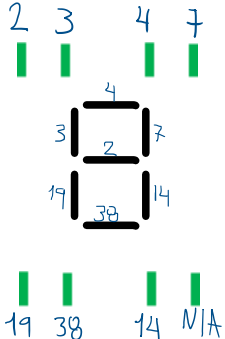


## Proyecto #4: Control de Registro para Parqueos

### Circuitos Utilizados

Para simular los indicadores y sensores de el parqueo para verificar si esta disponible o no, su utilizaron 8 Leds, 4 rojas y 4 verdes. Cada una conectada mediante una resistencia a distintos pines de la tiva. Y para sensar si el parqueo está ocupado o no, se utilizaron 4 push buttons correspondientes a los 4 parqueos. Además, se utilizó un display de 7 segmentos para indicar la cantidad de parqueos actualmente disponibles. Para la comunicación con el ESP32 se utilizó el UART1 de la tiva en los pines PC4 y PC5. A continuación, una tabla con la información de los pines a los cuales se conectaron cada uno de los componentes de cada parqueo.

Parqueo	Componente	Numero de pin	Pin según puerto
PARQUEO 1	Led roja	5	PE_4
	Led Verde	6	PE_5
	Push button como sensor	28	PE_2
PARQUEO 2	Led roja	11	PA_2
	Led Verde	12	PA_3
	Push button como sensor	29	PE_3
PARQUEO 3	Led roja	13	PA_4
	Led Verde	8	PA_5
	Push button como sensor	36	PC_6
PARQUEO 4	Led roja	9	PA_6
	Led Verde	10	PA_7
	Push button como sensor	37	PC_7
INDICADOR DE CANTIDAD DE PARQUEOS		2	PB_5
		3	PB_0
		4	PB_1
		7	PB_4
		19	PB_2
		38	PB_3
		14	PB_6

## Datos y Variables

### En Tiva C:

En la tiva C se utilizaron las siguientes variables, se indica para cada una su uso:

- **Parqueo1, Parqueo2, Parqueo3, Parqueo4:** toma el valor del sensor para indicar si dicho parqueo esta disponible o no.
- **Disp:** Indica con cada uno de sus bits la disponibilidad de cada parqueo. Se utiliza para poder enviar la información de la disponibilidad de cada parqueo de una manera mas rápida y sencilla mediante comunicación UART al ESP32. A continuación se detalla el significado de cada bit:
  - **Disp\_0:** disponibilidad de parqueo 1. Vale 1 si está disponible, 0 si no lo está.
  - **Disp\_1:** disponibilidad de parqueo 2. Vale 1 si está disponible, 0 si no lo está.
  - **Disp\_2:** disponibilidad de parqueo 3. Vale 1 si está disponible, 0 si no lo está.
  - **Disp\_3:** disponibilidad de parqueo 4. Vale 1 si está disponible, 0 si no lo está.
  - **Disp\_4-7:** Sin uso

Entonces, suponiendo que Disp tiene un valor de 11. Quiere decir que en binario seria b1011, por lo que estarían disponibles los parqueos 1, 2 y 4. Y no está disponible el parqueo 3.

- **suma:** recopila la cantidad de parqueos disponibles dependiendo del valor de Disp. Va desde 0 hasta 4.

### En ESP32 – Arduino:

En el ESP32 se utilizaron las siguientes variables, se indica para cada una su uso:

- **Disp:** tiene exactamente la misma funcionalidad explicada para la tiva c
- **suma:** tiene exactamente la misma funcionalidad explicada para la tiva c
- **disponible:** es una variable de tipo string que contiene el código HTML para mostrar en la tabla el texto de disponible con el formato siguiente:

Disponible 😊

- **ocupado:** es una variable de tipo string que contiene el código HTML para mostrar en la tabla el texto de ocupado con el formato siguiente:

Ocupado 😞

## Explicaciones

En la tiva C se realizó el código que controla la disponibilidad de parqueos mediante la lectura de 4 botones correspondientes a los 4 parqueos. Además, muestra mediante leds rojos y verdes si el parqueo está ocupado o disponibles, respectivamente. Y finalmente, despliega la cantidad de parqueos libres en un display de 7 segmentos. El dato de qué parqueos están disponibles, lo guarda en la variable Disp explicada anteriormente y la envía por comunicación serial al ESP32.

El ESP32 recibe el dato de los parqueos disponibles mediante comunicación serial por el módulo UART2. Luego toma ese dato y dependiendo de los valores de cada bit, envía el código HTML hacia el servidor web con una interfaz gráfica que indica mediante una tabla, si el parqueo está disponible o no. Además se suma la cantidad de parqueos disponibles y ese valor también se envía para desplegarlo en la interfaz gráfica.

## Funciones

### Tiva C:

En la tiva C se crearon las siguientes funciones para simplificar el código, se detallan y se explican a continuación:

**void UART1config(void);** esta función es simplemente la configuración completa del UART1 de la tiva mediante las librerías.

**void display(uint8\_t valor);** esta función realiza un switch case para desplegar en el display el valor que se coloque como parámetro de la función.

### ESP32

En el ESP32 se crearon las siguientes funciones necesarias para el funcionamiento, se detallan y se explican a continuación:

**void handle\_OnConnect();** esta función es el handler que se ejecuta cada vez que la página web del servidor se refresca. Y dado que según el código implementado en el HTML, esta se refresca cada 500ms, entonces esta función se ejecuta cada 500ms. Dentro de la función se envía el String que contiene todo el código del HTML que describe la interfaz gráfica de la página web

**String SendHTML2();** esta función es la que recopila en un string, todo el código del HTML para enviarlo mediante el handle\_OnConnect(), va concatenando todo el string en una variable y dependiendo de los valores de las variables Disp y suma, se concatenan ciertos códigos HTML en el string que luego retornará la función.

## Código debidamente comentado:

### Tivaware:

```
#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/uart.h"
#include "driverlib/pin_map.h"

//*****PROTOTIPOS DE FUNCIONES*****

void UART1config(void);
void display(uint8_t valor);

/**
 * main.c
 */

//*****DECLARACION DE VARIABLES*****
uint8_t Parqueo1;
uint8_t Parqueo2;
uint8_t Parqueo3;
uint8_t Parqueo4;
uint8_t Disp = 0;
uint8_t suma = 0;

//*****PROGRAMA*****
void main(void)
//*****SETUP*****
{
    //Se configura reloj a 40MHz
    SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAIN);
    //Se activan los puertos E, C y A correspondientes a los que se usan para los
    puertos
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOE);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOC);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);
    //Se configuran como salida los LEDs de los distintos parqueos
    GPIOPinTypeGPIOOutput(GPIO_PORTE_BASE, GPIO_PIN_4|GPIO_PIN_5);
    GPIOPinTypeGPIOOutput(GPIO_PORTA_BASE,
GPIO_PIN_2|GPIO_PIN_3|GPIO_PIN_4|GPIO_PIN_5|GPIO_PIN_6|GPIO_PIN_7);
    GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE,
GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3|GPIO_PIN_4|GPIO_PIN_5|GPIO_PIN_6);
    //Se configuran como entrada los push de los distintos parqueos
    GPIOPinTypeGPIOInput(GPIO_PORTC_BASE, GPIO_PIN_6|GPIO_PIN_7);
    GPIOPinTypeGPIOInput(GPIO_PORTE_BASE, GPIO_PIN_2|GPIO_PIN_3);
    //IntMasterEnable();
    UART1config();

//*****MAIN LOOP*****
    while(1){
        //Se leen las entradas de los sensores (botones) para cada parqueo

```

```

Parqueo1 = GPIOPinRead(GPIO_PORTA_BASE, GPIO_PIN_2);
Parqueo2 = GPIOPinRead(GPIO_PORTA_BASE, GPIO_PIN_3);
Parqueo3 = GPIOPinRead(GPIO_PORTC_BASE, GPIO_PIN_6);
Parqueo4 = GPIOPinRead(GPIO_PORTC_BASE, GPIO_PIN_7);

//Para cada parqueo, se evalua si la variable correspondiente leída
anteriormente
//está en 0 y se enciende la luz verde y apaga la luz roja, y viceversa
/**PARQUEO 1**
if (Parqueo1 == 0){ //Parqueo 1 esta ocupado
    GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_4|GPIO_PIN_5, 16); //Se enciende
led roja
    Disp &= ~(1); //Clear del bit 0
}
else{ //Parqueo 1 esta disponible
    GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_4|GPIO_PIN_5, 32); //Se enciende
led verde
    Disp |= 1; //Set del bit 0
}

/**PARQUEO 2**
if (Parqueo2 == 0){ //Parqueo 2 esta ocupado
    GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_2|GPIO_PIN_3, 4); //Se enciende
led roja
    Disp &= ~(2); //Clear del bit 1
}
else{ //Parqueo 2 esta disponible
    GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_2|GPIO_PIN_3, 8); //Se enciende
led verde
    Disp |= 2; //Set del bit 1
}

/**PARQUEO 3**
if (Parqueo3 == 0){ //Parqueo 3 esta ocupado
    GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_4|GPIO_PIN_5, 16); //Se enciende
led roja
    Disp &= ~(4); //Clear del bit 2
}
else{ //Parqueo 3 esta disponible
    GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_4|GPIO_PIN_5, 32); //Se enciende
led verde
    Disp |= 4; //Set del bit 2
}

/**PARQUEO 4**
if (Parqueo4 == 0){ //Parqueo 4 esta ocupado
    GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_6|GPIO_PIN_7, 64); //Se enciende
led roja
    Disp &= ~(8); //Clear del bit 3
}
else{ //Parqueo 4 esta disponible
    GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_6|GPIO_PIN_7, 128); //Se enciende
led verde
    Disp |= 8; //Set del bit 3
}

```

```

        //Se suma en la variable suma, la cantidad total de parqueos disponibles,
        evaluando cada bit de la variable Disp
        if (1 & Disp){
            suma+=1;
        }
        if (2 & Disp){
            suma+=1;
        }
        if (4 & Disp){
            suma+=1;
        }
        if (8 & Disp){
            suma+=1;
        }

        display(suma); //Se despliega en el display la cantidad total de parqueos
        disponibles
        UARTCharPut(UART1_BASE, Disp); //Se envía el valor de Disp al ESP32
        suma = 0; //Se resetea el valor de suma para la proxima lectura de la
        disponibilidad
    }
}

//*****FUNCIONES*****
****

void UART1config(void){
    //Aqui se configura el UART2
    SysCtlPeripheralEnable(SYSCTL_PERIPH_UART1); //Activar clock para UART2
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOC); //Activar clock para puerto D de la
    tiva
    GPIOPinConfigure(GPIO_PC4_U1RX);
    GPIOPinConfigure(GPIO_PC5_U1TX);
    GPIOPinTypeUART(GPIO_PORTC_BASE, GPIO_PIN_4|GPIO_PIN_5); //Se activan los pines 6
    y 7 del puerto D
    //UARTClockSourceSet(UART1_BASE, UART_CLOCK_PIOSC);
    UARTConfigSetExpClk(UART1_BASE, SysCtlClockGet(), 115200, UART_CONFIG_WLEN_8 |
    UART_CONFIG_STOP_ONE | UART_CONFIG_PAR_NONE);
    UARTEnable(UART1_BASE);
    // UARTIntEnable(UART2_BASE, UART_INT_RT | UART_INT_RX); //Se activa interrupcion
    cada vez que se reciba un dato
    // UARTIntRegister(UART2_BASE, UARTIntHandler); //Se le coloca el nombre a la
    funcion del handler
}

void display(uint8_t valor){
    //Dependiendo del valor del parametro se encienden ciertos leds del display para
    mostrar visualmente el numero correspondiente
    switch(valor){

```

```

        case 0:
            GPIOPinWrite(GPIO_PORTB_BASE,
GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3|GPIO_PIN_4|GPIO_PIN_5|GPIO_PIN_6, 95);
//Se encienden los pines para mostrar un 0
            break;
        case 1:
            GPIOPinWrite(GPIO_PORTB_BASE,
GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3|GPIO_PIN_4|GPIO_PIN_5|GPIO_PIN_6, 80);
//Se encienden los pines para mostrar un 1
            break;
        case 2:
            GPIOPinWrite(GPIO_PORTB_BASE,
GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3|GPIO_PIN_4|GPIO_PIN_5|GPIO_PIN_6, 62);
//Se encienden los pines para mostrar un 2
            break;
        case 3:
            GPIOPinWrite(GPIO_PORTB_BASE,
GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3|GPIO_PIN_4|GPIO_PIN_5|GPIO_PIN_6, 122);
//Se encienden los pines para mostrar un 3
            break;
        case 4:
            GPIOPinWrite(GPIO_PORTB_BASE,
GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3|GPIO_PIN_4|GPIO_PIN_5|GPIO_PIN_6, 113);
//Se encienden los pines para mostrar un 4
            break;
    }
}

```

## **ESP32 en Arduino:**

```

/*****
*****

```

ESP32 Web Server

Ejemplo de creación de Web server

Basándose en los ejemplos de:

<https://lastminuteengineers.com/creating-esp32-web-server-arduino-ide/>

<https://electropeak.com/learn>

```

*****
*****/

```

```

//*****
*****

```

// Librerías

```

//*****
*****

#include <WiFi.h>

#include <WebServer.h>

//*****
*****

// Variables globales

//*****
*****

// SSID & Password

const char* ssid = "Nexxt_456560"; // Enter your SSID here

const char* password = "WadUFE3n"; //Enter your Password here


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


uint8_t suma;

unsigned char Disp;

String disponible = "\t <td style=\"background-color: #5BB564;\"> <h2>Disponible
&#128513</h2> </td>\t\n";

String ocupado = "\t <td style=\"background-color: #E64545;\"><h2>Ocupado
&#128546</h2></td>\t\n";


//*****
*****

// Configuración

//*****
*****

void setup() {

```



```
Serial.begin(115200);

Serial2.begin(115200, SERIAL_8N1, 16, 17);

while (! Serial2);


Serial.println("Try Connecting to ");

Serial.println(ssid);


pinMode(LED1pin, OUTPUT);


// 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.onNotFound(handle_NotFound);


server.begin();
```

```

Serial.println("HTTP server started");

delay(100);

}

//*****
*****

// loop principal

//*****
*****

void loop() {

    //Primero se lee el dato recibido del UART2 que viene de la tiva c
    if (Serial2.available() > 0) { //Solo entra si hay datos en el buffer serial
        Disp = Serial2.read();
    }

    suma = 0;

    //Codigo para sumar la cantidad de parqueos disponibles
    if (1 & Disp) { //Si el bit 0 de Disp esta encendido, se suma 1
        suma += 1;
    }

    if (2 & Disp) { //Si el bit 1 de Disp esta encendido, se suma 1
        suma += 1;
    }

    if (4 & Disp) { //Si el bit 2 de Disp esta encendido, se suma 1
        suma += 1;
    }

    if (8 & Disp) { //Si el bit 3 de Disp esta encendido, se suma 1
        suma += 1;
    }
}

```

```

server.handleClient();
}

//*****
*****

// Handler de Inicio página

//*****
*****

//Cada vez que se refresque la pagina, se enviara de nuevo el codigo del html
void handle_OnConnect() {
    server.send(200, "text/html", SendHTML2());
}

//*****
*****

// Procesador de HTML

//*****
*****

String SendHTML2() {
    String code = "<!DOCTYPE html> \n";
    code += "\t<html>\n";
    code += " <head><meta name=\\\\"viewport\\" content=\\\\"width=device-width, initial-
scale=1.0, user-scalable=no\\">\n";
    code += " <title>Parqueomatic</title>\n";
    code += " <style>html { font-family: Helvetica; display: inline-block; margin: 0px auto; text-
align: center;}\n";
    code += " body{margin-top: 50px;} h1 {color: #141C87;margin: 50px auto 30px;} h3 {color:
#141C87;margin-bottom: 50px;}\n";
    code += " p {font-size: 14px;color: #888;margin-bottom: 10px;}\n";
    code += " div {\n";

```

```
code += " width: 180px;\n";
code += " padding: 30px;\n";
code += " font-family: sans-serif;\n";
code += " background-color: gold;\n";
code += " }\n";
code += " \n";
code += "table {\n";
code += " width:70%;\n";
code += " \n";
code += "}\n";
code += "table, th, td {\n";
code += " border: 3px solid black;\n";
code += " border-collapse: collapse;\n";
code += "}\n";
code += "th, td {\n";
code += " padding: 5px;\n";
code += " text-align: center;\n";
code += "}\n";
code += "#t01 td:nth-child(odd) {\n";
code += " background-color: #FDF580;\n";
code += "}\n";
code += "#t01 th {\n";
code += " background-color: black;\n";
code += " color: white;\n";
code += "}\n";
code += " </style>\n";
```

```
//Codigo para que la pagina se refresque cada 0.5 segundos
code += "<script>\n";
code += "<!--\n";
code += "function timedRefresh(timeoutPeriod) {\n";
code += "\tsetTimeout(\"location.reload(true);\",timeoutPeriod);\n";
code += "}\n";
code += "\n";
code += "window.onload = timedRefresh(500);\n";
code += "\n";
code += "// -->\n";
code += "</script>";
```

```
//Codigo del visualizador principal de la pagina web
```

```
code += " </head>\n";
code += " <body>\n";
code += " <h1>Parqueo-matic 3000 &#128664</h1>\n";
code += " <h3>Santiago Fernandez 18171</h3>\n";
code += " <table id=\"t01\" align=\"center\">\n";
code += "\t<tr>\n";
```

```
//Aqui se muestra disponibilidad del parqueo 1
```

```
code += "\t <td> <h2> Parqueo 1 </h2></td>\n";
if ((Disp & 1) == 1) { //Verifica que el bit 0 de Disp este encendido
    code += disponible;
}
else {
    code += ocupado;
```

```
}
```

```
code += "\t</tr>\n";
```

```
code += "\t<tr>\n";
```

```
//Aqui se muestra disponibilidad del parqueo 2
```

```
code += "\t <td><h2> Parqueo 2 </h2></td>\n";
```

```
if ((Disp & 2) == 2) { //Verifica que el bit 1 de Disp este encendido
```

```
    code += disponible;
```

```
}
```

```
else {
```

```
    code += ocupado;
```

```
}
```

```
code += "\t</tr>\n";
```

```
code += "\t<tr>\n";
```

```
//Aqui se muestra disponibilidad del parqueo 3
```

```
code += "\t <td><h2> Parqueo 3 </h2></td>\n";
```

```
if ((Disp & 4) == 4) { //Verifica que el bit 2 de Disp este encendido
```

```
    code += disponible;
```

```
}
```

```
else {
```

```
    code += ocupado;
```

```
}
```

```
code += "\t</tr>\n";
```

```
code += "\t<tr>\n";
```

```

//Aqui se muestra disponibilidad del parqueo 4
code += "\t <td><h2> Parqueo 4 </h2></td>\n";

if ((Disp & 8) == 8) { //Verifica que el bit 3 de Disp este encendido
    code += disponible;
}

else {
    code += ocupado;
}

code += "\t</tr>\n";
code += " </table>\n";
code += " <h2>Parqueos disponibles:</h2>\n";

//Para evaluar los parqueos disponibles, se despliega en letra grande el valor de suma
convertido en string
code += " <font size=7><span style=\"border: 3px solid black\">" + String(suma) +
"</span></font>\n";
code += "\t</body>\n";
code += " </html>";

return code;
}

//*****
*****

// Handler de not found

//*****
*****

void handle_NotFound() {
    server.send(404, "text/plain", "Not found");
}

```