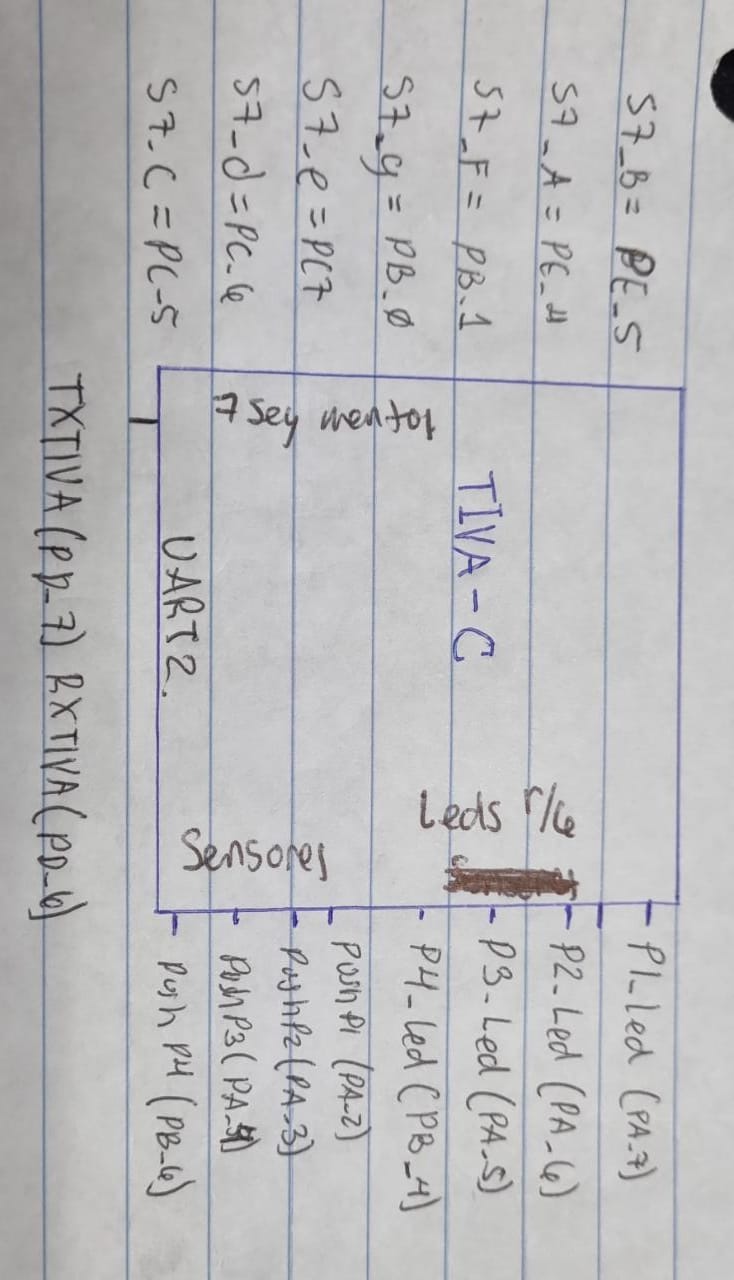
Jorge Rafael Hurtado Garcia 18052

Proyecto 4 Digital 2

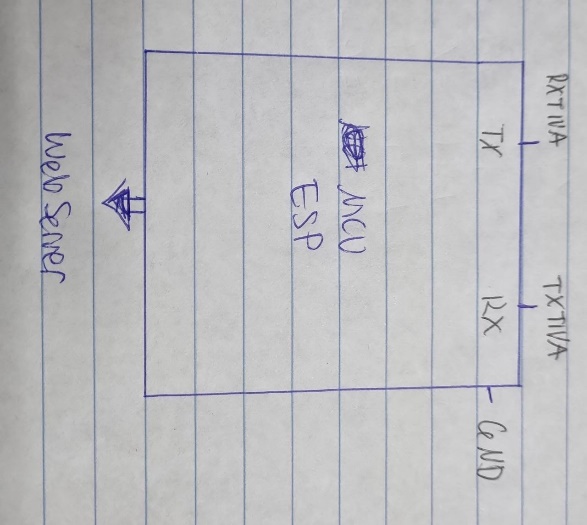
Se realizo la parte de lectura de sensores, muestra de ocupado o libre y cantidad de parqueos libres en el microcontrolador TIVAC mientras que el Web Server donde se pueden ver estos datos en tiempo real se realizo por medio del ESP. La comunicación entre estos dos MCUs es por medio de Comunicación UART. 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:

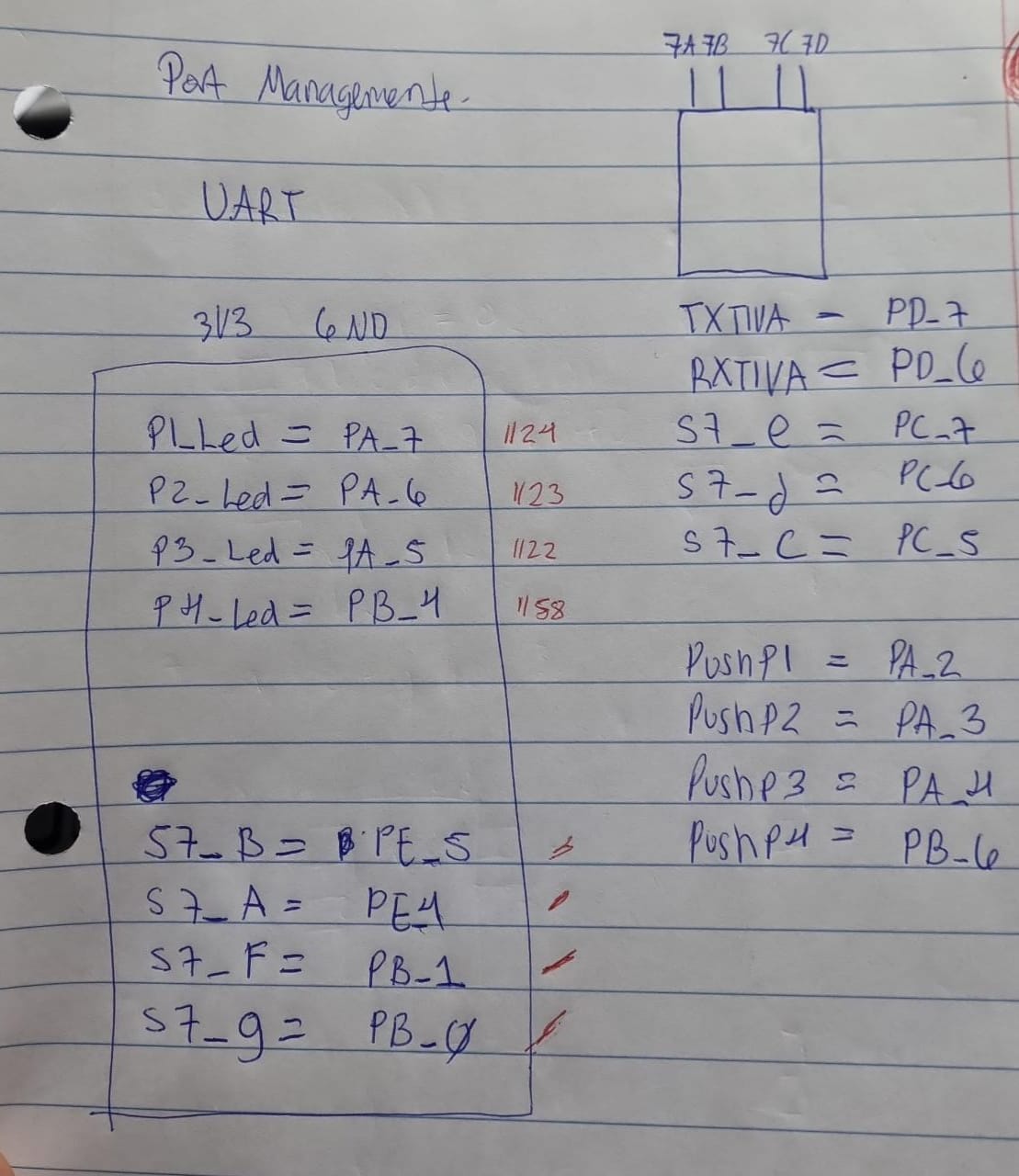
El circuito implementado en la TIVAC es el siguiente.

****

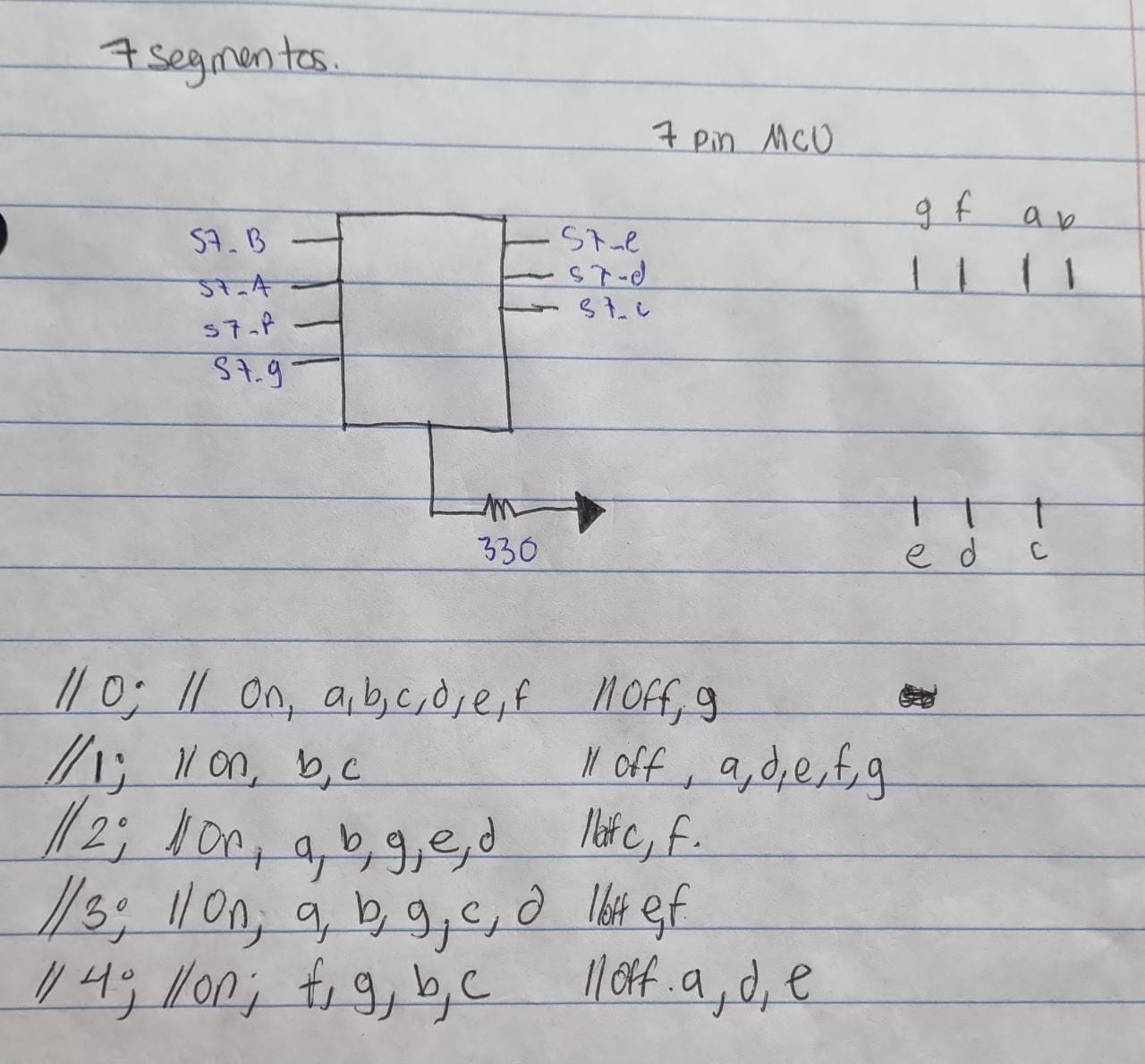
**El circuito implementado en el ESP es el siguiente.**

****

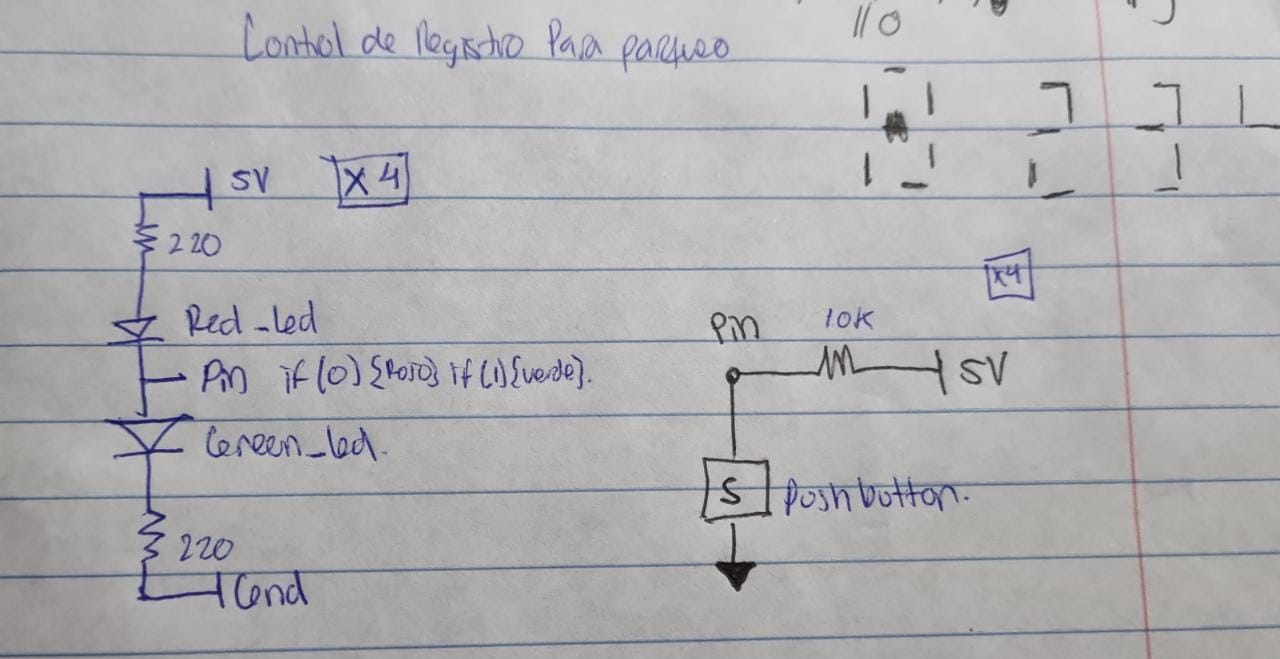
**Datos y Port Management TIVAC**

****

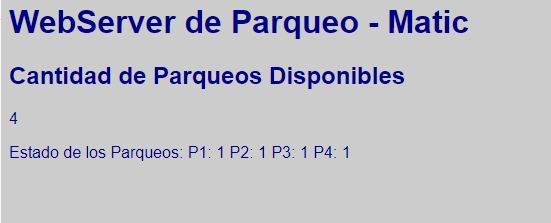
**7 segmentos Pin Map && Number map**

****

**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();

}