National Textile University, Faisalabad

Department of Computer Science

Name:	Muhammad Bilal
Class:	BSCS 5th B
Registration No:	23-NTU-CS-1064
Course Name:	Embedded system and IOT
Submitted To:	Sir Nasir Mehmood
Submission Date:	19/10/2025

Assignment#1

Question 1 — Short Questions (Simple Answers)

1. Why is volatile used for variables shared with ISRs?

   Because the variable can change anytime inside the interrupt, volatile tells the compiler not to optimize or cache it, so the latest value is always read.

2. Compare hardware-timer ISR debouncing vs. delay()-based debouncing.

   Hardware-timer ISR debouncing is more accurate and doesn't block the program.

   delay()-based debouncing stops the code for some time, so it's slower and less efficient.

3. What does IRAM_ATTR do, and why is it needed?

   It tells the compiler to store the function in internal RAM so it runs faster during an interrupt (since flash memory is slower).

4. Define LEDC channels, timers, and duty cycle.

   - Channels: control different LEDs or PWM outputs.
   - Timers: decide PWM speed or frequency.
   - Duty cycle: how long the signal stays ON in one cycle (brightness or power level).

5. Why should you avoid Serial prints or long code paths inside ISRs?

   Because interrupts must be short and fast — serial printing takes time and can make the system unstable or miss other interrupts.

6. What are the advantages of timer-based task scheduling?

   It runs tasks automatically at fixed times, so we don't need delay() and the program stays responsive.

7. Describe I²C signals SDA and SCL.

   - SDA (Serial Data): carries the data.

- SCL (Serial Clock): provides timing for sending/receiving bits.

8. What is the difference between polling and interrupt-driven input?
   - Polling: CPU keeps checking the input repeatedly.
   - Interrupt: CPU does other work and only reacts when input changes — more efficient.
9. What is contact bounce, and why must it be handled?

   When a button is pressed, it quickly connects and disconnects many times, causing false signals. Debouncing removes this noise.
10. How does the LEDC peripheral improve PWM precision?

    LEDC uses hardware timers and higher bit resolution, giving smoother and more accurate brightness control.
11. How many hardware timers are available on the ESP32?

    ESP32 has 4 hardware timers (two in each group).
12. What is a timer prescaler, and why is it used?

    It divides the main clock to slow down the timer, helping adjust timing for different applications.
13. Define duty cycle and frequency in PWM.
    - Duty cycle: % of time the signal stays HIGH.
    - Frequency: how many complete ON-OFF cycles happen per second.
14. How do you compute duty for a given brightness level?

    Duty = (Brightness % × Max Duty Value) ÷ 100
15. Contrast non-blocking vs. blocking timing.
    - Blocking (delay): stops everything for some time.
    - Non-blocking (millis/timers): allows other tasks to run while waiting.
16. What resolution (bits) does LEDC support?

    It supports 1 to 20-bit resolution (usually 8–10 bits for LED control).
17. Compare general-purpose hardware timers and LEDC (PWM) timers.

- Hardware timers: used for general timing and interrupts.
- LEDC timers: designed specially for PWM signals and brightness control.

18. What is the difference between Adafruit_SSD1306 and Adafruit_GFX?

    SSD1306 controls the OLED hardware, while GFX provides drawing functions like text, shapes, and graphics.

19. How can you optimize text rendering performance on an OLED?

    Only update the part that changes instead of redrawing the full screen.

20. Give short specifications of your selected ESP32 board (NodeMCU-32S).

    Dual-core 240 MHz CPU, Wi-Fi + Bluetooth, 4 MB Flash, 520 KB RAM, 30 GPIO pins, supports ADC, PWM, UART, SPI, I²C.

Question 2 — Logical Questions (Simple Answers)

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

   Duty cycle = (ON time / Total time) × 100

   = (10 ms / 100 ms) × 100 = 10%

   It means the signal stays ON 10% of each cycle.

2. How many hardware interrupts and timers can be used concurrently?

   ESP32 has many GPIO interrupts (one per pin) and 4 hardware timers.

   So multiple interrupts can run together, but only 4 hardware timers can work independently at once.

3. How many PWM-driven devices can run at distinct frequencies at the same time on ESP32? Explain constraints.

ESP32 has 4 LEDC timers, and each timer can run at its own frequency.

So, only 4 different frequencies can be used at the same time.

But since each timer supports up to 8 channels, we can control many devices — they just share the same frequency.

In short:

- 4 different frequencies (one per timer)
- Up to 32 channels total (8 × 4), but channels using the same timer must share the same frequency.

4. Compare a 30% duty cycle at 8-bit resolution and 1 kHz to a 30% duty cycle at 10-bit resolution (all else equal).

Both have the same brightness (30%), but the 10-bit resolution has smoother control because it can represent more levels (1024 steps vs. 256).

So, with 10-bit, the light changes look softer and more precise.

The 1 kHz frequency just means how fast it repeats; the duty cycle still controls brightness.

5.How many characters can be displayed on a 128×64 OLED at once with the minimum font size vs. the maximum font size? State assumptions.

Assuming we use the default Adafruit fonts:

- Minimum font (6×8 pixels) 128 ÷ 6 = ~21 characters per line, 64 ÷ 8 = 8 lines about 21 × 8 = 168 characters total.
- Maximum font (12×16 pixels) 128 ÷ 12 = ~10 characters per line, 64 ÷ 16 = 4 lines about 10 × 4 = 40 characters total.

So, smaller fonts can show more text, and larger fonts show fewer but clearer characters