İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

KABLOSUZ DUYARGA AĞLARINDA VERİ BİRLESTİRİLMESİ VE DEĞERLENDİRİLMESİ

YÜKSEK LİSANS TEZİ

Serkan ERBORAL

Anabilim Dalı: Mekatronik Mühendisliği

Programı: Mekatronik Mühendisliği

EYLÜL 2008

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

KABLOSUZ DUYARGA AĞLARINDA VERİ BİRLEŞTİRİLMESİ VE DEĞERLENDİRİLMESİ

YÜKSEK LİSANS TEZİ

Serkan ERBORAL

Tezin Enstitüye Verildiği Tarih : 15 Eylül 2008

Tez Danışmanı : Prof.Dr. Metin GÖKAŞAN
Diğer Jüri Üyeleri Prof.Dr. Hakan TEMELTAŞ
Prof.Dr. Sema OKTUĞ

EYLÜL 2008

ÖNSÖZ

Akademik hayatımın ilk aşamasında bana böyle güzel bir çalışma konusunu sunan ve gerekli materyalleri sağlayan değerli danışmanım Prof. Dr. Metin GÖKAŞAN'a en içten teşekkürlerimi sunarım.

Çalışmalarım boyunca, bana maddi destek sağlayan TÜBİTAK Bilim İnsanı Destekleme Daire Başkanlığına teşekkür etmeyi borç bilirim.

Çalışmanın başından sonuna kadar bana yol gösteren doktora öğrencisi Ahmet KUZU'ya teşekkür ederim.

Son olarak, bu çalışmanın yoğunluğunu benimle birlikte paylaşan aileme ve burada adını yazamadığım manevi destek olan tüm arkadaşlarıma SONSUZ teşekkür ederim.

Serkan ERBORAL

Mayıs, 2008

İÇİNDEKİLER

KISALTMALAR

NesC	: Network embedded systems C, ağlarda gömülü sistemler için C
ARM	: Advanced RISC machine
TX	: Gönderme
RX	: Alma
FIFO	: İlk giren ilk çıkar
I2C	: Inter-integrated circuit
UART	: Universal asynchronous receiver/transmitter
SDIO	: Secure digital input and output
AC97	: Audio codec 1997
I2S	: Inter-IC sound
SMA	: Subminiature version A – koaksiyel kablolar için A sürüm bağlayıcı
AES	: Advanced encryption standard
SDRAM	: eş zamanlı dinamik rasgele erişimli bellek, (synchronous dynamic random access memory
USB	: Evrensel seri veriyolu, (Universal Serial Bus)
MMX	: Matrix math extensions
IEEE	: Institute of electrical and electronics engineers
MAC	: Media access control
ADC	: Analog -sayısal çevirici, (Analog to digital converter)
SPI	: Serial peripheral interface bus
IP	: Internet protokolü
PDA	: Avuç içi bilgisayar, (personal digital assistant)

TABLO LİSTESİ

ŞEKİL LİSTESİ

Sayfa No:

KABLOSUZ DUYARGA AĞLARINDA VERİ BİRLEŞTİRİLMESİ VE DEĞERLENDİRİLMESİ

ÖZET

Kablosuz duyarga ağları, teknolojinin gelişmesine paralel olarak ucuzlayan ve küçülen mikroişlemcilerin, bunların çevre birimlerinin ve duyargaların birleştirilerek çeşitli alanlarda kullanım alanı bulmasıyla günümüzde önemli bir araştırma ve uygulama alanı olmuştur. Çok fazla miktarda duyargadan gelen verilerin gerek merkezde gerek yerel olarak işlenmesi ve sınıflandırılması gereklidir.

Bu çalışmada, kablosuz duyarga ağları kullanılarak, belirli bir bölgedeki canlı veya nesne takip edilmesi amaçlanmıştır. İlk olarak kablosuz duyarga ağlarının donanım ve yazılım özellikleri incelenmiş ve uygulama alanlarına göre donanım ihtiyaçları belirtilmiştir. Sonraki bölümde, kablosuz duyarga ağları için tasarlanmış nesneye yönelik programlama mimarisine sahip TinyOS işletim sistemi incelenmiştir. Uygulamanın gerçeklemesi için gerekli olan algılayıcılar (ışık, ivme ve ses) belirlenmiştir. Kablosuz duyarga düğümü Imote2'de gerekli bütün algılayıcılar düğüm üzerinde bulunmadığından ilave uyumlaştırıcı devreler tasarlanarak, düğüme ek bir kart olarak eklenmiştir. İşletim sistemi TinyOS'in bu donanımları kullanması için gerekli sürücüler de NesC programlama dili ile yazılmıştır. Son kısımda, geliştirilen kart ve korelasyon ile işlenecek veriler için gerekli olan düğümler arası senkronizasyon algoritması test edilmiştir. Kablosuz duyarga ağı kurularak, ışık, ses ve titreşim kaynağı özelliklerinden bir veya birkaçını bulunduran bir nesnenin tespit ve takip edilmesi için algılayıcılardan veriler kablosuz olarak toplanmış ve Kalman filtresi kullanılarak veri birleştirilmesi yapılmıştır. Elde edilen sonuçlar, sonuçlar ve tartışma bölümünde tartışılmıştır.

WIRELESS SENSOR NETWORKS BASED SENSOR FUSION

SUMMARY

The applications in Wireless Sensor Networks has been a challenging research area for almost 10 years by the help of developing technology which makes the microcontrollers, integrated circuits and sensor more cheaper and smaller. Many algorithms derived from numerous disciplines have been proposed and tested against different scenarios, especially processing of the collected sensor data.

In this study it is aimed to sense living or non living objects in an area, using wireless sensor networks. First the hardware and software properties of wireless network sensors are studied and the needed specifications are defined for application purposes. In the later section, the TinyOS operation system, which is designed by using object orientated programming architecture for wireless network sensors, is examined. The sensors needed for the application (light, acceleration, sound) to be active are defined since some needed sensors are not present on the wireless sensor node, the necessary additional converters and filters are designed as an additional card and attached to the node. For the operating system (TinyOS) to be able to use this hardware system, needed drivers are coded with using NesC programming language. In the last part the developed card and the synchronization algorithm which is going to be used to process the data with correlations is tested.

A wireless sensor network is created. For the detection and following up an object having at least one of the properties, (light, sound or vibration source) the data is collected through a wireless network and those data are combined by using Kalman Filter. At the end, the results are presented on the last section.

1. GİRİŞ

Telsiz Duyarga ağları, sıcaklık, ses, titreşim, basınç, hareket ve kirlilik gibi çevresel değişimleri izlemek için duyargaların birlikte hareket ederek belirli bir görevi başarabildikleri bir çeşit kablosuz ağdır. Bu ağda, küçük boyutlarda, otonom, genelde pil ile çalışan duyarga düğümleri bulunur. Her bir düğüm, algılama, veri işleme ve haberleşme özelliklerine sahiptir.

Günümüz teknolojisinin yardımıyla, kısa mesafelerde kablosuz ortamlarda birbirleriyle haberleşebilen, algılama ve işlem yeteneğine sahip duyargaların geliştirildiği ve bunların maliyetlerinin düşük, güç ihtiyacının az fakat fonksiyonlarının fazla olması araştırmacılar için yeni olanaklar doğurmuştur. Bir nesne veya olayın algılanması için, haberleşme, algılama ve işlem gücüne haiz bu kablosuz duyargalardan yüzlercesinin birbirleriyle müşterek çalışmasını sağlamak mümkün olmaktadır.

Bu çalışmalar için en önemli ihtiyaçlardan biri güçtür. Duyargaların çalışması için, kapasiteleri sınırlı pillerle depolanan enerjiden faydalanılır ve bu enerji kaynaklarının değiştirilmesi olanağı her zaman mümkün değildir. Güç kaynaklarının ebatlarının küçültülüp, kapasitelerinin arttırılması ve duyarga ağının çalışma ömrünün uzatacak özellikle güç tüketimi düşük tekniklerin geliştirilmesi üzerinde durulmaktadır.

Kablosuz duyarga ağlarının bir önemli özellikleri de birbirleriye yardımlaşarak çalışabilmeleridir. Ebatları çok küçük ($1cm^3$ gibi) olduğundan hedeflenen bölgelere gelişi güzel atılabilirler ve kendi aralarında haberleşerek, elde etmiş oldukları verileri kısmen işleyerek merkezi sistemi gerekli görülüyorsa bilgilendirebilirler. Duyargalardan gelen bilgilerin merkezi sisteme taşınabilmesi için iletişim ağının daha önceden özenle planlanması gereklidir.

Bu çalışmanın ilk bölümlerinde kablosuz duyarga ağları tanıtılmış özellikleri hakkında bilgi verilmiştir. Tasarım ve geliştirilmelerinde uygulama bağımlı sistemler oldukları için, özellikler anlatılırken uygulama alanlarından da bahsedilmiştir. Uygulama

alanları ve her bir düğümün nasıl etkinleştirilmesi gerektiği arasındaki bağlantılar tartışılmıştır.

Çalışmanın son bölümü, kablosuz duyarga ağları kullanılarak belirli bir bölgede hareket eden bir nesnenin, ışık yayma, ses çıkarma, titreşim yaratma gibi özellikleri algılanarak yerinin tespit edilmesi ve hareket yörüngesinin bulunması amaçlanmaktadır. Bu amaç doğrultusunda yapılan çalışmalar ve deneylerden bahsedilmiş, deney sonuçları sonuç bölümünde tartışılmıştır. Duyarga ağlarının çok geniş bir çalışma alanı oluşturduğu göz önüne alınırsa, haberleşme algoritmalarının etkinleştirilmesi, güç tüketiminin minimize edilip ağın çalışma ömrünün uzatılması gibi konular bu çalışmanın ilgi odağında değildirler.

2. KABLOSUZ DUYARGA AĞLARI VE GENEL ÖZELLİKLERİ

Kablosuz duyarga ağlarını oluşturan her bir düğüm, uygulamaya konulacak olan projenin özelliğine göre geliştirilir. Bu nedenle kablosuz duyarga ağlarının özelliklerini incelerken uygulama alanlarını da göz önüne almak gereklidir.

Düğümlerde bulunan her bir özellik için hem enerjiye hem de iletişim ağına ihtiyaç vardır. Düğümler içinde yer alan fakat projede kullanım amacı olmayan her özellik düğümde karışıklığa sebep olacak, enerji tüketimini arttıracak ve güvenilirlik katsayısını düşürecektir.

Duyarga ağlarının uzun süre tek başına çalıştığı ve çıkabilecek bir sorunda servis sağlanabilmesinin güç olduğu düşünüldüğünde, güvenilirlik olgusunun ne kadar önemli olduğu ortaya çıkar. Bu nedenle, düğümlerin sahip olacakları özellikler, direkt olarak uygulama alanlarına bağlıdır.

Kablosuz duyarga ağlarının belirgin özellikleri önem sırasına göre;

- Çalışma ömrü
- Kapsama alanı
- Güvenilirliği
- Geliştirme ve üretim maliyetleri
- Cevap süresi
- Zamanla doğruluğu
- Güvenliği
- Etkin örnekleme oranıdır.

Bu özellikler aşağıda ayrı ayrı incelenecektir. Ancak bilinmelidir ki, bu özelliklerin birbirleriyle sıkı bir bağı bulunmaktadır. Mesela, örnekleme oranının düşürülmesi, düğümün çalışma ömrünü önemli ölçüde arttırmaktadır.

Bu bölümün amacı, bu özellikleri ve bu özellikler arasındaki bağları incelemektir. Yapılan çalışma, uygulama alanının ilk örneği olduğu için, bu çalışmada bu özelliklere göre düğümün etkinleştirilmesi yapılmamıştır. Fakat hedefe ulaşılıp ulaşılamayacağı düğüm üzerinde olması gereken özellikler düşünülerek sonuç bölümünde tartışılmıştır.

2.1 Çalışma Ömrü

Her kablosuz duyarga ağının en önemli ve en belirgin özelliği beklenen/hesaplanan çalışma ömrüdür. Çalışma ömrünü kısıtlayan en önemli etken de enerji kaynağının sınırlı olmasıdır. Kablosuz duyarga ağının çalışma ömrü, ağdaki bir düğümün çalışma ömrü kadardır. Bu yüzden düğümler geliştirilirken, her bir düğüm kendi enerji kaynağını, ağın ömrünü maksimum yapacak şekilde kullanır.

Birçok uygulamada, kablosuz duyarga ağının ömrü, düğümlerin ortalama ömürleri kadar değil, en kısa ömürlü duyarganın ömrü kadardır. Örnek olarak, kablosuz güvenlik ağları uygulaması göz önüne alındığında, her bir düğümün ömrü birkaç yıl kadardır. Fakat herhangi bir nedenle bir düğümün enerjisinin beklenenden erken bitmesi, güvenlik sistemini etkisiz kılacaktır.

Bazı durumlarda, örnek olarak bir bina içine yerleştirilmiş olan kablosuz duyarga sistemlerinde, düğümlerin enerjilerini bina içerisindeki güç hatlarından alabilme olanağı varsa, mümkün olan her düğümün güç kaynağına bağlanacak şekilde yerleştirilmesi söz konusu olabilir. Alternatif bir başka yöntem ise, düğümün kendi enerjisini üretmesidir. piezzoelektrik jeneratörler veya güneş panelleri kullanılarak düğüm kendi harcadığı enerjiyi üretebilir. Tabii bunu yapabilmesi için, düğümün enerji tüketiminin oldukça az olması gerekmektedir. İlerleyen teknoloji sayesinde şu anda aktif halde 1μA enerji çeken mikrodenetleyiciler (msp430 gibi) bulunmaktadır. (Tablo 2.1**Hata! Başvuru kaynağı bulunamadı.**'de platformlar karşılaştırmalı olarak verilmiştir.) Gene anlık enerji tüketimini arttırmamak için düğümlerin farklı çevresel birimlerini aynı anda çalıştırmak yerine, farklı zamanlarda çalıştırması şeklinde yöntemler kullanılabilir. Enerji üretiminin kesilebilmesi ihtimaline (güneş panellerinin

ışık alamaması gece olması durumu gibi...) karşın düğüme enerji üreteçlerinin yanında enerji depolama kabiliyetinin de eklenmesi düşünülebilir. Düğüm bu senaryoda, enerji üretilebilir bir durumdaysa, üretilen enerjiyi depolar, enerji kesilmesi durumunda ise depolanmış enerjiyi kullanılarak çalışmaya devam eder.

Tablo 2.1: Çeşitli kablosuz duyarga ağlarının karşılaştırılması

Özellik	**Imote**	**Micaz**	**Telos**	**Mica2**	**Imote2**
Mikrodenetleyici tipi ve min Hızı [MHz]	32 bit ARM, 12	8 bit Atmel, 8	16 bit TI, 8	8 bit Atmel, 8	32 bit XScale, 13
Maksimum Hızı [MHz]	40	16	16	16	496
Radyo	Bluetooth	802.15.4 – ZigBee	802.15.4 – ZigBee	300-900 MHz	802.15.4 – ZigBee, Bluetooth/802.11
Band genişliği [kB]	720	250	250	15	250, 720/11000
Güç tüketimi Aktif/RX/TX [mA]	15/24/24	8/20/18	1/20/18	8/10/27	40/20/18
Güç tüketimi uykuda [µA]	1-250	27	6	19	1-100
İşletim sistemi	TinyOS	TinyOS	TinyOS	TinyOS	TinyOS

Bir kablosuz duyarga ağında çalışan düğümler incelendiğinde en çok güç harcayan birim, düğümün kablosuz haberleşmeyi sağlayan birimidir. Örnek olarak,

mikrodenetleyici aktif modda 270 µA harcarken, bir kablosuz haberleşme ünitesi alıcı modunda(RX) 18.8 mA harcarken(CC2420), verici modunda(TX) 17.4 mA harcamaktadır. [1,2]

Görüldüğü üzere minimum güç tüketimini sağlayabilmek için, haberleşme modülünün mümkün olduğunca az açık tutulması ve kullanılması gerekir. Bunu sağlayabilmek için, kablosuz haberleşme protokolünün olabilecek en az sürede veri iletimini sağlaması, tekrar gönderme ve protokolün devamlılığını sağlayan paket gönderimlerini minimumda tutacak şekilde etkinleştirilmiş olması gerekmektedir. Alternatif olarak, eğer düğümler birbirleriyle daha düşük güçte haberleşebilecekse, anten çıkış gücünün azaltılması da güç tüketimini azaltacaktır.

2.2 Kapsama Alanı

Bir kablosuz duyarga ağında çalışma ömründen sonraki en önemli özelliklerden biri, düğümün kablosuz haberleşme yapabileceği kapsama alanıdır. Kapsama alanı sadece tek bir duyarganın haberleşme yapabildiği yer olarak düşünülmemelidir. Düğümler arası haberleşme teknikleri sayesinde, teorik olarak bir kablosuz duyarga ağının kapsama alanı sonsuza kadar uzatılabilir. Ancak belirtmek gerekir ki, kablosuz duyarga ağının kapsama alanının belirli bir değerden daha da büyümesi, düğümler arası haberleşme protokollerinin daha fazla çalışmasına neden olmakta ve dolayısıyla enerji tüketimini ciddi bir şekilde arttırmaktadır ve ağın ömrünü azaltmaktadır.

Kablosuz haberleşme protokollerinin bir özelliği olarak, istenildiği takdirde kapsama alanı yeni düğümler ekleyerek veya çıkarılarak değiştirilebilir. Bir kullanıcı ilk önceleri az sayıda düğümle oluşturduğu kablosuz duyarga ağına sonraları yeni düğümler ekleyebilir veya tersi de geçerlidir. Burada unutulmaması gereken nokta, kablosuz duyarga ağının, eklenen yeni düğümleri destekleyebilmesidir. Bu gerek örnekleme hızının düşmemesi gerek kablosuz haberleşme protokolünün yeni eklenenler de dahil olmak üzere o kadar sayıda düğümü haberleştirebilme yeteneği olmasıdır.

2.3 Güvenilirliği

Bir sistemin veya bir modülün, istenen koşullarda istenilen süre içinde fonksiyonlarını yerine getirebilme olasılığına güvenilirlik denir. [3] Kablosuz duyarga ağları, uzun yıllar yazılımsal veya donanımsal bir sorun ile karşılaşmadan çalışabilmelidir. Buradaki limit genel olarak, düğüme ait olan enerji kaynağı bitene kadar olarak tanımlanır. Kablosuz duyarga biriminin fiziksel yeri kimi uygulamalarda bilinmemekte gene duyarga birimine kimi uygulamalarda da maliyet sebepleri ile servis verilemeyebilir. Bu tip nedenlerden dolayı kablosuz duyarga ağındaki her bir düğüm, uzun süreler boyunca hatasız çalışabilmelidir.

İlk ayarlamaya ek olarak, düğümler olası değişken çevresel koşullara uyum sağlayabilmelidirler. Bu, kimi zaman düğümlerin fiziksel yerlerinin değişmesi kimi zaman çevredeki haberleşmeye engel olacak büyüklükte cisimlerin yer değiştirmesi sonucu girişime maruz kalma şeklinde olabilir. [4–6]

Yer değişimlerinin fark edilebilmesi veya olası girişimlerin engellenmesi, haberleşme protokolündeki karmaşıklığı arttırır, çoğu zaman bu karmaşıklık band genişliğine ek yükler getirir ve düğümün daha fazla enerji harcayarak çalışma ömrünü olumsuz yönde etkiler. Aynı şekilde, protokoldeki karmaşıklıklar, yazılımlardaki olası böceklerin gizli kalmasını kolaylaştırarak, kablosuz duyarga birimlerinde kilitlenmelere neden olabilir. Bunların her biri güvenilirliği olumsuz etkiler. Duyargaların güvenilirlikleri, genellikle Poisson dağılımına dayanan (2.1) denklemi ile modellenir: [7]

$$R_k(t) = e^{-\lambda_k t} \tag{2.1}$$

λ_k : k numaralı duyarganın hata yapma oranı

t: duyarganın hata yapma olasılığının test edildiği zaman aralığı

Haberleşme protokolleri ve algoritmalar çeşitli hataya duyarlılık seviyeleri ile tasarlanabilirler. Bu tamamıyla duyarganın çalışacağı fiziksel koşulların ne derece sert olduğuna göre ayarlanır. Örnek olarak, savaş alanlarında kullanılacak bir sistem ile bir serada sıcaklık ve nem ölçecek olan sistemin hataya karşı toleranslarının aynı olmaması gerekliliği verilebilir.

2.4 Geliştirme, Üretme ve Kurulum Maliyetleri

Kablosuz duyarga ağlarının dikkat çeken özelliklerinden biri de kurulum maliyetlerinin çok az olmasıdır. Duyarga ağlarını kullanan insanların, bir duyarganın nasıl çalıştığını veya haberleşme protokolünün arka planında koşan algoritmaları bilmeleri beklenmemelidir. Kablosuz duyarga ağları, kolay bir şekilde onu kullananlar tarafından da kurulabilmektedir. Bunun sırrı, düğümlerin birbirlerini tanıyıp ona göre kendilerini ayarlayabilmelerinden gelir. Bu sayede kullanıcı, hiçbir ayar yapmaya gerek duymadan düğümleri kolayca yerleştirir. (tabii burada düğümlerden herhangi birinin ağın diğer üyeleriyle bağlantıda olan bir düğümün kapsama alanında olmasına özen gösterilmelidir.)

Her bir düğümün otonom olarak kendini ayarlayabilmesi, etrafında bulunduğu düğümleri tanıması, bunların arasındaki haberleşmenin kalitesini ve yeni olası haberleşme olasılıklarını bulması gibi bilgileri takip etmesi gerekliliğini doğurur.

Kurulum maliyetleri, geliştirme maliyetlerine göre daha azdır. Geliştirme esnasında, sistemlerin bozulması veya yazılımlarının çökmesi gibi olumsuzlukların tespiti için çok miktarda test yapılması gereklidir. Bir güvenlik uygulamasında kullanılacak kablosuz duyarga ağlarının oldukça dayanıklı olması gereklidir. Bu dayanıklılık gerek fiziksel zorluklara ve çevresel bozuculara karşı hem de ağdaki bazı düğümlerin sabote edilerek iletişimlerini kaybetmesi ihtimaline karşıdır. Bu durumda düğümler haberleşme için alternatif yollar bulabilmelidirler. Tabii bu tip testler ve çözümler maliyeti arttırır.

Aynı şekilde, sistem farklı uygulamalara uyarlanabilir olması açısından da mümkün olduğunca modüler yapıda olmalıdır. Bu sayede geliştirilmiş olan duyarga ağı farklı uygulamalara kolayca uyarlanabilir. Fakat bu şekilde modülerlik de geliştirme maliyetini arttırır.

Üretme maliyetleri ise sistemden beklenen dayanıklılığa ve istenen algılayıcı verilerine göre değişiklik gösterir. İstenen algılama çeşitliliği arttıkça, karmaşıklık ve kullanılan malzeme miktarı artar ve maliyeti arttırır.

2.5 Cevap Süresi

Girişte de belirtildiği üzere, özellikler uygulamaya bağımlıdırlar. Bir alarm uygulaması düşünüldüğünde, alarm durumu oluştuktan ne kadar süre sonra alarm bildirildiği veya alarm durumuna geçtiği kritik derecede önemlidir. Çevresel koşulları algılayan bir uygulama düşünüldüğünde, cevap süresi önem derecesini yitirmektedir. Alarm durumu, periyodik olmayan bir durum olduğundan, fabrika veya endüstriyel makineler ile ilgili veri toplayan bir ağda alarm durumu oluştuğu zaman, sistemin algılayıcı verilerinden ziyade alarm durumunu haber veren mesajlara öncelik vermesi gerekir. Ancak bu şekilde belirli bir cevap süresini garanti eden uygulamalar gerçek dünyada uygulama alanı bulabilir. Bu noktada düşük güç tüketimi ile cevap süresi birbiri ile ters orantılı davranır. Eğer bir sistemin çalışma periyodu ne kadar az ise haberleşme modülü de o kadar az açık kalacağı için çalışma ömrü daha uzun olabilir. Tabii böyle bir uygulamanın alarm gibi uygulama alanlarının gerektirdiği özellikleri desteklemesi beklenemez.

2.6 Zamanla Doğruluk

Bazı çevresel koşulları algılayan uygulamalarda ve nesne veya canlı takip uygulamalarında, olayın tanımlanabilmesi için algılayıcılardan alınan verilerin zaman ekseninde korelasyona tabi tutulması gereklidir. Korelasyon yapılabilmesi için, düğümler arasında zaman eşlemesi bulunmalıdır. Zaman eşlemesinin doğruluğu, elektromanyetik dalgaların yayılım hızına ve kablosuz duyarga düğümünün zaman eşlemesini sağlayan algoritmasının gelişmişliğine bağlıdır. Bir binanın sıcaklığının o andaki ortalama değeri isteniyorsa birkaç saniyelik doğruluk yeterli iken, aynı binanın sismik davranışı inceleniyorsa mikro saniyeler bazında doğruluk gerekli olacaktır.

Zamanla doğruluğu sağlamak için, kablosuz duyarga ağında belirli bir evrensel zamanın tanımlanması gerekir. Diğer düğümler ise, bu evrensel zamanı tutan düğümden aldıkları komut ile algılayıcılarını çalıştırıp cevap verecekler veya bu evrensel zamana göre kendilerini kalibre edip ona göre çalışacaklardır. Düğümlerin kendilerini evrensel bir zamana göre ayarlarlarken beklenen doğruluk, haberleşme modülünün ne sıklıkta haberleşmesi gerektiğini ve dolayısıyla da kablosuz duyarga ağının çalışma ömrünü etkiyecektir.

2.7 Güvenlik

Çevresel koşulları izleme uygulamasında, sıcaklık ve ışık bilgilerinin toplanması çok zararsız gibi gözükse de, bu bilgileri güvende tutmak aslında çok önemlidir. Bir bina için uygulandığı düşünüldüğünde, o binadaki ofislerin belirgin kullanım saatleri veya güvenlik zaafları hakkında bilgiler elde edilebilir. Kötü niyetli kişilerin ellerinde bu bilgiler, fiziksel bir zarar verme veya stratejik bir saldırı düzenleme olarak geri dönebilir. Bu yüzden, kablosuz duyarga ağları, topladıkları ve birbirlerine aktardıkları bilgilerin başkaları tarafından ele geçirilemeyecek şekilde güvenlikli olmalarını sağlamak zorundadır.

Farklı kodlamaların ve şifreleme algoritmalarının kullanımı güvenlik sorunu çözse de bir yandan da güç tüketimini ve kablosuz ağın daha fazla band genişliğine ihtiyaç duymasına neden olmaktadır. [8,9] Her bir pakette, şifreleme ve şifre çözme için ekstra veri işleme gerekecek ve düğümleri doğrulamak için de ekstra bitler kullanılacaktır. Bu durum birim zamanda alınabilecek örnek sayısını azaltacak ve ağın çalışma ömrünü azaltacaktır.

2.8 Etkin Örnekleme

Bir veri toplama uygulaması olarak çalışan ağda, etkin örnekleme zamanı en önemli parametre olarak ortaya çıkar. Etkin örnekleme zamanı; algılayıcı verilerinin toplandığı bir ağda her bir algılayıcının verisinin okunup, veri toplama merkezine ulaştırılmasına kadar geçen süre olarak tanımlanır. Çevresel veri toplama uygulamalarında algılayıcı verilerinin 1–2 dakikalık periyotlarla toplanmasına ihtiyaç vardır. Düğümün çalışacağı periyot düşünülürken, algılayıcıdan örnek alınıp bunun haberleşme modülü ile aktarılmasına ilaveten, haberleşme paketinin düğümler üzerinden veri toplama merkezine gidene kadar harcadığı zaman da göz önünde bulundurulmalıdır. Aynı şekilde, bir düğüm sadece veri toplama işi yapmayacak, onun üzerinden merkeze ulaşacak bütün verileri de merkeze doğru gidecek şekilde iletecektir. Bu da, kablosuz duyarga ağındaki düğüm sayısının artması ile düğümlere binen iletim yükünün artacağı ve etkin örnekleme süresinin de artacağı anlamına gelmektedir. [10]

2.9 Boyut

Bir duyarga ağındaki her bir düğüm, verici-alıcı, işlemci, güç kaynağı ve algılama birimi olmak üzere dört temel parçadan oluşur. Bunlara ek olarak, uygulamaya bağlı olarak, yer konumlandırma sistemi, yönlendirici de bulunabilir. Bir duyarga biriminde birden fazla algılayıcı da olabilir. Örnek olarak, hem ısı hem titreşim ölçmesi istenebilir. Eklenen her bir ek parça duyarganın boyutunu da büyültür. Bir duyarganın normal koşullarda bir kibrit kutusuna sığabilecek kadar küçük olması istenir. [11]

Bazı uygulamalarda ise, duyarganın havada uçabilecek kadar küçük ve hafif olması dahi istenebilmektedir. Bu amaçla $1cm^3$'lük bir hacme sığabilen duyargalar geliştirilmiştir. [12]

2.10 Uygulama Alanları

Duyarga ağları, akustik, kızılötesi, termal, ivme, manyetik ve benzeri algılayıcılar kullanarak aşağıdakilere benzer algılamalar yapabilmektedir.

- Isı
- Nem
- Araç hareketi
- Işık
- Basınç
- Ses
- İvme değişiklikleri
- Toprak ve sıvı özellikleri
- Nesneler üzerindeki gerilimler ve burulmalar
- Nesnelerin hız, istikamet ve hacim gibi özellikleri

- Nesnelerin tespiti

Duyargaların kablosuz olarak haberleşmeleri konsepti ve algılayıcılardaki çeşitlilik, kablosuz duyarga ağlarına çok çeşitli uygulama alanları sunmaktadır. Bunlara örnek olarak; uzay, askeri, çevre, ticari, kimyasal işlem, ev, doğal afet gibi uygulama alanları verilebilir.

Duyarga ağları,

- Bir canlının üzerine
- Bir binaya
- Bir nehire
- Hızla hareket eden bir aracın üzerine
- Bir okyanusun dibine
- Bir fırtınanın içine
- Bir fırtınanın içinde deniz yüzeyine
- Makinelerin içine
- Bir savaş alanında düşman hatlarının gerisine
- Biyolojik veya kimyasal kirlilik bölgelerine

yerleştirilmek veya atılmak suretiyle çalışabilmelidirler.

Bu çalışmada, kablosuz duyarga ağlarının uygulama alanları 4 temel başlıkta incelenecektir. Bunlar, güvenlik, çevresel koşulları izleme ve belirli bir alanda bir nesneyi veya canlıyı takip etmeyi amaçlayanlar olarak sınıflandırılabilir. Bu üç başlığa ilaveten, bir dördüncü başlık ise bu bahsedilen amaçların birden fazlasını içeren karma durumlar olarak ele alınabilir.

2.10.1 Çevresel koşulları izleme

Çevresel koşulları izleme uygulaması, temel olarak bir araştırmacının belirli bir bölgeye yayılmış çeşitli algılayıcı verilerini belirli periyotlarla toplamak istemesi ihtiyacına cevaben ortaya çıkmıştır. Araştırmacı, yüzlerce farklı noktadan çeşitli algılayıcı verilerini tek bir merkezde toplar ve bunları gerçek zamanlı veya depolayarak belirli aralıklarla inceler. [13,14]

Çevresel değişkenlerin durumunu gösteren bu veriler depolanarak, uzun süreli veya kısa süreli istatistikler oluşturulabilir. Burada dikkat edilmesi gereken husus, verilerin okunuş periyodunun sabit tutulması ve algılayıcıların yerlerinin değiştirilmemesidir. Veri toplama periyodu genel olarak bir dakika veya daha çok olduğu için, ihtiyaç duyulan band genişliği çok değildir ve kablosuz duyarga ağının çalışma ömrü uzundur.

Kablosuz duyarga ağının düğümleri yerleştirildikten sonra, düğümler etraflarındaki diğer düğümleri bulmaya ve olası haberleşme yollarını bulmaya çalışırlar.[15] Bu aşamadan sonra her bir düğüm, algılayıcılarından örneklediği verileri, verileri depolayacak olan düğüme iletmeye çalışır. Çevresel koşulları izleme uygulamalarında düğümler, veriyi depolayacak olan düğüme erişmek için haberleşme paketlerinin izleyecekleri en etkin yolu bulmak zorunda değildirler çünkü alternatif ek olarak yapılacak her hesap işlem yükü olarak işlemciye binecek ve çalışma ömrünü azaltacak bir etki yaratacaktır. Bunun yerine, düğümlere, haberleşme paketlerini aktaracakları en etkin yol, belirli bir merkezden bildirilebilir. Böyle bir yöntemin kullanılabilir olmasında ki en önemli etken, düğümlerin fiziksel yerlerinin zamanla değişmeyecek olmasıdır.

Çevresel koşulları izleme algoritmaları topoloji olarak genelde ağaç topolojisini kullanırlar. Veri, periyodik olarak oğul düğümlerden ana düğümlere ve son olarak da veri depolayacak olan merkez düğüme doğru akar. Ağaç yapısında her bir düğüm kendisine bağlı olan oğul düğümlerin verisini ağaç mimarisinin üst katlarına doğru iletmekle yükümlüdür. Kendi verileri dışında ekstradan oğul düğümlerinin de verilerini merkez düğüme doğru ileten düğümler, onlara bağlı olan oğul düğümlere göre daha çok enerjiye ihtiyaç duyarlar, bu yüzden de çalışma ömürleri daha kısadır. [16,17]

Kablosuz duyarga ağı, kendi kendini ayarladıktan sonra, düğümler algılayıcılardan 1 dakika ile 15 dakika arasında bir periyotta örnek alarak bunu duyarga ağında yayınlarlar. Teknolojik olarak, 1 dakikanın altında da örnekleme yapılması mümkündür fakat pratik olarak gerekli değildir çünkü çevresel koşulları oluşturan değişkenler sıcaklık, nem gibi yavaş değişirler.

Çevresel koşulları izleme uygulamalarının en belirgin özellikleri, uzun çalışma ömrü, senkronize çalışma, düşük veri iletim hızları ve sabit ağ topolojileridir.

2.10.2 Güvenlik uygulamaları

Bir diğer uygulama alanı güvenlik uygulamalarıdır. Güvenlik uygulamasında, kablosuz duyarga ağının düğümleri devamlı olarak etraflarındaki normal dışı durumları bir veya birden fazla algılayıcısı ile algılamak amacıyla belirli noktalara sabitlenmişti. Çevresel koşulları izleme uygulamaları ile güvenlik uygulamaları arasındaki temel fark, güvenlik uygulamalarında verilerin depolanmak için toplanmamasıdır. Bu durum duyargaların ve ağın özelliklerinde belirgin değişikliklere neden olur. Her bir düğüm, düzenli aralıklarla algılayıcılarından örnekler alır fakat bunların verilerini direk olarak aktarmaz, anormal bir durum var ise bir rapor halinde bildirir. Bu tip raporlar alarm mesajı olarak adlandırılır ve ağın en önemli özelliği bu alarm mesajlarının öncelikli olarak iletilebilmesi ve gideceği yere ulaşmasıdır.

Dikkat edilmesi gereken bir başka nokta ise, düğümlerin belirli aralıklarla orada olduklarını ve hala çalıştıklarını teyit etmeleridir. Eğer bir düğümün istenmeyen bir şekilde yeri değişmişse veya çalışamaz duruma gelmişse, bir güvenlik açığına neden olacağı için, bunun fark edilmesi ve ağın bundan haberdar edilip gerekiyorsa bir alarm mesajı oluşturulması gereklidir.

Yukarıda sayılan nedenlerden dolayı, güvenlik uygulamalarında kullanılacak en etkin topoloji, çevresel koşulları izleme uygulamalarında kullanılacak olan topolojiden farklılık gösterir. Eğer ağaç yapısı topoloji kullanılsaydı, oğul düğümü olan düğümler, oğullarının da paketlerini yukarı iletmek zorunda olduklarından daha çok enerji harcayacak ve çalışma ömürleri kısalacaktı. Tersine, doğrusal bir topoloji kullanarak, her bir düğüme eşit sayıda oğul düğüm atanarak her birinin enerji tüketimini birbirine yakın tutmak mümkündür.

Günümüzde kullanılan uygulamalarda, her bir düğümün varlığının ve fonksiyonelliğinin devam ettiğinin her saatte 1 defa teyit edilmesi yeterli görülmektedir.

Bir alarm durumu oluştuğunda, alarm mesajı hemen merkeze iletilebilmelidir. Alarmın algılanıp, merkeze kadar iletilmesi için geçen süre kritiktir. Yangın gibi. uygulamalarda bu sürenin 1–2 saniyenin altında kalması yeterli görülür. Yeterliliği sağlayabilmek için, algılamayı yapan düğümün yanındaki düğümlerin de alarm mesajını hemen alıp, komşu düğümlere doğru aktarması gerekir. Bu tip uygulamalarda, alarm mesajlarının hızlı bir şekilde yerine ulaşması, bu sırada yapılacak olan enerji tasarrufundan daha önemlidir. Alarm mesajının yerine daha hızlı ulaşabilmesi için düğümlerin alıcılarının daha sık sürelerde haberleşme ortamını dinlemesi gerekir.

Güvenlik uygulamalarında, enerjinin büyük bölümü düğümlerin fonksiyonelliklerinin sınanması ve hala aynı yerde olduklarının teyit edilmesi sırasında harcanır. Alarm mesajının iletilmesi, uygulamaya göre alarm mesajı ağın ömrü boyunca hiç gelmeye de bilir (örnek: yangın alarmı uygulaması), çok ufak bir miktar güç harcanmasına neden olur.

2.10.3 Takip ve izleme uygulamaları

Bu uygulamalar, kablosuz duyarga ağları ile donatılmış bir bölgeye giren bir canlı veya nesnenin tespit ve takip edilmesidir.

Günümüzde kargo takip sistemlerini örnek olacak olursak, kargolar üzerlerine yapıştırılan barkotların okutulması ile takip edilirler. Bu okuma, kargonun o an için nerede olduğunu değil, en son hangi okuyucudan geçtiğinin tespitidir. Çoğu zaman kargo firmaları için önemli olan, yer ve zamanın aynı anda bilinmesidir. Günümüz sisteminde bu bir eksiklik olarak yer almaktadır.

Oysa bu takip, kablosuz duyarga ağları ile kargolar üzerine yerleştirilecek basit bir düğüm ile kolayca yapılabilirdi. Kargoları taşıyacak vasıtalara yerleştirilen düğümler ile de taşıyıcıdaki kargoların tam listesine ve mevkiine ulaşılabilir. Düğümler hareketli oldukları için bu tip uygulamalarda topoloji çok sık bir şekilde değişir ve bu da

haberleşme ağında aşırı yüke sebep olur. Dolayısıyla da enerji ihtiyacı daha çok haberleşme modülleri tarafından gelir.

2.10.4 Karma durumlar

Genelde, pratik uygulamalar bu üç başlıkta sayılan uygulama alanlarından bir veya daha fazlasını içerirler. Örnek olarak, belirli bir alandan geçen araçları sayan bir kablosuz duyarga ağında, ağ kimi zaman algılayıcılarından gelen verileri analiz ederek araç mı yoksa başka bir şey mi geçtiğini inceleyip kendi başına karar verirken, kimi zaman ise sadece bir alarm üretebilir.

Uygulama alanlarının çeşitliliğinden dolayı, uygulama geliştirilme aşamasında kablosuz duyarga ağlarının algoritmalarının ve donanımlarının gerektiğine her bir uygulama alanında çalışabilecek şekilde modüler olması amaçlanır.

3. KABLOSUZ DUYARGA AĞLARINDA KULLANILAN YAZILIMLAR

Kablosuz duyarga ağlarının hedeflenen amaçları gerçekleştirebilmesi için, donanım özelliklerinin iyi planlanmış olması kadar, yazılımının da donanımı destekleyebilecek kadar iyi olması gereklidir. Bu nedenle yazılımın bir takım özelliklerinin olması gerekir:

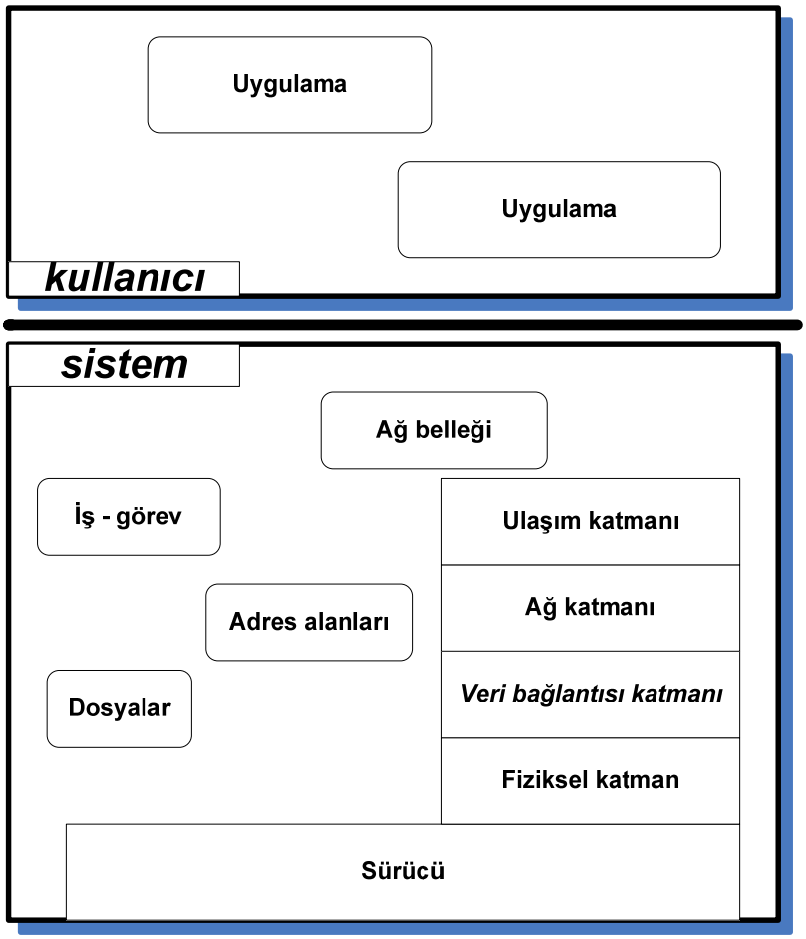
- Hafızayı etkin kullanmalı,
- Mikroişlemciyi etkin kullanmalı,
- Güç kullanımı etkin olmalı,
- Algılayıcı girişlerine cevaben kesme üretebilmeli,
- Uygulamaya yönelik özelleştirilebilmeli,
- Nesneye yönelik programlama mantığı ile çalışmalı,
- Veri toplama ve veri işleme süreçleri birbirinden bağımsız olabilmeli,
- Çok farklı donanımlar ile çalışabilmeli, esnek/modüler olmalı,
- Birden fazla uygulamayı bir arada yürütebilecek bir mimariye sahip olmalıdır.

Geleneksel gerçek zamanlı gömülü işletim sistemlerinden bazıları; VxWorks [18], WinCE [19], PalmOS [20], QNX [21], ve benzeri [22-24] olarak sayılabilir. Sayılanların çoğu küçük bir çekirdek yapısına sahip olup, uygulamanın gerektirdiği özelliklere göre paketler eklenebilir olarak tasarlanmıştır. (Tablo 3.1) Klasik bilgisayar işletim sistemi geliştirme arayüzüne benzer yüzler kullanılarak bilgisayar ortamında geliştirilerek hedeflenen donanıma yüklenir.

Tablo 3.1: İşletim sistemlerinin karşılaştırılması

İsim	Koruma	ROM boyutu	Ayarlanabilirlik	Derlenen yer −> Hedef işlemci
VxWorks	var	≈286Kbyte	Dinamik	Pentium −> ARM
QNX Neutrino	var	>100Kbyte	Dinamik	PentiumII −> NEC
QNX Gerçek zamanlı	var	100Kbyte	Dinamik	PentiumII −> 386
TinyOS	var	400 byte	Statik	Pentium −> ARM
CREEM	yok	560 byte	Statik	Atmel 8051
Chorus OS	opsiyonel	10Kbyte	Dinamik	Pentium −> ARM
pOSEK	yok	2Kbyte	Statik	mikrodenetleyicler
Ariel	yok	19Kbyte	Statik	ARM Thumb

İşletim sistemleri büyüdükçe ve geliştikçe, korumalı hafıza gibi özellikler taşır. Bu tip özellikler işletim sisteminin ihtiyaç duyduğu hafıza miktarını arttırır fakat bir nedenle yanlış yeri göstermeye başlayarak ulaşmaması gereken yere yazmaya çalışan işaretçilerin meydana getirdiği hatalara karşı da işletim sistemini korurlar. Bu tip işletim sistemleri, avuç içi bilgisayarlar, cep telefonları ve gömülü bilgisayarlar için çok kullanışlıdır. Buna rağmen, kablosuz duyarga ağlarının ihtiyaç duydukları düşük hafızaya ihtiyaç duymak gibi özellikleri sağlamaktan çok uzaktırlar. Büyük işletim sistemlerinin yanında, Creem [25], pOSEK [26], Ariel [27] gibi mikrodalga fırınlarında, motor kontrol uygulamalarında kullanılan nispeten daha küçük işletimleri de mevcuttur.



Şekil 3.1: Klasik bir işletim sistemi mimarisi

Şekil 3.1'de, klasik bir işletim sistemi mimarisi gösterilmiştir. Bu mimaride, katmanlar net bir şekilde belirlenmiş ve kesin çizgilerle birbirlerinden ayrılmışlardır. Kaynakların yeterli ve çok olduğu kabul edilerek tasarlanmıştır. Bunun nedeni ve etkisi, son uygulamaların tamamıyla birbirinden bağımsız hafıza ve kod bloklarına sahip olması ve bir diğer işletim sistemiyle veri alışverişini noktadan noktaya erişim prensibi ile kurmasıdır.

Kablosuz duyarga ağları ile çalışılırken, iki nokta ön plana çıkmaktadır:

- Olay önceliklidir ve olaylar rastgele meydana gelir. Birden fazla veri akışı ve işlemesi eşzamanlı olarak devam edebilmelidir.

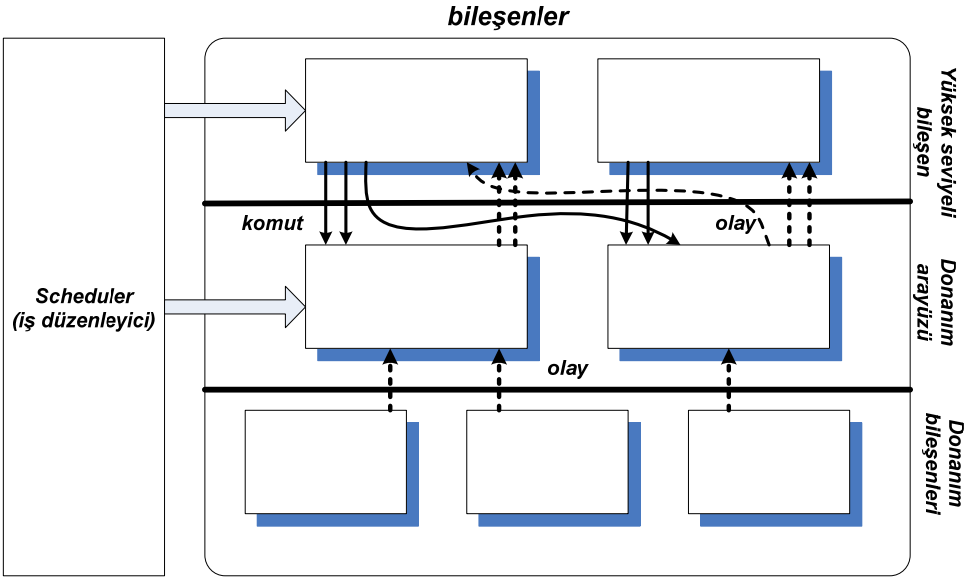
- Uygulamaya ve donanıma özgü bileşenler birbirlerine kolayca uyarlanabilmelidir.

TinyOS işletim sistemi, kablosuz duyarga ağlarının gereksinimlerini destekleyecek şekilde tasarlanmıştır. TinyOS, kablosuz duyarga ağlarında kullanılmak üzere tamamen ücretsiz ve açık kaynak kodlu olarak dağıtılan bir gömülü işletim sistemidir. Kaliforniya ve Berkeley Üniversitelerinin ve Intel'in iş birliği ile geliştirilmesine başlanmıştır. Daha sonraları gelişerek TinyOS Alliances [28] adında uluslararası bir birlik kurulmuştur.

TinyOS işletim sistemi, C programlama dilinin bir varyasyonu olan NesC programlama dili ile yazılmıştır.

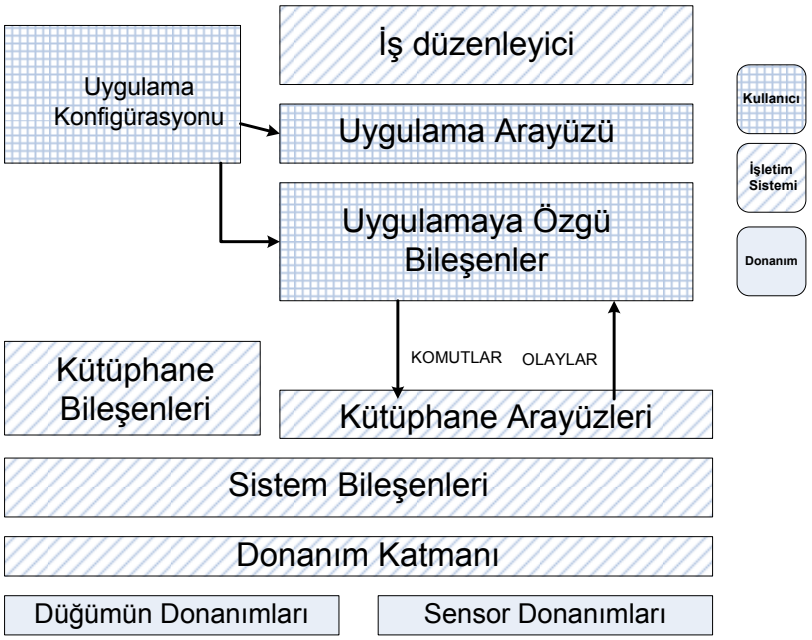
Bileşen tabanlı bir mimariye sahiptir. Klasik işletim sistemlerinden farklı olarak, işletim sisteminde çekirdek ve kullanıcı uygulamaları diye bir ayrım bulunmamaktadır. Bu yapı, programının tamamının analizinin ve eniyilemenin daha etkin bir şekilde yapılabilmesine olanak sağlar. İşletim sisteminin çekirdeği sadece 400 byte kod ve veri içerir. Statik bellek yönetimini kullanır. Böylece malloc/free dinamik bellek yönetimlerinden oluşabilecek bellek sızıntısı da engellenmiş olur. Sanal bellek oluşturulmaz ve her yerden ulaşılabilecek bir tek evrensel bellek kullanır.

TinyOS işletim sistemi, güç tüketimini azaltmak ve çalışma ömrünü arttırmak için "acele et ve uyu" diye bilinen bir strateji izler. Bu strateji, olabildiğince az güç harcamak için mikrodenetleyicinin mümkün olduğunca uyuması prensibine dayanır. Görev, olabilecek en kısa sürede bitirilip, işlemci en az güç harcayacağı uyku modunda bekletilir. İlaveten, uygulamaları sonsuz döngülerde bekletmek yerine kesmeler veya zamanlayıcı/sayıcı fonksiyonları kullanılır, fonksiyon çağrıları yerine makro ve "inline" gibi C programlama deyimleri kullanılarak kod eniyilemesini arttırır ve işlem süresini kısaltarak güç tüketimini azaltmayı hedefler. Bunların gerçekleştirilebilmesi içinse, TinyOS işletim sisteminde nesneye yönelik programlama yöntemi kullanılır. Şekil 3.2'de TinyOS işletim sisteminin temel bileşenlerini ve birbirleri ile olan bağlantılarını göstermektedir.



Şekil 3.2: TinyOS işletim sisteminin bileşenleri ve birbirleriyle olan bağlantıları

İşletim sistemi, tekrar kullanılabilen ve başka bileşenler tarafından çağırılabilen bileşenlerden ve bunların birbiriyle haberleşmesini sağlayan komut ve olaylardan oluşur. İş düzenleyici de komut ve olaylar tarafından gönderilen görevlerin hangi sırada yapılacağını kontrol eder. FIFO (ilk giren ilk çıkar) yapısında çalışır, öncelik tanımlı değildir. Şekil 3.3'de TinyOS işletim sisteminin katman yapısı gösterilmektedir.



Şekil 3.3: TinyOS işletim sisteminin katmanları

3.1 Nesneye Yönelik Programlama

Nesneye yönelik bir mimaride, tek bir uygulama birbirinden bağımsız görevler tarafından paylaşılarak yapılır. TinyOS işletim sisteminde, her bir sistem bileşeni, sürekli olarak kendilerine gelen olaylara cevap vererek çalışır. Bir olay meydana geldiğinde, ilgili görev çalışır ve istenilenleri yerine getirir sonra tekrar sisteme geri döner.

Yüksek performanslı işlem gerektiren alanlarda çalışan araştırmacılar, olay tabanlı uygulamalarda nesneye yönelik programlama yönteminin kullanılmasının gerekliliğini daha önceki çalışmalarda görmüşlerdir. [29–30]

3.2 Görevler

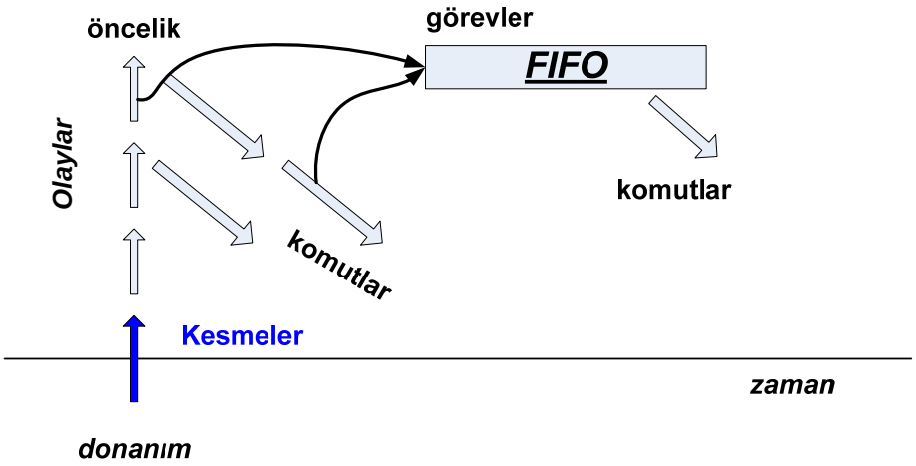
Nesneye yönelik programlamanın bir dezavantajı, çok uzun süren hesaplamaların, oluşum zamanı önemli olan olayların gecikmesine neden olabilmesidir. Aynı şekilde, bir hatadan dolayı, bir olay hiçbir zaman bitmezse, diğer sistem fonksiyonlarının durmasına diğer bir değişle hiçbir zaman yürütülememesine neden olabilecektir. Çok uzun süren hesaplamaların yapılabilmesi için, TinyOS görev adı verilen bir yürütme mekanizmasına sahiptir. Bir görev, arka planda çalışan ve ilgili hesaplama işlerini olaylara ve kesmelere engel olmadan yapan yürütme mekanizmasıdır. Görevler, iş düzenleyici tarafından işlemcinin boş olduğu zamanlarda yerine getirilir. İlaveten; görevler, kendilerinden daha düşük seviyeli sistem uygulamaları tarafından kesilebilirler. Böylece bir olay oluştuğunda görevler beklemeye geçer ve sistemin olayları zaman geciktirmeden işlemesine imkân tanınmış olur. İş düzenleyici, FIFO mimarisinde çalışır. Görevler arasında öncelik yoktur, iş düzenleyiciye ilk verilen görev bitmeden diğer görevler işlenemez. İş düzenleyici de yürütülecek herhangi bir görev yoksa işlemci saati haricindeki bütün uygulama ve donanımları uyku moduna sokar. Böylece gereksiz yere enerji harcanmaz.

İş düzenleyicinin çalışma yapısı da aşağıda gösterilen bir program kodu ile (Tablo 3.2) açıklanabilir.

Tablo 3.2: İş düzenleyicinin çalışma kodu

```
main() {
        …
        while(1) {
                while (more_tasks)
                        schedule_task();
                sleep();
        }
}
```

Aşağıdaki Şekil 3.4'te çalışma yapısı bir şekille gösterilmiştir. Donanım tarafından olaylar oluşturulmakta, olayların bazıları bir komut ile işini bitirirken, bazıları ise iş düzenleyiciye görevler vermektedir. Görevler yürütülürken, yeni komutlar çağırılabilir veya iş bitince sisteme geri dönülebilir.



Şekil 3.4: TinyOS işletim sisteminde olayların oluşumu ve sıralanması

3.3 Atomik Yapılar

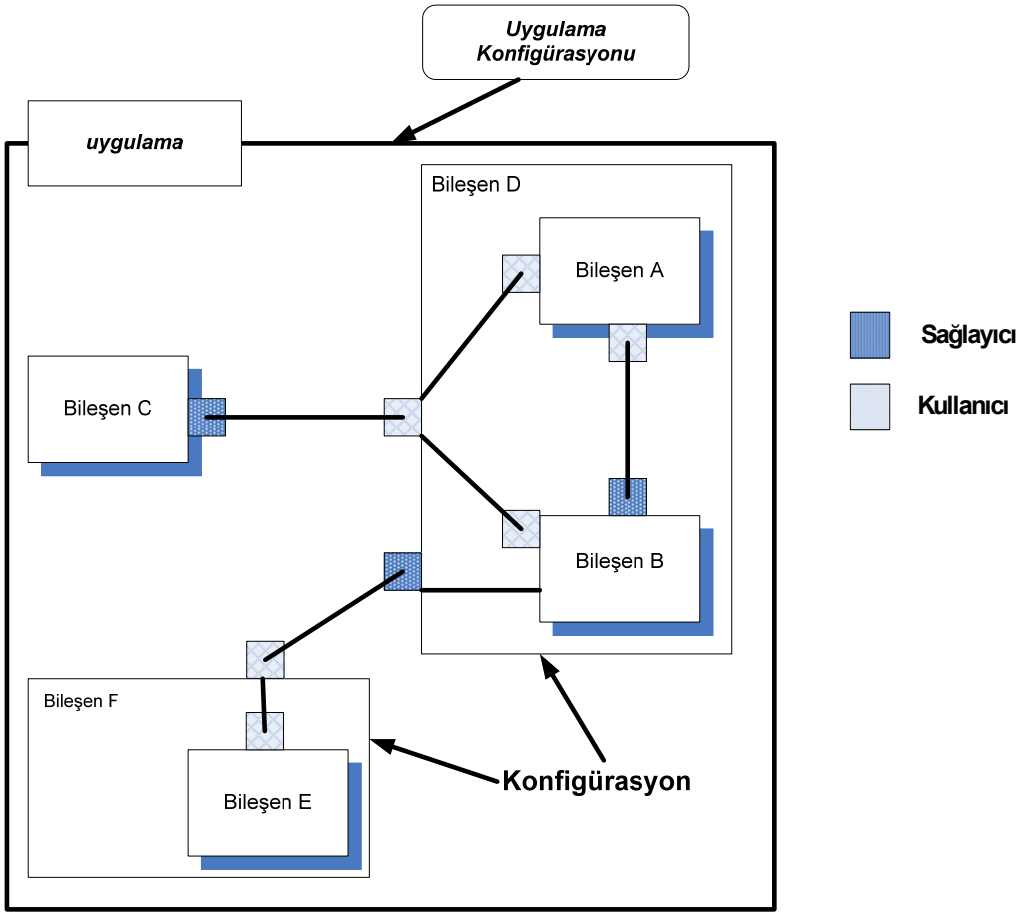
Çok uzun süren hesaplamalar için görev yapısı tanımlandığı gibi, TinyOS, yürütülmesi sırasında birbirinden ayrılmaması gereken kodlar için de atomikleştirme adında bir mekanizma tanımlamıştır. Nesneye yönelik programlamada hatalara yol açan

durumlardan biri farklı modüllerin ulaşabildiği bir bellek alanına aynı anda erişimin oluşturduğu yarışma durumu olarak adlandırılan durumlardır. İşlemin yapıldığı yer bir görevin içiyse ve görev bitene kadar daha öncelikli bir kodun bu alana erişmesi engellenmek isteniyorsa, bu bellek alanına erişilen kod parçası atomik kod olarak tanımlanır.

Atomik kodların çok uzun tutulması sistemde oluşan kesmelerin yürütülmesinin gecikmesine neden olabilir. Bu yüzden atomik olarak tanımlanan kod parçaları mümkün olduğunca kısa olmalıdır. Genel kullanım yerleri, diğer görev veya fonksiyonların erişebildikleri bellek bölgelerindeki işlemler olarak bilinir.

3.4 TinyOS Bileşen Mimarisi

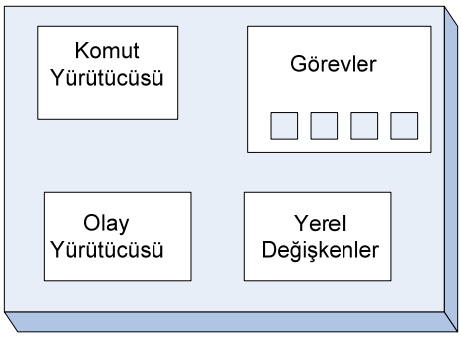
TinyOS işletim sisteminin nesneye yönelik olan yapısına ek olarak, TinyOS işletim sistemine özgü bir bileşen mimarisi vardır. Bileşen mimarisi, modüler ve kolayca birleştirilebilir bir şekilde geliştirilmiştir. Bileşen mimarisi, uygulama geliştirici için, birbirinden bağımsız bileşenleri kendi uygulamasına özgü bir şekilde bağlayarak, kolayca yeni bir uygulama geliştirmesine olanak sağlayacak şekildedir. Başka bir deyişle, TinyOS işletim sisteminde geliştirilen bir uygulama aslında o uygulamada kullanılan bileşenlerin listesi ve bunların birbirine bağlantılarını gösteren konfigürasyon dosyalarından oluşur. Şekil 3.5'de sembolik bir uygulama gösterilmiştir.



Şekil 3.5: Bir uygulamanın yapısı

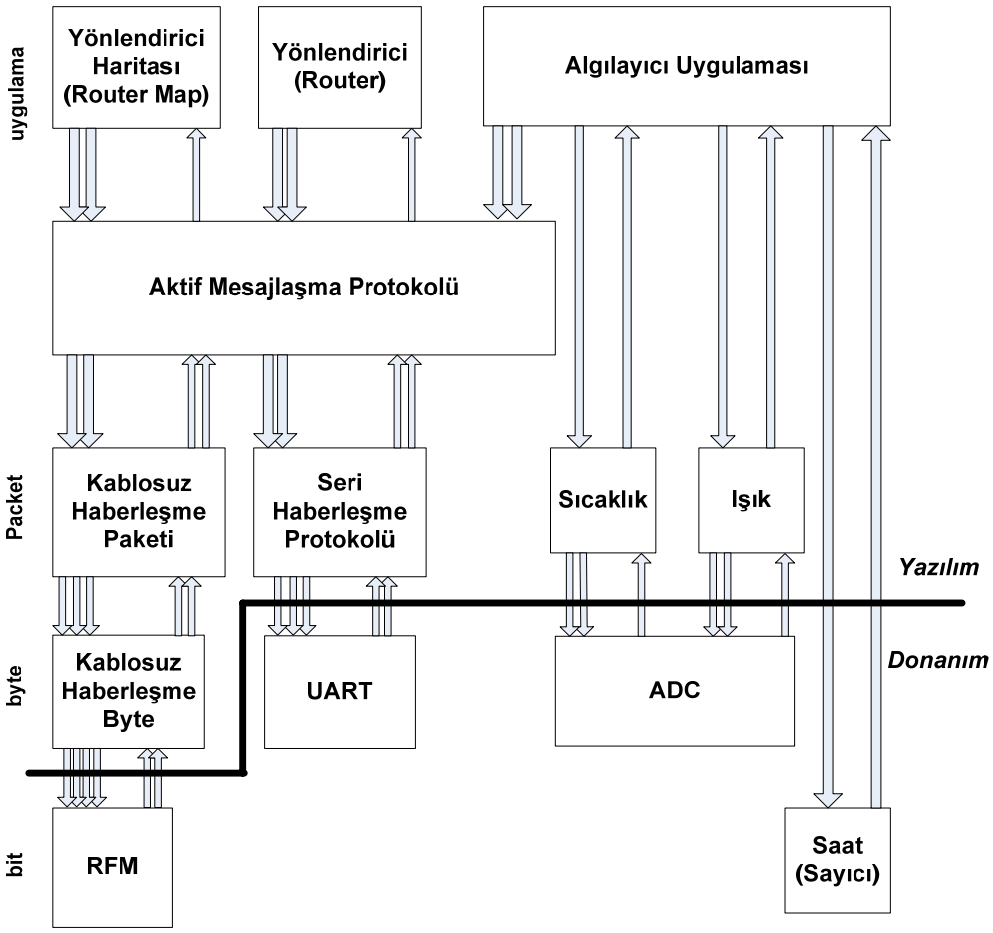
TinyOS işletim sisteminde her bir bileşen kendisine ait komutları ve olayları içerir. Bileşenin oluşturabildiği olaylar ve yürütmek için başka bileşenlere gönderdiği komutlar da genel olarak o bileşenin arayüzü olarak tanımlanır. Her bir bileşen Şekil 3.6'da gösterildiği üzere dört bölümden oluşur:

- Görevler
- Komutlar
- Olaylar
- Yerel değişkenler ve yerel fonksiyonlar



Şekil 3.6: TinyOS işletim sistemi bileşen içeriği

Modülerliğin sağlanabilmesi için, her bir bileşen kendisine sunulabilecek komutları ve kendisinin oluşturabileceği olayları bileşen dosyanın başında bildirir. Şekil 3.7'de sıcaklık ve ışık değerini algılayıp kablosuz haberleşerek gönderen bir uygulamanın bileşenleri ve bileşenler arası bağlantıları gösterilmektedir. Burada kullanılan bileşenlerin yanı sıra oklar kullanılarak, üst katmandaki bileşenlerin alt katmanlara gönderdikleri komutlar ve alt katmandaki bileşenlerin üst katmandakiler için oluşturdukları olaylar görülmektedir. En aşağı seviyedeki bileşenler direk olarak donanım ile haberleşirler.

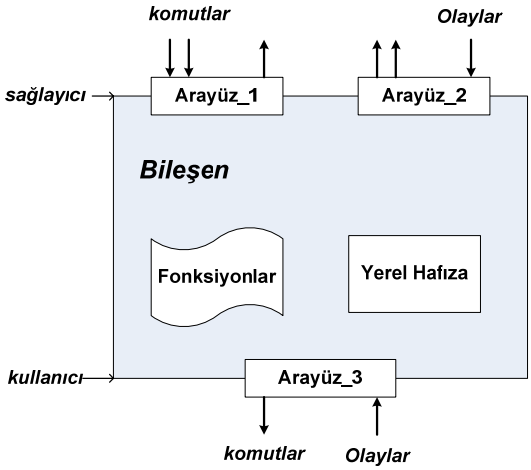


Şekil 3.7: Bir sıcaklık uygulaması örneği

Bileşenler içindeki yerel değişkenlerin bellekteki yerleri statik olarak belirlenir. Böylece derleme zamanında o bileşenin ihtiyaç duyacağı bellek alanı tespit edilebilir. Böyle olması aynı zamanda yerel değişken veya fonksiyonlara başka bileşenlerin dışardan erişmesini imkânsız kılar ve hafıza korumasına ihtiyaç duyulmamasını sağlar. Bellek alanının statik olarak belirlenmesi ve dinamik bellek mimarisine izin verilmemesi, hatalı işaretçi kullanımından dolayı meydana gelebilecek hataların da önüne geçilmesini sağlar. Bir başka olumlu yönü ise, değişkenlere işaretçiler üzerinden erişim yerine değişkenler hafızadaki yerlerine statik olarak derleme zamanında yerleştirilerek yürütme zamanı kısaltılmış olur.

TinyOS işletim sisteminde komutlar daha düşük katmanlardaki bileşenlere bloke edilemez olarak gönderilen mesajlardır. Tipik olarak, her komut çağırıcısına başarılı veya başarısız olduğunu belirten bir geri dönüş değeri döndürür. Komutun başarılı veya başarısız olduğunu bildiren geri dönüş değeri, o işlemin başarısını değil, komutun

ilgili bileşen tarafından alınıp alınmadığını gösterir. Bunun nedeni, her bileşenin dışarıya sadece arayüzlerini göstermesi ve arayüzler Şekil 3.8 tarafından başarılı olarak alınan bir çağrının sorunsuz olarak yürütülmesinin bileşenin kendi sorumluluğunda olmasındandır.



Şekil 3.8: Bileşenlerin arayüzlerinin gösterimi

Olaylar, donanıma yakın modüllerin veya başka bileşenler tarafından kurulmuş sayıcı kesmeleri sonucunda oluşan ve bileşenleri veya görevleri tetikleyen mesajlardır. Olaylar, komutları çağırabilir, görevleri başlatabilir veya başka olayları tetikleyebilir. Donanım tarafından tetiklenen olaylar, en düşük seviyeli başka bir deyişle en yüksek öncelikli olaylardır. Hızlı çalışması, mümkün olduğunca kısa süre yürütülerek işlemcinin olaylarda çok fazla zaman geçirmesi engellenmelidir.

3.5 Bileşen Tipleri

Tipik olarak, bileşenler üç kategoriye ayrılabilir. Donanım bileşenleri, donanım arayüzleri ve yüksek seviyeli bileşenler.

Donanım bileşenleri, TinyOS bileşen mimarisinde donanımın yerini tutar. Şekil 3.8'de gösterilen RFM bileşeni bu sınıftandır. Bu bileşen gelen komutlara göre, alıcı/verici modülünün giriş ve çıkışlarını ayarlar, modülün veya haberleşmenin durumu hakkında diğer bileşenler için olaylar yaratır. Yerel değişkenlerinde, alıcı/verici modülünün haberleşme hızı, aktif olan modu, gibi o anki durumu tutulur. RFM bileşeni tamamıyla kesmelerden oluşmuştur. TX veya RX bitinde oluşan kesmeleri üst katmandaki

bileşenle olay oluşturarak bildirir. Yerel hafıza bölgesinde herhangi bir görev bulundurmaz.

Donanım arayüz bileşenleri, daha karmaşık donanım davranışlarının yapılmasını sağlar. Buna iyi bir örnek Şekil 3.7'deki "kablosuz haberleşme byte" olarak adlandırılan bileşendir. Kendisine gönderilen veriyi, RFM bileşenine teker teker kaydırır ve bir bytelık veri aktarımı bitince üst katmana bir olay oluşturur. Yerel hafıza bölgesinde, basit kodlama algoritmalarını yürütür. (Manchester kodlama gibi.) Üst katmanlar için bu verileri kablosuz haberleşme yolu ile göndermek için bir arayüzdür.

Yüksek seviyeli bileşenler, kontrol, yönlendirme ve veri transferleri gibi karmaşık uygulamaları barındırırlar. Buna bir örnek olarak Şekil 3.7'deki mesajlaşma bileşeni gösterilebilir.

3.6 NesC

TinyOS işletim sistemine, C programlama dilinin bir varyasyonu olan NesC programlama dili ile program kodu yazılır. Kablosuz duyarga ağlarının düşük hafıza gereksinimi göz önüne alınarak C programlama diline olay tabanlı çalışabilme ve nesneye yönelik programlama özelliklerini ekler. NesC programlama dili, NesC önderleyicisi ile standart C koduna dönüştürülür ve açık kaynak kodlu C derleyicisi (gnu-gcc) ile ikili düzende PXA271 mikrodenetleyicisine yüklenmeye hazır dosya haline getirilir.

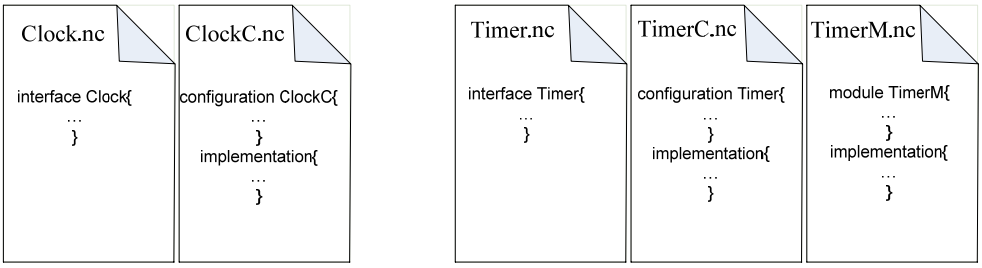
TinyOS işletim sisteminin statik bellek yönetimi sayesinde derleme zamanında programın tam çalışma şekli bilinir. Bu nedenle derleme zamanında program kodunda optimizasyon yapılır. Olası birden fazla bileşenin aynı bellek bölgesine aynı anda yazması şeklinde adlandırılan yarışma durumlarını tespit edebilir.

NesC programlama dilinde iki tip dosya bulunur: Modüller ve Konfigürasyonlar. (Şekil 3.9)

Modüller, modüllerin dışarıya verecekleri servisleri ve dışardan alacakları servisleri içeren arayüz tanımlarını içerirler. Dışarıya servis veren veya alan bir modül, aldığı veya verdiği veriyi kendi içinde de işleyebilir. Bunun için gerekli olan kod yazımı da modüller içerisinde yazılır ve ANSI-C yazım şekli kullanılır.

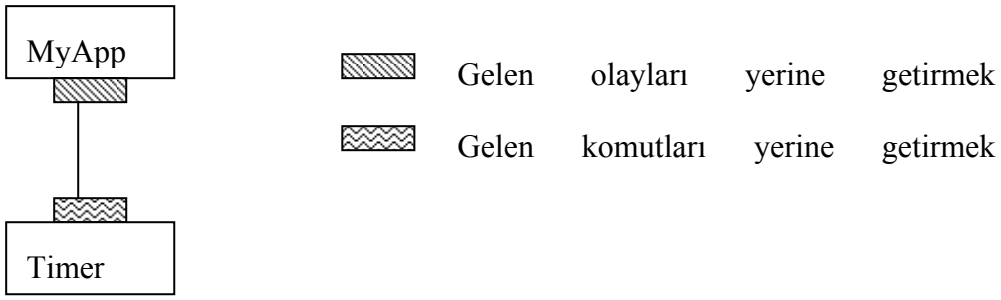
Konfigürasyonlar ise, bileşenlerin birbirleri ile olan ilişkilerini içerir. Bağlama olarak adlandırılan yazım şekli ile bir bileşen başka bileşenleri birbirine bağlayarak bir görevin yerine getirilmesini sağlayabilir.

Bir bileşen hem modül hem konfigürasyon dosyalarını içerebilir. Bu, o bileşenin hem başka bileşenleri kullandığı hem de kendi içinde C kodu bulundurduğu anlamına gelir.



Şekil 3.9: Modüller ve konfigürasyonlar

NesC programlama dilinde arayüzler, bir bileşenin başka bileşenler ile olan ilişkilerini düzenler. (Şekil 3.10) Bu ilişkiler hizmet verme veya hizmet alma şeklinde olabilir.



Şekil 3.10: Bileşenler arası bağlantılar

Standartlaştırabilmek amacıyla, her bir modül, daha önceden belirlenmiş kontrol arayüzlerini içerir:

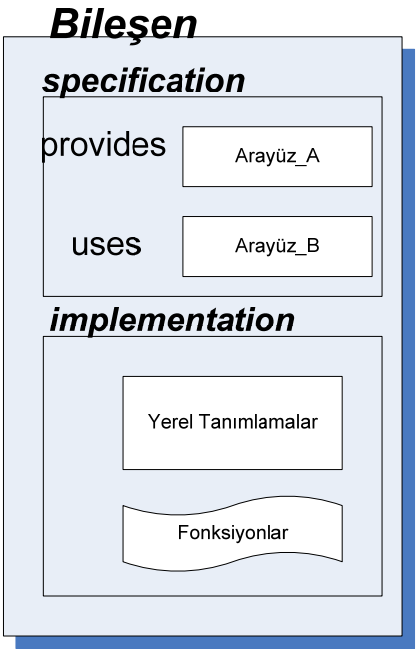
- init();
- start();
- stop();

Aynı şekilde tanımlanmış görevlerde iki aşamada ele alınır:

- send();

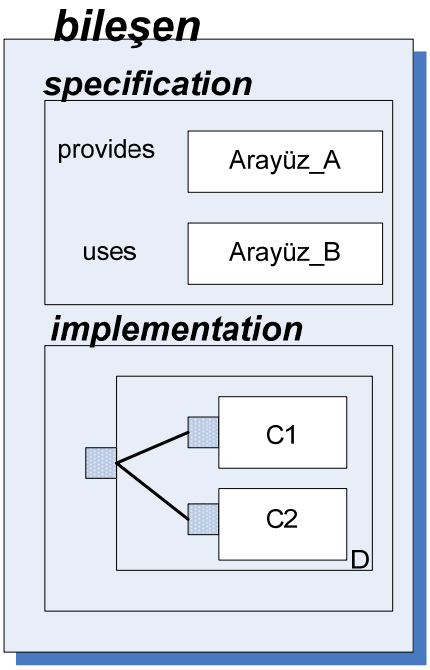
- sendDone();

NesC programlama dilinde modüller öncelikle bileşenin (Şekil 3.11) sunduğu hizmetlerin ve alacağı hizmetlerin sırasıyla "provides" ve "uses" anahtar kelimeleriyle listelendiği "module" fonksiyon tanımı ile başlarlar. Sonrasında ise, "implementation" fonksiyon tanımı ile modül içinde bulunan C kodunun yazılımı gelir. Bu kısım da yerel değişkenlerin tanımlanması ve fonksiyonların tanımlanması şeklinde iki kısımda incelenebilir.



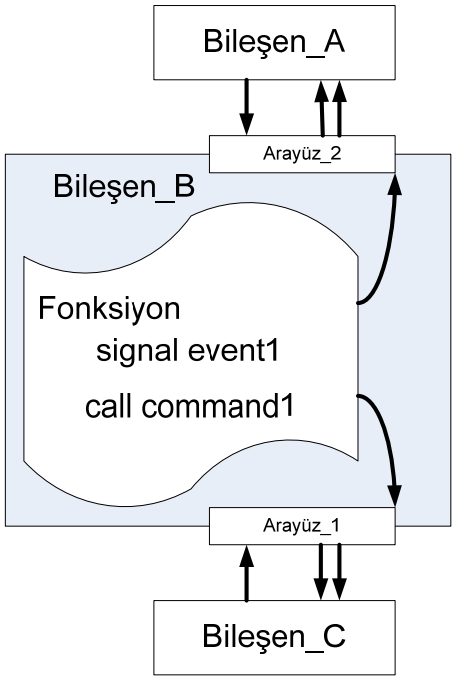
Şekil 3.11: Bileşenin iç yapısı

Konfigürasyon dosyaları (Şekil 3.12), "configuration" fonksiyonu ile başlarlar. Burada kullanılacak ve verilecek olan hizmetler belirtilir. Ardından "implementation" fonksiyonu ile hangi bileşen ve arayüzlerin birbirlerine bağlanacağı belirtilir. Burada herhangi bir C kodu yazılmaz. Bağlama amacıyla kullanılan "–>, <–, =" ifadeleri bulunur.



Şekil 3.12: Konfigürasyon bileşeni

Bağlanacak olan bileşenlerin birbirleriyle uyumlu olması gerekliliği vardır. Arayüz – arayüz, komut – komut ve olay – olay arasında bağlantılara izin vardır. Örnek olarak, "send" arayüzü, "receive" arayüzüne bağlanamaz. Ancak "send – send" arayüzü birbirine bağlanabilir.



Şekil 3.13 : Komutlar ve olaylar

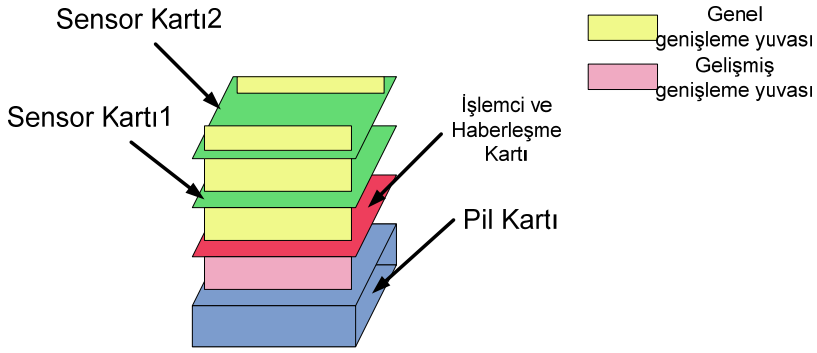
Şekil 3.13'de NesC programlama dilinde komutların ve olayların kullanılış şekilleri gösterilmiştir. Bir bileşen, başka bir bileşenden veya donanımdan bir olay çağrısı aldığında, bunu hemen koşmaya başlar. Aynı şekilde, bir bileşen başka bir bileşene "command" çağrısı yaptığında, ilgili bileşen kendi C kodundaki ilgili "command" fonksiyonunu işletir. Komutlar, daha düşük seviyedeki bileşenlere yapılırken, olaylar, komutlara veya donanımsal kesmelere cevaben daha yukarı seviyelere yapılır.

4. TEZDE KULLANILAN KABLOSUZ DUYARGA AĞI PLATFORMU

Çalışmada Crossbow® firmasının [31] Imote2 platformu kullanılacaktır. Imote2 platformu(Şekil 4.1), IPR2400 işlemci ve haberleşme kartı, ITS400 algılayıcı kartı ve pil kartından oluşmaktadır. Geliştirme aşamalarında kullanılmak ve ekstra bir USB çıkışı eklenebilmesi amacıyla hata ayıklama kartı (debug kartı) da isteğe bağlı olarak takılabilmektedir.

Şekil 4.1: Imote2 platformu

Platformda, Intel® firmasının düşük güç tüketimli PXA271 XScale mikrodenetleyicisi, Texas Instruments® firmasının [1] CC2420 haberleşme modülü ve çeşitli algılayıcıların bulunduğu algılayıcı kartı bulunmaktadır. Imote2 işlemci kartının her iki yüzünde bulunan genişleme yuvaları ile kartın modüler (Şekil 4.2) ve geliştirmeye açık olması sağlanmıştır. Üst yüzdeki genişleme yuvaları, standart giriş/çıkış bacaklarını sağlamaktadır. Alt yüzdeki genişleme yuvaları ise, yüksek hızlı arayüzlerin dışa olan bacak bağlantılarını oluşturmaktadır. Her iki yüzdeki genişleme yuvalarında da, pil kartının bağlantısı için gerekli bacaklar bulunmaktadır.

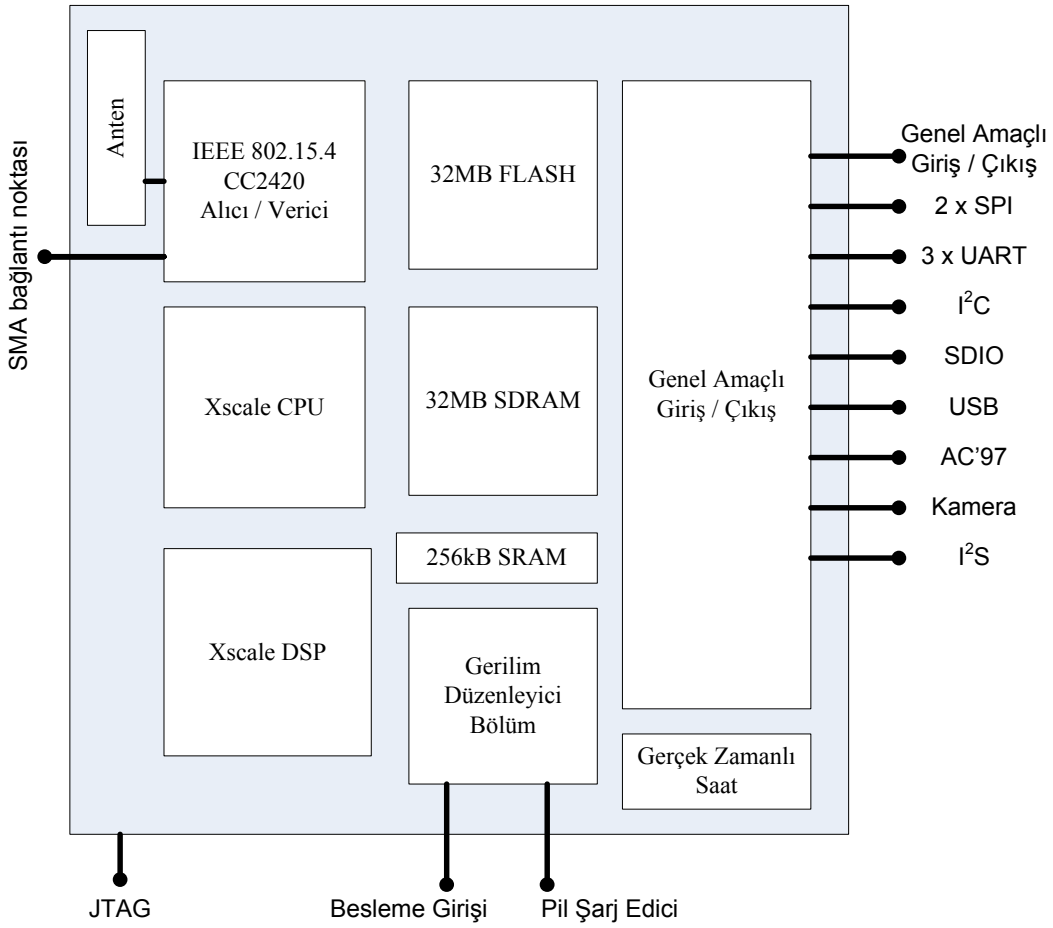


Şekil 4.2: Platformun monte edilmesi

4.1 Imote2 – İşlemci ve Haberleşme Kartı

İşlemci kartı üzerinde düşük güç tüketimi ve yüksek işlem gücü için Intel® tarafından tasarlanmış olan ARM V5TE komut yapısı ile uyumlu PXA271 mikrodenetleyicisini bulundurur. ARM komut sistemini desteklediği için, ARM için geliştirilmiş uygulamalar kolaylıkla XScale için de derlenebilir. XScale işlemcisinin en etkin şekilde kullanılabilmesi için, ARM için geliştirilmiş yazılımlarda XScale için gerekli etkinleştirilmelerin ve değişikliklerin yapılması gereklidir.

Mikrodenetleyicinin düşük güç tüketimli olabilmesi için düşük çekirdek voltajında (0,85 V) ve düşük saat çarpanında (13 MHz) çalışabilmektedir. En düşük çekirdek voltajı olan 0,85 V gerilim seviyesinde, saat çarpanı maksimum 104 MHz hıza erişebilmektedir. İşlemci üzerindeki dinamik gerilim ayarlayıcısı kullanılarak saat çarpanı kullanım kitapçığındaki maksimum 416 MHz hıza, aktif soğutucu kullanarak 520 MHz hıza ulaşabilmektedir. Yüksek saat çarpanı ile çalışması gerekli güç ihtiyacını arttırmaktadır. Kablosuz duyarga ağlarının genel prensibi olan “acele et ve uyu” kuralı için yüksek saat çarpanına sahip olmak önemli bir avantajdır. Aynı şekilde, işlemcinin uyku modunda da çok az miktarda güç çekmesi istenir. PXA271 mikrodenetleyicisi, gerilim izleyici ve saat çarpanı hariç diğer bütün çevre birimlerinin kapalı olduğu uyku modunda 0,15 mW, sadece saat çarpanın açık kaldığı geri kalan bütün çevre birimlerinin kapalı olduğu derin uyku modunda ise 0,1 mW (387µA) güç harcar.



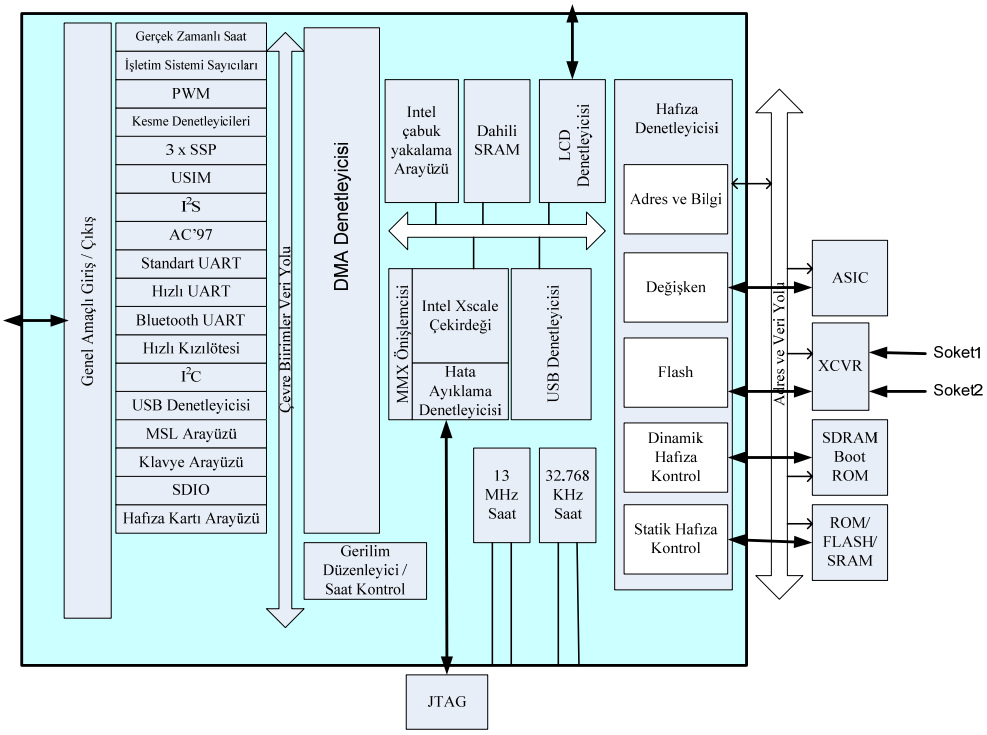
Şekil 4.3: Imote2 platformuna genel bakış

Çeşitli fiyat aralıklarında birden fazla sürümü olan PXA271 mikrodenetleyicisinin en gelişmiş ve en pahalı sürümünde (Şekil 4.3) tümleşik olarak, 32 MB SDRAM ve 32 MB Flash bellek içerir. Imote2 platformu da bu sürümünü kullanmaktadır.

Mikrodenetleyici, genel amaçlı birçok giriş/çıkış bacağına ilaveten kullanımı kolaylaştırmak ve standartlaştırmak amacıyla değişik arayüzleri donanım seviyesinde destekler:

- I2C
- 3 adet eşzamanlı seri arayüz
- 3 adet hızlı UART arayüzü
- SDIO arayüzü

- USB arayüzü
- AC97 ve I2S ses sıkıştırma arayüzleri
- Hızlı kızılötesi arayüz
- Darbe genişlik modülasyon (pulse-width modulation) arayüzü
- Kamera arayüzü
- Gerçek zamanlı saat

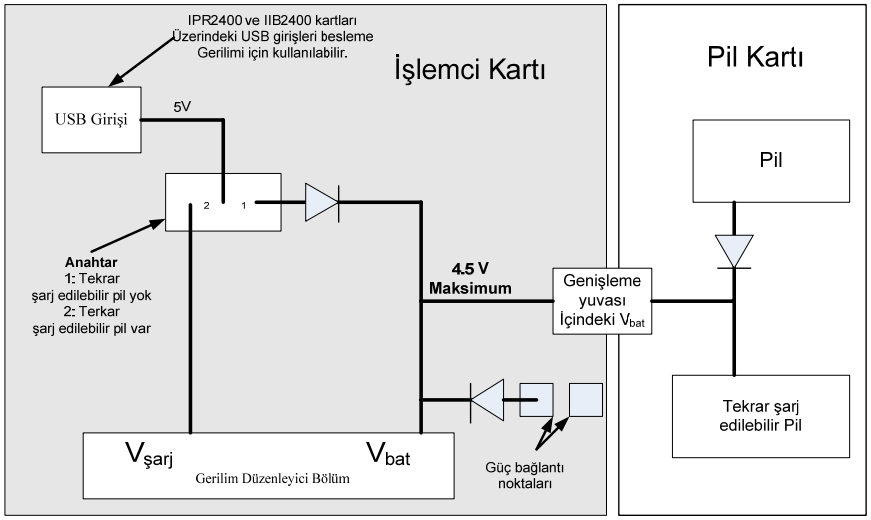


Şekil 4.4: PXA271 mikrodenetleyicisinin iç modülleri

PXA271 mikrodenetleyicisi, ses ve veri işleme uygulamalarındaki yüksek veri işleme ihtiyacına cevap verebilmek için bir de MMX önişlemcisine sahiptir. Bu işlemci ile 30 yeni komut tanımlanmış ve Intel®'in MMX ve SSE komut yapısı ile de uyumluluğu sağlanmıştır. (Şekil 4.4)

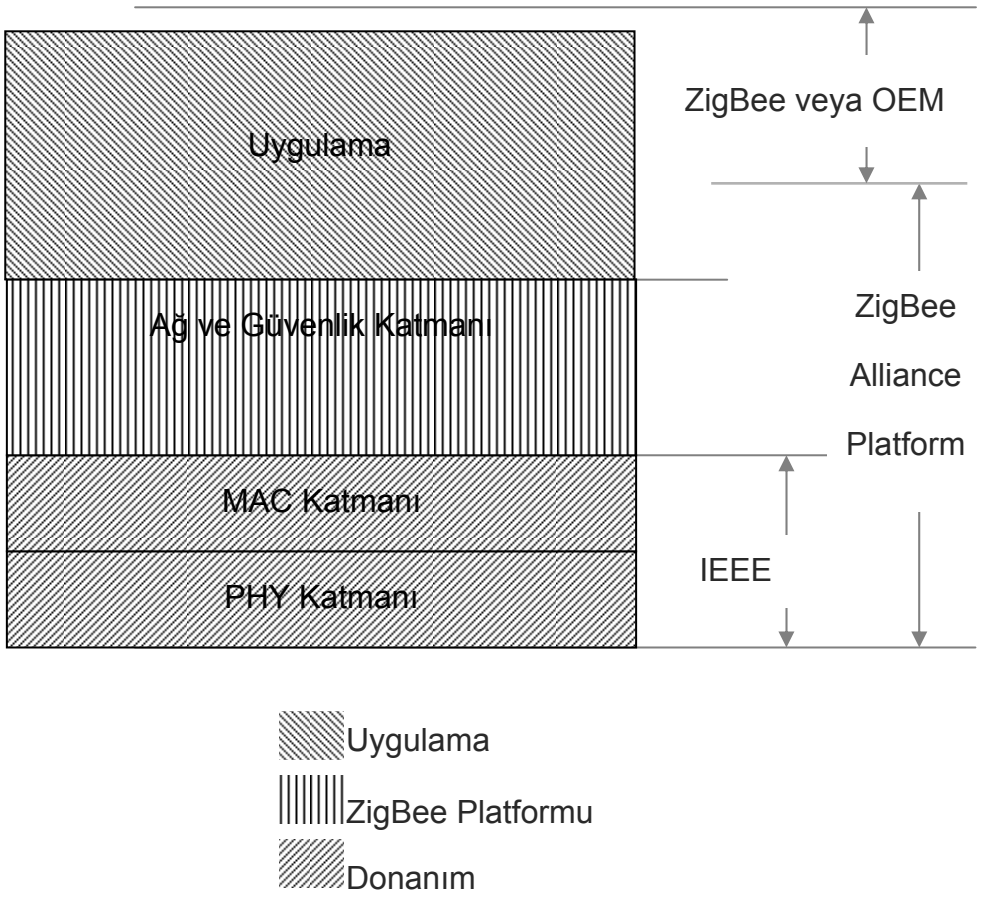
İşlemci kartı için gerekli dinamik olarak ayarlanabilir besleme gerilimi ve işlemci kartına bağlanan algılayıcı kartları için gerekli besleme gerilimi (1,8 ve 3 V) Dialog firmasının DA9030 gerilim düzenleyici modülü tarafından sağlanmaktadır. Gerilim

düzenleyici tümleşik devre, USB bağlantısından, 3 adet AAA boyutundaki pillerden, tekrar şarj edilebilir pillerden veya V_{bat} bacağına bağlanan bir güç kaynağından gerekli duyulan gerilim seviyelerini üretmektedir. Aşağıdaki Şekil 4.5'de Imote2 platformunun güç bağlantı şeması verilmiştir.



Şekil 4.5: Güç bağlantı şeması

Imote2 Platformunda haberleşme alıcı/verici modülü olarak, IEEE 802.15.4 standardını destekleyen CC2420 entegresi kullanılmıştır. IEEE 802.15.4 standardı, kontrol ve çevresel koşulları izleme uygulamaları için düşük güç tüketimli ve düşük kapsama alanına sahip radyo alıcı/vericileri için belirlenmiş bir standarttır. Sayısal doğrudan dizilerle yayılı spektrum çoğullaması [32] kullanarak, 16 kanalda toplam 250kb/s veri hızına ulaşılabilir. Bu çoğullama yönteminde, gönderilmek istenen bilgi işaretindeki her bir bit, 13 bitlik özel bir kod ile çarpılmaktadır. Çarpım sonucunda veri oranı artmakta ve işaret frekans spektrumunda yayılmaktadır. Alıcıda gelen sinyalin tekrar elde edilebilmesi için tekrar 13 bitlik kod ile çarpılmalıdır. Alıcıda yapılan çarpma işlemi sonucunda dar bantta olan bozucu ve boğucu işaretler de yayıldığından haberleşme güvenliği artmaktadır. Tüm kullanıcılar aynı frekansı paylaşırlar ve sadece kod sözcüklerinin uzunlukları arttırılarak, kullanıcı sayısının arttırılması mümkündür.

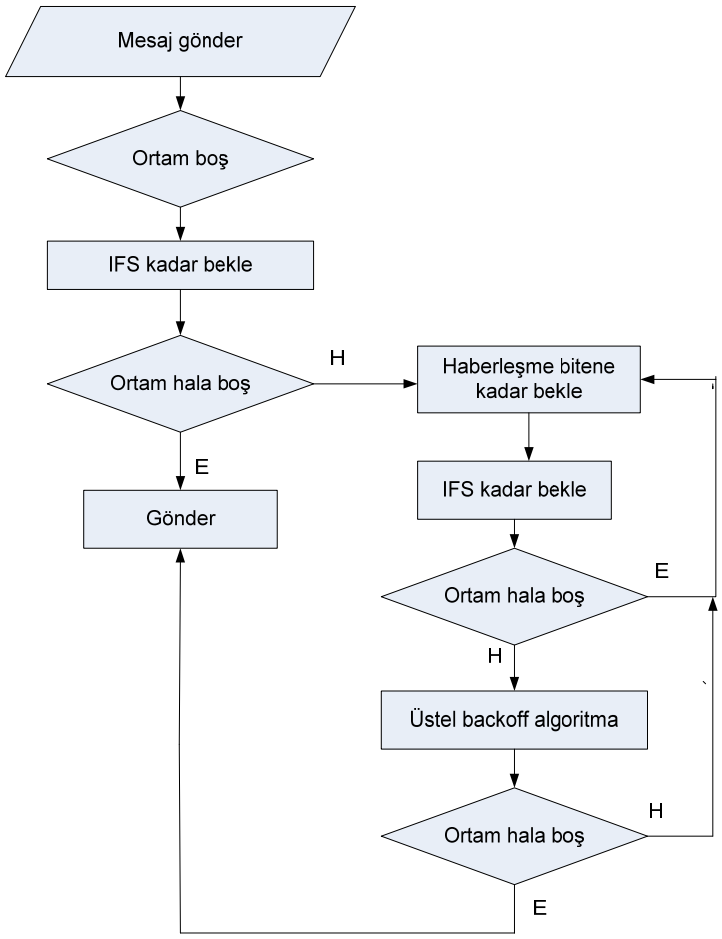


Şekil 4.6: ZigBee mimarisi

ZigBee platformunun, Şekil 4.6'da gösterildiği üzere fiziksel katmanı ve MAC katmanı IEEE standardı olarak belirtilmiştir. Ağ ve Güvenlik Katmanları ise, ZigBee Alliance tarafından belirlenmiştir. Bu katmanlar TinyOS işletim sisteminin içine adapte edilmiş ve modül olarak uygulama geliştirmek isteyenlere sunulmuştur.

ZigBee, MAC katmanında CSMA/CA (Şekil 4.7) protokolünü kullanır. Haberleşme kanalına Beacon çerçeveleri kullanarak veya kullanmayarak erişilebilir. Beacon çerçevesi kullanılırsa, yönetici birim, belirli aralıklarla bulunduğu ortama beaconlar gönderir. Kanalın denetimini elinde bulundurur. Kimin ne zaman konuşacağını yönetici belirler. Böylece band genişliği ihtiyacı önceden belirlenebilir. Beacon süreleri önceden tanımlanacağı için, yönetici birimin de güç tüketimi önceden hesaplanabilir. Büyük veri aktarımları sağlanacağı durumlarda başlık dosyalarının tekrar gönderilmesini engelleyen süper çerçeve olanağına da olanak sağlar. Beacon

süreleri 15ms ile 252s arasında ayarlanabilir. Beacon gönderilmeden erişim ise, en çok bilinen ve kullanılan yöntemdir. Ulaşan paketler için pozitif onaylama mesajı gönderilir.



Şekil 4.7: CSMA/CD protokolü

Imote2 platformu, haberleşme modülünün kullanımı için 30 metreye kadar kapsama alanı sağlayabilen yüzey montajlı bir anten bulundurur. Daha yüksek bir kapsama alanı istendiği takdirde, platform üzerinde harici bir anten bağlanabilmesi için gerekli SMA bağlantı noktası da bulunmaktadır. Aynı şekilde, harici haberleşme modülleri kullanımına destek vermek amacıyla PXA271 mikrodenetleyicisi, 802.11 kablosuz internet, 802.15.3 mavi diş arayüzleri de donanım seviyesinde desteklenmektedir. CC2420 haberleşme modülü, paket işleme, veri tamponlama, veri şifreleme (AES–128), veri onaylama, daha az kullanılan kanalın bulunması ve bu kanala atlama, paket

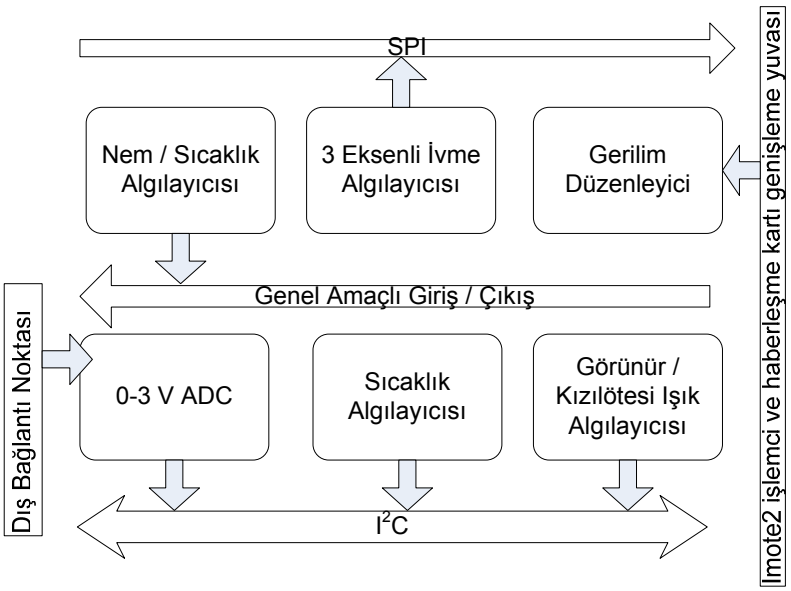
zamanlaması, gelen sinyalin gücünü tespit edebilme gibi görevleri kendi üzerinde yapmakta ve böylece ana işlemciye binen yükü azaltmaktadır.

4.2 ITS400 – Algılayıcı Kartı

Kablosuz duyarga ağları için düşük güç tüketim ihtiyacı göz önünde bulundurularak Intel® tarafından geliştirilmiş ve Crossbow® tarafından üretilmiş Imote2 platformu ile de uyumlu bir algılayıcı kartıdır. (Şekil 4.8)

- ST Micro LIS3L02DQ 3D 12 bit ±2g SPI ve I^2C arayüzlerini destekleyen ivmeölçer
- Sensirion SHT15 14 bit sıcaklık ±0.3 ^oC ve 12 bit nem algılayıcısı
- TAOS TSL2651 Görünür / Kızılötesi 16 bit Işık Algılayıcısı
- Maxim MAX1363 4 kanallı 0-3 V I^2C arayüzü destekleyen ADC
- TI TMP175 ±1.5 ^oC Sayısal Sıcaklık Algılayıcısı

  (Çalışma Aralığı: -25 ^oC'den 85 ^oC'ye kadar)



Şekil 4.8: ITS400 algılayıcı kartının şeması

5. KABLOSUZ DUYARGA AĞLARI İLE YER TESPİT ALGORİTMASI

Bu çalışmada hedeflenen amaç, kablosuz duyarga ağları kullanılarak, belirli bir alana dağıtılmış düğümler ile o alana giren canlıların veya cansız nesnelerin yerlerinin tespit edilmesidir. Bu amaç doğrultusunda ilk yapılması gereken, algılanmak istenen nesnelerin bulundukları ortamda yaptıkları değişikliklerin neler olduklarının belirlenmesidir. Bu çalışmada, nesnelerin ışık, ses ve titreşim kaynağı olarak davrandıkları varsayılacaktır. Böylece, ışık, ses ve titreşim algılayıcılarından bir veya birkaçı kullanılarak, kablosuz duyarga ağı yardımıyla nesnelerin yerlerinin tespit edilmesi öngörülmüştür.

5.1 Giriş

Nesnelerin 3 farklı kaynak gibi davrandıkları kabul edilerek, ışık, ivme ve mikrofon algılayıcıları olmak üzere üç çeşit algılayıcı kullanılmıştır.

Literatürde nesnelerin yerlerinin belirlenmesi için çeşitli yöntemler kullanılmaktadır. Bu yöntemler nesnelerin hangi seviyede tanımlanmak istendiğine, nesnelerin fiziksel ve kimyasal yapılarına göre çeşitlilik göstermektedir. Örnek olarak, görüntü işleme yöntemini kullanarak nesnenin yeri belirlenirken, arka fon ile aynı renkteki bir nesneyi ayırmak da veya çok fazla benzeri nesne içeren bir ortamda istenilen nesnenin yerinin tespit edilmesi olanaksızlaşmaktadır. Aynı şekilde büyük bir alan denetlenmek istediğinde, gerekli cihazların kurulum maliyetleri çok yüksek olmaktadır. Farklı bir örnek olarak, çevresindeki ortamdan farklı manyetik özellikleri olan bir nesneyi tespit etmek için manyetik sensörler kullanılabilir. Bu nesnenin şekilsel özellikleri hakkında bilgi veremezken algılama maliyetini çok azaltmış olur.

Bu çalışmada, belirli bir alandaki sensör birimleri kullanılarak, o alana giren bir canlının veya nesnenin yerinin tespit edilmesi amaçlanmıştır. Canlı veya nesne, bir titreşim, kızılötesi ışık ve ses kaynağı olarak kabul edilmiştir. Bu varsayıma göre,

canlının veya nesnenin çevrede oluşturduğu bu değişimler ilgili algılayıcılar tarafından algılanabilir ve ortama bir nesnenin girip girmediği belirlenebilir.

Nesnenin yerinin tespit edilebilmesi içinse, algılayıcılardan alınan verilerin zaman ekseninde korelasyona tabi tutulması gereklidir. Işık ve ivme sensörlerinde, ışığın ve titreşimin ortamda yayılma hızı çok yüksek olduğundan, sensör birimlerine gelen veriler arasındaki fark çok küçük olacaktır. Bu durumda bu küçük zaman farkında korelasyon yapılabilmesi için çok hassas algılayıcılar ve hızlı işlemciler kullanılmalıdır. Ses ise diğer verilere göre daha yavaş yayılmaktadır. Bu nedenle bu çalışmada farklı sensör birimlerinden alınan ses sinyali korelasyona sokularak, ses kaynağının yeri tespit edilmeye çalışılmıştır. Nesnenin, ışık ve titreşim kaynağı olma özellikleri ise, ses verisinin yanında doğrulayıcı etkenler olarak yer alması düşünülmüştür.

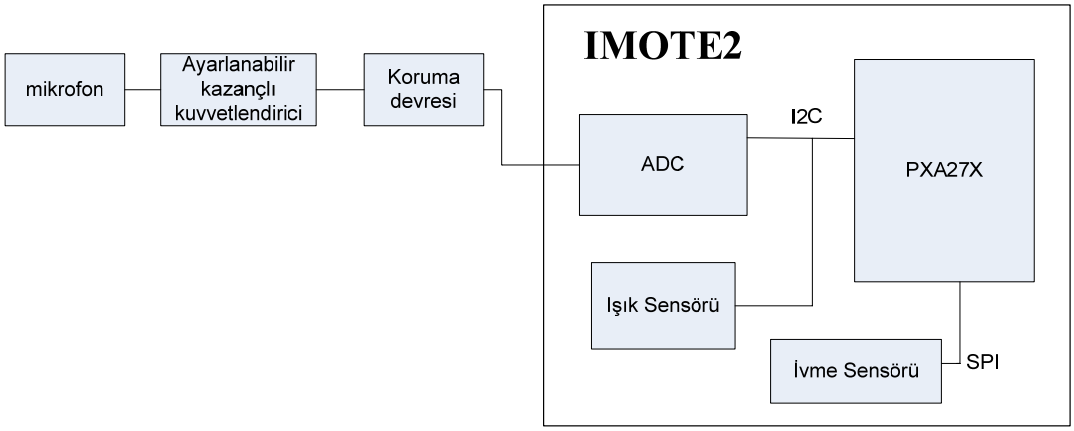
Sesin tespit edilebilmesi için, imote2 modüllerinde ses sinyalinin zarfı bulunmuş ve bulunan zarf belirli bir eşik değeri ile karşılaştırılarak sesin var olup olmadığı belirlenmiştir. Ses kaynağının yerinin belirlenebilmesi içinse, zarfın bulunduğu anın zamanı, ana bilgisayara kablosuz haberleşme ile gönderilmiştir. Korelasyonun doğru alınabilmesi için, her bir sensör biriminde, kablosuz duyarga ağına ait global bir zaman tutulmuştur. Böylece her düğümün zarfı bulduğu anı, bu global zamana göre belirleyip ana bilgisayara göndermesi sağlanmıştır.

5.2 Deneyde Kullanılan Sistem

Deneyde, ses verilerin kullanılarak yer tespiti yapılacağından, her bir Imote2 modülüne, bir mikrofon alıcısı eklenmiştir. Ses sinyalinin imote2 modülüne hangi yönden geleceği önceden tespit edilemeyeceği için, mikrofon olarak yönsüz bir mikrofon kullanılmıştır.

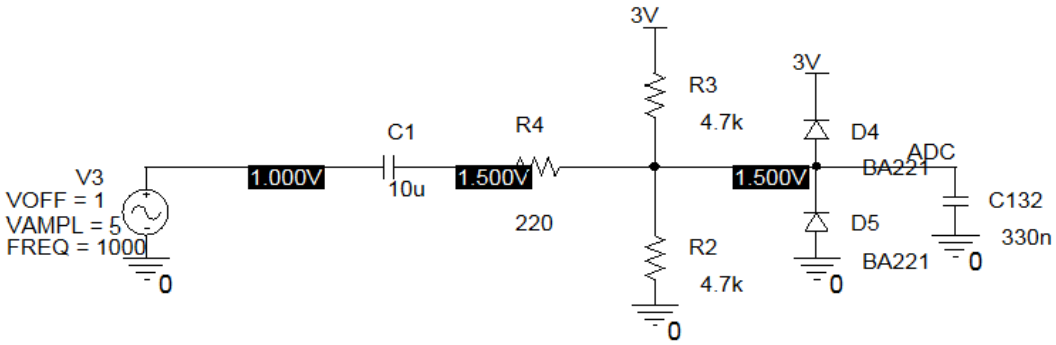
Mikrofondan çıkan verilerin sayısala çevrilmesi için Maxim® firmasının MAX1363 modelli ADC'si kullanılmıştır. Bu ADC, 4 kanallı, 12bit çözünürlükte, 100ksps çevirme hızına sahip, 0-3V elektriksel seviyede analog-sayısal çevrim yapabilmektedir. I^2C haberleşme yolu ile mikrodenetleyici ile haberleşmektedir.

Mikrofonun çıkışının ADC tarafından okunabilmesi için, mikrofonun çıkışı ayarlanabilir kazançlı bir kuvvetlendirici ile gerilim seviyesi ADC'nin giriş gerilimi olan 0-3V gerilim seviyesine yükseltilmiştir. Şekil 5.1'de Imote2 platformu ve üzerinde kullanılan sensörler gösterilmiştir.



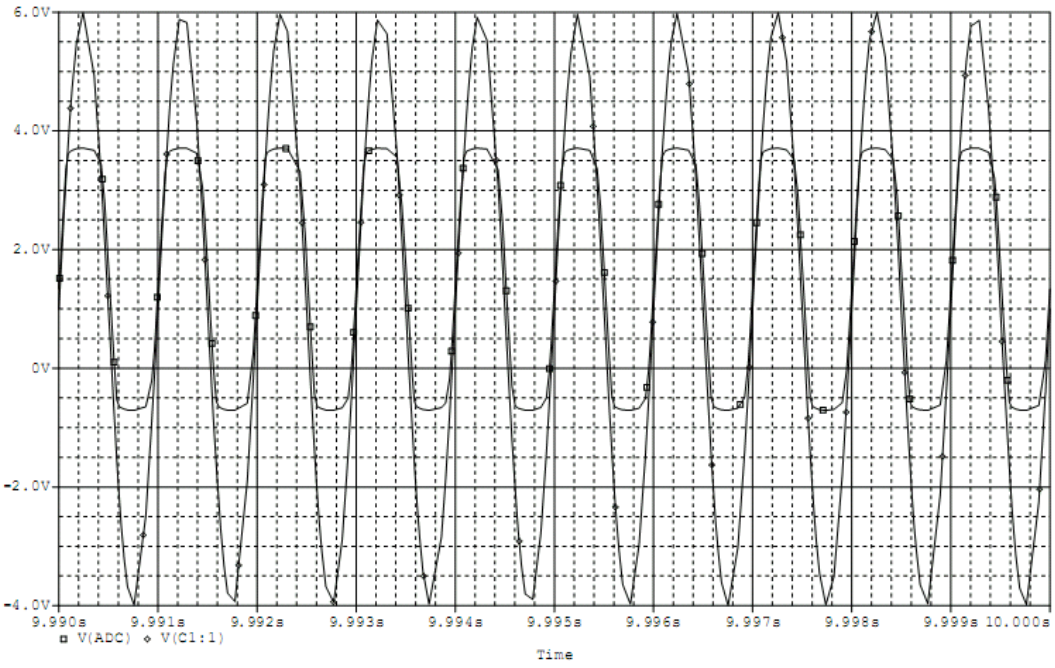
Şekil 5.1: Sensör Birimi

Ayarlanabilir kazançlı kuvvetlendiricinin çıkış gerilimi, deney öncesinde osiloskop yardımıyla belirli bir uzaklıktaki sesi, 3V maksimum genliğe kuvvetlendirecek şekilde ayarlanmıştır. Bu noktada, kuvvetlendiricinin kazancı yeni ayarlanan değerde sabit olacağından, daha yakın mesafeden gelen sinyalleri 3V genlik seviyesinin üzerine de kuvvetlendirebilecektir. MAX1363 entegresinin maksimum dayanabileceği giriş gerilimi 6V olarak verilmiştir. Giriş voltajının bu gerilimi geçmemesi için kuvvetlendirici ile ADC arasına koruyucu bir devre eklenmiştir. (Şekil 5.2)



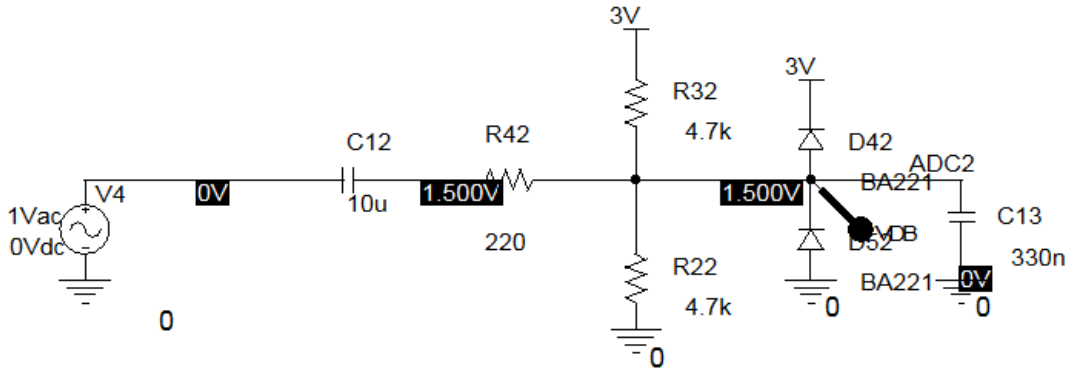
Şekil 5.2: Koruma devresi ve DC analiz parametreleri

Koruma devresinde, kuvvetlendiricinin çıkışı ±1.5V olduğundan, bu gerilim öncelikle ADC'nin çevirme aralığı olan 0-3V'a gerilim bölücü dirençlerle (R3 ve R4) yükseltilmiştir. Devrenin 3V'un üzerine çıkmaması ve 0V'un altına düşmemesi için de, D4 ve D5 schottky diyotları ADC girişine bağlanmıştır. Entegre girişlerinde kullanılan diyotların schottky olması nedeniyle, D4 ve D5 diyotları da schottky diyot seçilmiştir. Aksi takdirde, schottky diyotlar normal diyotlardan daha önce iletime geçip, entegrenin girişinin yine yanmasına neden olacaktır. R4 direnci ise, gerilimin beklenen seviyenin üzerine geçmesi halinde, schottky diyotlarından geçecek olan akımı sınırlamak için konmuştur.



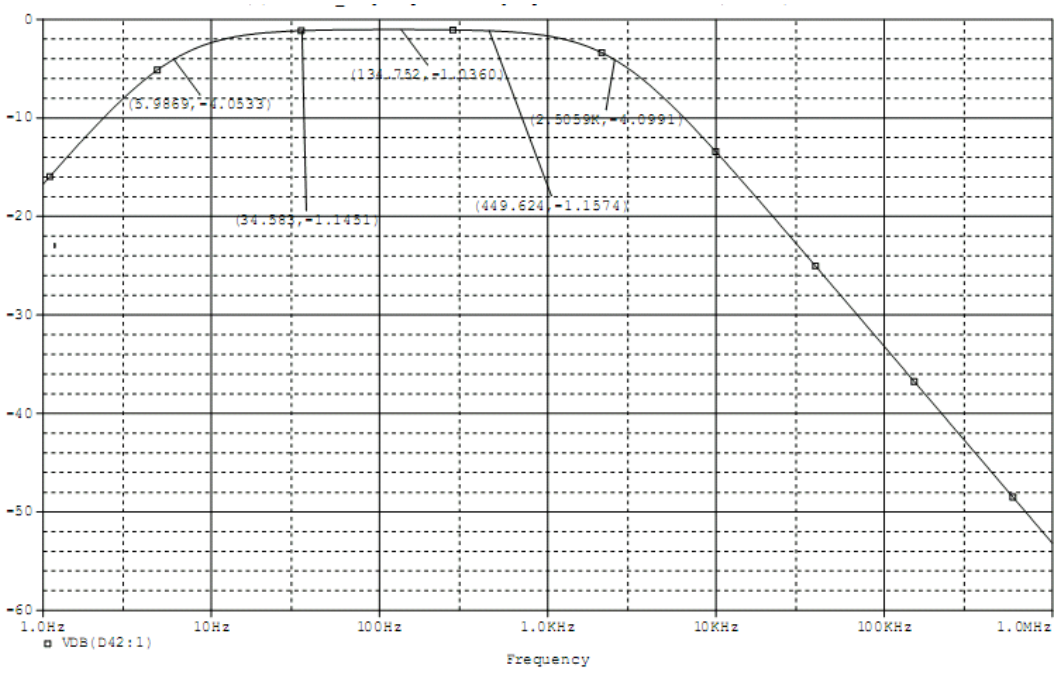
Şekil 5.3: Koruma devresinin giriş/çıkış gerilimleri

Şekil 5.3'de koruma devresinin girişine uygulanan 1+5sinwt sinyaline karşılık çıkış gerilimi grafiği verilmiştir. Beklenildiği üzere, çıkış gerilimi 3.6V'un üzerine çıkmamaktadır.



Şekil 5.4: Koruma devresinin frekans analizi

Koruma devresinin frekans analizinin yapıldığı şematik Şekil 5.4'de gösterilmiştir.



Şekil 5.5: Koruma devresinin frekans cevabı

Koruma Devresinin frekans cevabının simülasyon çıktısı Şekil 5.5'de verilmiştir. Devre, -3dB alt kesim frekansı 5Khz, -3dBüst kesim frekansı 2Khz olan bir bant geçiren bir filtre gibi davranmaktadır. Ses frekansının 2-20Khz arasında olduğu bilindiğinden, duyulabilir frekans bandında bir miktar kayba neden olmaktadır. Bu çalışmada, örnekleme frekansı olarak 1Khz seçildiğinden, daha üstündeki frekanslarda oluşan zayıflamalar sistemin çalışmasında herhangi bir bozucu etkene neden olmayacaktır.

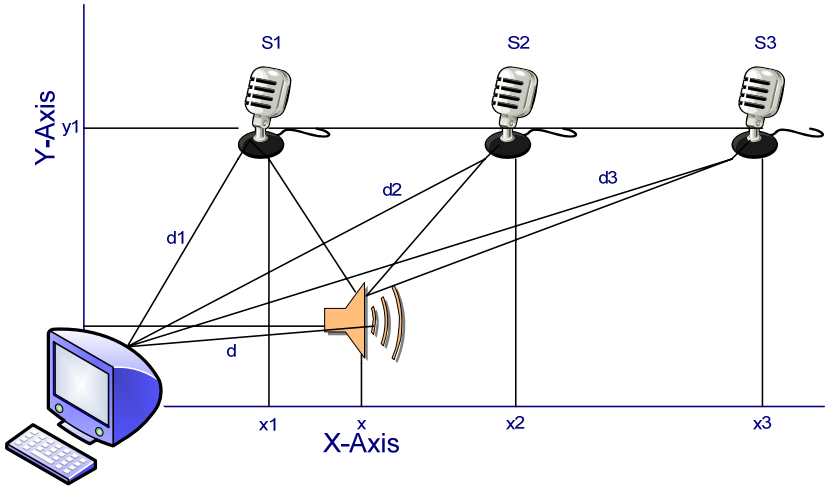
Mikrofon sensörüne ek olarak, kablosuz duyarga biriminin bulunduğu yerdeki titreşimleri ölçmek için kullanmak amacıyla STMicroelectronics© firmasının LIS3L02DQ olarak adlandırılmış ivmeölçeri kullanılmıştır. İvmeölçer, mikrodenetleyiciye SPI arayüzü ile bağlanmıştır. SPI arayüzünün kullanılabilmesi içinse TinyOS işletim sisteminde ilgili sürücüler yazılmıştır.

İvmeölçerden gelen veri, kablosuz duyarga düğümünün bulunduğu bölgede düğümü etkileyebilecek boyutta büyük bir hareket olup olmadığını gösterir. Sadece ivmeölçer verisi ile hareketi oluşturan nedenin gerçekten bir canlı mı yoksa çevresel koşullar mı olduğu tespit edilemez.

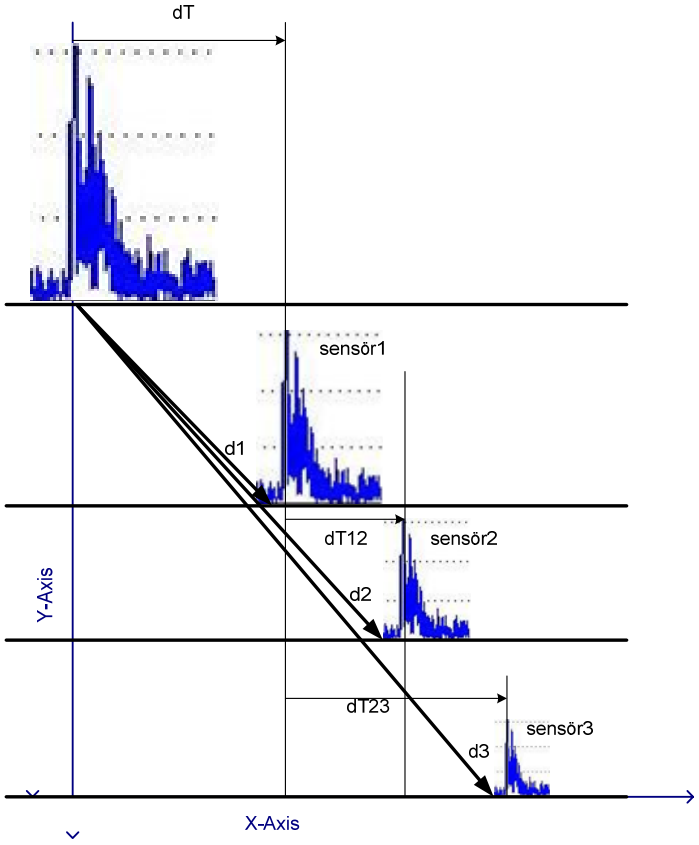
ITS400 algılayıcı kartının üzerinde TAOS© firmasının TSL2560 kızılötesi ve görünür ışık algılayıcısı bulunmaktadır. Bu algılayıcı, ışık yoğunluğunu içindeki ADC yardımıyla dijital sinyale dönüştürebilmekte ve I2C veri yolu üzerinden gönderebilmektedir. Algılayıcının ışık yoğunluğunu algılamasını sağlayan, geniş bir banda (kızılötesi ışık ve görünür ışığı kapsayan) sahip bir foto diyottur.

5.3 Tespit Algoritmasının Uygulanması ve Deney Sonuçları

Çalışmada kullanılan senaryo Şekil 5.6'da gösterilmiştir. Burada S1 sensör biriminin, ses kaynağını ilk olarak duyduğu varsayılarak T saniye olarak nitelendirilmiştir. Anabilgisayar ile ses kaynağı arasındaki uzaklık d metre olarak gösterilmiş, ses kaynağının konumu (x,y) olarak verilmiştir. Sensör birimleri S1, S2 ve S3'ün koordinatları ise sırasıyla, (x_1,y_1), (x_2,y_2) ve (x_3,y_3)'dür



Şekil 5.6: Çalışmada kullanılan senaryo



Şekil 5.7: Ses kaynağından gelen sinyallerin sensörler ve anabilgisayara erişim zamanlamaları

Ses sinyali, hareket eden bir dalga olarak düşünüldüğünde;

$$f(d,t,v) = K(t) \cdot f(t) \tag{5.1}$$

olarak modellenebilir. Burada K(t) zamanla azalan genlik değerini, f(t) ses sinyalini, v sesin havada yayılma hızını nitelemektedir. Şekil 5.7'de, T süresi, S1 sensör biriminden d kadar uzakta oluşan bir sinyali göstermektedir. Burada T süresinin bilinmemekte fakat T_{12} ve T_{13} süreleri ölçülebilmektedir.

Bu tanımlamalara göre aşağıdaki denklemler yazılabilir:

$$d_1 = v \cdot T$$

$$d_2 = v \cdot (T + dT_{12})$$

$$d_3 = v \cdot (T + T_{23}) \tag{5.2}$$

2 numaralı denklem takımındaki denklemleri kullanarak, sensörler ile ses kaynağının koordinatları arasında aşağıdaki bağıntılar kurulabilir:

$$(x - x_1)^2 + (y - y_1)^2 = (v \cdot T)^2 \tag{5.3}$$

$$(x - x_2)^2 + (y - y_2)^2 = (v \cdot (T + T_{12}))^2 \tag{5.4}$$

$$(x - x_3)^2 + (y - y_3)^2 = (v \cdot (T + T_{23}))^2 \tag{5.5}$$

(5.3) ve (5.4) numaralı denklem takımlarından (5.6) numara, (5.3) ve (5.5) numaralı denklem takımlarından (5.7), (5.4) ve (5.5) numaralı denklem takımlarından (5.8) numaralı denklem takımları oluşturulabilir.

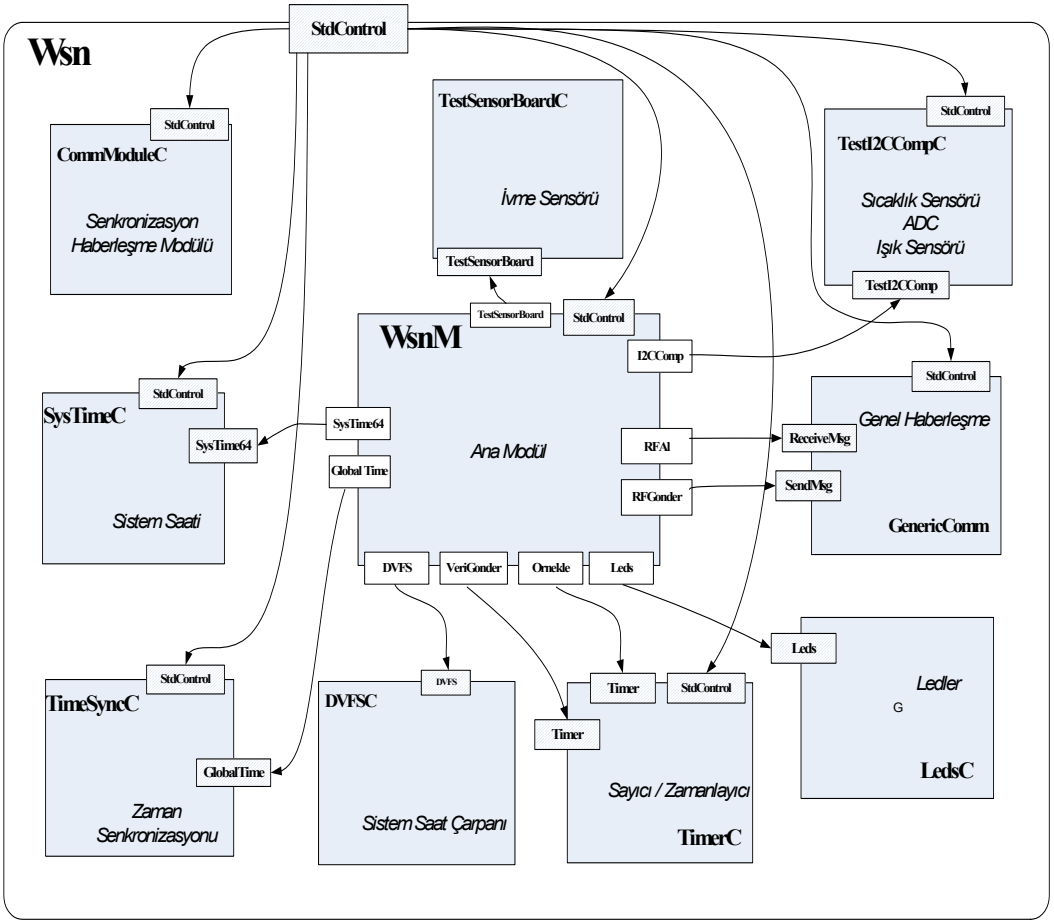
$$2(x_2 - x_1)x + 2(y_2 - y_1)y + 2v^2 T_{12} T = -v^2 T_{12}^2 - x_2^2 - y_2^2 + x_1^2 + y_1^2 \tag{5.6}$$

$$2(x_3 - x_2)x + 2(y_3 - y_2)y + 2v^2 (T_{23} - T_{12})T = -v^2 T_{12}^2 + v^2 T_{23}^2 - x_3^2 - y_3^2 + x_2^2 + y_2^2 \tag{5.7}$$

$$2(x_1 - x_3)x + 2(y_1 - y_3)y + 2v^2 T_3 T = v^2 T_{23}^2 - x_1^2 - y_1^2 + x_3^2 + y_3^2 \tag{5.8}$$

Denklemlerden görüldüğü üzere, x,y ve T parametreleri bilinmeyen(ölçülemeyen) parametrelerdir. Buna karşın, sağ taraftaki parametrelerin hepsi ise ölçülebilen parametrelerdir. Buradan yola çıkarak, en küçük kareler kestirimi yöntemiyle bilinmeyen parametreler hesaplanmıştır.

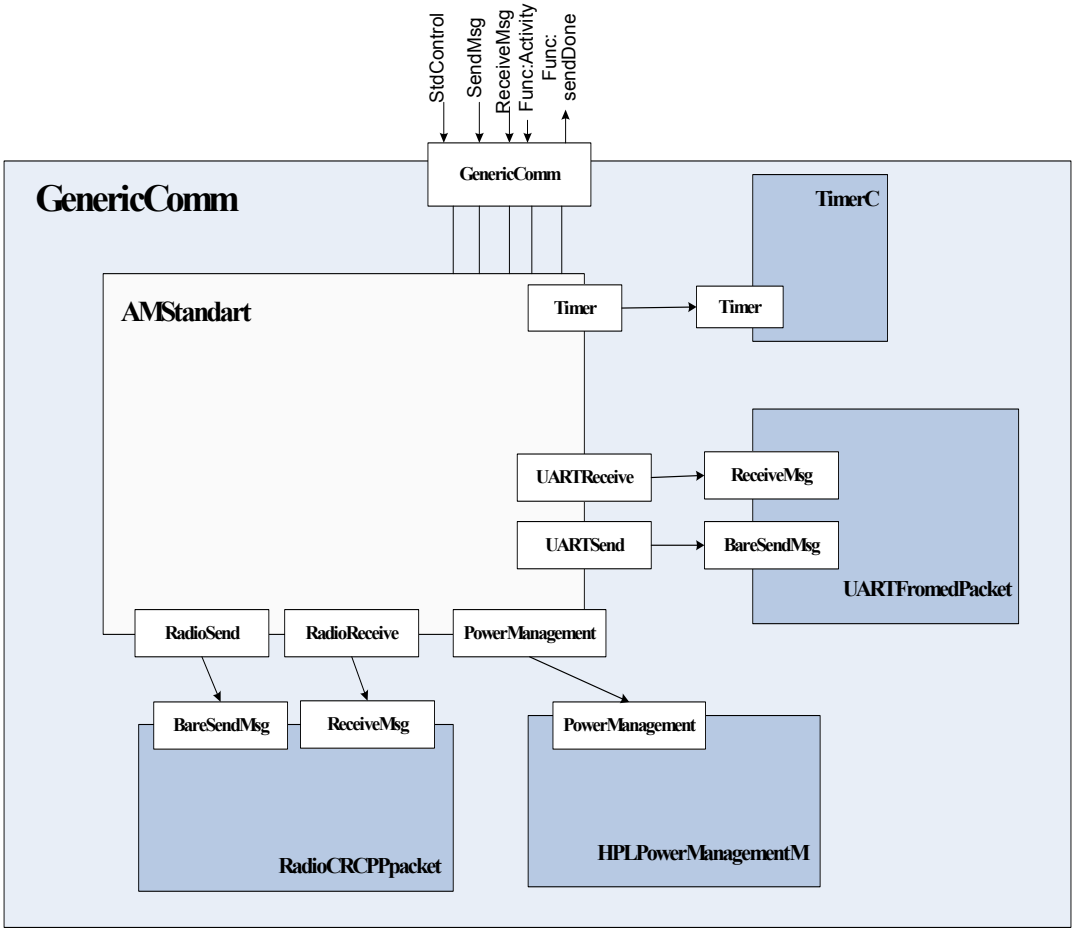
Bu çalışmada kullanılan TinyOS işletim sisteminin mimarisine uygun bir şekilde; program kodları modül şemaları ve çalışan ana programın algoritması olarak verilmiştir.



Şekil 5.8: NesC konfigürasyon dosyası

Şekil 5.8'de verilen konfigürasyon dosyası, modülleri ve bileşenleri bir araya getirerek, bu çalışmanın esas amacını yerine getirecek olan proje dosyasını oluşturur. Burada, sensörlerden veri toplamak için yazılan bileşenler ve zaman senkronizasyonu için gerekli bileşenler daha detaylı olarak da incelenmiştir. WsnM modül dosyası içinde, sistemin çalışması için gerekli kesmeler başlatılmış, ADC verisi alma ve zarf algoritmasını koşturma işlemleri yapılmıştır. Zarfın bulunduğu zamanı belirlemek için,

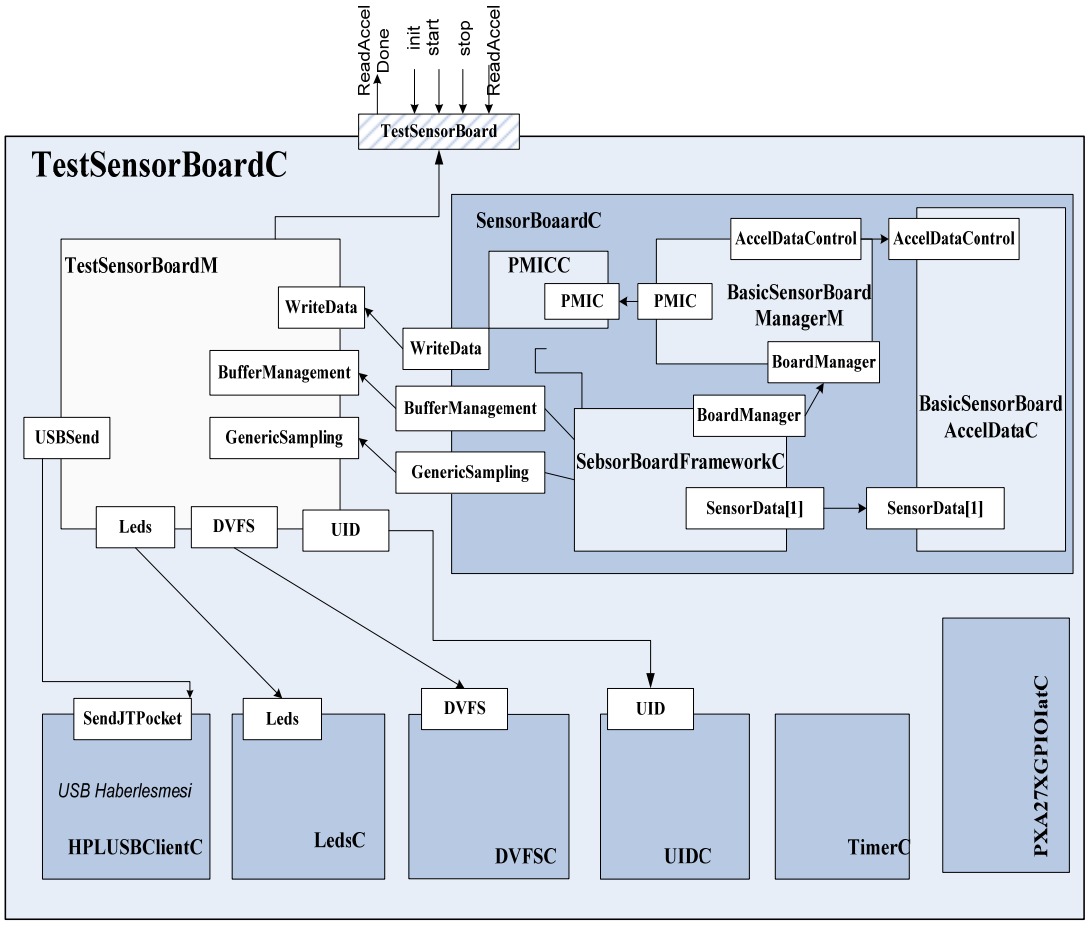
TimeSyncC bileşeni WsnM modülüne GlobalTime arayüzünü sağlamıştır. TimeSyncC konfigürasyonu, global zamanı belirlemek için gerekli algoritmanın koşmasını sağlayacak modül ve bileşenleri de birbirine bağlayan bir altyapıya sahip olacak şekilde tasarlanmıştır. DVSFC bileşeni, sistemin saat çarpan hızının değiştirilmesini ve öğrenilmesini sağlayan DVSF arayüzünü sağlamaktadır. Zamanlayıcıların kurulabilmesini ve koşullar oluştuğunda kesme üretmesini sağlayan TimerC bileşenidir. Radyo frekansı ve kablolu haberleşme hizmetleri WsnM modülüne GenericComm bileşeni ile sağlanmıştır. Işık, sıcaklık ve ADC sensörlerini üzerinde bulunduran I2C veri yolunu kontrol etmek için TestI2CCompC bileşeni yazılmıştır. İvme sensörünün bulunduğu SPI veri yolunun kontrolü ise TestSensorBoardC bileşeni ile sağlanmaktadır.



Şekil 5.9: GeneriComm bileşeni

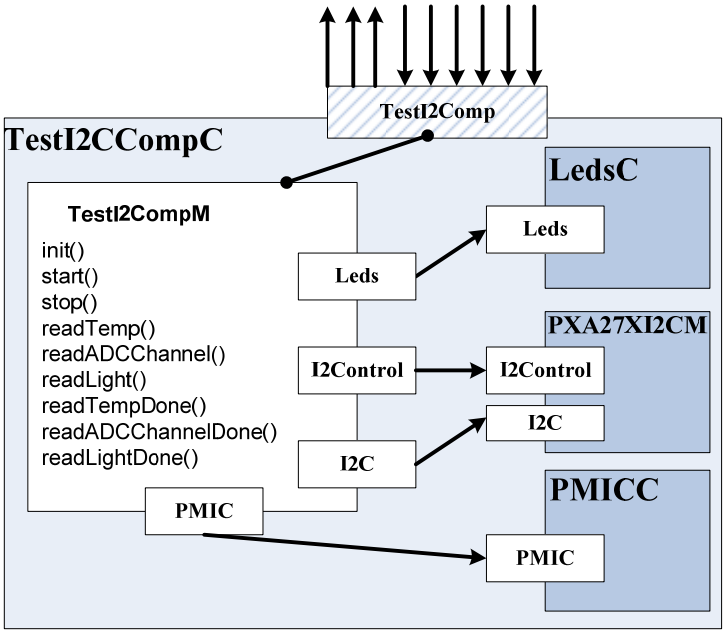
Şekil 5.9'da çalışmanın kablosuz haberleşme kullanılarak toplanan verilerin ana bilgisayara aktarılmasını sağlayan haberleşme bileşeni gösterilmiştir. Bu bileşen

kullanılarak, gerek UART gerek radyo kanalına TOS_Msg formatında paketler gönderilebilir. Bileşen, dışarıya mesaj gönderme, dışardan mesaj alma, mesaj aktivitelerini sayma gibi özellikler sunar.



Şekil 5.10: TestSensorBoardC bileşeni

TestSensorBoardC bileşeni (Şekil 5.10), WsnM modülüne ivme sensöründen veri okuyabilmek için gerekli, olan iki adet fonksiyon sunmaktadır. Bunlar dışındakiler, TestSensorBoardC arayüzünün özelliklerini değiştirmek için kullanılmıştır. Bu bileşen direk olarak kendisi SPI veri yolunu kullanmamaktadır. CC2420 ZigBee entegresine veri göndermek/almak ve başka işler için de kullandığından SensorBoardC bileşeni vasıtasıyla ivme verilerini okuyabilmektedir.



Şekil 5.11: TestI2CCompC bileşeni

TestI2CCompC bileşeni, bu çalışmada kullanılacak olan sensörlerden ışık, ADC ve sıcaklık sensörlerinden veri alınabilmesi için, WsnM modülüne gerekli hizmetleri sunmaktadır. (Şekil 5.11)

Her bir sensör biriminde zaman, üzerlerinde bulunan kristal osilatör kullanılarak tutulmaktadır. Sensör birimleri birbirlerinden bağımsız olarak çalıştıklarından osilatörlerinin aralarında bir faz bulunmaktadır. Her kristal osilatör tipik olarak 20ppm toleransa sahip olarak üretilmektedir. Bu da milyonda 20 kayma anlamına gelmektedir. Kristal osilatörler gerçekte olduğundan farklı frekanslarda çalışırlar.

Bu çalışmada olduğu gibi, birbirinden fiziksel olarak bağımsız çalışabilen sensör birimlerinin verileri arasında zaman korelasyonu önemliyse, sensör ağında tanımlı tek bir zamana (global zaman) ihtiyaç bulunmaktadır. Bu amaç doğrultusunda, (Şekil 5.14) ile gösterilen bileşen hazırlanmıştır. Zaman senkronizasyonu tasarımında önemli seçim kriterlerinden biri, senkronizasyonun doğruluğudur. İlaveten, senkronizasyon protokolünün, işlemciye getirdiği ek yüklerin de az olması düşük güç tüketimli bir uygulama geliştirildiği göz önüne alındığında önem kazanmaktadır.

Senkronizasyon algoritmalarını kullanan sistemler genelde iki farklı yöntemden birini kullanmaktadırlar: Senkronize bir ağda global zamanın kullanımı ve senkronize

olmayan bir sensör ağında global zamanın kullanımı. Eğer sensör birimleri, sürekli uyku modundaysa ve harici bir dış tetikleyicinin belirli bir eşik değeri geçmesi sonucu kesme ile uyanıyorsa, bu ilk söylenen yöntem şeklinde demektir ve uyandığında çabuk bir şekilde sensör ağına senkronize olmalı ve gerekli örnekleme işlemlerine başlamalıdır. Burada bir sensör ağının global zamanına senkronize olmak isteyen bir senkronizasyon modülünün, senkronizasyon için harcadığı süre çok önemlidir.

Bu çalışmada, kullanılan yöntem, ikinci yöntem olarak anılan ve pratikte daha fazla uygulamaya sahip olan, düğümlerin sürekli uyanık kalıp, sensör ağıyla senkronizasyonlarını koparmamaları şeklinde olandır. Böylece örnek alındığı anda, senkronizasyon da olduğundan, sensör ağının global zamanı hemen kullanılabilecektir.

Bu çalışmada kullanılan zaman senkronizasyonu yöntemi, temelde Vanderbilt's FTSP protokolüne dayanmaktadır. Ancak bazı bileşenler aşağıdaki iki özelliği içerebilmesi için yeniden yazılmıştır:

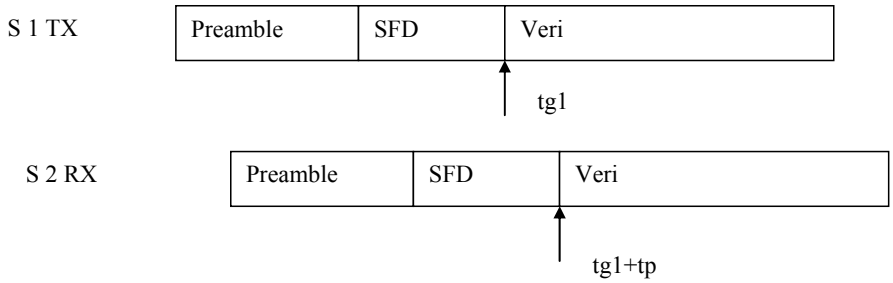
- Sensör ağına yeni katılan veya senkronizasyonunu kaybeden bir sensör biriminin daha çabuk senkronize olması için hızlı senkronizasyon modu
- Düğümler arasındaki faz ve frekans farkının daha hassas belirlenmesini sağlayan geliştirilmiş model.

Zaman senkronizasyonunun amacı, bütün düğümlerin kendilerinde yerel olarak tuttuğu zamanı, merkez sensör biriminin yerel zamanına eşitlemektir. Tipik olarak, eşitlemek şeklinde değil, gerçek zaman ile yerel zaman arasındaki denklemin katsayılarını tespit etmek şeklinde olmaktadır.

$$\text{yerel Zaman} = \text{kayıklık} \times \text{global Zaman} + \text{sapma} \qquad (5.9)$$

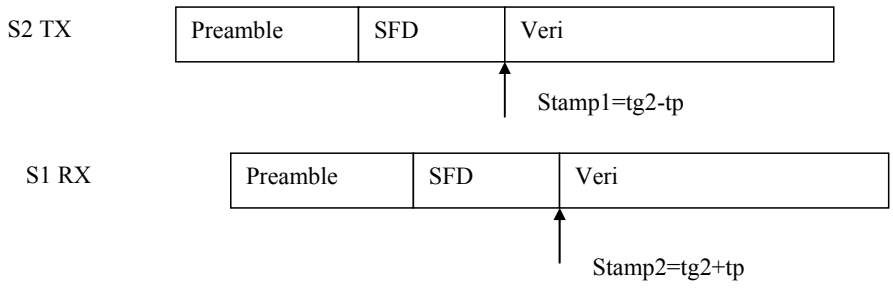
Eğer bir şekilde kayıklık ve sapma değerleri hesaplanabilirse, yerel zaman her zaman için global zamana kolayca dönüştürülebilmektedir.

Tipik olarak, gönderilen bir mesajın gönderildiği andaki global zaman ile alındığı andaki global zaman birbirine eşit değildir. Havada işaretin çok hızlı yayıldığını göz önüne alarak, havadaki yayılma süresini ihmal edebiliriz fakat haberleşme entegresinin içinde yani donanım da yayılma süresi ihmal edilemeyecek kadar fazladır. (CC2420 için 3000 saat çarpanı kadar)



Şekil 5.12: TX ve RX arasındaki gecikme

Bu süre Şekil 5.12'de gösterildiği gibi tp olarak adlandırılsın. Böylece alıcı sensör birimi global zamanın tp kadar gerisine senkronlanacaktır.



Şekil 5.13: RX ve TX arasında gecikme

Bir sonraki adımda (Şekil 5.13) S2 sensör birimi, S1 sensör birimine tg2 global zamanı ile bir paket gönderilsin. Daha önceki yanlış senkronlanmadan kaynaklı olarak sensör birimi 2'nin zamanı tp kadar gecikmeliydi. Göndericinin zamanı tg2-tp ve alıcınınki tg2+tp olacaktır. Donanımdaki yayılım gecikmesi buradan;

$$tp = (Stamp2 - Stamp1) / 2 \quad (5.10)$$

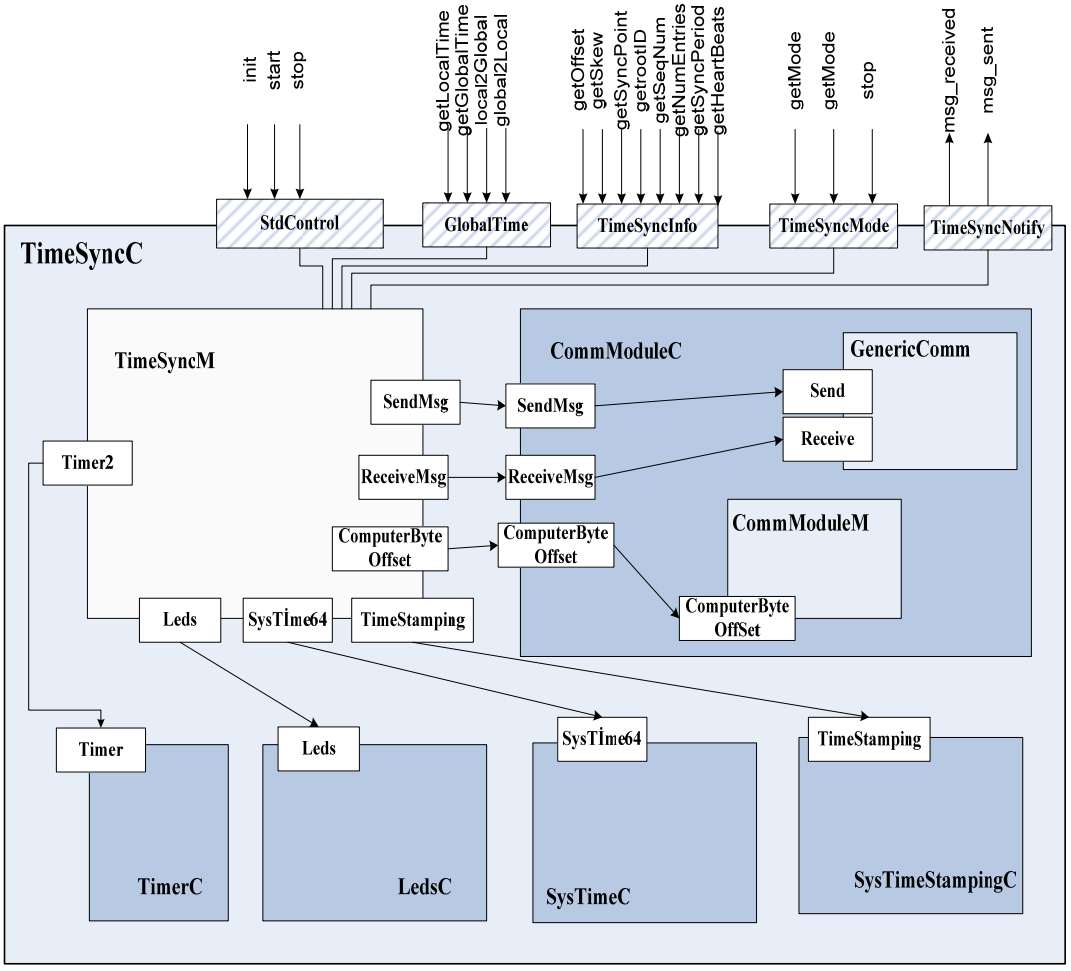
olarak hesaplanabilir. (5.10) numaralı denklem rekürsif hale getirilmek istendiğinde;

$$Tp(n) = (1 - alpha) \times Tp(n - 1) + alpha \times tp(n) \quad (5.11)$$

olarak yazılabilir. Burada alpha katsayısı 0 ile 1 arasında seçilmelidir.Tablo 5.1'de senkronizasyon süresine göre hatanın değişim miktarı gösterilmiştir.

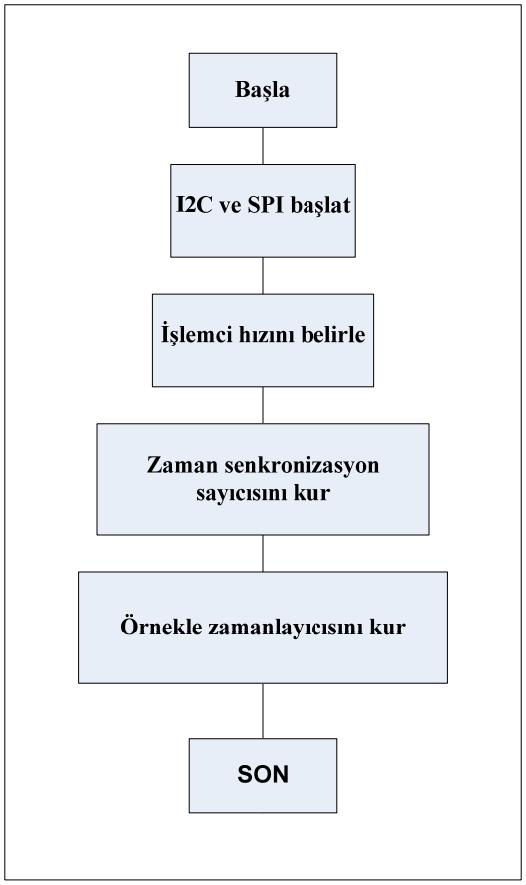
Tablo 5.1: Senkronizasyon hatası

Tekrar Senkronizasyon süresi (s)	Maksimum hata(us)	Ortalama hata (us)
2	1.86	.118
5	2.15	.150
30	4.92	.199
70	16.0	.558



Şekil 5.14: TimeSyncC bileşeni

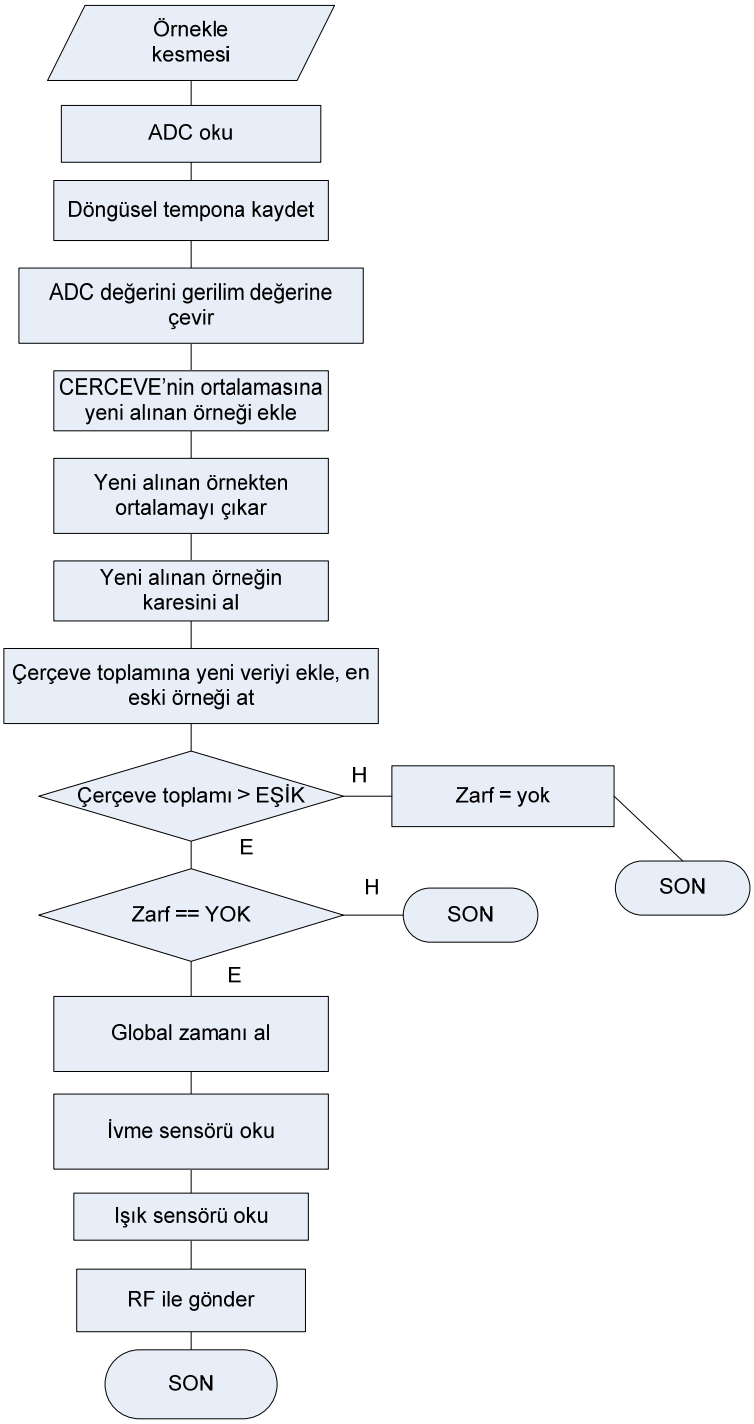
TimeSyncC bileşeni (Şekil 5.14), haberleşmek için arka planda GenericComm arayüzünü kullanan CommModuleC bileşenini kullanmaktadır. Tekrar senkronizasyon için TimerC bileşenini, yerel zamanı sistemden alabilmek için SysTimeC bileşenini kullanmaktadır.



Şekil 5.15: Program algoritması

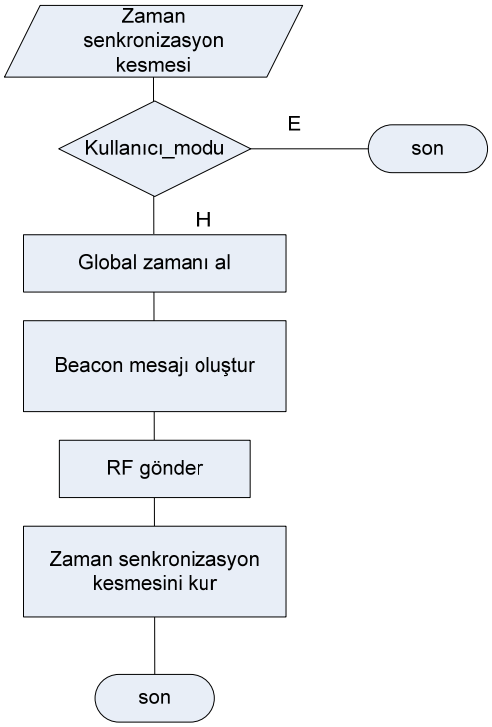
Şekil 5.15'de programın temel algoritması verilmiştir. Sistemin gerçek zamanlı işlem yapması beklendiğinden, her şey zamanlayıcı/sayıcılar ile belirli periyotlarla kendi kendini çağırabilen modüller olarak tasarlanmıştır. Örnekle zamanlayıcısı 1ms 'de bir örnek alacak şekilde kurulmuştur. Bu da sesin örnekleme frekansının 1Khz olmasını sağlar. Ses işaretini kullanarak ses tanıma veya geri oluşturma yapılmayacağı için, bir bant geçiren işaret olduğu göz önüne alınarak Shannon teoremine göre düşük örnekleme frekansı seçilmiştir. Örnekleme frekansının düşük seçilmesinin nedeni, I^2C veri yolu hızının 400Khz ile sınırlı olmasıdır.

Yukarıda kurulan örnekle kesmesinin algoritması Şekil 5.16'da gösterilmiştir.

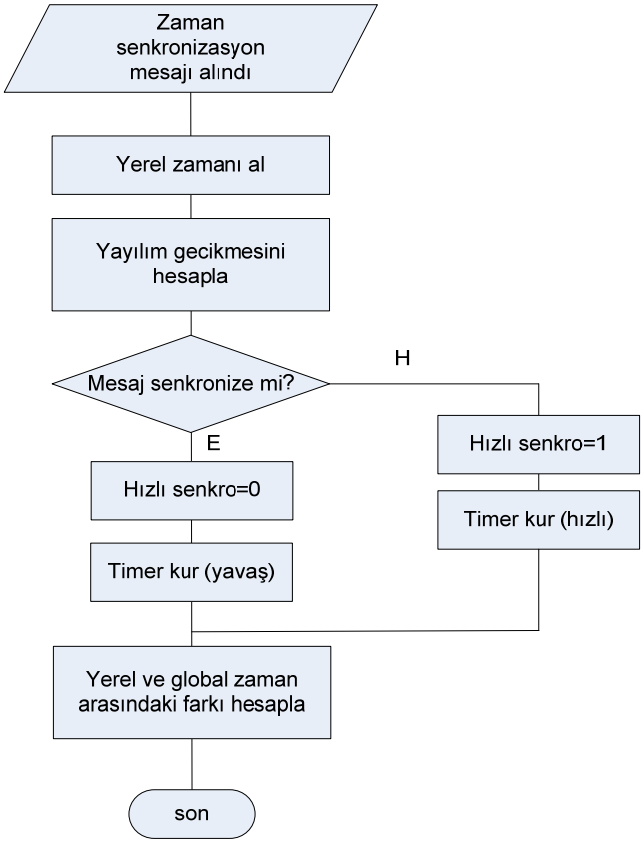


Şekil 5.16: Örnekle kesmesi algoritması

Şekil 5.15'de kurulan zaman senkronizasyonu kesmesinin algoritması (Şekil 5.17)'de verilmiştir. Burada kullanıcı modu, senkronizasyonun gerekirse kullanıcı tarafından kontrollü olarak yapılabilmesini seçmek için eklenmiştir.



Şekil 5.17: Zaman senkronizasyonu kesmesi



Şekil 5.18: Zaman senkronizasyon mesajı alındı kesmesi

Şekil 5.18'de Zaman senkronizasyon mesajı alındıktan sonra uygulanan algoritma gösterilmiştir.

Algoritma ve program kodları hazırlandıktan sonra doğruluklarının Matlab simülasyonu ile karşılaştırılması için, test amacı ile bilgisayara aktarıldı. (Tablo 5.2)

Tablo 5.2: Test amaçlı Matlab kodu

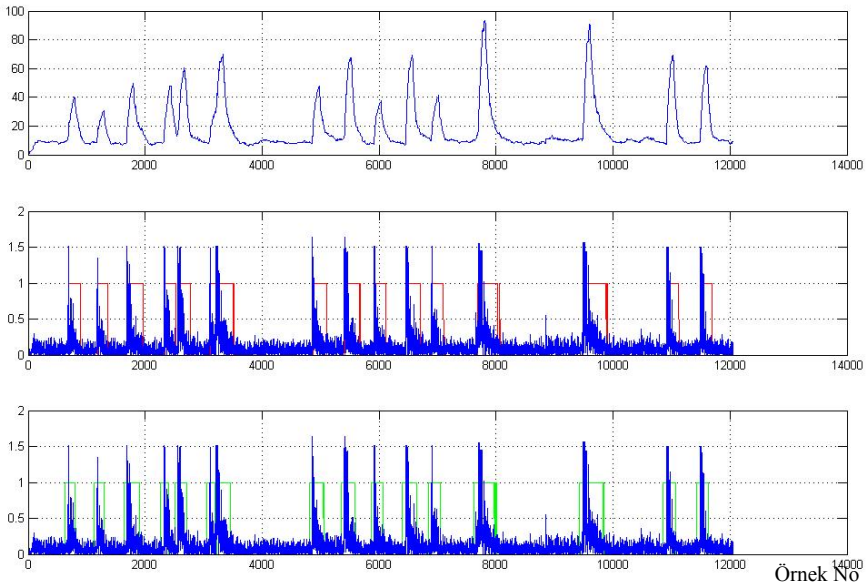
```
veri = load('c:\\veri.txt','ascii');          %Verilerin dosyadan yüklenmesi
zarf = load('c:\\veri2.txt','ascii');
ornek_sayisi = size(veri,1);
veri = veri .* 3 ./ 4096;                     %ADC verisinin gerilim değerine
dönüştürülmesi
sesKare = abs(veri);                          %sesin mutlak değeri alınarak negatif
bileşenlerinin
%sesKare = veri .^ 2;                         %yok edilmesi.

zarf1 = conv(veri, ones(cerceve, 1));         %sesin zarfının bulunması
zarf1 = zarf1(1:ornek_sayisi-1);
zarf2 = zeros(size(zarf1,1),1);
for i = 1: size(zarf1,1)                      %zarfın ESIK sabiti ile karşılaştırılması
  if zarf1(i) > esik
     zarf2(i) = 1;
  end
end

figure
subplot(311)
plot(zarf1)
grid
subplot(312)
plot(zarf2,'r')
hold on
plot(veri)
hold off
grid
subplot(313)
plot(zarf,'g')
hold on
plot(veri)
hold off
grid
```

Şekil 5.19'da ilk grafik gelen verilerin Matlab ortamında konvülasyona sokularak zarfın bulunması sonucunda elde edilmiştir. İkinci grafik, ses verileri ile Matlab'da

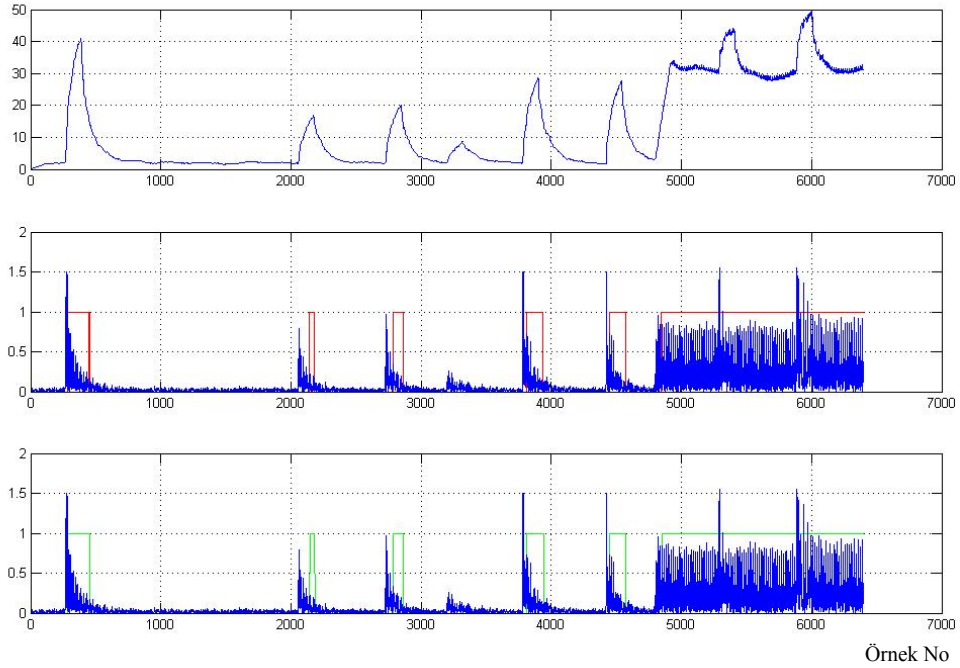
hesaplanan zarf verilerinin üst üste bindirilmesi sonucunda elde edilmiştir. Sesin belli bir eşik değerin üzerine çıkması, zarfı da yükseltmekte ve ESIK sabiti ile karşılaştırılması sonucunda sesin olup olmadığına karar verilmektedir. Ses kırmızı çizgilerin 1 olduğu durumlarda tespit edilmiştir.



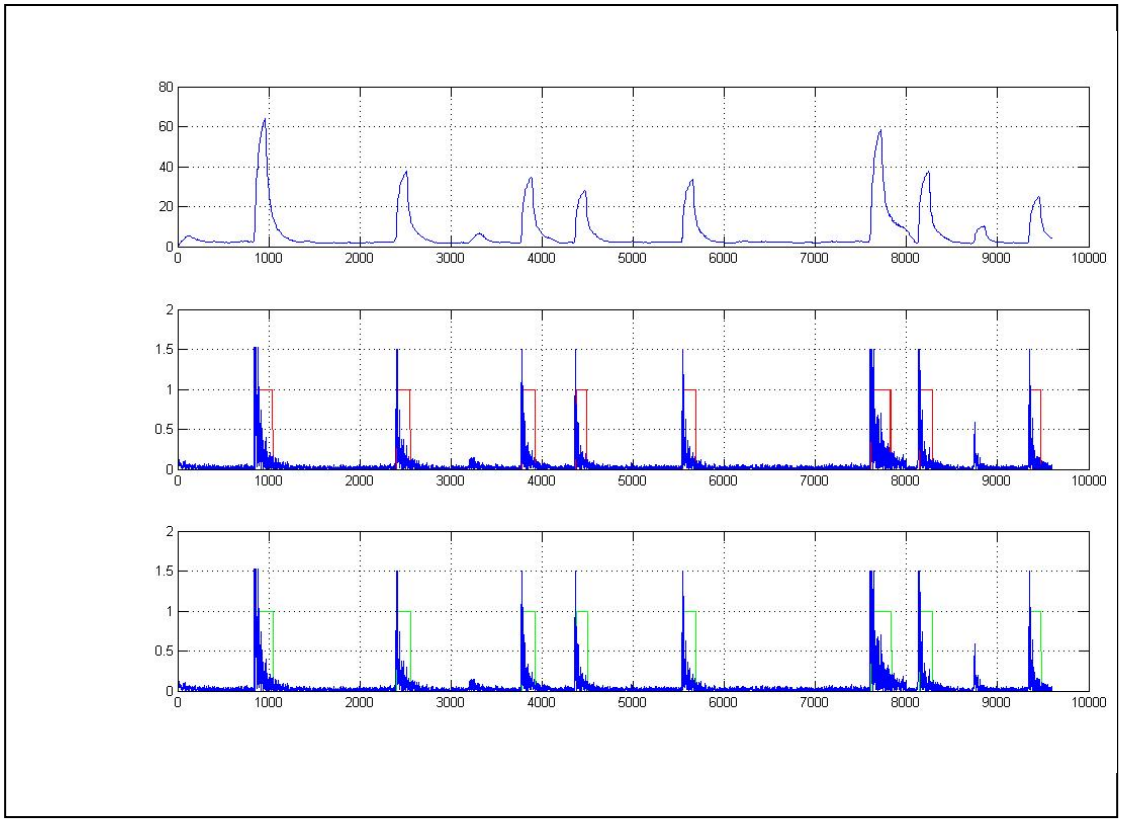
Şekil 5.19: Zarf grafiği -1

Üçüncü grafik ise, ses işaretinin üzerine, imote2 platformunda hesaplanarak bulunan ses var işaretinin bindirilmesi sonucunda çizdirilmiştir. Kırmızı ile yeşil grafiklerin birbiri ile paralellik göstermesi, Matlab'da hesaplanan zarf işareti ile imote2'de hesaplanan zarf işaretinin benzer olduğunun göstergesi olarak kabul edilebilir.

Şekil 5.20'de verilen grafikte ses sinyalinin yanında 4800'üncü örnekten sonra ortama gürültü de eklenmiştir. Birinci grafik incelendiğinde, zarfın ortam gürültüsüne çok bağlı olduğu fark edilmiştir. Bunu azaltmak için, işlemciye ek yük getirmek pahasına, gelen sinyalin mutlak değeri değil, karesi alınmasına karar verilmiştir. Böylece 1'den küçük değerlerin, sıfıra yaklaştırılması ve sinyal ile gürültü arasındaki farkın açılması hedeflenmiştir.



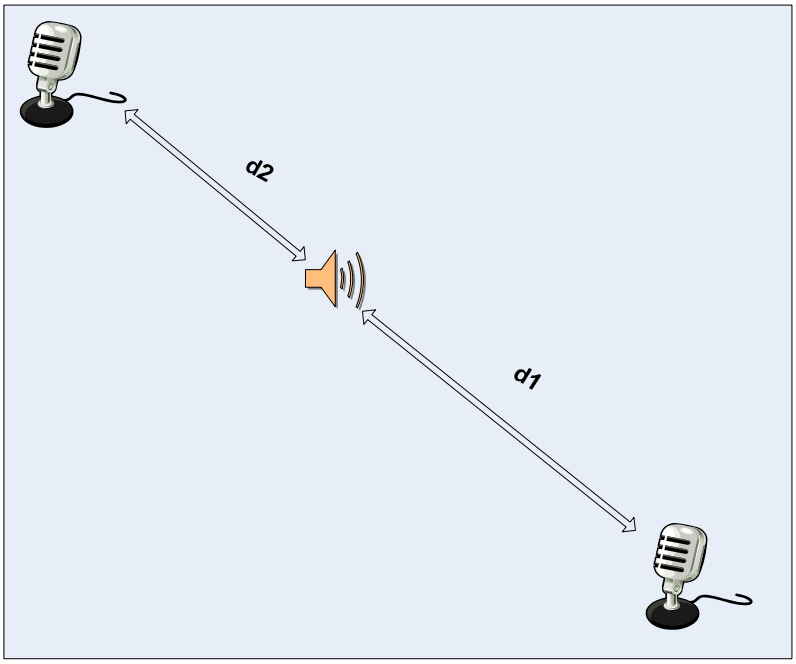
Şekil 5.20: Zarf grafiği -2



Şekil 5.21: Zarf grafiği -3

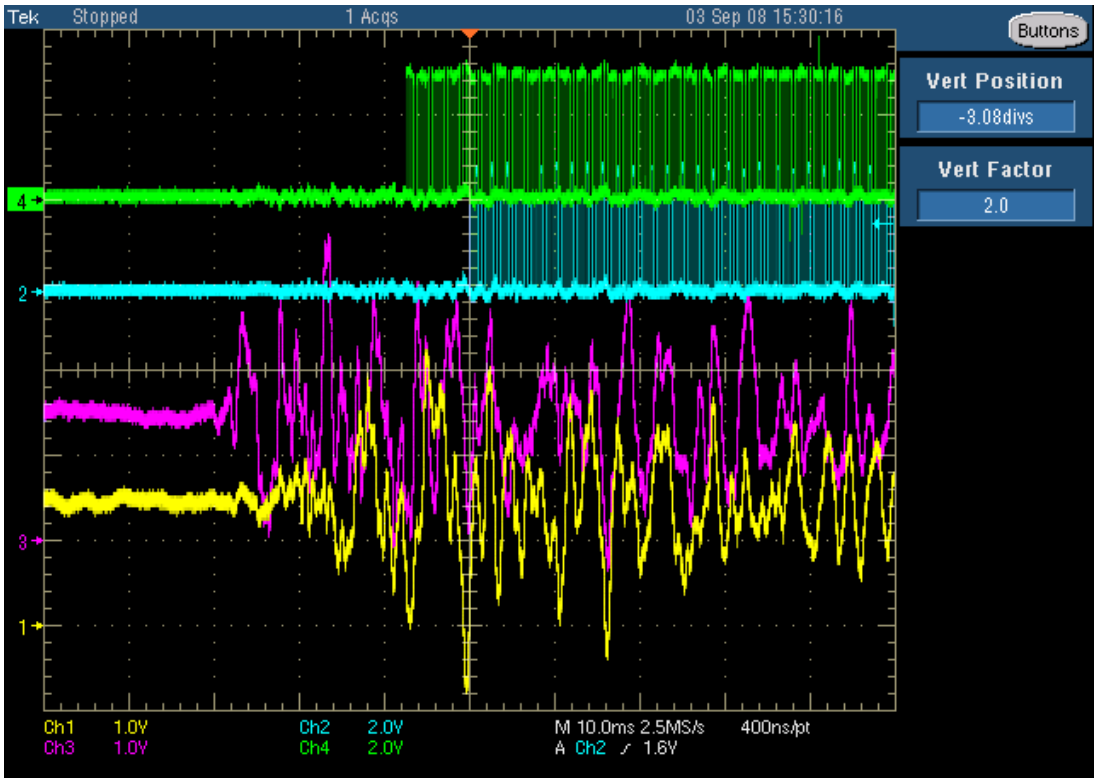
Karesi alınarak, Matlab'da ve imote2 platformunda hesaplanan zarf değerleri yukarıdaki grafikte gösterilmiştir. (Şekil 5.21) Grafiklerin birbirlerini tuttuğu göz önüne alınarak, sistem gerçek zamanlı haline geri dönüştürülmüş ve çeşitli senaryolar için testler yapılmıştır.

İki Imote2 kullanılarak alınan ölçümlerde, ses kaynağı iki sensör birimi arasında farklı uzaklıklara konularak, sistemin zarf çıktıları osiloskop kullanılarak incelenmiştir. Şekil 5.22'de deney gösterilmiştir. Mikrofonlar, sensör birimlerini temsil etmektedirler.



Şekil 5.22: İki Imote2 ile alınan ölçümler

Bu deney sonucunda bulunan değerler Tablo 5.3'de listelenmiştir. Deney No.1 için örnek osiloskop görüntüsü de Şekil 5.23'de verilmiştir. Şekil 5.23'de mor ve sarı renkteki sinyaller, sensör birimlerinin ADC'sine giren sinyali, yeşil ve mavi renkteki sinyaller ise, sensör birimlerinin zarfın eşik değerini geçtiği zaman genel amaçlı çıkış pinlerinde oluşturdukları sinyalleri göstermektedir. Mor ve yeşil renkteki sinyaller bir sensör birimine, sarı ve mavi renktekiler ise diğer sensör birimine bağlıdırlar. Bu bilgilere dayanarak, mor renkli sesin yakındaki sensöre geldiği ve sesin hesaplanan zarfının eşik değerini geçtiği ve sinyalinin daha uzaktaki sensör birimine bağlı olan mavi renkli sinyalden daha önce 1'e çıktığı söylenebilir. Yeşil ile mavinin aralarındaki fark, sesin sensörlere ne kadar gecikmeli geldiğini göstermektedir.



Şekil 5.23: Deney No.1 için osiloskop görüntüsü

$$c_{hava} = (331.5 + (0.6 \cdot \vartheta))ms^{-1} \qquad (5.12)$$

20ºC hava sıcaklığında sesin yayılım hızı, (5.12) numaralı formülden 343.5m/s olarak bulunur.

Tablo 5.3: İki Imote2 için bulunan deney sonuçları

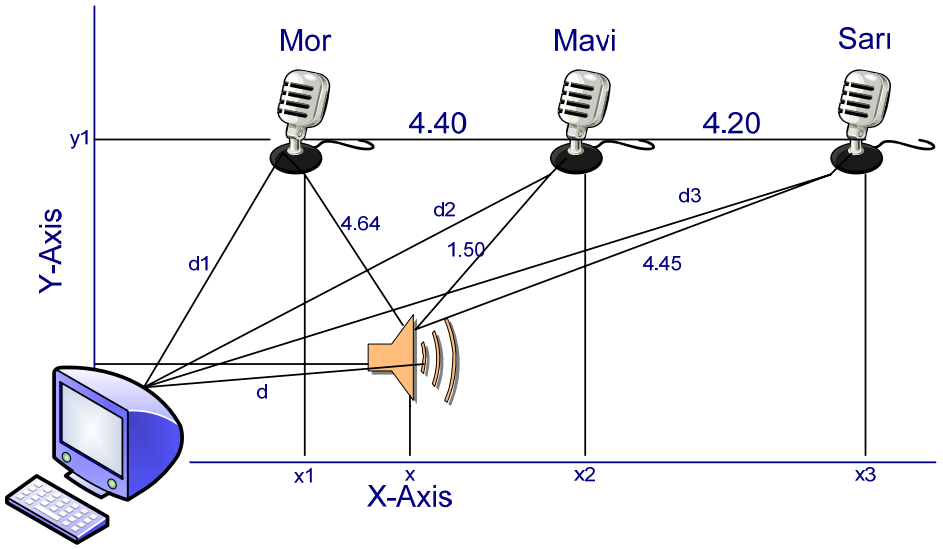
Deney No	Gerçek d1 uzaklığı (m)	Gerçek d2 uzaklığı (m)	d1-d2 (m)	Zarflar arasındaki zaman farkı (s)	Hata (m)
1	4.7	1.9	2.80	7.2ms -> 2.473m	0.327
2	3	3.6	0.6	0.8ms -> 0.274m	0.326
3	3	3.6	0.6	2ms -> 0.687m	0.087
4	3	3.6	0.6	3.2ms -> 1.099m	0.409
5	0.5	6.1	5.6	26ms -> 8.84m	3.224

Tablodaki zarflar arasındaki zaman farkı ile ses hızı çarpılarak hesaplanan uzaklık değeri, sesin sensör birimlerine olan uzaklığının farkına (d1-d2 değerine) eşit olmalıdır. Deney No.1 için yer 0,3 metre hatalı olarak tespit edilmiştir. Örnekleme frekansının 1ms olduğu göz önüne alınırsa çözünürlük;

$$1e\text{-}3\ (s) \times 340\ (m/s) = 0.34\ (m) \qquad (5.13)$$

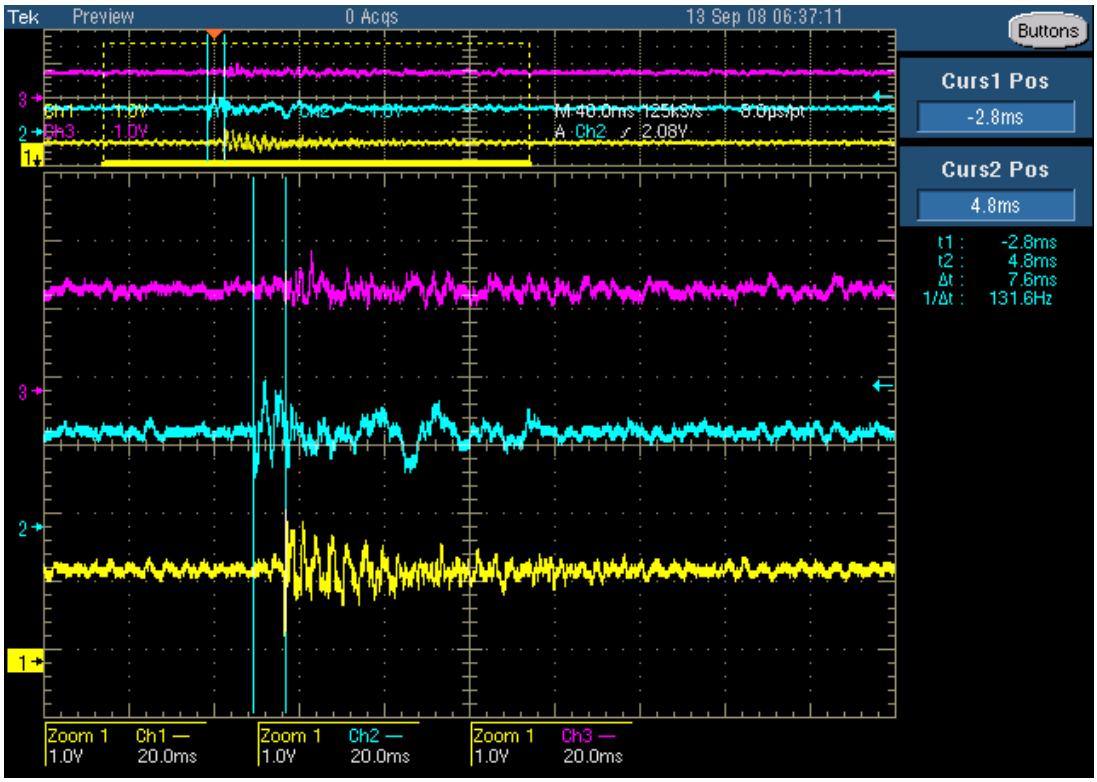
olarak bulunur. Bu koşullarda, yer tespitinin ancak ±0.15m ile bulunması beklenmelidir. Bu bilgi ışığında, 1, 2, 3 ve 4 numaralı deneylerin tutarlı sonuç verdiği söylenebilir. 5 numaralı deneyde ise, hata 3 metre olarak bulunmuştur. Burada dikkati çeken bir nokta, 5 numaralı deneyde, ses sinyalinin bir sensör birimine çok yakın, diğerine ise diğer deneylere göre çok uzakta olmasıdır.

Bu çalışmanın amacı olan 3 veya daha fazla sensör birimi ile iki eksenli uzayda ses kaynağının koordinatlarının belirlenmesi için oluşturulan deney düzeneği aşağıdadır. (Şekil 5.24)

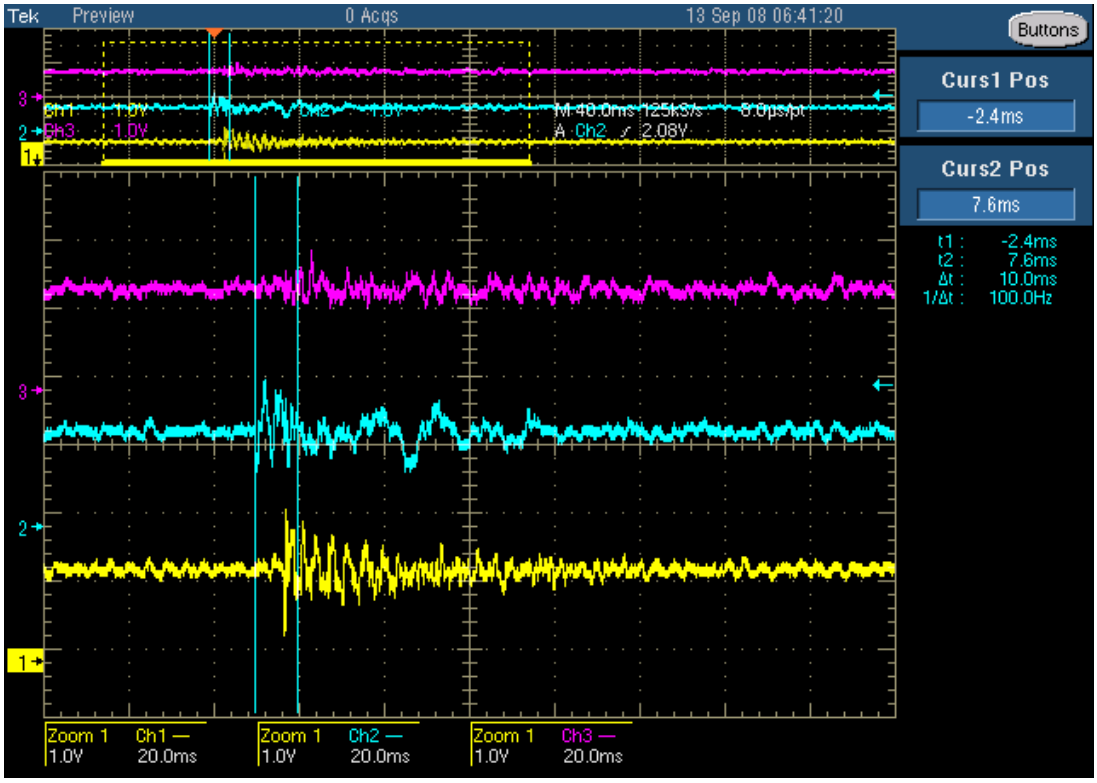


Şekil 5.24: Deney No.6 üç imote2 ile alınan ölçüm

Sensör birimlerinin üzerindeki renk isimleri, Şekil 5.25 ve Şekil 5.26'da sinyal renklerini temsil etmektedir. Bu deneyde, ses sinyalinin her bir sensör biriminin ADC girişine farklı zamanlarda girdiğinin tespit edilmesi amaçlanmıştır.



Şekil 5.25: Deney No.6 için osiloskop görüntüsü

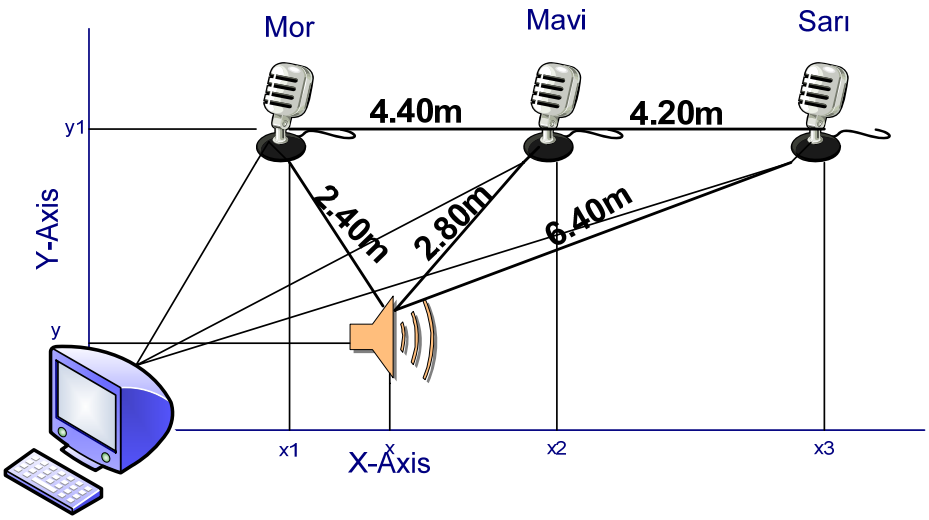


Şekil 5.26: Deney No.6 için osiloskop görüntüsü

Tablo 5.4: Deney No.6 için veriler tablosu

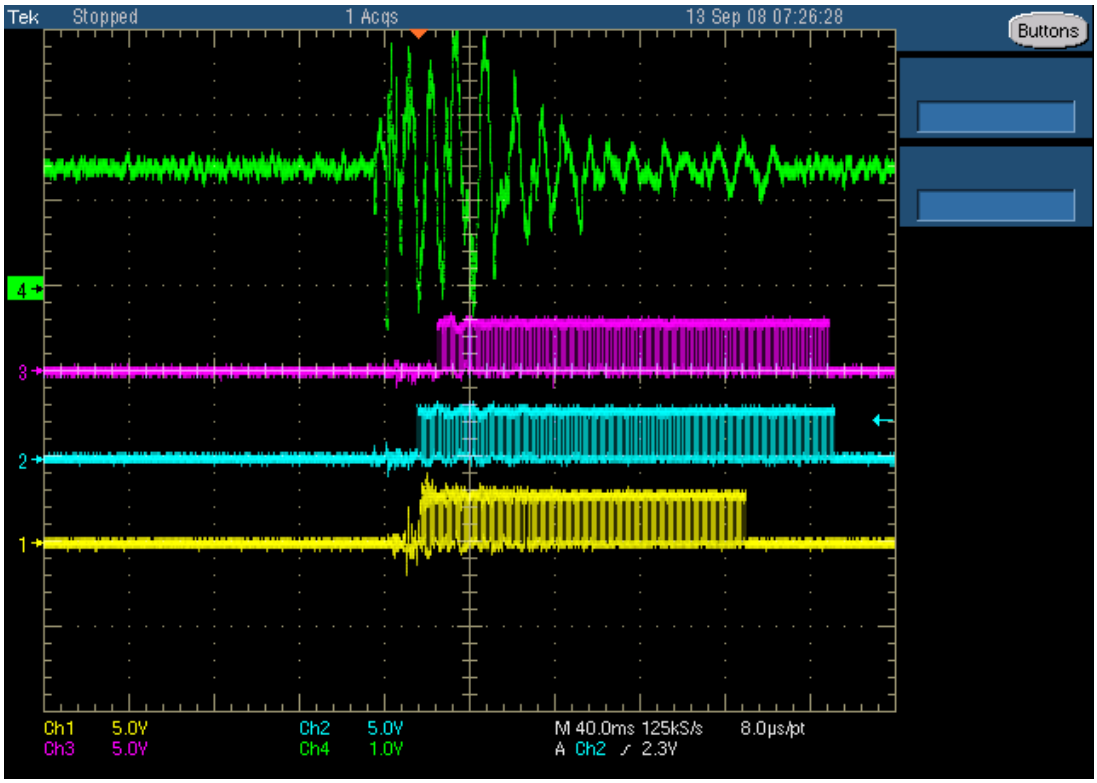
Sensör Birimi	Süre(ms)	Mesafe(m)	Gerçek Mesafe (m)	Hata(m)
Mavi				
Sarı	7.6	2.58	3.95	0.37
Mor	10	3.4	3.14	0.26

Tablo 5.4'de, Şekil 5.26 ve Şekil 5.26 osiloskop görüntüleri kullanılarak bulunan farklar ile gerçek mesafeler karşılaştırılmış ve en yakın sensöre göre ne kadar gecikmeli geldiği belirtilmiştir.

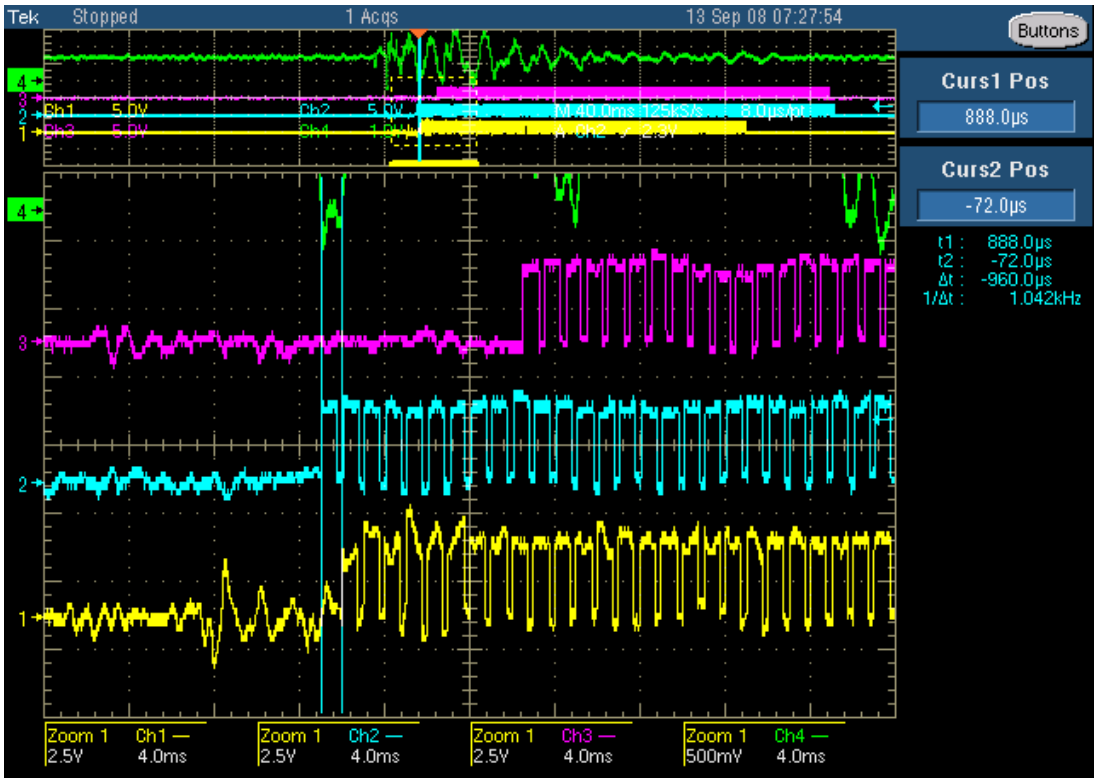


Şekil 5.27: Deney No.7 üç imote2 ile alınan ölçüm

7 numaralı deneyde, sensör birimlerinin ürettikleri zarfların aralarındaki zaman farkları karşılaştırılmıştır.(Şekil 5.27) Böylece, en yakın sensöre göre ne kadar geç geldiği bulunmuştur. Şekil 5.28 ve Şekil 5.29'da ilgili osiloskop görüntüleri gösterilmiştir.



Şekil 5.28: Deney No.7 için osiloskop Görüntüsü

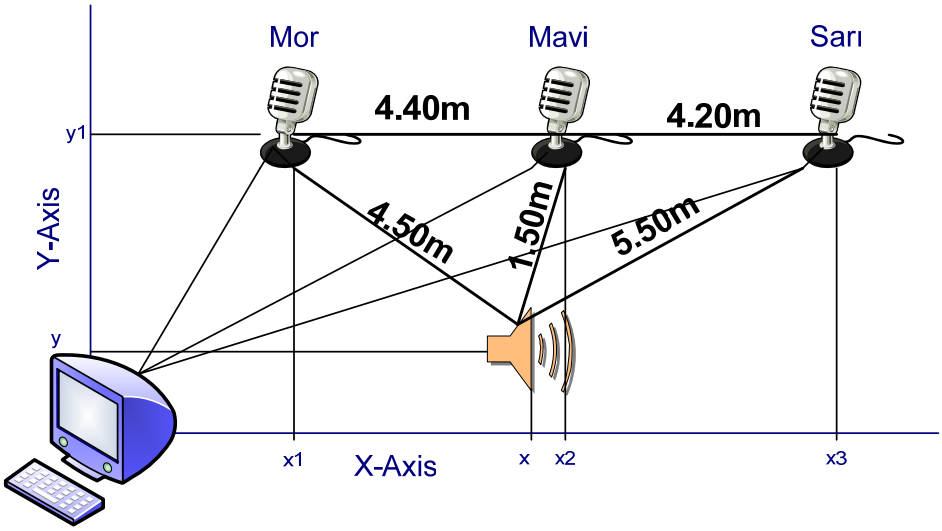


Şekil 5.29: Deney No.7 için osiloskop görüntüsü

Tablo 5.5: Deney No.7 için veriler tablosu

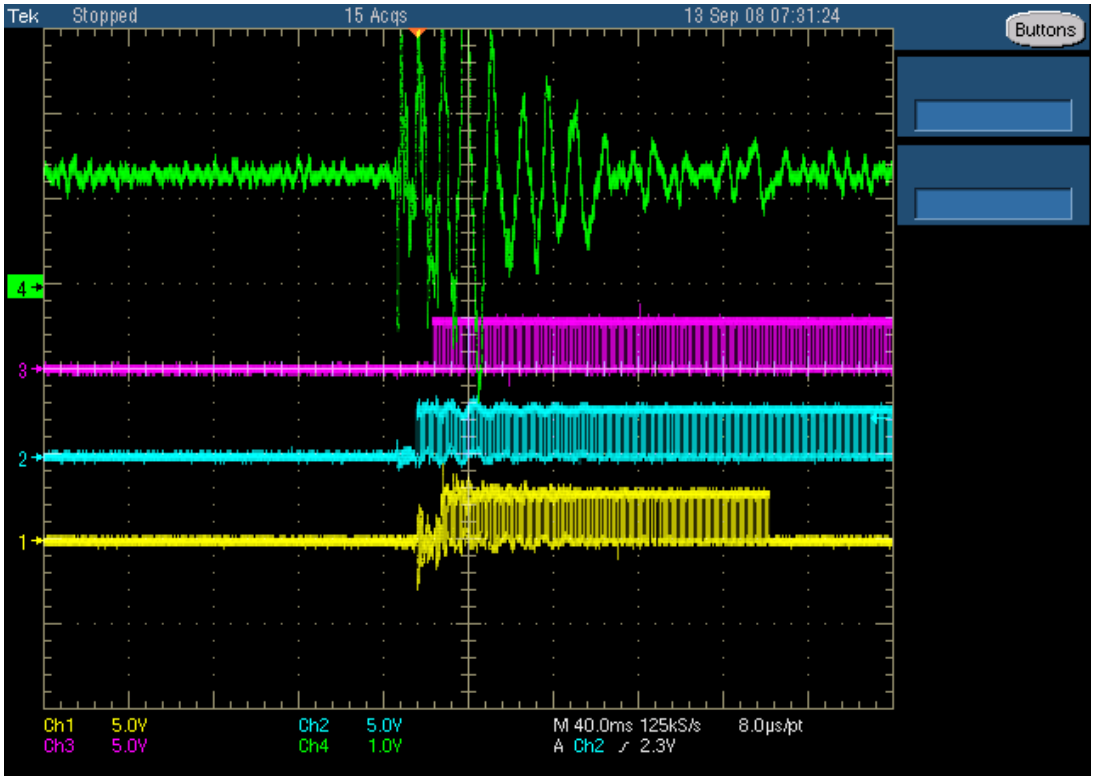
Sensör Birimi	Süre(ms)	Mesafe(m)	Gerçek Mesafe (m)	Hata(m)
Mavi				
Sarı	0.96	0.326	3.6	3.274
Mor	9.44	3.247	0.40	2.847

Tablo 5.5'de bulunan değerler listelenmiştir. Mor sinyal ile gösterilen sensör biriminin daha yakın olmasına rağmen sinyali daha geç aldığı görülmüştür. Aynı şekilde Sarı ile gösterilen sensör birimi en uzakta olmasına rağmen Mor'dan önce sinyali algılamıştır.

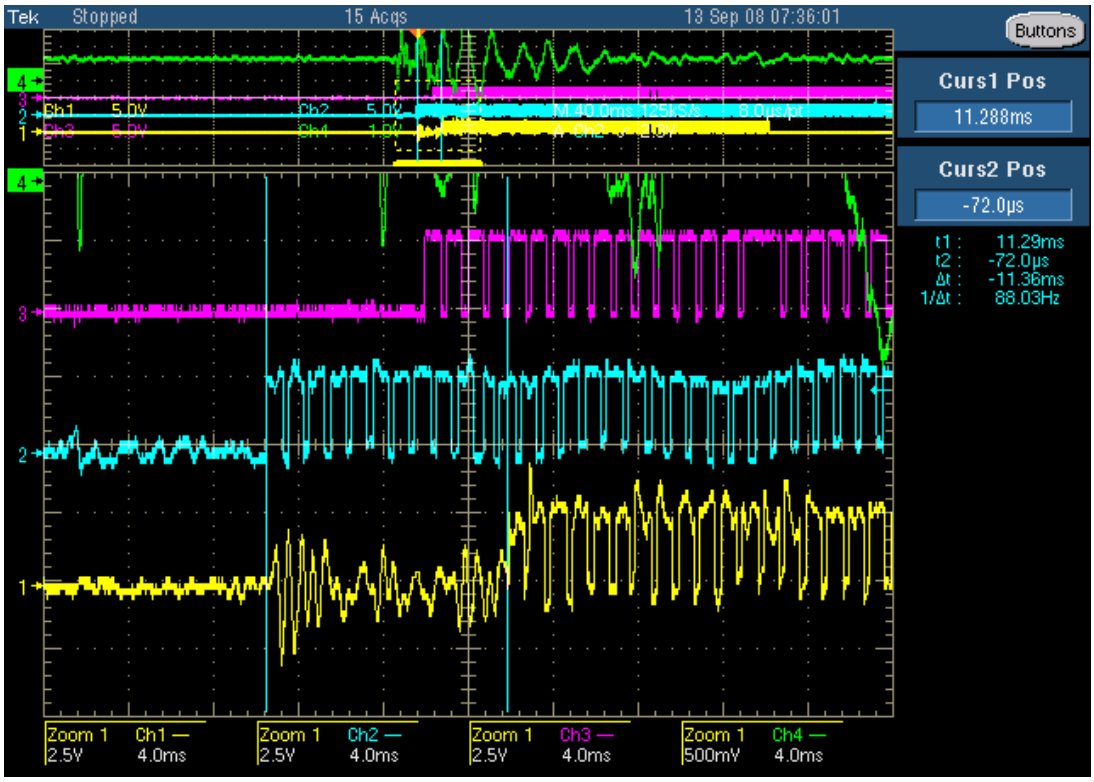


Şekil 5.30: Deney No.8 üç Imote2 ile alınan ölçüm

Şekil 5.30'da gösterilen deney düzeneği ile .Şekil 5.31 ve Şekil 5.32 osiloskop görüntüleri elde edilmiştir.



Şekil 5.31: Deney No.8 için osiloskop görüntüsü



Şekil 5.32: Deney No.8 için osiloskop görüntüsü

Tablo 5.6: Deney No.8 için veriler tablosu

Sensör Birimi	Süre(ms)	Mesafe(m)	Gerçek Mesafe (m)	Hata(m)
Mavi				
Sarı	11.36	3.86	4	0.14
Mor	7.44	2.52	3	0.48

Tablo 5.6'de listelenen verilerde, ses sinyali ilk olarak mavi renkli sensör birimine varmış, ardından mor ve sonra da sarı renktekine varmıştır. Aralardaki gecikmeler hesaplandığında, hesaplanan mesafelerin gerçekte var olanlarla tutarlılık gösterdiği görülmektedir.

6. SONUÇLAR

Bu çalışmada kablosuz duyarga ağlarından toplanan çeşitli algılayıcı verilerinin birleştirilerek, belirli bir alana giren ışık, ses ve titreşim kaynağı olan bir nesnenin algılanması amaçlanmıştır.

Gerekli algılayıcılar belirlendikten sonra, ilgili algılayıcılardan veri toplayabilmek için bu algılayıcıların sürücülerinin TinyOS işletim sisteminde olup olmadığı araştırılmış, sürücüsü olmayanlara ise baştan sürücü yazılmıştır. Bu noktada, Imote2 platformunun geliştirilmesinin ve temel sürücülerinin bazılarının hazır olarak TinyOS işletim sisteminde sunulmamasından dolayı, beta veya alpha sürümü durumundaki sürücüler alınarak geliştirilmek zorunda kalınmıştır. Sürücüler içindeki hatalar bulunmuş ve bu hatalar düzeltilerek bu çalışmada belirlenen amacı gerçekleyecek olan programın yazımı tamamlanarak, algılayıcılardan veri toplanabilmiştir.

Kablosuz duyarga ağının bir avantajı olarak, toplanan veriler kablosuz ortamdan merkez bir bilgisayara iletilmiştir. Gerekli protokoller ve bu protokolleri destekleyen arayüzler TinyOS işletim sisteminden alınmıştır.

Deney sonuçlarında, ses sinyali için Nyquist ölçütüne göre beklenen örnekleme frekansının altında örnekleme yapıldığı için, sistemin yer belirlemesinde hatalar meydana gelmiştir. Bir sonraki çalışmada örnekleme frekansını arttırmak için, ya daha hızlı I^2C veri yolu olan bir işlemci veya işlemci değiştirmeden SPI üzerinden veya paralel haberleşebilen bir ADC kullanılmalıdır. Bunlara ilaveten, TinyOS işletim sisteminin imote2 platformu için olan SPI sürücüleri baştan yazılarak optimizasyona gidilebilir.

Sistemde kullanılan ses yükseltecinin kazancının sabit olmaması ve her bir yükseltecinin özelliklerinin de birbirinden farklı olması, deney düzeneğinin kurulumu sırasında her seferinde bunların tekrar baştan elle kalibre edilmesine neden olmuştur. Elle yapılan ayarlamalar da hatalara neden olmuştur. Bu tarzda bir yükselteç yerine

otomatik kazançlı bir kuvvetlendirici kullanılmalıdır. Günümüz teknolojisinde IP telefonlarda, PDA'lerde bu tarzda ses kuvvetlendiricileri sıkça kullanılmaktadır.

Bu yukarıda sayılan geliştirmelerin yapılması ve modüleritenin bozulmaması için, imote2 platformunun soket yapısına uygun bir kart tasarlanabilir.

KAYNAKLAR

[1] **Texas Instruments**, 2008, ZigBee Solutions, Datasheet: <http://focus.ti.com/lit/ds/symlink/cc2420.pdf>, (2008, Mayıs 4)

[2] **Texas Instruments**, 2008, MSP430F2274 Low Power Solutions, Datasheet: <http://focus.ti.com/lit/ds/symlink/msp430f2274.pdf>, (2008, Mayıs 4)

[3] **IEEE** Reliability Society, 2008: <http://www.ieee.org/portal/site/relsoc/menuitem.112d36a56667b078fb2275875bac26c8/index.jsp?&pName=relsoc_home>, (2008, Mayıs 4)

[4] **C. Shen, C. Srisathapornphat, and C. Jaikaeo**, 2001, *Sensor Information Networking Architecture and Applications,* IEEE Personal Communications, 52-59.

[5] **D. Nadig, and S. S. Iyengar,** 1993, A New Architecture for Distributed Sensor Integration, *Proceedings of IEEE Southeastcon '93,* Charlotte NC, April 1993.

[6] **G. Hoblos, M. Staroswiecki, and A. Aitouche**, 2000, Optimal Design of Fault Tolerant Sensor Networks, *IEEE International Conference on Control Applications,* Anchorage AK, September 2000, 467–472.

[7] **G. Hoblos, M. Staroswiecki, and A. Aitouche,** 2002, Optimal Design of Fault Tolerant Sensor Networks, *IEEE International Conference on Control Applications*, Anchorage, AK, September 2002, 366-370.

[8] **Perrig, A.**, 2002, SPINS: Security protocols for sensor networks. *Proceedings of MOBICOM*, 2001.

[9] **Rivest, R. L**. (1994), *Proceedings of the Second International Workshop on Fast Software Encryption (FSE)* 1994e: 86–96.

[10] **Doherty, L**., 2000, Algorithms for Position and Data Recovery in Wireless Sensor Networks UC Berkeley EECS Masters Report, Berkeley.

[11] **C. Intanagonwiwat, R. Govindan, and D. Estrin,** 2000, Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks, *ACM MobiCom'00*, Boston, MA, 56–67.

[12] **G.J. Pottie and W.J. Kaiser,** 2000, Wireless Integrated Network Sensors, Communications of the ACM, vol. 43, No. 5, pp. 551-8, May.

[13] **Cerpa, A.**, 2001, Habitat monitoring: Application driver for wireless communications technology, *ACM SIGCOMM Workshop on Data Communications,* Latin America and the Caribbean.

[14] **Mainwaring**, **A.**, 2002, Wireless Sensor Networks for Habitat Monitoring, *ACM International Workshop on Wireless Sensor Networks and Applications* (WSNA'02), April, 18.

[15] **RF Monolithics Inc**, 1999, TR1000 Data Sheet, <http://www.rfm.com/products/data/tr1000.pdf>, (2008, Mayıs 04)

[16] **Xu, Y., J. Heidemann, and D. Estrin,** 2001, Geography-informed energy conservation for Ad Hoc routing, *ACM Press: SIGMOBILE : ACM Special Interest Group on Mobility of Systems Users, Data and Computing,* p. 70 - 84.

[17] **Yarvis**, **M.D.**, *Real-World Experiences with an Interactive Ad Hoc Sensor Network.: International Conference on Parallel Processing Workshops*, USA, May 2002.

[18] **Windriver Systems**, 2008,VxWorks 5.4 Datasheet <http://www.windriver.com/products/html/vxwks55_ds.html>, (2008, Mayıs 04)

[19] **Microsoft Corp**., 2008, Microsoft Windows CE <http://www.microsoft.com/windowsce/embedded

[20] **PalmOS Software 3.5 Overview**,2008 <http://www.palm.com/devzone/docs/palmos35.html>, (2008, Mayıs 04)

[21] **Hildebrand**, **D.,** 2008 An architectural Overview of QNX, <http://www.qnx.com/literature/whitepapers/archoverview.html>, (2008, Mayıs 04)

[22] **Windriver Systems**, 2008, pSOSystem Datasheet <http://www.windriver.com/products/html/psosystem_ds.html>, (2008, Mayıs 04)

[23] **QNX Software Systems Ltd**., 2008, QNX Neutrino Realtime OS, <http://www.qnx.com/products/os/neutrino.html>, (2008, Mayıs 04)

[24] **Microware**, 2008, Microware OS-9, <http://www.microware.com/ProductsServices/Technologies/os-91.html>, (2008, Mayıs 04)

[25] **Kauler, B.,** 2008, CREEM: Concurrent Realtime Embedded Executive for Microcontrollers, <http://members.dodo.net.au/~void/old/creem.htm>, (2008, Mayıs 04)

[26] **pOSEK**: A super-small scalable real-time operating system for high-volume, deeply embedded applications: <http://www.isi.com/products/posek/index.htm>, (2008, Mayıs 04)

[27] **Microware**, 2008, Microware Ariel Technical Overview: <http://www.microware.com/ProductsServices/Technologies/ariel_technology_brief.html>, (2008, Mayıs 04)

[28] **TinyOS Alliance**, 2008, <http://www.tinyos.net/>, (2008, Mayıs 04)

[29] **Blumofe, R.**, 1994, Cilk: An Efficient Multithreaded Runtime System, *Proceedings of the 5th Symposium on Principles and Practice of Paralel Programming*, 1995.

[30] **Hu, J., Pyarali I., and Schmidt D.**, Measuring the Impact of Event Dispatching and Concurrency Models on Web Server Performance Over High-speed Networks, *Proceedings of the 2 nd Global Internet Conference*, IEEE, 1997.

[31] **Crossbow**, 2008,
<http://www.xbow.com/Products/productdetails.aspx?sid=253>, (2008, Mayıs 04)

[32] **Ertürk,** S., 2005. Sayısal Haberleşme, Birsen Yayınevi, İstanbul

[33] **Kuzu, A., Gökaşan, M., Bogosyan, S.,** Novel Approach for Detection and Tracking of Explosive Carrying Mobile Humans with Odor-Sensor Based Multisensor Networks, *WSEAS Transactions on Sytems and Control*, vol. 3, pp. 63–70, April 2008.

[34] **Kuzu, A., Bogosyan S., Gökaşan, M.,** Survey: Odor Source Localization, *WSEAS Int. Conf. On Applied Computer & Applied Computational Science*, pp. 25–30, April 2008.

[35] **Huang J.,** "Time Sync for Intel Motes", ver.0.01, Intel Corp., April 2008.

ÖZGEÇMİŞ

Serkan Erboral 1983 yılında İstanbul'da doğdu. 2001 yılında Eskişehir Fatih Fen Lisesi'nden mezun olduktan sonra, Kocaeli Üniversitesi Elektronik ve Haberleşme Mühendisliği'nde yüksek öğrenim hakkı kazandı. Bir yıl İngilizce hazırlık eğitimi aldı. 2006 yılında 3.26/4 not ortalaması ile bölümü en iyi ilk 10 not ortalaması içinde bitirdi. Aynı yıl, yüksek lisans öğrenimine Mekatronik Mühendisliği Anabilimdalı'nda başladı.