Universidad del Valle de Guatemala
Electrónica Digital II
Sección 10
17 de noviembre del 2023

Proyecto #3 - Parqueo Inteligente

Judah Sebastian Pérez Zeiset 21536, Carlos Daniel Valdez Coreas 21976.

Enlaces

- Video demostrativo: https://youtu.be/QCs0UzHwoEA
- GitHub: https://github.com/Ritmop/Proyecto2_Digital2_Parqueo

Figura 1. Ensamble del proyecto.

Código TivaC

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

Librerías incluidas.

```
//*****************************************************************************
```

```
*
// Variables Globales
//*****************************************************************
*
#define SENSORES_PORTA_PINS  GPIO_PIN_5|GPIO_PIN_6|GPIO_PIN_7
#define SENSORES_PORTB_PINS  GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_4
#define SENSORES_PORTE_PINS  GPIO_PIN_4|GPIO_PIN_5

#define RED_PORTD_MASK  GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3
#define RED_PORTE_MASK  GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3
#define RED_PORTF_MASK  GPIO_PIN_1

#define GREEN_PORTA_MASK  GPIO_PIN_2
#define GREEN_PORTB_MASK  GPIO_PIN_3
#define GREEN_PORTC_MASK  GPIO_PIN_4|GPIO_PIN_5|GPIO_PIN_6|GPIO_PIN_7
#define GREEN_PORTF_MASK  GPIO_PIN_2|GPIO_PIN_3

#define UART2_PINS      GPIO_PIN_7|GPIO_PIN_6

uint8_t sensorValues;
uint8_t parqueosDisponibles;
uint8_t parqueosOcupados;
```

Definición de máscaras y variables globales.

```
//*****************************************************************
*
// Prototipos de Función
//*****************************************************************
*
uint8_t readSensors(void);
void writeRed(uint8_t redVal);
void writeGreen(uint8_t greenVal);
void sendUART0(void);
void sendUART2(void);
uint8_t countSetBits(uint8_t bitwise);
void binaryToASCII(uint8_t bitwise);
```

Prototipos de funciones.

```
//*****************************************************************************
*
// Código Principal
//*****************************************************************************
*
int main(void)
{
      // configuración del reloj a 40MHz con el oscilador externo de
16MHz y PLL

SysCtlClockSet(SYSCTL_OSC_MAIN|SYSCTL_USE_PLL|SYSCTL_SYSDIV_5|SYSCTL_XTA
L_16MHZ);

      // Habilitar Puertos
      SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);
      SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);
      SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOC);
      SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOE);
      SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);
      SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
```

Configuración del reloj principal y habilitación de periféricos.

```
// Configurar Pines de sensores
      GPIOPinTypeGPIOInput(GPIO_PORTA_BASE, SENSORES_PORTA_PINS);
      GPIOPinTypeGPIOInput(GPIO_PORTB_BASE, SENSORES_PORTB_PINS);
      GPIOPinTypeGPIOInput(GPIO_PORTE_BASE, SENSORES_PORTE_PINS);

      // Configurar Pines LEDs
      GPIOPinTypeGPIOOutput(GPIO_PORTA_BASE, GREEN_PORTA_MASK);
      GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GREEN_PORTB_MASK);
      GPIOPinTypeGPIOOutput(GPIO_PORTC_BASE, GREEN_PORTC_MASK);
      GPIOPinTypeGPIOOutput(GPIO_PORTD_BASE, RED_PORTD_MASK);
      GPIOPinTypeGPIOOutput(GPIO_PORTE_BASE, RED_PORTE_MASK);
      GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE,
RED_PORTF_MASK|GREEN_PORTF_MASK);
```

Configuración de entradas y salidas.

```
//Habilitar módulo UART2 para la comunicación con ESP32
```

```
      SysCtlPeripheralEnable(SYSCTL_PERIPH_UART2);
      GPIOPinTypeUART(GPIO_PORTD_BASE, UART2_PINS);
      UARTConfigSetExpClk(UART2_BASE, SysCtlClockGet(), 115200,
                          (UART_CONFIG_WLEN_8 | UART_CONFIG_STOP_ONE |
UART_CONFIG_PAR_NONE));

      //Habilitar módulo UART0 para la comunicación con computadora
      SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);
      GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0 | GPIO_PIN_1);
      UARTConfigSetExpClk(UART0_BASE, SysCtlClockGet(), 115200,
                          (UART_CONFIG_WLEN_8 | UART_CONFIG_STOP_ONE |
UART_CONFIG_PAR_NONE));
```

Configuración de módulos UART.

```
//*****************************************************************************
*
//Loop Principal
//*****************************************************************************
*
      while (1)
      {
      sensorValues = readSensors();   //Lectura de sensores
      parqueosOcupados = sensorValues;          //Sensor tapado indica
ocupado
      parqueosDisponibles = (~sensorValues);
      writeRed(parqueosOcupados);
      writeGreen(parqueosDisponibles);


      sendUART0();
      sendUART2();
      SysCtlDelay(300000);
```

```
//*****************************************************************************
*
// Lectura de sensores LDR
//*****************************************************************************
*
uint8_t readSensors(void){
      return GPIOPinRead(GPIO_PORTA_BASE, SENSORES_PORTA_PINS)|
            GPIOPinRead(GPIO_PORTB_BASE, SENSORES_PORTB_PINS)|
```

```
            GPIOPinRead(GPIO_PORTE_BASE, SENSORES_PORTE_PINS) >> 2;
}

//*****************************************************************************
*
// Escrituda de LEDS rojos
//*****************************************************************************
*
void writeRed(uint8_t redVal){
      GPIOPinWrite(GPIO_PORTD_BASE, RED_PORTD_MASK, redVal);
      GPIOPinWrite(GPIO_PORTE_BASE, RED_PORTE_MASK, redVal >> 3);
      GPIOPinWrite(GPIO_PORTF_BASE, RED_PORTF_MASK, redVal >> 6);
}

//*****************************************************************************
*
// Escrituda de LEDS verdes
//*****************************************************************************
*
void writeGreen(uint8_t greenVal){
      GPIOPinWrite(GPIO_PORTA_BASE, GREEN_PORTA_MASK, greenVal >> 5);
      GPIOPinWrite(GPIO_PORTB_BASE, GREEN_PORTB_MASK, greenVal << 1);
      GPIOPinWrite(GPIO_PORTC_BASE, GREEN_PORTC_MASK, greenVal << 1);
      GPIOPinWrite(GPIO_PORTF_BASE, GREEN_PORTF_MASK, greenVal << 2);
}
```

Lectura y escritura de Puertos.

```
//*****************************************************************************
*
// Envio de datos a la computadora
//*****************************************************************************
*
void sendUART0(void){
```

```
      uint8_t disponibles = countSetBits(parqueosDisponibles) + '0';
      uint8_t ocupados = countSetBits(parqueosOcupados) + '0';
      UARTCharPut(UART0_BASE,'D');
      UARTCharPut(UART0_BASE,':');
      UARTCharPut(UART0_BASE,disponibles);
      UARTCharPut(UART0_BASE,' ');
      UARTCharPut(UART0_BASE,'-');
      UARTCharPut(UART0_BASE,' ');
      UARTCharPut(UART0_BASE,'O');
      UARTCharPut(UART0_BASE,':');
      UARTCharPut(UART0_BASE,ocupados);
      UARTCharPut(UART0_BASE,' ');
      binaryToASCII(parqueosOcupados);
      UARTCharPut(UART0_BASE,' ');
      UARTCharPut(UART0_BASE,parqueosOcupados);
      UARTCharPut(UART0_BASE,'\r');
}

//*********************************************************************
*
// Envio de datos a ESP32
//*********************************************************************
*
void sendUART2(void){
      //uint8_t disponibles = countSetBits(parqueosDisponibles);
      UARTCharPut(UART2_BASE, parqueosOcupados);
}
```

Funciones para enviar datos por UART.

```
//*********************************************************************
*
// Obtener cantidad de bits seteados de una variable
//*********************************************************************
*
uint8_t countSetBits(uint8_t bitwise){
      uint8_t count = 0;
```

```
	uint8_t i;
	for (i = 0; i < 8; i++){
	if ((bitwise % 2) == 1){
		count++;
	}
	bitwise = bitwise >> 1;
	bitwise &= 0x7F;
	}
	return count;
}

//**********************************************************************
**********************************
// Convertir en una cadena binaria los valores de una variable
//**********************************************************************
**********************************
void binaryToASCII(uint8_t bitwise){
	uint8_t i;
	for (i = 0; i < 8; i++){
	if (((bitwise & 0x01) % 2) == 0){ //último bit es 0
		UARTCharPut(UART0_BASE,'0');
	}
	else {
		UARTCharPut(UART0_BASE,'1');
	}
	bitwise = bitwise >> 1;
	}
}
```

Otras funciones.

Código ESP

```
// Librerías
//*************************************************************************************************
#include <Arduino.h>
#include <WiFi.h>
#include <WebServer.h>
#include <HardwareSerial.h>

// put function declarations here:
void handle_actualizar();
void handle_OnConnect();
void handle_NotFound();
void procesarDatos(char data);
void display(int num);
```

Librerías y funciones utilizadas

```
// Variables globales

//Variables booleanas para cada parqueo
bool bit0, bit1, bit2, bit3, bit4, bit5, bit6, bit7;
// variable que recibe los datos de la Tiva 1
char data;
// variable que recibe los datos de la Tiva 2
//char data1;
// SSID & Password
const char* ssid = "danielvaldez";  // Enter your SSID here
const char* password = "daniel21976";  //Enter your Password here
const int pin7s[] = {15,2,4,5,18,19,21};  //pines para display de 7 segmentos (a,b,c,d,e,f,g)
const char prueba[] = {0,1,3,7,15,31,63,127,255};

int a = 15;
int b = 2;
int c = 4;
int d = 5;
int e = 18;
int f = 19;
int g = 21;

WebServer server(80);  // Object of WebServer(HTTP port, 80 is defult)
HardwareSerial MySerial1(1);  //Se define un serial para UART1
const int Serial1RX = 16;
const int Serial1TX = 17;
const int totalParqueos = 8;
int cantidadBitsTrue;
int resultado;
```

Variables globales

```
String SendHTML(){
String ptr = "<!DOCTYPE html>\n";
ptr += "<html lang=\"es\">\n";
ptr += "<head>\n";
ptr += "  <meta charset=\"UTF-8\">\n";
ptr += "  <meta name=\"viewport\" content=\"width=device-width,
initial-scale=1, shrink-to-fit=no\">\n";
ptr +="  <title>Parqueo JD</title>\n";
ptr +="  <!-- Vincula las hojas de estilo de Bootstrap -->\n";
ptr +="  <link rel=\"stylesheet\"
href=\"https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.
css\">\n";
ptr +="  <style>\n";
ptr +="    body {\n";
ptr +="      background: linear-gradient(to left, #333, #000); /* Fondo
oscuro en gradiente */\n";
ptr +="      color: ;#fff; /* Color de texto blanco */\n";
ptr +="      text-align: center;\n";
ptr +="      font-family: 'Arial', sans-serif; /* Tipo de letra */\n";
ptr +="    }\n";
ptr +="\n";
ptr +="    h1 {\n";
ptr +="      margin-top: 50px; /* Espaciado superior del título */\n";
ptr +="    }\n";
ptr +="\n";
ptr +="    .progress {\n";
ptr +="      width: 70%; /* Ancho de la barra de progreso */\n";
ptr +="      margin: 30px auto; /* Centrar la barra de progreso */\n";
ptr +="    }\n";
ptr +="\n";
ptr +="    button {\n";
ptr +="      margin-top: 20px; /* Espaciado superior del botón */\n";
ptr +="      box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1); /* Sombra del botón
*/\n";
ptr +="    }\n";
ptr +="\n";
ptr +="    table {\n";
ptr +="      width: 80%; /* Ancho de la tabla */\n";
ptr +="      margin: 30px auto; /* Centrar la tabla */\n";
ptr +="      box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1); /* Sombra de la tabla
*/\n";
ptr +="    }\n";
ptr +="\ttable th, table td {\n";
```

```
ptr +="    color: #fff; /* Color de texto blanco para las celdas de la tabla
*/}\n";
ptr +="\n";
ptr +="\tth.numero-parqueo, td.numero-parqueo {\n";
ptr +="\twidth: 50%; /* Ancho igual para ambas columnas */\n";
ptr +="    text-align: center; /* Alinea el texto a la izquierda para la
columna \"Número de Parqueo\" */\n";
ptr +="    color: #fff; /* Color de texto blanco para las celdas de esta
columna */\n";
ptr +="  }\n";
ptr +="\n";
ptr +="  \tth.estado, td.estado {\n";
ptr +="\twidth: 50%; /* Ancho igual para ambas columnas */\n";
ptr +="    text-align: center; /* Alinea el texto a la derecha para la
columna \"Estado\" */\n";
ptr +="    color: #fff; /* Color de texto blanco para las celdas de esta
columna */\n";
ptr +="  }\n";
ptr +="\n";
ptr +="  .contadorD {\n";
ptr +="    position: absolute;\n";
ptr +="    top: 50px;\n";
ptr +="    left: 20px;\n";
ptr +="    color: #fff;\n";
ptr +="  }\n";
ptr +="  .contadorO {\n";
ptr +="    position: absolute;\n";
ptr +="    top: 80px;\n";
ptr +="    left: 20px;\n";
ptr +="    color: #fff;\n";
ptr +="  }\n";
ptr +="\n";
ptr +="  </style>\n";
ptr +="</head>\n";
ptr +="<body>\n";
ptr +="\n";
ptr +="  <div class=\"container\">\n";
ptr +="    <h1><font color=white>Parqueo JD &#x1F17F🚗 </h1>\n";
ptr +="\n";
ptr +="    <!-- Barra de progreso -->\n";
ptr +="    <div id=\"barraProgreso\" class=\"progress\">\n";
ptr +="    <div class=\"progress-bar progress-bar-striped bg-danger\"
role=\"progressbar\" aria-valuenow=\"0\" aria-valuemin=\"0\"
aria-valuemax=\"100\">  \n";
ptr +="    </div>\n";
ptr +="      </div>\n";
```

```
ptr +="\n";
ptr +="    <!-- Botón de actualizar -->\n";
ptr +="    <button type=\"button\" class=\"btn btn-primary btn-lg\"
href=\"/actualizar\">Actualizar</button>\n";
ptr +="\n";
ptr +="    <div class=\"contadorO\" id=\"contadorOcupados\">Parqueos
Ocupados: </div>\n";
ptr +="    <div class=\"contadorD\" id=\"contadorLibres\">Parqueos Libres:
</div>\n";
ptr +="\n";
ptr +="    <!-- Tabla de parqueos -->\n";
ptr +="    <table class=\"table table-striped\">\n";
ptr +="      <thead>\n";
ptr +="        <tr>\n";
ptr +="          <th class=\"numero-parqueo\" scope=\"col\">Número de
Parqueo</th>\n";
ptr +="          <th class=\"estado\" scope=\"col\">Estado</th>\n";
ptr +="        </tr>\n";
ptr +="      </thead>\n";
ptr +="      <tbody>\n";
ptr +="        <tr>\n";
ptr +="          <td class=\"numero-parqueo\">1</td>\n";
ptr +="          <td class=\"estado\" id=\"par1\"></td>\n";
ptr +="        </tr>\n";
ptr +="        <tr>\n";
ptr +="          <td class=\"numero-parqueo\">2</td>\n";
ptr +="          <td class=\"estado\" id=\"par2\"></td>\n";
ptr +="        </tr>\n";
ptr +="        <tr>\n";
ptr +="          <td class=\"numero-parqueo\">3</td>\n";
ptr +="          <td class=\"estado\" id=\"par3\"></td>\n";
ptr +="        </tr>\n";
ptr +="        <tr>\n";
ptr +="          <td class=\"numero-parqueo\">4</td>\n";
ptr +="          <td class=\"estado\" id=\"par4\"></td>\n";
ptr +="        </tr>\n";
ptr +="        <tr>\n";
ptr +="          <td class=\"numero-parqueo\">5</td>\n";
ptr +="          <td class=\"estado\" id=\"par5\"></td>\n";
ptr +="        </tr>\n";
ptr +="        <tr>\n";
ptr +="          <td class=\"numero-parqueo\">6</td>\n";
ptr +="          <td class=\"estado\" id=\"par6\"></td>\n";
ptr +="        </tr>\n";
ptr +="        <tr>\n";
ptr +="          <td class=\"numero-parqueo\">7</td>\n";
```

```
ptr +="          <td class=\"estado\" id=\"par7\"></td>\n";
ptr +="        </tr>\n";
ptr +="        <tr>\n";
ptr +="          <td class=\"numero-parqueo\">8</td>\n";
ptr +="          <td class=\"estado\" id=\"par8 \"></td>\n";
ptr +="        </tr>\n";
ptr +="      </tbody>\n";
ptr +="    </table>\n";
ptr +="  </div>\n";
ptr +="\n";
ptr +="  <!-- Vincula los scripts de Bootstrap y jQuery (necesario para
algunas funciones de Bootstrap) -->\n";
ptr +="  <script
src=\"https://code.jquery.com/jquery-3.3.1.slim.min.js\"></script>\n";
ptr +="  <script
src=\"https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min
.js\"></script>\n";
ptr +="  <script
src=\"https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js
\"></script>\n";
ptr +="\n";
ptr +="</body>\n";
ptr +="</html>\n";
ptr +="\n";
ptr +="<script>\n";

if (bit0)
  {
    ptr +="  var P1 = 1;\n";
  }
  else
  {
    ptr +="  var P1 = 0;\n";
  }
if (bit1)
  {
    ptr +="  var P2 = 1;\n";
  }
  else
  {
    ptr +="  var P2 = 0;\n";
  }

if (bit2)
  {
    ptr +="  var P3 = 1;\n";
```

```
  }
  else
  {
    ptr +="  var P3 = 0;\n";
  }

if (bit3)
  {
    ptr +="  var P4 = 1;\n";
  }
  else
  {
    ptr +="  var P4 = 0;\n";
  }

if (bit4)
  {
    ptr +="  var P5 = 1;\n";
  }
  else
  {
    ptr +="  var P5 = 0;\n";
  }

if (bit5)
  {
    ptr +="  var P6 = 1;\n";
  }
  else
  {
    ptr +="  var P6 = 0;\n";
  }

if (bit6)
  {
    ptr +="  var P7 = 1;\n";
  }
  else
  {
    ptr +="  var P7 = 0;\n";
  }

if (bit7)
  {
    ptr +="  var P8 = 1;\n";
  }
```

```
  else
  {
    ptr +="  var P8 = 0;\n";
  }

ptr +="  // Ejemplo de valores iniciales\n";
ptr +="  var parqueosOcupados = P1 + P2+ P3 + P4 + P5 + P6 + P7 + P8 ;\n";
ptr +="  var totalParqueos = 8;\n";
ptr +="  var parqueosLibres = (totalParqueos - parqueosOcupados);\n";
ptr +="  // Calcula el porcentaje de parqueos ocupados\n";
ptr +="  var porcentajeOcupados = (parqueosOcupados / totalParqueos) *
100;\n";
ptr +="  // Actualiza los contadores\n";
ptr +="  var estadoCelda1 = P1 === 1 ? \"Ocupado\" : \"Libre\" \n";
ptr +="  var estadoCelda2 = P2 === 1 ? \"Ocupado\" : \"Libre\" \n";
ptr +="  var estadoCelda3 = P3 === 1 ? \"Ocupado\" : \"Libre\" \n";
ptr +="  var estadoCelda4 = P4 === 1 ? \"Ocupado\" : \"Libre\" \n";
ptr +="  var estadoCelda5 = P5 === 1 ? \"Ocupado\" : \"Libre\" \n";
ptr +="  var estadoCelda6 = P6 === 1 ? \"Ocupado\" : \"Libre\" \n";
ptr +="  var estadoCelda7 = P7 === 1 ? \"Ocupado\" : \"Libre\" \n";
ptr +="  var estadoCelda8 = P8 === 1 ? \"Ocupado\" : \"Libre\" \n";
ptr +="  document.getElementById(\"contadorOcupados\").innerText =
\"Parqueos Ocupados: \" + parqueosOcupados;\n";
ptr +="  document.getElementById(\"contadorLibres\").innerText = \"Parqueos
Libres: \" + parqueosLibres;\n";
ptr +="\n";
ptr +="  // Actualiza la barra de progreso\n";
ptr +="  var barraProgreso =
document.getElementById(\"barraProgreso\").firstElementChild;\n";
ptr +="  barraProgreso.style.width = porcentajeOcupados + \"%\";\n";
ptr +="  barraProgreso.innerText = porcentajeOcupados + \"% Ocupados\";\n";
ptr +="  document.getElementById(\"par1\").innerText = estadoCelda1;\n";
ptr +="  document.getElementById(\"par2\").innerText = estadoCelda2;\n";
ptr +="  document.getElementById(\"par3\").innerText = estadoCelda3;\n";
ptr +="  document.getElementById(\"par4\").innerText = estadoCelda4;\n";
ptr +="  document.getElementById(\"par5\").innerText = estadoCelda5;\n";
ptr +="  document.getElementById(\"par6\").innerText = estadoCelda6;\n";
ptr +="  document.getElementById(\"par7\").innerText = estadoCelda7;\n";
ptr +="  document.getElementById(\"par8\").innerText = estadoCelda8;\n";
ptr +="</script>\n";
ptr +="\n";
return ptr;
}
```

Código de Servidor Web. Se utilizó Bootstrap para el apartado visual y JavaScript para utilizar elementos dinámicos, como tablas, botones y barras de progreso. Este código se guarda como una variable tipo string en el ESP.

```
void setup() {
  Serial.begin(115200);
```

```
 //Serial1.begin(115200, SERIAL_8N1, 16, 17, false, 20000UL, 112);
//Habilitar el UART1 y cambiar los pins
  Serial.println("Try Connecting to ");
  Serial.println(ssid);
  MySerial1.begin(115200, SERIAL_8N1, Serial1RX, Serial1TX);


    pinMode(15, OUTPUT);
    pinMode(2, OUTPUT);
    pinMode(4, OUTPUT);
    pinMode(5, OUTPUT);
    pinMode(18, OUTPUT);
    pinMode(19, OUTPUT);
    pinMode(21, OUTPUT);

    digitalWrite(15, LOW);
    digitalWrite(2, LOW);
    digitalWrite(4, LOW);
    digitalWrite(5, LOW);
    digitalWrite(18, LOW);
    digitalWrite(19, LOW);
    digitalWrite(21, LOW);

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

  // 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_actualizar);

  server.onNotFound(handle_NotFound);

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

Configuración del ESP32

```
void loop() {
  server.handleClient();

  for (int i=0; i < 8; i++){
    data = prueba[i];
    server.handleClient();
    Serial.println("data = ");
    Serial.println(data);
    procesarDatos(data);
    display(resultado);
    handle_actualizar();
    delay(5000);
  }

}
```

Bucle Principal

```
// Handler de Inicio página
//***********************************************
void handle_OnConnect() {
  Serial.println("server up");
  server.send(200, "text/html", SendHTML());
}
//***********************************************
// Handler de not found
//***********************************************
void handle_NotFound() {
  server.send(404, "text/plain", "Not found");
}
```

Handle_OnConnect inicia la conexión al servidor web. Handle_NotFound envía un mensaje de error en caso que no se logre la conexión.

```
void procesarDatos(char bitsp) {

    // Se aplica una máscara para extraer cada bit recibido de la tiva
    bit0 = (bitsp & 0b00000001) != 0;
    bit1 = (bitsp & 0b00000010) != 0;
    bit2 = (bitsp & 0b00000100) != 0;
    bit3 = (bitsp & 0b00001000) != 0;
    bit4 = (bitsp & 0b00010000) != 0;
    bit5 = (bitsp & 0b00100000) != 0;
    bit6 = (bitsp & 0b01000000) != 0;
    bit7 = (bitsp & 0b10000000) != 0;

    // obtener el número de parqueos libres
    cantidadBitsTrue = bit0 + bit1 + bit2 + bit3 + bit4 + bit5 + bit6 +
bit7;
    resultado = totalParqueos - cantidadBitsTrue;

    Serial.println("parqueos disponibles");
    Serial.println("p1");
    Serial.println(bit0);
    Serial.println("p2");
    Serial.println(bit1);
    Serial.println("p3");
    Serial.println(bit2);
    Serial.println("p4");
    Serial.println(bit3);
    Serial.println("p5");
    Serial.println(bit4);
    Serial.println("p6");
    Serial.println(bit5);
    Serial.println("p7");
    Serial.println(bit6);
    Serial.println("p8");
    Serial.println(bit7);
    Serial.println("Cantidad de Parqueos disponibles");
    Serial.println(resultado);
    delay(500);
}
```

Toma la variable char y la separa en bits individuales para saber el estado de cada parqueo. Se suman los parqueos libres para encontrar el total de parqueos disponibles.

```
void display(int num) {
```

```
  // Tabla de mapeo para mostrar dígitos en el display de 7 segmentos
  const byte segmentMapping[] = {
    B11111100,  // 0
    B01100000,  // 1
    B11011010,  // 2
    B11110010,  // 3
    B01100110,  // 4
    B10110110,  // 5
    B10111110,  // 6
    B11100000,  // 7
    B11111110,  // 8
    B11110110   // 9
  };

  // Validar el número para asegurarse de que esté en el rango adecuado
  if (num >= 0 && num <= 9) {
    // Apagar todos los segmentos
    for (int i = 0; i < 7; i++) {
      digitalWrite(pin7s[i], LOW);
    }

switch (num){
    case 0:
  digitalWrite(a, HIGH);
  digitalWrite(b, HIGH);
  digitalWrite(c, HIGH);
  digitalWrite(d, HIGH);
  digitalWrite(e, HIGH);
  digitalWrite(f, HIGH);
  digitalWrite(g, LOW);
break;
```

Función para mapear el display de 7 segmentos. Se utiliza una estructura switch-case para encender cada LED dependiendo del número obtenido.