PRACTICE REPORT

Practice 5

Participants:

Lorena Villalobos Carrillo

Ximena Lucía Rodríguez Oliva

Brayam De Jesús De La Cerda Valdivia

Tuesday, February 25, 2025

Seminar on Embedded Programming Problems

(DO1 - I9893)

This document was prepared in LaTeX.

Contents

0.1 Abstract

In this practice, a brightness control of an LED was implemented using a potentiometer and an ESP32. The ADC converter was used to read the analog signal from the potentiometer, and the PWM module was used to adjust the LED's intensity. A weighted moving average filter was applied to smooth the ADC reading, and the value was converted into different scales, such as brightness (0-255) and angle (0-360). The results show the direct relationship between the potentiometer reading and the LED brightness.

0.2 Objectives

The main focus of this practice is to understand how analog-to-digital signal conversions are performed and how they can be used in practical applications. The potentiometer acts as an interface between the user and the system, allowing manual control of a variable (in this case, the brightness of an LED). Through the ESP32's ADC, the analog signal from the potentiometer is converted into a digital value, which is processed and converted into different scales for use in controlling the LED via PWM.

Additionally, the aim is to understand the importance of these conversions in embedded systems, where it is common to interact with analog sensors and control actuators using digital signals.

0.3 Introduction

The conversion between analog and digital signals is fundamental in electronics and embedded systems control. In this practice, the reading of a potentiometer is implemented using the ESP32's ADC, and the brightness of an LED is controlled via PWM. The relationship between the digital value read and the pulse width modulation (PWM) is analyzed to simulate an analog output. Additionally, a scale conversion to angles is performed, demonstrating applications in motor or servo control.

0.4 Theoretical Framework

This section presents the necessary concepts to fully understand the practice.

- **Digital Variables:** Also called discrete variables. They are characterized by having two distinct states and can therefore be called binary. These variables are easier to handle (in logic, they would be the values True (T) and False (F) or could be 1 or 0, respectively). An example of a digital signal is a doorbell switch, as this switch has two states: pressed and not pressed.
- **Analog Variables:** These are variables that can take an infinite number of values between two limits. Most real-world phenomena are signals of this type (sound, temperature, voice, video, etc.). An example of an analog electronic system is a loudspeaker, which is used to amplify sound so that it can be heard by a large audience.
- **Pulse Width Modulation (PWM):** Pulse width modulation (also known as PWM) of a signal or power source is a technique in which the duty cycle of a periodic signal (such as a sine or square wave) is modified, either to transmit information through a communication channel or to control the amount of power sent to a load.
- **ADC Converter:** ADC converters are responsible for converting an analog voltage level into a corresponding digital word. If the word is n bits, then the digital output signal will have 2^n different voltage levels that can be identified at the input. To ensure that the code indicates exactly the corresponding analog level, the number of bits would have to be infinite, so all converters strive to obtain the highest number of bits possible; however, this may result in a loss of conversion speed. For this reason, there are different types of converters that offer different performance characteristics depending on the intended need. The most important characteristic of an A/D converter is the transfer function or trace, which reflects the input voltage range and the format of the digital output data.
- **DAC Converter:** DAC (Digital-to-Analog Converter) converters are responsible for converting a digital code into an analog current or voltage level. If the code contains n bits, then there are 2^n possible binary combinations; each combination corresponds to a different voltage level at the output. When choosing a DAC converter, the first requirement is to consider the ideal transfer function, i.e., what type of output is required (unipolar or bipolar) and what digital input format is available.
- **PWM-Based Converter:** This type of converter is similar to the sigma-delta converter, but instead of using sigma-delta modulation, it uses PWM modulation. PWM modulation involves varying the duty cycle of a square wave depending on the input voltage, thereby achieving different pulse widths for different values. PWM modulation is present in many microcontrollers, so it can be used alternatively to implement a DAC, allowing any type of user to implement a converter.
- A potentiometer is an electronic component that acts as a variable resistor. It consists of a resistive track and a slider that moves along it, allowing the resistance between the slider and the ends of the track to be varied. By rotating the potentiometer's shaft, the position of the slider is modified, which changes the resistance and, therefore, the voltage at the connection point. Potentiometers are commonly used to control variables such as volume in audio devices, brightness in displays, or, as in this practice, the intensity of an LED.

0.5 Methodology

MATERIALS AND TOOLS

- ESP32 development board.
- Computer with Arduino IDE installed.
- USB cable to upload the code to the ESP32.
- 10K potentiometer.
- LED.
- 330 Ohm resistor.
- Connection wires.

PROCEDURE

1. **Code development in Arduino IDE:** A code was designed in Arduino IDE to read the analog signal from the potentiometer and control the LED brightness via PWM.
2. **Code designed for the practice execution:**

```
const int pinPot = 15;  // Potentiometer pin
const int LED = 13;     // LED pin

void setup() {
Serial.begin(115200); // Start serial communication at 115200 baud
pinMode(LED, OUTPUT); // Configure the LED pin as output
}
```

Explanation:

- Two constants are declared: pinPot for the pin where the potentiometer is connected (pin 15) and LED for the pin where the LED is connected (pin 13).
- In the setup() function, serial communication is initiated to monitor the values in the Arduino IDE serial monitor.
- The LED pin is configured as an output using pinMode().

```
void loop() {
int valorADC = analogRead(pinPot); // Read the potentiometer value
valorADC = (valorADC * 0.7) + (analogRead(pinPot) * 0.3); // Weighted moving average filter
}
```

Explanation:

- In the loop() function, the analog value of the potentiometer is read using analogRead(). This value is in the range of 0 to 4095 (since the ESP32´s ADC has a 12-bit resolution).
- To smooth the reading and reduce noise, a weighted moving average filter is applied. This filter combines the current reading with the previous reading, giving more weight (70%) to the current reading and less weight (30%) to the previous reading. This helps stabilize the signal.

```
    int brightness = map(valorADC, 0, 4095, 0, 255); // Convert ADC to brightness (0-255)
    int angle = map(valorADC, 0, 4095, 0, 360); // Convert ADC to angle (0-360)
```

Explanation:

- The ADC value (0-4095) is converted to a range of 0 to 255 using the map() function. This value is used to control the LED brightness, as the analogWrite() function in the ESP32 accepts values between 0 and 255 for PWM.
- The ADC value is also converted to a range of 0 to 360 degrees, simulating an angle. This could be useful in applications such as servo motor control or angular position visualization.

```
    analogWrite(LED, brightness); // Adjust the LED brightness

    Serial.print("ADC Value: ");
    Serial.print(valorADC);
    Serial.print("  -  Brightness: ");
    Serial.print(brightness);
    Serial.print("  -  Angle: ");
    Serial.println(angle);

delay(200); // Delay to stabilize the reading
```

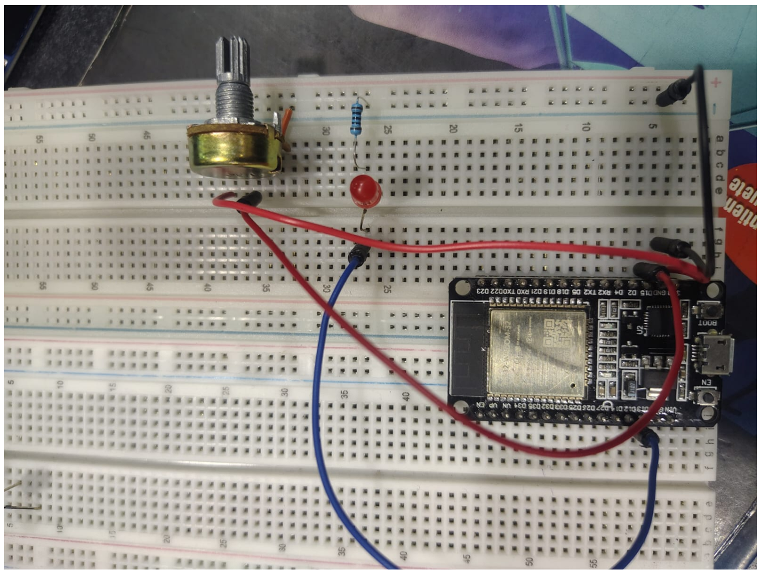
Explanation:

- The analogWrite() function adjusts the LED brightness using the brightness value (0-255). This controls the PWM signal's duty cycle, which in turn regulates the LED's light intensity.
- The values of valorADC, brightness, and angle are sent to the serial monitor for visualization. This allows real-time monitoring of the system's behavior.
- A delay of 200 ms (delay(200)) is added to stabilize the reading and avoid rapid fluctuations in the signal.

3. **Loading the code into the ESP32:** The ESP32 board was connected to the computer via a USB cable, and the code was uploaded using Arduino IDE.

4. **Testing and observations:** Tests were conducted to verify the correct operation of the system. It was observed that the LED brightness varied according to the potentiometer's position, and the ADC, brightness, and angle values were recorded in the serial monitor.

0.6 Results



After executing the code on the ESP32, the following results were obtained:

- **System behavior:** The system worked correctly, showing a direct relationship between the potentiometer's position and the LED brightness. The ADC, brightness, and angle values were correctly displayed in the serial monitor.
- **Difficulties encountered and adjustments made:** During testing, it was observed that the potentiometer reading was somewhat unstable. To solve this, a weighted moving average filter was applied in the code, which allowed for more stable readings.

0.7 Conclusion

A system was successfully implemented where the brightness of an LED depends on the position of a potentiometer, using the ESP32. It was understood how the analog signal from the potentiometer is converted to digital via the ADC and how PWM is used to generate a simulated analog signal. The relationship between the PWM duty cycle and the LED intensity was also verified. These concepts are fundamental for applications such as motor control and LED displays.