Universidad de Guadalajara

Centro Universitario de Ciencias Exactas e Ingenierías

Seminario de problemas de programación de sistemas embebidos

I19893 – D01

Activity 2

Design and Implementation of a Seven-Segment Display Controller Using ESP32

Prof. Martínez Sandoval, Daniel Giovanni

Students:

Cabrera Gómez, Sergio Nicolas

Code: 222328115

Davalos Magaña, Jonathan Israrel

Code: 219339068

Martinez Flores, Cassandra Frine

Code: 218456125

Design and Implementation of a Seven-Segment Display Controller Using ESP32

Abstract

This report presents the implementation of a seven-segment display control system using an ESP32 microcontroller. The system consists of a seven-segment common-cathode display, eight 220-ohm resistors, a push-button, a 10k-ohm pull-down resistor, and a 0.1µF capacitor rated at 50V. The program increments and displays a digit (0-9) upon each press of the push-button. The design methodology, circuit configuration, and software implementation are discussed.

Keywords

ESP32, seven-segment display, digital electronics, push-button, embedded systems, microcontroller, development board.

Introduction

Seven-segment displays are widely used in embedded systems for numerical representation. The objective of this practice was to design a simple numerical counter utilizing an ESP32 microcontroller and a seven-segment display. The system registers a push-button press and increments the displayed number cyclically from 0 to 9.

Theoretical Framework

The ESP32 is a low-power, high-performance microcontroller developed by Espressif Systems, widely used in embedded applications due to its integrated Wi-Fi and Bluetooth capabilities. This development board features a dual-core Tensilica LX6 processor, making it suitable for real-time processing. Seven-segment displays operate by selectively activating individual segments to represent numerical values. Each segment corresponds to a specific bit in a predefined mapping scheme, allowing the representation of digits 0 through 9.

Push-button switches are mechanical components used for user input, requiring debouncing mechanisms to avoid false triggers due to mechanical bouncing. In this design, a pull-down resistor and a capacitor mitigate these effects, ensuring reliable input detection.

Methodology

The methodology followed in this project consists of the following stages:

1. Component Selection and Circuit Design: The ESP32 development board was chosen due to its versatility, built-in GPIO capabilities, and compatibility with the Arduino framework. A common-cathode seven-segment display was selected, with current-limiting resistors to protect the circuit components.

2. Circuit Implementation: The push-button was connected to GPIO15 of the ESP32 with a 10kΩ pull-down resistor to ensure a stable low signal in the absence of user input. A 0.1μF capacitor was added to mitigate switch bouncing. The display segments were connected to seven GPIO pins through 220Ω resistors.

3. Software Development: The program was written using C++ in the Arduino IDE. It initializes GPIOs, reads button states, and updates the display based on input conditions. A lookup table was employed for efficient segment control.

4. Testing and Validation: The circuit was tested in various conditions to ensure correct operation. Debugging was performed using serial communication for real-time feedback.

Materials and methods

Hardware Components

The following components were used in the implementation:

- ESP32 development board (dual-core microcontroller with Wi-Fi and Bluetooth capabilities)
- Seven-segment common-cathode display
- 8 x 220Ω resistors (current limiting)
- 1 x 10kΩ pull-down resistor
- 1 x Push-button
- 1 x 0.1μF capacitor (50V)

Circuit Desing

The seven-segment display is interfaced with the ESP32 through seven GPIO pins, each connected to a segment through a 220Ω resistor to limit current. The push-button is wired to GPIO15, with a 10kΩ pull-down resistor and a 0.1μF capacitor for debounce.

Software Implementation

The ESP32 was programmed using the Arduino framework. The code initializes the GPIO pins, detects button presses, and updates the display accordingly. The segment mapping is stored in an array, and the display is refreshed based on the detected input. The implemented code is as follows:

```
#define PBTN_PIN 15
const int segments[] = {23, 22, 21, 19, 18, 5, 4};
int count = 0; bool pbtn = false, anpbtn = false;
void setup() {
  Serial.begin(115200);
  pinMode(PBTN_PIN, INPUT_PULLUP);
  for (int seg : segments) pinMode(seg, OUTPUT);
}
void displayNumber(int num) {
  const uint8_t segmentMap[10] = {0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 
0x6F};
  for (int i = 0; i < 7; i++) digitalWrite(segments[i], (segmentMap[num] >> i) & 0x01);
}
void loop() {
  if ((pbtn = digitalRead(PBTN_PIN)) == LOW && anpbtn == HIGH) displayNumber(count = (count 
+ 1) % 10), Serial.println(count);
  anpbtn = pbtn;
}
```

Results and Discussion

Upon testing, the system correctly displayed numerical values from 0 to 9 on the seven-segment display. Each press of the push-button incremented the displayed number, confirming the correct implementation of button debouncing and segment control. The use of a pull-down resistor ensured stable button operation, preventing unintended activations.

The ESP32 proved to be an efficient development board for this application due to its extensive GPIO support and real-time processing capabilities. The debounce mechanism implemented using a capacitor provided reliable push-button readings, minimizing erroneous triggers.

Conclusion

The implementation successfully demonstrated the interfacing of a seven-segment display with an ESP32 development board. The approach utilized an efficient digital representation for segment control and a simple push-button interface for user input. This setup can serve as a foundation for more complex numerical display applications in embedded systems.
Future improvements could include implementing a software-based debounce algorithm, adding multiple displays for multi-digit representation, or integrating a real-time clock module for time-based applications.

References

- LME Editorial Staff, “ESP32 Pinout reference,” Last Minute Engineers, Nov. 23, 2023. https://lastminuteengineers.com/esp32-pinout-reference/
- “7-SEGMENT DISPLAY Datasheet(PDF) - TOS5161 - List of unclassifed manufacturers.” https://www.alldatasheet.com/category/index.jsp?sSearchword=7-SEGMENT%20DISPLAY
- C. Electricos, “ESP32 – Especificaciones y diseños,” Circuitos Eléctricos, Jun. 04, 2020. https://www.circuitos-electricos.com/esp32-especificaciones-y-disenos/
- R. Teja, “Getting Started with ESP32 | Introduction to ESP32,” ElectronicsHub, Apr. 05, 2024. https://www.electronicshub.org/getting-started-with-esp32/
- “Technical Documents | Espressif Systems.” https://www.espressif.com/en/support/documents/technical-documents