Proyecto #4 - Parqueos

Erick Daniel Aquino 17319

Universidad del Valle de Guatemala - IE3027-Electrónica Digital 2

aqu17319@uvg.edu.gt

GITHUB, YOUTUBE

Circuito armado y funcionando

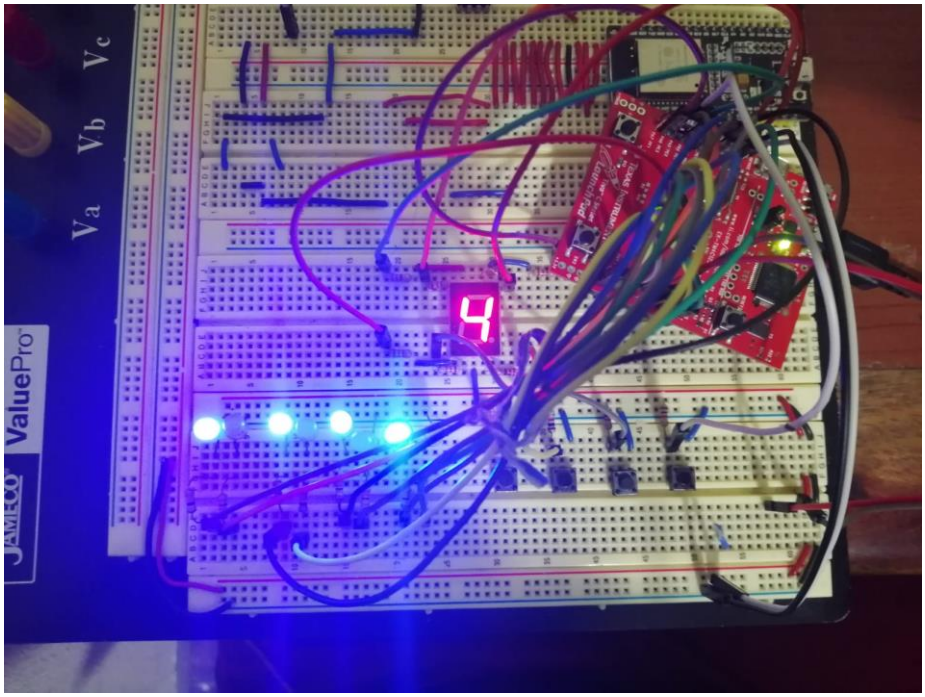

Figura No.1 – Los 4 parqueos disponibles

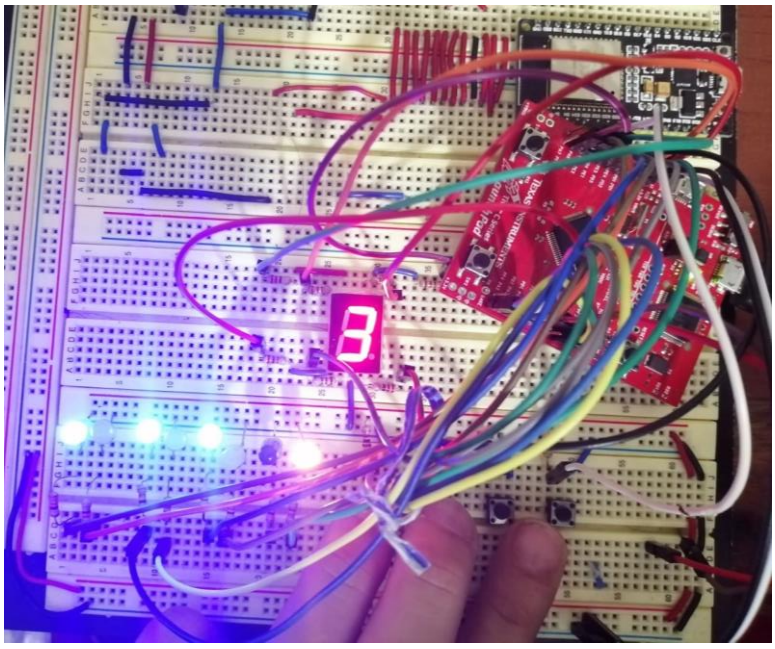

Figura No.2 – Tres parqueos disponibles y cambia el parque ocupado a amarillo.

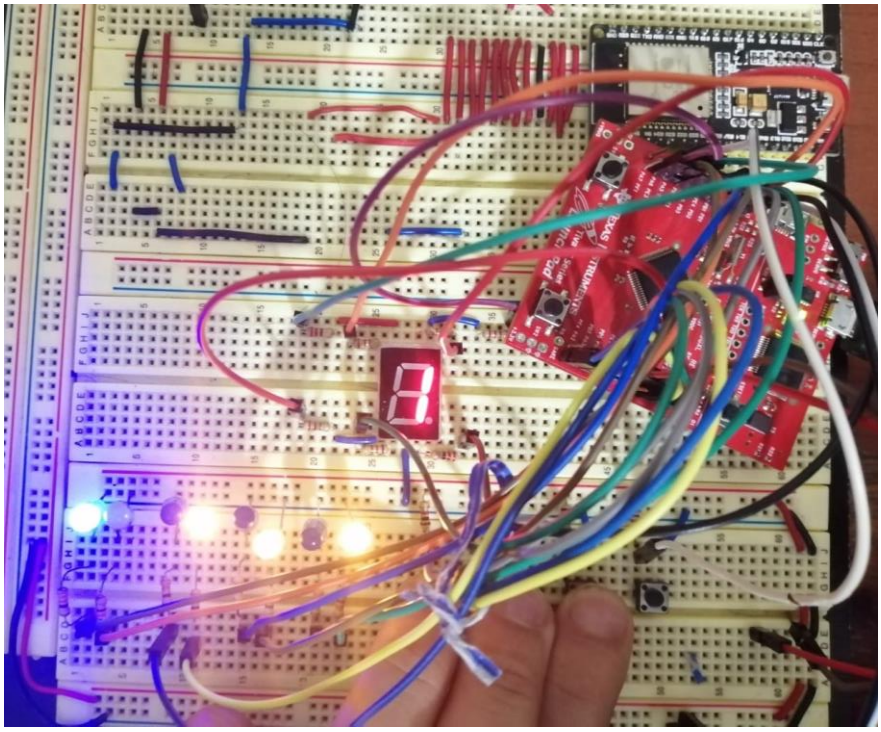

Figura No.3 – 3 parqueos ocupados y 1 parqueo disponible

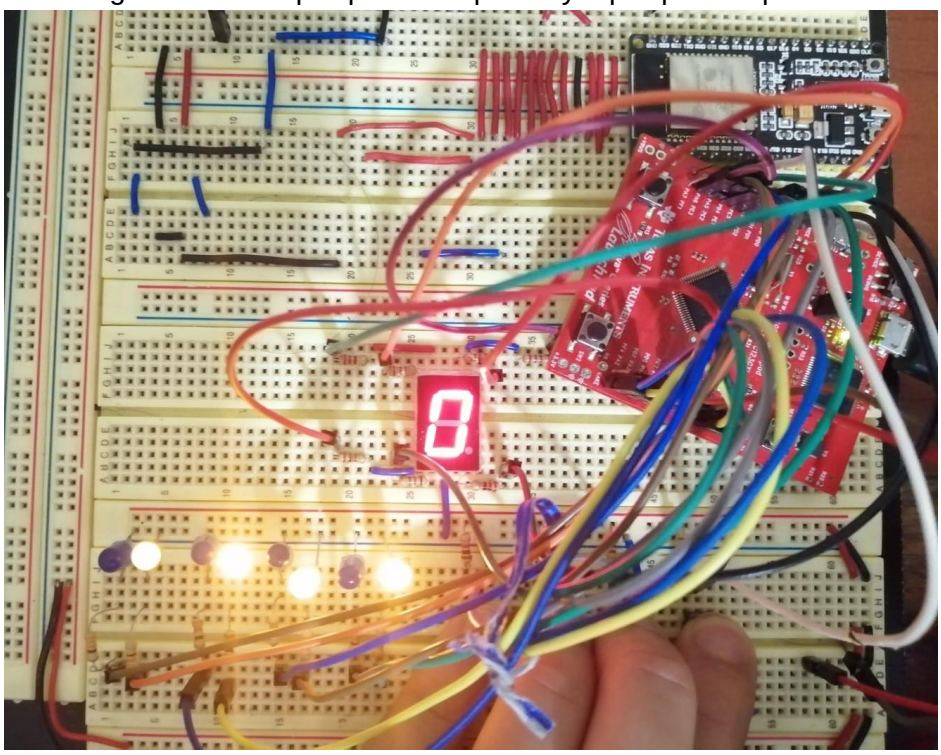

Figura No.4 – Todos los parqueos ocupados y leds amarillas encendidas.

Figura No.5 – Pagina web con todos los parqueos ocupados

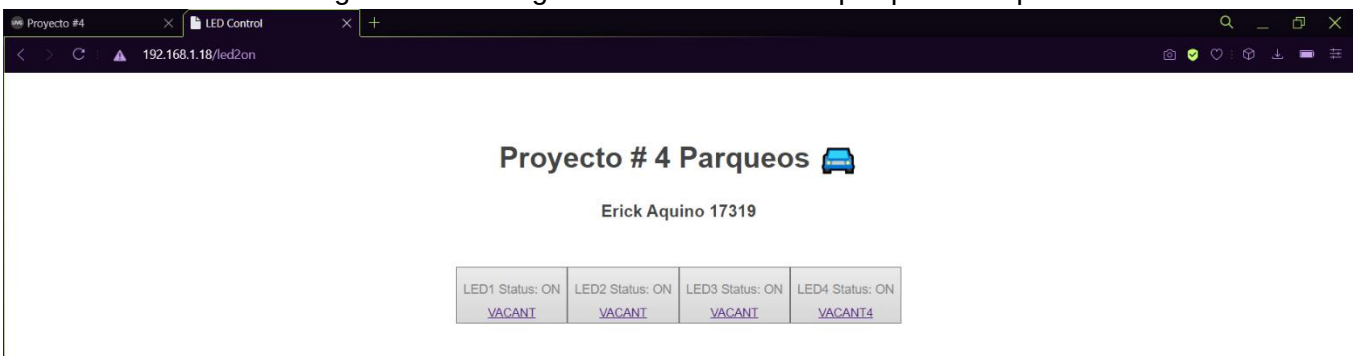

Figura No.6 – Pagina web con todos los parqueos disponibles

Funcionamiento:

Los push buttons representan los sensores, entonces cuando están sin presionar el estado del parqueo está en vacío, representando un parqueo vacío con el led azul encendido. Cuando un push button se presiona el estado del parqueo cambia a ocupado y el led azul se apaga para encenderse el led amarillo y el display mostrara la cantidad de parqueos vacíos.

No importa el orden en que se presionen los push buttons el display siempre mostrará los parqueos disponibles.

Cuando un push button es presionado hay una variable que cambia es enviada a través de la comunicación serial al esp32 y del esp32 cambia el estado en la página web. Cada vez que esta variable cambia en la pagina web se redirige a otra dirección, ya sea “led1on” o “led1off”, “led2on” ó “led2off” y así para el led 3 y 4, esto es para que cuando se redireccione a una página el botón diga “VACANT” diciendo que esta vacío ese parqueo ó “PARKED” diciendo que está ocupado ese parqueo.

Código Code Composer:

```
int main(void)
{

    SysCtlClockSet(
    SYSCTL_SYSDIV_5 | SYSCTL_USE_OSC | SYSCTL_OSC_MAIN | SYSCTL_XTAL_16MHZ); // Reloj de 40MHz
    //Timer0Config(); // Configuracion del TIMER 0
    //InitUART(); // Configuracion de UART 0
    initGPIOF(); //configuracion de leds y puerto F
    uartc(); // configurar el uart1 en el puerto c

    while (1)
    {
        push();
        parqueo();
        display();
        UARTCharPutNonBlocking(UART1_BASE, cont);

    }
}
```

Figura No.7 – Main code y sus funciones

En esta parte del código lo que hago es configurar la frecuencia de reloj a 40MHz, con la función InitGPIOF configuro el puerto B,D,E y F como salidas y el puerto A como entradas, como se muestra en la figura No.8.
Con las funciones dentro del while son las qu quiero que se repitan para leer el estado de los push buttons tal como lo es la función “push()”
La función “parqueo ()” hago que se enciendan o se apaguen las luces azules según el estado de los push buttons. Dentro de esta función también hay un contador que suma o resta de 1 en 1 según el push button se presione o se suelte. El valor que contenga el contador será de utilidad para la función “display()” el cual dependiendo del numero del contador hará que muestre un valor en el display de 7 segmentos.
Por ultimo envio por el UART1 del puerto C un char con el valor del contador el cual lo leerá la esp32 y hace el cambio en la pagina web.

```
void initGPIOF(void)
{
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOE);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);

    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE,
    GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_3);
    GPIOPinTypeGPIOInput(GPIO_PORTA_BASE,
    GPIO_PIN_2 | GPIO_PIN_3 | GPIO_PIN_4 | GPIO_PIN_5);
    GPIOPinTypeGPIOOutput(
            GPIO_PORTD_BASE,
            GPIO_PIN_0 | GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_3 | GPIO_PIN_6
                    | GPIO_PIN_7);
    GPIOPinTypeGPIOOutput(
            GPIO_PORTE_BASE,
            GPIO_PIN_0 | GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_3 | GPIO_PIN_4
                    | GPIO_PIN_5);
    GPIOPinTypeGPIOOutput(
            GPIO_PORTB_BASE,
            GPIO_PIN_0 | GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_3 | GPIO_PIN_4
                    | GPIO_PIN_5);
}
```

Figura No.8 – Configuración de puertos como entrada y salida

```
void uartc(void)
{
    //ENABLE UART PERIPHERAL
    SysCtlPeripheralEnable(SYSCTL_PERIPH_UART1); //Enable the peripheral UART Module 3
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOC); //Enable the GPIO Port C
    //SET RX AND TX PINS
    GPIOPinConfigure(GPIO_PC4_U1RX);
    GPIOPinConfigure(GPIO_PC5_U1TX);
    GPIOPinTypeUART(GPIO_PORTC_BASE, GPIO_PIN_4 | GPIO_PIN_5);
    //BAUDRATE
    UARTConfigSetExpClk(
            UART1_BASE, SysCtlClockGet(), 115200,
            (UART_CONFIG_WLEN_8 | UART_CONFIG_STOP_ONE | UART_CONFIG_PAR_NONE));
    UARTEnable(UART1_BASE);
}
```

Figura No.9 – Configuración del UART1 del puerto C

```
void push(void)
{

    //antirrebote push 1
    push1 = GPIOPinRead(GPIO_PORTA_BASE, GPIO_PIN_2);
    if (((push1 & GPIO_PIN_2) == 0) && anti1 == 0)
    {
        anti1 = 1;
        anti11 = 0;
        cont--;
    }
    push1 = GPIOPinRead(GPIO_PORTA_BASE, GPIO_PIN_2);
    if (((push1 & GPIO_PIN_2) != 0) && (anti11 == 0))
    {
        anti1 = 0;
        anti11 = 1;
        cont++;
    }
```

Figura No.10 – Función Push() el cual cambia de estado y suma o resta al contador

En la figura no.10 tenemos un antirrebote lo cual hace que aunque mantengamos presionado o se mantenga sin presionar solo sumará o restará en una unidad a la variable “cont” la cual es el contador de los parqueos disponibles y el valor que mandamos a través de la UART1. Este código se repite para los botones, 2,3 y 4.

```
if (cont > 4)
{
    cont = 4;
}
if (cont < 0)
{
    cont = 0;
}
```

Figura No.11 – Limitar el rango del contador

El código de la figura No.11 se encuentra al final de la función “push()” y estas líneas de código limitan al contador que sume hasta 4 y que al restar llegue hasta 0.

```
void display(void)
{
    //MOSTRAR 4
    if (cont == 4)
    {
        GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_1, GPIO_PIN_1);  //1
        GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_2, 0);  //2
        GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_3, 0);  //3
        GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_4, GPIO_PIN_4);  //4
        GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_0, GPIO_PIN_0);  //0 pin medio
        GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_5, 0);  //5
        GPIOPinWrite(GPIO_PORTE_BASE, GPIO_PIN_3, GPIO_PIN_3);  //6
    }
    //MOSTRAR 3
    if (cont == 3)
    {
        GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_1, GPIO_PIN_1);  //1
        GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_2, GPIO_PIN_2);  //2
        GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_3, 0);  //3
        GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_4, 0);  //4
        GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_0, GPIO_PIN_0);  //0 pin medio
        GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_5, GPIO_PIN_5);  //5
        GPIOPinWrite(GPIO_PORTE_BASE, GPIO_PIN_3, GPIO_PIN_3);  //6
```

Figura No.12 – Valores mostrados en el display

La función display mostrará un numero en el display físico dependiendo del contador el cual es controlado por los push buttons.

Codigo en C para el esp32

```
Serial.begin(115200);

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

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.on("/led1on", handle_led1on);
server.on("/led1off", handle_led1off);
server.on("/led2on", handle_led2on);
server.on("/led2off", handle_led2off);
server.on("/led3on", handle_led3on);
server.on("/led3off", handle_led3off);
server.on("/led4on", handle_led4on);
server.on("/led4off", handle_led4off);

server.onNotFound(handle_NotFound);

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

Figura No.13 – función setup()

En esta función inicio la comunicación serial la primer es para la conexión con la página web y el Serial2 es para recibir los datos que manda la tiva c al esp32. También enciendo las pagina a las cuales se redirecciona cuando un estado cambia de ocupado a vacante y de vacante a ocupado.

```
//********************************************************************************************************
// Handler de Inicio página
//********************************************************************************************************
void handle_OnConnect() {
  LED1status = LOW;
  Serial.println("GPIO2 Status: OFF");
  server.send(200, "text/html", SendHTML(LED1status));

  LED2status = LOW;
  Serial.println("GPIO3 Status: OFF");
  server.send(200, "text/html", SendHTML(LED2status));

  LED3status = LOW;
  Serial.println("GPIO4 Status: OFF");
  server.send(200, "text/html", SendHTML(LED3status));

  LED4status = LOW;
  Serial.println("GPIO5 Status: OFF");
  server.send(200, "text/html", SendHTML(LED4status));
}
```

Figura no.14 - Inicio de los botones como OFF es decir que están vacíos

```
void handle_led1on() {
  LED1status = HIGH;
  Serial.println("GPIO2 Status: ON");
  server.send(200, "text/html", SendHTML(LED1status));

  LED2status = HIGH;
  Serial.println("GPIO3 Status: ON");
  server.send(200, "text/html", SendHTML(LED2status));

  LED3status = HIGH;
  Serial.println("GPIO4 Status: ON");
  server.send(200, "text/html", SendHTML(LED3status));

  LED4status = HIGH;
  Serial.println("GPIO5 Status: ON");
  server.send(200, "text/html", SendHTML(LED4status));
}
void handle_led2on() {
  LED2status = HIGH;
  Serial.println("GPIO3 Status: ON");
  server.send(200, "text/html", SendHTML(LED2status));

}
void handle_led3on() {
  LED3status = HIGH;
  Serial.println("GPIO4 Status: ON");
  server.send(200, "text/html", SendHTML(LED3status));
}

void handle_led4on() {
  LED3status = HIGH;
  Serial.println("GPIO5 Status: ON");
  server.send(200, "text/html", SendHTML(LED3status));
}
```

Figura No.15 – texto mostrado cuando un botón pasa al estado ON

```
void handle_led1off() {
  LED1status = LOW;
  Serial.println("GPIO2 Status: OFF");
  server.send(200, "text/html", SendHTML(LED1status));

  LED2status = LOW;
  Serial.println("GPIO3 Status: OFF");
  server.send(200, "text/html", SendHTML(LED2status));

  LED3status = LOW;
  Serial.println("GPIO4 Status: OFF");
  server.send(200, "text/html", SendHTML(LED3status));

  LED4status = LOW;
  Serial.println("GPIO5 Status: OFF");
  server.send(200, "text/html", SendHTML(LED4status));
}
void handle_led2off() {
  LED2status = LOW;
  Serial.println("GPIO3 Status: OFF");
  server.send(200, "text/html", SendHTML(LED2status));

}
void handle_led3off() {
  LED3status = LOW;
  Serial.println("GPIO4 Status: OFF");
  server.send(200, "text/html", SendHTML(LED3status));
}
void handle_led4off() {
  LED3status = LOW;
  Serial.println("GPIO5 Status: OFF");
  server.send(200, "text/html", SendHTML(LED3status));
}
//**************************************************************************
```

Figura No.16 – Texto mostrado cuando un botón pasa al estado OFF

```
//*****************************************************************************************
String SendHTML(uint8_t led1stat) {
  String ptr = "<!DOCTYPE html> <html>\n";
  ptr += "<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0, user-scalable=no\">\n";
  ptr += "<title>LED Control</title>\n";
  ptr += "<style>html { font-family: Helvetica; display: inline-block; margin: 20px auto; text-align: center;}\n";
  ptr += "body{margin-top: 50px;} h1 {color: #444444;margin: 50px auto 30px;} h3 {color: #444444;margin-bottom: 50px;}\n";

  ptr += "p {font-size: 14px;color: #888;margin-bottom: 10px;}\n";
  ptr += "</style>\n";
  ptr += "</head>\n";
  ptr += "<body>\n";
  ptr += "<h1>Proyecto # 4 Parqueos &#128664</h1>\n";
  ptr += "<h3>Erick Aquino 17319</h3>\n";

  //BOTON1
  ptr += "<button>";
  if (led1stat)
  {
    ptr += "<p>LED1 Status: ON</p><a class=\"button button-off\" href=\"/led1off\">VACANT</a>\n";
  }
  else
  {
    ptr += "<p>LED1 Status: OFF</p><a class=\"button button-on\" href=\"/led1on\">PARKED</a>\n";
  }
  //ptr += "<script>function timedRefresh(a){setTimeout(\"location.reload(true);\",a)}window.onload=timedRefresh(1000);</script>\n";
  ptr += "</button>";

  //BOTON2
  ptr += "<button>";

  if (led1stat)
  {
    ptr += "<p>LED2 Status: ON</p><a class=\"button button-off\" href=\"/led2off\">VACANT</a>\n";
  }
  else
  {
    ptr += "<p>LED2 Status: OFF</p><a class=\"button button-on\" href=\"/led2on\">PARKED</a>\n";
  }
  //ptr += "<script>function timedRefresh(a){setTimeout(\"location.reload(true);\",a)}window.onload=timedRefresh(1000);</script>\n";
  ptr += "</button>";



  //BOTON3
  ptr += "<button>";

  if (led1stat)
  {
    ptr += "<p>LED3 Status: ON</p><a class=\"button button-of\" href=\"/led3off\">VACANT</a>\n";
  }
  else
  {
    ptr += "<p>LED3 Status: OFF</p><a class=\"button button-on\" href=\"/led3on\">PARKED</a>\n";
  }
  //ptr += "<script>function timedRefresh(a){setTimeout(\"location.reload(true);\",a)}window.onload=timedRefresh(1000);</script>\n";
  ptr += "</button>";


  //BOTON4
  ptr += "<button>";

  if (led1stat)
  {
    ptr += "<p>LED4 Status: ON</p><a class=\"button button-off\" href=\"/led4off\">VACANT4</a>\n";
  }
  else
  {
    ptr += "<p>LED4 Status: OFF</p><a class=\"button button-on\" href=\"/led4on\">PARKED4</a>\n";
  }
  //ptr += "<script>function timedRefresh(a){setTimeout(\"location.reload(true);\",a)}window.onload=timedRefresh(1000);</script>\n";
  ptr += "</button>";


  ptr += "</body>\n";
  ptr += "</html>\n";
  return ptr;
}
//*****************************************************************************************
```

Figura No.14 – Mezcla entre C y HTML para la pagina web