Universidad de Guadalajara

Centro Universitario de Ciencias Exactas e Ingenierías

Seminario de problemas de programación de sistemas embebidos

I19893 – D01

Activity 4

Time Unit Counter using ESP32 and Seven-Segment Displays

Prof. Martínez Sandoval, Daniel Giovanni

Students:

Cabrera Gómez, Sergio Nicolas

Code: 222328115

Davalos Magaña, Jonathan Israrel

Code: 219339068

Martinez Flores, Cassandra Frine

Code: 218456125

Time Unit Counter using ESP32 and Seven-Segment Displays

Abstract

This report presents the design and implementation of a time unit counter using an ESP32-WROOM-32 development kit, two dual seven-segment displays, 2N2222 transistors, and 3.9kΩ resistors. The system counts from 00:00 to 59:59, representing minutes and seconds, and is implemented using Arduino IDE. A multiplexing technique is employed to drive the displays efficiently.

Keywords

ESP32, seven-segment display, multiplexing, time counter, Arduino IDE.

Introduction

Seven-segment displays are widely used for numerical data representation. Due to the limited GPIO pins on microcontrollers, a multiplexing technique is required to control multiple digits. This project aims to develop a functional time unit counter using an ESP32, implementing time updates and display multiplexing.

Related work

Several previous works have explored the implementation of seven-segment display counters using different microcontrollers such as Arduino and PIC. ESP32-based implementations have gained popularity due to their advanced processing capabilities and integrated Wi-Fi and Bluetooth. Compared to other methods, this project emphasizes efficient multiplexing and minimal hardware components.

Theoretical Framework

Time counters are essential in various applications such as clocks, timers, and digital measurement devices. The ESP32 microcontroller is chosen due to its high processing speed and multiple I/O capabilities. Multiplexing allows multiple displays to be controlled with fewer pins by rapidly switching between digits.

Methodology

The project follows a structured methodology:

- Selection of components based on power requirements and pin availability.
- Design and simulation of the circuit.
- Implementation of multiplexing and timing control in software.
- Testing and debugging to ensure accurate counting behavior.

Materials and methods

Components used

- ESP32 development board (dual-core microcontroller with Wi-Fi and Bluetooth capabilities)
- Two dual 7-segment displays (common cathode)
- Four NPN transistors (2N2222)
- 3.9kΩ resistors
- Connecting wires and a breadboard

Circuit Desing

Each segment of the displays is connected to the ESP32 via current-limiting resistors. The transistors act as digit selectors, allowing only one digit to be active at a time. By rapidly switching between digits, persistence of vision ensures the appearance of a continuous display.

Software Implementation

The software is written in C++ using the Arduino IDE. The main functions include:

- setup(): Initializes the pins as outputs.
- loop(): Handles time updates and display multiplexing.
- Multiplexing logic: Sequentially enables each digit and displays the corresponding value.

Key Lines of Code Explained

- Line 7: Segment Mapping

```
const uint8_t segmentMap[10] = {0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F,
0x6F};
```

This array stores the binary representations of digits (0-9) for a common cathode seven-segment display. Each value corresponds to which segments should be turned on.

- Line 37: Time increment logic

```
if (currentTime - lastUpdateTime >= updateInterval) {
```

This condition ensures that one second has passed before updating the time. It prevents the counter from incrementing too quickly by using the millis() function.

- Line 51: Seconds display multiplexing

```
int digit = (seconds / (int)pow(10, i)) % 10;
```

This extracts the digit to be displayed by performing integer division and modulo operations. It isolates each decimal place for display.

- Line 54: Segment control for display

```
digitalWrite(segmentPins1[j], (segments >> j) & 0x01);
```

This line determines whether each segment should be turned on or off. The bitwise shift (segments >> j) moves the required bit into position, and the bitwise AND (& 0x01) extracts that bit. If the result is 1, the segment turns on; if 0, it turns off.

Results and Discussion

The system successfully counted from 00:00 to 59:59 with a stable display. The multiplexing approach allowed efficient use of GPIOs, reducing power consumption and ensuring legibility.

Conclusion

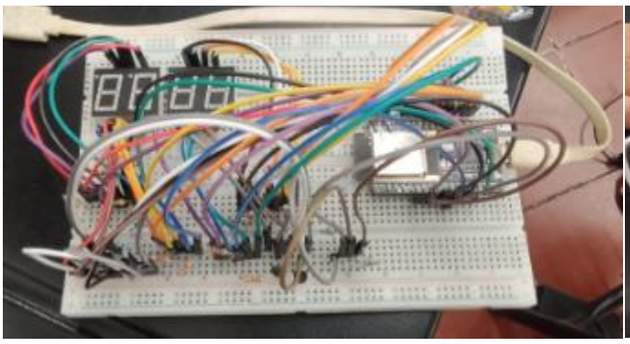
The ESP32-based counter demonstrated the feasibility of multiplexed seven-segment display control. Future improvements could include external RTC integration for precise timekeeping.

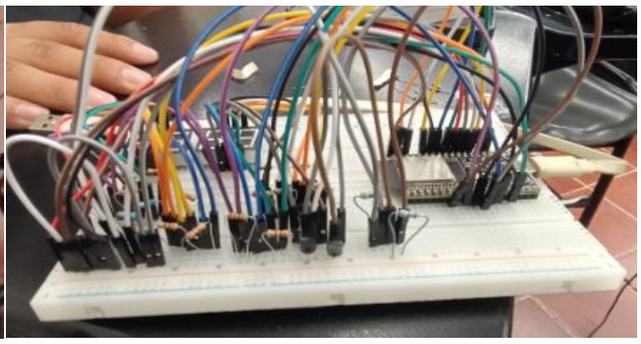
Appendices

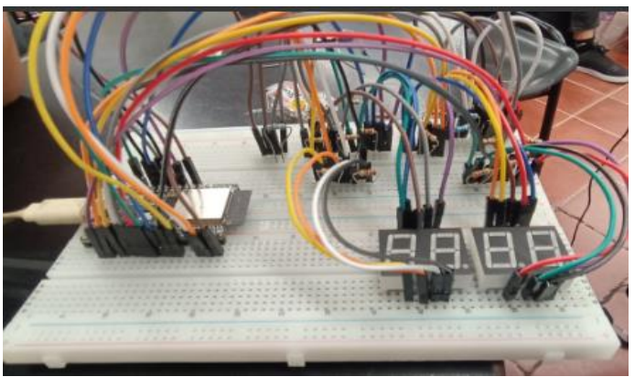
Appendix A: Source Code

```
const int segmentPins1[7] = {2, 4, 5, 12, 13, 14, 15}; // Display 1 (Segundos)
const int transistorPins1[2] = {16, 17}; // Display 1 (Segundos)

const int segmentPins2[7] = {18, 19, 21, 22, 23, 25, 26}; // Display 2 (Minutos)
const int transistorPins2[2] = {27, 32}; // Display 2 (Minutos)

const uint8_t segmentMap[10] = {0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x6F};

unsigned int seconds = 0; // Segundos a mostrar
unsigned int minutes = 0; // Minutos a mostrar
unsigned long lastUpdateTime = 0; // Última vez que se actualizó el tiempo
const int updateInterval = 50; // Intervalo de actualización en milisegundos (1 segundo)

void setup() {
  for (int i = 0; i < 7; i++) {
    pinMode(segmentPins1[i], OUTPUT);
    digitalWrite(segmentPins1[i], LOW);
  }
  for (int i = 0; i < 2; i++) {
    pinMode(transistorPins1[i], OUTPUT);
    digitalWrite(transistorPins1[i], LOW);
  }

  for (int i = 0; i < 7; i++) {
    pinMode(segmentPins2[i], OUTPUT);
    digitalWrite(segmentPins2[i], LOW);
  }
  for (int i = 0; i < 2; i++) {
    pinMode(transistorPins2[i], OUTPUT);
    digitalWrite(transistorPins2[i], LOW);
  }
}

void loop() {
  unsigned long currentTime = millis();

  if (currentTime - lastUpdateTime >= updateInterval) {
    lastUpdateTime = currentTime;
    seconds++;
    if (seconds > 59) { // Reiniciar a 0 cuando llegue a 59 segundos
      seconds = 0;
      minutes++;
      if (minutes > 59) { // Reiniciar a 0 cuando llegue a 59 minutos
        minutes = 0;
      }
    }
  }

  // Mostrar segundos
  for (int i = 0; i < 2; i++) {
    int digit = (seconds / (int)pow(10, i)) % 10;
    uint8_t segments = segmentMap[digit];
    for (int j = 0; j < 7; j++) {
      digitalWrite(segmentPins1[j], (segments >> j) & 0x01);
    }
    digitalWrite(transistorPins1[i], HIGH);
    delay(5);
    digitalWrite(transistorPins1[i], LOW);
  }

  // Mostrar minutos
  for (int i = 0; i < 2; i++) {
    int digit = (minutes / (int)pow(10, i)) % 10;
    uint8_t segments = segmentMap[digit];
    for (int j = 0; j < 7; j++) {
      digitalWrite(segmentPins2[j], (segments >> j) & 0x01);
    }
    digitalWrite(transistorPins2[i], HIGH);
    delay(5);
    digitalWrite(transistorPins2[i], LOW);
  }
}
```

Appendix B: Circuit







Note: Teacher, I have not been able to do the git For reasons of time, that and because I scheduled a good number of subjects, so for the moment I attach the code in the document, in the next report I attach the git, have a good day.
Thanks for reading this. 😊 💖

Учитель: Я не смог сделать что-то в git из-за нехватки времени, а также потому, что я запланировал большое количество предметов, поэтому на данный момент я прикрепляю код в документе, в следующем отчете я прикрепляю что-то git, хорошего дня,
Спасибо, что прочитали 😊 💖

References

- LME Editorial Staff, "ESP32 Pinout reference," Last Minute Engineers, Nov. 23, 2023. https://lastminuteengineers.com/esp32-pinout-reference/
- 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
- "7-SEGMENT DISPLAY Datasheet(PDF) - TOS5161 - List of unclassifed manufacturers." https://www.alldatasheet.com/category/index.jsp?sSearchword=7-SEGMENT%20DISPLAY
- alldatasheet.com, "2N2222 Datasheet, PDF," Alldatasheet. https://www.alldatasheet.com/view.jsp?Searchword=DATASHEET%202N2222&msclkid=24f8173ec06517028df470893feb6084
- "map() - Guía de Referencia de Arduino." https://www.arduino.cc/reference/es/language/functions/math/map/
- T. Mante V., "ESP32 WRoom DevKit V1 Datasheet," Manual and Engine Fix Library, Dec. 22, 2023. https://bylinoahblibguide.z14.web.core.windows.net/esp32-wroom-devkit-v1-datasheet.html