UVG UNIVERSIDAD DEL VALLE DE GUATEMALA

UNIVERSIDAD DEL VALLE DE GUATEMALA

Electrónica digital 2

Sección 30

Ing. Pablo Mazariegos

Proyecto # 4

Control de Parqueos (Parqueo-Matic)

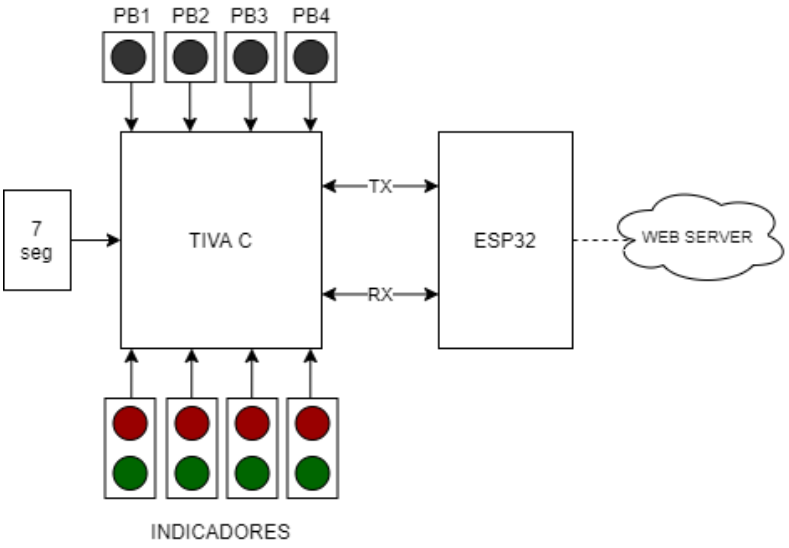
Katharine Senn Salazar - 18012

Guatemala, 28 de mayo de 2021

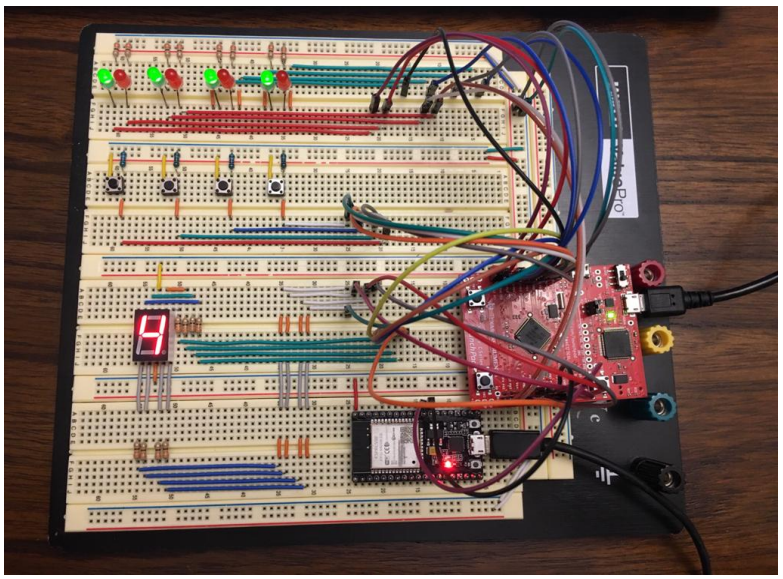
Link Github:
https://github.com/sen18012/Labs_Digital_2/tree/main/Proyecto_4

Link Youtube:
https://youtu.be/p4eNVZlBHQg

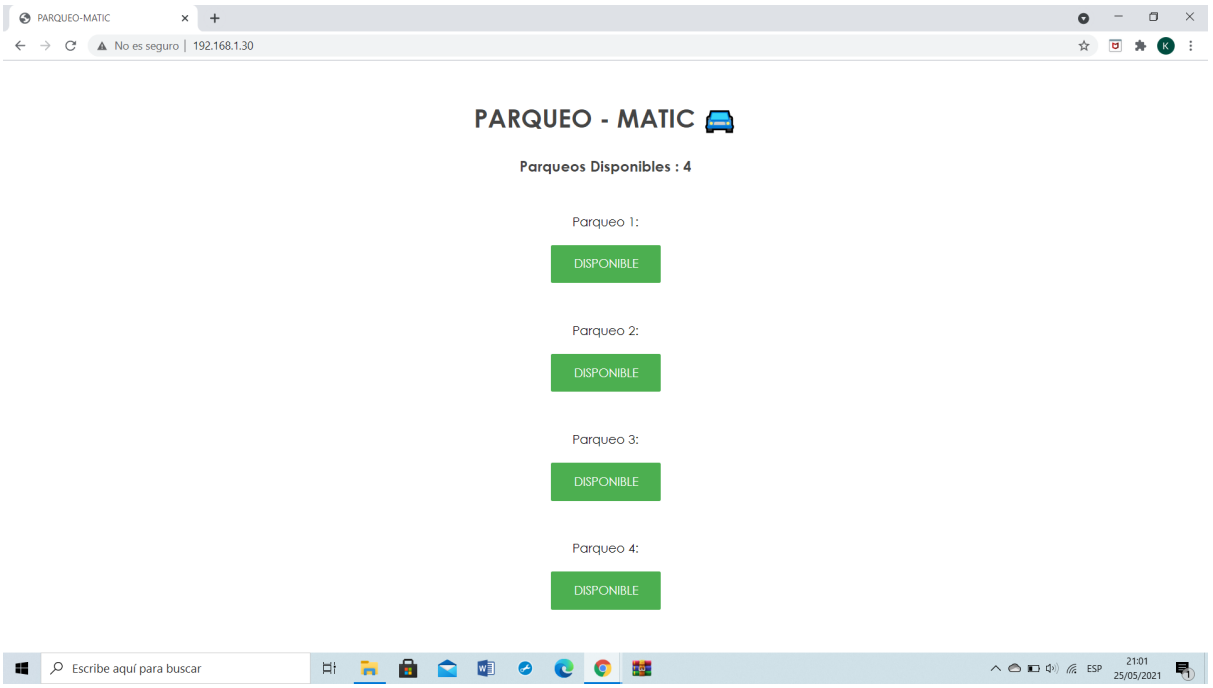
Diagrama:



Circuito Físico:



Interfaz Web Server:



Explicación del Proyecto:

El proyecto consiste en realizar un prototipo funcional de un sistema de control de parqueo por medio de una Tiva C y un ESP32.
Este prototipo será conformado por 4 pushbuttons que actúan como sensores para saber si cada parqueo se encuentra disponible u ocupado, Leds de colores rojo y verde los cuales indicarán el estado actual del parqueo y un display 7 segmentos que indicará la cantidad de parqueos disponibles.
De la misma manera, por medio de comunicación UART la Tiva C se comunica con el ESP32 el cual por medio de un Web Server nos permite monitorear en tiempo real el estado del estacionamiento.

Datos:

```
//*************************************
// VARIABLES
//*************************************
uint32_t parqueo = 0;     //Valor para el Display

uint32_t P1 = 0;          //Banderas para cada parqueo
uint32_t P2 = 0;
uint32_t P3 = 0;
uint32_t P4 = 0;

char val = '0';          //Valor para enviar
```

Código Tiva C:

Configuración Inicial

```
    //CONGIF DE FRECUENCIA (40 MHz)
    SysCtlClockSet(SYSCTL_SYSDIV_5 | SYSCTL_USE_OSC | SYSCTL_OSC_MAIN| SYSCTL_XTAL_16MHZ);  //FREQ - 40MHz

    //CONFIG HABILITAR PUERTOS
        SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);
        SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);
        SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOC);
        SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);
        SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOE);
        SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);

    while (!SysCtlPeripheralReady (SYSCTL_PERIPH_GPIOA)){
    }
    while (!SysCtlPeripheralReady (SYSCTL_PERIPH_GPIOB)){
        }
    while (!SysCtlPeripheralReady (SYSCTL_PERIPH_GPIOC)){
            }
    while (!SysCtlPeripheralReady (SYSCTL_PERIPH_GPIOD)){
        }
    while (!SysCtlPeripheralReady (SYSCTL_PERIPH_GPIOE)){
        }
    while (!SysCtlPeripheralReady (SYSCTL_PERIPH_GPIOF)){
        }


    //IO CONFIG
    GPIOPinTypeGPIOOutput(GPIO_PORTA_BASE, GPIO_PIN_2 | GPIO_PIN_3 | GPIO_PIN_6 | GPIO_PIN_7); //Indicadores Parqueo 1 y 2
    GPIOPinTypeGPIOOutput(GPIO_PORTE_BASE, GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_3 | GPIO_PIN_4); //Indicadores Parqueos 3 y 4
    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 | GPIO_PIN_7); //Indicador 7seg

    GPIOPinTypeGPIOInput(GPIO_PORTD_BASE, GPIO_PIN_0| GPIO_PIN_1| GPIO_PIN_2 | GPIO_PIN_3); //Sensores todos los parqueos

    GPIOPadConfigSet(GPIO_PORTD_BASE, GPIO_PIN_0, GPIO_STRENGTH_2MA, GPIO_PIN_TYPE_STD_WPD);
    GPIOPadConfigSet(GPIO_PORTD_BASE, GPIO_PIN_1, GPIO_STRENGTH_2MA, GPIO_PIN_TYPE_STD_WPD);
    GPIOPadConfigSet(GPIO_PORTD_BASE, GPIO_PIN_2, GPIO_STRENGTH_2MA, GPIO_PIN_TYPE_STD_WPD);
    GPIOPadConfigSet(GPIO_PORTD_BASE, GPIO_PIN_3, GPIO_STRENGTH_2MA, GPIO_PIN_TYPE_STD_WPD);

    InitUART();
```

Main Loop

```
//*************************************
// MAIN LOOP
//*************************************
    while(1){
        revisar();
        display_7seg();
        datos_uart();
        UARTCharPut(UART1_BASE, val);
        }
    }
```

Funciones

Función para revisar el estado actual de cada parqueo y contar los parqueos disponibles
(Para el parque 3 y 4 las condiciones de if se repiten de la misma manera que para los parqueos 1 y 2)

```
void revisar(void)
{
    parqueo = 0;
    P1 = 0;
    P2 = 0;
    P3 = 0;
    P4 = 0;

    if ( !GPIOPinRead(GPIO_PORTD_BASE, GPIO_PIN_0) )  { //Revisamo Parqueo 1

        GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_6 | GPIO_PIN_7, GPIO_PIN_7); //Parqueo Desocupado
        parqueo = parqueo+1;
        P1 = 1;
    }

    else if ( GPIOPinRead(GPIO_PORTD_BASE, GPIO_PIN_0) ) {
            GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_6 | GPIO_PIN_7, GPIO_PIN_6); //Parqueo Ocupado
            P1 = 0;
       }

    if ( !GPIOPinRead(GPIO_PORTD_BASE, GPIO_PIN_1) )  { //Revisamo Parqueo 2

        GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_2 | GPIO_PIN_3, GPIO_PIN_3); //Parqueo Desocupado
        parqueo = parqueo+1;
        P2 = 1;
    }

    else if ( GPIOPinRead(GPIO_PORTD_BASE, GPIO_PIN_1) ) {
            GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_2 | GPIO_PIN_3, GPIO_PIN_2); //Parqueo Ocupado
            P2 = 0;
    }
```

Función para encender leds del display 7 segmentos según cantidad de parqueos disponibles

```
void display_7seg(void)
{

    if (parqueo == 0){ //0 parqueos libres
        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 | GPIO_PIN_7, 0x3F);
    }
    else if (parqueo == 1) {//1 parqueo libre
        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 | GPIO_PIN_7,0x06);
       }
    else if (parqueo == 2) {//2 parqueos libres
            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 | GPIO_PIN_7,0x5B);
           }
    else if (parqueo == 3) {//3 parqueos libres
            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 | GPIO_PIN_7,0x4F);
           }
    else if (parqueo == 4) {//4 parqueos libres
            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 | GPIO_PIN_7,0x66);
           }


    }
```

Función para Inicializar la Comunicación UART

```
void InitUART(void){

    /*Enable the peripheral UART Module 1*/
    SysCtlPeripheralEnable(SYSCTL_PERIPH_UART1);

    while(!SysCtlPeripheralReady(SYSCTL_PERIPH_UART1)){
    }

    /*Enable the GPIO Port C*/
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOC);

    GPIOPinConfigure(GPIO_PC4_U1RX);
    GPIOPinConfigure(GPIO_PC5_U1TX);

    // Se habilitan las interrupciones Globales
    IntMasterEnable();

    /* Make the UART pins be peripheral controlled. */
    GPIOPinTypeUART(GPIO_PORTC_BASE, GPIO_PIN_4 | GPIO_PIN_5);

    UARTDisable(UART1_BASE);
    /* Sets the configuration of a UART. */
    UARTConfigSetExpClk(
            UART1_BASE, SysCtlClockGet(), 115200,
            (UART_CONFIG_WLEN_8 | UART_CONFIG_STOP_ONE | UART_CONFIG_PAR_NONE));

    IntEnable (INT_UART1);

    UARTIntEnable(UART1_BASE, UART_INT_RX | UART_INT_RT);
    UARTEnable (UART1_BASE);
}
```

Función para asignar el valor del dato que se enviará al ESP32 por USART según los parqueos que se encuentran ocupados (la función se repite para los 16 valores según las combinaciones posibles)

```
void datos_uart(void){
    if (P1 == 0 && P2 == 0 && P3 == 0 && P4 == 0){
        val = 'P';
    }
    else if (P1 == 0 && P2 == 0 && P3 == 0 && P4 == 1){
        val = 'O';
    }
    else if (P1 == 0 && P2 == 0 && P3 == 1 && P4 == 0){
        val = 'N';
    }
    else if (P1 == 0 && P2 == 0 && P3 == 1 && P4 == 1){
        val = 'M';
    }
    else if (P1 == 0 && P2 == 1 && P3 == 0 && P4 == 0){
        val = 'L';
    }
    else if (P1 == 0 && P2 == 1 && P3 == 0 && P4 == 1){
        val = 'K';
    }
    else if (P1 == 0 && P2 == 1 && P3 == 1 && P4 == 0){
        val = 'J';
    }
    else if (P1 == 0 && P2 == 1 && P3 == 1 && P4 == 1){
        val = 'I';
    }
    else if (P1 == 1 && P2 == 0 && P3 == 0 && P4 == 0){
        val = 'H';
    }
    else if (P1 == 1 && P2 == 0 && P3 == 0 && P4 == 1){
        val = 'G';
    }
    else if (P1 == 1 && P2 == 0 && P3 == 1 && P4 == 0){
        val = 'F';
    }
    else if (P1 == 1 && P2 == 0 && P3 == 1 && P4 == 1){
        val = 'E';
    }
```

Código ESP32:

Configuración Inicial

```
// Configuración
//*************************************************************************************************
void setup() {
  Serial.begin(115200);
  Serial.println("Try Connecting to ");
  Serial.println(ssid);

  Serial2.begin(115200, SERIAL_8N1, 16, 17); //Config Serial 2 para recibir datos


  // 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);
}
```

Main Loop

```
/
//*************************************************************************************************
// loop principal
//*************************************************************************************************
void loop() {

  server.handleClient();

  if (Serial2.available()>0){
    //read the incoming byte:
    parqueos = Serial2.read();
    }
```

UVG UNIVERSIDAD DEL VALLE DE GUATEMALA

Web Server (HTML)

```
//************************************************************************************************
// Procesador de HTML
//************************************************************************************************
String SendHTML(uint8_t parqueos) {
  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>PARQUEO-MATIC</title>\n";
  ptr += "<style>html { font-family: Century Gothic; display: inline-block; margin: 0px 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 += ".button {display: block;width: 80px;background-color: #3498db;border: none;color: white;padding: 15px 30px;text-decoration: none;display: inline-block;
  ptr += ".button1 {background-color: #4CAF50;}\n"; /* Green */
  ptr += ".button2 {background-color: #DC143C;}\n";/* Red */
  ptr += "</style>\n";
  ptr += "</head>\n";
  ptr += "<body>\n";
  ptr += "<h1>PARQUEO - MATIC &#128664</h1>\n";
  //ptr += "<h3>Control de Parqueos</h3>\n";

    if (parqueos == 65)
  {
    ptr += "<h3>Parqueos Disponibles : 4</h3>\n";
    ptr += "<p>Parqueo 1:</p><a class=\"button button1\">DISPONIBLE</a>\n";
    ptr += "<p>Parqueo 2:</p><a class=\"button button1\">DISPONIBLE</a>\n";
    ptr += "<p>Parqueo 3:</p><a class=\"button button1\">DISPONIBLE</a>\n";
    ptr += "<p>Parqueo 4:</p><a class=\"button button1\">DISPONIBLE</a>\n";
  }
```

Refresh de Web Server Automático

```
// Automatic Refresh 500ms
ptr += "<script>\n";
ptr += "<!--\n";
ptr += "function timedRefresh(timeoutPeriod) {\n";
ptr += "\tsetTimeout(\"location.reload(true);\",timeoutPeriod);\n";
ptr += "}\n";
ptr += "\n";
ptr += "window.onload = timedRefresh(500);\n";
ptr += "\n";
ptr += "//   -->\n";
ptr += "</script>\n";
ptr += "</body>\n";
ptr += "</html>\n";

return ptr;
```