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

Transmisión de información entre dos Pi Pico W

Prof. Rene Cabrera

Elaborado por:

Brazon Eulises

Barcelona, 4 de Marzo del 2023

ABSTRACCION

La presente práctica está basada en una práctica anterior denominada ¨ *Servidor con Raspberry*¨. El cual se encuentra alojado en https://github.com/EulisesBrazon/server_picoW, en él se explica los diferentes elementos que deben ser considerados al momento de la creación de un servidor web con el raspberry pi pico W. Esta práctica tiene la finalidad de entablar una comunicación entre dos raspberry pi pico W

OBJETIVOS

Objetivo General:

- Entablar una comunicación entre dos Rapsberry pi pico W

Objetivos Específicos

- Crear un servidor web con su propia red wifi.
- Conectarse a la red wifi desde un pico cliente.
- Enviar información de temperatura, a través de la red wifi para visualizarla en una pantalla.

MATERIALES REQUERIDOS

- Jumpers para la conexión
- Protoboard.
- Modulo RGB
- Pantalla LCD TFT 1,8 pulgadas chip de control ST7735, de 128X160px.
- Pantalla LCD TFT 1,8 pulgadas chip de control ST7735, de 128X160px.
- Sensor LM35
- 2 Raspberry pi pico w
- Alimentación 5V.
- Herramientas para la manipulación y testeo (multímetro, pinzas, entre otros).
-

DESCRIPCION DEL PROYECTO

En la práctica anterior se vio como a través de un navegador web, se puede acceder a una pequeña página web alojada en el raspberry pi pico W, para controlar una luz rgb, a esta red se le agrega un nuevo pico W cliente, que le estará enviando datos de temperatura al pico servidor, para mostrar esta información a través de una pequeña pantalla. Además de poder manipular el estado de encendido y apagado de los les, cada pulso de información será controlado a través de botones

El pico W servidor, genera su propia red wifi, a la cual los dispositivos deberán conectarse para acceder a él, esto se logra mediante el uso de la librería network, donde en sus parámetros de inicialización cuenta con un modo de apertura en configuración accest Point denominado network.AP_IF,

Con ayuda de la consulta generada mediante la librería urequests.post(url), se enviando los diferentes comandos, tanto para la manipulación del led, como para el envió de la información de la temperatura que es censando, para que el servidor entienda esta información, la petición debe estar identificada correctamente, el servidor al detectar el tipo de petición, realizara lois respectivos procedimientos para cada tipo de consulta.

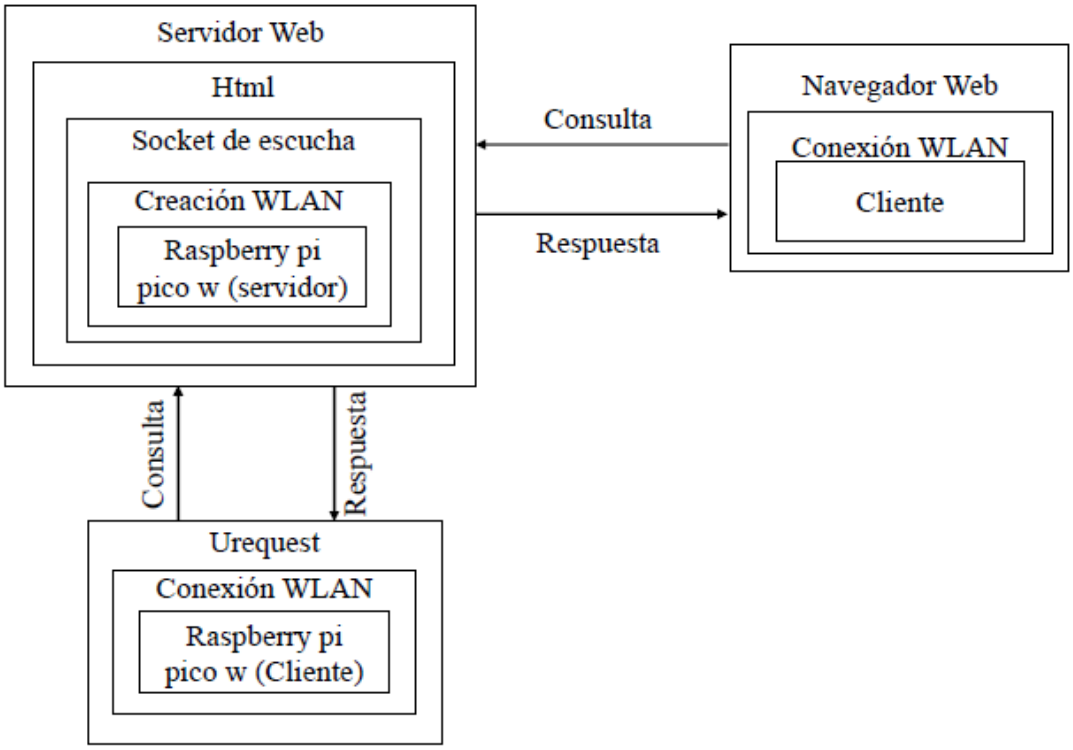
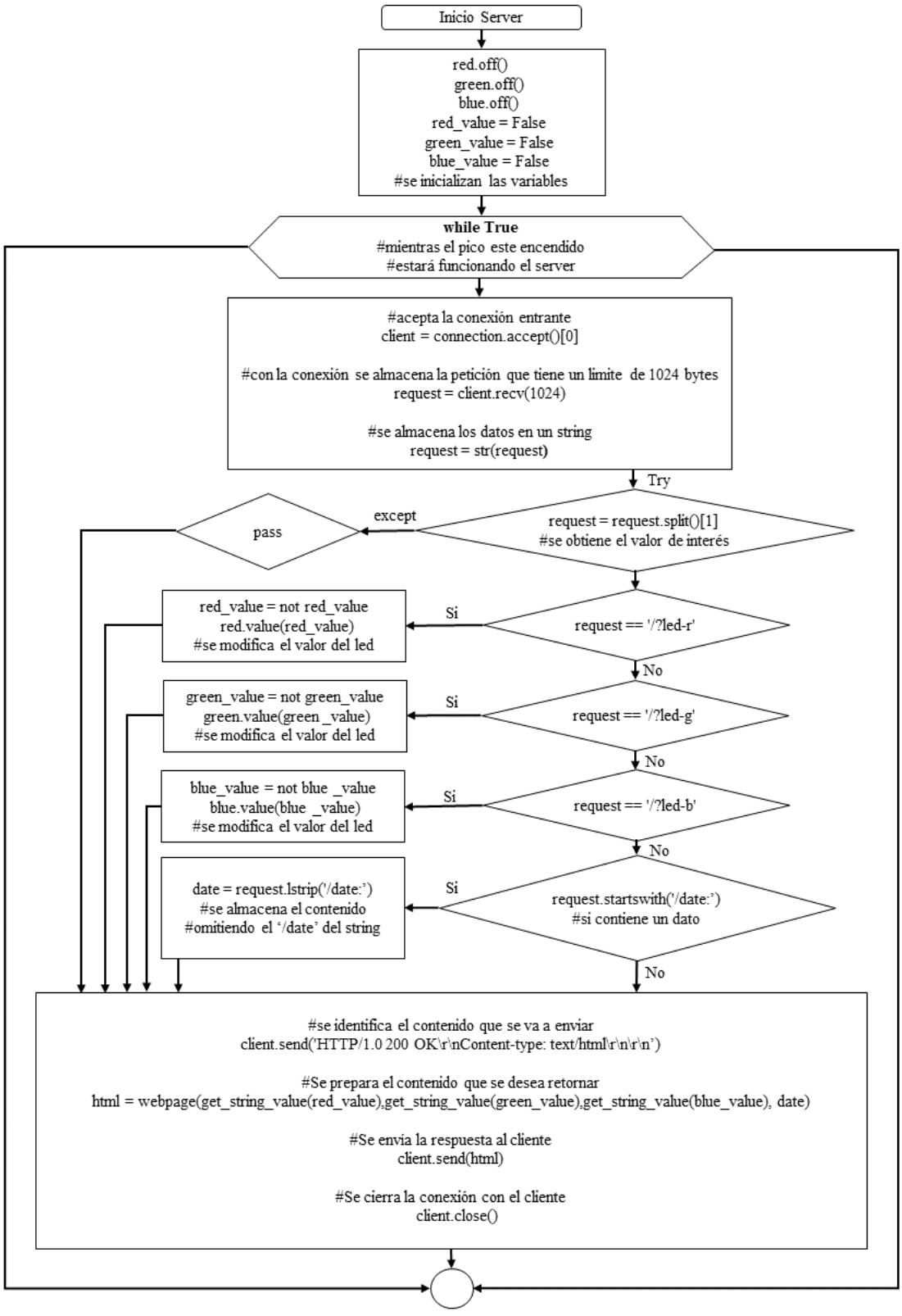
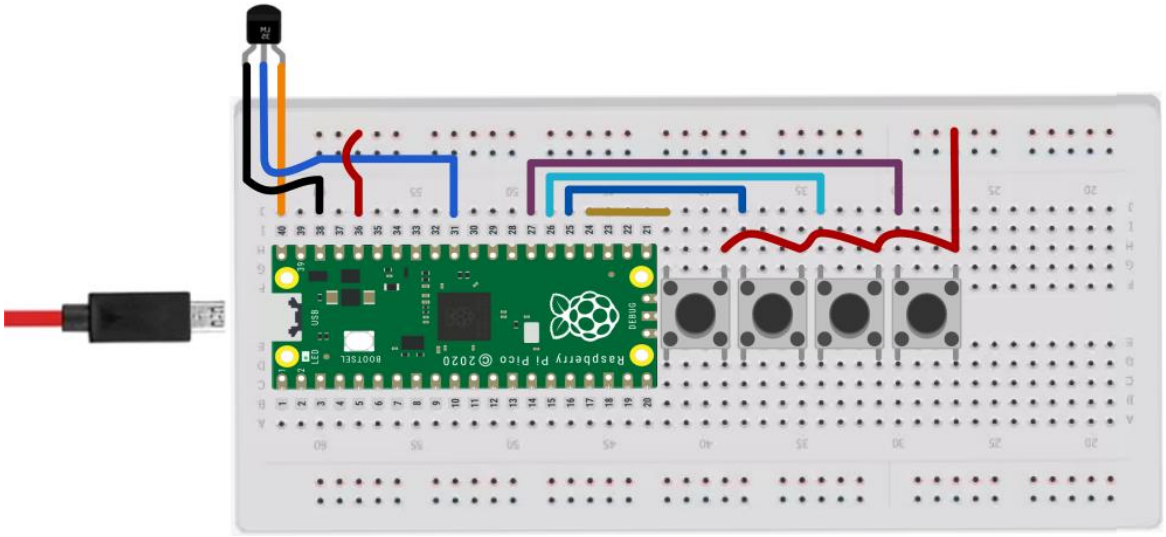


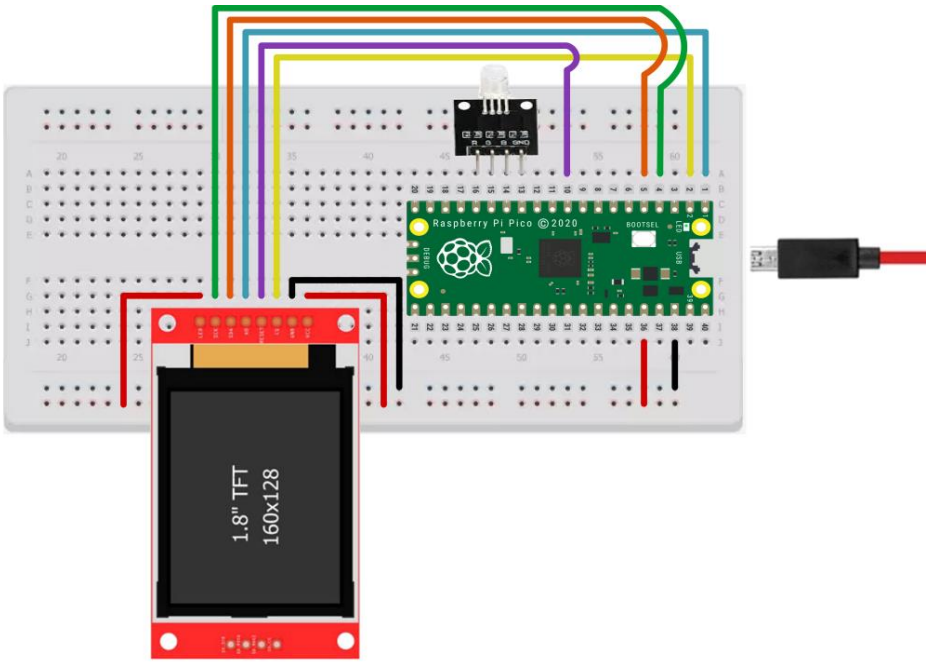
Diagrama de Flujo



Circuito Utilizado (Cliente)



Circuito Utilizado (Servidor)



Código de las funciones modificadas al Servidor del código del informe anterior

```
#displey
from sysfont import sysfont
from machine import SPI
from ST7735 import TFT
#other tools
from time import sleep
from machine import Pin
import gc
gc.enable()

def accesPoint():
    wlan = network.WLAN(network.AP_IF) #se modifica el modo de apertura
    wlan.config(essid=ssid, password=password)
    wlan.active(True)
    while wlan.active() == False:
        print('Create connection...')
        sleep(1)
    name_ssid = wlan.config('ssid')
    print(f'Acces Point {name_ssid} Created ')
    ip = wlan.ifconfig()[0]
    print(f'Connected on {ip}')
    led.on()
    return ip

# se agrega un nuevo parámetro a mostrar (date)
def webpage(isOnRed, isOnGreen, isOnBlue, date):
    #Template HTML
    html = """
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>RGB Controller</title>
    <style>
        html {
                font-family: Helvetica;
                display: inline-block;
                margin: 0px auto;
                text-align: center;}
            h1 {
                color: #0F3376;
                padding: 2vh;}
            p {font-size: 1.5rem;}
            table {margin: auto;          }
```

```
            td{padding: 10px ;}
            .Button {
               border-radius: 31px;
               display: inline-block;
               cursor: pointer;
               color: #ffffff;
               font-family: Arial;
               font-size: 17px;
               font-weight: bold;
               font-style: italic;
               padding: 17px 19px;
               text-decoration: none;          }
            .ButtonR {
               background-color: #ec4949;
               border: 6px solid #991f1f;
               text-shadow: 0px 2px 2px #471e1e;}
            .ButtonR:hover {background-color: #f51616;}
            .Button:active {
               position: relative;
               top: 1px;}
            .ButtonG {
               background-color: #49ec56;
               border: 6px solid #23991f;
               text-shadow: 0px 2px 2px #1e4723;}
            .ButtonG:hover {
               background-color: #29f516;}
            .ButtonB {
               background-color: #4974ec;
               border: 6px solid #1f3599;
               text-shadow: 0px 2px 2px #1e2447              }
            .ButtonB:hover {background-color: #165df5;}
   </style>
</head>
<body>
   <h1>Raspberry Pi Pico W Web Server</h1>
            <p>RGB Control</p>

            <table>
               <tbody>
                  <tr>
                     <td>
                        <p><a href="/led-r"><button class="ButtonR
                                          Button">R</button></a></p>
                     </td>
                     <td>
                        <strong> """+ isOnRed +"""</strong>
                     </td>
```

```
                    </tr>
                    <tr>
                        <td>
                            <p><a href="/led-g"><button class="ButtonG
                                                Button">G</button></a></p>
                        </td>
                        <td>
                            <strong> """+ isOnGreen +"""</strong>
                        </td>
                    </tr>
                    <tr>
                        <td>
                            <p><a href="/led-b"><button class="ButtonB
                                        Button">B</button></a></p>
                        </td>
                        <td>
                        <strong> """+ isOnBlue +""" </strong>
                        </td>
                    </tr>
                    <tr>
                        <td>
                            <p>Temperatura: </p> # campo para mostrar los datos desde la pagina
                        </td>
                        <td>
                        <strong> """+ date +""" </strong>
                        </td>
                    </tr>
                </tbody>
            </table>
        “
</body>
<footer>
    """+ loadFile() +"""
</footer>
</html>

        """
    return str(html)

def startDisplay(date):

    size=2
    separation = 4
    h=0
    #data reading
    date = str(date)
    tft.fill(TFT.BLACK)#clean screen
```

```
    tft.text((0, h), "IP: ", TFT.RED, sysfont, size, nowrap=True)
    h += sysfont["Height"]*size+separation
    tft.text((0, h), str(ip), TFT.GREEN, sysfont, size, nowrap=True)
    h += sysfont["Height"]*size+separation
    tft.text((0, h), "SSID: ", TFT.RED, sysfont, size, nowrap=True)
    h += sysfont["Height"]*size+separation
    tft.text((0, h), ssid, TFT.GREEN, sysfont, size, nowrap=True)
    h += sysfont["Height"]*size+separation
    tft.text((0, h), "Temperatura: ", TFT.RED, sysfont, size, nowrap=True)
    h += sysfont["Height"]*size+separation
    tft.text((0, h), date, TFT.GREEN, sysfont, size, nowrap=True)

def refreshDisplay(date):
    global oldDate
    size=2
    #data reading
    date = str(date)
    tft.text((0, 100), oldDate, TFT.BLACK, sysfont, size, nowrap=True)
    tft.text((0, 100), date, TFT.GREEN, sysfont, size, nowrap=True)
    oldDate = date

def serve(connection):
    #Start a web server
    global date

    red.off()
    green.off()
    blue.off()

    red_value = False
    green_value = False
    blue_value = False


    while True:
        client = connection.accept()[0]
        request = client.recv(1024)
        request = str(request)
        try:
            request = request.split()[1]
        except IndexError:
            pass

        print(f'Request {request}')
        if request == '/led-r':
            red_value = not red_value
            pave = red_value
```

```
            red.value(red_value)
        elif request == '/led-g':
            green_value = not green_value
            green.value(green_value)
        elif request == '/led-b':
            blue_value = not blue_value
            blue.value(blue_value)
        elif request.startswith('/date:'): #si la peticion contiene un dato
            date = request.lstrip('/date:')# almacena el contenido ignorando date del string

        html webpage(get_string_value(red_value),get_string_value(green_value),
                     get_string_value(blue_value), date)
        client.send('/1.0 200 OK\r\nContent-type: text/html\r\n\r\n')
        client.send(html)
        client.close()
        refreshDisplay(date)

def runServer():
    try:
        global ip
        global date
        ip = accesPoint()
        connection = open_socket(ip)
        startDisplay(date)
        serve(connection)
    except KeyboardInterrupt:
        machine.reset()
runServer()
```

Código (Cliente)

```
#server tools
import network
try:
  import usocket as socket
except:
  import socket
import urequests
#other tools
from machine import Pin
from time import sleep
from machine import SPI,Pin,ADC
import gc
gc.enable()

#define value for use LM35
analog_value = ADC(26)
#define pin RGB

def readTemperature():
    conversion_factor = 3.3/ 65535
    temp_voltage_raw =
analog_value.read_u16()
    convert_voltage =
temp_voltage_raw*conversion_factor
    tempC = convert_voltage/(10.0 / 1000)
    return tempC
```

```
buttonRed = machine.Pin(21,machine.Pin.IN, machine.Pin.PULL_DOWN)
buttonGreen = machine.Pin(20,machine.Pin.IN, machine.Pin.PULL_DOWN)
buttonBlue = machine.Pin(19,machine.Pin.IN, machine.Pin.PULL_DOWN)
buttonTemp = machine.Pin(18,machine.Pin.IN, machine.Pin.PULL_DOWN)

delay = 0.3
def buttons():

    if buttonRed.value()==1:
        print("Red")
        http_get("led-r")
        sleep(delay)

    if buttonGreen.value()==1:
        print("Green")
        http_get("led-g")
        sleep(delay)

    if buttonBlue.value()==1:
        print("Blue")
        http_get("led-b")
        sleep(delay)
    if buttonTemp.value()==1:
        sentTemp()
        sleep(delay)
def sentTemp():
    temp = readTemperature()
    print(temp)
    saveFile(temp)
    http_get('date:'+str(temp))
def runClient():
    try:
        conenection = connect()
        while True:
            buttons()
    except KeyboardInterrupt:
        machine.reset()
runClient()
```

```
def http_get(path: str):
    try:
        url = 'http://192.168.4.1/'+path
        response = urequests.post(url)

        print(response)
        if response is not None:
            response.close()
    except Exception as e:
        print(e)

def connect():

    #Connect to WLAN
    wlan = network.WLAN(network.STA_IF)
    wlan.active(True)
    wlan.connect(ssid, password)
    while wlan.isconnected() == False:
        print('Waiting for connection...')
        sleep(1)
    ip = wlan.ifconfig()[0]
    print(f'Connected on {ip}')
    return ip
```

Código completo alojado en:

https://github.com/EulisesBrazon/server_picoW

Información Referencial:

Sensors_and_display_UART. Recuperado de:https://github.com/EulisesBrazon/sensors_and_display_UART

Raspberry Pi Pico W :08 web server - WIFI - RGB control. Recuperado de :https://youtu.be/606NxqqJr9I

Getting started with your Raspberry Pi Pico W. Recuperado de: https://projects.raspberrypi.org/en/projects/get-started-pico-w/0