**Embeded**

**And**

**IOT System**

Name: Arslan Ali

Class: BSCS(5th-A)

Reg : 23-NTU-CS-1019

Course: Embedded IOT System

Professor: Mr. Nasir Mehmood.

Assignment: 1

Embedded Systems — ESP32

Short & Logical

Questions

**Question 1**

**Q: Why is volatile used for variables shared with ISRs?**

A: The 'volatile' keyword is used for variables that can be modified unexpectedly by an interrupt service routine (ISR). Without 'volatile', the compiler might optimize access to such variables, assuming their value doesn’t change. This can cause incorrect program behavior since ISR may update the variable outside the normal flow. Hence, 'volatile' ensures every read and write happens directly from memory.

**Q: Compare hardware-timer ISR debouncing vs. delay()-based debouncing.**

A: Hardware-timer ISR debouncing uses precise, non-blocking interrupts to filter switch bounces. It allows the microcontroller to handle other tasks concurrently and provides consistent timing. On the other hand, delay()-based debouncing blocks code execution during the delay, wasting CPU time and potentially missing other critical events. Hardware-timer ISR debouncing is therefore more efficient and responsive.

**Q: What does IRAM\_ATTR do, and why is it needed?**

A: The 'IRAM\_ATTR' attribute tells the compiler to store a function, usually an ISR, in the internal RAM (IRAM) instead of flash memory. This is necessary because flash memory may become temporarily unavailable during certain operations. Placing critical routines in IRAM ensures they run instantly and reliably even during flash operations.

**Q: Define LEDC channels, timers, and duty cycle.**

A: In the ESP32 LEDC (LED Control) module, channels represent individual PWM outputs that can control LEDs, motors, or other devices. Timers define the frequency and resolution of the PWM signals. The duty cycle is the ratio of the ON time to the total period of the PWM, expressed as a percentage, controlling the brightness or speed of the output device.

**Q: Why should you avoid Serial prints or long code paths inside ISRs?**

A: Interrupt Service Routines should execute as quickly as possible to avoid delaying other interrupts. Serial printing takes significant time due to UART transmission delays, and long code paths increase latency. Both can cause missed interrupts or system instability, hence should be avoided.

**Q: What are the advantages of timer-based task scheduling?**

A: Timer-based scheduling allows periodic task execution without blocking code using delay(). It improves system efficiency, timing accuracy, and responsiveness, allowing multiple tasks to execute concurrently without interference.

**Q: Describe I²C signals SDA and SCL.**

A: I²C uses two bidirectional lines: SDA (Serial Data) for transmitting data, and SCL (Serial Clock) for synchronizing communication. These lines are open-drain and require pull-up resistors to maintain stable high logic levels.

**Q: What is contact bounce, and why must it be handled?**

A: Contact bounce occurs when mechanical switches rapidly open and close during a press, producing multiple signals. If not debounced, it can cause the microcontroller to misinterpret multiple presses. Debouncing ensures stable input signals.

**Q: How does the LEDC peripheral improve PWM precision?**

A: The LEDC module uses high-resolution hardware timers and fine-grained control registers to produce smooth, flicker-free PWM signals. This allows for more precise brightness or speed control even at high frequencies.

**Q: Give short specifications of your selected ESP32 board (NodeMCU-32S).**

A: NodeMCU-32S is built on the ESP32-WROOM-32 module featuring a dual-core 240 MHz Xtensa processor, 520 KB SRAM, 4 MB Flash memory, Wi-Fi 802.11 b/g/n, Bluetooth 4.2, 30 GPIO pins, 12-bit ADC, DAC channels, and multiple communication interfaces like I²C, SPI, and UART.

**Question 2**

**Q: A 10 kHz signal has an ON time of 10 ms. What is the duty cycle?**

A: The duty cycle formula is: Duty Cycle = (ON Time / Period) × 100.  
For a 10 kHz signal, the period = 1 / 10,000 = 0.1 ms. Given ON time = 10 ms, which exceeds the period, the value is unrealistic. The ON time must be smaller than 0.1 ms. If ON time = 0.03 ms, then Duty Cycle = (0.03 / 0.1) × 100 = 30%.

**Q: How many hardware interrupts and timers can be used concurrently?**

A: ESP32 supports up to 32 interrupt sources per core and 4 general-purpose hardware timers (two per group). All can operate simultaneously since each timer and interrupt vector is independently configurable, allowing parallel event handling.

**Q: How many PWM-driven devices can run at distinct frequencies at the same time on ESP32?**

A: ESP32 has 8 LEDC PWM channels but only 4 LEDC timers. Each timer can operate at a unique frequency. Hence, up to 4 distinct frequencies can be generated at once, while multiple channels may share the same timer and thus the same frequency.

**Q: Compare a 30% duty cycle at 8-bit resolution and 1 kHz to a 30% duty cycle at 10-bit resolution.**

A: Both have the same average output power (30% ON time). However, 10-bit PWM offers 1024 discrete steps versus 256 in 8-bit, allowing smoother brightness or motor control transitions, especially noticeable in gradual dimming applications.

**Q: How many characters can be displayed on a 128×64 OLED at once with different font sizes?**

A: With the smallest 5×7 font (plus 1 pixel spacing), columns = 128 / 6 ≈ 21, and rows = 64 / 8 = 8, yielding about 168 characters. With a large 16×16 font, columns = 128 / 16 = 8 and rows = 64 / 16 = 4, yielding 32 characters. Thus, smaller fonts maximize display capacity.