University of Guadalajara
University Center of Exact Sciences and Engineering

Practice report

Increase numbers from 0 to 9 with a push button using the ESP32

Presented by:
Zahara Nathalia Ibarra Casillas
Kevin Eduardo López Martínez
Brian Michel Rubio Martínez

I9398 - D01
Embedded Systems Programming Troubleshooting Seminar
Made in LaTeX

Índice

1. Introduction **2**
1.1. Objective 2

2. Development **2**
2.1. Theoretical framework 2
2.2. Methodology/Procedure 2

3. Results **4**

4. Conclusion **5**

1. Introduction

This activity was made inside of the class schedule, which is why we are going to explain only topics that we studied at the time.

1.1. Objective

(i) Use a push button to switch to the next number consecutively.

(ii) Show the numbers on the 7-segment display.

2. Development

First, we will see some fundamental concepts to understand why to use the ESP32 microcontroller and the development of our code.

2.1. Theoretical framework

(i) What is the purpose of the digitalWrite function in Arduino?
The digitalWrite() function, normally used in the loop() function, is used to write a value (or set a state) to a digital pin. The possible values or states are HIGH or LOW.

(ii) What is the purpose of the digitalRead function in Arduino?
The digitalRead() function, normally used in the loop() function, is used to read a value (or set a state) of a digital pin. The possible values or states are HIGH or LOW. The value read can be stored in a variable or dynamically checked in a condition.

2.2. Methodology/Procedure

The code defines the pins required for system operation. Pin 15 is assigned to connect a push button, which will be used to increment the counter each time it is pressed. In addition, an array is declared with the pins that control the segments of the 7-segment display. Each index of this array represents a specific segment of the display, allowing its activation or deactivation independently. Auxiliary variables such as "count", which stores the current number shown on the display, and "pbtn" and "anpbtn", which are used to read and compare the button status, are also defined.

In the setup() function, the necessary components are initialized. The button pin is configured as an input with internal pull-up resistor, which means that, by default, its state will be HIGH and will change to LOW when pressed. Then, the pins associated with the segments of the 7-segment display are configured as outputs and are initially turned off to ensure that they are not left on unexpectedly.

The function "displayNumber(int num)" is responsible for turning on the appropriate segments of the display to represent a number from 0 to 9. To do this, it uses an array "segmentMap[10][7]", where each row corresponds to a number and each column represents a specific segment of the display. A value of 1 in the matrix indicates that the segment should be turned on, while a 0 indicates that it should remain off. Then, a loop goes through the array and sets the segment

pins according to the number to be displayed.

In the loop() cycle, the detection of the button and the change of the number on the display is handled. First, the current state of the button is read and stored in the variable pbtn. Then, it is checked if the button was pressed when detecting a transition from HIGH to LOW. If this condition is met, the counter count is incremented by one and, when it reaches 9, it returns to 0 thanks to the operation count = (count + 1) /10. Next, "displayNumber(count)" is called to update the display with the new value and the number is printed on the serial monitor. Finally, the variable anpbtn is updated with the current state of the button, allowing future state changes to be detected and preventing the counter from incrementing multiple times with a single long press.

The code is attached below:

```
// Practice 2 Increase numbers from 0 to 9 with a push button using the ESP32
// Team Septiembre(9)
// 04/02/2025

#define PBTN_PIN 15

/* Pines del display de 7 segmentos */
                    //| a | b | c | d | e | f | g |//
const int segments[] = {23, 22, 21, 19, 18, 5, 4};

/* Variables */
int count = 0;
bool pbtn = false;
bool anpbtn = false;

/* Configuraci n de los pines */
void setup() {
  Serial.begin(115200);
  pinMode(PBTN_PIN, INPUT_PULLUP);
  for (int i = 0; i < 7; i++) {
    pinMode(segments[i], OUTPUT);
    digitalWrite(segments[i], LOW);
    /* Asegurarse de que todos los segmentos est n apagados al inicio */
  }
}

/* Funci n para mostrar el n mero en el display */
void displayNumber(int num) {
  const bool segmentMap[10][7] = {
  /* a, b, c, d, e, f, g */
    {1, 1, 1, 1, 1, 1, 0}, // 0
    {0, 1, 1, 0, 0, 0, 0}, // 1
    {1, 1, 0, 1, 1, 0, 1}, // 2
```

```
    {1, 1, 1, 1, 0, 0, 1}, // 3
    {0, 1, 1, 0, 0, 1, 1}, // 4
    {1, 0, 1, 1, 0, 1, 1}, // 5
    {1, 0, 1, 1, 1, 1, 1}, // 6
    {1, 1, 1, 0, 0, 0, 0}, // 7
    {1, 1, 1, 1, 1, 1, 1}, // 8
    {1, 1, 1, 1, 0, 1, 1}  // 9
  };
  /* Recorre el mapa para mostrar los n meros */
  for (int i = 0; i < 7; i++) {
    digitalWrite(segments[i], segmentMap[num][i]);
  }
}

void loop() {
  pbtn = digitalRead(PBTN_PIN);
  /* Activa la funcion cada que se presione el pinshi boton */
  if (pbtn == LOW && anpbtn == HIGH) {
    count = (count + 1) % 10;
    displayNumber(count);
    Serial.println(count);
  }
  /* Actualiza el estado del pinshi boton */
  anpbtn = pbtn;
}
```

3. Results

Below, we add some pictures of the microcontroller with the code already loaded and the circuit assembled.

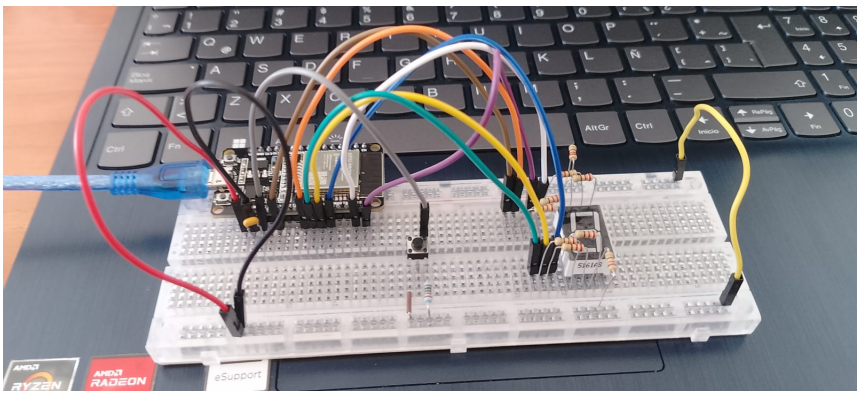

Figura 1: Photo taken of the mounted circuit.

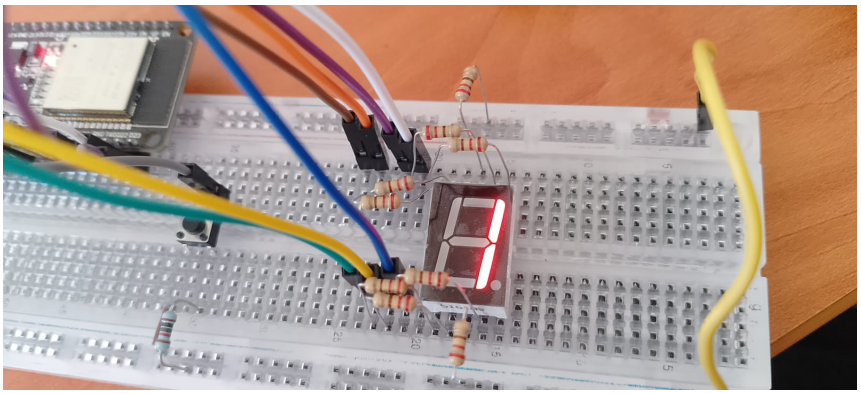

Figura 2: Photo taken of the mounted circuit in operation.

4. Conclusion

To read the state of the push button it was necessary to use the digitalRead function and to change the state of the pins (or LEDS) of the 7segment display it was necessary to use the digitalWrite function.

Referencias

[1] M. Bellan, (2019, July 04). *Salidas digitales con Arduino*[Online]. Available: https://www.programoergosum.es/tutoriales/salidas-digitales-con-arduino/

[2] M. Bellan, (2019, July 04). *Entradas digitales con Arduino*[Online]. Available: https://www.programoergosum.es/tutoriales/entradas-digitales-con-arduino/