

Instituto Politécnico Nacional

Escuela Superior de Cómputo

Teoría Computacional

Reporte primer bloque de programas

Alumno: Javier Said Naranjo Miranda

Grupo: 2CM4

Índice

1. Alfabeto	2
1.1. Descripción del programa	2
1.2. Código	2
1.3. Pruebas	5
2. Números primos	8
2.1. Descripción	8
2.2. Código	8
2.3. Pruebas	12
3. Palabras terminadas con ere(Autómata)	15
3.1. Descripción	15
3.2. Código	15
3.3. Pruebas	21
4. Autómata Paridad	23
4.1. Descripción	23
4.2. Código	23
4.3. Pruebas	29
5. Protocolo	31
5.1. Descripción	31
5.2. Código	31
5.3. Pruebas	36
6. Cadenas terminadas en 01	40
6.1. Descripción	40
6.2. código	40
6.3. Pruebas	41

1. Alfabeto

1.1. Descripción del programa

El siguiente programa dado un alfabeto binario $\Sigma = \{0, 1\}$ calcula todas las palabras posibles que pueden ser formadas a partir de dicho alfabeto.

Es decir $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \dots \cup \Sigma^n$.

Este programa calculara a partir de una longitud de $0 \leq n \leq 1000$. EL programa tendrá dos modos, el manual y el automático, en el primero el usuario elegirá la potencia a la cual quiere hacer las cadenas y si desea hacer un conteo nuevo, mientras que en el segundo se generara un numero aleatorio y de igual forma la opción de nuevo conteo será aleatoria. Cada una de las palabras se guardara en un archivo de texto en ambos casos.

1.2. Código

EL lenguaje utilizado para la resolución del problema fue python.
El código utilizado para la resolución del problema se muestra a continuación:

Código: main.py

```
import cadenas

cadenas.iniciar_archivo()

opcion = cadenas.menu()

if (opcion == 1):
    #Modo manual
    while True:
        tamano = cadenas.tamano()
        cadenas.iniciar_programa(tamano)
        while True:
            op = cadenas.nuevo_conteo()
            if (op == 1):
                break
            elif (op == 2):
                exit()
            else:
                continue

elif (opcion == 2):
    #Modo automatico
    while True:
        num = cadenas.num_aleatorio()
        print (num)
```

```

        cadenas.iniciar_programa(num)
    while True:
        opa=cadenas.opcion_aleatoria()
        print("1.-Nuevo_conteo\n2.-salir", opa)

        if (opa == 1):
            break
        elif (opa == 2):
            exit()
        else:
            continue

    else:

        print("Introduce_una_opcion_valida")
        exit()

```

Código:cadenas.py

```

import random
def menu():
    try:
        opcion = input("1.-Modo_Manual\n2.-Modo_Automatico_")
        opcion = int(opcion)
        return opcion
    except:
        print("Introduce_una_opcion_valida")
def nuevo_conteo():
    try:
        conteo = input("1.-Nuevo_conteo\n2.-salir_")
        conteo = int(conteo)
        return conteo
    except:
        print("opcion_invalida")

def num_aleatorio():
    numero = random.randrange(1000)
    return numero
def opcion_aleatoria():
    opcion = random.randrange(1,3)
    return opcion

def iniciar_archivo():
    try:
        archivo = open("cadenas.txt","w")
        archivo.close()
    except:
        exit()

```

```

def tamaño():
    try:
        tam = input("Introduce el tamaño de la cadena:")
        tam = int(tam)
        return tam
    except:
        print("No es un número")
        exit()

def iniciar_programa(tamaño):
    cadena=['0','1']
    cad_aux=[]
    cont=0
    continuar = True

    archivo = open("cadenas.txt","a")
    archivo.write("S={}_e")

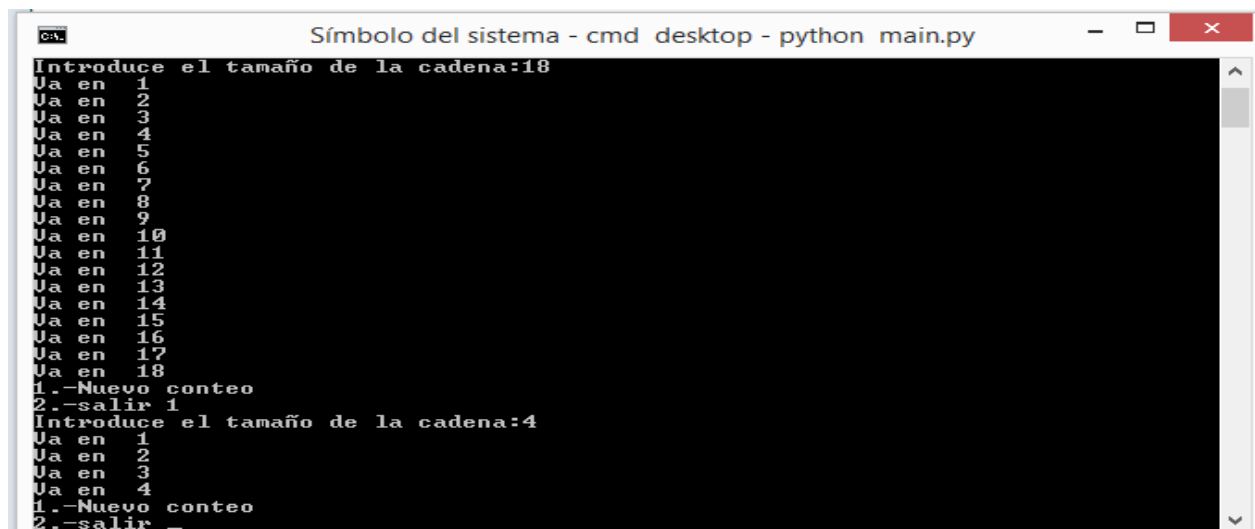
    for indice in range(1,tamaño+1):
        print("Va en",indice)
        cad_aux = [0] * indice

        while continuar:
            archivo.write(",_")

            for i in range(indice):
                archivo.write(cadena[cad_aux[i]])
            cont = 0
            while (cont < indice):
                cad_aux[cont]=cad_aux[cont] + 1
                if (cad_aux[cont] > 1):
                    cad_aux[cont] = 0
                else:
                    break
            cont = cont +1
            if (cont>=indice):
                cad_aux=[]
                break

    archivo.write("_}\n")
    archivo.close()

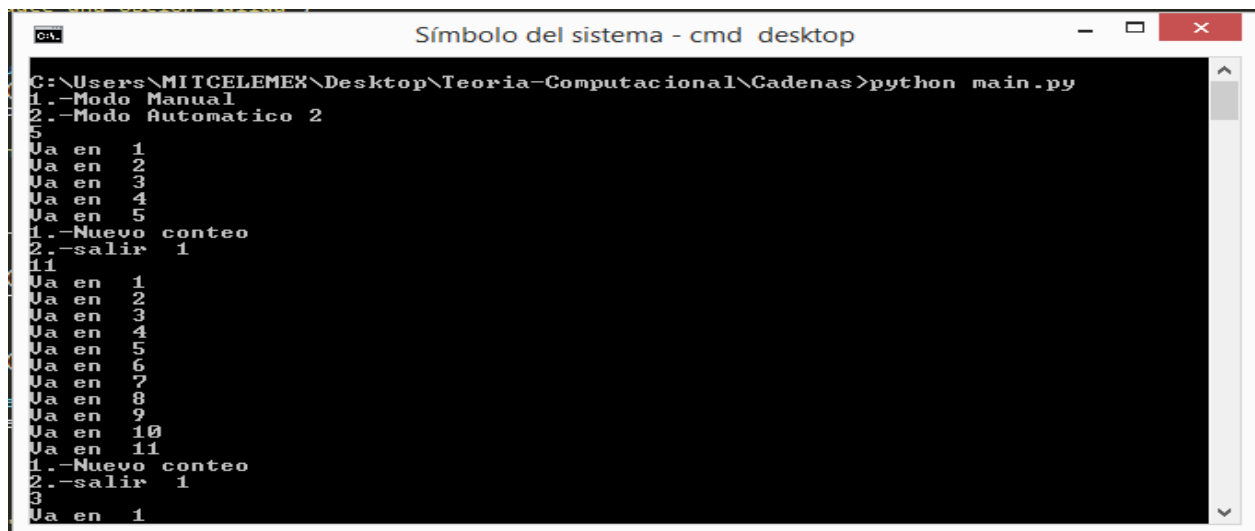
```

```
Símbolo del sistema - cmd desktop - python main.py
Introduce el tamaño de la cadena:18
Va en 1
Va en 2
Va en 3
Va en 4
Va en 5
Va en 6
Va en 7
Va en 8
Va en 9
Va en 10
Va en 11
Va en 12
Va en 13
Va en 14
Va en 15
Va en 16
Va en 17
Va en 18
1.-Nuevo conteo
2.-salir 1
Introduce el tamaño de la cadena:4
Va en 1
Va en 2
Va en 3
Va en 4
1.-Nuevo conteo
2.-salir
```

Figura 3: Nuevo conteo de tamaño 4

Para el modo automático:



```
Símbolo del sistema - cmd desktop
C:\Users\MITCELEME\ Desktop\Teoria-Computacional\Cadenas>python main.py
1.-Modo Manual
2.-Modo Automatico 2
5
Va en 1
Va en 2
Va en 3
Va en 4
Va en 5
1.-Nuevo conteo
2.-salir 1
11
Va en 1
Va en 2
Va en 3
Va en 4
Va en 5
Va en 6
Va en 7
Va en 8
Va en 9
Va en 10
Va en 11
1.-Nuevo conteo
2.-salir 1
3
Va en 1
```

Figura 4: Selección del modo automático con un tamaño de 5 y posteriormente un nuevo conteo con un tamaño de 11



Figura 5: Parte del texto de las cadenas obtenidas por el modo automático

2. Números primos

2.1. Descripción

El siguiente programa calcula todos los números primos que se encuentran por debajo del límite que se establezca, cabe resaltar que un número primo es aquel que solo puede ser dividido entre si mismo y entre 1. Actualmente no existe un método directo para calcular todos los números primos

El programa hará un calculo que tendrá como un límite máximo el numero 1000 El programa tendrá dos modos, el manual y el automático, en el primero el usuario elegirá el numero máximo y si desea calcular los números de nuevo, mientras que en el manual todo se hará por medio de numero aleatorios. En ambos casos se guardara en un archivo los números primos encontrados así como su conversión a número binario.

2.2. Código

EL lenguaje utilizado para la resolución del problema fue python.

El código utilizado para la resolución del problema se muestra a continuación:

Código: mainNP.py

```
import primos
primos.iniciar_archivo()

opcion = primos.menu()

if (opcion == 1):
    #Modo manual
    while True:
        numero = primos.numero()
        primos.iniciar_programa(numero)

        while True:
            op = primos.nuevo_conteo()
            if (op == 1):
                break
            elif (op == 2):
                exit()
            else:
                continue

# modo automatico
elif (opcion == 2):
    while True:
        numero = primos.num_aleatorio()
        primos.iniciar_programa(numero)
        while True:
```

```

op = primos.opcion_aleatoria()
print ("1.-Nuevo_conteo\n2.- salir_")
if (op == 1):
    break
elif (op==2):
    exit()
else:
    continue

```

Código:primos.py

```

import random

def menu():
    try:

        opcion = input ("1.-Modo_Manual\n2.-Modo_Automatico_")
        opcion = int(opcion)
        return opcion

    except:
        print ("Introduce_una_opcion_valida")
def nuevo_conteo():
    try:
        conteo = input ("1.-Nuevo_conteo\n2.- salir_")
        conteo = int(conteo)
        return conteo

    except:
        print ("opcion_invalida")

def num_aleatorio():
    numero = random.randrange(1,1001)
    return numero
def opcion_aleatoria():
    opcion = random.randrange(1,3)
    return opcion
def iniciar_archivo():
    try:
        archivo = open ("numprimo.txt","w")
        archivo.close

    except:
        exit()
def numero():
    try:
        num = input ("Introduce_un_numero:_")
        num = int(num)
        return num

    except:
        print ("No_es_un_numero")

```

```

        exit ()

def iniciar_programa (numero) :
    archivo = open ("numprimo.txt", "a")

    primos = [2,3,5,7,11]
    binario = []
    obtener_primos (primos, binario, archivo, numero)
    escribir_archivo_binario (archivo, binario)
    print ("numero_de_ceros")
    contar_ceros (binario)
    print ("numero_de_unos")
    contar_unos (binario)
    archivo.close ()

def obtener_primos (primos, binario, archivo, numero) :
    archivo.write ("{}_")
    if (numero >= 2):
        for numero in range (2, numero+1):
            cont=0
            for valor in primos:
                if (numero == valor):
                    archivo.write (str (numero)+",_")
                    binario.append (bin (numero))
                    break
            else:
                mod = numero % valor
                if (mod == 0):
                    break
                else:
                    if (cont < len (primos)) :
                        cont = cont +1
                        if (cont == (len (
                            primos) -1)) :
                            primos.append (
                                numero)
                            archivo.write (
                                str (numero)
                            )
                            archivo.write (
                                ",_"
                            )
                            binario.append
                                (bin (numero)
                                )
                            break
                    else:
                        continue

    archivo.write ("_}\n")

else:
    print ("no_tiene_numeros_primos")

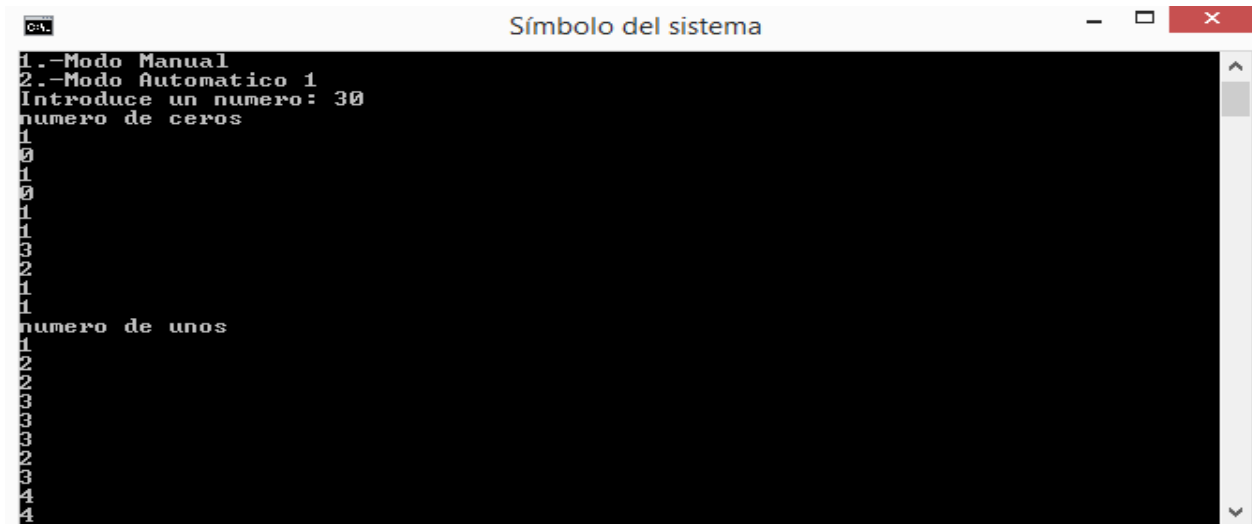
```

```
def escribir_archivo_binario(archivo, binario):  
    archivo.write("{_")  
    for num_bin in binario:  
        archivo.write(str(num_bin)+",")  
    archivo.write("}\n")  
  
def contar_ceros(binario):  
    for bin in binario:  
        bin = str(bin)  
        print((bin.count("0"))-1)  
  
def contar_unos(binario):  
    for bin in binario:  
        bin = str(bin)  
        print(bin.count("1"))
```

2.3. Pruebas

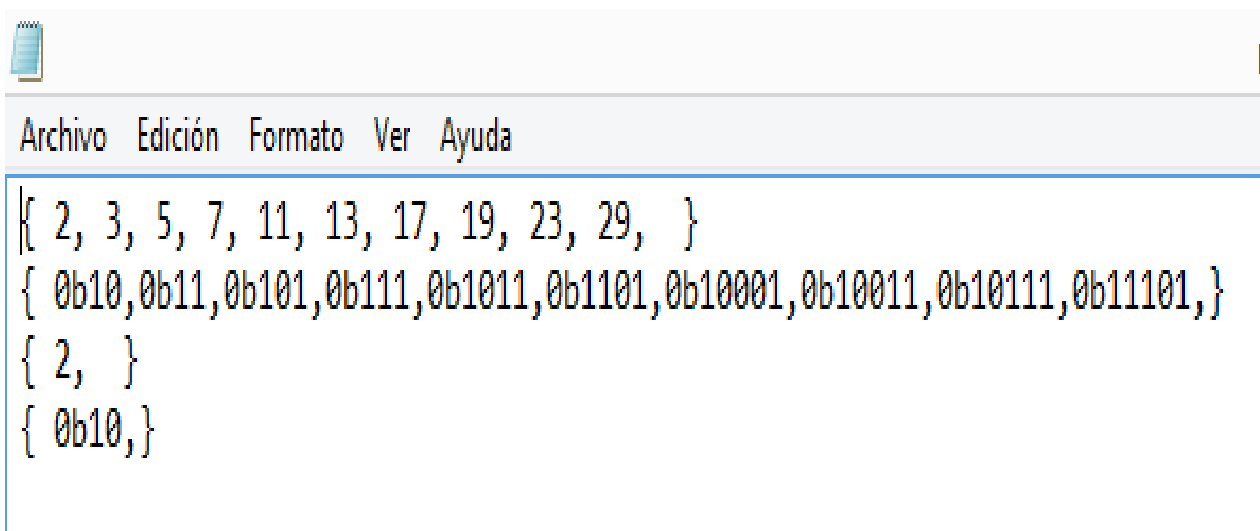
A continuación se mostraran algunas imágenes capturadas al momento de ejecutar el programa, dichas imágenes mostraran los resultados obtenidos.

Para el modo manual:



```
1.-Modo Manual
2.-Modo Automatico 1
Introduce un numero: 30
numero de ceros
1
0
1
0
1
1
1
3
2
1
1
1
2
2
3
3
3
2
3
4
4
numero de unos
1
2
2
3
3
3
2
3
4
4
```


Figura 6: Selección de un número limite de 30



```
{ 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, }
{ 0b10,0b11,0b101,0b111,0b1011,0b1101,0b10001,0b10011,0b10111,0b11101,}
{ 2, }
{ 0b10,}
```

Figura 7: Salida en archivo de texto del modo manual

Para el modo Automático:



```
C:\Users\MITCELEMEX\Desktop\Teoria-Computacional\NumPrimo>python NPmain.py
1.-Modo Manual
2.-Modo Automatico 2
883
numero de ceros
numero de unos
1.-Nuevo conteo
2.-salir
295
numero de ceros
numero de unos
1.-Nuevo conteo
2.-salir
165
numero de ceros
numero de unos
1.-Nuevo conteo
2.-salir
880
numero de ceros
numero de unos
1.-Nuevo conteo
2.-salir
301
numero de ceros
```

Figura 8: Ejecución del modo automático

numprimo.txt: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
{ 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167,
{ 0b10,0b11,0b101,0b111,0b1011,0b1101,0b10001,0b10011,0b10111,0b11101,0b11111,0b100101,0b101001,0b101011,0b101111,0b110101,0b111011,0b111101,0b1000011,0b1000111,0b1001
111,0b11111101,0b1000001001,0b1000001011,0b1000011101,0b1000100011,0b1000101101,0b1000110011,0b1000111001,0b1000111011,0b100100001,0b1001001011,0b1001010001,0b100101
{ 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167,
{ 0b10,0b11,0b101,0b111,0b1011,0b1101,0b10001,0b10011,0b10111,0b11101,0b11111,0b100101,0b101001,0b101011,0b101111,0b110101,0b111011,0b111101,0b1000011,0b1000111,0b1001
{ 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, }
{ 0b10,0b11,0b101,0b111,0b1011,0b1101,0b1101,0b10001,0b10011,0b10111,0b11101,0b11111,0b100101,0b101001,0b101011,0b101111,0b110101,0b111011,0b111101,0b1000011,0b1000111,0b1001
{ 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167,
{ 0b10,0b11,0b101,0b111,0b1011,0b1101,0b10001,0b10011,0b10111,0b11101,0b11111,0b100101,0b101001,0b101011,0b101111,0b110101,0b111011,0b111101,0b1000011,0b1000111,0b1001
111,0b11111101,0b1000001001,0b1000001011,0b1000011101,0b1000100011,0b1000101101,0b1000110011,0b1000111001,0b1000111011,0b100100001,0b1001001011,0b1001010001,0b100101
{ 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167,
{ 0b10,0b11,0b101,0b111,0b1011,0b1101,0b1101,0b10001,0b10011,0b10111,0b11101,0b11111,0b100101,0b101001,0b101011,0b101111,0b110101,0b111011,0b111101,0b1000011,0b1000111,0b1001
```

Figura 9: Parte de la salida de texto del modo automático

Se realizo una gráfica en la cual se puede visualizar la relacion entre 1's y 0's de cada número primo.

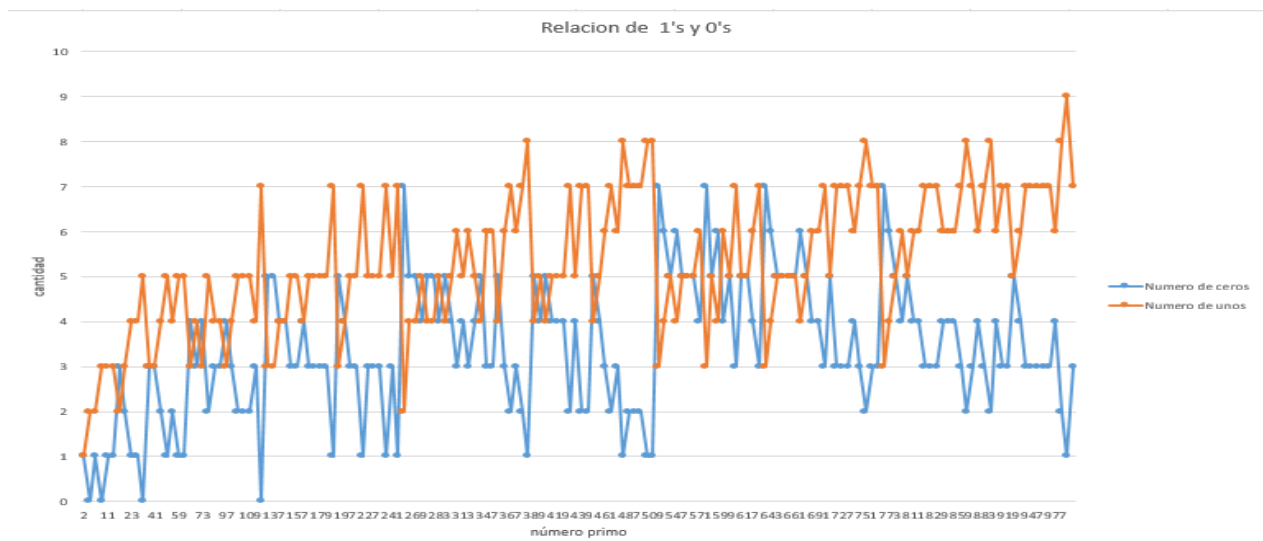


Figura 10: Gráfica de la relación entre 1's y 0's de cada número primo.

3. Palabras terminadas con ere(Autómata)

3.1. Descripción

El siguiente autómata reconocerá todas las palabras terminadas con ere, este autómata podrá reconocer las palabras terminadas con ere de textos en inglés y en español, de este ultimo siempre y cuando no tengan acentos.

El autómata tendrá un modo para introducir un texto de manera manual y otro para leer un archivo, este devolverá en pantalla las palabras encontradas así como en que posición se encuentran.

El autómata utilizado para modelar el problemas es el siguiente:

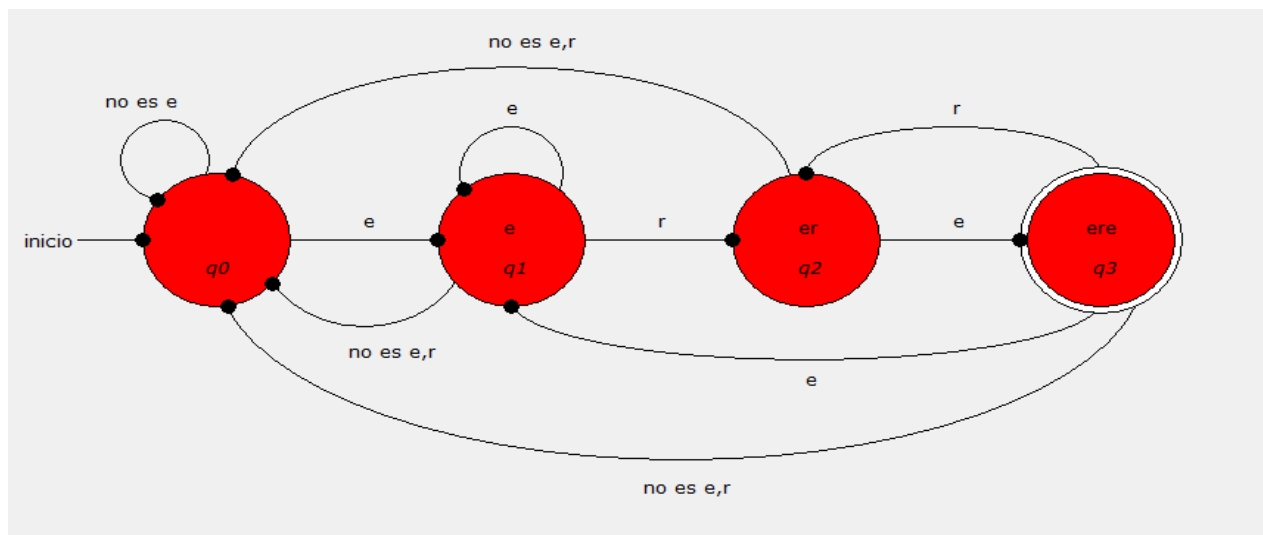


Figura 11: Autómata.

3.2. Código

Se mostrara el código del gráfico del autómata
Código:Diagrama1.py

```
from tkinter import *

def crearDiagrama():
    ventana = Tk()
    ventana.geometry("900x500") #geometry(widthxheight)
    ventana.title("Automata_ERE")
    ventana.resizable(width=False, height=False)
    AreaDibujo=Canvas(ventana, width=890, height=490)
    AreaDibujo.pack()

    #Creacion de arcos para transicion
```



```

AreaDibujo.create_arc(315,115,385,185,start=-25,extent=250)
AreaDibujo.create_arc(550,115,750,185,start=8,extent=180,style=ARC)
AreaDibujo.create_arc(350,200,750,290,start=180,extent=168,style=ARC)
AreaDibujo.create_arc(180,140,320,265,start=180,extent=168,style=ARC)
AreaDibujo.create_arc(160,70,540,250,start=-20,extent=210,style=ARC)
AreaDibujo.create_oval(85,110,145,170)
AreaDibujo.create_arc(150,75,780,365,start=180,extent=180,style=ARC)

```

#Creacion de Circulos de estado

```

AreaDibujo.create_oval(695,145,805,255,fill="white")
AreaDibujo.create_oval(100,150,200,250,fill="red")
AreaDibujo.create_oval(300,150,400,250,fill="red")
AreaDibujo.create_oval(500,150,600,250,fill="red")
AreaDibujo.create_oval(700,150,800,250,fill="red")

```

#Creacion de circulo de referencia

```

AreaDibujo.create_oval(95,195,105,205,fill="black")
AreaDibujo.create_oval(295,195,305,205,fill="black")
AreaDibujo.create_oval(495,195,505,205,fill="black")
AreaDibujo.create_oval(690,195,700,205,fill="black")
AreaDibujo.create_oval(313,157,323,167,fill="black")
AreaDibujo.create_oval(545,145,555,155,fill="black")
AreaDibujo.create_oval(345,245,355,255,fill="black")
AreaDibujo.create_oval(105,165,115,175,fill="black")
AreaDibujo.create_oval(156,146,166,156,fill="black")
AreaDibujo.create_oval(183,228,193,238,fill="black")
AreaDibujo.create_oval(153,245,163,255,fill="black")

```

#Creacion de lineas para transicion

```

AreaDibujo.create_line(50,200,100,200)
AreaDibujo.create_line(200,200,300,200)
AreaDibujo.create_line(400,200,500,200)
AreaDibujo.create_line(600,200,700,200)

```

#Creacion de etiquetas

#estados

```

inicio=Label(ventana,text="inicio",font="Verdana_10_roman").place(x
    =20,y=190)
qo=Label(ventana,text="q0",background="red",font="Verdana_10_italic").
    place(x=143,y=210)
q1=Label(ventana,text="q1",background="red",font="Verdana_10_italic").
    place(x=345,y=210)
q2=Label(ventana,text="q2",background="red",font="Verdana_10_italic").
    place(x=545,y=210)
q3=Label(ventana,text="q3",background="red",font="Verdana_10_italic").

```

```

        place (x=745,y=210)

qoe=Label(ventana , text="e" ,background="red" ,font="Verdana_10_roman") .
    place (x=345,y=180)
qo=Label(ventana , text="er" ,background="red" ,font="Verdana_10_roman") .
    place (x=545,y=180)
qo=Label(ventana , text="ere" ,background="red" ,font="Verdana_10_roman") .
    place (x=740,y=180)

#transiciones
letra_e=Label(ventana , text="e" ,font="Verdana_10_roman") . place (x=250,y
    =175)
letra_r=Label(ventana , text="r" ,font="Verdana_10_roman") . place (x=450,y
    =175)
letra_eFinal=Label(ventana , text="e" ,font="Verdana_10_roman") . place (x
    =650,y=175)
letra_eQ0=Label(ventana , text="e" ,font="Verdana_10_roman") . place (x=347,
    y=90)
letra_rQ3Q2=Label(ventana , text="r" ,font="Verdana_10_roman") . place (x
    =650,y=90)
letra_eQ3Q1=Label(ventana , text="e" ,font="Verdana_10_roman") . place (x
    =550,y=294)
noes_e=Label(ventana , text="no_es_e" ,font="Verdana_10_roman") . place (x
    =75,y=85)
noes_er=Label(ventana , text="no_es_e , r" ,font="Verdana_10_roman") . place (
    x=440,y=375)
noes_er=Label(ventana , text="no_es_e , r" ,font="Verdana_10_roman") . place (
    x=335,y=40)
noes_er=Label(ventana , text="no_es_e , r" ,font="Verdana_10_roman") . place (
    x=240,y=273)

ventana . mainloop ()

```

Código:main.py

```

import automata
import funciones
import Diagrama1

while True:
    fila=[]
    palabra=[]
    try:
        opcion=input( '1.- Introducir_Texto\n2.- Leer_un_texto\n3.-
            Grafico_del_automata\nElige_una_opcion:_ ' )
        opcion=int(opcion)
    except:
        print( "Introduce_una_opcion_valida" )

```

```

continue

if (opcion==1):
    texto=funciones.texto()
    palabras = automata.automata_ere(texto, fila, palabra)
    funciones.imprimir_palabras_ere(palabras, fila, palabra)
    funciones.nuevo_texto()

elif(opcion==2):
    texto=funciones.leer_archivo()
    palabras = automata.automata_ere(texto, fila, palabra)
    funciones.imprimir_palabras_ere(palabras, fila, palabra)
    funciones.nuevo_texto()

elif(opcion==3):
    Diagrama1.crearDiagrama()

else:
    print( 'Introduce_una_opcion_valida ')
    continue

```

Código:funciones.py

```

import tkinter as tk
def texto():
    texto=input( 'Introduce_un_texto\n')
    return texto
def nuevo_texto():
    while True:
        try:
            seguir=input( '1.- Si\n2.- No\nDesea_iniciar_el_
                        automata_de_nuevo:_ ')
            seguir=int(seguir)
        except:
            print( 'Introduce_una_opcion_valida ')
            continue
        if(seguir==1):
            break
        elif(seguir==2):
            exit()
        else:
            continue

def imprimir_palabras_ere(palabras, fila, palabra):
    i=0
    for pala in palabras:
        fil=fila[i]
        fil=str(fil)
        pal=palabra[i]
        pal=str(pal)

```

```

        print("Palabra:_" + pala+"_fila:_" + fil+"_palabra_numero:_" + pal
        )
        i=i+1

def leer_archivo():
    archivo=open("archivo.txt","r")
    lineas=str(archivo.read())
    archivo.close()
    return lineas

```

Código:automata.py

```

def automata_ere(texto, fila, palabra):
    estado=0
    palabras_ere=[]
    auxiliar_palabra = ''
    contfilas=1
    contpalabra=1

    for caracter in texto:

        aux_caracter = caracter.lower()

        if (estado ==0):
            estado = estadoCero(aux_caracter)
        elif(estado==1):
            estado= estadoUno(aux_caracter)
        elif(estado==2):
            estado= estadoDos(aux_caracter)
        elif(estado==3):
            estado= estadoTres(aux_caracter)

        if(ord(aux_caracter)>=97 and ord(aux_caracter)<=122):
            auxiliar_palabra = auxiliar_palabra + aux_caracter
            if (estado==-1):
                estado =0
                auxiliar_palabra=''
        else:
            if(estado ==-1):
                palabras_ere.append(auxiliar_palabra)
                fila.append(contfilas)
                palabra.append(contpalabra)
                auxiliar_palabra=''
                estado=0
            else:
                estado=0
                auxiliar_palabra = ''
        if(caracter=="_"):
            contpalabra=contpalabra+1
        if(caracter=="\n"):

```

```

        contfilas=contfilas+1
        contrpalabra=1
    if(estado==3):
        palabras_ere.append(auxiliar_palabra)
        fila.append(contfilas)
        palabra.append(contrpalabra)

    return palabras_ere

def estadoCero(caracter):
    if(caracter == 'e'):
        print('Q0→Q1')
        return 1
    else:
        print('Q0→Q0')
        return 0

def estadoUno(caracter):
    if(caracter == 'e'):
        print('Q1→Q1')
        return 1
    elif(caracter == 'r'):
        print('Q1→Q2')
        return 2
    else:
        print('Q1→Q0')
        return 0

def estadoDos(caracter):
    if(caracter == 'e'):
        print('Q2→Q3')
        return 3
    else:
        print('Q2→Q0')
        return 0

def estadoTres(caracter):
    if(caracter == 'r'):
        print('Q3→Q2')
        return 2
    elif(caracter == 'e'):
        print('Q3→Q1')
        return 1
    else:
        return -1

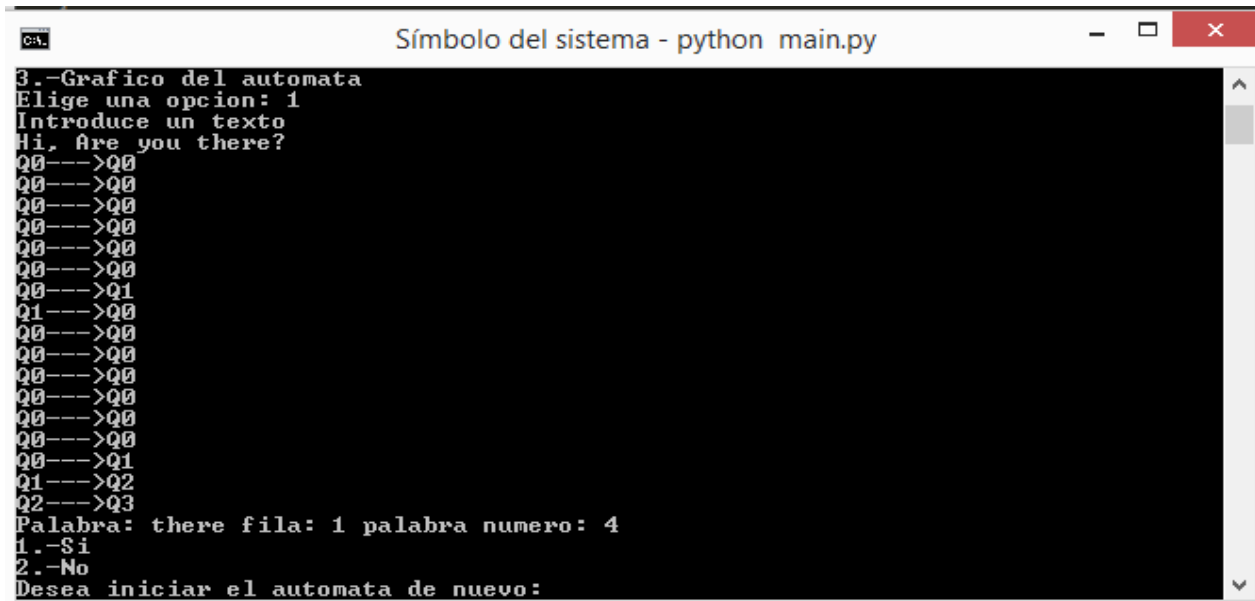
```

#ASCII letras minusculas 97-122

3.3. Pruebas

Se mostraran algunas pruebas realizadas de los modos manual, leer archivo y mostrar diagrama.

Para el modo manual en el que se introducirá una cadena.



```
Símbolo del sistema - python main.py
3.-Grafico del automata
Elige una opcion: 1
Introduce un texto
Hi, Are you there?
Q0--->Q0
Q0--->Q0
Q0--->Q0
Q0--->Q0
Q0--->Q0
Q0--->Q0
Q0--->Q1
Q1--->Q0
Q0--->Q0
Q0--->Q0
Q0--->Q0
Q0--->Q0
Q0--->Q0
Q0--->Q1
Q1--->Q2
Q2--->Q3
Palabra: there fila: 1 palabra numero: 4
1.-Si
2.-No
Desea iniciar el automata de nuevo:
```

Figura 12: Modo manual en el cual se introdujo una cadena

Para el modo leer archivo. En este modo el archivo que se leera sera la biografia de William Shakespeare[1]

```
Simbolo del sistema - python main.py
Q1--->Q0
Q0--->Q0
Q0--->Q0
Q0--->Q0
Q0--->Q0
Q0--->Q0
Q0--->Q0
Palabra: there fila: 1 palabra numero: 37
Palabra: were fila: 6 palabra numero: 65
Palabra: there fila: 7 palabra numero: 7
Palabra: where fila: 7 palabra numero: 15
Palabra: there fila: 7 palabra numero: 27
Palabra: there fila: 9 palabra numero: 3
Palabra: there fila: 9 palabra numero: 53
Palabra: were fila: 11 palabra numero: 85
Palabra: were fila: 13 palabra numero: 12
Palabra: were fila: 13 palabra numero: 77
Palabra: were fila: 16 palabra numero: 5
Palabra: there fila: 16 palabra numero: 81
Palabra: were fila: 18 palabra numero: 12
Palabra: where fila: 23 palabra numero: 161
Palabra: there fila: 23 palabra numero: 169
1.-Si
2.-No
Desea iniciar el automata de nuevo: _
```

Figura 13: Modo leer archivo

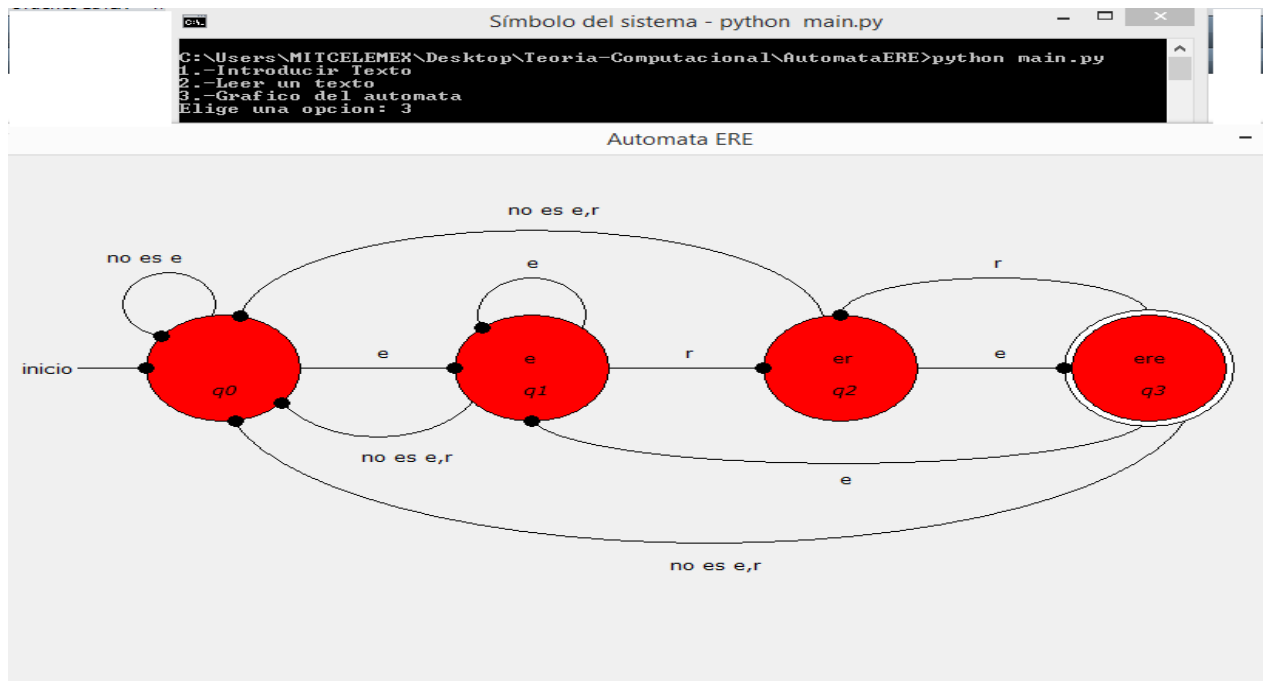


Figura 14: Visualizar diagrama del autómata

4. Autómata Paridad

4.1. Descripción

El siguiente programa determina si una cadena binaria tiene paridad de 1's y 0's, es decir tiene una cantidad par de 1's y una cantidad par de 0's

El programa tendrá dos modos, el modo manual y automático, en el modo manual el usuario ingresará una cadena binaria de la longitud deseada, el modo automático generará una cadena binaria con una cardinalidad de entre 1 a 1000, en ambos casos se mostrara en pantalla si la cadena tiene o no paridad.

De igual forma se podrá observar el diagrama del autómata.

El autómata a utilizar es el siguiente:

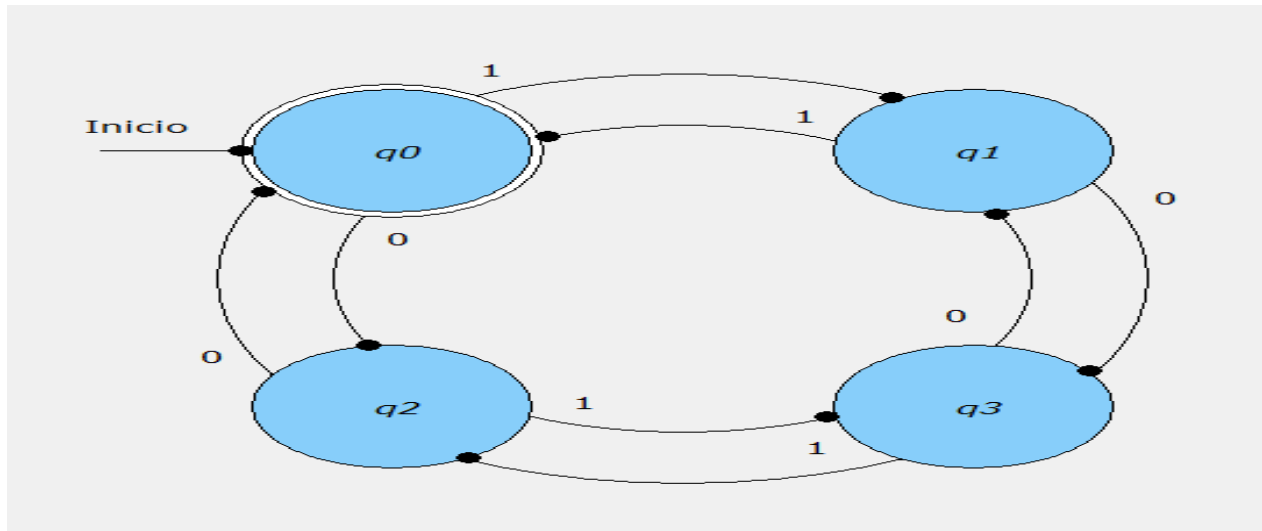


Figura 15: Autómata.

4.2. Código

Se mostrara el código del diagrama de paridad

Código:Diagrama2.py

```
from tkinter import *

def mostrarDiagrama():
    ventana1 = Tk()
    ventana1.geometry("600x600+400+50") #geometry(widthxheight)
    ventana1.title("Automata_Paridad")
    ventana1.resizable(width=False, height=False)
    AreaDibujo1=Canvas(ventana1, width=595, height=595)
    AreaDibujo1.pack()
```



```

#Circulos transicion
AreaDibujo1.create_oval(100,100,500,500)
AreaDibujo1.create_oval(150,150,450,450)

#Circulos de estado
AreaDibujo1.create_oval(110,110,240,240, fill="white")
AreaDibujo1.create_oval(115,115,235,235, fill="light_sky_blue")
AreaDibujo1.create_oval(365,115,485,235, fill="light_sky_blue")
AreaDibujo1.create_oval(115,365,235,485, fill="light_sky_blue")
AreaDibujo1.create_oval(365,365,485,485, fill="light_sky_blue")

#Linea de inicio
AreaDibujo1.create_line(50,175,110,175)

#Circulos de sentido
AreaDibujo1.create_oval(105,170,115,180, fill="black")
AreaDibujo1.create_oval(385,118,395,128, fill="black")
AreaDibujo1.create_oval(470,385,480,395, fill="black")
AreaDibujo1.create_oval(203,470,213,480, fill="black")
AreaDibujo1.create_oval(115,210,125,220, fill="black")

AreaDibujo1.create_oval(237,156,247,166, fill="black")
AreaDibujo1.create_oval(160,360,170,370, fill="black")
AreaDibujo1.create_oval(357,430,367,440, fill="black")
AreaDibujo1.create_oval(430,232,440,242, fill="black")

#etiquetas
inicio=Label(ventanal, text="Inicio", font="Verdana_12_roman").place(x
    =40,y=140)
qo=Label(ventanal, text="q0", font="Verdana_12_italic", background="light
    _sky_blue").place(x=165,y=165)
q1=Label(ventanal, text="q1", font="Verdana_12_italic", background="light
    _sky_blue").place(x=415,y=165)
q2=Label(ventanal, text="q2", font="Verdana_12_italic", background="light
    _sky_blue").place(x=165,y=415)
q3=Label(ventanal, text="q3", font="Verdana_12_italic", background="light
    _sky_blue").place(x=415,y=415)

#etiquetas transicion
qoq1=Label(ventanal, text="1", font="Verdana_12_roman").place(x=210,y
    =85)
q1q0=Label(ventanal, text="1", font="Verdana_12_roman").place(x=345,y
    =130)

q2q3=Label(ventanal, text="1", font="Verdana_12_roman").place(x=250,y
    =410)
q3q2=Label(ventanal, text="1", font="Verdana_12_roman").place(x=350,y
    =454)

qoq2=Label(ventanal, text="0", font="Verdana_12_roman").place(x=170,y

```

```

        =250)
q2qo=Label(ventana1 , text="0" , font="Verdana_12_roman" ) . place (x=90,y
        =365)

q1q3=Label(ventana1 , text="0" , font="Verdana_12_roman" ) . place (x=500,y
        =210)
q3q1=Label(ventana1 , text="0" , font="Verdana_12_roman" ) . place (x=410,y
        =325)

ventana1 . mainloop ()

```

Código:mainPar.py

```

import automataPar
import Diagrama2
import funcionesPar

#menu
opcion=funcionesPar . menu()

if (opcion==1):
    while True:

        cadena=input ( "Introduce_una_cadena_binaria:_")
        print (cadena)
        print ( "numero_de_ceros:_ " + str (cadena . count ("0")))
        print ( "numero_de_unos:_ " + str (cadena . count ("1")))

        resultado=automataPar . automata (cadena)
        if (resultado==1):
            print ( "Es_una_cadena_con_paridad")
        elif (resultado==0):
            print ( "No_es_una_cadena_con_paridad")
        else :
            print ( "no_es_una_cadena_binaria")
        while True:
            nC=funcionesPar . nueva_cadena ()
            if (nC==1):
                break
            elif (nC==2):
                exit ()
            else :
                continue

elif (opcion==2):
    while True:

```

```

cadena=funcionesPar.generarCadena()
print(cadena)
print("numero_de_ceros:_ " + str(cadena.count("0")))
print("numero_de_unos:_ " + str(cadena.count("1")))
resultado=automataPar.automata(cadena)
if(resultado==1):
    print("Es_una_cadena_con_paridad")
elif(resultado==0):
    print("No_es_una_cadena_con_paridad")

while True:
    print("1.-Nuevo_conteo\n2.- salir_")
    nC=funcionesPar.op_aleatorio()
    if(nC==1):
        break
    elif(nC==2):
        exit()
    else:
        continue

elif(opcion==3):
    Diagrama2.mostrarDiagrama()

else:
    print("opcion_invalida")

```

Código:funcionesPar.py

```

import random

def menu():
    try:
        opcion=input(" \t\t\t Paridad\n1.-Modo_manual\n2.-Modo_Automatico\n3.-Mostrar_Diagrama\nElige_una_opcion:_ ")
        opcion=int(opcion)
        return opcion
    except:
        exit()

def nueva_cadena():
    try:
        conteo = input("1.-Nuevo_conteo\n2.- salir_")
        conteo = int(conteo)
        return conteo
    except:
        print("opcion_invalida")

def generarCadena():
    cardinalidad=cardinalidad_aleatoria()
    tamaño=0
    cadena=""

```

```

while (tamano<cardinalidad):
    num=num_aleatorio()
    num=str(num)
    cadena=cadena+num
    tamano=tamano + 1
return cadena

def cardinalidad_aleatoria():
    numero = random.randrange(1,1000)
    return numero
def num_aleatorio():
    bit = random.randrange(0,2)
    return bit
def num_aleatorio():
    bit = random.randrange(0,2)
    return bit

```

Código:automataPar.py

```

#codigo ascii del 0-1 "48-49"
def automata(cadena):
    estado=0

    for bit in cadena:
        if (bit == "_"):
            break
        if (estado==0):
            estado=estadoCero(bit)
        elif (estado==1):
            estado=estadoUno(bit)
        elif (estado==2):
            estado=estadoDos(bit)
        elif (estado==3):
            estado=estadoTres(bit)

    if (estado==0):
        return 1
    elif (estado==1):
        return -1
    else:
        return 0

def estadoCero(elemento):
    if (elemento=="0"):
        print ("Q0_--_ %s_-->Q2" %elemento)

```

```

        return 2
    elif(elemento=="1"):
        print("Q0_--_%s_-->Q1" %elemento)
        return 1
    else:
        return -1
def estadoUno(elemento):
    if (elemento=="0"):
        print("Q1_--_%s_-->Q3" %elemento)
        return 3
    elif(elemento=="1"):
        print("Q1_--_%s_-->Q0" %elemento)
        return 0
    else:
        return -1

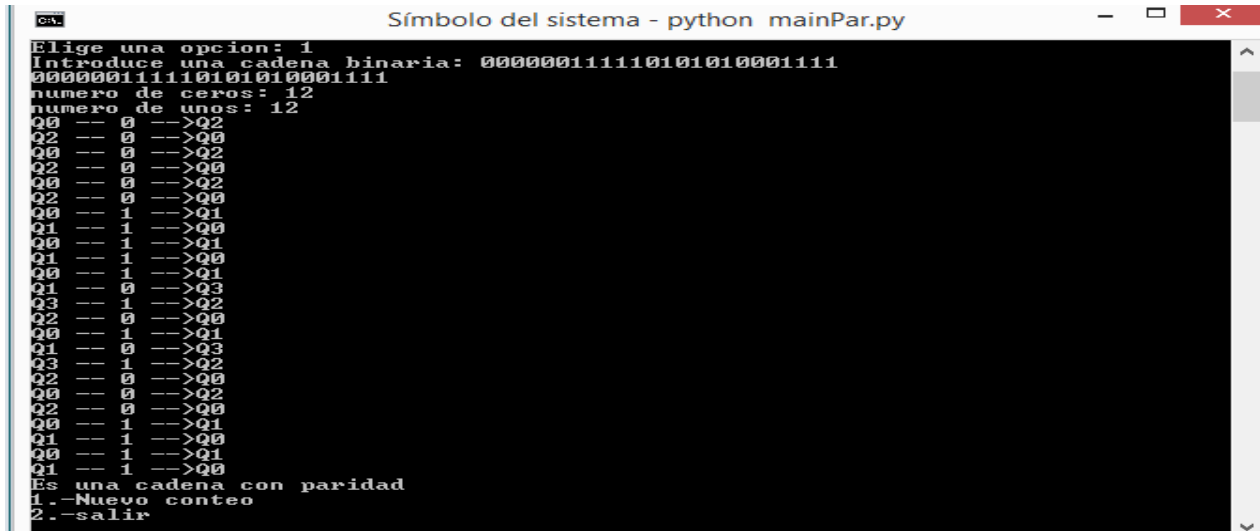
def estadoDos(elemento):
    if(elemento=="0"):
        print("Q2_--_%s_-->Q0" %elemento)
        return 0
    elif(elemento=="1"):
        print("Q2_--_%s_-->Q3" %elemento)
        return 3
    else:
        return -1

def estadoTres(elemento):
    if(elemento=="0"):
        print("Q3_--_%s_-->Q1" %elemento)
        return 1
    elif(elemento=="1"):
        print("Q3_--_%s_-->Q2" %elemento)
        return 2
    else:
        return -1

```

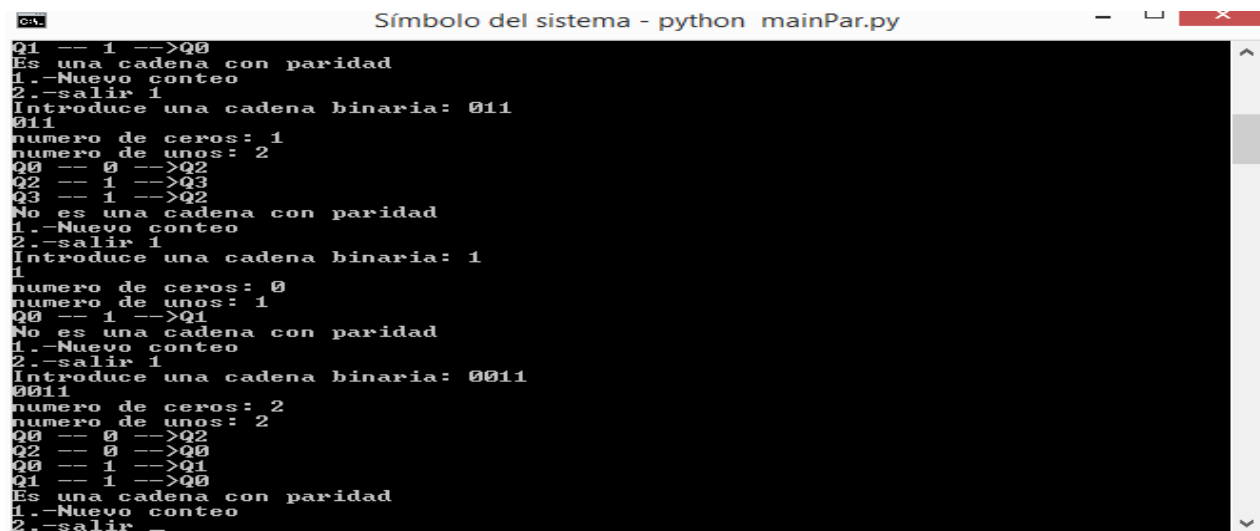
4.3. Pruebas

A continuación se mostraran las pruebas del programa.
Pruebas del modo manual.



```
Símbolo del sistema - python mainPar.py
Elige una opcion: 1
Introduce una cadena binaria: 000000111110101010001111
000000111110101010001111
numero de ceros: 12
numero de unos: 12
Q0 -- 0 --> Q2
Q2 -- 0 --> Q0
Q0 -- 0 --> Q2
Q2 -- 0 --> Q0
Q0 -- 0 --> Q2
Q2 -- 0 --> Q0
Q0 -- 1 --> Q1
Q1 -- 1 --> Q0
Q0 -- 1 --> Q1
Q1 -- 1 --> Q0
Q0 -- 1 --> Q1
Q1 -- 0 --> Q3
Q3 -- 1 --> Q2
Q2 -- 0 --> Q0
Q0 -- 1 --> Q1
Q1 -- 0 --> Q3
Q3 -- 1 --> Q2
Q2 -- 0 --> Q0
Q0 -- 0 --> Q2
Q2 -- 0 --> Q0
Q0 -- 1 --> Q1
Q1 -- 1 --> Q0
Q0 -- 1 --> Q1
Q1 -- 1 --> Q0
Es una cadena con paridad
1.-Nuevo conteo
2.-salir
```

Figura 16: Modo manual al introducir una cadena binaria



```
Símbolo del sistema - python mainPar.py
Q1 -- 1 --> Q0
Es una cadena con paridad
1.-Nuevo conteo
2.-salir 1
Introduce una cadena binaria: 011
011
numero de ceros: 1
numero de unos: 2
Q0 -- 0 --> Q2
Q2 -- 1 --> Q3
Q3 -- 1 --> Q2
No es una cadena con paridad
1.-Nuevo conteo
2.-salir 1
Introduce una cadena binaria: 1
1
numero de ceros: 0
numero de unos: 1
Q0 -- 1 --> Q1
No es una cadena con paridad
1.-Nuevo conteo
2.-salir 1
Introduce una cadena binaria: 0011
0011
numero de ceros: 2
numero de unos: 2
Q0 -- 0 --> Q2
Q2 -- 0 --> Q0
Q0 -- 1 --> Q1
Q1 -- 1 --> Q0
Es una cadena con paridad
1.-Nuevo conteo
2.-salir
```

Figura 17: Modo manual al introducir una nueva cadena

Pruebas del modo automático.

```

S mbolo del sistema
Elige una opcion: 2
0011001111110011
numero de ceros: 6
numero de unos: 10
Q0 -- 0 --> Q2
Q2 -- 0 --> Q0
Q0 -- 1 --> Q1
Q1 -- 1 --> Q0
Q0 -- 0 --> Q2
Q2 -- 0 --> Q0
Q0 -- 1 --> Q1
Q1 -- 1 --> Q0
Q0 -- 1 --> Q1
Q1 -- 1 --> Q0
Q0 -- 1 --> Q1
Q1 -- 1 --> Q0
Q0 -- 0 --> Q2
Q2 -- 0 --> Q0
Q0 -- 1 --> Q1
Q1 -- 1 --> Q0
Es una cadena con paridad
1.-Nuevo conteo
2.-salir
0011101111001010
numero de ceros: 8
numero de unos: 9
Q0 -- 0 --> Q2
Q2 -- 0 --> Q0
Q0 -- 1 --> Q1

```

Figura 18: Modo autom tico generando cadena aleatoria

Prueba de la visualizaci n del diagrama.

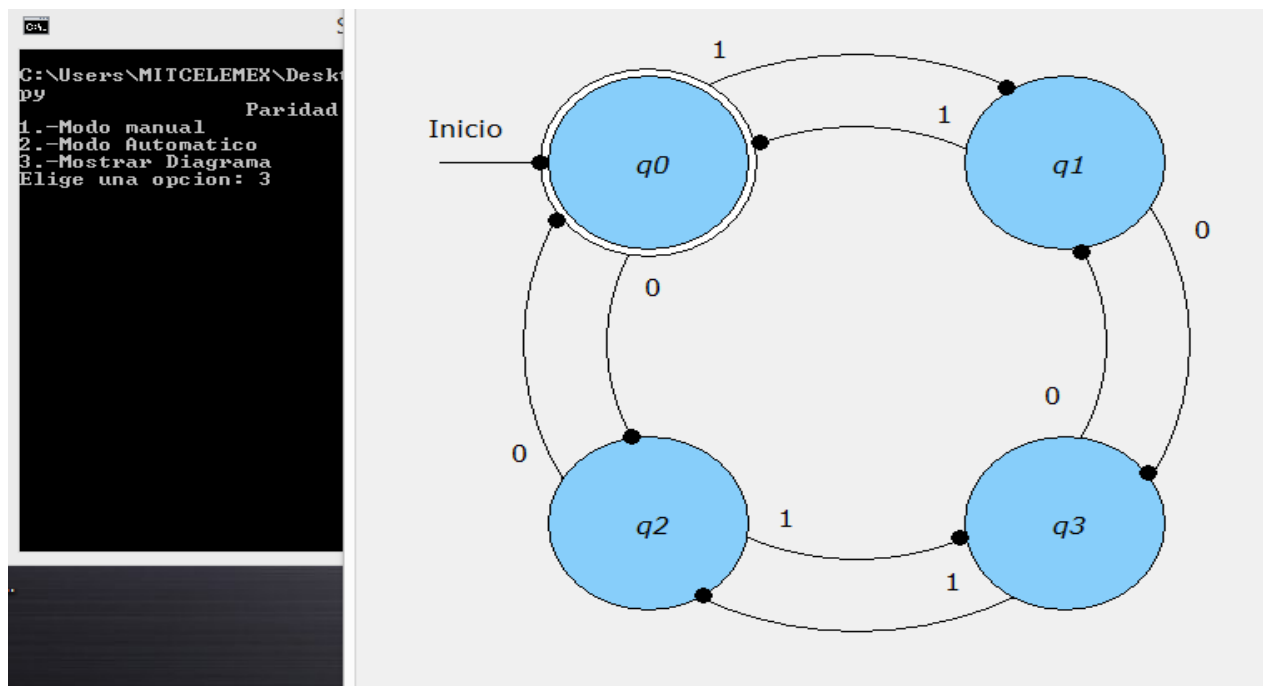


Figura 19: Ejecuci n del diagrama desde el programa

5. Protocolo

5.1. Descripción

Se mostrara el funcionamiento de un protocolo, el siguiente programa genera un archivo con 50 cadenas binarias aleatorias, posteriormente se esperara un segundo de time out para posteriormente ser validadas por medio del autómata de paridad, así después se preguntara el estado del protocolo y si esta encendido se realizara lo antes dicho, de lo contrario acabara el programa. Diagrama del protocolo

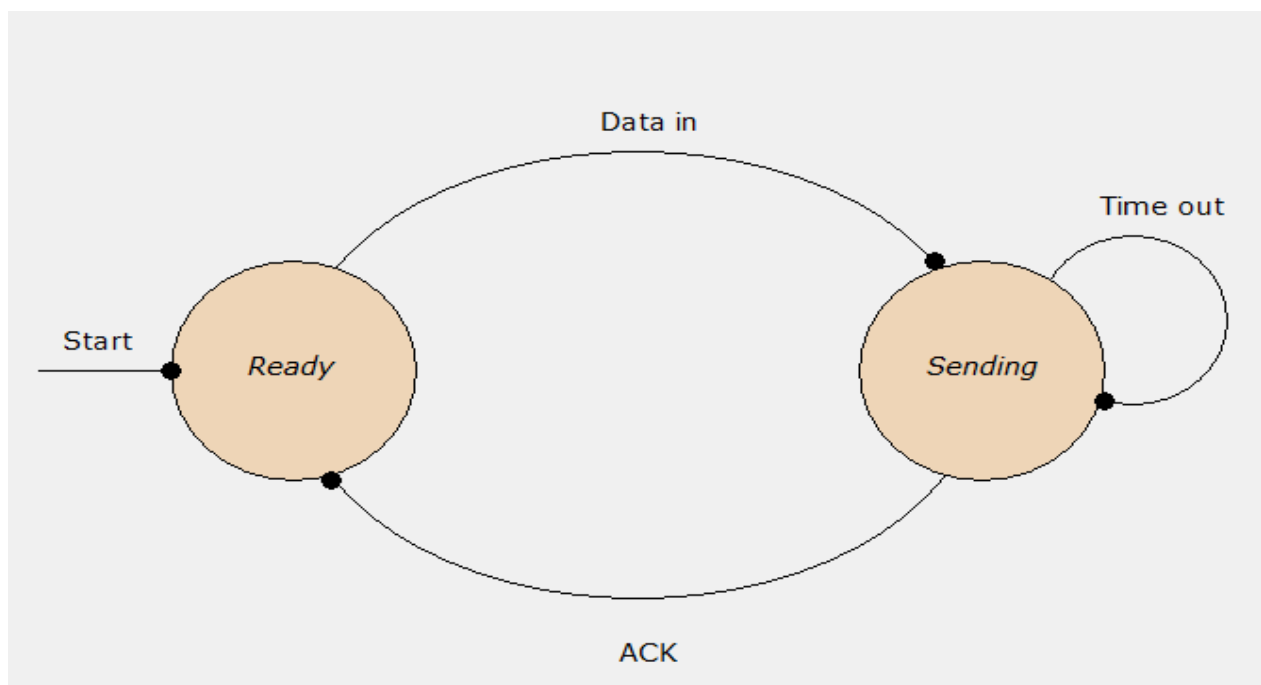


Figura 20: Protocolo

5.2. Código

A continuación se muestra el código del protocolo

Código:DiagramaProtocolo.py

```
from tkinter import *  
  
def mostrarDiagrama():  
    ventana = Tk()  
    ventana.geometry("700x500") #geometry(widthxheight)
```



```

ventana.title("Protocolo")
ventana.resizable(width=False,height=False)
AreaDibujo=Canvas(ventana,width=700,height=500)
AreaDibujo.pack()

#Ovalo transicion
AreaDibujo.create_oval(165,120,530,385)
AreaDibujo.create_oval(560,170,660,270)

#Estados del protocolo
AreaDibujo.create_oval(100,185,230,315,fill="bisque2")
AreaDibujo.create_oval(465,185,595,315,fill="bisque2")

#linea de inicio
AreaDibujo.create_line(30,250,100,250)
AreaDibujo.create_oval(95,245,105,255,fill="black")

#circulos de sentido
AreaDibujo.create_oval(180,310,190,320,fill="black")
AreaDibujo.create_oval(500,180,510,190,fill="black")
AreaDibujo.create_oval(590,263,600,273,fill="black")

#etiqueta inicio
start=Label(ventana,text="Start",font="Verdana_11_roman").place(x=40,y=220)
#etiquetas
datain=Label(ventana,text="Data_in",font="Verdana_11_roman").place(x=325,y=90)
timeout=Label(ventana,text="Time_out",font="Verdana_11_roman").place(x=590,y=140)
ACK=Label(ventana,text="ACK",font="Verdana_11_roman").place(x=335,y=405)

#etiquetas de estado del protocolo
ready=Label(ventana,text="Ready",font="Verdana_11_italic",background="bisque2").place(x=138,y=235)
sending=Label(ventana,text="Sending",font="Verdana_11_italic",background="bisque2").place(x=498,y=235)

ventana.mainloop()

```

Código:automataPar.py

```

#codigo ascii del 0-1 "48-49"
def automata(cadena):
    estado=0

```

```

for bit in cadena:
    if (bit == "_"):
        break
    if (estado==0):
        estado=estadoCero ( bit )
    elif (estado==1):
        estado=estadoUno ( bit )
    elif (estado==2):
        estado=estadoDos ( bit )
    elif (estado==3):
        estado=estadoTres ( bit )

if (estado==0):
    return 1
elif (estado== -1):
    return -1
else:
    return 0

def estadoCero (elemento):
    if (elemento=="0"):
        return 2
    elif (elemento=="1"):
        return 1
    else:
        return -1
def estadoUno (elemento):
    if (elemento=="0"):
        return 3
    elif (elemento=="1"):
        return 0
    else:
        return -1

def estadoDos (elemento):
    if (elemento=="0"):
        return 0
    elif (elemento=="1"):
        return 3
    else:
        return -1

def estadoTres (elemento):
    if (elemento=="0"):
        return 1
    elif (elemento=="1"):
        return 2
    else:

```

```
return -1
```

Código:mainProtocolo.py

```
import protocolo
import DiagramaProtocolo

try:
    opcion=input( "1.- Iniciar_Protocolo\n2.- Mostrar_diagrama_del_protocolo\n
        n_Elija_una_opcion:_")
    opcion=int(opcion)
except:
    exit()

if(opcion==1):
    protocolo.IniciarProtocolo()
elif(opcion==2):
    DiagramaProtocolo.mostrarDiagrama()
else:
    print( "Opcion_invalida")
```

Código:protocolo.py

```
import random
import automataPar
import time

def IniciarArchivo():
    archivo=open( "cadenasGeneradas.txt", "w")
    archivo.close
def IniciarArchivoValidas():
    archivol=open( "cadenasValidas.txt", "w")
    archivol.close

def num_aleatorio():
    bit = random.randrange(0,2)
    return bit

def generarCadena():
    cardinalidad=32
    tamaño=0
    cadena=""
    while (tamaño<cardinalidad):
        num=num_aleatorio()
        num=str(num)
```

```

        cadena=cadena+num
        tamano=tamano + 1
    return cadena

```

```

def IniciarProtocolo():

```

```

    IniciarArchivoValidas()
    encendido=True

```

```

    while encendido:

```

```

        IniciarArchivo()

```

```

        print ("——Iniciando_Proocolo——")

```

```

        archivo=open("cadenasGeneradas.txt","a")

```

```

        for x in range(1,51):

```

```

            cadena=generarCadena()

```

```

            archivo.write(cadena)

```

```

            archivo.write("_")

```

```

        archivo.close()

```

```

        print ("——Enviando_datos_generados——")

```

```

        try:

```

```

            archivo=open("cadenasGeneradas.txt","r")

```

```

        except:

```

```

            print ("Error_al_enviar_el_archivo")

```

```

            exit()

```

```

        texto=archivo.read()

```

```

        archivo.close()

```

```

        print ("——Esperando——")

```

```

        time.sleep(1)

```

```

        print ("——Validando_archivo——")

```

```

        archivo1=open("cadenasValidas.txt","a")

```

```

        archivo1.write("\n\n")

```

```

        palabra=""

```

```

        for bit in texto:

```

```

            if(bit != "_"):

```

```

                palabra=palabra+bit

```

```

            if(bit == "_"):

```

```

                escribir=automataPar.automata(palabra)

```

```

                if(escribir==1):

```

```

                    archivo1.write(palabra)

```

```

                    archivo1.write("\n")

```

```

                    palabra=""

```

```

                elif(escribir==0):

```

```

                    palabra=""

```

```

                else:

```

```

                    continue

```

```

        archivo1.close()

```

```

        print ("Validando_estado(encendido/apagado)")

```

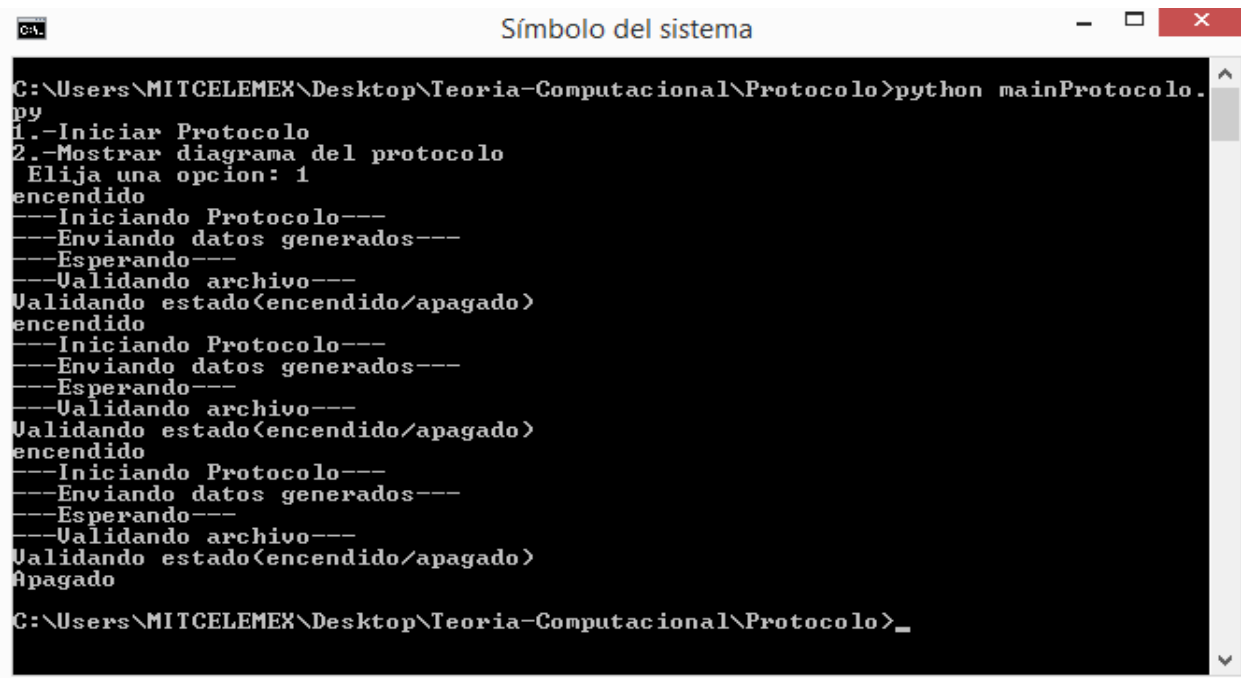
```

        encendido=random.choice([True, False])

```

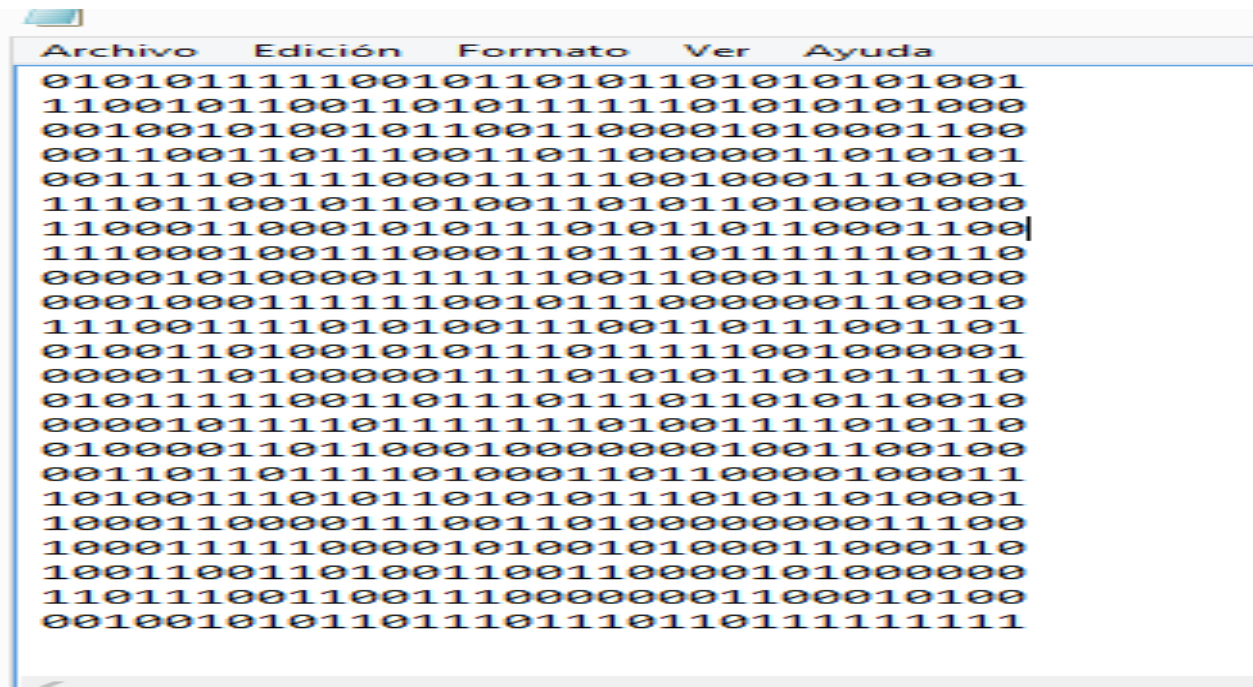
5.3. Pruebas

A continuación se mostraran los resultados de las pruebas obtenidas



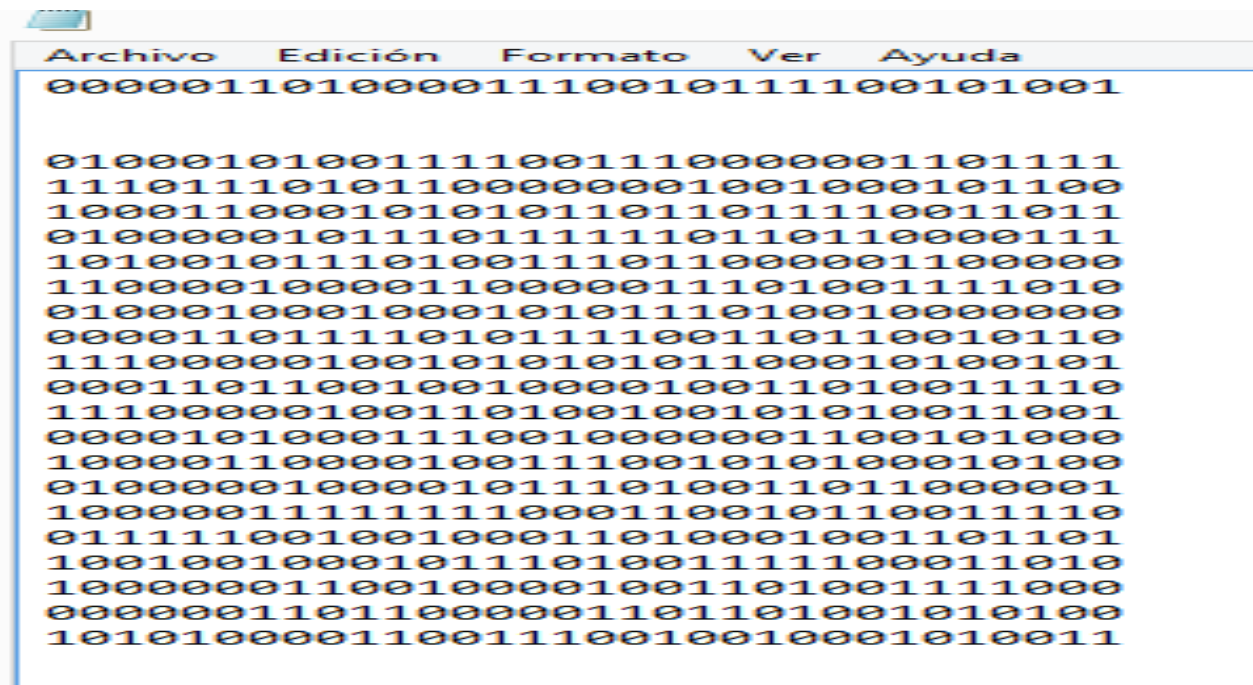
```
C:\Users\MITCELEMEX\Desktop\Teoria-Computacional\Protocolo>python mainProtocolo.py
1.-Iniciar Protocolo
2.-Mostrar diagrama del protocolo
Elija una opcion: 1
encendido
---Iniciando Protocolo---
---Enviando datos generados---
---Esperando---
---Validando archivo---
Validando estado(encendido/apagado)
encendido
---Iniciando Protocolo---
---Enviando datos generados---
---Esperando---
---Validando archivo---
Validando estado(encendido/apagado)
encendido
---Iniciando Protocolo---
---Enviando datos generados---
---Esperando---
---Validando archivo---
Validando estado(encendido/apagado)
Apagado
C:\Users\MITCELEMEX\Desktop\Teoria-Computacional\Protocolo>_
```

Figura 21: iniciar protocolo



```
Archivo  Edición  Formato  Ver  Ayuda
010101111110010110101101010101001
11001011001101011111101010101000
00100101001011001100001010001100
00110011011100110110000011010101
00111101111000111110010001110001
11101100101101001101011010001000
11000110001010111010110110001100
11100010011100011011101111110110
0000101000001111110011000111110000
000100011111100101110000000110010
11100111101010011100110111001101
01001101001010111011111001000001
000011010000001111010101101011110
010111111001101110111011010110010
00001011110111111101001111010110
01000011011000100000001001100100
00110110111101000110110000100011
10100111010110101011101011010001
100011000011100110100000000011100
1000111100001010010100011000110
10011001101001100110000101000000
110111001100111000000001100010100
00100101011011101110110111111111
```

Figura 22: Cadenas aceptadas por el protocolo



```
Archivo  Edición  Formato  Ver  Ayuda
00000110100001110010111100101001

0100010100111110011100000001101111
111011101011000000001001000101100
1000110001010101011011110011011
01000001011101111110110110000111
101001011101001110110000001100000
110000100001100000011101001111010
010001000100010101110100100000000
00001101111010111100110110010110
11100000100101010101100010100101
00011011001001000010011010011110
11100000100110100100101010011001
000010100011100100000001100101000
10000110000100111001010100010100
01000001000010111010011011000001
100000111111111000110010110011110
011111100100100011010001001101101
10010010001011101001111100011010
10000001100100001001101001111000
000000110110000011011010010100
10101000011001110010010001010011
```

Figura 23: Cadenas aceptadas por el protocolo

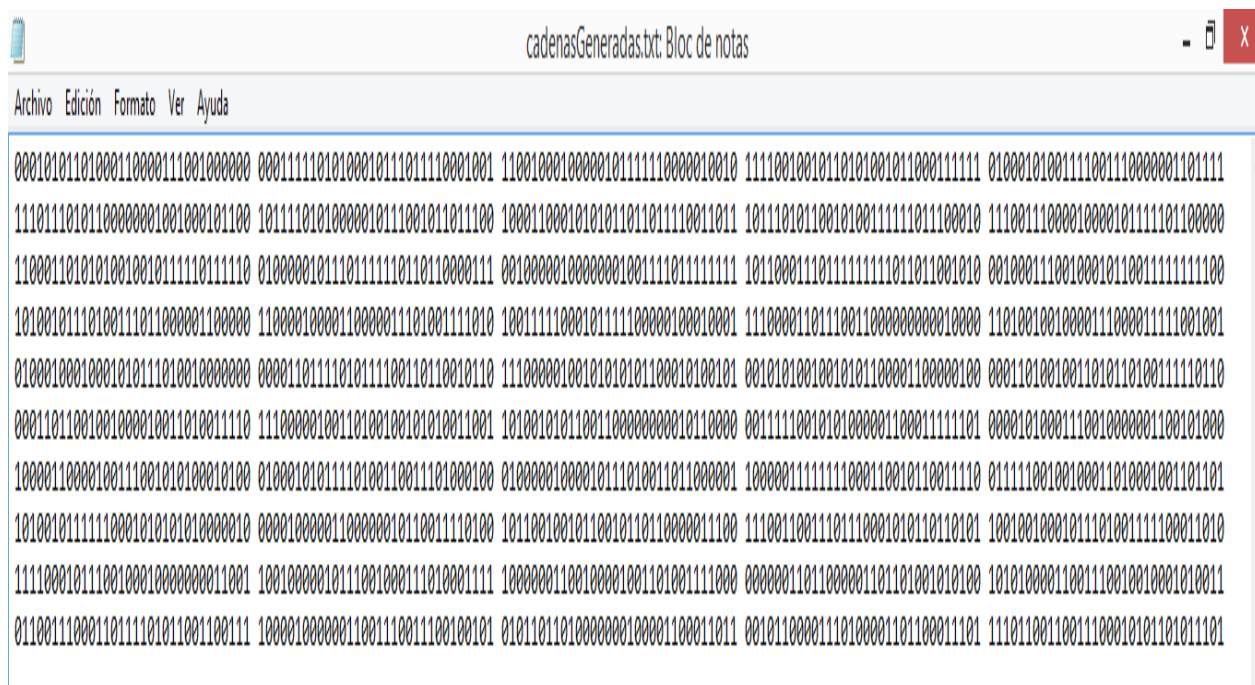


Figura 24: Ultimas 50 cadenas binarias generadas(Las cadenas están acomodadas en esta imagen de tal forma que se puedan visualizar todas)

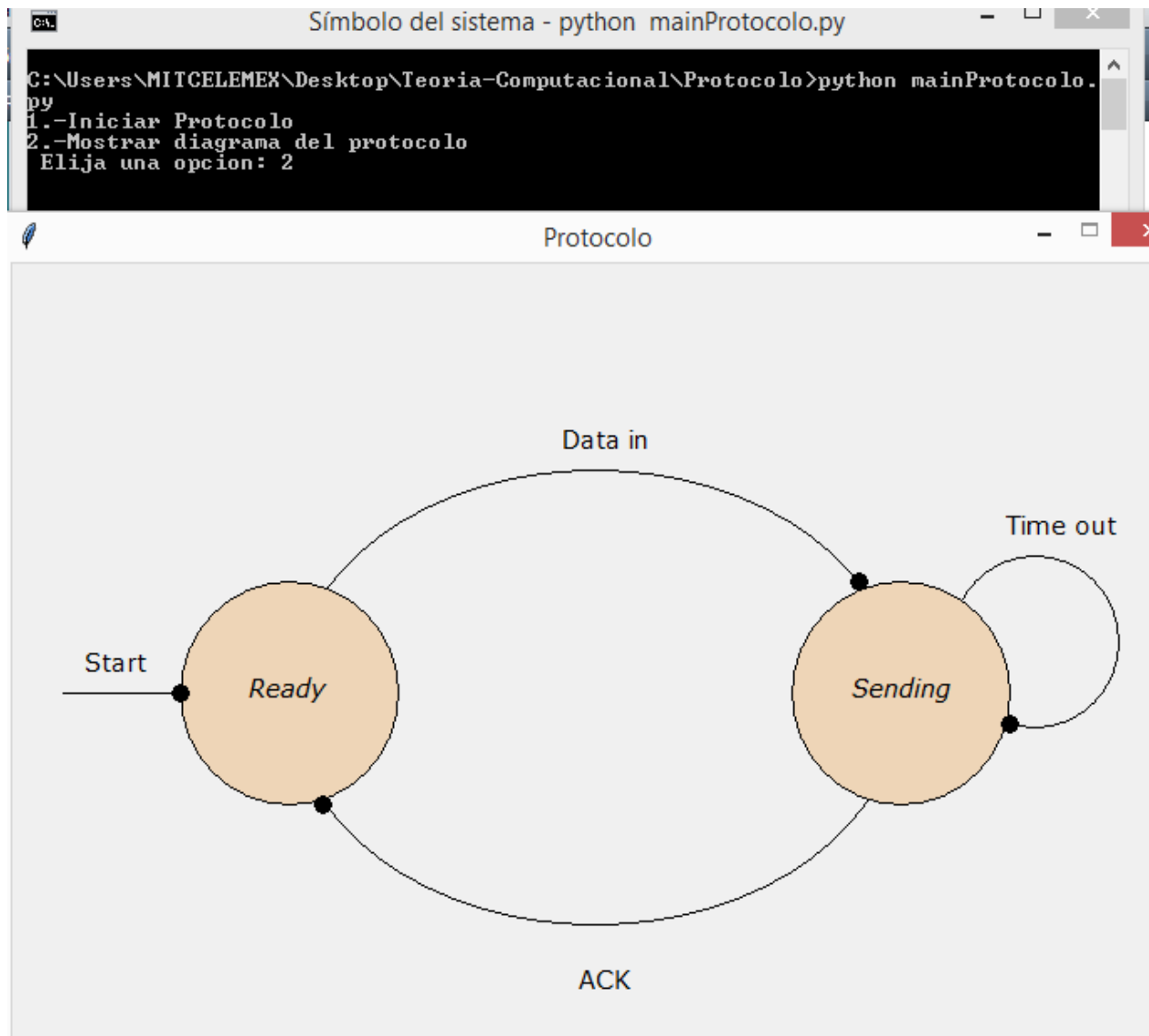


Figura 25: Diagrama del protocolo ejecutado desde el programa

6. Cadenas terminadas en 01

6.1. Descripción

El programa localiza cada una de las cadenas terminadas en 01, esto se hará mediante un autómata no determinístico, tendrá un modo para ingresar la cadena manualmente y uno mediante el cual se genere.

El autómata que se utilizara será el siguiente

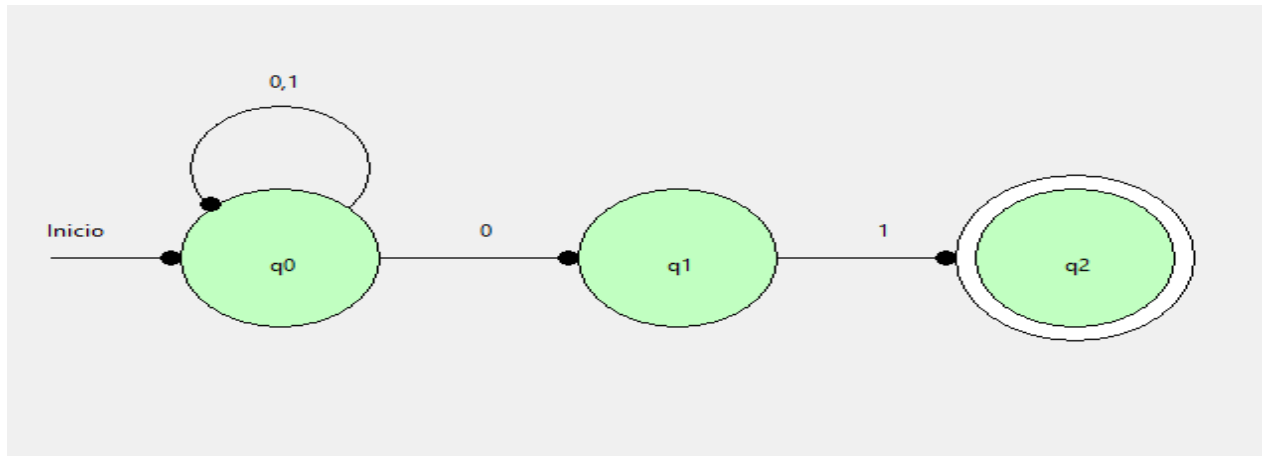


Figura 26: Diagrama del autómata

6.2. código

Código:DiagramaAND.py

```
from tkinter import *

def mostrarAND():
    ventana = Tk()
    ventana.geometry("650x500") #geometry(widthxheight)
    ventana.title("Automata_no_deterministico")
    ventana.resizable(width=False, height=False)
    AreaDibujo=Canvas(ventana, width=650, height=500)
    AreaDibujo.pack()

    #circulo transicion
    AreaDibujo.create_oval(105,140,195,230)

    #Circulos de estado
    AreaDibujo.create_oval(100,200,200,300, fill="DarkSeaGreen1")
    AreaDibujo.create_oval(300,200,400,300, fill="DarkSeaGreen1")
```

```

AreaDibujo.create_oval(490,190,610,310,fill="white")
AreaDibujo.create_oval(500,200,600,300,fill="DarkSeaGreen1")

#lineas de transicion

AreaDibujo.create_line(35,250,100,250)
AreaDibujo.create_line(200,250,300,250)
AreaDibujo.create_line(400,250,490,250)

#Circulos de sentido

AreaDibujo.create_oval(90,245,100,255,fill="black")
AreaDibujo.create_oval(290,245,300,255,fill="black")
AreaDibujo.create_oval(480,245,490,255,fill="black")
AreaDibujo.create_oval(110,206,120,216,fill="black")

#Etiquetas
inicio=Label(ventana,text="Inicio").place(x=30,y=220)
q0=Label(ventana,text="q0",background="DarkSeaGreen1").place(x=142,y
=245)
q1=Label(ventana,text="q1",background="DarkSeaGreen1").place(x=342,y
=245)
q2=Label(ventana,text="q2",background="DarkSeaGreen1").place(x=542,y
=245)
cero=Label(ventana,text="0").place(x=248,y=220)
uno=Label(ventana,text="1").place(x=448,y=220)
cero_uno=Label(ventana,text="0,1").place(x=142,y=112)

ventana.mainloop()

```

6.3. Pruebas

Se mostrara la prueba del diagrama.

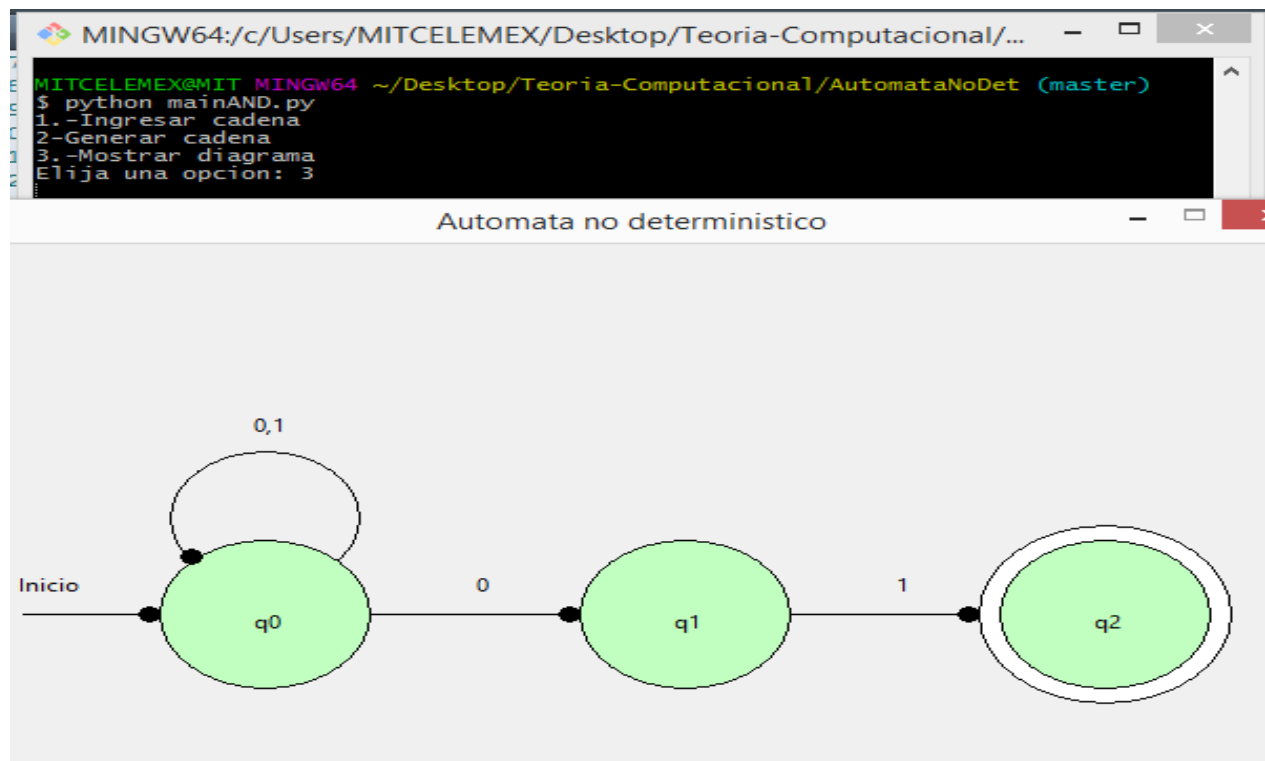


Figura 27: Diagrama del autómata en ejecución

Referencias

- [1] BIOGRAFÍAS Y VIDA, *William Shakespeare*, url:
<http://www.biografiasyvidas.com/biografia/s/shakespeare.html>
- [2] HOPCROFT JOHN, MOTWANI RAJEEV, «Introduction to Automata Theory, Languages and Computation», *Addison Wesley*, 2008