Pablo Javier Peña Echeverría – 21210

Electrónica Digital 2

Proyecto 3: Control de parqueos

Código TIVA C:

De primero se inicializa el reloj a 40Mhz y luego la comunicación UART mediante la función InitConsole. Luego se inician los puertos que se usaron para controlar los leds.

```
void InitConsole(void){
    // se inicia el puerto UART 7
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOE);

    GPIOPinConfigure(GPIO_PE0_U7RX);
    GPIOPinConfigure(GPIO_PE1_U7TX);

    SysCtlPeripheralEnable(SYSCTL_PERIPH_UART7);

    GPIOPinTypeUART(GPIO_PORTE_BASE, GPIO_PIN_0 | GPIO_PIN_1);

    UARTConfigSetExpClk(UART7_BASE, SysCtlClockGet(),115200, (UART_CONFIG_WLEN_8 | UART_CONFIG_STOP_ONE | UART_CONFIG_PAR_NONE));
}

int main(void)
{
    SysCtlClockSet(SYSCTL_SYSDIV_5 | SYSCTL_USE_PLL | SYSCTL_XTAL_16MHZ| SYSCTL_OSC_MAIN);//se inicial el reloj a 40Mhz

    InitConsole();

    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);// se inicializan los puertos que se usaran para controlar los leds
    GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_6 | GPIO_PIN_7);

    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOC);
    GPIOPinTypeGPIOOutput(GPIO_PORTC_BASE, GPIO_PIN_4 | GPIO_PIN_5 | GPIO_PIN_6 | GPIO_PIN_7);

    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);
    GPIOPinTypeGPIOOutput(GPIO_PORTA_BASE, GPIO_PIN_2 | GPIO_PIN_3);
```

Luego se inicializan 4 puertos ADC que se usan para recibir las lecturas de las fotorresistencias, se les asigna una secuencia diferente y se activan las interrupciones correspondientes.

```
    SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);

    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);
    GPIOPinTypeADC(GPIO_PORTD_BASE, GPIO_PIN_0 | GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_3);//se inicializan los puertos que se ADC para leer las LDR

    ADCSequenceConfigure(ADC0_BASE, 3, ADC_TRIGGER_PROCESSOR, 0);
    ADCSequenceConfigure(ADC0_BASE, 2, ADC_TRIGGER_PROCESSOR, 0);
    ADCSequenceConfigure(ADC0_BASE, 1, ADC_TRIGGER_PROCESSOR, 0);
    ADCSequenceConfigure(ADC0_BASE, 0, ADC_TRIGGER_PROCESSOR, 0);

    ADCSequenceStepConfigure(ADC0_BASE, 3, 0, ADC_CTL_CH5|ADC_CTL_IE|ADC_CTL_END);
    ADCSequenceStepConfigure(ADC0_BASE, 2, 0, ADC_CTL_CH4|ADC_CTL_IE|ADC_CTL_END);
    ADCSequenceStepConfigure(ADC0_BASE, 1, 0, ADC_CTL_CH6|ADC_CTL_IE|ADC_CTL_END);
    ADCSequenceStepConfigure(ADC0_BASE, 0, 0, ADC_CTL_CH7|ADC_CTL_IE|ADC_CTL_END);

    ADCSequenceEnable(ADC0_BASE, 3);
    ADCSequenceEnable(ADC0_BASE, 2);
    ADCSequenceEnable(ADC0_BASE, 1);
    ADCSequenceEnable(ADC0_BASE, 0);

    ADCIntClear(ADC0_BASE, 3);
    ADCIntClear(ADC0_BASE, 2);
    ADCIntClear(ADC0_BASE, 1);
    ADCIntClear(ADC0_BASE, 0);
```

Dentro del loop principal de primero se activan las secuencias para que lean su fotorresistencia correspondiente y se espera hasta que haya alguna lectura. Cuando hubo una lectura se borran las anderas de interrupción y se obtiene el valor leído por los ADC el cual va de 0 a 4096.

```
84    while(1){
85        ADCProcessorTrigger(ADC0_BASE, 3);// se activa la lectura de los ADCs
86        ADCProcessorTrigger(ADC0_BASE, 2);
87        ADCProcessorTrigger(ADC0_BASE, 1);
88        ADCProcessorTrigger(ADC0_BASE, 0);
89
90        while(!ADCIntStatus(ADC0_BASE, 3, false) || !ADCIntStatus(ADC0_BASE, 2, false) || !ADCIntStatus(ADC0_BASE, 1, false) || !ADCIntStatus(ADC0_BASE, 0, false)){
91
92        }
93
94        ADCIntClear(ADC0_BASE, 3); // se limpian las banderas de interrupcion
95        ADCIntClear(ADC0_BASE, 2);
96        ADCIntClear(ADC0_BASE, 1);
97        ADCIntClear(ADC0_BASE, 0);
98        ADCSequenceDataGet(ADC0_BASE, 3, adc0); //se obtiene la lectura de los ADC
99        ADCSequenceDataGet(ADC0_BASE, 2, adc1);
100       ADCSequenceDataGet(ADC0_BASE, 1, adc2);
101       ADCSequenceDataGet(ADC0_BASE, 0, adc3);
```

A partir de las lecturas recibidas se determina cuál de los dos leds asignados a cada fotorresistencia deberá de prenderse. Luego de prenderse se apaga casi de forma inmediata para así también reiniciar las leds y que no pasara el caso en que las dos leds estuvieran prendidas al mismo tiempo.

```
103       if(adc0[0] < 500){
104           GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_6 | GPIO_PIN_7, 64);//se determina si hay un objeto obstruyendo la luz y se prender la luz respectiva
105           parking0 = '0';
106       }else{
107           GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_6 | GPIO_PIN_7, 128);
108           parking0 = '1';
109       }
110
111       GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_6 | GPIO_PIN_7, 0);
112
113       if(adc1[0] < 500){
114           GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_6 | GPIO_PIN_7, 64);
115           parking1 = '0';
116       }else{
117           GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_6 | GPIO_PIN_7, 128);
118           parking1 = '1';
119       }
120
121       GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_6 | GPIO_PIN_7, 0);
122
123       if(adc2[0] < 500){
124           GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_4 | GPIO_PIN_5, 32);
125           parking2 = '0';
126       }else{
127           GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_4 | GPIO_PIN_5, 16);
128           parking2 = '1';
129       }
130
131       GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_4 | GPIO_PIN_5, 0);
132
133       if(adc3[0] < 500){
134           GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_2 | GPIO_PIN_3, 4);
135           parking3 = '0';
136       }else{
137           GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_2 | GPIO_PIN_3, 8);
138           parking3 = '1';
139       }
140
141       GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_2 | GPIO_PIN_3, 0);
```

Finalmente, se mandan las lecturas de los leds al ESP32 aproximadamente cada 1.5 segundos.

```
142           count++;
143           if(count == 60000){//se envian las lecturas por uart al esp32 cada cierto tiempo
144               UARTCharPut(UART7_BASE, 'x');
145               UARTCharPut(UART7_BASE, parking3);
146               UARTCharPut(UART7_BASE, parking2);
147               UARTCharPut(UART7_BASE, parking0);
148               UARTCharPut(UART7_BASE, parking1);
149               UARTCharPut(UART7_BASE, 10);
150               UARTCharPut(UART7_BASE, 13);
151               count = 0;
152           }
```

Código ESP32:

Lo más importante es indicar el nombre de la red y la contraseña de la red, en este caso tiene la del internet de mi casa.

```
#include <WiFi.h>
#include <WebServer.h>

/* Put your SSID & Password */
const char* ssid = "Comunidad";  // Enter SSID here
const char* password = "Comunidad3312";  //Enter Password here
```

Luego se inicia el webserver y se crean las variables que se usaran para almacenar las lecturas y para controlar el display de 7 segmentos de forma mas sencilla.

```
WebServer server(80);

char park0[1];
char park1[1];
char park2[1];
char park3[1];

int place0 = 0;
int place1 = 0;
int place2 = 0;
int place3 = 0;

int a = 25;
int b = 23;
int c = 26;
int d = 27;
int e = 33;
int f = 14;
int g = 22;

int cont = 0;
```

Luego se inicia la comunicación serial y los pines a utilizar para controlar el display de 7 segmentos.

```
35  void setup() {
36    // put your setup code here, to run once:
37    Serial.begin(9600);
38    Serial2.begin(115200, SERIAL_8N1, 16, 17);
39    while(!Serial);
40
41    pinMode(a, OUTPUT);
42    pinMode(b, OUTPUT);
43    pinMode(c, OUTPUT);
44    pinMode(d, OUTPUT);
45    pinMode(e, OUTPUT);
46    pinMode(f, OUTPUT);
47    pinMode(g, OUTPUT);
```

Después se intenta establecer conexión con la red indicada al principio del código y se indican las funciones con las que el server va a estar trabajando.

```
52    Serial.println("Try connecting to");
53    Serial.println(ssid);
54
55    WiFi.begin(ssid, password);
56    while(WiFi.status()!= WL_CONNECTED){
57      delay(1000);
58      Serial.println(".");
59    }
60    Serial.println("");
61    Serial.println("WiFi conected success");
62    Serial.print("Got IP: ");
63    Serial.println(WiFi.localIP());
64
65    server.on("/", handle_actualizar);
66    server.on("/actualizar", handle_actualizar);
67    server.onNotFound(handle_NotFound);
68
69    server.begin();
70    Serial.println("HTTP server started");
71  }
72  void handle_NotFound(){
73    server.send(404, "text/plain", "Not found");
74  }
75
76  void handle_actualizar(){
77    server.send(200, "text/html", SendHTML(place0, place1, place2, place3));
78  }
79
```

Luego dentro del loop se almacenan los valores del estado de los parques cada vez que la comunicación UART esta disponible.

```
80  void loop() {
81    server.handleClient();
82    if(Serial2.available()>0){
83      if(Serial2.read() == 'x'){
84        Serial2.readBytesUntil(10, park0, 1);
85        Serial2.readBytesUntil(10, park1, 1);
86        Serial2.readBytesUntil(10, park2, 1);
87        Serial2.readBytesUntil(10, park3, 1);
88        if(park0[0] == '0'){
89          place0 = 0;
90        }else{
91          place0 = 1;
92        }
93        if(park1[0] == '0'){
94          place1 = 0;
95        }else{
96          place1 = 1;
97        }
98        if(park2[0] == '0'){
99          place2 = 0;
100       }else{
101         place2 = 1;
102       }
103       if(park3[0] == '0'){
104         place3 = 0;
105       }else{
106         place3 = 1;
107       }
```

A partir de estas lecturas se determina el valor que se mostrara en el display de 7 segmentos.

```
112         cont = place0 + place1 + place2 + place3;
113         if (cont == 0){
114           digitalWrite(a, LOW);
115           digitalWrite(b, LOW);
116           digitalWrite(c, LOW);
117           digitalWrite(d, LOW);
118           digitalWrite(e, LOW);
119           digitalWrite(f, LOW);
120           digitalWrite(g, HIGH);
121         }else if(cont == 1){
122           digitalWrite(a, HIGH);
123           digitalWrite(b, LOW);
124           digitalWrite(c, LOW);
125           digitalWrite(d, HIGH);
126           digitalWrite(e, HIGH);
127           digitalWrite(f, HIGH);
128           digitalWrite(g, HIGH);
129         }else if(cont == 2){
130           digitalWrite(a, LOW);
131           digitalWrite(b, LOW);
132           digitalWrite(c, HIGH);
133           digitalWrite(d, LOW);
134           digitalWrite(e, LOW);
135           digitalWrite(f, HIGH);
136           digitalWrite(g, LOW);
137         }else if(cont == 3){
138           digitalWrite(a, LOW);
139           digitalWrite(b, LOW);
140           digitalWrite(c, LOW);
141           digitalWrite(d, LOW);
142           digitalWrite(e, HIGH);
143           digitalWrite(f, HIGH);
144           digitalWrite(g, LOW);
145         }else if(cont == 4){
146           digitalWrite(a, HIGH);
147           digitalWrite(b, LOW);
148           digitalWrite(c, LOW);
149           digitalWrite(d, HIGH);
150           digitalWrite(e, HIGH);
151           digitalWrite(f, LOW);
152           digitalWrite(g, LOW);
153         }
```

Finalmente, define la función HTML donde se carga la página, y a partir de los valores recibidos se determina el estado del parque que se mostrara en la página.

```
String SendHTML(int a, int b, int c, int d){
  String ptr = "<!DOCTYPE html> <html>\n";
  ptr+="<head> <meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0, user-scalable=no\">\n";
  ptr+="<!-- Bootstrap CSS --><link rel=\"stylesheet\" href=\"https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css\"integrity=\"sha384-9aIt2nRpC12Uk9gS9baDl411NQApF
  ptr+="<title>Proyecto #3</title><script>function autoRefresh() {window.location = window.location.href;}setInterval('autoRefresh()', 2000);</script><style>html { font-family: Helveti
  ptr+="body{margin-top: 50px;} h1 {color: #444444;margin: 50px auto 30px;} h3 {color: #444444;margin-bottom: 50px;}.button {display: block;width: 160px;background-color: #3498db;borde
  ptr+=".button-on:active {background-color: #2980b9;}.button-off {background-color: #34495e;}.button-off:active {background-color: #2c3e50;}p {font-size: 14px;color: #888;margin-botto
  ptr+="<body><h1>Parking control &#128680 </h1><br><a class=\"button button-off\" href=\"actualizar\">Update</a><br><table class=\"table\"; border-radius: 4px><tbody><tr><th scope=\"r
  if(a == 0){
    ptr+="<td class=\"table-danger\">Taken &#128721</td>\n";
  }else{
    ptr+="<td class=\"table-primary\">Available &#128665</td>\n";
  }
  if(b == 0){
    ptr+="<td class=\"table-danger\">Taken &#128721</td>\n";
  }else{
    ptr+="<td class=\"table-primary\">Available &#128665</td>\n";
  }
  if(c == 0){
    ptr+="<td class=\"table-danger\">Taken &#128721</td>\n";
  }else{
    ptr+="<td class=\"table-primary\">Available &#128665</td>\n";
  }
  if(d == 0){
    ptr+="<td class=\"table-danger\">Taken &#128721</td>\n";
  }else{
    ptr+="<td class=\"table-primary\">Available &#128665</td>\n";
  }
  ptr+="</tr></tbody></table><!-- Optional JavaScript --><!-- jQuery first, then Popper.js, then Bootstrap JS --><script src=\"https://code.jquery.com/jquery-3.5.1.slim.min.js\"\n";
  ptr+="integrity=\"sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj\"crossorigin=\"anonymous\"></script><script src=\"https://cdn.jsdelivr.net/npm/popper.js@1.1
  ptr+="<script src=\"https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js\"integrity=\"sha384-OgVRvuATP1z7JjHLkuOU7Xw704+h835Lr+6QL9UvYjZE3Ipu6Tp75j7Bh/kR0JKI\"cross
  return ptr;
}
```