UNIVERSIDAD DE ORIENTE
NÚCLEO ANZOÁTEGUI
ESCUELA DE INGENIERÍA Y CIENCIAS APLICADAS
DEPARTAMENTO DE COMPUTACIÓN
LABORATORIO DE PROYECTOS DIGITALES AVANZADOS

Recreación UART

Prof. Rene Cabrera

Elaborado por:

Brazon Eulises

Barcelona, 25 de enero del 2023

ABSTRACTO

En la presente practica se busca recrear el comportamiento del protocolo de comunicación UART, de manera simplificada, creando los métodos necesarios para entablar esta comunicación serial. Con el uso de dos raspberry pi pico.

OBJETIVOS

Objetivo General:

- Entender el funcionamiento del protocolo UART

Objetivos Específicos

- Crear el código necesario para la comunicación UART
- Entablar una comunicación uart entre dos raspberry pi pico,

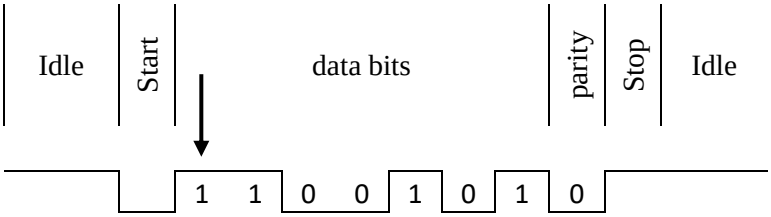
MATERIALES REQUERIDOS

- Jumpers para la conexión
- Protoboard.
- Resistencia 220Ω (17unidades).
- 8 leds rojos
- 8 leds verdes
- 1 led amarillo
- 9 pulsadores.
- 2 raspberry
- Alimentación 5V.
- Herramientas para la manipulación y testeo (multímetro, pinzas, entre otros).

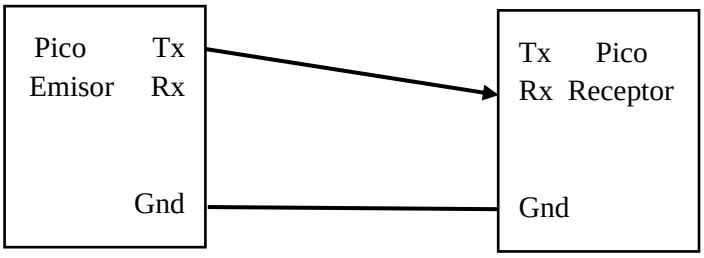
DESCRIPCION DEL PROYECTO

En el presente informe se busca comprender el funcionamiento del protocolo de comunicación UART, mediante la creación de la lógica necesaria, que permita la manipulación del pin de salida Tx y pin de entrada Rx, para poder establecer la comunicación serial, cómo una forma de demostrar el correcto funcionamiento del algoritmo implementado, se estará enviando un grupo de 8 bits desde un emisor a un receptor, tanto el emisor cómo el receptor poseen LEDS, utilizados para indicar el estado de los byte (1 o 0), el emisor además contará con 9 pulsadores, 8 de ellos están asignados para la manipulación del estado de cada byte, y el noveno pulsador se utilizará para enviar la información que se encuentra almacenada en el pico emisor, hacia el receptor, el cuál una vez recibido el byte, usará esa información para encender los LEDS correspondientes, si el receptor refleja los mismos estados de los LEDS que el emisor, podremos afirmar que se logró cumplir con el objetivo.

Por temas de simplificación el bit de paridad no será tomado en cuenta en la lógica del programa, para entender el funcionamiento del código, pero primero debemos entender el funcionamiento de uart:

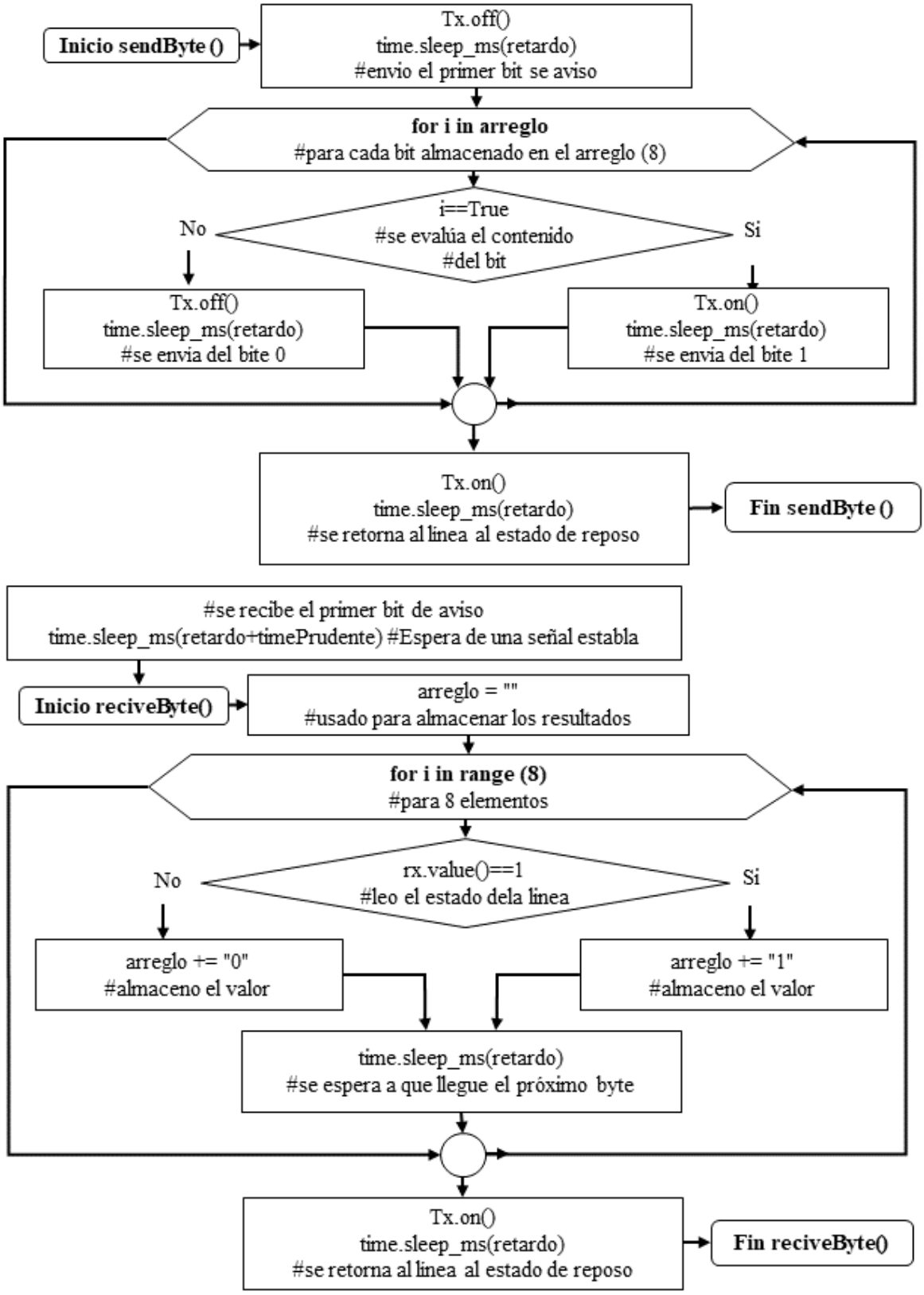


Mientras no se está transmitiendo información la línea se mantiene en un estado de reposo (idle), para comenzar la transmisión es necesario generar una primera señal de aviso (start), el receptor al detectar esta señal se prepara y sincroniza (tanto el emisor como el receptor deben estar configurado para funcionar a una misma frecuencia en la señal, tener el mismo tiempo de duración de cada bit), una vez sincronizado, se van realizando las lecturas a intervalos de tiempos adecuados, hasta obtener los 8 bits, luego de esto, la línea retorna nuevamente al estado de reposo, hasta que se vuelva a enviar otro byte de información.



Los pulsadores conectados al pico emisor, se conectaron en configuración pull down, el pico posee resistencias internas parar implementar esta configuración, que son utilizadas con ayuda de la librería machine.'

Diagramas de estado



Circuito Utilizado

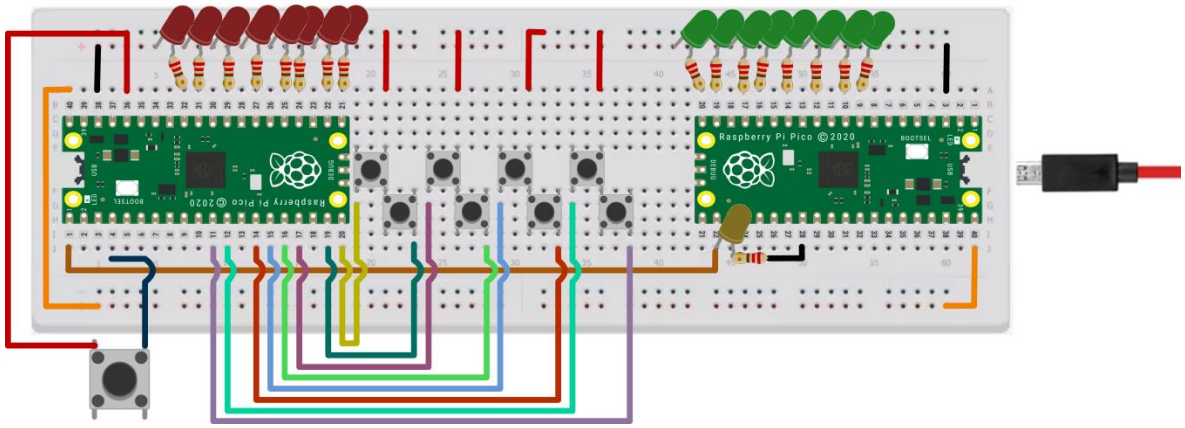
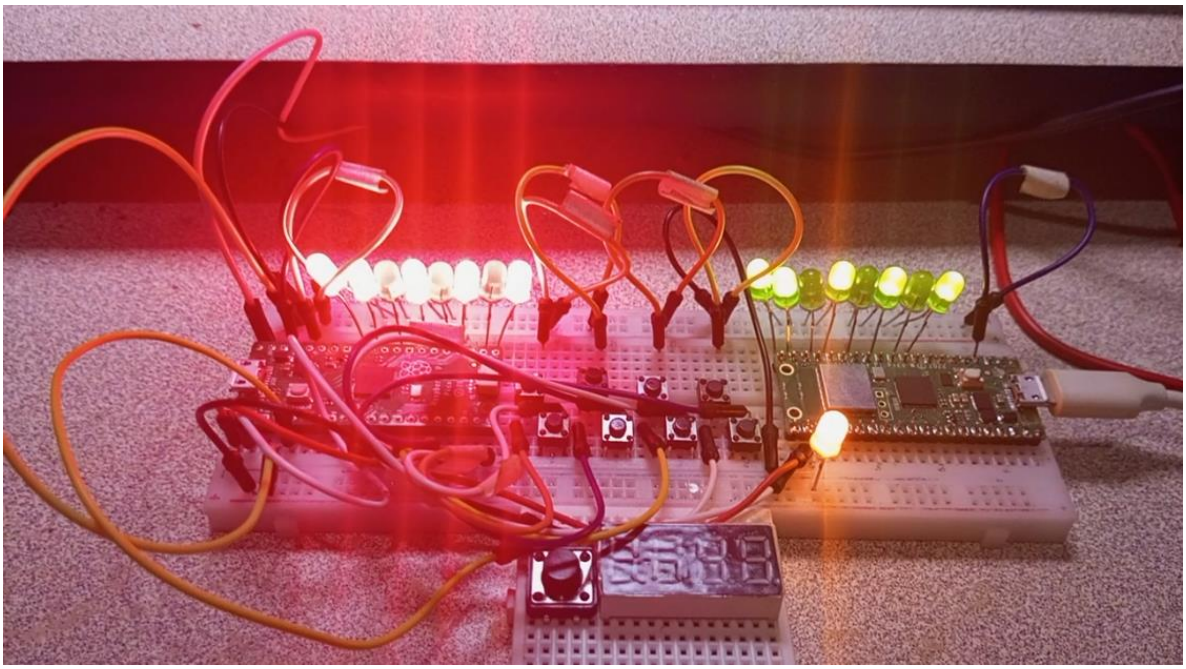


Imagen del Circuito



Nota: El uso de resistencia es importante dado que regula el flujo de corriente y evita un consumo excesivo de energía, que puede perjudicar el puerto usb de un computador.

Código del emisor

```
import machine
import time

espera = 0.3 #tiempo de demora para las pulsacione de los botones en segundos
retardo = 1000 #tiempo de retardo en la señal uart, en el envio de cada bit (en ms)
pinTx = 0 #pin para el envio de la informacion
tx = machine.Pin(pinTx,machine.Pin.OUT)


buttonSend = machine.Pin(3,machine.Pin.IN, machine.Pin.PULL_DOWN)

button0 = machine.Pin(8,machine.Pin.IN, machine.Pin.PULL_DOWN)
button1 = machine.Pin(9,machine.Pin.IN, machine.Pin.PULL_DOWN)
button2 = machine.Pin(10,machine.Pin.IN, machine.Pin.PULL_DOWN)
button3 = machine.Pin(11,machine.Pin.IN, machine.Pin.PULL_DOWN)
button4 = machine.Pin(12,machine.Pin.IN, machine.Pin.PULL_DOWN)
button5 = machine.Pin(13,machine.Pin.IN, machine.Pin.PULL_DOWN)
button6 = machine.Pin(14,machine.Pin.IN, machine.Pin.PULL_DOWN)
button7 = machine.Pin(15,machine.Pin.IN, machine.Pin.PULL_DOWN)

led0 = machine.Pin(16,machine.Pin.OUT)
led1 = machine.Pin(17,machine.Pin.OUT)
led2 = machine.Pin(18,machine.Pin.OUT)
led3 = machine.Pin(19,machine.Pin.OUT)
led4 = machine.Pin(21,machine.Pin.OUT)
led5 = machine.Pin(22,machine.Pin.OUT)
led6 = machine.Pin(26,machine.Pin.OUT)
led7 = machine.Pin(27,machine.Pin.OUT)

isOn0 = False
isOn1 = False
isOn2 = False
isOn3 = False
isOn4 = False
isOn5 = False
isOn6 = False
isOn7 = False

entrada = [button0,button1,button2,button3,button4,button5,button6,button7]
salida = [led0,led1,led2,led3,led4,led5,led6,led7]
estadoLed = [isOn0,isOn1,isOn2,isOn3,isOn4,isOn5,isOn6,isOn7]

```

```

#para asegurar que todos los leds esten apagados antes de iniciar
def turnOffLed():
    for i in range (8):
        salida[i].value(0)

```

```
43 def sendByte(char):
44    #arreglo = completar_ceros(decimal_to_binary(ord(char)))
45     arreglo = char
46     print('enviando informacion'+str(arreglo))
47     tx.off()#se envia el primer bit de aviso
48 
49     print('primer bit de aviso')
50     time.sleep_ms(retardo)
51     #se envia los siguientes 8 bits
52     contador = 0
53     for i in arreglo:
54         #dependiendo del valor, se cambia el estado del la linea
55         if(i==True):
56             print('enviando el bit ' + str(contador) + ' con el valor '+ str(i))
57             tx.on()
58             time.sleep_ms(retardo)
59         else:
60             print('enviando el bit ' + str(contador) + ' con el valor '+ str(i))
61             tx.off()
62             time.sleep_ms(retardo)
63         contador += contador
64     pinTx.on()#estado de reposo
65     time.sleep_ms(retardo)#tiempo prudente antes de poder enviar otro byte
```

```
72 def emisor():
73     turnOffLed()
74     while True:
75         #se recorre las entradas para saber si algun boton a sido precionado
76         for i in range (8):
77             if entrada[i].value()==1:
78                 estadoLed[i] = not estadoLed[i]
79                 salida[i].value(estadoLed[i])
80                 print('el boton '+ str(i) + ' a sido precionado')
81                 time.sleep(espera)
82         #si el boton de envio se preciono, se procede a enviar la informacio
83         if buttonSend.value()==1:
84             print('enviando informacion')
85             time.sleep(espera)
86             sendByte(estadoLed)
```

Código Receptor

```
1  import machine
2  import time
3  
4  espera = 0.3 #tiempo de demora para las pulsaciones de los botones en segundos
5  retardo = 1000 #tiempo de retardo en la señal uart, entre el envio de cada bit (en ms)
6  prudentTime = int(retardo/2)
7  pinRx = 17 #pin para la recepcion de informacion
8  rx = machine.Pin(pinRx,machine.Pin.IN, machine.Pin.PULL_DOWN)#inicializando el pin rx
9  
10 #inicializando los 8 leds para mostrar el valor de cada bit
11 led0 = machine.Pin(6,machine.Pin.OUT)
12 led1 = machine.Pin(7,machine.Pin.OUT)
13 led2 = machine.Pin(9,machine.Pin.OUT)
14 led3 = machine.Pin(10,machine.Pin.OUT)
15 led4 = machine.Pin(12,machine.Pin.OUT)
16 led5 = machine.Pin(13,machine.Pin.OUT)
17 led6 = machine.Pin(14,machine.Pin.OUT)
18 led7 = machine.Pin(15,machine.Pin.OUT)
```

```
19
20  #cargando las de claraciones en un unico arreglo
21  salida = [led0,led1,led2,led3,led4,led5,led6,led7]
22
23  def reciveByte():
24      array = ""
25      #se evalua y almacena el resultado
26      for i in range (8):
27          if rx.value()==1:
28              array += "1"
29          else:
30              array += "0"
31          #se espera a que lelgue el proximo byte
32          time.sleep_ms(retardo)
33      return array
34
35  #para asegurar que todos los leds esten apagados antes de iniciar
36  def turnOffLed():
37      for i in range (8):
38          salida[i].value(0)
39
38          salida[i].value(0)
39
40  def receiver():
41      turnOffLed()
42      while True:
43          if rx.value()==0:
44              print('Recibiendo informacion')
45              #espera de tiempo prudente para empezar la lectura en un punto estable
46              time.sleep_ms(retardo)
47              time.sleep_ms(prudentTime)
48              #carga del byte
49              dateReceiver=reciveByte()
50              print('orden de los datos recibidos'+str(datoEviar))
51              #utilizo la informazion para encernder los leds
52              for i in range(len(dateReceiver)):
53                  if dateReceiver[i]=='1':
54                      salida[i].value(1)
55                  else:
56                      salida[i].value(0)
```

Código alojado en:

https://github.com/EulisesBrazon/UART_recreation

Información Referencial:

COMUNICACIONES - ¿Cómo funcionan?. Recuperado de: https://youtu.be/J63_7AcvvMU

Entendiendo UART. Recuperado de: https://www.rohde-schwarz.com/lat/productos/prueba-y-medicion/essentials-test-equipment/digital-oscilloscopes/entendiendo-el-uart_254524.html#gallery-8