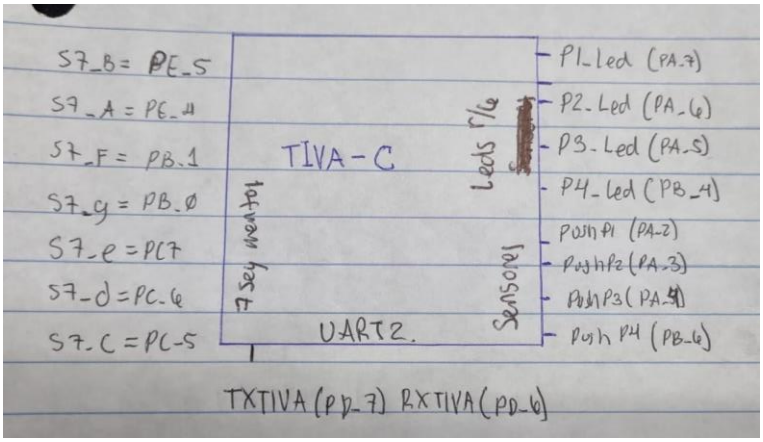
Jorge Rafael Hurtado Garcia 18052

Proyecto 4 Digital 2

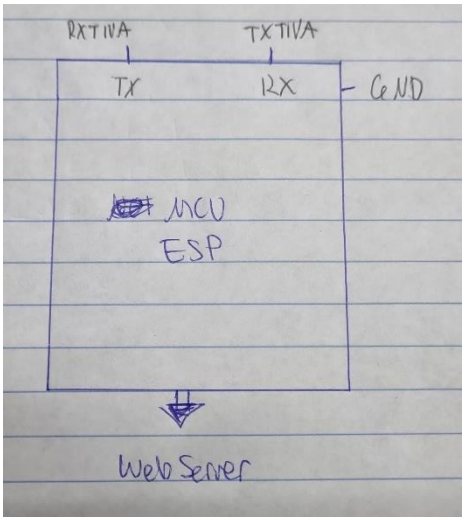
Se realizo la parte de lectura de sensores, muestra de ocupado o libre y cantidad de parqueos libres en el microcontrolador TIVAC mientras que el Web Server donde se pueden ver estos datos en tiempo real se realizo por medio del ESP. La comunicación entre estos dos MCUs es por medio de Comunicación UART. Para los sensores se utilizaron PushButtons para simular el carro estacionado, se recomienda utilizar sensores infrarrojos o de proximidad para detectar esto de forma más precisa y real.

Circuito utilizado:

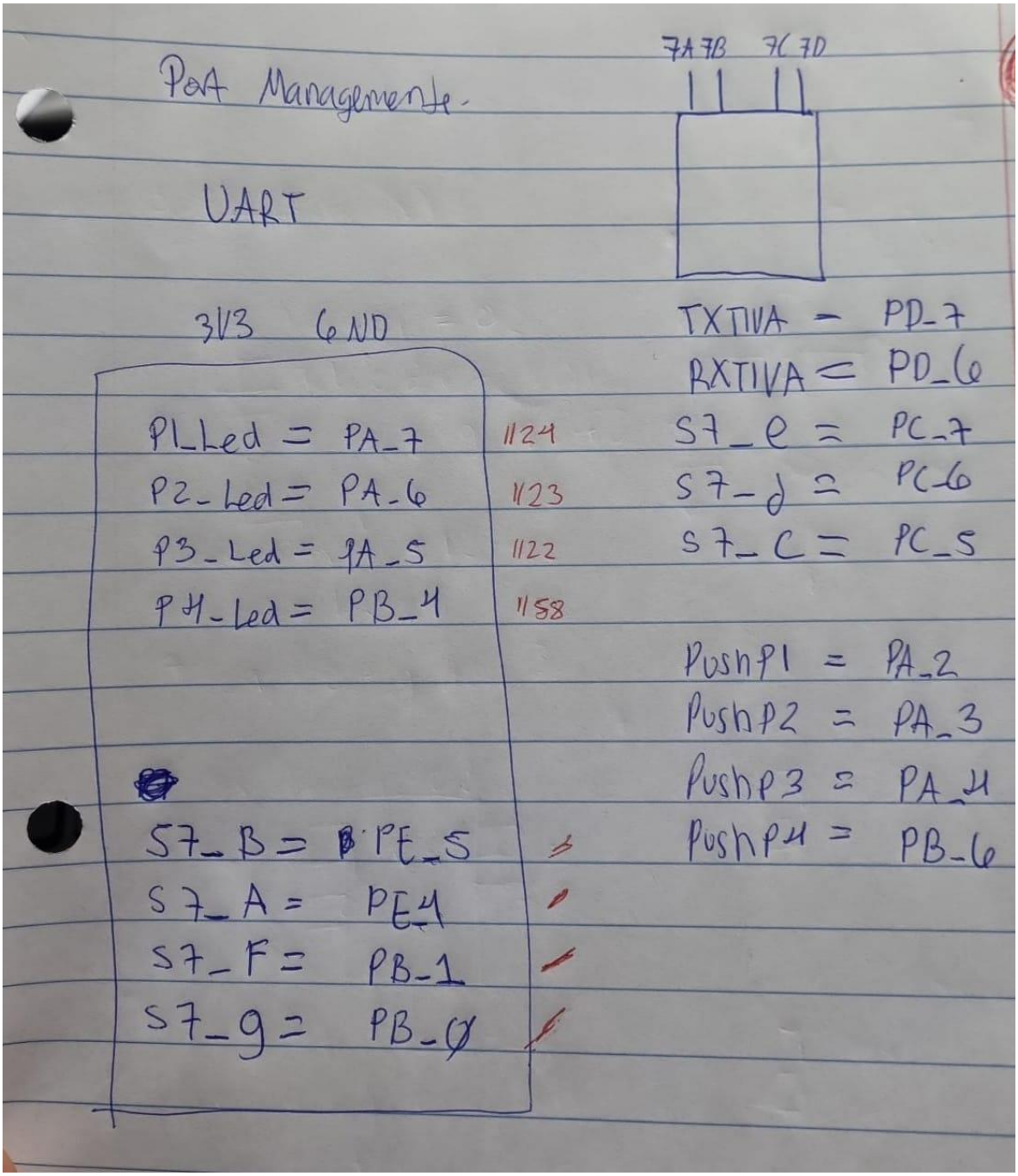
El circuito implementado en la TIVAC es el siguiente.



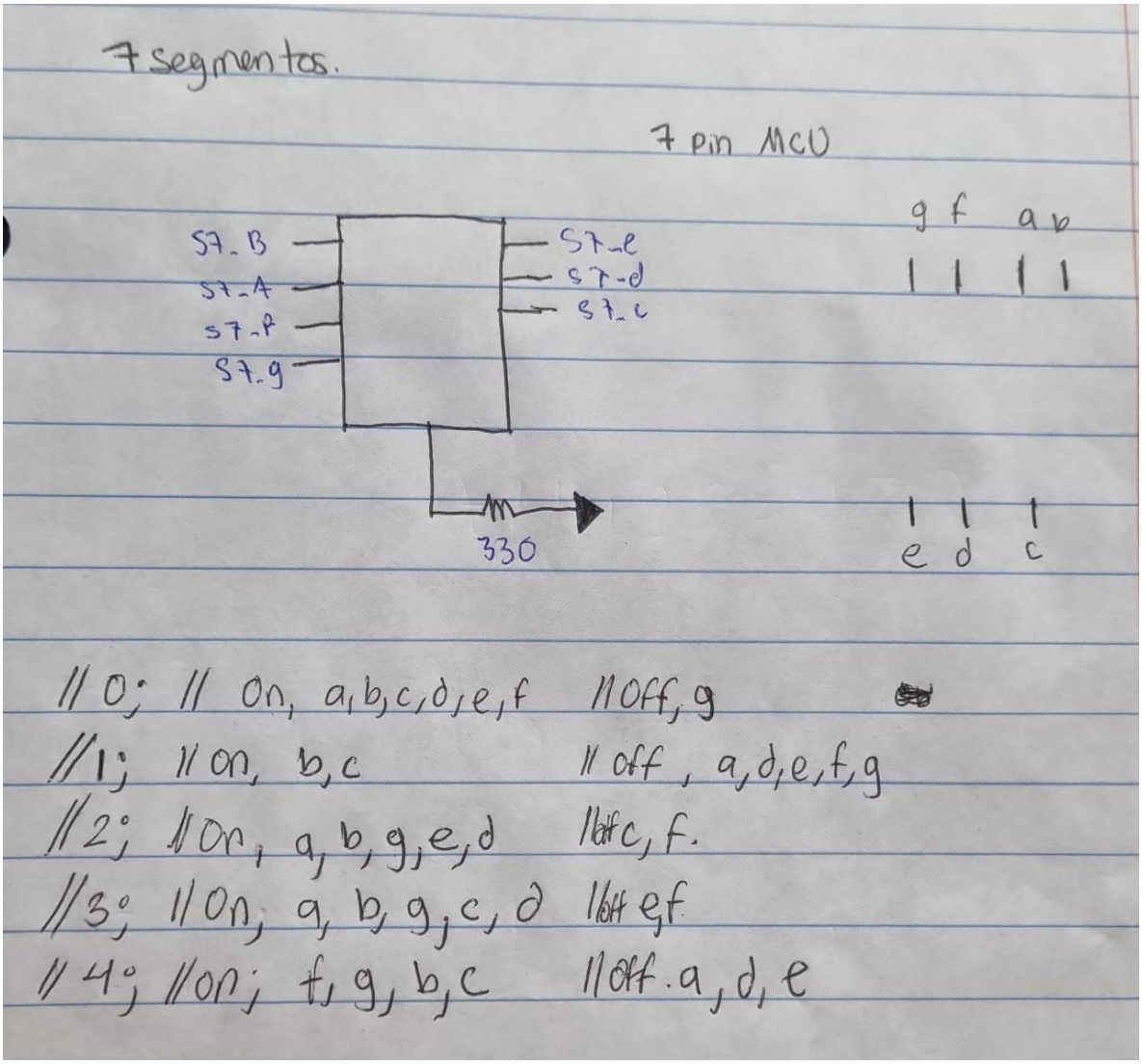
El circuito implementado en el ESP es el siguiente.



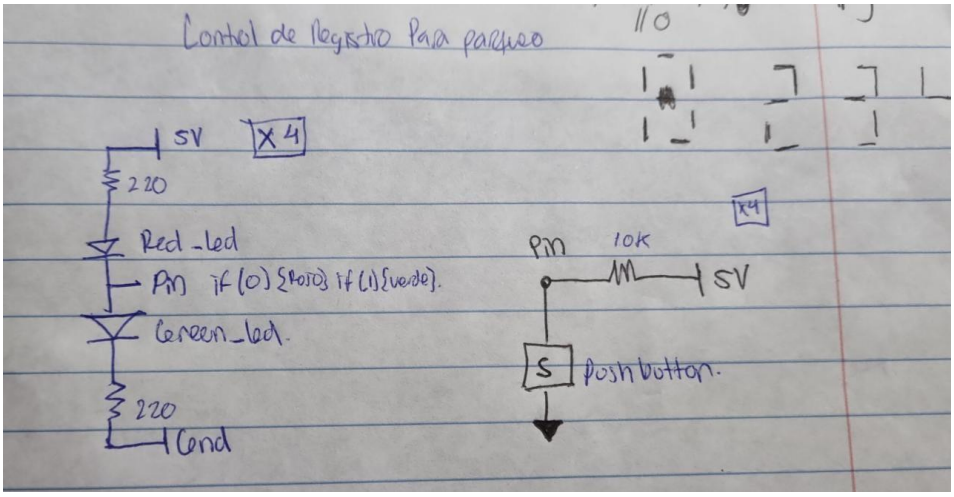
Datos y Port Management TIVAC



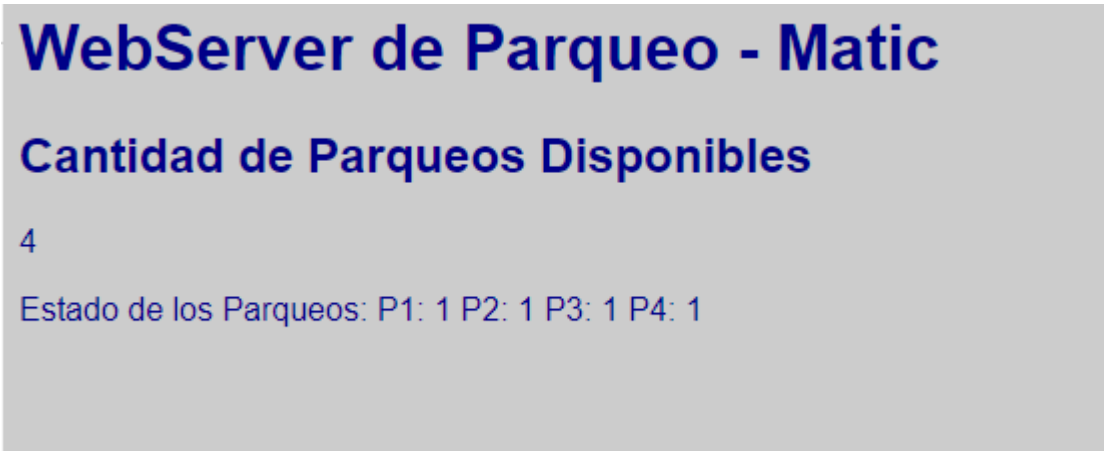
7 segmentos Pin Map && Number map



Sensor and LEDs Schematic



Web Server Display



Se tiene la lectura de los push por medio de la tiva c la cual luego es enviada sin procesar al esp, luego esta lectura se procesa en la tivac para conocer que parqueo esta disponible y cual ocupado. Conociendo si el parqueo esta ocupado o libre se procede a encender o apagar un pin el cual es el mismo ya que como los leds son diodos se aprovecha de esto para manejar solo con un pin los dos leds Set-Verde Reset-rojo. Mientras que luego esta información se "concatena" por medio de una suma y esta nos da la cantidad de parqueos disponibles. Esta información se procesa por medio de una serie de ifs que contienen el mapeo digital del display de 7 segmentos logrando así imprimir este dato directamente en el display.

Desde el lado del Esp se realizó el webserver dejándolo dependiente solamente de 5 variables siendo estas los 4 sensores y la suma. Los datos que vienen por medio de uart contienen los valores en tiempo real de los sensores y al recibirlos en el esp se "imprimen" en las variables que se alistaron del webserver y se procede a hacer la suma para saber cuántos parqueos están disponibles.

Código Comentado de la TIVAC por medio de CCS.

```
//Jorge Rafael Hurtado Garcia
//18052
//Programacion de TIVAC para sistema de parqueos


//Llamado de Librerias
#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 "inc/hw_gpio.h"
#include "driverlib/sysctl.h"
#include "driverlib/interrupt.h"
#include "driverlib/gpio.h"
#include "driverlib/uart.h"
#include "driverlib/pin_map.h"

int main(void){

    //Set System Clock

SysCtlClockSet(SYSCTL_SYSDIV_1|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAIN)
;
    //Enable Peripheral Control GPIO
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOC);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOE);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    //Prepare System for Uart2
    HWREG(GPIO_PORTD_BASE + GPIO_O_LOCK) = GPIO_LOCK_KEY;
    HWREG(GPIO_PORTD_BASE + GPIO_O_CR) |= GPIO_PIN_7;
    //Config UART2 pinout Config BaudRate 115200
    GPIOPinConfigure(GPIO_PD7_U2TX);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_UART2); // enable uart2
    GPIOPinTypeUART(GPIO_PORTD_BASE, GPIO_PIN_6 | GPIO_PIN_7); // pines de
control del uart
    UARTConfigSetExpClk(UART2_BASE, SysCtlClockGet(), 115200,
(UART_CONFIG_WLEN_8 | UART_CONFIG_STOP_ONE | UART_CONFIG_PAR_NONE));
    UARTIntClear(UART2_BASE, UART_INT_RX | UART_INT_RT | UART_INT_TX |
UART_INT_FE | UART_INT_PE | UART_INT_BE | UART_INT_OE | UART_INT_RI |
UART_INT_CTS | UART_INT_DCD | UART_INT_DSR);
    //Assign GPIO Inputs for Sensors
    GPIOPinTypeGPIOInput(GPIO_PORTA_BASE, GPIO_PIN_2|GPIO_PIN_3|GPIO_PIN_4);
    GPIOPinTypeGPIOInput(GPIO_PORTB_BASE, GPIO_PIN_6);
    //Assign GPIO Outputs for Leds and 7 segments Display.
    GPIOPinTypeGPIOOutput(GPIO_PORTA_BASE, GPIO_PIN_5|GPIO_PIN_6|GPIO_PIN_7);
    GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_4);
    GPIOPinTypeGPIOOutput(GPIO_PORTC_BASE, GPIO_PIN_5|GPIO_PIN_6|GPIO_PIN_7);
    GPIOPinTypeGPIOOutput(GPIO_PORTE_BASE, GPIO_PIN_4|GPIO_PIN_5);
```

```
    //Variables de Control Display
    uint8_t DISPLAYVALUE=4;
    // Sensor Variables
    uint8_t P1_LED=0;
    uint8_t P2_LED=0;
    uint8_t P3_LED=0;
    uint8_t P4_LED=0;
    //Printing Variables
    uint32_t P1_LED_DISPLAY=1;
    uint32_t P2_LED_DISPLAY=1;
    uint32_t P3_LED_DISPLAY=1;
    uint32_t P4_LED_DISPLAY=1;

    //Loop Para lectura e impresion de datos.
    while(1)
    {
        // Lectura de los PushButtons.

        // If variable Names are not familiar please refer to the associated pdf
file with pinout and Electric Scheme.
        // Same applies for port management in GPIO read and write.

        P1_LED = GPIOPinRead(GPIO_PORTA_BASE, GPIO_PIN_2);
        P2_LED = GPIOPinRead(GPIO_PORTA_BASE, GPIO_PIN_3);
        P3_LED = GPIOPinRead(GPIO_PORTA_BASE, GPIO_PIN_4);
        P4_LED = GPIOPinRead(GPIO_PORTB_BASE, GPIO_PIN_6);
        //Enviamos estado de los PushButtons por UART a la TIVAC
        UARTCharPut(UART2_BASE, P1_LED);
        UARTCharPut(UART2_BASE, P2_LED);
        UARTCharPut(UART2_BASE, P3_LED);
        UARTCharPut(UART2_BASE, P4_LED);
        //Iniciamos la revision de que parqueos estan ocupados.

        //if P1 busy, reset red led (same led, reset>red, set>green )print
parking not available in display(0)
        if(P1_LED==0)
        {
        GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_7, 0);
        P1_LED_DISPLAY=0;
        }
        //if P1 clear, set green led and print parking available in display(1)
        if(P1_LED!=0)
        {
        GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_7, GPIO_PIN_7);
        P1_LED_DISPLAY=1;
        }
        //Same from P1 applies to P2, P3, P4. Uncommented for this reason.
        if(P2_LED==0)
        {
        GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_6, 0);
        P2_LED_DISPLAY=0;
        }
        if(P2_LED!=0)
        {
        GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_6, GPIO_PIN_6);
```

```
        P2_LED_DISPLAY=1;
        }
        if(P3_LED==0)
        {
        GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_5, 0);
        P3_LED_DISPLAY=0;
        }
        if(P3_LED!=0)
        {
        GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_5, GPIO_PIN_5);
        P3_LED_DISPLAY=1;
        }
        if(P4_LED==0)
        {
        GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_4, 0);
        P4_LED_DISPLAY=0;
        }
        if(P4_LED!=0)
        {
        GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_4, GPIO_PIN_4);
        P4_LED_DISPLAY=1;
        }
        //Concat Display Value from the sum of individual values from each
parking.

DISPLAYVALUE=P1_LED_DISPLAY+P2_LED_DISPLAY+P3_LED_DISPLAY+P4_LED_DISPLAY;

        // Display value concat printer. Set the necessary pins for display
numbers. PINMAP available in aditional pdf.
        // Individual Sections uncommented due to if statement is clear enough.
        if(DISPLAYVALUE==4)
        {
           GPIOPinWrite(GPIO_PORTE_BASE, GPIO_PIN_5, GPIO_PIN_5);
           GPIOPinWrite(GPIO_PORTE_BASE, GPIO_PIN_4, 0);
           GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_1, GPIO_PIN_1);
           GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_0, GPIO_PIN_0);
           GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_7, 0);
           GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_6, 0);
           GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_5, GPIO_PIN_5);
        }

        if(DISPLAYVALUE==3)
        {
           GPIOPinWrite(GPIO_PORTE_BASE, GPIO_PIN_5, GPIO_PIN_5);
           GPIOPinWrite(GPIO_PORTE_BASE, GPIO_PIN_4, GPIO_PIN_4);
           GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_1, 0);
           GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_0, GPIO_PIN_0);
           GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_7, 0);
           GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_6, GPIO_PIN_6);
           GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_5, GPIO_PIN_5);
        }
        if(DISPLAYVALUE==2)
        {
           GPIOPinWrite(GPIO_PORTE_BASE, GPIO_PIN_5, GPIO_PIN_5);
           GPIOPinWrite(GPIO_PORTE_BASE, GPIO_PIN_4, GPIO_PIN_4);
```

```
            GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_1, 0);
            GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_0, GPIO_PIN_0);
            GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_7, GPIO_PIN_7);
            GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_6, GPIO_PIN_6);
            GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_5, 0);
        }
        if(DISPLAYVALUE==1)
        {
            GPIOPinWrite(GPIO_PORTE_BASE, GPIO_PIN_5, GPIO_PIN_5);
            GPIOPinWrite(GPIO_PORTE_BASE, GPIO_PIN_4, 0);
            GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_1, 0);
            GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_0, 0);
            GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_7, 0);
            GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_6, 0);
            GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_5, GPIO_PIN_5);
        }
        if(DISPLAYVALUE==0)
        {
            GPIOPinWrite(GPIO_PORTE_BASE, GPIO_PIN_5, GPIO_PIN_5);
            GPIOPinWrite(GPIO_PORTE_BASE, GPIO_PIN_4, GPIO_PIN_4);
            GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_1, GPIO_PIN_1);
            GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_0, 0);
            GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_7, GPIO_PIN_7);
            GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_6, GPIO_PIN_6);
            GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_5, GPIO_PIN_5);
        }
        //In case something breaks display may show 0 available parking spots to
prevent mayor issues.
        else{DISPLAYVALUE=0;}
        //System delay on loop to restart system
        SysCtlDelay(1);
    }

}
```

Código Comentado del ESP por medio de Arduino IDE.

```
/*  Jorge Rafael Hurtado Garcia
 *  18052
 *  Proyecto Final Digital 2
 * */


// se incluyen las librerias necesarias para la implementacion del webserver
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
```

```
#include <ESP8266mDNS.h>
//Se cargan los datos de la red wifi
#ifndef STASSID
#define STASSID "CLARO_6EE204"
#define STAPSK  "495654EFB68s"
//#define STASSID "123456789"
//#define STAPSK  "test1234"
#endif

const char *ssid = STASSID;
const char *password = STAPSK;

ESP8266WebServer server(80);

const int led = 13;
// Se inicia el handle root del webserver
void handleRoot() {
  digitalWrite(led, 1);
  char temp[400];
  int sec = millis() / 1000;
  int min = sec / 60;
  int hr = min / 60;
  //variable de recepcion serial
  uint8_t P[4];
  //Variables de impresion en Webserver
  int CantParqueos = 4;
  int P1= 2;
  int P2= 3;
  int P3= 4;
```

```
int P4= 5;
//Lectura serial de los datos que provienen de la tivac
if (Serial.available()>0)
{
  for (int i = 0; i<=3; i++)//se lee 4 bytes y se almacenan en un array
  {
    P[i] = Serial.read();
  }
}
//Map de datos que vienen de la tivac a las variables de impresion
if(P[3]!=0){P1=1;}
if(P[3]==0){P1=0;}
if(P[0]!=0){P2=1;}
if(P[0]==0){P2=0;}
if(P[1]!=0){P3=1;}
if(P[1]==0){P3=0;}
if(P[2]!=0){P4=1;}
if(P[2]==0){P4=0;}
//suma de parqueos disponibles
CantParqueos = P1 + P2 + P3 + P4;
// Sprint con el contenido del webserver colores y especificaciones.
snprintf(temp, 400,

        "<html>\
<head>\
  <meta http-equiv='refresh' content='5'/>\
  <title>Proyecto Digital 2</title>\
  <style>\
    body { background-color: #cccccc; font-family: Arial, Helvetica, Sans-Serif; Color: #000088; }\
```

```
    </style>\
  </head>\
  <body>\
    <h1>WebServer de Parqueo - Matic </h1>\
    <h2> Cantidad de Parqueos Disponibles </h2>\
    <p> %d </p>\
    <p>Estado de los Parqueos: P1: %1d P2: %1d P3: %1d P4: %1d    </p>\
    <img>\
  </body>\
</html>",

          CantParqueos, P1, P2, P3, P4
          );
  server.send(200, "text/html", temp);
  digitalWrite(led, 0);
}

void handleNotFound() {
  digitalWrite(led, 1);
  String message = "File Not Found\n\n";
  message += "URI: ";
  message += server.uri();
  message += "\nMethod: ";
  message += (server.method() == HTTP_GET) ? "GET" : "POST";
  message += "\nArguments: ";
  message += server.args();
  message += "\n";

  for (uint8_t i = 0; i < server.args(); i++) {
```

```
    message += " " + server.argName(i) + ": " + server.arg(i) + "\n";
  }

  server.send(404, "text/plain", message);
  digitalWrite(led, 0);
}

void setup(void) {
  pinMode(led, OUTPUT);
  digitalWrite(led, 0);
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  //Serial.println("");

  // Wait for connection
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    //Serial.print(".");
  }

  //Serial.println("");
  //Serial.print("Connected to ");
  //Serial.println(ssid);
  //Serial.print("IP address: ");
  //Serial.println(WiFi.localIP());

  if (MDNS.begin("esp8266")) {
    //Serial.println("MDNS responder started");
```

```
  }

  server.on("/", handleRoot);

  server.on("/inline", []() {
    server.send(200, "text/plain", "this works as well");
  });
  server.onNotFound(handleNotFound);
  server.begin();
  //Serial.println("HTTP server started");
}

void loop(void) {
  server.handleClient();
  MDNS.update();
}
```