Kablosuz Sensör Ağları Raporu

İsim Ve Soyisim: Mehmed Emre AKDİN

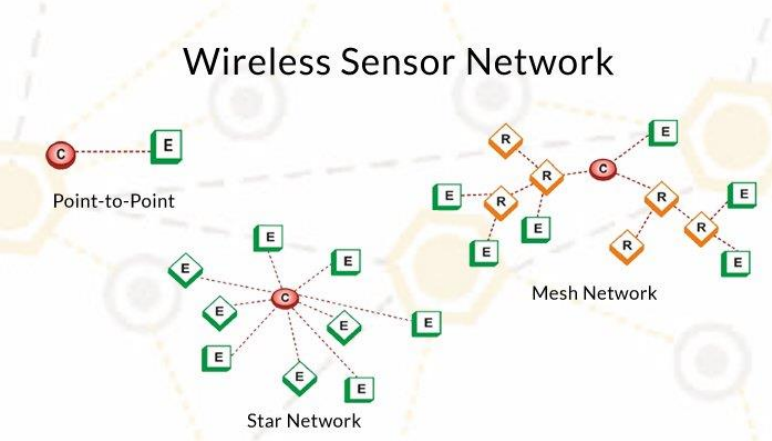
Numara: 374684

Öğretim: 2

KABLOSUZ SENSÖR AĞLARI

Sürekli gelişmekte olan sensör düğümleri çevrelerinde meydana gelen değişiklikleri algılayabilir (sensing), elde ettikleri veriyi işleyebilir (processing) ve haberleşebilirler (communicating). Birbirleriyle işbirliği içerisinde çalışan sensör düğümleri kablosuz sensör ağlarını meydana getirirler.

Örnek sensör ağları;



Bu sensör ağlarının amacı bir takım verileri algılamak ve algılanan veriyi kurulu ağ üzerinden işbirliği içerisinde ana bir merkeze iletmektir. Bu ağlar çeşitli uygulama alanlarına sahiptir. Askeri,sağlık ve endüstriyel gibi vb. alanları vardır. Bu ağları daha iyi anlayabilmek için en çok bilinen iki örnek aşağıda verilmiştir

Sniper Algılama Sistemi: Askeri alanda kullanılmak üzere geliştirilmiş bu sistemin amacı savaş alanında keskin nişancının yerini tam olarak tespit etmektir. Bu sistem üç ana bileşene sahiptir:

- Coğrafi olarak dağılmış veya coğrafi olarak dağılmış bir dizi mikrofon veya sensör
- İşlem birimi
- Silah sesleri uyarılarını gösteren bir kullanıcı arayüzü

Aşağıda bir sniper algılama sisteminin görüntüsüne yer verilmiştir;

Hasta İzleme Sistemi: Sağlık alanında kullanılan bu sistem, giyilebilir sensörler aracılığıyla hastanın EKG ve EMG gibi değerlerini ölçmeyi amaçlamaktadır. Bu sistem ile hastanın çeşitli değerleri periyodik olarak ölçülebilmektedir.

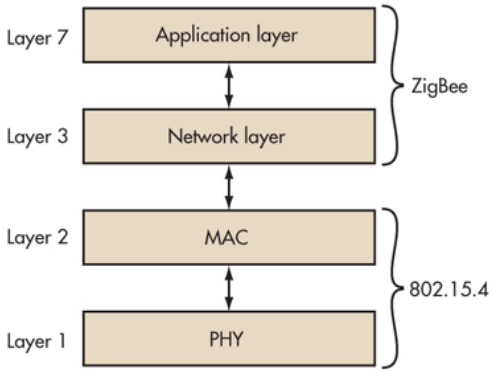
Aşağıda çeşitli alanlarda ölçüm yapan sensörleri olan cihazlar gösterilmiştir;

KABLOSUZ SENSÖR AĞLAR VE ZIGBEE PROTOKOL YIĞINI

Bu bölümde kablosuz sensör düğümlerinin katmanları ve protokol yığınları kısaca incelenecek ve haberleşme standartları anlatılacaktır. Bir kablosuz sensör ağı düğümü aşağıdaki katmanlı protokol yığınını barındırır;

- Physical
- Data Link
- Network
- Transport
- Application

ZigBee, IEEE 802.15.4 standardı ile tanımlanan ve kişisel alan ağında haberleşmeyi sağlayan bir standarttır. Bundan dolayı ZigBee ve 802.15.4 kavramları karıştırılabilir. Aşağıda görüldüğü her ikiside farklı katmanları tanımlayan farklı standartlardır;



Physical Layer: Frekans seçimi, taşıyıcı frekans üretimi, gelen sinyali algılama, modülasyon ve bit düzeyinde veri şifrelemekle görevlidir.

Data Link Layer : Veri akışını çoğullama (multiplexing), ortam erişim kontrolü (MAC) ve hata kontrolüyle görevlidir.

Network Layer : WSN düğümlerini adresleme, yönlendirme (routing) ve tıkanıklık kontrolüyle görevlidir. Normalde ZigBee IP adresiyle uyumlu değildir. Çünkü IPv6 başlık ve adres alanları 40 byte iken ZigBee toplam paketin 127 byte olmasına izin verir. Bundan dolayı ZigBee cihazları internete açılamaz ve sunucu veya web tabanlı cihazlarla haberleşemezler. Bu problemi ortadan kaldırmak ve ZigBee cihazlarını IP adresiyle uyumlu hale getirmek için IETF (Internet Engineering Task Force) 6LowPAN standardını geliştirmiştir.

Transport Layer : Datagramların güvenli olarak iletimiyle görevlidir. TCP protokolü güvenli bir taşıma katmanı protokolü olmasına rağmen WSN için uygun bir protokol değildir. Çünkü WSN düğümleri için harcanan güç çok önemlidir ve kısıtlı güce ve belleğe sahip olan WSN düğümleri ACK mesajlarıyla fazla enerji harcayacaktır. Bundan dolayı güvenli bir UDP protokolü WSN için daha uygundur.

Application Layer : Kullanıcı ihtiyaçlarına göre tasarlanmış protokolleri içeren katmandır. Örneğin, CoAP, WSN düğümleri için geliştirilmiş bir web transfer protokolüdür.

Contiki İşletim Sistemi Nedir?

2002 yılında Adam Dunkels yönetiminde C programlama diliyle geliştirilmiş açık kaynaklı bir işletim sistemidir. Contiki işletim sistemi, **kısıtlı hafızaya** ve **düşük güce** sahip olan nesnelerin interneti cihazları ve WSN düğümleri için geliştirilmiştir. Contiki IPv4 ve IPv6 üzerinden haberleşmeyi sağlayabilmektedir

Contiki İşletim Sisteminde Prosesler

Contiki işletim sistemi process'lerden oluşmaktadır. Temel olarak bir process, contiki işletim sistemi tarafından çalıştırılan kod bloklarından ibarettir. Contiki prosesleri, sistem yüklenince yada proses içeren bir modül sisteme yüklendiğinde başlatılır. Contiki işletim sistemi için yazılan kodlar iki şekilde çalışır. Bunlar;

- **Cooperative Kod:** Sırayla birbiri ardınca çalışan kodlardır.
- **Preemptive Kod:** Bu kodlar önceliğe sahiptirler. Ve herhangi bir zamanda cooperative kodun çalışmasını durdurabilirler. Preemptive kod bittiğinde cooperative kod kaldığı yerden devam eder.

Bir contiki prosesi iki kısımdan oluşur. Bunlar;

- **Proses Kontrol Bloğu:** RAM'de tutulurlar. Process ile ilgili genel bilgileri içerir. Bunlar;

```
struct process {
struct process *next;
const char *name;
int (* thread)(struct pt *, process_event_t, process_data_t);
struct pt pt;
unsigned char state, needspoll;
};
```

- **Proses Thread:** Process'in asıl kodlarını içeren kısımdır.

```
PROCESS_THREAD(hello_world_process, ev, data) {
PROCESS_BEGIN();
 printf("Hello, world\n");
 PROCESS_END();
}
```

Kablosuz sensör düğümlerinde çalışması 3 aşama olarak düşünülebilir;

1-) Algılama(Sensing): Sensörler vasıtasıyla fiziksel büyüklükler algılanır ve sayısal değer olarak iletilir.

2-) Bilgi İşleme(Processing): Contiki vb. kısıtlı hazıfa ve düşük güç gerektiren işletim sistemlerine yazılan yazılımlarla algılanan veriler işlenmektedir.

3-) Haberleşme: Sensör düğümü(mote) ile diğer sensör düğümlerinin iş birliği içerisinde çalışarak verileri ana bir merkeze iletir.

- Physical , Data Link, Network, Transport, Application katman yardımıyla haberleşme sağlanır.
- Kullanılan standartlar IEEE 802.15.4 ve onun üzerinde çalışan ZigBee protokolüdür.

Deneyde Gerçekleştirilen Uygulamalar ve Deneyimlerim:

Öncelikle deneyde kullanılan programlardan bahsedelim:

VMware Workstation Player: Kullanıcıların tek bir fiziksel makinede sanal makinalar kurmalarını ve gerçek makine ile aynı anda kullanmalarını sağlar.

Contiki İşletim Sistemi: Düşük güçlü kablosuz nesnelerin interneti içi,cihazlarına odaklanır. Ağa bağlı, bellek kısıtlamalı sistemler için kullanılan bir işletim sistemidir.

Cojaa Simulator: Kablosuz sensör ağları için özel olarak tasarlanmış bir network simülasyonudur.

Mote nedir?

Mote'lar derlenmiş ve çalışan bir contiki sistemidir. Biz uygulamada skymote kullanacağız skymote'lar çok düşük güç ile çalışabilen modüllerdir.

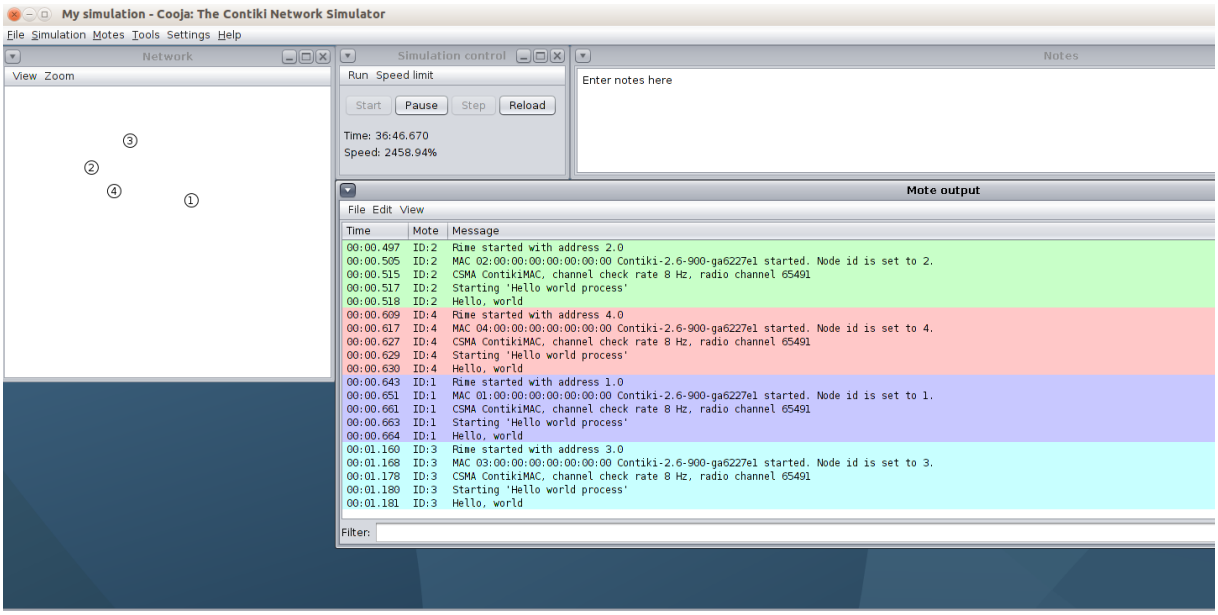
Uygulama -1

Bu uygulamada Her bir mote'un ekrana "Hello World" yazması sağlanacaktır.

Kaynak kodun bir kısmı :

```
PROCESS_THREAD(hello_world_process, ev, data){
  PROCESS_BEGIN();  //Ana process üzerinden bir process başlatılır
  printf("Hello, world\n"); //Her bir mote'un ekrana "Hello, world" yazması sağlanır
  PROCESS_END();
}
```

Ekran Görüntüsü:



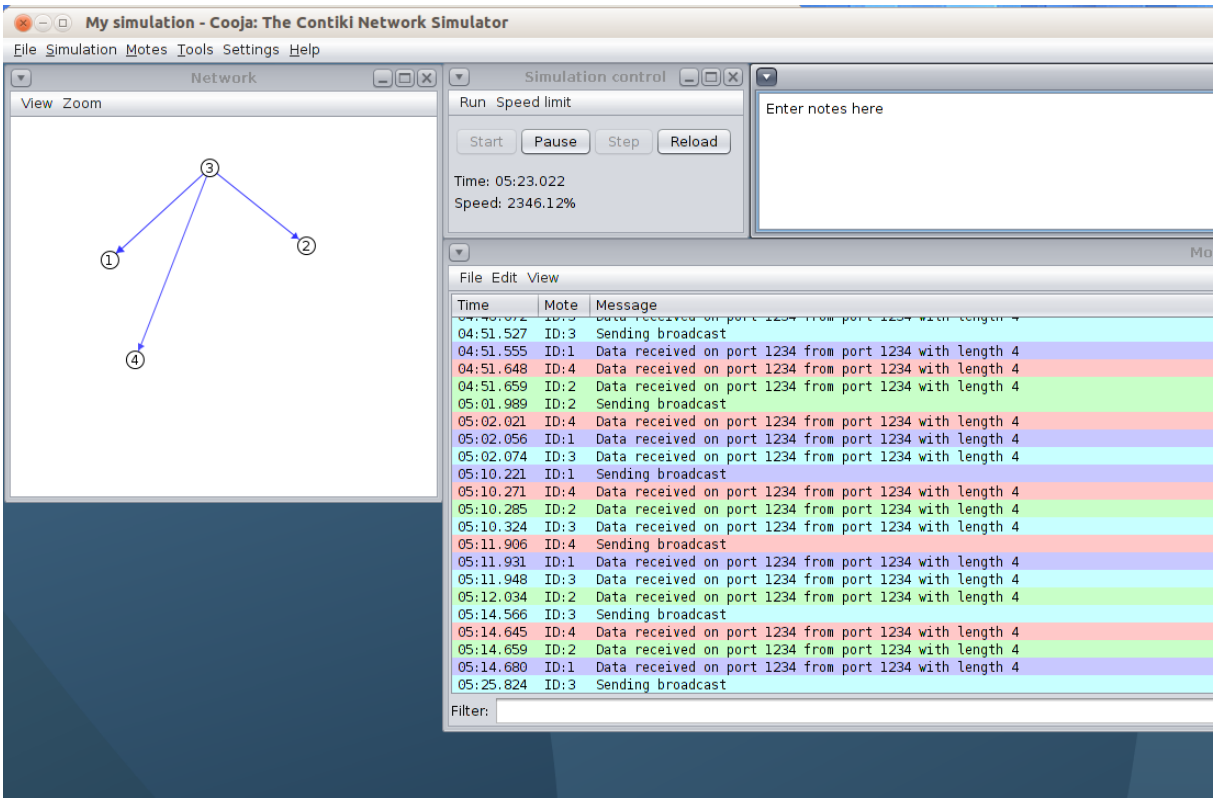
Uygulama -2

Bu uygulamada tüm mote'lar diğer mote'lara paket gönderecektir. Yani tüm mote'lar broadcast yapacaktır.

Kodun bir Kısmı:

```
static void
receiver(struct simple_udp_connection *c,   //Düğümler receiver sayesinde verileri alır ve ekrana yazdırır. Bu işlem sürekli tekrarlanır.
        const uip_ipaddr_t *sender_addr,
        uint16_t sender_port,
        const uip_ipaddr_t *receiver_addr,
        uint16_t receiver_port,
        const uint8_t *data,
        uint16_t datalen)
{
  printf("Data received on port %d from port %d with length %d\n",
        receiver_port, sender_port, datalen);
}
PROCESS_THREAD(broadcast_example_process, ev, data)
{
  static struct etimer periodic_timer;
  static struct etimer send_timer;
  uip_ipaddr_t addr;
  PROCESS_BEGIN();
  simple_udp_register(&broadcast_connection, UDP_PORT,NULL, UDP_PORT,receiver);  //Bu fonksiyon bir UDP bağlantısını kaydeder. İki tip
  geri dönüş değeri vardır. Eğer  ise hiçbir UDP bağlantısı tahsis edilmez ise geri dönüş değeri sıfırdır. Bağlantı başarıyla tahsis edilirse geri
  dönüş değeri birdir.
  etimer_set(&periodic_timer, SEND_INTERVAL);
  while(1) {
  PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&periodic_timer));
  etimer_reset(&periodic_timer); //Bu fonksiyon etimer_set() fonksiyonuyla event timer'a verilen aralık ile event timer'ı sıfırlar
  etimer_set(&send_timer, SEND_TIME);  //Gelecekteki bir süre için olay zamanlamada kullanılır.
  PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&send_timer)); //Bu sistem çağrısı bir olayı beklemek için kullanılır. Aldığı argüman FALSE
olarak değerlendirilirse, process direk olarak diğer evente geçer.
    printf("Sending broadcast\n");
    uip_create_linklocal_allnodes_mcast(&addr);
    simple_udp_sendto(&broadcast_connection, "Test", 4, &addr);   //Bu fonksiyon, belirtilen bir IP adresine UDP paketi gönderir. Paket,
    simple_udp_sendto() ile kaydedildiğinde belirtilen UDP paketi bağlantı noktaları yardımıyla  gönderilir.
  }
  PROCESS_END(); }
```

Ekran Görüntüsü:



Uygulama -3

Bu uygulama server-client uygulamasıdır.

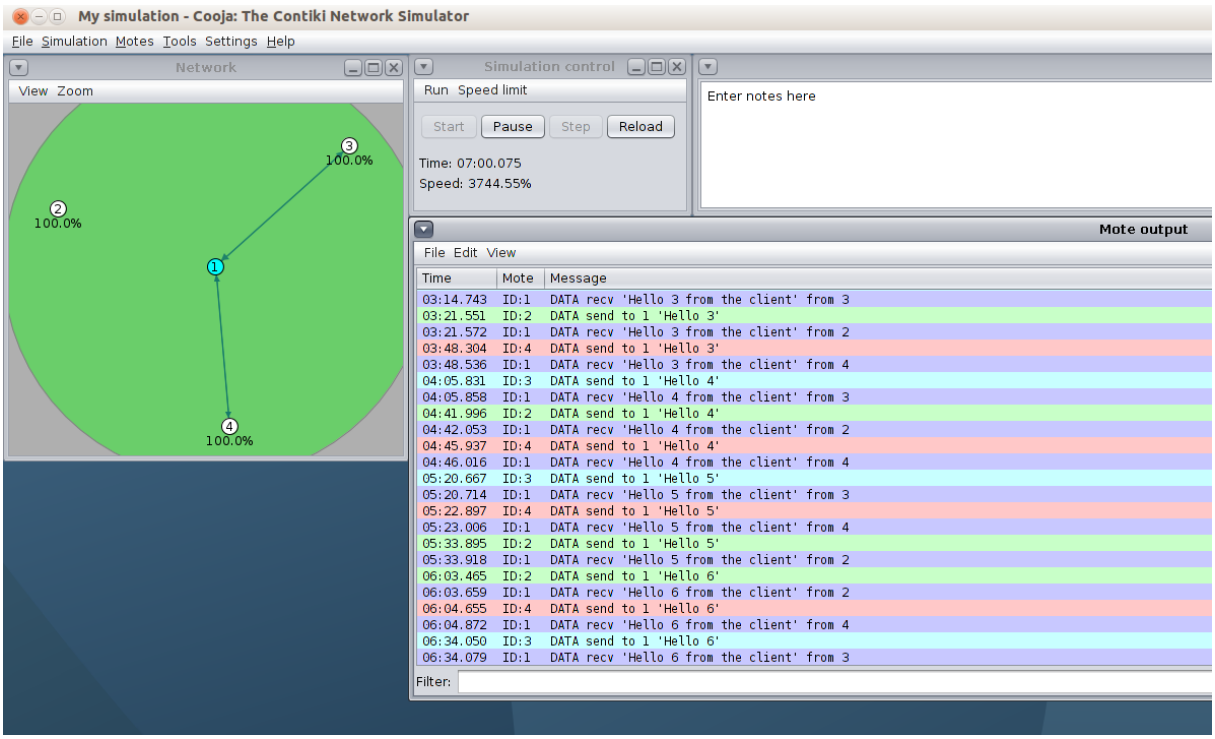
Client kısmının kaynak kodunun bir kısmı:

```
static void send_packet(void *ptr)
{
  static int seq_id; // ”Hello” mesajının kaçıncı mesaj olduğunu belirtmek için bir id değeri tanımlanmış
  char buf[MAX_PAYLOAD_LEN]; //Bu buffer’a mesaj yazılıp servera yollanır.
  seq_id++; // id değeri her mesajda bir arttırılmış
  PRINTF("DATA send to %d 'Hello %d'\n",server_ipaddr.u8[sizeof(server_ipaddr.u8) - 1], seq_id);
  sprintf(buf, "Hello %d from the client", seq_id);
  uip_udp_packet_sendto(client_conn, buf, strlen(buf),&server_ipaddr, UIP_HTONS(UDP_SERVER_PORT)); //Servera yollama işlemi yapılır
}
```

Server kısmının kaynak kodunun bir kısmı:

```
static void tcpip_handler(void)  // Servera mesaj geldiğinde bu fonksiyon çalışır.
{
 char *appdata;
 if(uip_newdata()) {  //Karakter dizisi varsa true döner
   appdata = (char *)uip_appdata; //Gelen mesaj appdataya cast edilir
   appdata[uip_datalen()] = 0; //Mesajın son karakteri sıfıra eşitlenir.
   PRINTF("DATA recv '%s' from ", appdata); // Alınan veri içindeki mesaj ekrana yazılır.
   PRINTF("%d",UIP_IP_BUF->srcipaddr.u8[sizeof(UIP_IP_BUF->srcipaddr.u8) - 1]); // Mesajın kim geldiği ekrana yazılır.
   PRINTF("\n");
#if SERVER_REPLY  //Eğer sever cevap verecekse bu kısım çalışır.
   PRINTF("DATA sending reply\n");
   uip_ipaddr_copy(&server_conn->ripaddr, &UIP_IP_BUF->srcipaddr);
   uip_udp_packet_send(server_conn, "Reply", sizeof("Reply"));
   uip_create_unspecified(&server_conn->ripaddr);
#endif
 }
}
```

Ekran Görüntüsü:



DENEY VİDEOSU:
https://drive.google.com/file/d/1NMWp8waE1vJAkOFXf_EzJl1VWKnQHADd/view?usp=sharing