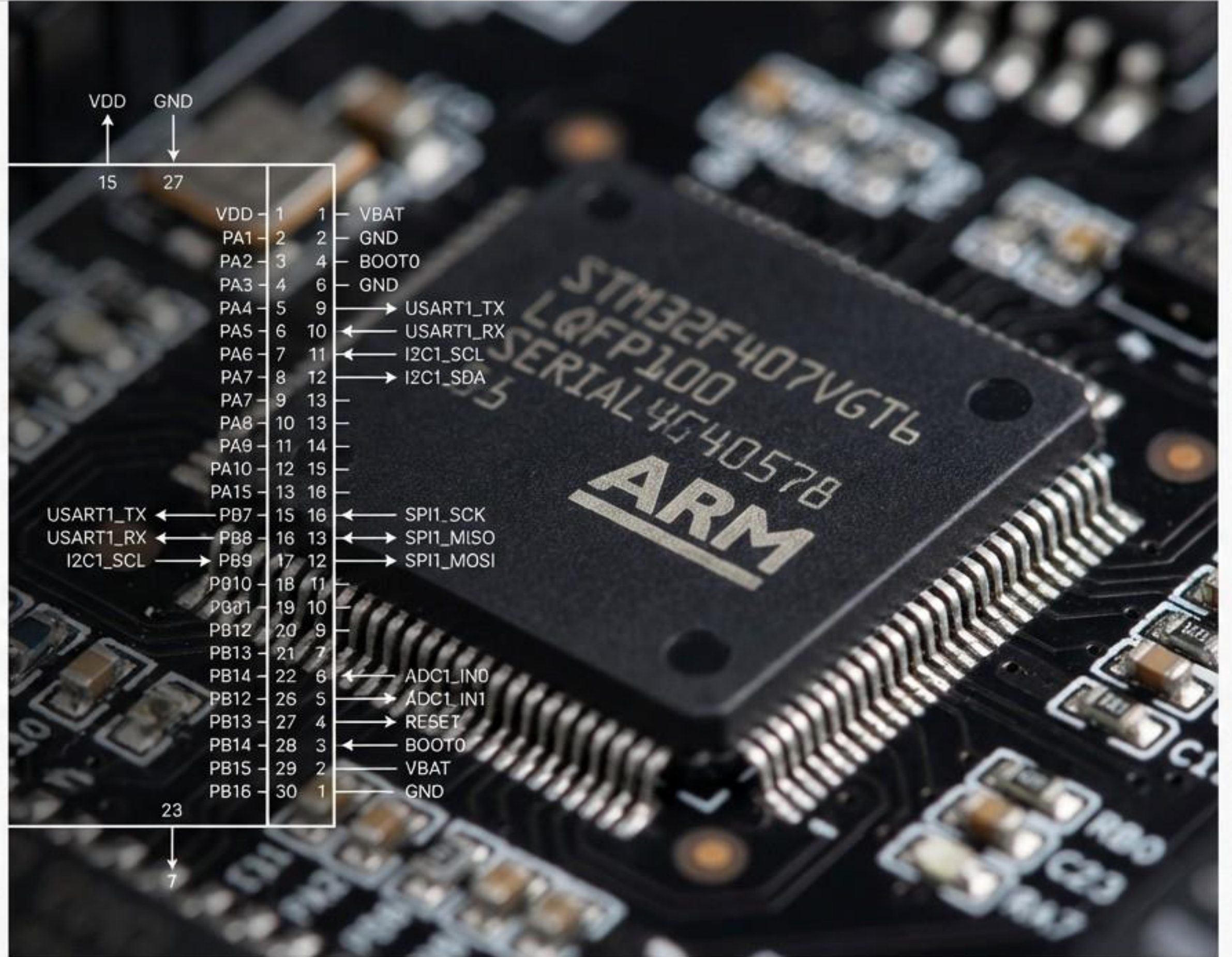


# STM32F407VGT6 ile FreeRTOS Eğitimi

Gerçek Zamanlı İşletim  
Sistemi Programlama  
Uzmanlığına Yolculuk

Profesyonel gömülü sistem geliştirme  
dünyasına adım atın.





# Neden Bu Eğitim?

Modern gömülü sistemlerde karmaşık uygulamalar geliştirmek için RTOS artık bir seçenek değil, bir zorunluluktur.



## Profesyonel Yetkinlik

Endüstri standardı beceriler ve iş dünyasında aranan kritik yetenekler.



## Karmaşıklık Yönetimi

Çok görevli (multitasking) mimariler tasarlama becerisi.



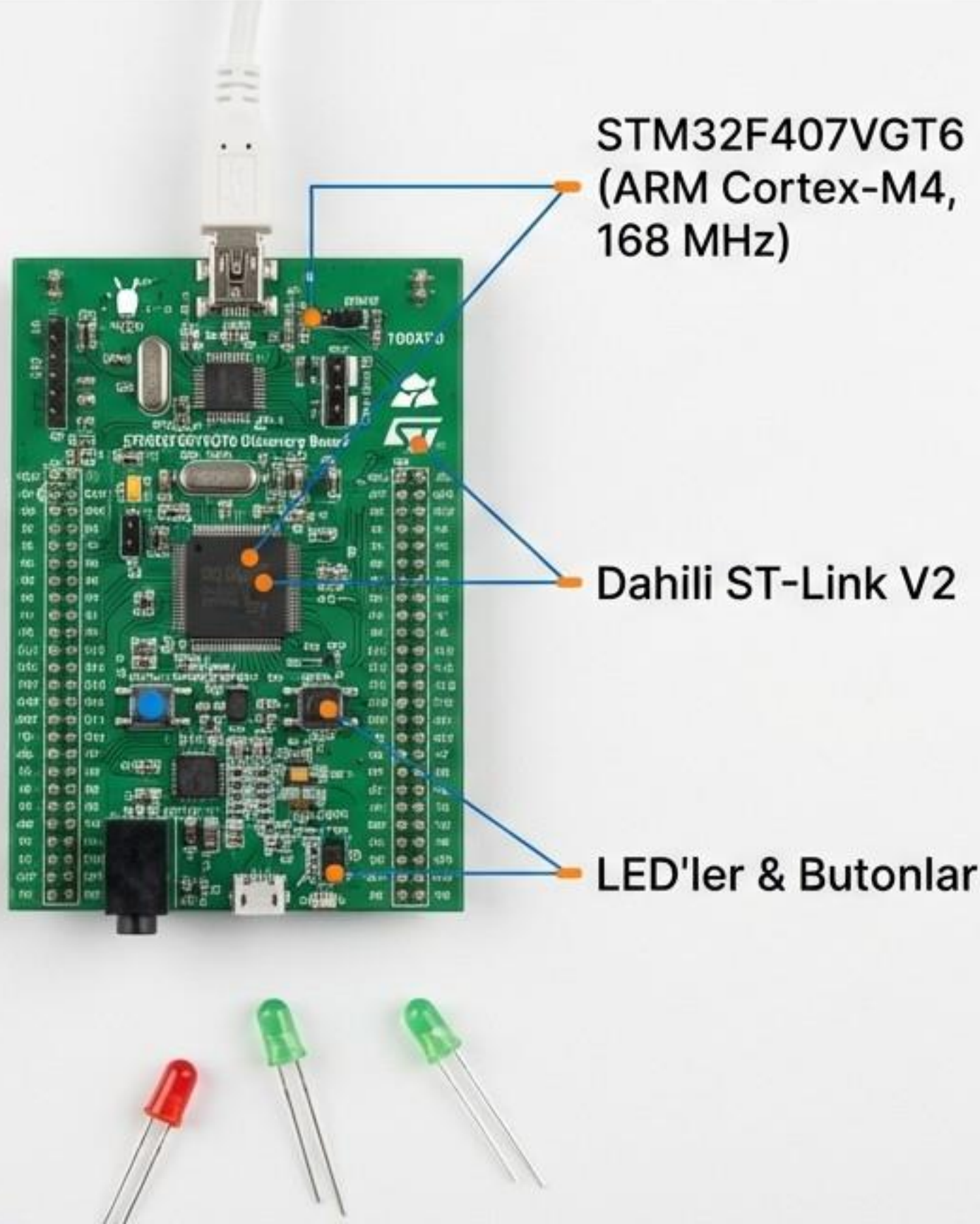
## Güvenilir Çözümler

Açık kaynaklı ve güvenilir FreeRTOS mimarisi.

Hedef Kitle: C bilenler, STM32 kullanıcıları ve kariyerini ilerletmek isteyen mühendisler.



# Geliştiricinin Araç Seti



STM32F407VGT6  
(ARM Cortex-M4,  
168 MHz)

Dahili ST-Link V2

LED'ler & Butonlar

## Yazılım Stack



IDE: STM32CubeIDE (Önerilen)



Konfigürasyon: STM32CubeMX



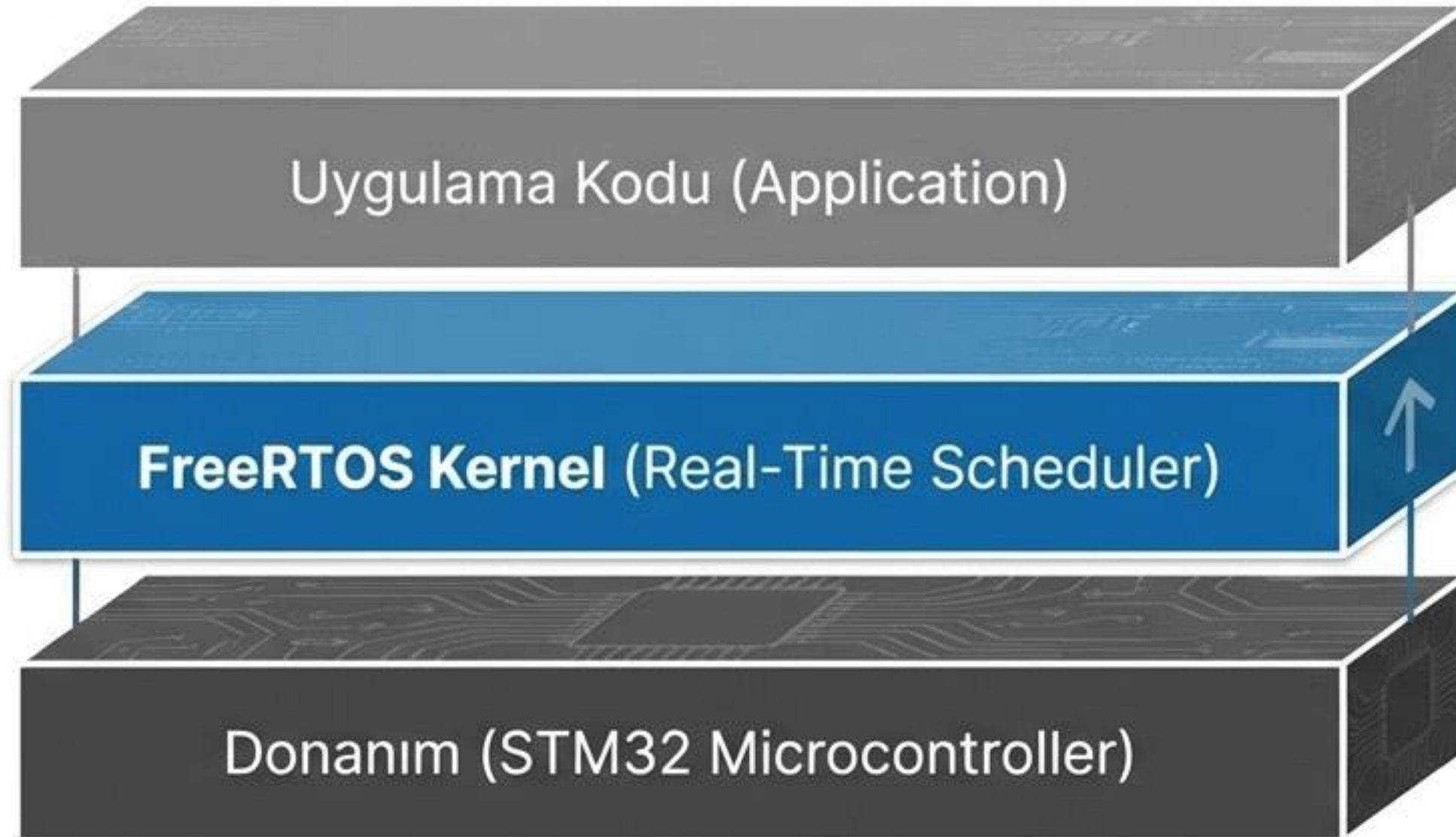
Kütüphane: FreeRTOS Kernel



Araçlar: Terminal (PuTTY/Tera Term)

# FreeRTOS Nedir?

Mikrodenetleyiciler için özel olarak tasarlanmış, açık kaynaklı (MIT Lisans), gerçek zamanlı işletim sistemi çekirdeği.



## TEKNİK ÖZELLİKLER

- Küçük Ayak İzi: 4-5 KB ROM, 256 byte RAM
- Platform: 40+ Mikrodenetleyici Ailesi
- Deterministik: Gerçek zamanlı yanıt
- Modüler: Bakımı kolay yapı



# Uzmanlığa Giden Yol Haritası

## Aşama 1: Temeller



- Görevler (Tasks), Zamanlayıcı (Scheduler)

## Aşama 2: İletişim



- Kuyruk Yapısı (Queues), Veri Transferi

## Aşama 3: Senkronizasyon



- Semaforlar, Mutex, Kaynak Koruması

## Aşama 4: İleri Seviye



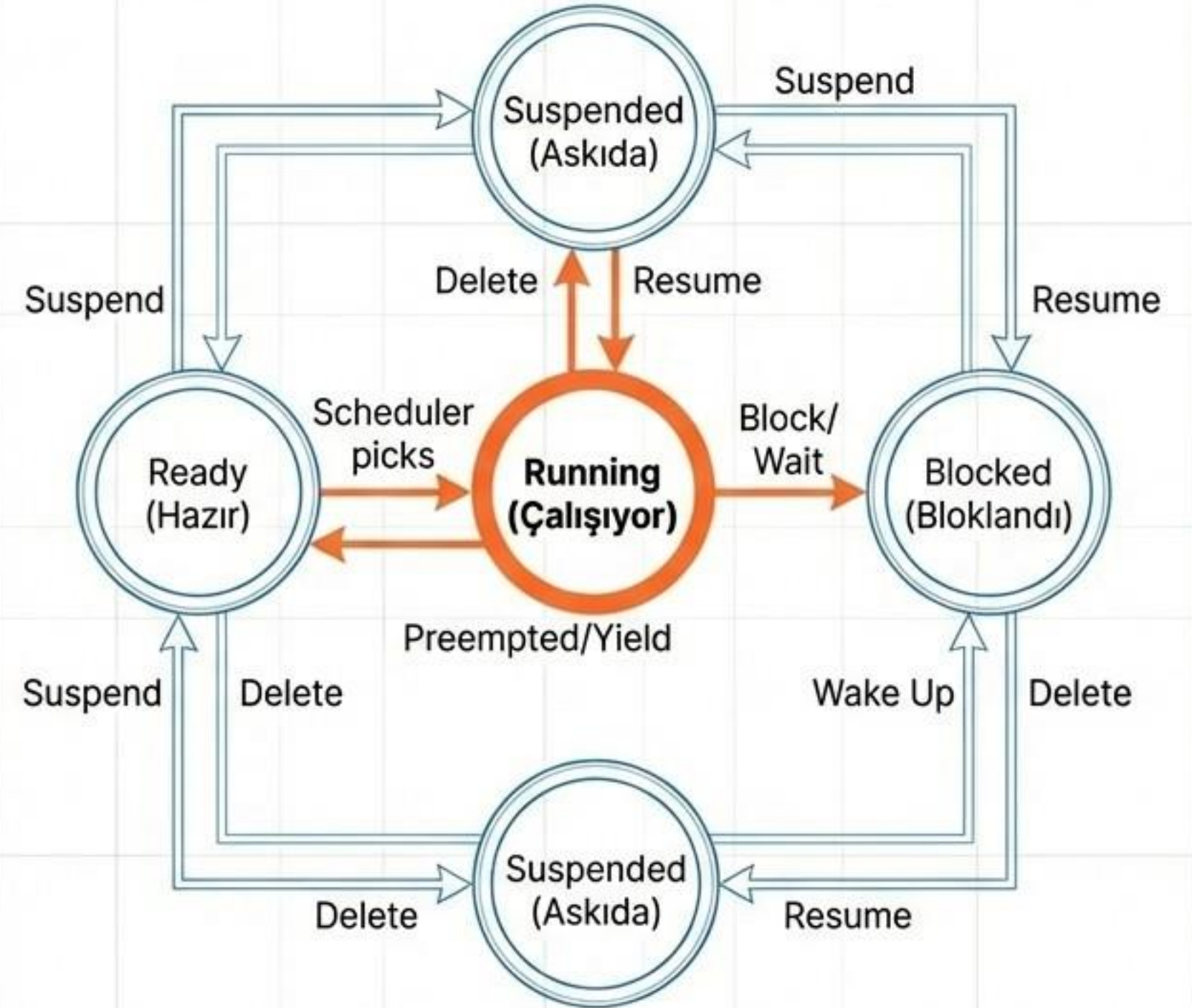
- Software Timers, Event Groups, Hook Fonksiyonları



# Bölüm 1: Görev Yönetimi (Tasks)

Uygulamayı yönetilebilir parçalara bölmek.

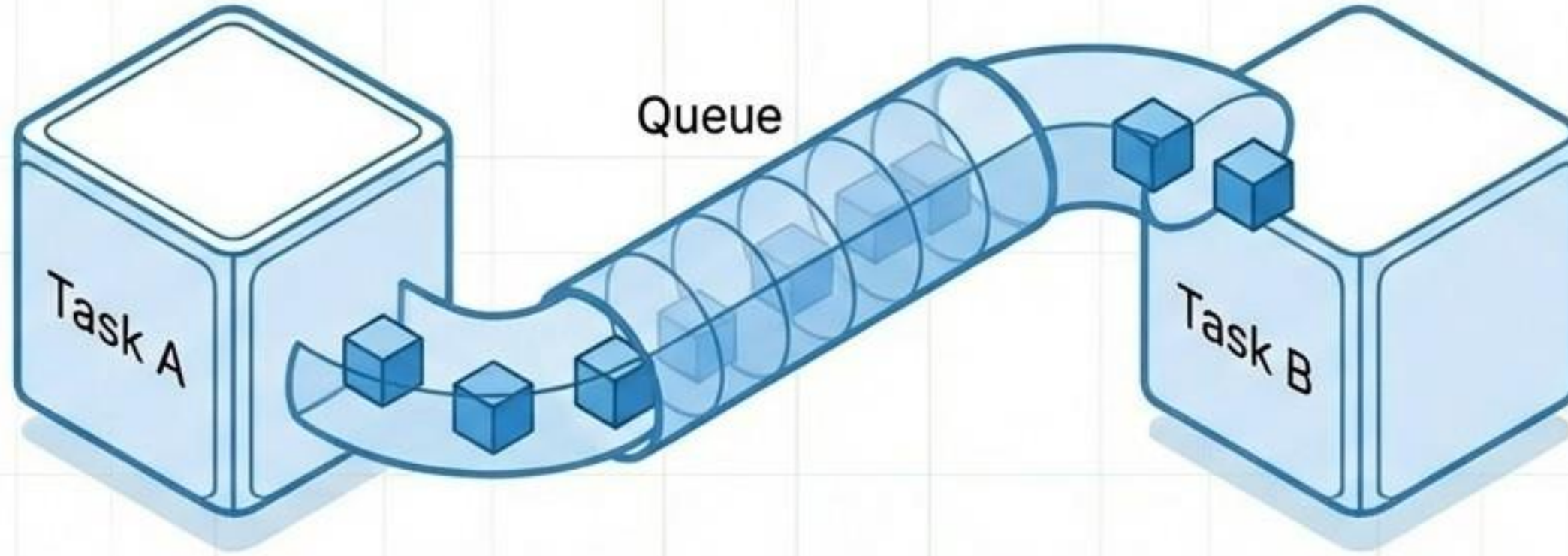
- **Zamanlayıcı (Scheduler):** Öncelik (Priority) sırasına göre çalışma mantığı.
- **Sistem Görevleri:** Idle task ve tick interrupt.
- **Operasyonlar:** Görev oluşturma (Creation), silme ve önceliklendirme.





# Bölüm 2: Veri Akışı ve Kuyruklar (Queues)

Görevler arası güvenli veri transferi.



## Mekanizma

- FIFO (First In First Out) yapısı
- Veri kopyalama ile transfer

## Operasyonlar

- Blocking: Veri beklerken uyuma
- Non-blocking: Anlık kontrol

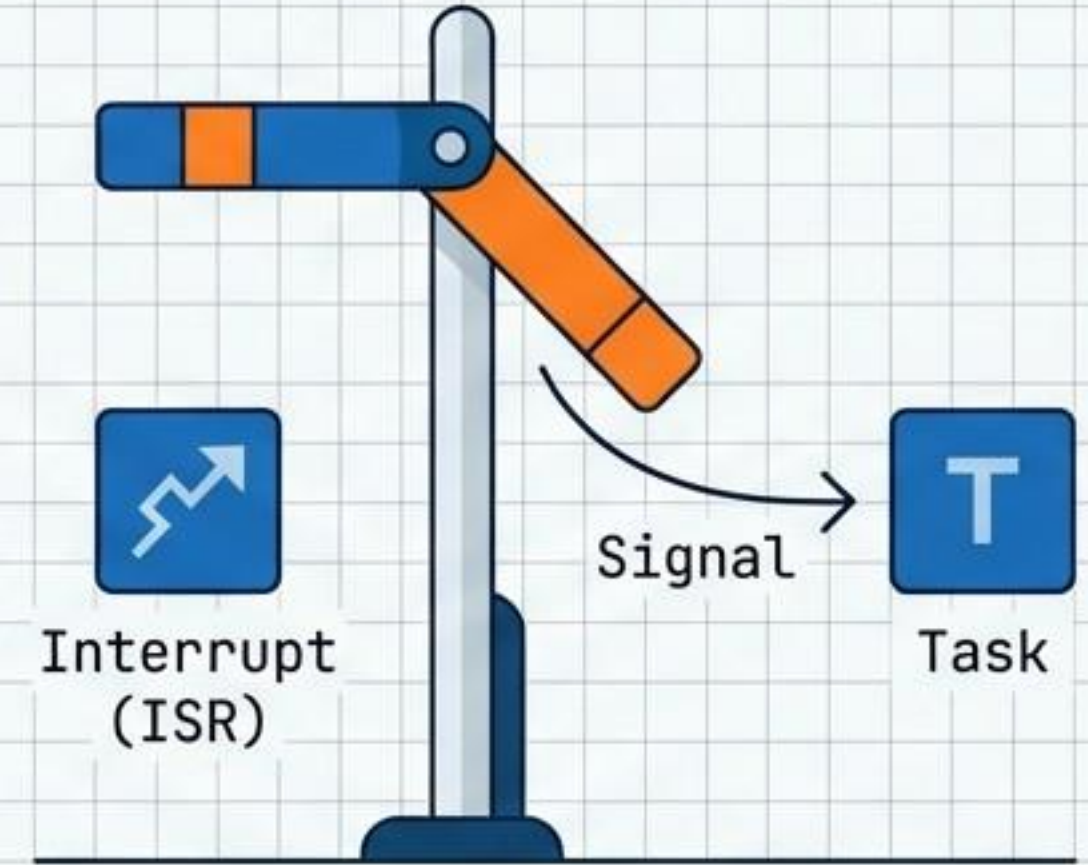
## İleri Teknikler

- Queue Sets
- Mailbox Pattern



# Bölüm 3: Senkronizasyon (Semaphores)

Görevler arası koordinasyon ve sinyalleşme.



## Binary Semaphore

- Tek bitlik sinyal mekanizması.
- Kullanım: Bir olay gerçekleştiğinde (örn. butona basılması) görevi uyandırmak.

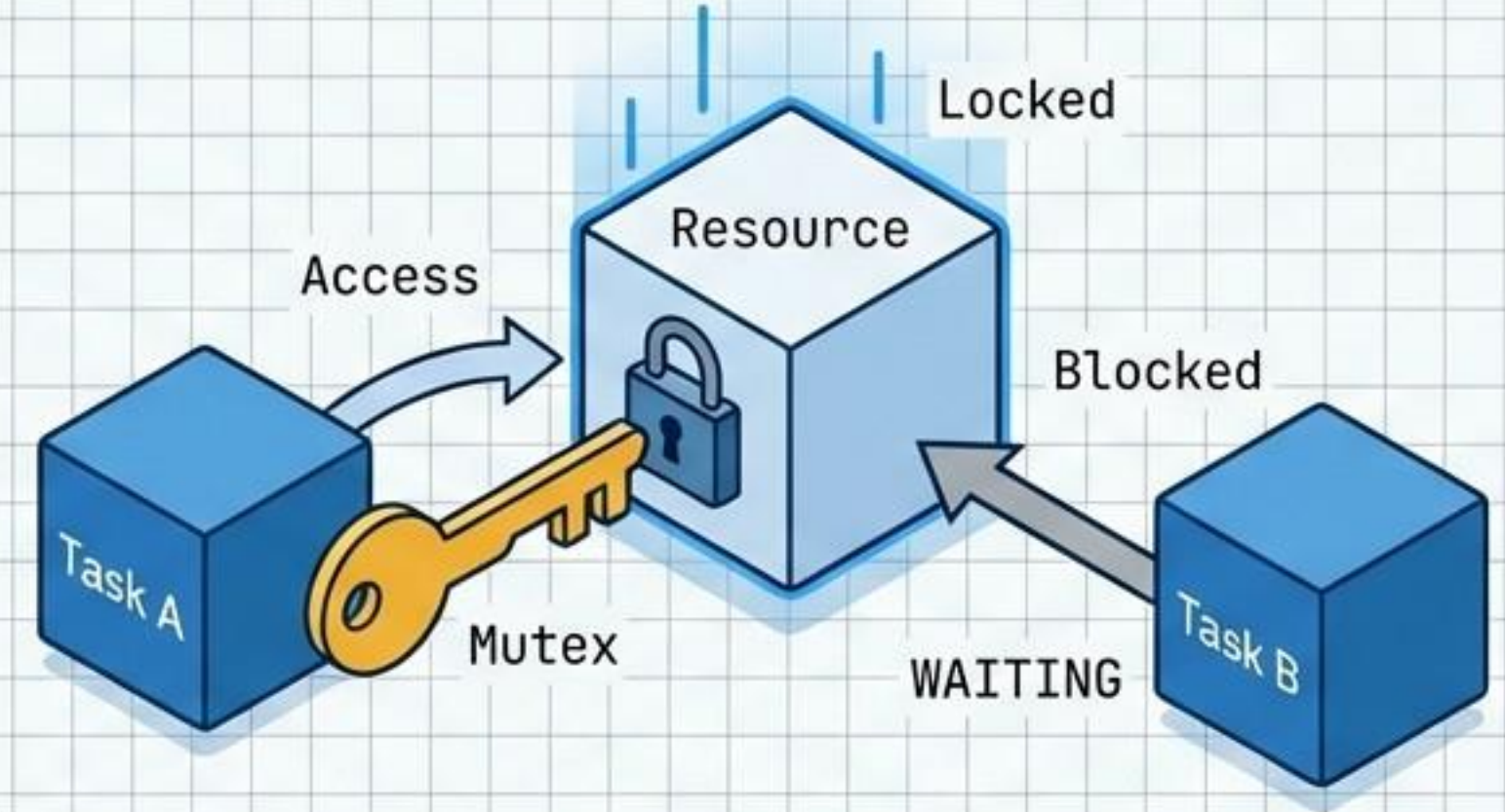
## Counting Semaphore

- Kaynak sayımı ve yönetimi.
- Kullanım: Birden fazla erişim hakkı olan kaynakların takibi.



# Bölüm 4: Kaynak Koruması (Mutex)

Paylaşılan kaynaklara kontrollü erişim.



## Kritik Sorun: Priority Inversion

- Düşük öncelikli bir görev, yüksek öncelikli görevi bloklayabilir.
- Çözüm: **Priority Inheritance** (Öncelik Mirası) mekanizması.

## Güvenlik Protokolü

- **Deadlock (Kilitlenme)**: İki görevin birbirini sonsuza kadar beklemesi.
- **Recursive Mutex**: Aynı görevin kilidi birden çok kez alabilmesi.



# Bölüm 5 & 6: Zamanlama ve Verimlilik



## Task Notifications (Bildirimler)

- Lightweight senkronizasyon.
- Daha az RAM ve CPU kullanımı sağlar.
- Doğrudan bir göreve sinyal veya değer göndermek için kullanılır.



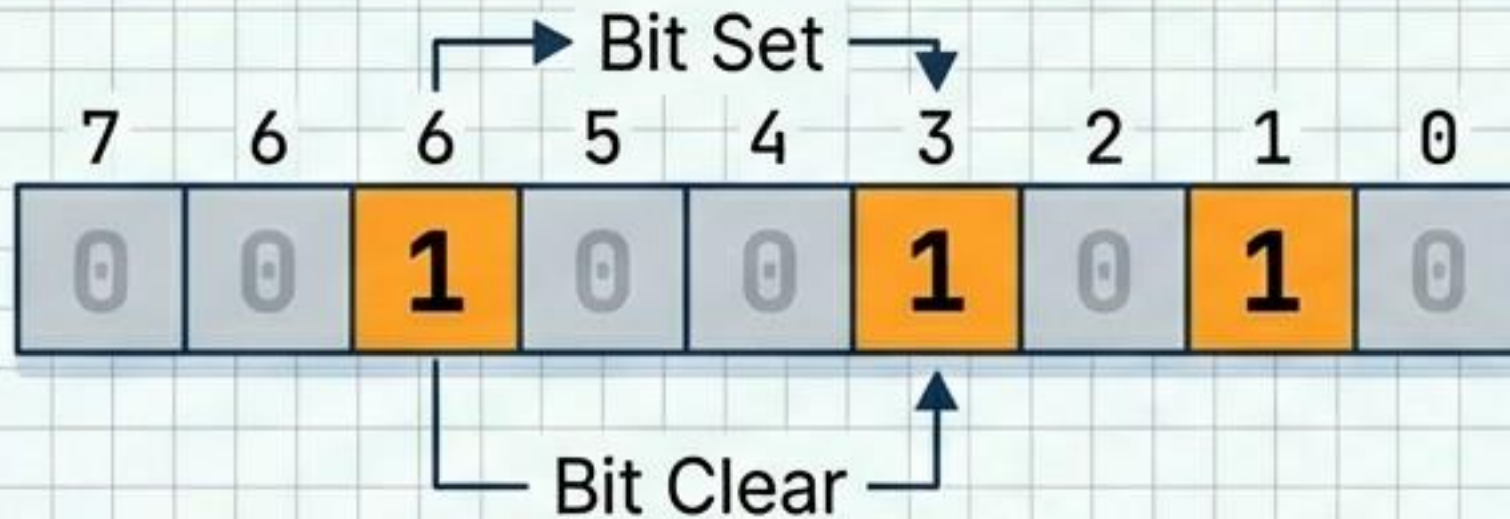
## Software Timers (Yazılım Zamanlayıcıları)

- Donanım timer'ı kullanmadan zamanlama işlemleri.
- One-shot (tek seferlik) veya **Auto-reload** (tekrarlı) modlar.
- Callback fonksiyonları ile çalışır.



# Bölüm 7 & 8: Gelişmiş Sistem Kontrolü

## Event Groups (Olay Grupları)



Flag tabanlı kontrol. Birden fazla olayın gerçekleşmesini beklemek (AND/OR mantığı) için idealdir. Event bit set/clear işlemleri.

## Hook Fonksiyonları (Sistem Kancaları)

Sistemin kritik anlarında devreye giren callback fonksiyonları.

- `vApplicationIdleHook()`
- `vApplicationTickHook()`
- `vApplicationStackOverflowHook()`
- `vApplicationMallocFailedHook()`



# Teoriden Pratiğe: Uygulamalı Projeler



**Çok Görevli LED Kontrolü**



**Seri Port İletişim Sistemi**



**Buton Debounce**



**Kaynak Yöneticisi**



**Hızlı Bildirim Sistemi**



**Sensör Okuma**



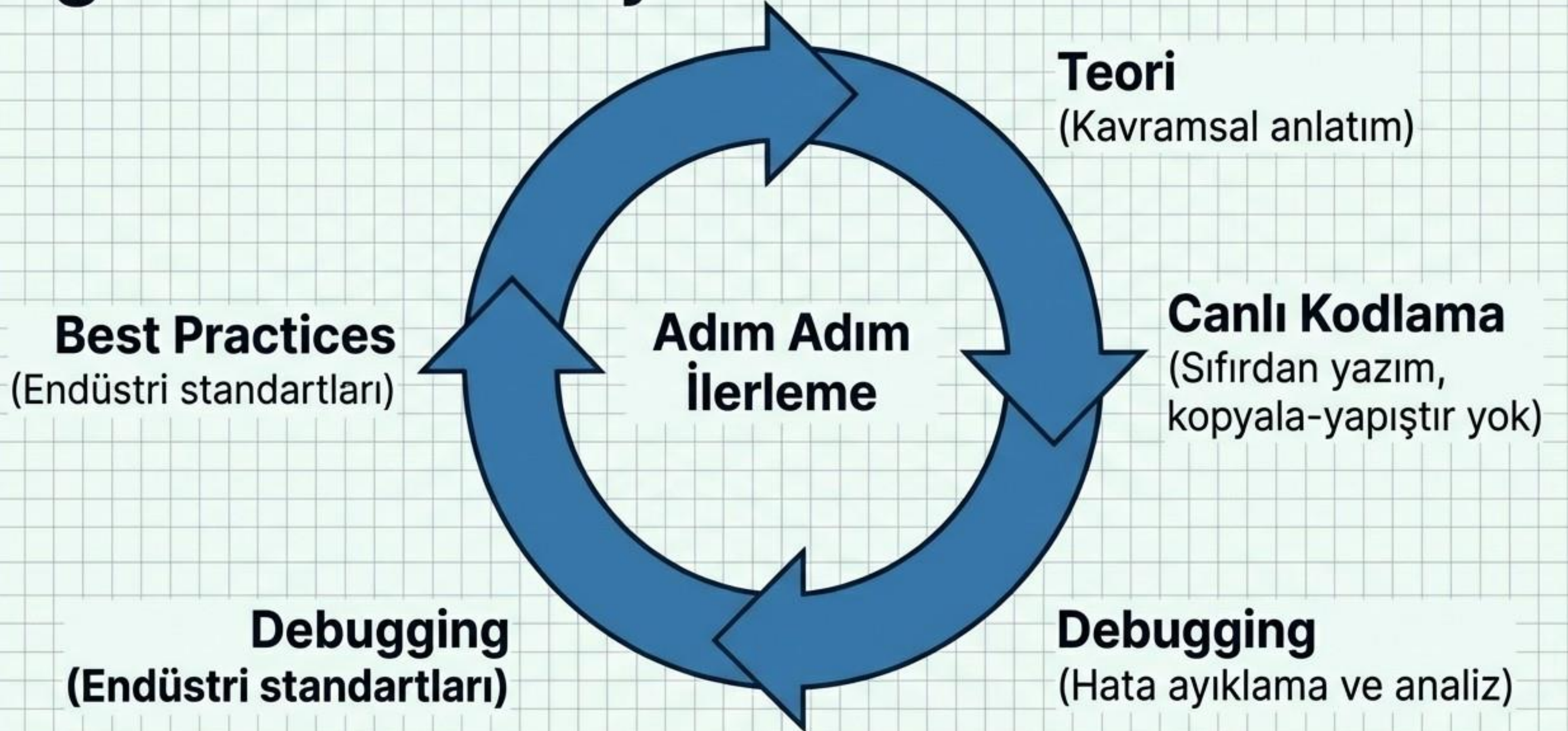
**Sistem İzleme Dashboard**



**Event Bazlı Kontrol**



# Eğitim Metodolojisi

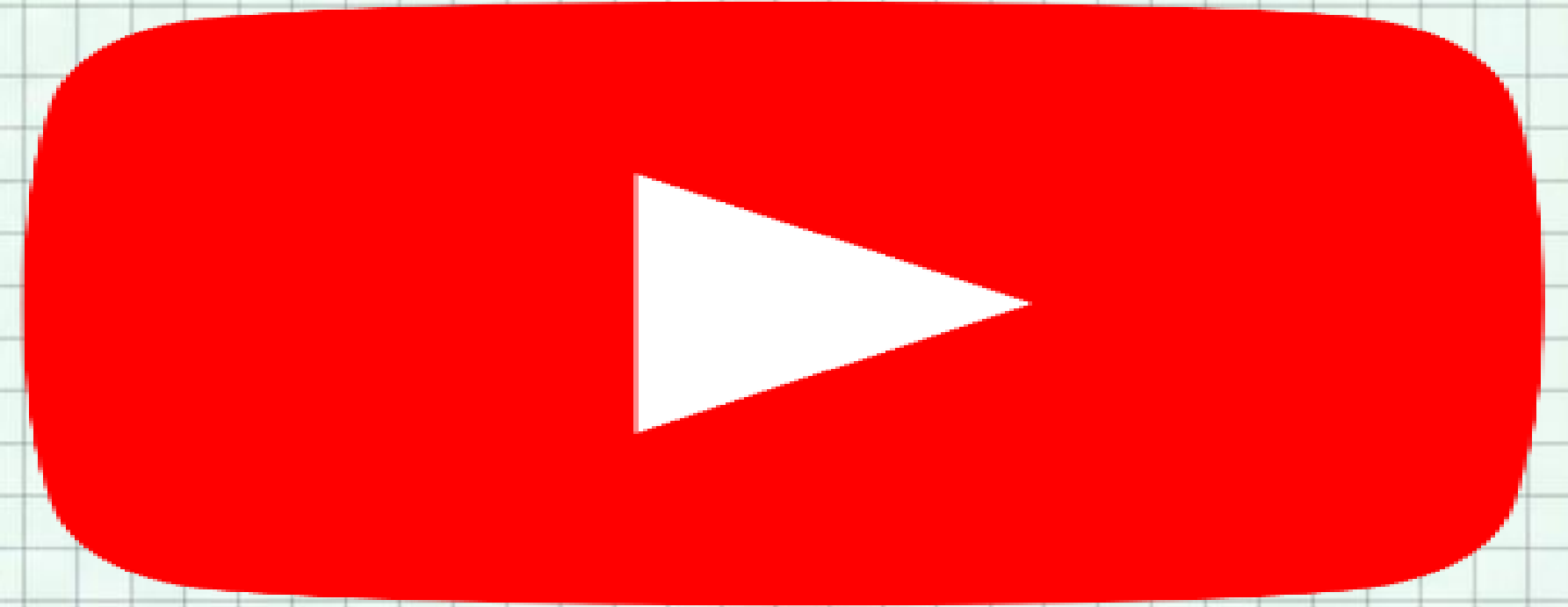




# Kaynaklar ve Destek Ekosistemi

## Dokümantasyon

- GitHub: Tüm kaynak kodlar
- PDF Sunum Notları
- STM32 Reference Manual
- FreeRTOS Developer's Guide



GitHub Repo





# Başlamaya Hazır Mısınız?

1. [ ] Donanımınızı hazırlayın.
2. [ ] STM32CubeIDE kurulumunu yapın.
3. [ ] Sonraki Adım: 'Geliştirme Ortamı Kurulumu' videosunu izleyin.

# Hadi Başlayalım!