3
2720	20.5

V_1	V_2
32.1	2756
43.7	2777
50.8	2789
60.8	2802
80.3	2823
100.0	2841
121.5	2856
148.1	2872
191.5	2896
278	2909
296	2940
389	2970
540	3012

V_1	V_2
687	3045
759	3052
1105	3123
Se satura	
3640	3355
4560	3526

ROJO 2

V_1	V_2
0.5	1561
9.5	1631
13.9	1664
25.4	1685
37.2	1698
49.8	1708
56.6	1713
70.2	1720
92.8	1730
115.3	1738
140.8	1746

V_1	V_2
1699	1753
217	1764
322	1781
413	1793
538	1807
655	1820
916	1840
1191	1855
1487	1870
Se satura	
5960	2039

18:56

Se satura demasiado rápido con el circuito 2. Medirlo directamente cuando se disparo o se quiemo el diodo.

→

El riesgo es modificar el voltaje de saturación en vez del voltaje umbral.

19:08

mediciones directas

	V_{ab} (mV)
Rojo 1	2030 ± 50
Rojo 2	1960 ± 50
Verde 1	2420 ± 50
Verde 2	2430 ± 50
Amarillo	2160 ± 50
Azul 2	2720 ± 50
Azul 2	2850 ± 50
IR	1150 ± 50

3.12.20

Andrés dice que tomar las frec. como todo el espectro modo sumo resistores.

20:17

→ Encontré tabla con frecuencia de LED / código de color.

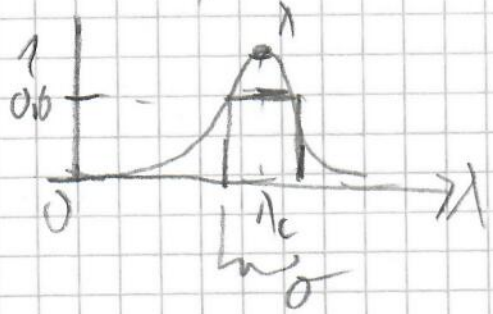
→ Tiene curvas para el espectro de longitudes de onda.

20:27

Coinciden con mejores colores.

Para 0

Hay curvas para cada λ de la fuente



aproximar a $\exp\left[-\left(\frac{(\lambda-\lambda_c)^2}{2\sigma^2}\right)\right.$

(no está normalizada)

Así si $\lambda-\lambda_c=\sigma \rightarrow \exp\left(-\frac{\sigma^2}{2\sigma^2}\right)=\frac{1}{\sqrt{e}}$

El error es el ancho de la curva a $\frac{1}{\sqrt{e}}=0.607$

Con esto se obtiene

LED	λ (nm)
Rojo1	657 ± 15
Rojo2	657 ± 15
Verde 1	525 ± 17
Verde 2	525 ± 17
Amarillo	583 ± 16

LED	λ (nm)
Azul 1	470 ± 27
Azul 2	430 ± 35
Infrarrojo	

La bitácora de Andrés esta anexa y contiene el análisis y sus calculos para la frecuencia.

Bitácora Práctica 3

Diodos: R, V, Amarillo, Cyan, Azul, Infrarojo
(R: dos, V: dos, Amarillo: dos, "Azules distintos")

Ajuste lineal: $y = m(x - x_0)$ ← El ajuste a x_0 será $V_{2,sat}$

$$(\sigma_y)^2 = \left(\frac{\partial y}{\partial m}\right)^2 \sigma_m^2 + \left(\frac{\partial y}{\partial x}\right)^2 \sigma_x^2 + \left(\frac{\partial y}{\partial x_0}\right)^2 \sigma_{x_0}^2 = (x - x_0)^2 \sigma_m^2 + m^2(\sigma_x^2 + \sigma_{x_0}^2)$$

$y = I$, $x = V_1$

$$I = V_2/R \Rightarrow \sigma_I^2 = \frac{V_2^2}{R^4}\sigma_R^2 + \frac{\sigma_{V_2}^2}{R^2}$$

Para ajuste a h:

$$E = h\nu, \quad V = E/q, \quad \nu = C/\lambda \Rightarrow V = \frac{h\nu}{q}$$

Para generalizar el ajuste tomamos $V = \frac{h\nu}{q} + V_0$

q, C se toman como exactas: $q = 1.602176634 \times 10^{-19} C$, $C = 299792458\ m/s$

$$\sigma_V = \sqrt{\left(\frac{\nu\sigma_h}{q}\right)^2 + \left(\frac{h\sigma_\nu}{q}\right)^2 + \sigma_{V_0}^2}$$

$$\Rightarrow \sigma_V^2 = \frac{\nu^2}{q^2}\sigma_h^2 + \frac{h^2}{q^2}\sigma_\nu^2, \quad \sigma_V = \frac{1}{q}\sqrt{(\nu\sigma_h)^2 + (h\sigma_\nu)^2}$$

$$\sigma_\nu^2 = \frac{C^2}{\lambda^4}\sigma_\lambda^2, \quad \sigma_\nu = \frac{C}{\lambda^2}\sigma_\lambda = \nu\frac{\sigma_\lambda}{\lambda}$$

Espectro visible:

		Propuesta:
Rojo:	625 - 700 nm	$\lambda = 662.5 \pm 37.5$ nm
Amarillo:	565 - 590 nm	$\lambda = 577.5 \pm 12.5$ nm
Verde:	500 - 565 nm	$\lambda = 532.5 \pm 32.5$ nm
Cyan:	485 - 500 nm	$\lambda = 492.5 \pm 7.5$ nm
Azul:	450 - 485 nm	$\lambda = 467.5 \pm 17.5$ nm

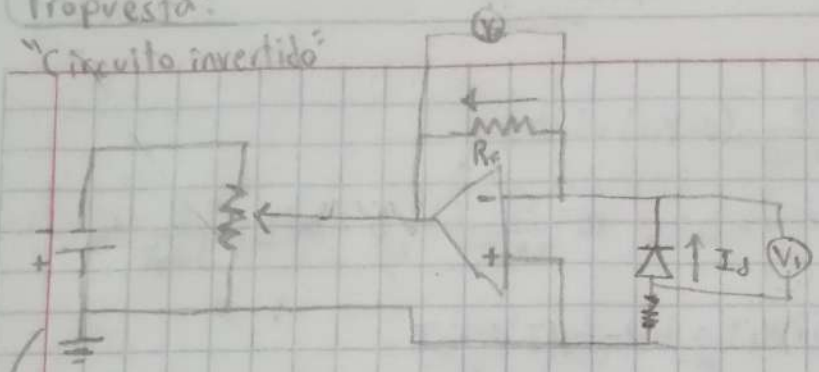
Rango máximo en el que se puede poner la longitud de onda

Nueva propuesta: Gráfica de espectro de diodos

$V_2 = -I_s R_f$

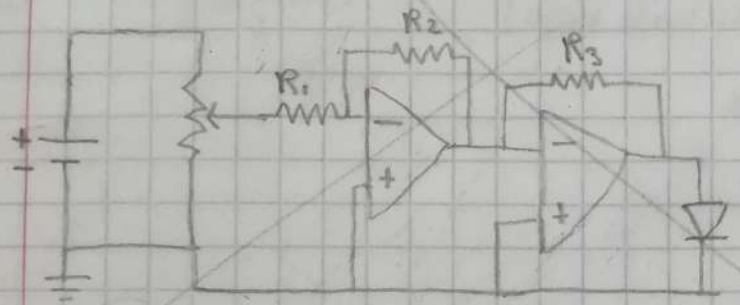
Propuesta:

"Circuito invertido"



No estamos seguros de las polarizaciones
Si están bien las polaridades ✓

Propuesta:



Assoziogramm

Arbeits B

Avances bitácora gramática

Vi 11

$R_1 = 100.2\ \Omega$

Experimento 1: "Muy sensible"

	V_{isat} [V] ← Cuando enciende
Rojo 1	2.03
Verde 1	2.42 ±1
Amarillo 1	2.16
Azul 1	2.72
IR	1.19
Rojo 2	1.96
Verde 2	3.18 (?!)
Azul 2	2.89

Experimento 2:

	V_{isat} [V] →
Rojo 1	1.79
Verde 1	2.01
Amarillo 1	1.91
Azul 1	2.75
IR	1.12
Rojo 2	1.82
Verde 2	2.43
Azul 2	2.75

$c = \lambda \nu$

Lado P, Lado N

erzeugen = generar

Gesch erhöhen

$I = I_0 e^{e(V-V_{th})/nK_BT}$, $\ln(I) = \frac{e}{nK_BT}(V-V_{th}) + \ln(I_0)$

ie $\ln(I) = mV + b$, $m = \frac{e}{nK_BT}$, $b = \ln(I_0) - \frac{e}{nK_BT}V_{th}$

de ahí: $Y(0) = -be/m = eV_{th} - \ln(I_0)nK_BT = h\nu - \ln(I_0)nK_BT$

$\uparrow E = eV_{th} = h\nu$ ↑ ordenada cambia en cada diodo! (inviable)

Depurado Ro 1a y Ve 1 (circuito viejo)

Ent. Ro 1b, Ro 2 → Ro 1, Ro 2

Ve 2, Ve 3 → Ve 1, Ve 2

Proc.: determinó

Carátula presentación:
Univ. Escudo
Fac. Título. Nombre autor(es)

UNAM → Grande
Fac. → Grande

Solo 1 par de gráficas I_d-V_0

Infrarojo → IR

$M = h/e$

$I_d(V_1)$, lo relevante es V_{th}

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
import os #Para adquirir el path global y guardar figuras
```

In [39]:
```
h_real = 6.62607015e-34 #J*s
e = 1.602176634e-19 #C
c = 299792458 #m/s
#R = 102.7 #Ohm En los primeros dos experimentos
R = 100.2 #Ohm
sR = 0.05
M_real = h_real/e

def y(x,m,x0): #Modelo lineal general para ajustar
    return m*(x-x0)

def sy(x,sx,m,sm,x0,sx0):
    return np.sqrt(m**2*(sx**2+sx0**2)+sm**2*(x-x0)**2)

#En deshuso
#def V(nu,M,V0): #Modelo lineal para ajustar h, q exacta. M = h/q
#    return M*nu+V0
#
#def sV(nu,snu,M,sM,V0,sV0):
#    return np.sqrt((nu*sM)**2+(M*snu)**2+sV0**2)

def V(nu,M): #Modelo lineal para ajustar h, q exacta. M = h/q
    return M*nu

def sV(nu,snu,M,sM):
    return np.sqrt((nu*sM)**2+(M*snu)**2)
```

In [3]:
```
#Lambdas directo del espectro visible
lamb_rojo, slamb_rojo = (625+700)/2, (700-625)/2
lamb_verde, slamb_verde = (500+565)/2, (565-500)/2
lamb_amarillo, slamb_amarillo = (565+590)/2, (590-565)/2
#lamb_cyan, slamb_cyan = (485+500)/2, (500-485)/2
lamb_azul, slamb_azul = (450+485)/2, (485-450)/2
lamb_IR, slamb_IR = 940, 940-760
```

In [3]:
```
#Lambdas y desviaciones estándar estimadas de la gráfica técnica
lamb_IR, slamb_IR = 945, 22 #Infrared (1)
lamb_rojo, slamb_rojo = 657, 15 #Se tomó el denominado rojo, pero hay otros tonos de rojo posibles
lamb_verde, slamb_verde = 525, 27 #Ultra Green (A)
lamb_amarillo, slamb_amarillo = 583, 18 #Yellow
lamb_azul, slamb_azul = 430, 35 #Ultra Blue
lamb_cyan, slamb_cyan = 470, 27 #Pure Blue
```

In [77]:
```
#Con cyan
L = ["Ro1","Ro2","Ve1","Ve2","Am1","Azul1","Azul2","IR"]
Nombre = ["rojo 1","rojo 2", "verde 1", "verde 2","amarillo", "azul 1","azul
 2","infrarrojo"]
Lamb = np.array([lamb_rojo,lamb_rojo,lamb_verde,lamb_verde,lamb_amarillo,lamb_
cyan,lamb_azul,lamb_IR])*1e-9 #m
sLamb = np.array([slamb_rojo,slamb_rojo,slamb_verde,slamb_verde,slamb_amarillo
,slamb_cyan,slamb_azul,slamb_IR])*1e-9 #m
V1th = [0,0,0,0,0,0,0,0]
sV1th = [0,0,0,0,0,0,0,0]
```

Método 1

In [5]:
```
def Read(i):
    Datos = np.loadtxt("Datos"+L[i]+".txt",skiprows=1).T
    Datos = Datos[:, Datos[0].argsort()] #Ordena datos

    LOW = [25,16,19,10,20,23,10,21]#indice menor
    low = LOW[i] #indice menor
    ind = np.arange(low,len(Datos[0])) #Datos a ajustar desde low hasta el máx
imo
    ind_extra = np.arange(0,low)
    d_ajustados = len(Datos[0])-low

    V1 = Datos[1][ind]
    V2 = Datos[0][ind]
    sV1 = Datos[3][ind]
    sV2 = Datos[2][ind]
    Ic = V2/R
    sIc = np.sqrt(V2**2*sR**2/(R**4)+(sV2/R)**2)

    V1_extra = Datos[1][ind_extra]
    V2_extra = Datos[0][ind_extra]
    sV1_extra = Datos[3][ind_extra]
    sV2_extra = Datos[2][ind_extra]
    Ic_extra = V2_extra/R
    sIc_extra = np.sqrt(V2_extra**2*sR**2/(R**4)+(sV2_extra/R)**2)

    return V1, Ic, sV1, sIc, V1_extra, Ic_extra, sV1_extra, sIc_extra, d_ajust
ados
```

In [66]:
```
#Rojo1
i = 0
V1, Ic, sV1, sIc, V1_extra, Ic_extra, sV1_extra, sIc_extra, d_ajustados = Read
(i)
popt, pcov = curve_fit(y, xdata = V1, ydata = Ic, sigma = sIc)
m_fit, x0_fit = popt
sm_fit, sx0_fit = np.sqrt(pcov[0][0]), np.sqrt(pcov[1][1])
print("m_fit = "+str(m_fit)+" +- "+str(sm_fit))
print("x0_fit = "+str(x0_fit)+" +- "+str(sx0_fit))
```

```
m_fit = 0.08545421947741375 +- 0.0011276096106633958
x0_fit = 1713.9598265831926 +- 1.5476839387967183
```

In [96]:
```
#Rojo2
i = 1
V1, Ic, sV1, sIc, V1_extra, Ic_extra, sV1_extra, sIc_extra, d_ajustados = Read
(i)
popt, pcov = curve_fit(y, xdata = V1, ydata = Ic, sigma = sIc)
m_fit, x0_fit = popt
sm_fit, sx0_fit = np.sqrt(pcov[0][0]), np.sqrt(pcov[1][1])
print("m_fit = "+str(m_fit)+" +- "+str(sm_fit))
print("x0_fit = "+str(x0_fit)+" +- "+str(sx0_fit))
```

```
m_fit = 0.16156529172551726 +- 0.011506967589816156
x0_fit = 1780.8445484790097 +- 4.464394427173474
```

In [99]:
```
#Verde1
i = 2
V1, Ic, sV1, sIc, V1_extra, Ic_extra, sV1_extra, sIc_extra, d_ajustados = Read
(i)
popt, pcov = curve_fit(y, xdata = V1, ydata = Ic, sigma = sIc)
m_fit, x0_fit = popt
sm_fit, sx0_fit = np.sqrt(pcov[0][0]), np.sqrt(pcov[1][1])
print("m_fit = "+str(m_fit)+" +- "+str(sm_fit))
print("x0_fit = "+str(x0_fit)+" +- "+str(sx0_fit))
```

```
m_fit = 0.04932791908859819 +- 0.0007583532266485239
x0_fit = 1888.3127232716035 +- 2.507666460481306
```

In [102]:
```
#Verde2
i = 3
V1, Ic, sV1, sIc, V1_extra, Ic_extra, sV1_extra, sIc_extra, d_ajustados = Read
(i)
popt, pcov = curve_fit(y, xdata = V1, ydata = Ic, sigma = sIc)
m_fit, x0_fit = popt
sm_fit, sx0_fit = np.sqrt(pcov[0][0]), np.sqrt(pcov[1][1])
print("m_fit = "+str(m_fit)+" +- "+str(sm_fit))
print("x0_fit = "+str(x0_fit)+" +- "+str(sx0_fit))
```

```
m_fit = 0.013446659584589954 +- 0.0006951073478863978
x0_fit = 2666.230174816681 +- 4.659784290695643
```

In [85]:
```
#Amarillo1
i = 4
V1, Ic, sV1, sIc, V1_extra, Ic_extra, sV1_extra, sIc_extra, d_ajustados = Read
(i)
popt, pcov = curve_fit(y, xdata = V1, ydata = Ic, sigma = sIc)
m_fit, x0_fit = popt
sm_fit, sx0_fit = np.sqrt(pcov[0][0]), np.sqrt(pcov[1][1])
print("m_fit = "+str(m_fit)+" +- "+str(sm_fit))
print("x0_fit = "+str(x0_fit)+" +- "+str(sx0_fit))
```

```
m_fit = 0.060954826949696575 +- 0.0009134918552428922
x0_fit = 1822.0942188119 +- 1.974902156291858
```

In [87]:
```
#Azul1
i = 5
V1, Ic, sV1, sIc, V1_extra, Ic_extra, sV1_extra, sIc_extra, d_ajustados = Read
(i)
popt, pcov = curve_fit(y, xdata = V1, ydata = Ic, sigma = sIc)
m_fit, x0_fit = popt
sm_fit, sx0_fit = np.sqrt(pcov[0][0]), np.sqrt(pcov[1][1])
print("m_fit = "+str(m_fit)+" +- "+str(sm_fit))
print("x0_fit = "+str(x0_fit)+" +- "+str(sx0_fit))
```

```
m_fit = 0.039416306341173726 +- 0.0014936905064095037
x0_fit = 2801.8917644478333 +- 8.075792756863747
```

In [89]:
```
#Azul2
i = 6
V1, Ic, sV1, sIc, V1_extra, Ic_extra, sV1_extra, sIc_extra, d_ajustados = Read
(i)
popt, pcov = curve_fit(y, xdata = V1, ydata = Ic, sigma = sIc)
m_fit, x0_fit = popt
sm_fit, sx0_fit = np.sqrt(pcov[0][0]), np.sqrt(pcov[1][1])
print("m_fit = "+str(m_fit)+" +- "+str(sm_fit))
print("x0_fit = "+str(x0_fit)+" +- "+str(sx0_fit))
```

```
m_fit = 0.04324573968167225 +- 0.0026229092360033133
x0_fit = 2878.9018180118096 +- 8.698030044131542
```

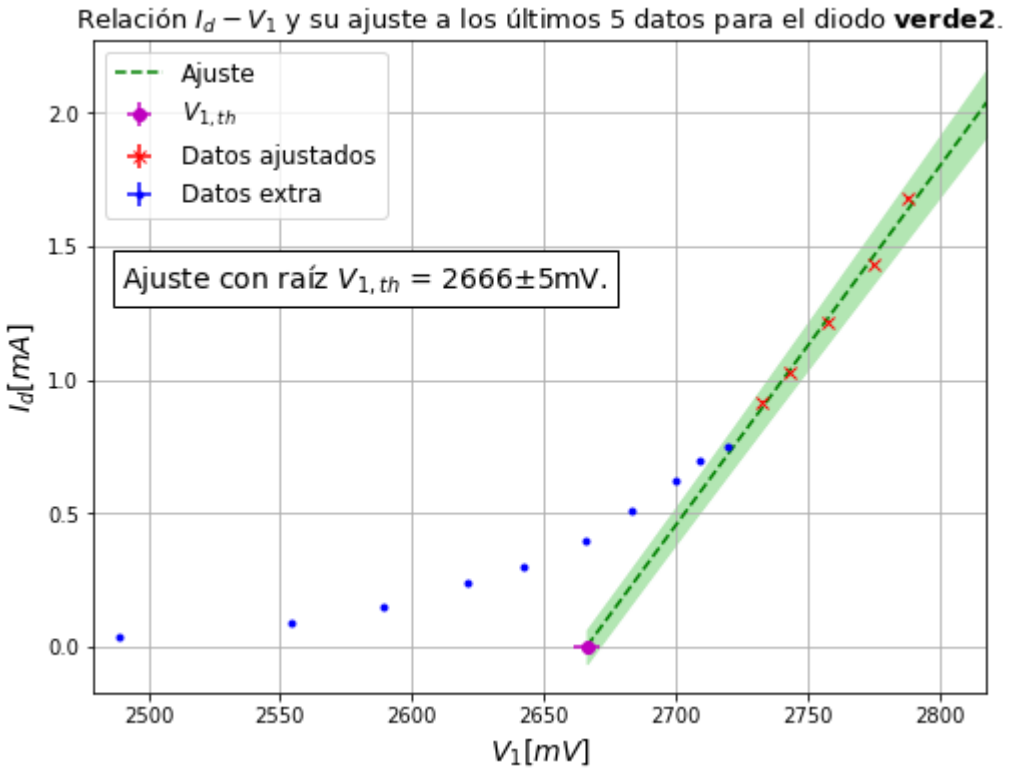
In [91]:
```
#IR
i = 7
V1, Ic, sV1, sIc, V1_extra, Ic_extra, sV1_extra, sIc_extra, d_ajustados = Read
(i)
popt, pcov = curve_fit(y, xdata = V1, ydata = Ic, sigma = sIc)
m_fit, x0_fit = popt
sm_fit, sx0_fit = np.sqrt(pcov[0][0]), np.sqrt(pcov[1][1])
print("m_fit = "+str(m_fit)+" +- "+str(sm_fit))
print("x0_fit = "+str(x0_fit)+" +- "+str(sx0_fit))
```

```
m_fit = 0.5937737750075394 +- 0.08962577526595315
x0_fit = 1132.8137087298692 +- 4.432573118138958
```

In [106]:
```
#Gráfica individual
V_cont = np.linspace(x0_fit,V1.max()+30,100)

fig = plt.figure(figsize=(8,6))
plt.errorbar(x0_fit,0,0,sx0_fit,"mo",label=r"$V_{1,th}$")
plt.errorbar(V1,Ic,sIc,sV1,"rx",label="Datos ajustados")
plt.errorbar(V1_extra,Ic_extra,sIc_extra,sV1_extra,"b.",label="Datos extra") #
Datos extra que fueron medidos, pero no ajustados
y_fit = y(V_cont,m_fit,x0_fit)
plt.plot(V_cont,y_fit,"g--",label="Ajuste")
sy_fit = sy(V_cont,0,m_fit,sm_fit,x0_fit,sx0_fit)
plt.fill_between(V_cont,y_fit+sy_fit,y_fit-sy_fit,color=(0.7,0.9,0.7))
plt.title(r"Relación $I_d - V_1$ y su ajuste a los últimos "+str(d_ajustados)+
" datos para el diodo "+ r"$\bf{"+Nombre[i]+ "}$.",fontsize=13)
plt.xlabel(r"$V_1[mV]$",fontsize=14)
plt.ylabel(r"$I_d[mA]$",fontsize=14)
plt.xlim(V1_extra.min()-10,V1.max()+30)
POSX = [1400,1560,1600,2490,1550,2050,2755,830] #DatosR1a, DatosR1b , DatosR
2...
POSY = [14,13,10,1.35,12,8,8,60] #DatosR1a, DatosR1b, DatosR2...
posx , posy = POSX[i], POSY[i]
props=dict(boxstyle='square', facecolor='white', alpha=1)
plt.text(posx, posy, r"Ajuste con raíz $V_{1,th}$ = "+"{:.0f}".format(x0_fit)+
"$\pm$"+"{:.0f}".format(sx0_fit)+"mV.",
         fontsize=14, bbox=props)
plt.legend(fontsize=12)
plt.grid()
plt.show()

#Para guardar la gráfica
fig.savefig("C:\Andrés\IX Semestre\Lab. Contemporánea I\Práctica 3\Plots\ "[:-
1]+str(i)+".png",bbox_inches='tight') #El [:-1] es para quitar el espacio, nec
esario para que se comente bien por el \
```



In [52]:
```
#Recopilación de voltajes de Vth
for i in range(0,8):
    V1, Ic, sV1, sIc, V1_extra, Ic_extra, sV1_extra, sIc_extra, d_ajustados =
Read(i)
    popt, pcov = curve_fit(y, xdata = V1, ydata = Ic, sigma = sIc)
    m_fit, x0_fit = popt
    sm_fit, sx0_fit = np.sqrt(pcov[0][0]), np.sqrt(pcov[1][1])

    V1th[i], sV1th[i] =  x0_fit*1e-3, sx0_fit*1e-3 #Convertimos a V
    nu = c/Lamb #Hz
    snu = nu*sLamb/Lamb #Hz

print(V1th)
print(sV1th)
```

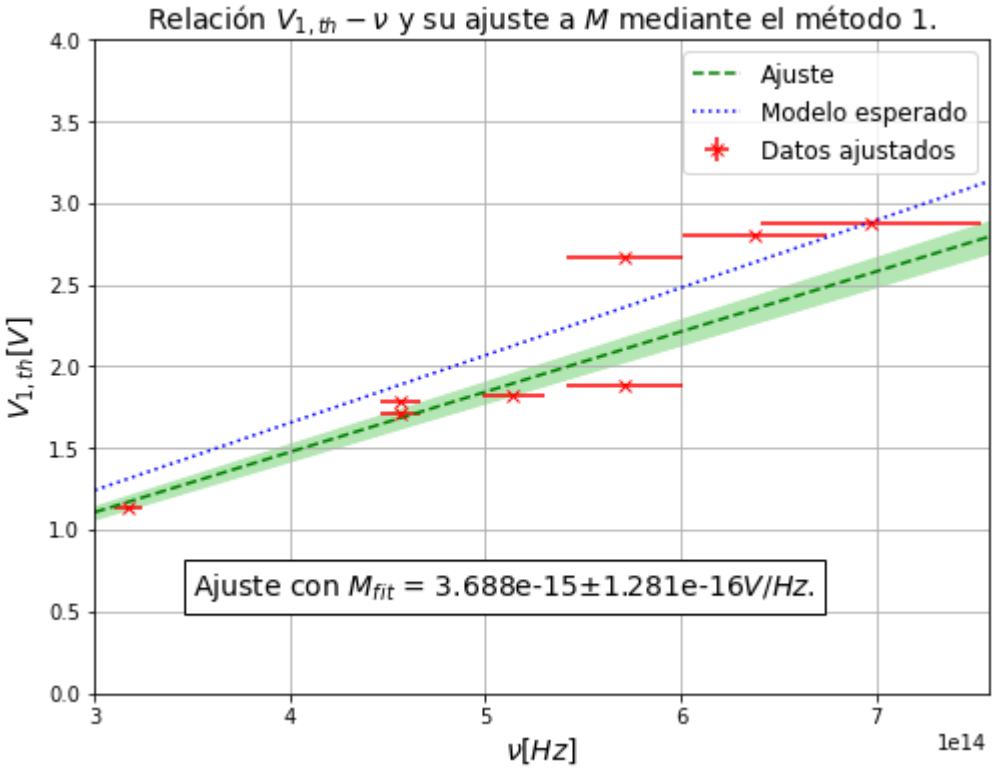
```
[1.71395983 1.78084455 1.88831272 2.66623017 1.82209422 2.80189176
 2.87890182 1.13281371]
[0.00154768 0.00446439 0.00250767 0.00465978 0.0019749  0.00807579
 0.00869803 0.00443257]
```

In [53]:
```
#Ajuste a h
popt, pcov = curve_fit(V, xdata = nu, ydata = V1th, sigma = sV1th)
M_fit = popt[0]
sM_fit = np.sqrt(pcov[0][0])
h_fit, sh_fit = M_fit*e, sM_fit*e
print("M_fit = "+str(M_fit)+" +- "+str(sM_fit))
print("h_fit = "+str(h_fit)+" +- "+str(sh_fit))
print("h_real = "+str(h_real))
print("Diferencia porcentual a la real: "+"{:.3f}".format(100*np.abs(h_fit-h_r
eal)/h_real)+"%")
```

```
M_fit = 3.687859400758826e-15 +- 1.2813465379821185e-16
h_fit = 5.908602161373032e-34 +- 2.0529434832117436e-35
h_real = 6.62607015e-34
Diferencia porcentual a la real: 10.828%
```

In [54]:
```
numin, numax = 3e14, nu.max()+0.6e14
nu_cont = np.linspace(numin,numax,100)

plt.figure(figsize=(8,6))
plt.errorbar(nu,V1th,sV1th,snu,"rx",label="Datos ajustados")
y_fit = V(nu_cont,M_fit)
plt.plot(nu_cont,y_fit,"g--",label="Ajuste")
sy_fit = sV(nu_cont,0,M_fit,sM_fit)
plt.fill_between(nu_cont,y_fit+sy_fit,y_fit-sy_fit,color=(0.7,0.9,0.7))
plt.plot(nu_cont,V(nu_cont,M_real),"b:",label="Modelo esperado")
plt.title(r"Relación $V_{1,th} - \nu$ y su ajuste a $M$ mediante el método 1."
,fontsize=14)
plt.xlim(numin,numax)
plt.ylim(0,4)
plt.xlabel(r"$\nu[Hz]$",fontsize=14)
plt.ylabel(r"$V_{1,th}[V]$",fontsize=14)
posx , posy = 3.5e14, 0.6
props=dict(boxstyle='square', facecolor='white', alpha=1)
plt.text(posx, posy, r"Ajuste con $M_{fit}$ = "+"{:.3e}".format(M_fit)+"$\pm$"
+"{:.3e}".format(sM_fit)+r"$V/Hz$.",
         fontsize=14, bbox=props)
plt.legend(fontsize=12)
plt.grid()
plt.show()
```



Método 2

In [55]:
```
#Con cyan
V1th = np.array([2.03,1.96,2.42,2.43,2.16,2.72,2.89,1.19]) #V
sV1th = np.array([0.05,0.05,0.05,0.05,0.05,0.05,0.05,0.05]) #V
Lamb = np.array([lamb_rojo,lamb_rojo,lamb_verde,lamb_verde,lamb_amarillo,lamb_
cyan,lamb_azul,lamb_IR])*1e-9 #m
sLamb = np.array([slamb_rojo,slamb_rojo,slamb_verde,slamb_verde,slamb_amarillo
,slamb_cyan,slamb_azul,slamb_IR])*1e-9 #m
nu = c/Lamb #Hz
snu = nu*sLamb/Lamb #Hz
```

In [65]:
```
print(snu)
```

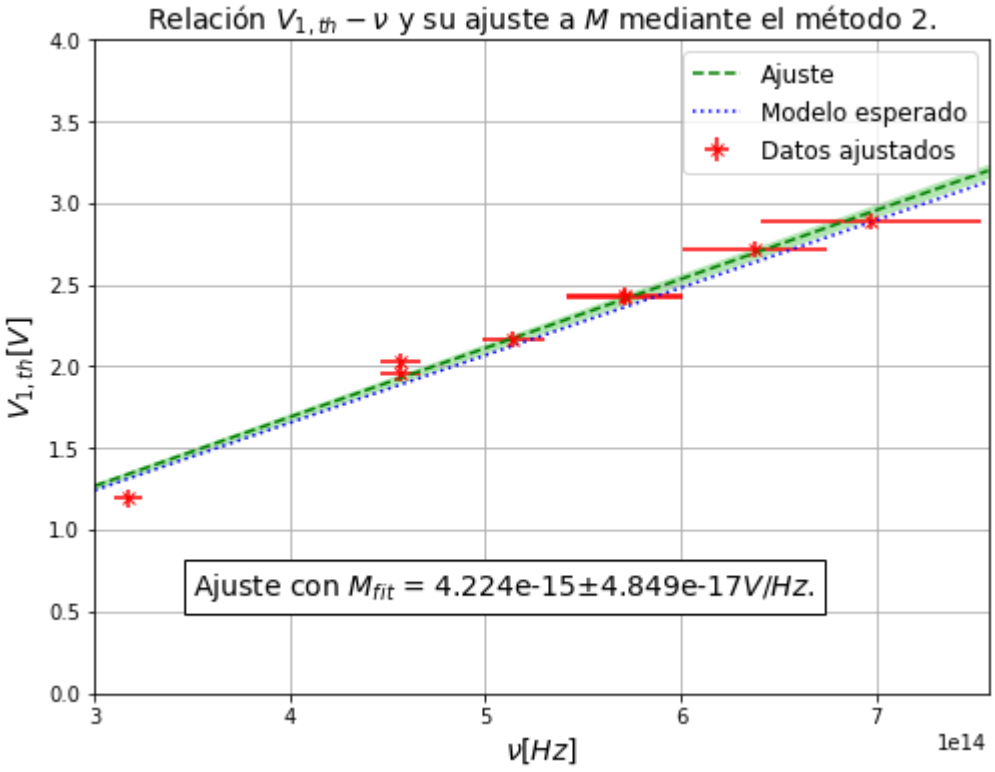
```
[1.04179249e+13 1.04179249e+13 2.93674245e+13 2.93674245e+13
 1.58765486e+13 3.66428084e+13 5.67481667e+13 7.38549769e+12]
```

In [57]:
```
#Ajuste a h
popt, pcov = curve_fit(V, xdata = nu, ydata = V1th, sigma = sV1th)
M_fit = popt[0]
sM_fit = np.sqrt(pcov[0][0])
h_fit, sh_fit = M_fit*e, sM_fit*e
print("M_fit = "+str(M_fit)+" +- "+str(sM_fit))
print("h_fit = "+str(h_fit)+" +- "+str(sh_fit))
print("h_real = "+str(h_real))
print("Diferencia porcentual a la real: "+"{:.3f}".format(100*np.abs(h_fit-h_r
eal)/h_real)+"%")
```

```
M_fit = 4.22431790349848e-15 +- 4.849292561101135e-17
h_fit = 6.7681034395731315e-34 +- 7.769423232826255e-36
h_real = 6.62607015e-34
Diferencia porcentual a la real: 2.144%
```

In [50]:
```
numin, numax = 3e14, nu.max()+0.6e14
nu_cont = np.linspace(numin,numax,100)

plt.figure(figsize=(8,6))
plt.errorbar(nu,V1th,sV1th,snu,"rx",label="Datos ajustados")
y_fit = V(nu_cont,M_fit)
plt.plot(nu_cont,y_fit,"g--",label="Ajuste")
sy_fit = sV(nu_cont,0,M_fit,sM_fit)
plt.fill_between(nu_cont,y_fit+sy_fit,y_fit-sy_fit,color=(0.7,0.9,0.7))
plt.plot(nu_cont,V(nu_cont,M_real),"b:",label="Modelo esperado")
plt.title(r"Relación $V_{1,th} - \nu$ y su ajuste a $M$ mediante el método 2."
,fontsize=14)
plt.xlim(numin,numax)
plt.ylim(0,4)
plt.xlabel(r"$\nu[Hz]$",fontsize=14)
plt.ylabel(r"$V_{1,th}[V]$",fontsize=14)
posx , posy = 3.5e14, 0.6
props=dict(boxstyle='square', facecolor='white', alpha=1)
plt.text(posx, posy, r"Ajuste con $M_{fit}$ = "+"{:.3e}".format(M_fit)+"$\pm$"
+"{:.3e}".format(sM_fit)+r"$V/Hz$.",
         fontsize=14, bbox=props)
plt.legend(fontsize=12)
plt.grid()
plt.show()
```



Generación de tablas con corrientes

In [6]:
```
def Write(i):
    """
    Escribe la tabla de datos con V1, V2 e Ic ordenados.
    """
    Datos = np.loadtxt("Datos"+L[i]+".txt",skiprows=1).T
    Datos = Datos[:, Datos[0].argsort()] #Ordena datos

    ind = np.arange(0,len(Datos[0])) #Indice total de datos
    V1 = Datos[1][ind]
    V2 = Datos[0][ind]
    sV1 = Datos[3][ind]
    sV2 = Datos[2][ind]
    Ic = V2/R
    sIc = np.sqrt(V2**2*sR**2/(R**4)+(sV2/R)**2)

    DatosW = np.array([V1,sV1,V2,sV2,Ic,sIc]).T
    np.savetxt("Datos_tabla_"+L[i]+".txt",DatosW,fmt="%.0f\t%.1f\t%.1f\t%.2f\t
%.2e\t%.2e") #Formato de cada columna, con tabulación
```

In [13]:
```
Write(7)
```