30-Day FreeRTOS Course for ESP32 Using ESP-IDF

(Day 6)

“Using vTaskDelay() and vTaskDelayUntil()”

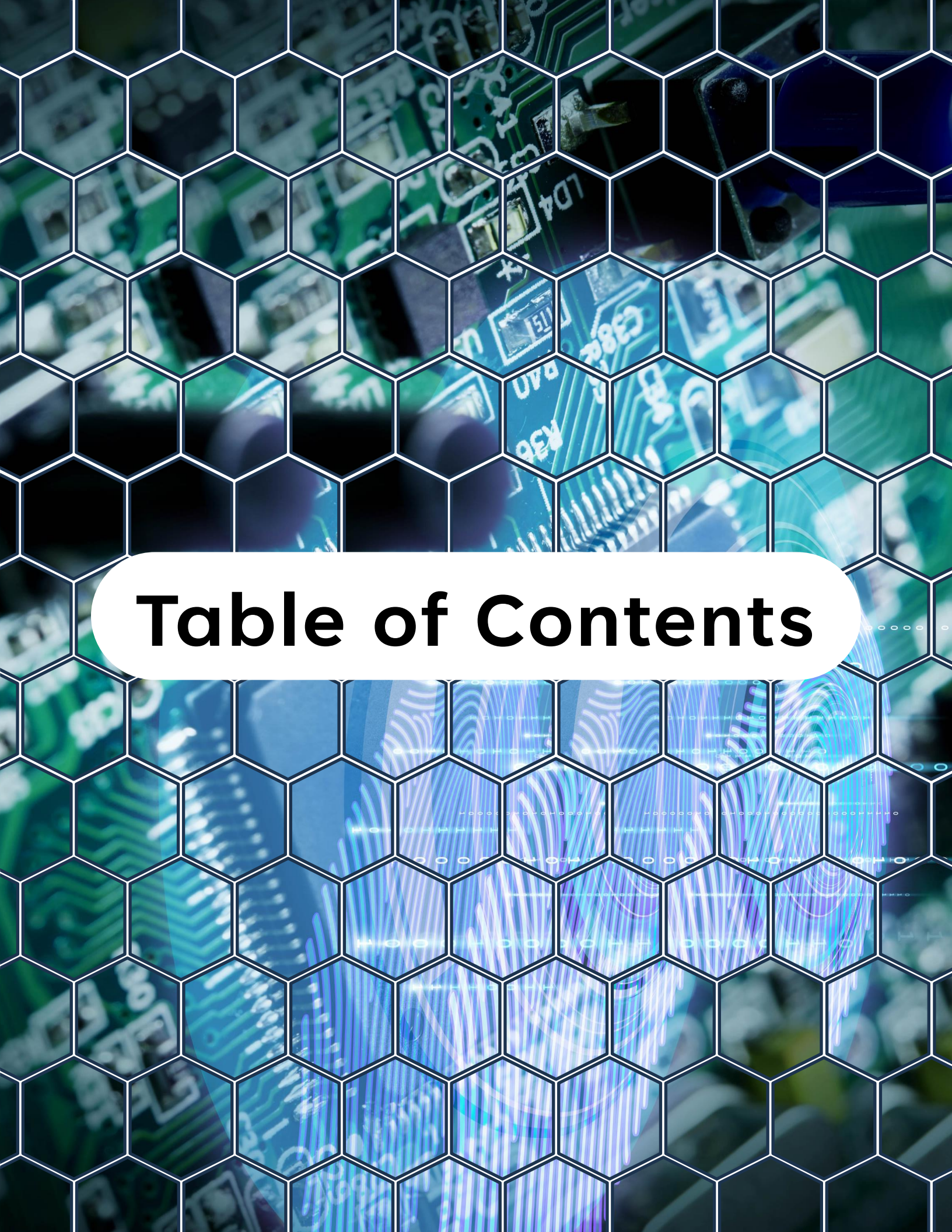
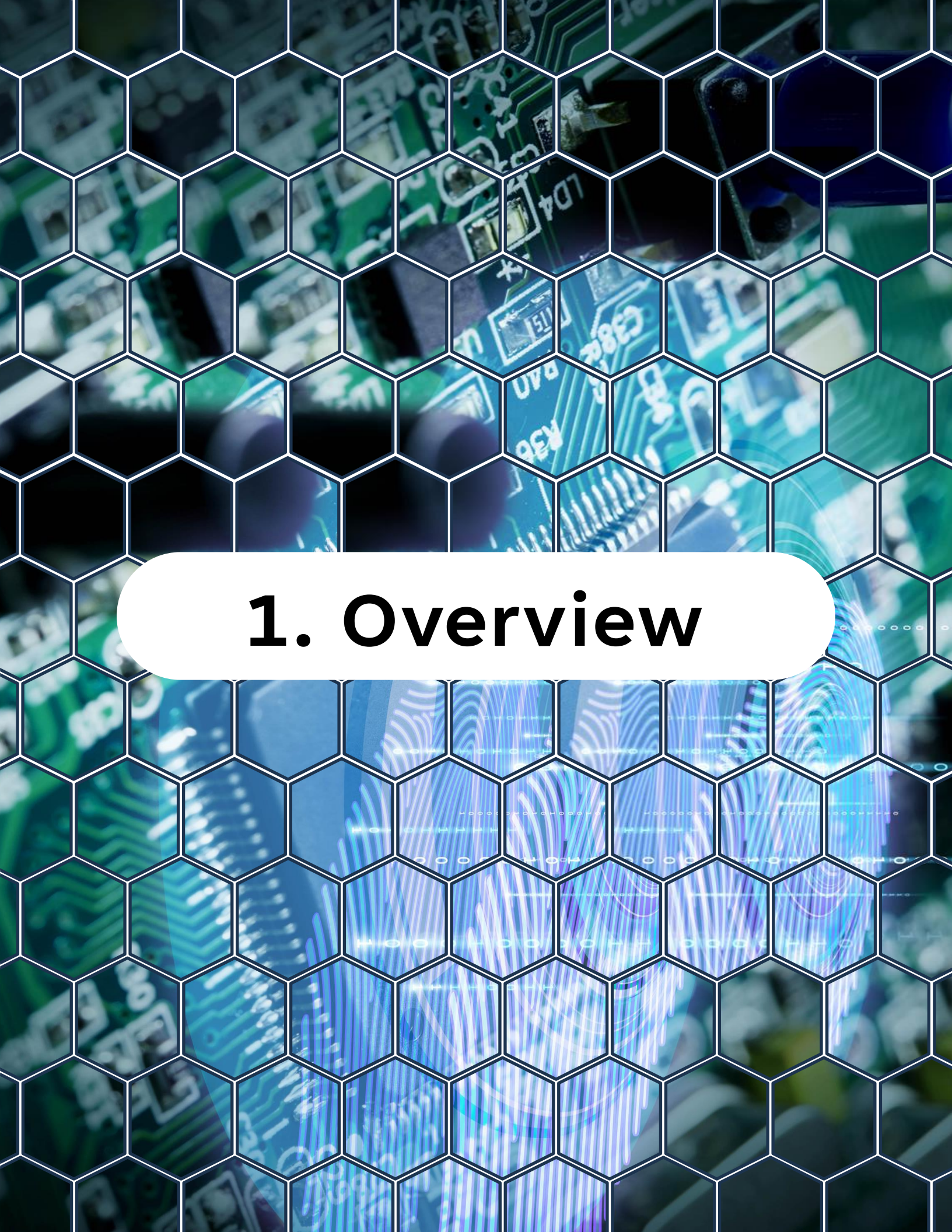


Table of Contents

Table of Contents



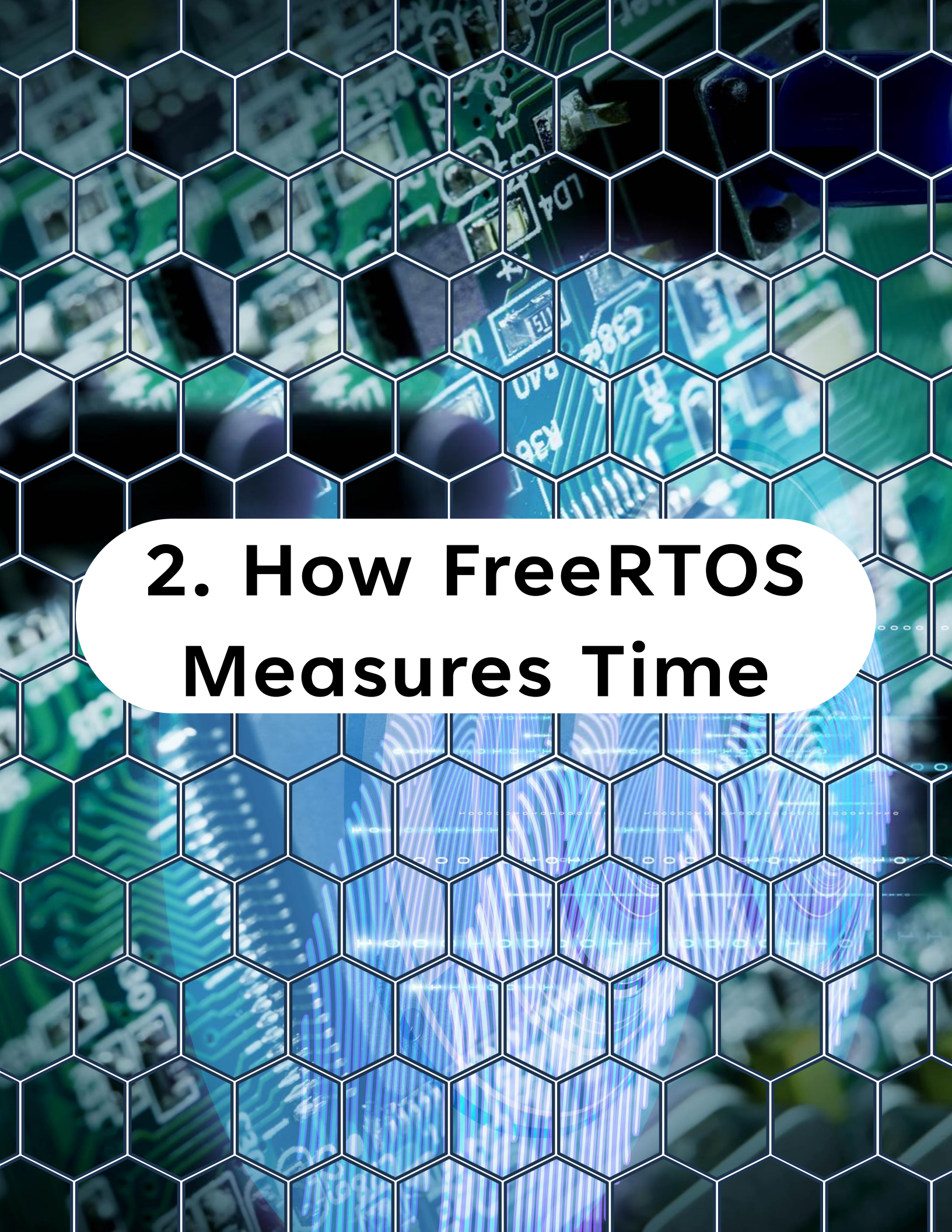
1. Overview

1. Overview

On **Day 6**, you'll learn:

- How FreeRTOS measures and manages time
- The difference between `vTaskDelay()` and `vTaskDelayUntil()`
- How to use these functions to implement periodic tasks
- When to choose one over the other
- Practical examples and timing considerations

By the end of this lesson, you'll know how to make your tasks run at precise intervals without wasting CPU time.



2. How FreeRTOS Measures Time

2. How FreeRTOS Measures Time

FreeRTOS uses **ticks** as its base time unit.

- A **tick** is a periodic interrupt generated by the system tick timer.
- The default tick rate in ESP-IDF is **100 Hz** (1 tick = 10 ms), but this can be changed in:

  `menuconfig → Component config → FreeRTOS→ Tick rate (Hz)`

📌 Use the macro `pdMS_TO_TICKS(ms)` to convert milliseconds to ticks.

Example:

```
vTaskDelay(pdMS_TO_TICKS(500)); // delay for 500 ms
```

3. vTaskDelay()

3. vTaskDelay()

`vTaskDelay()` suspends a task for a given relative period.

Syntax:

```
void vTaskDelay(const TickType_t xTicksToDelay);
```

- **Relative delay**: The count starts when `vTaskDelay()` is called.
- Allows drift if the task execution time varies.

Example:

```
vTaskDelay(pdMS_TO_TICKS(1000)); // Wait 1 second
```

📌 Best for: Non-critical periodic actions, or when exact timing is not essential.

4. vTaskDelayUntil()

4. vTaskDelayUntil()

`vTaskDelayUntil()` suspends a task until a specified absolute tick count.

Syntax:

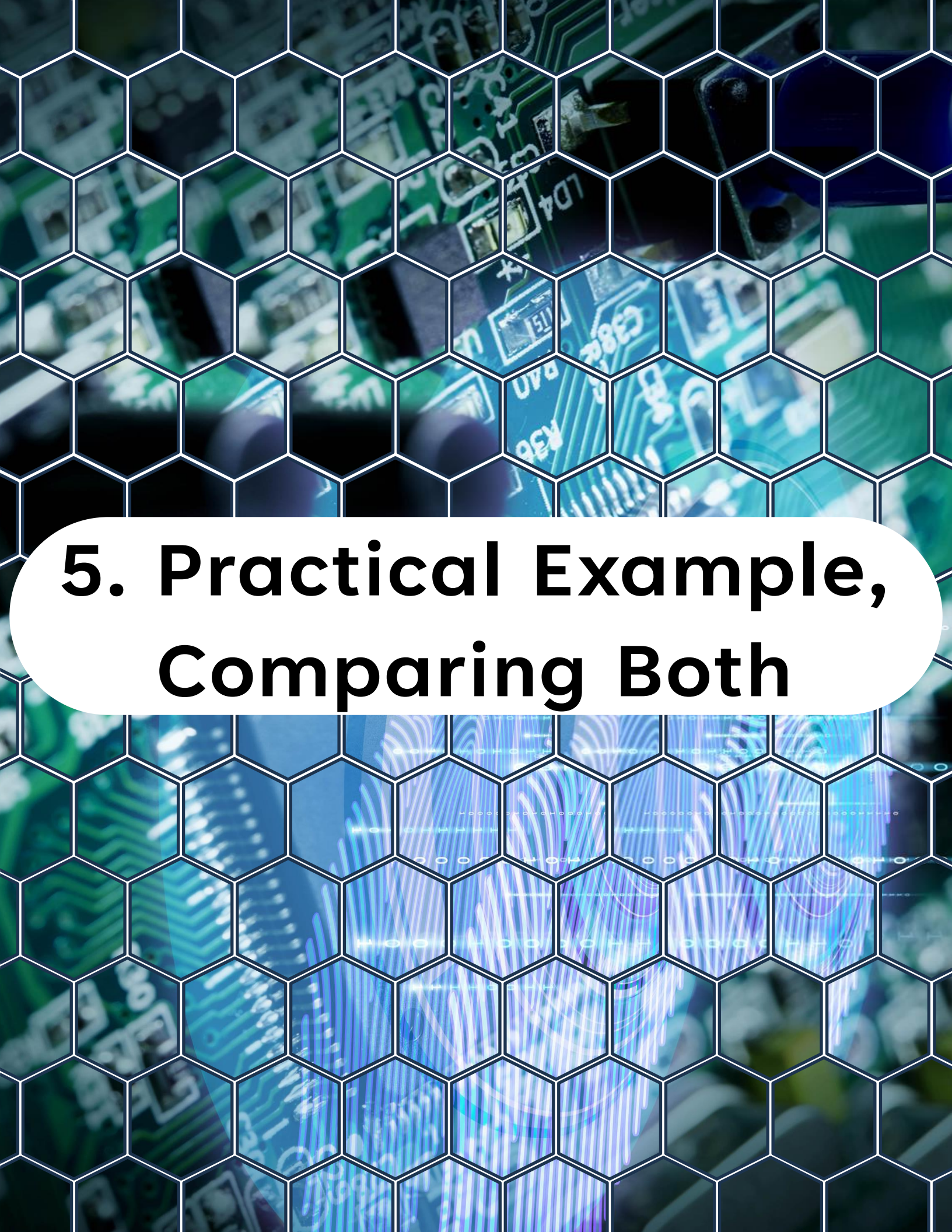
```
void vTaskDelayUntil(TickType_t *pxPreviousWakeTime,
                     const TickType_t xTimeIncrement);
```

- **Absolute delay**: Keeps a fixed schedule, minimizing drift.
- Requires you to store and maintain the last wake time.

Example:

```
TickType_t last_wake_time = xTaskGetTickCount();
vTaskDelayUntil(&last_wake_time, pdMS_TO_TICKS(1000));
```

📌 Best for: Periodic tasks that must run **exactly every N milliseconds**.



5. Practical Example, Comparing Both

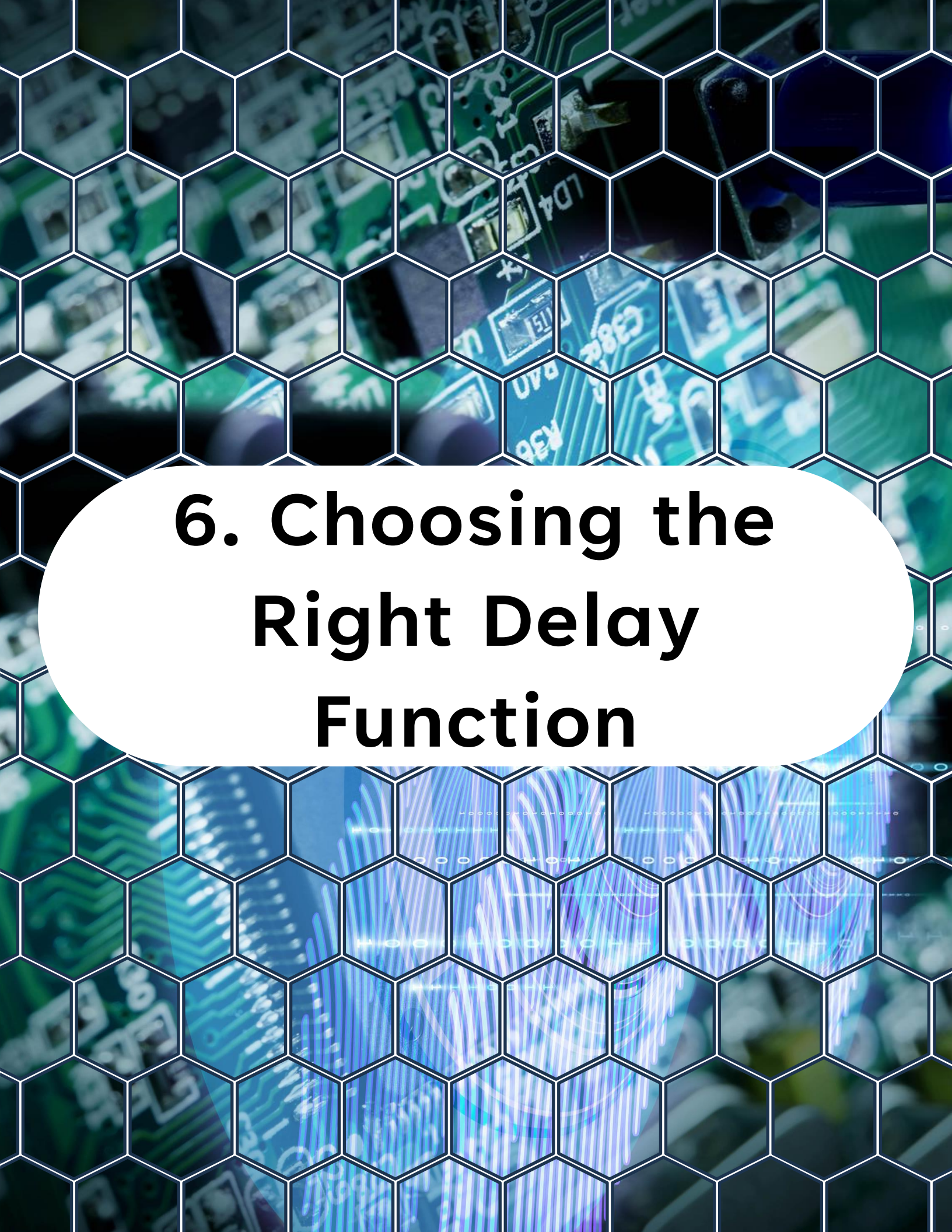
5. Practical Example, Comparing Both

```c
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"

void task_delay(void *pvParameter) {
    while (1) {
        printf("vTaskDelay: %lu ms\n",
               (unsigned long)(xTaskGetTickCount() * portTICK_PERIOD_MS));
        vTaskDelay(pdMS_TO_TICKS(1000)); // Delay 1s
    }
}

void task_delay_until(void *pvParameter) {
    TickType_t last_wake = xTaskGetTickCount();
    while (1) {
        printf("vTaskDelayUntil: %lu ms\n",
            (unsigned long)(xTaskGetTickCount() * portTICK_PERIOD_MS));
        // Delay until next second
        vTaskDelayUntil(&last_wake, pdMS_TO_TICKS(1000));
    }
}

void app_main() {
    xTaskCreate(task_delay, "TaskDelay", 2048, NULL, 5, NULL);
    xTaskCreate(task_delay_until, "TaskDelayUntil", 2048, NULL, 5, NULL);
}

```

5. Practical Example, Comparing Both

Expected Behavior

- `vTaskDelay()`: Each loop starts 1 second after the last one ended → small delays in execution accumulate over time.
- `vTaskDelayUntil()`: Each loop starts at fixed intervals regardless of execution time (as long as it’s less than the interval).



6. Choosing the Right Delay Function

6. Choosing the Right Delay Function

Function	Timing Type	Use Case
vTaskDelay()	Relative	General waits, non-critical timing
vTaskDelayUntil()	Absolute	Precise periodic scheduling

7. Best Practices

7. Best Practices

✅ Always use `pdMS_TO_TICKS()` for portability.

✅ Ensure your task's execution time is **less than the delay period** when using `vTaskDelayUntil()`.

✅ Avoid using `vTaskDelay(0)` — it does nothing; use `taskYIELD()` instead if you want to yield immediately.

✅ For very high timing precision, consider increasing tick rate or using hardware timers.

8. Summary

8. Summary

- `vTaskDelay()` → relative delay, simpler, may drift over time.
- `vTaskDelayUntil()` → absolute delay, consistent execution intervals.
- Use the right one depending on your application’s timing needs.

9. Challenge for Today

9. Challenge for Today

- Create a **sensor sampling task** that reads data every 200 ms exactly, using `vTaskDelayUntil()`.
- Add another task that blinks an LED every **1 second** using `vTaskDelay()` and observe the difference in timing stability.