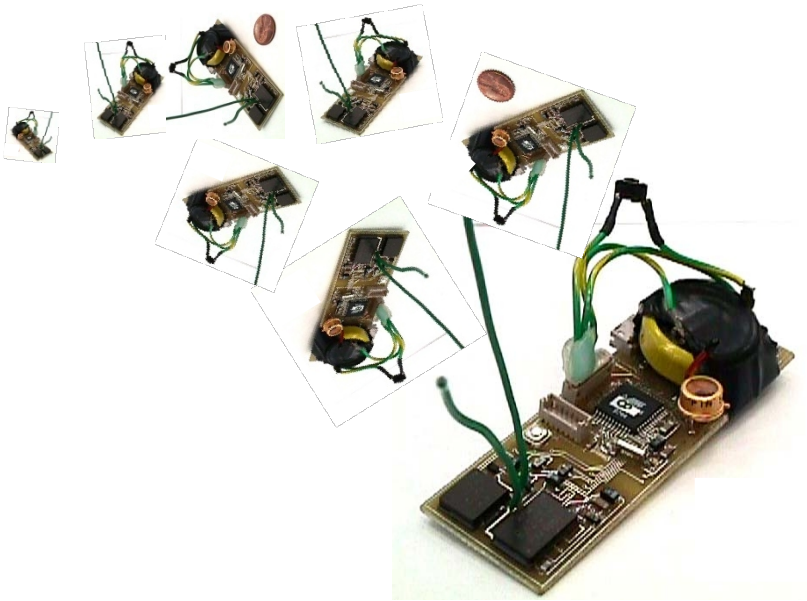
Kablosuz Algılayıcı Ağları İçin TinyOS İle Uygulama Geliştirme

Kasım Sinan YILDIRIM



AKADEMİK BİLİŞİM 2010

10 - 12 Şubat 2010

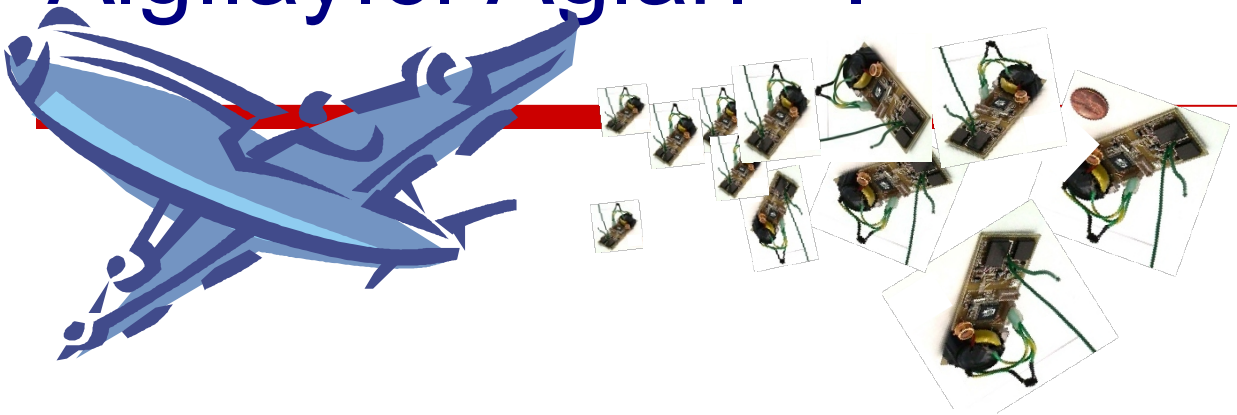
Muğla Üniversitesi, Muğla

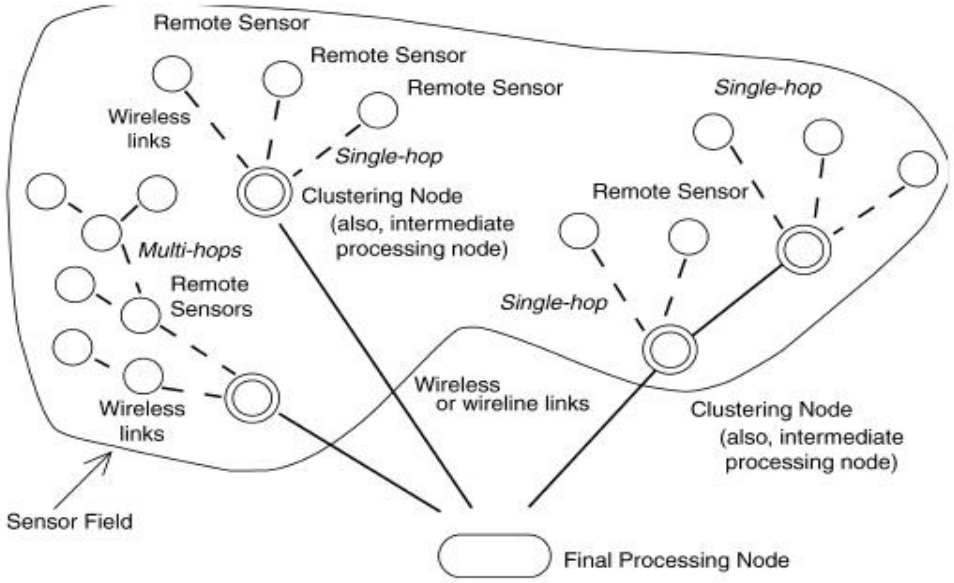
Ege Üniversitesi
Bilgisayar Mühendisliği Bölümü
İzmir, TÜRKİYE

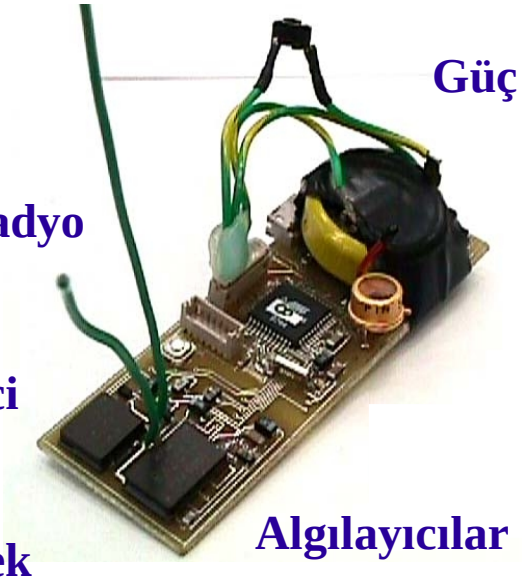
İçerik

- Algılayıcı Ağları
- TinyOS İşletim Sistemi
- Örnek Uygulama

Algılayıcı Ağları - I







Algılayıcı Ağları - II

- **Pil**
  - Güç tüketimi
    - İletişim donanımı
- **Kısıtlı kaynaklar**
  - Bellek
  - CPU
- **Genişleyebilirlik**
  - Çok sayıda düğüm
- **İşbirliği**

CPU	8 MHz TI Msp430
Memory	10 KB Ram 48 KB Flash
Radio	2.4 GHz 256 Kbps data rate
CPU Power	Sleep 0.1 microA Processing 2 microA
Radio Power	TX 18 microA RX 10 microA

TinyOS - I

- Gömülü, az güç kullanması gereken ve kablosuz iletişim yapan cihazlar için tasarlanmış bir işletim sistemi
- Açık kaynak kod
  - http://www.tinyos.net

TinyOS - II

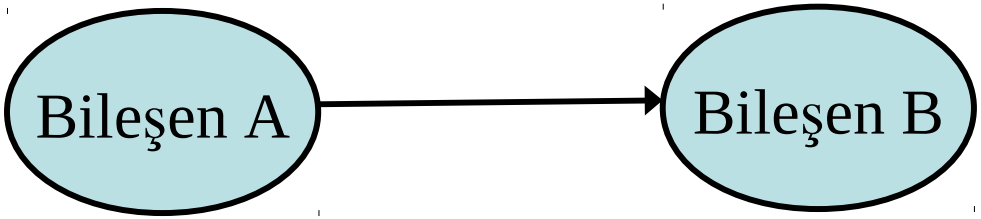
- **Modern** programlama kavramlarının gömülü sistemlerde kullanılmasını sağlar
  - **NesC** dili
- Faydalı birçok kütüphane içermektedir
- Açık kaynak kodlu ve **test edilmiş** birçok uygulama ve kütüphane TinyOS geliştiricileri tarafından paylaşılmaktadır.
- **Uygulama tabanlı** bir işletim sistemidir.

TinyOS - III

- TinyOs Uygulamaları **nesC** ile geliştirilir.
  - C diline yakın bir dil
  - Dinamik bellek kullanımı yok
  - Modüler yapı
- TinyOS'ta süreçler arası geçiş mekanizması (context switch) yoktur
  - Sonlana kadar çalış
  - **Tek yığıt**

Uygulama Temel Taşları - I

- **Bileşenler** (Components)
  - Temel nesC kod birimi
    - Yapılandırıcılar (Configuration)
    - Modüller (Module)
- **Arayüzler** (Interface)
  - Bileşenler arayüzlerle
    - Birbirlerine bağlanırlar
    - İletişim kurarlar



Uygulama Temel Taşları - II

- **Modül** (module) olarak isimlendirilen bileşenler bir gerçekleştirim barındırırlar.
  - Bir arayüz sunabilirler
  - Diğer modüllerin servislerini, onların sundukları arayüzler üzerinden kullanırlar.
- **Yapılandırıcı** (configuration)bileşenleri birbirlerine bağlarlar.
- **Arayüzler** (interface) bileşenlerin işlevselliğini belirlerler.
  - Tüm bileşenler ve arayüzlerin isimleri ile bunların gerçekleştirim dosyalarının isimleri aynı olmalıdır.

Örnek Uygulama

- Bir algılayıcı düğümü açıldığı anda o düğüme ait ledleri yakan bir uygulama
- İhtiyaçlar
  - **Led**'ler kullanılmalı (yak / söndür)
  - Sistemin açıldığından (**boot**) haberdar olunmalı

Powerup Modülü

```
module PowerupC {
   uses interface Boot ;
   uses interface Leds ;
}
implementation {
   event void Boot.booted () {
      call Leds.led0On();
   }
}
```

Boot ve Leds Arayüzleri

```
interface Boot {
  event void booted ();
}

interface Leds {
  command void led0On();
  command void led0Off();
  command void led0Toggle();
  ...
}
```

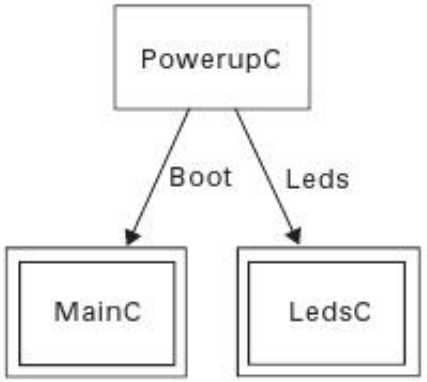
LedsC ve MainC Modülleri

```
configuration LedsC {
  provides interface Leds;
}
implementation {
   …
}
```

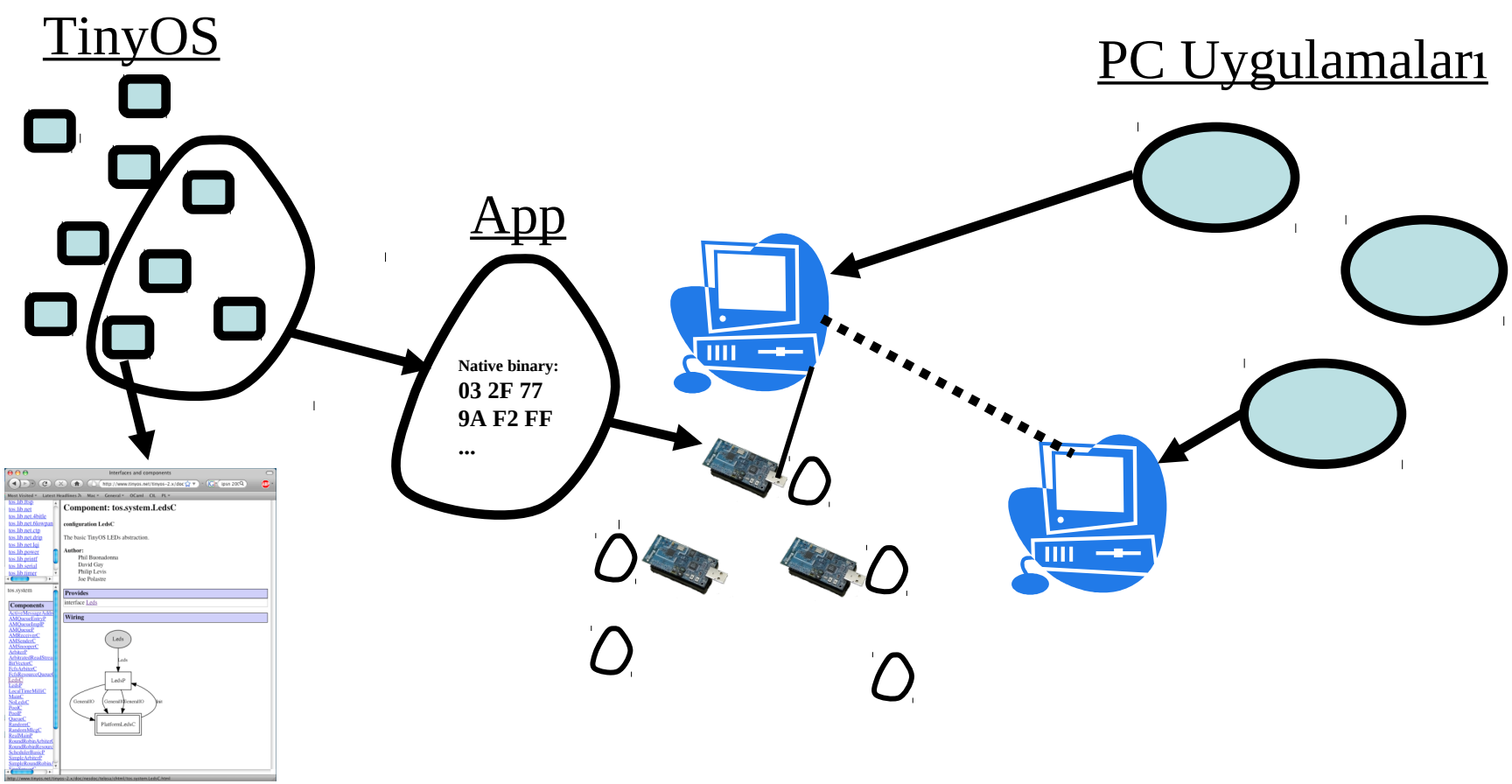
```
configuration MainC {
  provides interface Boot;
  …
}
implementation {
  …
}
```

PowerupApp Yapılandırıcısı

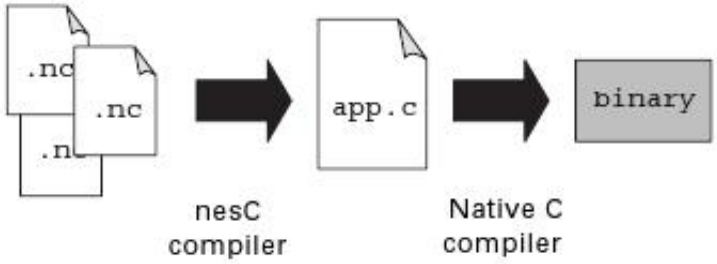
```
configuration PowerupAppC {

  implementation {
    components MainC , LedsC , PowerupC

    MainC.Boot -> PowerupC.Boot ;
    PowerupC.Leds -> LedsC.Leds ;
  }
}
```

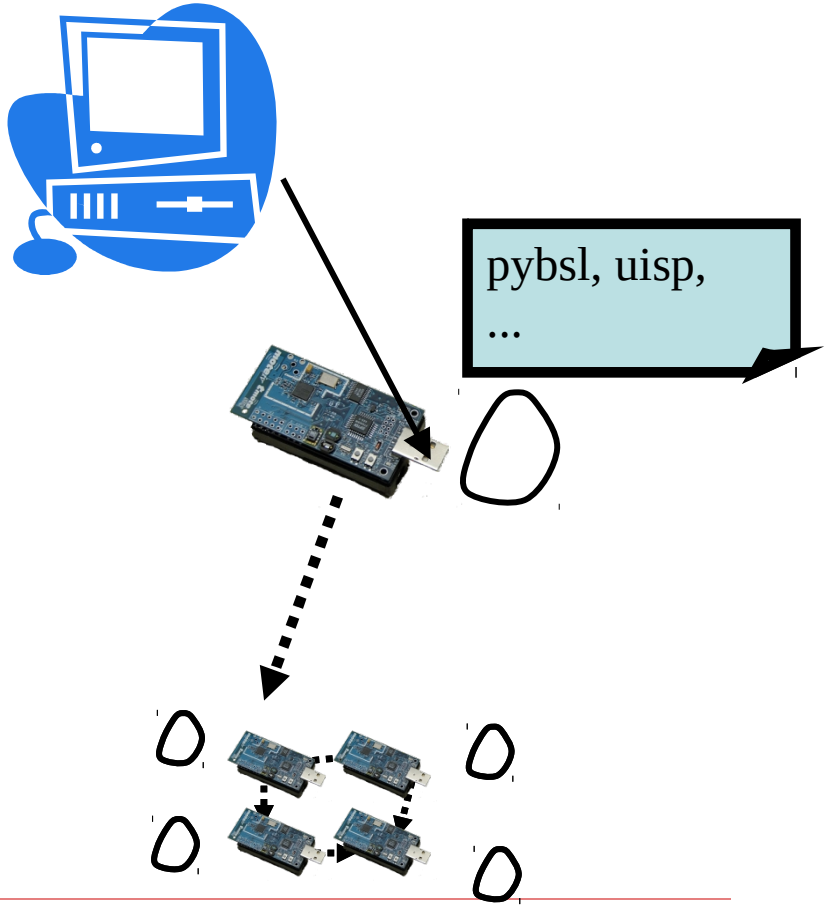


Derleme / Yükleme - I

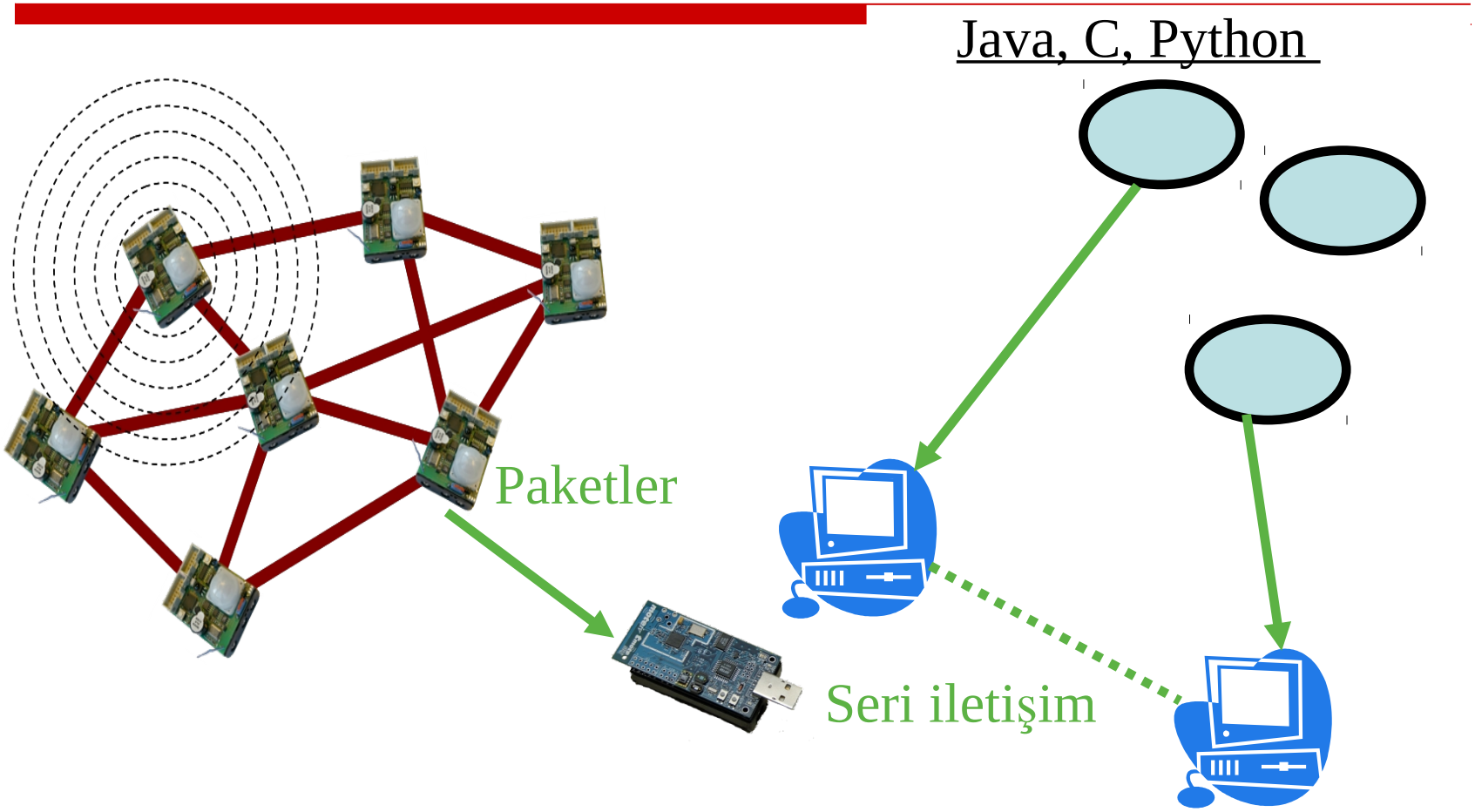


Derleme / Yükleme - II





Derleme / Yükleme - III



Sorular?