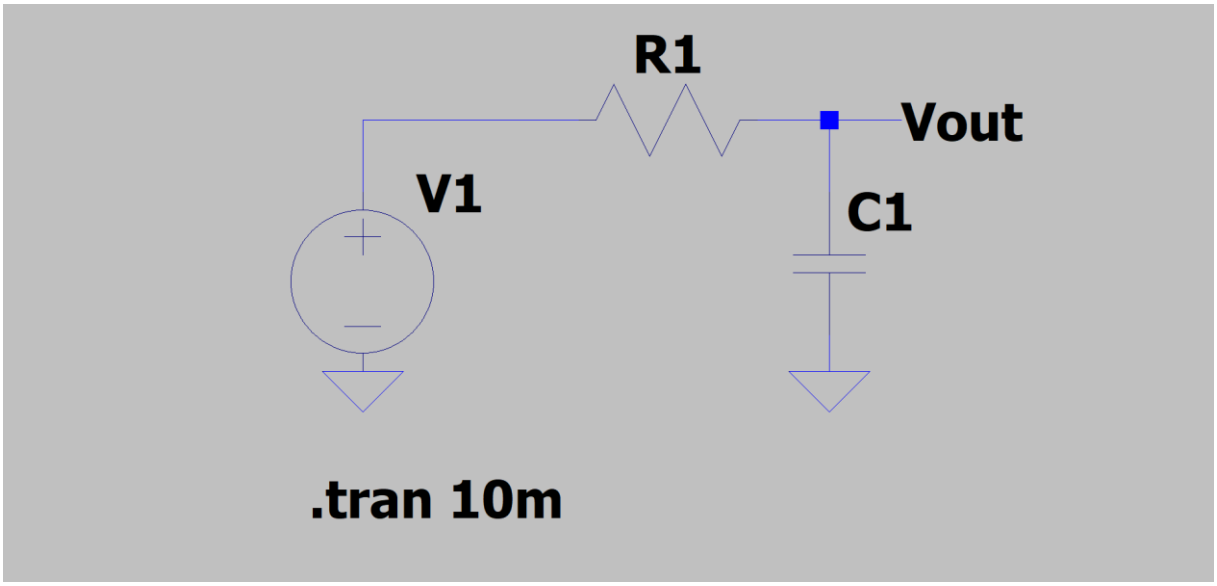
Andreea NICU
321AB

Tema 1

N – ASCII Code: 78
Nr = Cod_{ascii} % 4
Nr = 78 % 4
Nr = 2 => Circuit RC serie cu intrare pe rezistență, ieșire pe condensator legat la masă.



README

Limbaj de programare: Python
Biblioteci utilizate:

- NumPy

Utilizat pentru calcul științific, acesta permite să lucrăm cu tablouri multidimensionale. Implementarea Arrays nu există în Python. În principal, utilizatorii dezvoltă NumPy în proiectele lor de învățare automată.

- Matplotlib

Matplotlib este o bibliotecă de grafic 2D si se pot vizualiza datele. Se pot genera imagini ale figurilor în diferite formate.

- Math

Python are un set de funcții matematice încorporate, inclusiv un modul matematic extins, care permite efectuarea unor sarcini matematice pe numere.
math.e: Returnează numărul lui Euler (2.7182...)

Descrierea soluției implementate

Voi simula și analiza comportamentul circuitului pentru un semnal de comandă dreptunghiular ce comută între 0V și 5V. Intrarea este un semnal pătratic cu perioada 2T, T = 1ms.

- ⇨ v_in = 0;
- ⇨ v_out = 5; (axa oY)
- ⇨ T = 1;
- ⇨ time = 10; (axa oX – durata simulării)

Andreea NICU
321AB

În explicarea algoritmului de lucru, voi scrie echivalentul principalelor linii de cod în limbajul C.
Se citesc de la tastatură valorile rezistenței R și ale condensatorului C.

```
R = float(input("R = "))                    # scanf("%f", &R);
C = float(input("C = "))                    # scanf("%f", &C);
```

Tau – constanta de timp: reprezintă valoarea produsului dintre rezistență si condensator: tau = R * C
Valorile condensatorului și ale rezistenței vor fi alese astfel încat să se observe 2 cazuri distincte:
$3\tau < T$ și $3\tau > T$

```
if 3 * tau < T:
   print ("Cazul 3τ < T")
else:
   print ("Cazul 3τ > T")
```

```
if (3 * tau < T)
   printf ("Cazul 3τ < T\n");
else
   printf ("Cazul 3τ < T\n");
```

<u>Algoritm pentru a genera graficul semnalului de intrare: Vi</u>

Se declară lista x_in, având valori de la 0 la 20 (2 * 10) din 2 în 2.

```
x_in = np.arange(0, time * 2, T * 2)
```

```
int k = 0;
for (int i = 0; i < time * 2; i += T * 2)
  x_in[k++] = i;
```

Se declară lista y_in, iar variabila step numară pașii. Pe pași impari este reprezentat Toff, iar pe pași impari va fi reprezentat Ton.

```
y_in = []
step = 0
for i in range(0, time * 2, T * 2):
   if step % 2 == 0:
      y_in.append(v_out)
   elif step % 2 == 1:
      y_in.append(v_in)
   step += 1
```

```
int k = 0;
int step = 0;
for (int i = 0; i < time * 2;
          i += T * 2)
{
if (step % 2 == 0)
          y[k++] = v_out;
else if (step % 2 != 0)
          y[k++] = v_in;
step++;
}
```

Se generează graficul pentru semnalul de intrare.

```
figin = plt.figure()                        # creează figura pt semnal de intrare
ax = figin.add_subplot(111)                 # creează un grafic și îl pune pe figin
ax.step(x_in, y_in)                         # pune punctele pe grafic
ax.set_xlabel('Timp')                       # numele axei oX
ax.set_ylabel('V_in')                       # numele axei oY
ax.set_xlim((0, time))                      # setează zoom-ul pt oX
ax.set_title('Semnal de intrare cu R = {} si C = {}'.format(R, C))
# printf("Semnal de intrare cu R = %f si C = %f", R, C)
```

Andreea NICU
321AB

Algoritm pentru a genera graficul semnalului de ieşire: Vout

Frontul crescător al pulsului: $\mathrm{Vo(t)} = \mathrm{E}(1 - \mathrm{e}^{-\frac{t}{\tau}})$

Frontul descrescător al pulsului: $\mathrm{Vo(t)} = \mathrm{E} * \mathrm{e}^{-\frac{t}{\tau}}$

Se creează 2 liste pentru reprezentarea graficului: x_in, coordonatele x ale punctelor, pornesc din 0 până la 20 (2 * 10) din 0.1 in 0.1 (valoare aleasă pentru o bună ilustrare a graficului). Valorile din y_in, coordonatele y ale punctelor, sunt calculate cu ajutorul formulelor de calcul specifice atât frontului crescător, cât și al celui descrescător, utilizate cu ajutorul variabilei front $\in \{\pm 1\}$. While-ul numară timpul și crește până când ajunge la timpul țintă, apoi crește t-ul, variabila care reprezintă timpul numărat pe perioadă. La final, când t devine mai mare decât p (când t trece de perioada p), frontul se schimbă, iar parioada se resetează, deci t = 0, astfel încat funcția să o ia de la capăt. Din formula frontului descrescător, scad first_y_desc = vârful funcției descrescătoare și last_y_asc = ultimul y generat crescător, pentru a începe următorul front din punctul în care cel crescător s-a oprit.

```
x_in = np.arange(0, time * 2, 0.1)                # facem x din nou cu arange
y_in = []                                         # resetăm y
front = 1                                         # 1 pentru front cresc si -1 pentru front descr
p = T * 2                                         # perioada
t = 0                                             # timpul numărat pe perioadă
t_total = 0                                       # timpul total de până acum
first_y_desc = v_out                              # primul y generat descrescător de fiecare dată
last_y_asc = v_out * (1 - math.e ** (- p / tau))  # ultimul y generat crescător
while t_total < time * 2:
   if front == 1:
      y_in.append(v_out * (1 - math.e ** (- t / tau)))
   elif front == -1:
      y_in.append(v_out * (math.e ** (- t / tau)) - (first_y_desc - last_y_asc))
   t += 0.1
   t_total += 0.1
   if t > p:
      front *= -1
      t = 0                                       # resetez perioada
```

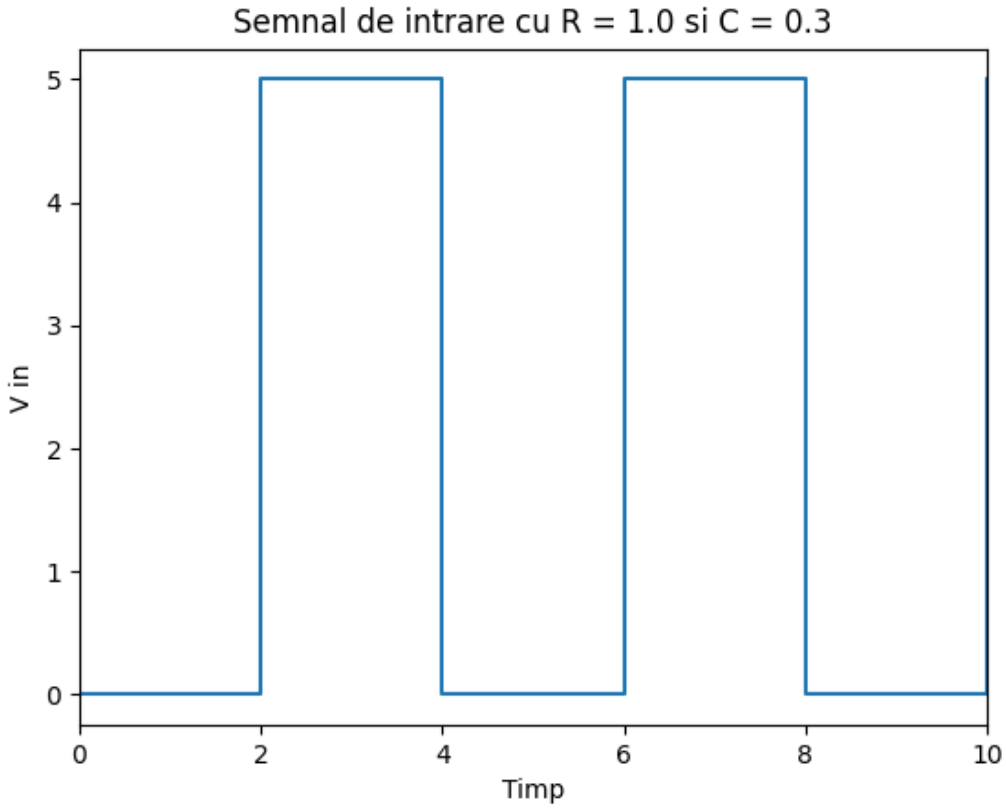
Se genereaza graficul pentru semnalul de iesire.

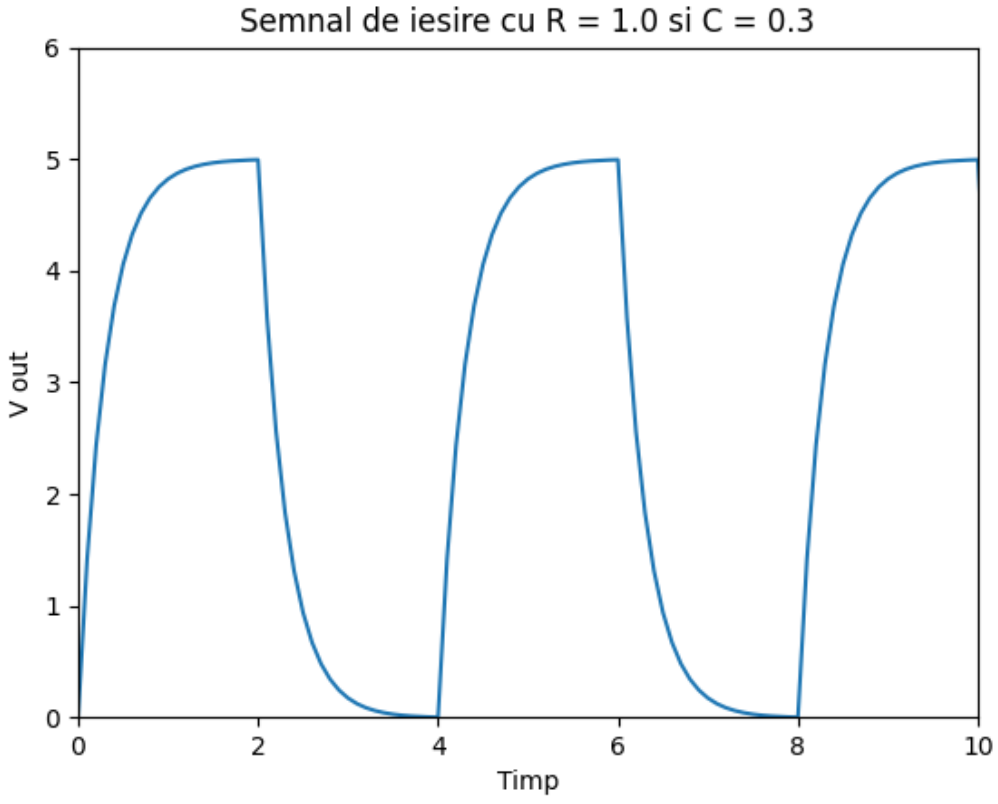
```
figout = plt.figure()                             # creează figura pt semnal de ieșire
bx = figout.add_subplot(111)                      # creează un grafic și îl pune pe figin
bx.plot(x_in, y_in)                               # pune punctele pe grafic
bx.set_xlabel('Timp')                             # numele axei oX
bx.set_ylabel('V_out')                            # numele axei oY
bx.set_xlim((0, time))                            # setează zoom-ul pt oX
bx.set_ylim((0, v_out + 1))                       # setează zoom-ul pt oY
bx.set_title('Semnal de iesire cu R = {} si C = {}'.format(R, C))
# se salvează figurile ca png
figin.savefig("semnal_intrare_r={}_c={}.png".format(R, C))
figout.savefig("semnal_iesire_r={}_c={}.png".format(R, C))
plt.show()                                        # arată graficele
```

Andreea NICU
321AB

Testare soluție

Cazul $3\tau < T$





Andreea NICU
321AB

Cazul $3\tau > T$