Universidad del Valle de Guatemala
Electrónica Digital 2
Proyecto 3

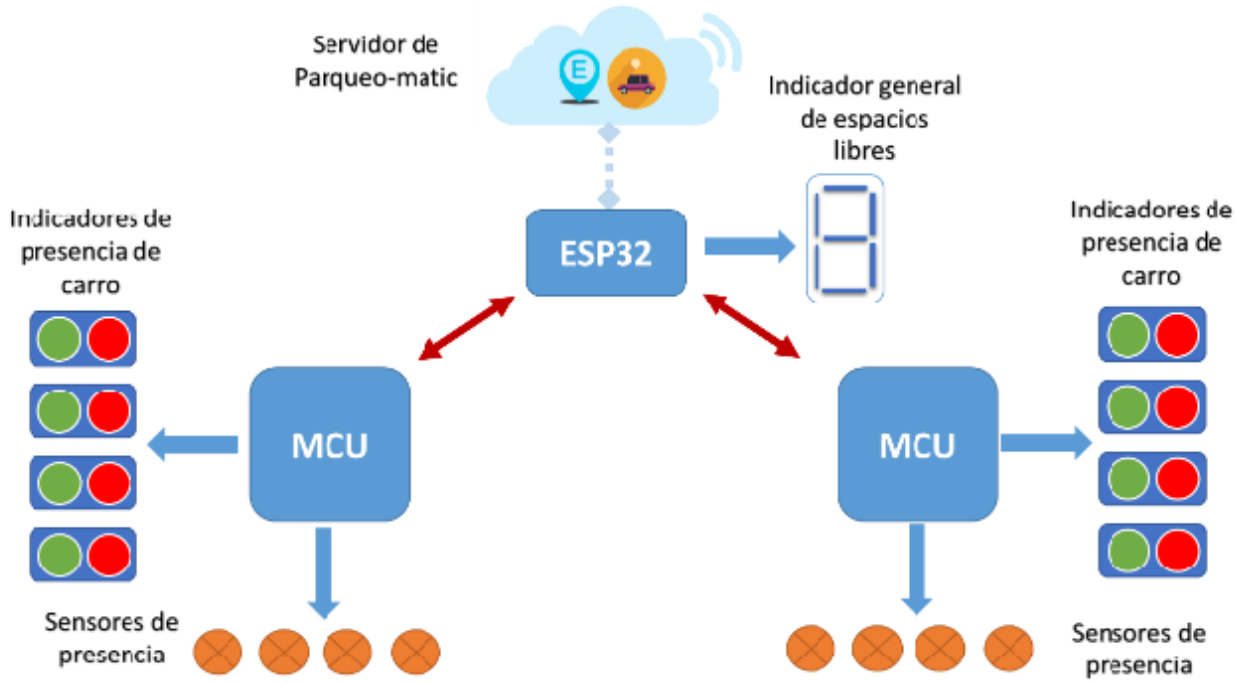
Estudiantes:
- Pablo Andres Figueroa Gámez 21775
- Sebastián Mayén Dávila 21215

Introducción:
En el presente trabajo se detalla la información y análisis de los circuitos que se realizaron, en conjunto a la explicación del código utilizado para el correcto funcionamiento del proyecto 3 del curso digital 2 de la Universidad del Valle de Guatemala el cual consistió en la realización de un sistema de control de estacionamiento.

Componentes utilizados:

- TM4C123 Tiva C Launch Pad
- Display 7 segmentos
- LED´s
- Fotoresistores (LDR)
- Resistencias varias.

Circuito realizado:



Como se puede observar en el diagrama anterior, donde se tiene MCU, se utilizó como Controlador la TM4C123 Tiva C Launch Pad. Cada Tiva C tiene conectados 4 sensores de presencia los cuales son las fotorresistencias (LDR) y los indicadores de presencia de carro los cuales son 2 leds por sensor de presencia. Ambas Tiva C se conectan al ESP32 por comunicación UART y el ESP32 se encuentra configurado como servidor, el cual tiene la funcionalidad de obtener el dato de parqueos disponibles en cada una de las Tiva C y luego lo transmite a la página web realizada en HTML donde se puede observar el indicador general de espacios libres.

Código Tiva C #1:

```
#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/sysctl.h"
#include "driverlib/gpio.h"
#include "inc/tm4c123gh6pm.h"
#include "driverlib/interrupt.h"
#include "driverlib/timer.h"
#include "driverlib/systick.h"
#include "driverlib/uart.h"
#include "driverlib/pin_map.h"

int i;

// Funciones para controlar los LEDs
void toggleRedLED(uint32_t index, bool state);
void toggleGreenLED(uint32_t index, bool state);
void InitUART(void);

int main(void)
{
   // Configura el sistema para 16MHz
            SysCtlClockSet(SYSCTL_SYSDIV_5 | SYSCTL_USE_PLL | SYSCTL_OSC_MAIN |
SYSCTL_XTAL_16MHZ);

   SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);  // Habilita el reloj para el puerto B
   while(!SysCtlPeripheralReady(SYSCTL_PERIPH_GPIOB));  // Espera a que el periférico esté listo

   // Se inicializa la comunicación UART
   InitUART();

   // Habilita los periféricos GPIO, los botones y los LEDs
   SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);
   SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOE);
   SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);
```

```
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOC);

    while (!SysCtlPeripheralReady(SYSCTL_PERIPH_GPIOD) ||
           !SysCtlPeripheralReady(SYSCTL_PERIPH_GPIOE) ||
           !SysCtlPeripheralReady(SYSCTL_PERIPH_GPIOB) ||
           !SysCtlPeripheralReady(SYSCTL_PERIPH_GPIOA) ||
           !SysCtlPeripheralReady(SYSCTL_PERIPH_GPIOC)) {}

    // Configura los pines GPIO para los LEDs rojos en PD0, PD1, PD2 y PD3
       GPIOPinTypeGPIOOutput(GPIO_PORTD_BASE,  GPIO_PIN_0 | GPIO_PIN_1 | GPIO_PIN_2 |
GPIO_PIN_3);

    // Configura los pines GPIO para los LEDs verdes en PE4, PE5, PB4 y PA5
    GPIOPinTypeGPIOOutput(GPIO_PORTE_BASE, GPIO_PIN_4 | GPIO_PIN_5);
    GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_4);
    GPIOPinTypeGPIOOutput(GPIO_PORTA_BASE, GPIO_PIN_5);

    // Configura los pines GPIO para los botones en PC4, PC5, PC6 y PC7
        GPIOPinTypeGPIOInput(GPIO_PORTC_BASE,  GPIO_PIN_4 | GPIO_PIN_5 | GPIO_PIN_6 |
GPIO_PIN_7);
          GPIOPadConfigSet(GPIO_PORTC_BASE,  GPIO_PIN_4  |  GPIO_PIN_5  |  GPIO_PIN_6  |
GPIO_PIN_7, GPIO_STRENGTH_2MA, GPIO_PIN_TYPE_STD_WPU);

    // Inicialmente, todos los LEDs verdes están encendidos
    toggleGreenLED(0, true);
    toggleGreenLED(1, true);
    toggleGreenLED(2, true);
    toggleGreenLED(3, true);

    // Variables para controlar el estado de los botones
      bool buttonState[4] = {true, true, true, true}; // Inicialmente, todos los botones están en estado
soltado

    while (1)
    {
       // Evalúa el estado de los botones y controla los LEDs de forma independiente
       for (i = 0; i < 4; i++)
       {
          bool isPressed = GPIOPinRead(GPIO_PORTC_BASE, 1 << (i + 4)) == 0;
          if (isPressed && buttonState[i])
          {
             // Botón presionado
             buttonState[i] = false;
             toggleRedLED(i, true);
             toggleGreenLED(i, false);
```

```
    // Lógica para el botón 0
    if (i == 0)
    {
        UARTCharPut(UART1_BASE, '0');
    }
    // Lógica para el botón 1
    else if (i == 1)
    {
        UARTCharPut(UART1_BASE, '2');
    }
    // Lógica para el botón 2
    else if (i == 2)
    {
        UARTCharPut(UART1_BASE, '4');
    }
    // Lógica para el botón 3
    else if (i == 3)
    {
        UARTCharPut(UART1_BASE, '6');
    }
}
else if (!isPressed && !buttonState[i])
{
    // Botón soltado
    buttonState[i] = true;
    toggleRedLED(i, false);
    toggleGreenLED(i, true);

    // Lógica para el botón 0 cuando se suelta
    if (i == 0)
    {
        UARTCharPut(UART1_BASE, '1');
    }
    // Lógica para el botón 1 cuando se suelta
    else if (i == 1)
    {
        UARTCharPut(UART1_BASE, '3');
    }
    // Lógica para el botón 2 cuando se suelta
    else if (i == 2)
    {
        UARTCharPut(UART1_BASE, '5');
    }
    // Lógica para el botón 3 cuando se suelta
    else if (i == 3)
```

```
            {
                UARTCharPut(UART1_BASE, '7');
            }
        }
    }
}

// Función para controlar los LEDs rojos
void toggleRedLED(uint32_t index, bool state)
{
    if (state)
            GPIOPinWrite(GPIO_PORTD_BASE, 1 << index, 1 << index); // Enciende el LED rojo correspondiente
    else
        GPIOPinWrite(GPIO_PORTD_BASE, 1 << index, 0); // Apaga el LED rojo correspondiente
}

// Función para controlar los LEDs verdes
void toggleGreenLED(uint32_t index, bool state)
{
    switch (index)
    {
        case 0:
            GPIOPinWrite(GPIO_PORTE_BASE, GPIO_PIN_4, state ? GPIO_PIN_4 : 0);
            break;
        case 1:
            GPIOPinWrite(GPIO_PORTE_BASE, GPIO_PIN_5, state ? GPIO_PIN_5 : 0);
            break;
        case 2:
            GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_4, state ? GPIO_PIN_4 : 0);
            break;
        case 3:
            GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_5, state ? GPIO_PIN_5 : 0);
            break;
        default:
            break;
    }
}
void InitUART(void)
{
    /Enable the GPIO Port A/
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);

    /Enable the peripheral UART Module 0/
    SysCtlPeripheralEnable(SYSCTL_PERIPH_UART1);
```

```
   /* Make the UART pins be peripheral controlled. */
    GPIOPinConfigure(GPIO_PB0_U1RX);  // Configura los pines PA0 y PA1 para ser utilizados como pines de UART
    GPIOPinConfigure(GPIO_PB1_U1TX);  // Configura los pines PA0 y PA1 para ser utilizados como pines de UART
   GPIOPinTypeUART(GPIO_PORTB_BASE, GPIO_PIN_0 | GPIO_PIN_1);

   /* Sets the configuration of a UART. */
   UARTConfigSetExpClk(
      UART1_BASE, SysCtlClockGet(), 115200,
      (UART_CONFIG_WLEN_8 | UART_CONFIG_STOP_ONE | UART_CONFIG_PAR_NONE));
}
```

Código Tiva C #2:

```
#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/sysctl.h"
#include "driverlib/gpio.h"
#include "inc/tm4c123gh6pm.h"
#include "driverlib/interrupt.h"
#include "driverlib/timer.h"
#include "driverlib/systick.h"
#include "driverlib/uart.h"
#include "driverlib/pin_map.h"

int i;

// Funciones para controlar los LEDs
void toggleRedLED(uint32_t index, bool state);
void toggleGreenLED(uint32_t index, bool state);
void InitUART(void);

int main(void)
{
   // Configura el sistema para 16MHz
            SysCtlClockSet(SYSCTL_SYSDIV_5  |  SYSCTL_USE_PLL  |  SYSCTL_OSC_MAIN  |  SYSCTL_XTAL_16MHZ);

   SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);  // Habilita el reloj para el puerto B
   while(!SysCtlPeripheralReady(SYSCTL_PERIPH_GPIOB));  // Espera a que el periférico esté listo
```

```
    // Se inicializa la comunicación UART
    InitUART();

    // Habilita los periféricos GPIO, los botones y los LEDs
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOE);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOC);

    while (!SysCtlPeripheralReady(SYSCTL_PERIPH_GPIOD) ||
           !SysCtlPeripheralReady(SYSCTL_PERIPH_GPIOE) ||
           !SysCtlPeripheralReady(SYSCTL_PERIPH_GPIOB) ||
           !SysCtlPeripheralReady(SYSCTL_PERIPH_GPIOA) ||
           !SysCtlPeripheralReady(SYSCTL_PERIPH_GPIOC)) {}

    // Configura los pines GPIO para los LEDs rojos en PD0, PD1, PD2 y PD3
       GPIOPinTypeGPIOOutput(GPIO_PORTD_BASE,  GPIO_PIN_0 | GPIO_PIN_1 | GPIO_PIN_2 |
GPIO_PIN_3);

    // Configura los pines GPIO para los LEDs verdes en PE4, PE5, PB4 y PA5
    GPIOPinTypeGPIOOutput(GPIO_PORTE_BASE, GPIO_PIN_4 | GPIO_PIN_5);
    GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_4);
    GPIOPinTypeGPIOOutput(GPIO_PORTA_BASE, GPIO_PIN_5);

    // Configura los pines GPIO para los botones en PC4, PC5, PC6 y PC7
        GPIOPinTypeGPIOInput(GPIO_PORTC_BASE,  GPIO_PIN_4 | GPIO_PIN_5 | GPIO_PIN_6 |
GPIO_PIN_7);
           GPIOPadConfigSet(GPIO_PORTC_BASE,  GPIO_PIN_4 | GPIO_PIN_5 | GPIO_PIN_6 |
GPIO_PIN_7, GPIO_STRENGTH_2MA, GPIO_PIN_TYPE_STD_WPU);

    // Inicialmente, todos los LEDs verdes están encendidos
    toggleGreenLED(0, true);
    toggleGreenLED(1, true);
    toggleGreenLED(2, true);
    toggleGreenLED(3, true);

    // Variables para controlar el estado de los botones
       bool buttonState[4] = {true, true, true, true}; // Inicialmente, todos los botones están en estado
soltado

    while (1)
    {
        // Evalúa el estado de los botones y controla los LEDs de forma independiente
        for (i = 0; i < 4; i++)
        {
```

```
bool isPressed = GPIOPinRead(GPIO_PORTC_BASE, 1 << (i + 4)) == 0;
if (isPressed && buttonState[i])
{

    buttonState[i] = false;
    toggleRedLED(i, true);
    toggleGreenLED(i, false);

    // Lógica para fotorresistencia  0
    if (i == 0)
    {
        UARTCharPut(UART1_BASE, '8');
    }
    // Lógica para fotoresistencia 1
    else if (i == 1)
    {
        UARTCharPut(UART1_BASE, 'a');
    }
    // Lógica para fotorresistencia 2
    else if (i == 2)
    {
        UARTCharPut(UART1_BASE, 'c');
    }
    // Lógica para fotorresistencia 3
    else if (i == 3)
    {
        UARTCharPut(UART1_BASE, 'e');
    }
}
else if (!isPressed && !buttonState[i])
{

    buttonState[i] = true;
    toggleRedLED(i, false);
    toggleGreenLED(i, true);

    // lógica cuando deja de detectar
    if (i == 0)
    {
        UARTCharPut(UART1_BASE, '9');
    }

    else if (i == 1)
    {
        UARTCharPut(UART1_BASE, 'b');
    }
```

```
            else if (i == 2)
            {
               UARTCharPut(UART1_BASE, 'd');
            }

            else if (i == 3)
            {
               UARTCharPut(UART1_BASE, 'f');
            }
         }
      }
   }
}

// Función para controlar los LEDs rojos
void toggleRedLED(uint32_t index, bool state)
{
   if (state)
            GPIOPinWrite(GPIO_PORTD_BASE, 1 << index, 1 << index); // Enciende el LED rojo
correspondiente
   else
      GPIOPinWrite(GPIO_PORTD_BASE, 1 << index, 0); // Apaga el LED rojo correspondiente
}

// Función para controlar los LEDs verdes
void toggleGreenLED(uint32_t index, bool state)
{
   switch (index)
   {
      case 0:
         GPIOPinWrite(GPIO_PORTE_BASE, GPIO_PIN_4, state ? GPIO_PIN_4 : 0);
         break;
      case 1:
         GPIOPinWrite(GPIO_PORTE_BASE, GPIO_PIN_5, state ? GPIO_PIN_5 : 0);
         break;
      case 2:
         GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_4, state ? GPIO_PIN_4 : 0);
         break;
      case 3:
         GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_5, state ? GPIO_PIN_5 : 0);
         break;
      default:
         break;
   }
}
```

```
void InitUART(void)
{
   /Enable the GPIO Port A/
   SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);

   /Enable the peripheral UART Module 0/
   SysCtlPeripheralEnable(SYSCTL_PERIPH_UART1);

   /* Make the UART pins be peripheral controlled. */
    GPIOPinConfigure(GPIO_PB0_U1RX);  // Configura los pines PA0 y PA1 para ser utilizados como pines de UART
    GPIOPinConfigure(GPIO_PB1_U1TX);  // Configura los pines PA0 y PA1 para ser utilizados como pines de UART
   GPIOPinTypeUART(GPIO_PORTB_BASE, GPIO_PIN_0 | GPIO_PIN_1);

   /* Sets the configuration of a UART. */
   UARTConfigSetExpClk(
      UART1_BASE, SysCtlClockGet(), 115200,
      (UART_CONFIG_WLEN_8 | UART_CONFIG_STOP_ONE | UART_CONFIG_PAR_NONE));
}
```

Explicación Código:

Como se puede observar, anteriormente, el código en ámbas Tiva C es prácticamente el mismo, el único cambio es en la comunicación UART en donde el dato que se envía desde las Tiva C cambia para que el ESP32 pueda diferenciar que parqueo es en específico el que cambió de estado.

La comunicación UART se realiza con el RX0 & TX0 los cuales se encuentran en los pines PB0 y PB1 respectivamente. Los leds verdes se encuentran conectados a los pines PE4, PE5, PB4, PA5 y los leds rojos se encuentran conectados a los pines PD0, PD1, PD2, PD3 respectivamente de cada Tiva C. Estos pines se encuentran configurados como salidas digitales ya que su funcionalidad es cambiar únicamente de encendido a apagado y viceversa.

Los sensores en este caso son las fotorresistencias (LDR) las cuales se encuentran conectadas a los pines PC4, PC5, PC6 y PC7. Estos pines se configuran como entradas digitales ya que la fotoresistencia varía el valor de voltaje lo que produce un encendido o apagado (0 - 1 digital).

Las Tiva C constantemente están analizando el estado de las fotorresistencias y cuando cambia de estado manda por UART una bandera que en el ESP32 modifica el número de parqueos disponibles y luego el ESP32 actualiza el valor en la página web.

Código ESP32:

```
//*********************************
// Librerías
//*********************************
#include <WiFi.h>
#include <WebServer.h>
#include <HardwareSerial.h>
//*********************************
// Variables globales
//*********************************
// Definición de pines para el primer display (NIVEL 2)
#define DISPLAY1_A 13
#define DISPLAY1_B 12
#define DISPLAY1_C 14
#define DISPLAY1_D 27
#define DISPLAY1_E 26
#define DISPLAY1_F 25
#define DISPLAY1_G 23

// Definición de pines para el segundo display (NIVEL 1)
#define DISPLAY2_A 15
#define DISPLAY2_B 22
#define DISPLAY2_C 4
#define DISPLAY2_D 5
#define DISPLAY2_E 18
#define DISPLAY2_F 19
#define DISPLAY2_G 21

char datativa2;
char datativa1;
uint8_t Pstate;

int parqueo1 = 0;
int parqueo2 = 2;
int parqueo3 = 4;
int parqueo4 = 6;
int parqueo5 = 8;
int parqueo6 = 10;
int parqueo7 = 12;
int parqueo8 = 14;

int p1,p2,p3,p4,p5,p6,p7,p8 ;

int conteo = 0;
int conteo1 = 0;
```

```
int conteo2 = 0;

// SSID & Password
const char* ssid = "sebastian";  // Enter your SSID here
const char* password = "12345678";  //Enter your Password here

WebServer server(80);  // Object of WebServer(HTTP port, 80 is defult)

//*********************************
// Configuración
//*********************************
void setup() {
  Serial.begin(115200);     // Inicializa la comunicación serial en el puerto UART0
  Serial2.begin(115200);  // Inicializa la comunicación serial en el puerto UART2
  Serial.println("Try Connecting to ");
  Serial.println(ssid);

  // Connect to your wi-fi modem
  WiFi.begin(ssid, password);

  pinMode(DISPLAY1_A, OUTPUT);
  pinMode(DISPLAY1_B, OUTPUT);
  pinMode(DISPLAY1_C, OUTPUT);
  pinMode(DISPLAY1_D, OUTPUT);
  pinMode(DISPLAY1_E, OUTPUT);
  pinMode(DISPLAY1_F, OUTPUT);
  pinMode(DISPLAY1_G, OUTPUT);

  pinMode(DISPLAY2_A, OUTPUT);
  pinMode(DISPLAY2_B, OUTPUT);
  pinMode(DISPLAY2_C, OUTPUT);
  pinMode(DISPLAY2_D, OUTPUT);
  pinMode(DISPLAY2_E, OUTPUT);
  pinMode(DISPLAY2_F, OUTPUT);
  pinMode(DISPLAY2_G, OUTPUT);

  // Check wi-fi is connected to wi-fi network
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected successfully");
  Serial.print("Got IP: ");
  Serial.println(WiFi.localIP());  //Show ESP32 IP on serial
```

```
  server.on("/", handle_OnConnect); // Directamente desde e.g. 192.168.0.8
  server.on("/actualizar", handle_OnConnect);
  server.onNotFound(handle_NotFound);

  server.begin();
  Serial.println("HTTP server started");
  delay(100);
}

void mostrarNumero(int A, int B, int C, int D, int E, int F, int G, int display) {
  if (display == 2) {
    digitalWrite(DISPLAY1_A, A);
    digitalWrite(DISPLAY1_B, B);
    digitalWrite(DISPLAY1_C, C);
    digitalWrite(DISPLAY1_D, D);
    digitalWrite(DISPLAY1_E, E);
    digitalWrite(DISPLAY1_F, F);
    digitalWrite(DISPLAY1_G, G);
  } else if (display == 1) {
    digitalWrite(DISPLAY2_A, A);
    digitalWrite(DISPLAY2_B, B);
    digitalWrite(DISPLAY2_C, C);
    digitalWrite(DISPLAY2_D, D);
    digitalWrite(DISPLAY2_E, E);
    digitalWrite(DISPLAY2_F, F);
    digitalWrite(DISPLAY2_G, G);
  }
}
//*********************************
// loop principal
//*********************************
void loop() {
  server.handleClient();
  // Verifica si hay datos disponibles en UART0 (RX0)
  while (Serial.available()) {
    datativa1 = Serial.read();
  }

  switch (datativa1) {
  case '0':
    parqueo1 = 0;
    p1 = 0;
    break;
  case '1':
    parqueo1 = 1;
    p1 = 1;
```

```
    break;
  case '2':
    parqueo2 = 0;
    p2 = 0;
    break;
  case '3':
    parqueo2 = 1;
    p2 = 1;
    break;
  case '4':
    parqueo3 = 0;
    p3 = 0;
    break;
  case '5':
    parqueo3 = 1;
    p3 = 1;
    break;
  case '6':
    parqueo4 = 0;
    p4 = 0;
    break;
  case '7':
    parqueo4 = 1;
    p4 = 1;
    break;
}

  // Verifica si hay datos disponibles en UART2 (RX2)
  while (Serial2.available()) {
    datativa2 = Serial2.read();
  }

  switch (datativa2) {
  case '8':
    parqueo5 = 0;
    p5 = 0;
    break;
  case '9':
    parqueo5 = 1;
    p5 = 1;
    break;
  case 'a':
    parqueo6 = 0;
    p6 = 0;
    break;
  case 'b':
```

```
    parqueo6 = 1;
    p6 = 1;
    break;
  case 'c':
    parqueo7 = 0;
    p7 = 0;
    break;
  case 'd':
    parqueo7 = 1;
    p7 = 1;
    break;
  case 'e':
    parqueo8 = 0;
    p8 = 0;
    break;
  case 'f':
    parqueo8 = 1;
    p8 = 1;
    break;
}
  conteo1 = p1+p2+p3+p4;
  conteo2 = p5+p6+p7+p8;

if (conteo1 == 0) {
    mostrarNumero(1, 1, 1, 1, 1, 1, 0, 1); // Mostrar "0"
  } else if (conteo1 == 1) {
    mostrarNumero(0, 1, 1, 0, 0, 0, 0, 1); // Mostrar "1"
  } else if (conteo1 == 2) {
    mostrarNumero(1, 1, 0, 1, 1, 0, 1, 1); // Mostrar "2"
  } else if (conteo1 == 3) {
    mostrarNumero(1, 1, 1, 1, 0, 0, 1, 1); // Mostrar "3"
  } else if (conteo1 == 4) {
    mostrarNumero(0, 1, 1, 0, 0, 1, 1, 1); // Mostrar "4"
  } else if (conteo1 == 5) {
    mostrarNumero(1, 0, 1, 1, 0, 1, 1, 1); // Mostrar "5"
  } else if (conteo1 == 6) {
    mostrarNumero(1, 0, 1, 1, 1, 1, 1, 1); // Mostrar "6"
  } else if (conteo1 == 7) {
    mostrarNumero(1, 1, 1, 0, 0, 0, 0, 1); // Mostrar "7"
  } else if (conteo1 == 8) {
    mostrarNumero(1, 1, 1, 1, 1, 1, 1, 1); // Mostrar "8"
  }

  if (conteo2 == 0) {
    mostrarNumero(1, 1, 1, 1, 1, 1, 0, 2); // Mostrar "0"
  } else if (conteo2 == 1) {
```

```
    mostrarNumero(0, 1, 1, 0, 0, 0, 0, 2); // Mostrar "1"
  } else if (conteo2 == 2) {
    mostrarNumero(1, 1, 0, 1, 1, 0, 1, 2); // Mostrar "2"
  } else if (conteo2 == 3) {
    mostrarNumero(1, 1, 1, 1, 0, 0, 1, 2); // Mostrar "3"
  } else if (conteo2 == 4) {
    mostrarNumero(0, 1, 1, 0, 0, 1, 1, 2); // Mostrar "4"
  } else if (conteo2 == 5) {
    mostrarNumero(1, 0, 1, 1, 0, 1, 1, 2); // Mostrar "5"
  } else if (conteo2 == 6) {
    mostrarNumero(1, 0, 1, 1, 1, 1, 1, 2); // Mostrar "6"
  } else if (conteo2 == 7) {
    mostrarNumero(1, 1, 1, 0, 0, 0, 0, 2); // Mostrar "7"
  } else if (conteo2 == 8) {
    mostrarNumero(1, 1, 1, 1, 1, 1, 1, 2); // Mostrar "8"
  }

}
//*********************************
// Handler de Inicio página
//*********************************
void handle_OnConnect() {
  server.send(200, "text/html", SendHTML(Pstate));
}
//*********************************
// Procesador de HTML
//*********************************
String SendHTML(uint8_t state) {
String ptr = "<!DOCTYPE html>\n";
ptr += "<html lang=\"en\">\n";
ptr += "<head>\n";
ptr += "<meta charset=\"UTF-8\">\n";
ptr += "<meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0\">\n";
ptr += "<title>Parqueomatic</title>\n";
ptr += "<style>\n";
ptr += "body{\n";
ptr += "font-family:Arial,sans-serif;\n";
ptr += "margin:70px;\n";
ptr += "background-color:#f4f4f4;\n";
ptr += "}\n";
ptr += "h1{\n";
ptr += "color:#2c3e50;\n";
ptr += "text-align:center;\n";
ptr += "}\n";
ptr += "p{\n";
ptr += "text-align:center;\n";
```

```
ptr += "color:#2c3e50;\n";
ptr += "}\n";
ptr += "table{\n";
ptr += "width:80%;\n";
ptr += "border-collapse:collapse;\n";
ptr += "margin:70px auto;\n";
ptr += "background-color:#3498db;\n";
ptr += "}\n";
ptr += "th, td{\n";
ptr += "border:1px solid #ddd;\n";
ptr += "padding:70px;\n";
ptr += "text-align:center;\n";
ptr += "width:150px;\n";
ptr += "}\n";
ptr += "th{\n";
ptr += "background-color:#2c3e50;\n";
ptr += "color:white;\n";
ptr += "}\n";
ptr += ".nivel2-parqueo5{\n";
if (parqueo5 == 1){
  p5 = 1;
  ptr += "background-color:#25d366;\n";
}
if (parqueo5 == 0){
  p5 = 0;
ptr += "background-color:#e74c3c;\n";
}
ptr += "}\n";
ptr += ".nivel2-parqueo6{\n";
if (parqueo6 == 1){
  p6 = 1;
  ptr += "background-color:#25d366;\n";
}
if (parqueo6 == 0){
  p6 = 0;
ptr += "background-color:#e74c3c;\n";
}
ptr += "}\n";
ptr += ".nivel2-parqueo7{\n";
if (parqueo7 == 1){
  p7 = 1;
  ptr += "background-color:#25d366;\n";
}
if (parqueo7 == 0){
  p7 = 0;
ptr += "background-color:#e74c3c;\n";
```

```
}
ptr += "}\n";
ptr += ".nivel2-parqueo8{\n";
if (parqueo8 == 1){
  p8 = 1;
  ptr += "background-color:#25d366;\n";
}
if (parqueo8 == 0){
  p8 = 0;
ptr += "background-color:#e74c3c;\n";
}
ptr += "}\n";
ptr += ".nivel1-parqueo1{\n";
if (parqueo1 == 1){
  p1 = 1;
  ptr += "background-color:#25d366;\n";
}
if (parqueo1 == 0){
  p1 = 0;
ptr += "background-color:#e74c3c;\n";
}
ptr += "}\n";
ptr += ".nivel1-parqueo2{\n";
if (parqueo2 == 1){
  p2 = 1;
  ptr += "background-color:#25d366;\n";
}
if (parqueo2 == 0){
  p2 = 0;
ptr += "background-color:#e74c3c;\n";
}
ptr += "}\n";
ptr += ".nivel1-parqueo3{\n";
if (parqueo3 == 1){
  p3 = 1;
  ptr += "background-color:#25d366;\n";
}
if (parqueo3 == 0){
  p3 = 0;
ptr += "background-color:#e74c3c;\n";
}
ptr += "}\n";
ptr += ".nivel1-parqueo4{\n";
if (parqueo4 == 1){
  p4 = 1;
  ptr += "background-color:#25d366;\n";
```

```
}
if (parqueo4 == 0){
  p4 = 0;
ptr += "background-color:#e74c3c;\n";
}
conteo = p1+p2+p3+p4+p5+p6+p7+p8;
conteo1 = p1+p2+p3+p4;
conteo2 = p5+p6+p7+p8;
ptr += "}\n";
ptr += ".button{\n";
ptr += "display:block;\n";
ptr += "width:150px;\n";
ptr += "margin:20px auto;\n";
ptr += "padding:10px;\n";
ptr += "text-align:center;\n";
ptr += "text-decoration:none;\n";
ptr += "font-size:16px;\n";
ptr += "border:2px solid #2c3e50;\n";
ptr += "border-radius:5px;\n";
ptr += "color:#2c3e50;\n";
ptr += "background-color:white;\n";
ptr += "cursor:pointer;\n";
ptr += "}\n";
ptr += ".button:hover{\n";
ptr += "background-color:#2c3e50;\n";
ptr += "color:white;\n";
ptr += "}\n";
ptr += "</style>\n";
ptr += "</head>\n";
ptr += "<body>\n";
ptr += "<h1>Parqueomatic</h1>\n";
if (conteo == 0){
ptr += "<p>Parqueos disponibles: 0</p>\n";
}
if (conteo == 1){
ptr += "<p>Parqueos disponibles: 1</p>\n";
}
if (conteo == 2){
ptr += "<p>Parqueos disponibles: 2</p>\n";
}
if (conteo == 3){
ptr += "<p>Parqueos disponibles: 3</p>\n";
}
if (conteo == 4){
ptr += "<p>Parqueos disponibles: 4</p>\n";
}
```

```
if (conteo == 5){
ptr += "<p>Parqueos disponibles: 5</p>\n";
}
if (conteo == 6){
ptr += "<p>Parqueos disponibles: 6</p>\n";
}
if (conteo == 7){
ptr += "<p>Parqueos disponibles: 7</p>\n";
}
if (conteo == 8){
ptr += "<p>Parqueos disponibles: 8</p>\n";
}
ptr += "<table>\n";
ptr += "<tbody>\n";
ptr += "<tr>\n";
if (conteo2 == 0){
ptr += "<th>Nivel 2 Parqueos Disponibles: 0</th>\n";
}
if (conteo2 == 1){
ptr += "<th>Nivel 2 Parqueos Disponibles: 1</th>\n";
}
if (conteo2 == 2){
ptr += "<th>Nivel 2 Parqueos Disponibles: 2</th>\n";
}
if (conteo2 == 3){
ptr += "<th>Nivel 2 Parqueos Disponibles: 3</th>\n";
}
if (conteo2 == 4){
ptr += "<th>Nivel 2 Parqueos Disponibles: 4</th>\n";
}
if (parqueo5 == 1){
ptr += "<td class=\"nivel2-parqueo5\">Parqueo 5 - Disponible</td>\n";
}
if (parqueo5 == 0){
ptr += "<td class=\"nivel2-parqueo5\">Parqueo 5 - Ocupado</td>\n";
}
if (parqueo6 == 1){
ptr += "<td class=\"nivel2-parqueo6\">Parqueo 6 - Disponible</td>\n";
}
if (parqueo6 == 0){
ptr += "<td class=\"nivel2-parqueo6\">Parqueo 6 - Ocupado</td>\n";
}
if (parqueo7 == 1){
ptr += "<td class=\"nivel2-parqueo7\">Parqueo 7 - Disponible</td>\n";
}
if (parqueo7 == 0){
```

```
ptr += "<td class=\"nivel2-parqueo7\">Parqueo 7 - Ocupado</td>\n";
}
if (parqueo8 == 1){
ptr += "<td class=\"nivel2-parqueo8\">Parqueo 8 - Disponible</td>\n";
}
if (parqueo8 == 0){
ptr += "<td class=\"nivel2-parqueo8\">Parqueo 8 - Ocupado</td>\n";
}
ptr += "</tr>\n";
ptr += "<tr>\n";
if (conteo1 == 0){
ptr += "<th>Nivel 1 Parqueos Disponibles: 0</th>\n";
}
if (conteo1 == 1){
ptr += "<th>Nivel 1 Parqueos Disponibles: 1</th>\n";
}
if (conteo1 == 2){
ptr += "<th>Nivel 1 Parqueos Disponibles: 2</th>\n";
}
if (conteo1 == 3){
ptr += "<th>Nivel 1 Parqueos Disponibles: 3</th>\n";
}
if (conteo1 == 4){
ptr += "<th>Nivel 1 Parqueos Disponibles: 4</th>\n";
}
if (parqueo1 == 1){
ptr += "<td class=\"nivel1-parqueo1\">Parqueo 1 - Disponible</td>\n";
}
if (parqueo1 == 0){
ptr += "<td class=\"nivel1-parqueo1\">Parqueo 1 - Ocupado</td>\n";
}
if (parqueo2 == 1){
ptr += "<td class=\"nivel1-parqueo2\">Parqueo 2 - Disponible</td>\n";
}
if (parqueo2 == 0){
ptr += "<td class=\"nivel1-parqueo2\">Parqueo 2 - Ocupado</td>\n";
}
if (parqueo3 == 1){
ptr += "<td class=\"nivel1-parqueo3\">Parqueo 3 - Disponible</td>\n";
}
if (parqueo3 == 0){
ptr += "<td class=\"nivel1-parqueo3\">Parqueo 3 - Ocupado</td>\n";
}
if (parqueo4 == 1){
ptr += "<td class=\"nivel1-parqueo4\">Parqueo 4 - Disponible</td>\n";
}
```

```
if (parqueo4 == 0){
ptr += "<td class=\"nivel1-parqueo4\">Parqueo 4 - Ocupado</td>\n";
}
ptr += "</tr>\n";
ptr += "</tbody>\n";
ptr += "</table>\n";
ptr += "<a class=\"button button-off\" href=\"actualizar\">Actualizar</a>\n";
ptr += "</body>\n";
ptr += "</html>\n";

  return ptr;
}
//*********************************
// Handler de not found
//*********************************
void handle_NotFound() {
  server.send(404, "text/plain", "Not found");
}
```

Explicación del Código del ESP32:

En este código, se configura el ESP32 para funcionar como un servidor web en una red local específica. El propósito principal es generar una página web accesible desde cualquier dispositivo conectado a esa red. La página web en cuestión proporciona información en tiempo real sobre el estado de un estacionamiento de dos niveles.

La comunicación es posible gracias a la conexión UART entre las plataformas Tiva C y el ESP32, establecida en los pines RX0 y RX2. Cada Tiva envía un valor correspondiente al estado actual de un espacio de estacionamiento, permitiendo así la actualización continua de la información.

Adicionalmente, se incorporan dos displays. El primero, conectado a los pines D13, D12, D14, D27, D26, D25 y D23, indica la disponibilidad de espacios en el nivel 2 del estacionamiento. El segundo display, conectado a los pines D15, D22, D4, D5, D18, D19 y D21, refleja la disponibilidad en el nivel 1.

La parte final del código presenta una página web en HTML, con condiciones dinámicas dependientes de la información recibida a través de la UART. Estas condiciones determinan la apariencia de la página, mostrando el estado de cada espacio de estacionamiento y contadores totales para cada nivel.