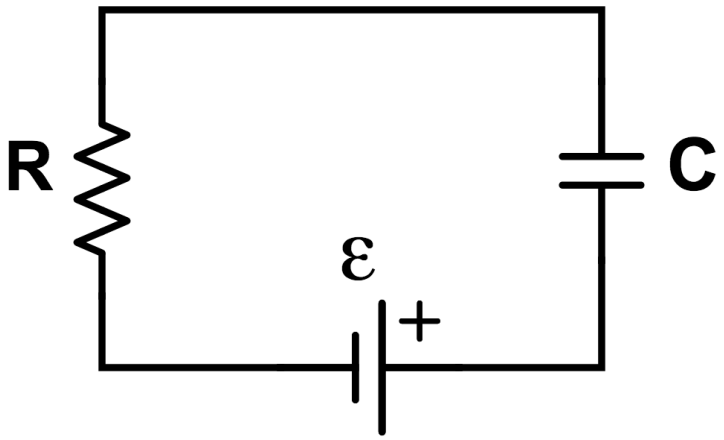
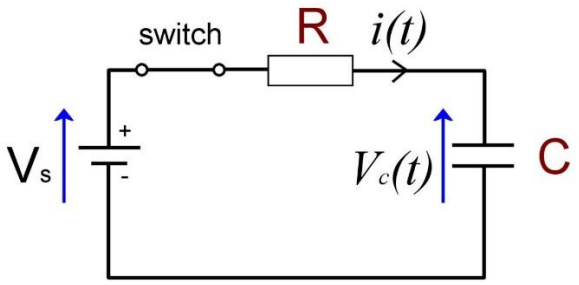


CURVAS DE UN CIRCUITO RC

Teoría del control

Circuito RC:

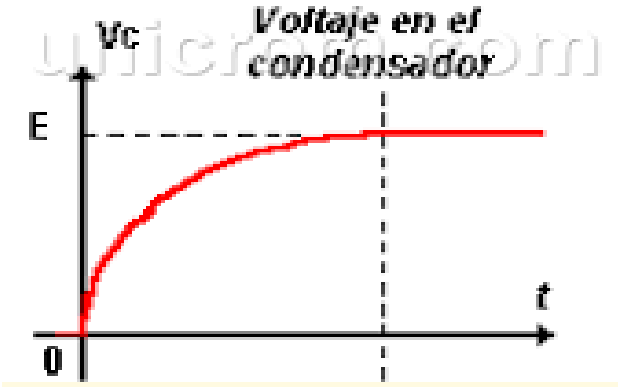


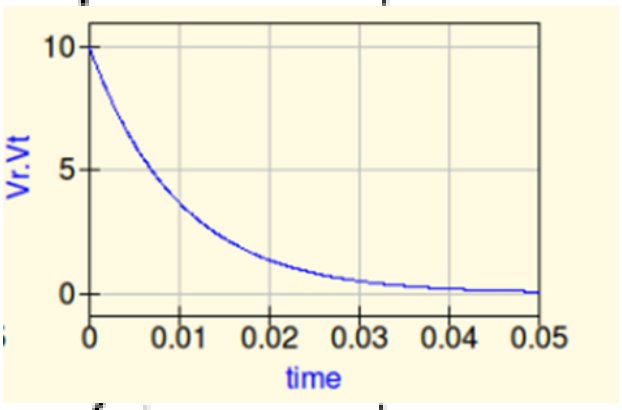
$$V_s - i(t)R - V_c(t) = 0$$

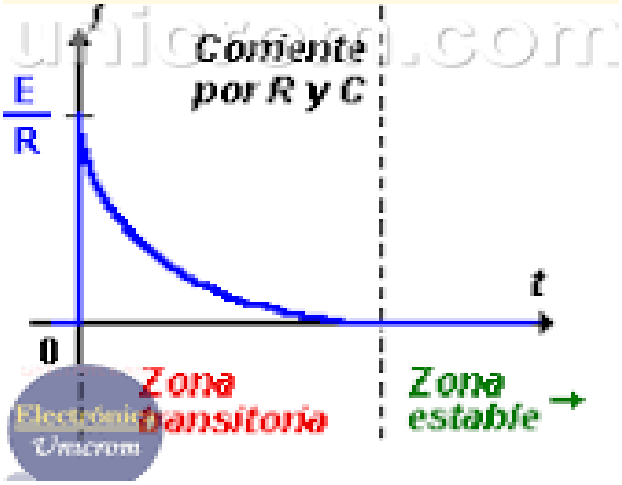
Un circuito RC, también llamados circuitos integradores pues en su salida muestran un voltaje que es la integral de la función de entrada, obedece como todo circuito eléctrico, las leyes de Kirchhoff, por lo que todo surge de la ecuación que se muestra en la imagen. Al considerar que Vs es una fuente DC y que el capacitor se encuentre descargado, se deduce la siguiente expresión para cuando el switch cierre y provoque la carga del capacitor en un tiempo dado:

$$v_{\mathbb{C}}(t) = v_S(1 - e^{-t/RC})$$

En donde la constante RC es conocido como τ, la constante de tiempo de este circuito. Para cuando t = 5τ, habrá un cociente $v_S/v_{\mathbb{C}}(t) \approx 0.99$, lo que significa que la tensión ha prácticamente igualado a la de la fuente. Con este sistema, tenemos las siguientes curvas (Lo relevante son las formas de las curvas:







Estas pueden ser descritas por las siguientes funciones respectivamente, todas originadas de la ecuación establecida de la ley de los voltajes de Kirchhoff, mostrada en el primer gráfico:

$$v_{\mathbb{C}}(t) = v_S(1 - e^{-t/RC})$$

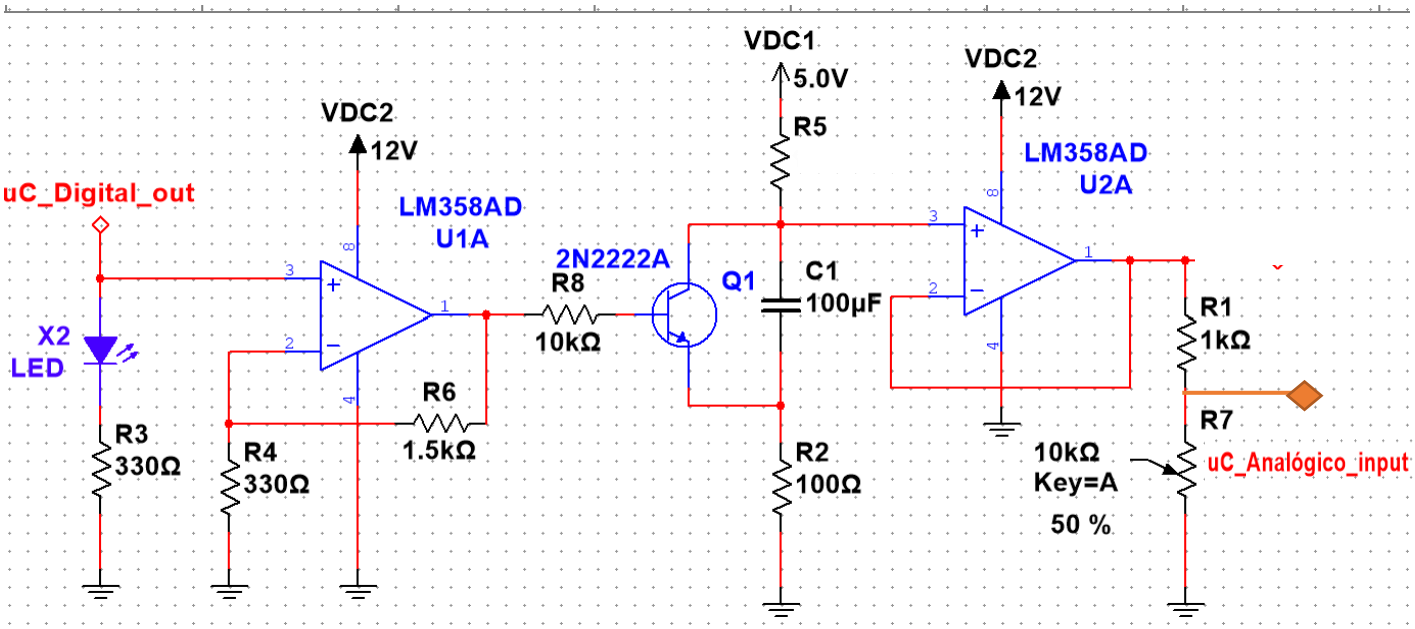
$$v_R(t) = v_S - v_{\mathbb{C}}(t)$$

$$i(t) = v_R(t)/R$$

Puntos para denotar en estas curvas:

- Son todas funciones exponenciales
- Este circuito es un sistema que se estabiliza después de un tiempo dado, en este caso como se dijo, aproximadamente después de t = 5τ.

Comprobación práctica:



En la práctica se utilizó el microcontrolador ESP32 DEVKIT V1 y se ideó el circuito de arriba para poder mostrar el voltaje de carga de un capacitor de 100µF en un circuito RC con una fuente DC de 5v, resistor de 1KΩ (Caso uno) y después de 100KΩ (caso2), para esto, R5 tomó los dos valores para cada caso.

El funcionamiento del circuito es simple. Debido a el microcontrolador tiene una ADC con un rango de 0v-3.33v, se solucionó el problema con un IC Lm358 que cuenta con dos Op-Amp de uso general, aprovechado también para asilar el sistema aceptablemente con el microcontrolador.

Para implementar un buffer (véase en el diagrama U2A) para aislar la muestra del voltaje que cae en C1 y así con un divisor de voltaje R1 y R7 que cumpla la proporción de 33/50, por lo que R7 debió ser calibrado para acercase lo más posible y en consecuencia escalar 0v-5v a 0-3.3v haciendo posible la lectura en el microcontrolador.

Mientras que, en la parte izquierda del circuito, U1A en configuración de amplificador no inversor, con valores de R4 y R6 no necesariamente obligatorios, pero sí intencionalmente usados para saturar a Q1 por acción de dicho Op-amp cada vez que el microcontrolador mande un 1 lógico en 3.3v, contrariamente si se recibe un 0 lógico en 0v, Q1 se corta.

En resumen, el circuito con el microcontrolador hace este proceso:

1. El microcontrolador manda un 1 lógico (Confirmando el estado con el LED) para comenzar el muestreo en C1 con un voltaje inicial aproximadamente a cero. R2 es despreciable con R5 en ambos casos, pero se colocó para delimitar la corriente pico que puede haber al momento de asegurarnos de descargar C1.

2. Después se manda un 0 lógico para cortar Q1 y empezar casi instantáneamente el muestreo del voltaje en C1 por parte del input analógico.
3. De nuevo el paso uno se ejecuta y se mantiene en ese estado indefinidamente hasta que se ordene al microcontrolador lo contrario.

```
from machine import Pin, ADC
import time

Des_cap = Pin(25, Pin.OUT, value=1) # activado para comenzar con
el capacitor descargado. Satura el transistor
adc = ADC(Pin(32))#pin de lectura adc
adc.atten(ADC.ATTN_11DB)#Atenuación de 11dB (150mV - 2450mV)


#Variables del sistema RC
Capacitor = 0.000100 #faradios
Resistor = 100000 #ohmios
t = 5*Capacitor*Resistor #Tiempo de carga para 99% aprox
muestreo = t/0.010 #en segundos

print("BD_int:")#Inicio de mediciones

time.sleep_ms(1000) # 1seg de tiempo más que suficiente para que
tenga aprox 0v
Des_cap.off()#Da comienzo a la carga del capacitor. Corta el
transistor

#Comienza muestreo
num=0 #Contador
while num<=muestreo:#Cuantos muestreos?

    val = adc.read_u16()*(3.3/65535)#lectura analogica
    time.sleep_ms(10)#Periodo del muestreo en ms
    print("volts"+str(val*(5/3.3)))#imprimimos lectura
    num += 1

print("BD_end")#Fin de mediciones

while True:
      Des_cap.on()#Descarga capacitor. Vuelve a saturar el
transistor

#Para detener el programa CTRL+C
```

En la izquierda se muestra detalladamente el código en micropython que se encontraba en el microcontrolador ESP32 DEVKIT V1. Los variables Capacitor y Resistor eran modificadas según el caso de R5 antes de procedes a correr el programa. Dicho programa se corría cada vez que el comando de línea desde un pc que se comunica seriamente a la tarjeta:

mpremote connect COM7 repl --capture D:\"doc.tx"

El cual aparte de ordenar desde la pc al microcontrolador de ejecutar su programa, este comando también hace la labor de registrar todo lo que se imprima en el REPL de micropython en un archivo .txt. Es entonces que la información (el muestreo de la tensión en C1) puede ser accedida para su tratamiento con mayor facilidad desde un pc.

```
import matplotlib.pyplot as plt
#Adquisición de datos por parte del txt
f = open("D:\RC_dos", "r")#Buscamos archivo txt existente o
creamos uno nuevo
Lines = f.readlines() #Leemos las lineas del documento

data = [] #Hacemos una lista en donde asignar cada medición a un
espacio
#Filtramos la información relevante
for l in Lines:
    if "volts" in l:
        data.append(float(l[5:].strip()))
f.close()

#Proceso de creación de gráficas requeridas

ypointsvc = data #Lecturas medidas del voltaje en el capacitor

#Varieables del circuit0
fuentedc = 5 #En volts
Resistor = 100000 #En ohms
Capacitor = 0.0001 #En faradios

#Conseguimos el voltaje que cae en el resistor
ypointsvr = []
for d in data:
    ypointsvr.append(fuentedc - d)

#Conseguimos la corriente de la malla en el tiempo
ypointsI = []
for d in ypointsvr:
    ypointsI.append(d/Resistor)

#Convertimos la cantidad de muestras en tiempo
xpoints = []
t = 0
for d in ypointsI:
    xpoints.append(t)
    t = t+0.010

#proceso para mostrar las graficas necesarias
plt.subplot(3, 1, 1)
plt.plot(xpoints, ypointsvc)
plt.ylabel('Vcapacitor')
plt.title('Voltages en R Y C')

plt.subplot(3, 1, 2)
plt.plot(xpoints, ypointsvr)
plt.ylabel('Vresistor')
```

Ahora, este es el programa en Python que se corría una vez que los datos fueran registrados en un archivo .txt, una ejecución por cada caso, pues se consiguieron dos archivos diferentes .txt.

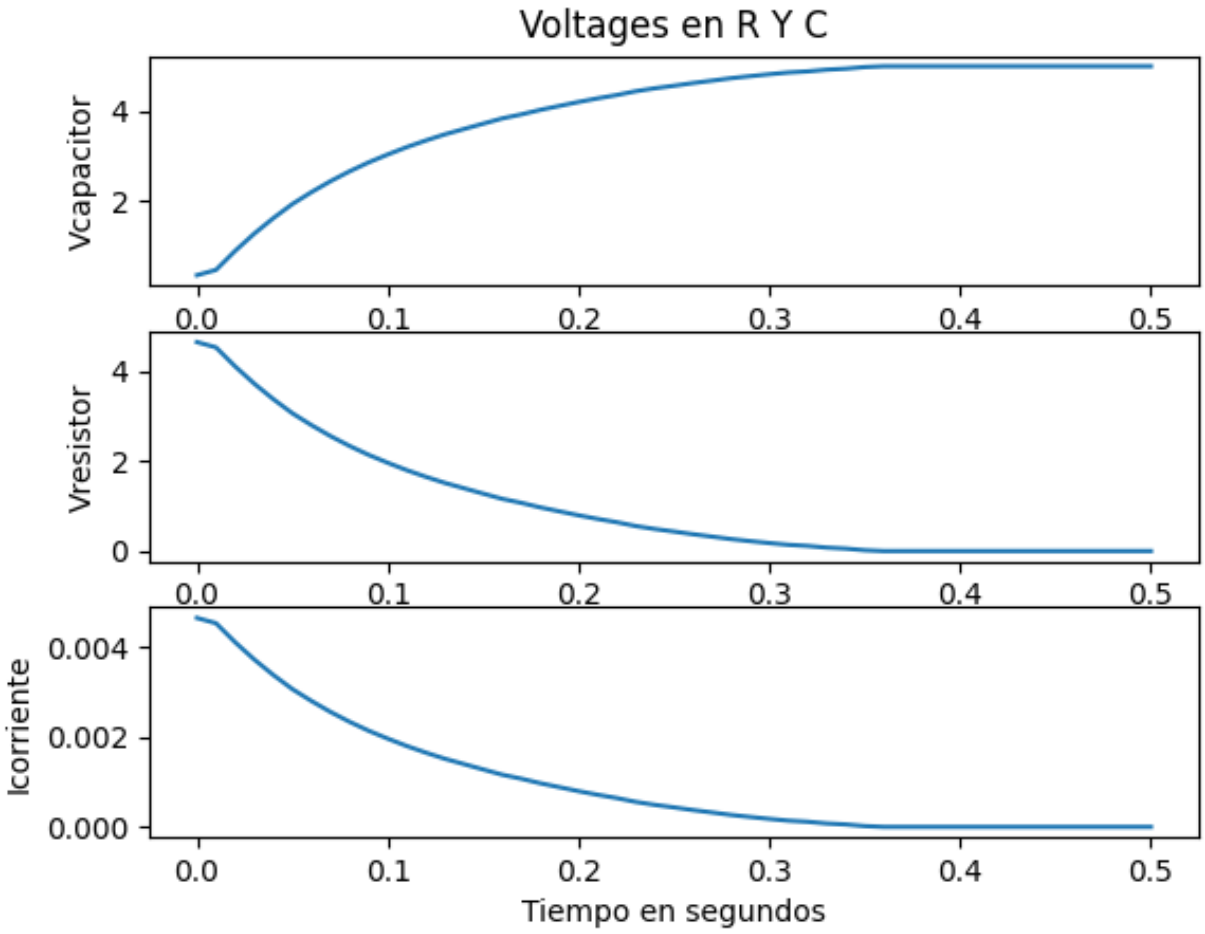
El programa se encarga de filtrar únicamente los voltajes que se muestrearon en C1 y graficar las funciones: $v_{\mathbb{C}}(t),\ v_R(t),\ y\ i(t)$

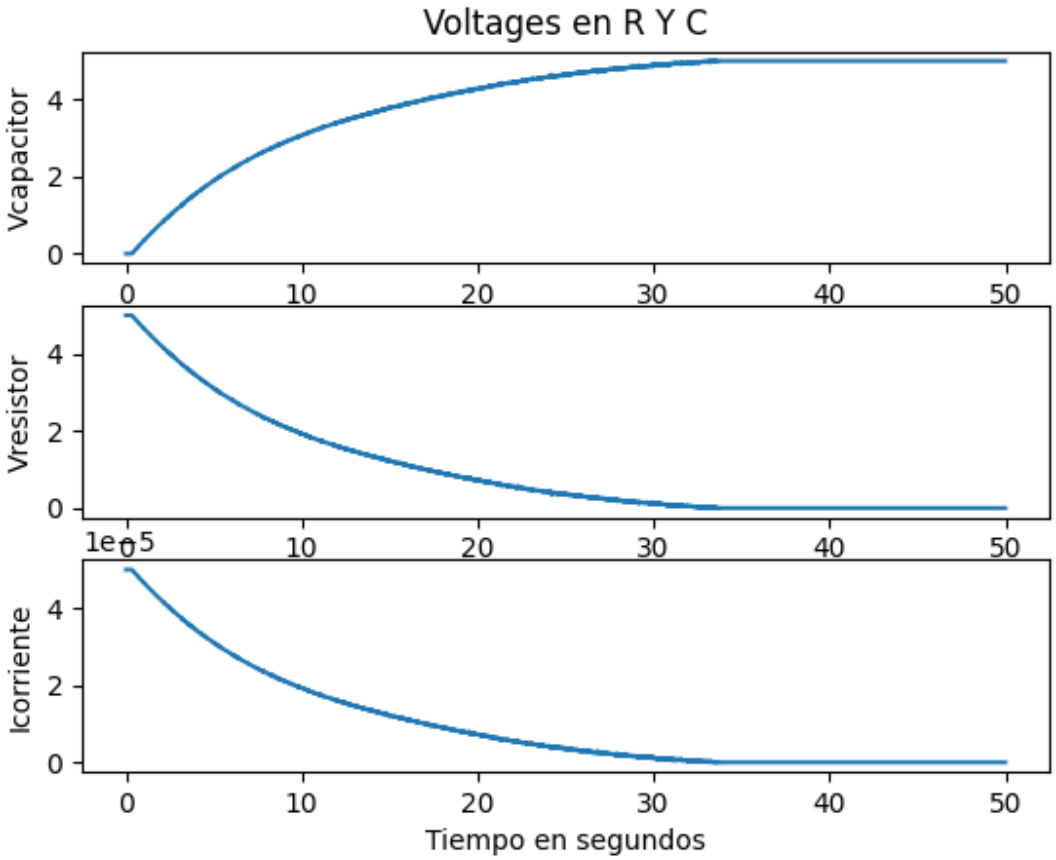
De nuevo, como pasó en el programa de micropython, las variables Resistor, Capacitor, y el nombre del archivo de interés, se modificaron según el caso antes de ejecutar el script.

```
plt.subplot(3, 1, 3)
plt.plot(xpoints, ypointsI)
plt.ylabel('Icorriente')
plt.xlabel('Tiempo en segundos')

plt.show()
```

Estas son las curvas que se consiguieron en el caso uno (R5=1KΩ), y caso 2 (R5=100KΩ), respectivamente. Comprobando la teoría.





Bibliografías:

http://webpersonal.uma.es/~jmpeula/carga_y_descarga.html

https://docs.micropython.org/en/latest/esp32/quickref.html

https://micropython.org/discord