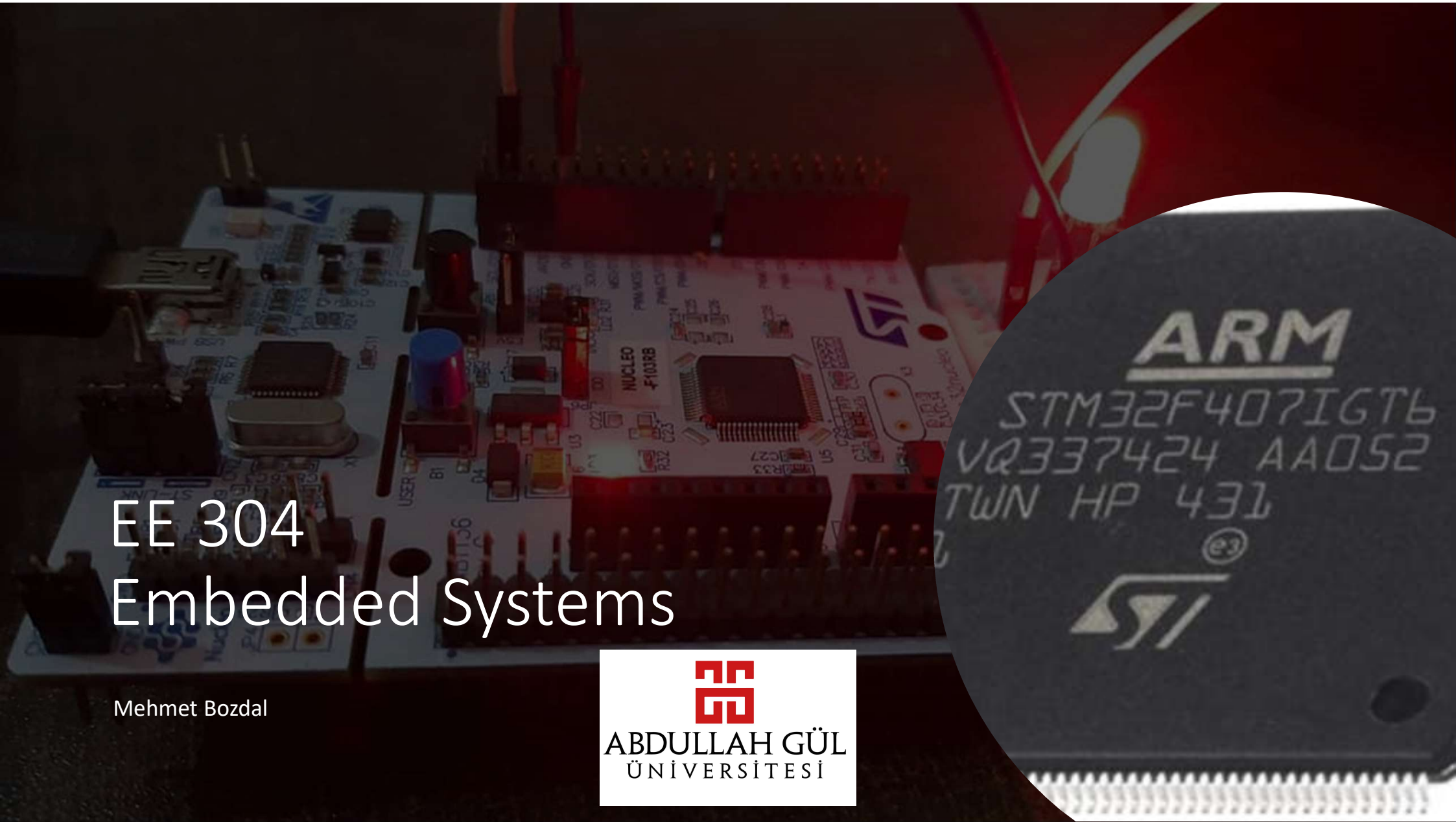

EE 304
Embedded Systems

Mehmet Bozdal

ABDULLAH GÜL
ÜNİVERSİTESİ

Agenda

- System Timer – SysTick Aplications

Delay Generation

- One common use of SysTick is to generate precise delays in embedded systems. This is useful in scenarios where you need to introduce delays between different parts of your code.

Delay Generation

```
void delay_ms(uint32_t milliseconds) {
    // Set the reload value for SysTick
    SysTick->LOAD = SystemCoreClock / 1000 - 1;

    // Clear the current value and enable SysTick
    SysTick->VAL = 0;
    SysTick->CTRL |= SysTick_CTRL_ENABLE;

    // Wait until the count flag is set
    while (milliseconds > 0) {
        if (SysTick->CTRL & SysTick_CTRL_COUNTFLAG) {
            milliseconds--;
        }
    }

    // Disable SysTick
    SysTick->CTRL &= ~SysTick_CTRL_ENABLE;
}
```


AGÜ

Periodic Interrupt

- SysTick can be configured to generate periodic interrupts, providing a time reference for periodic tasks in your system.

Periodic Interrupt

```c
#include <stdint.h>
volatile uint32_t milliseconds = 0;
void SysTick_Handler(void) {
    // Increment the milliseconds counter
    milliseconds++;
}
int main(void) {
    // Set the reload value for SysTick
    SysTick->LOAD = SystemCoreClock / 1000 - 1;
    // Set SysTick interrupt priority
    NVIC_SetPriority(SysTick_IRQn, 0);
    // Enable SysTick interrupt and enable SysTick
    SysTick->CTRL |= SysTick_CTRL_TICKINT | SysTick_CTRL_ENABLE;
    // Main loop
    while (1) {
        // Your main code goes here
    }
}
```


AGÜ

Timekeeping

- SysTick can be used for timekeeping in an embedded system. You can keep track of elapsed time or measure the execution time of specific code segments.

Timekeeping

```
volatile uint32_t execution_time = 0;
volatile uint32_t start_time = 0;
volatile uint32_t counter = 0;
void SysTick_Handler(void) {
    counter++;
}
int main(void) {
    // Initialize SysTick for microsecond-based timekeeping
    SysTick_Init();

    while (1) {
        // Start measuring the execution time of a specific code segment
        start_time = counter;
        // The code segment you want to measure
        // ...
        // Stop measuring the execution time
        execution_time = counter - start_time;
    }
}
```

Timekeeping

```
volatile uint32_t execution_time = 0;
volatile uint32_t start_time = 0;
volatile uint32_t counter = 0;
void SysTick_Handler(void) {
    counter++;
}
int main(void) {
    // Initialize SysTick for microsecond-based timekeeping
    SysTick_Init();

    while (1) {
        // Start measuring the execution time of a specific code segment
        start_time = counter;
        // The code segment you want to measure
        // ...
        // Stop measuring the execution time
        execution_time = counter - start_time;
    }
}
```

Be careful about overflow

Energy-Efficient Sleep Mode

```
void SysTick_Handler(void) {
    // Perform actions needed before entering sleep mode
}
int main(void) {
    // Set the reload value for SysTick (adjust for desired sleep interval)
    SysTick->LOAD = SystemCoreClock / 2 - 1;
    // Set SysTick interrupt priority
    NVIC_SetPriority(SysTick_IRQn, 0);
    // Enable SysTick interrupt and enable SysTick
    SysTick->CTRL |= SysTick_CTRL_TICKINT_Msk | SysTick_CTRL_ENABLE_Msk;
    // Main loop
    while (1) {
        // Your main code goes here
        // Example: Enter sleep mode when idle
        __WFI();
    }
}
```

Does NOT work

AGÜ

Q&A

Any questions?