TOBB EKONOMİ VE TEKNOLOJİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

Çok Atlamalı Kablosuz Algılayıcı Ağlar için Dağıtık Bir Eşzamanlama Tekniğinin Tasarımı ve Deneysel İncelenmesi

YÜKSEK LİSANS TEZİ

Muhammed Fatih İNANÇ

Elektrik ve Elektronik Mühendisliği Ana Bilim Dalı

Tez Danışmanı: Prof. Dr. Bülent TAVLI

Nisan 2017

Fen Bilimleri Enstitüsü Onayı

.....
Prof. Dr. Osman EROĞUL
Müdür

Bu tezin Yüksek Lisans derecesinin tüm gereksinimlerini sağladığını onaylarım.

.....
Doç. Dr. Tolga GİRİCİ
Anabilimdalı Başkanı

TOBB ETÜ, Fen Bilimleri Enstitüsü'nün 131211023 numaralı Yüksek Lisans öğrencisi **Muhammed Fatih İNANÇ** 'ın ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı **"Çok Atlamalı Kablosuz Algılayıcı Ağlar için Dağıtık Bir Eşzamanlama Tekniğinin Tasarımı ve Deneysel İncelenmesi"** başlıklı tezi 07.04.2017 tarihinde aşağıda imzaları olan jüri tarafından kabul edilmiştir.

Tez Danışmanı:	**Prof. Dr. Bülent TAVLI** TOBB Ekonomi ve Teknoloji Üniversitesi	.....
Jüri Üyeleri:	**Yrd. Doç. Dr. Harun Taha HAYVACI** TOBB Ekonomi ve Teknoloji Üniversitesi	.....
	Prof. Dr. Ali KARA (Başkan) Atılım Üniversitesi	.....

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, alıntı yapılan kaynaklara eksiksiz atıf yapıldığını, referansların tam olarak belirtildiğini ve ayrıca bu tezin TOBB ETÜ Fen Bilimleri Enstitüsü tez yazım kurallarına uygun olarak hazırlandığını bildiririm.

Muhammed Fatih İNANÇ

ÖZET

Yüksek Lisans Tezi

Çok Atlamalı Kablosuz Algılayıcı Ağlar için Dağıtık Bir Eşzamanlama Tekniğinin Tasarımı ve Deneysel İncelenmesi

Muhammed Fatih İNANÇ

TOBB Ekonomi ve Teknoloji Üniversitesi
Fen Bilimleri Enstitüsü
Elektrik ve Elektronik Mühendisliği Ana Bilim Dalı

Tez Danışmanı: Prof. Dr. Bülent TAVLI

Tarih: Nisan 2017

Genelde tüm haberleşme ağları, özelde ise Kablosuz Algılayıcı Ağlar (KAA), sağladıkları hizmetlerin sıkı gereksinimlerini yerine getirebilmek için eşzamanlamaya ihtiyaç duyarlar. KAA'larda eşzamanlama birden çok amaç için gereklidir (örneğin, uyuma-uyanma çizelgelemesi, olay tespit etiketlemesi). Literatürde KAA'lar için çeşitli eşzamanlama protokolleri önerilmiş ve bunların bazıları da deneysel olarak test yataklarında sınanmıştır. Fakat, KAA'larda çok-atlamalı eşzamanlama deneysel olarak hiç incelenmemiştir. KAA literatüründeki bu açıklığı kapatmak amacıyla bu çalışmada özgün bir eşzamanlama tekniğinin tasarım ve gerçeklemesi sunulmuştur. Ayrıca, önerilen tekniğin başarımı doğrudan deneylerle irdelenmiştir. Sonuçlarımız önerilen tekniğin üstün başarısını ortaya koymaktadır.

Anahtar Kelimeler: Kablosuz algılayıcı ağlar, Gömülü sistemler, Eşzamanlama, Çok-atlamalı haberleşme, Saat kayması.

ABSTRACT

Master of Science

Design and Experimental Evaluation of a Distributed Time Synchronization Technique for Multi-Hop Wireless Sensor Networks

Muhammed Fatih İNANÇ

TOBB University of Economics and Technology
Institute of Natural and Applied Sciences
Department of Electrical and Electronics Engineering

Supervisor: Prof. Dr. Bülent TAVLI

Date: April 2017

All communications networks, in general, and Wireless Sensor Networks (WSNs), in particular, need time synchronization for fulfilling the stringent requirements of the services they are providing. Time synchronization in WSNs is needed for multiple purposes (*e.g.*, sleep-wakeup scheduling, event detection annotation). In literature, various time synchronization protocols for WSNs are proposed and some of these designs are evaluated in experimental testbeds. However, multi-hop time synchronization in WSNs have never been investigated experimentally. Therefore, to fill the gap in the WSN literature, in this study, we present the design and implementation of a novel time synchronization technique. Furthermore, we investigate the performance of the proposed technique through direct experimentation. Our results reveal the superior performance of our technique.

Keywords: Wireless sensor networks, Embedded systems, Time synchronization, Multi-hop ommunications, Clock drift.

TEŞEKKÜR

Çalışmalarım boyunca değerli yardım ve katkılarıyla beni yönlendiren ve yardıma ihtiyaç duyduğum her anda baba destek olan tez danışmanım ve kıymetli hocam Prof. Dr. Bülent TAVLI'ya, çok değerli tecrübelerinden faydalandığım Prof. Dr. Ali KARA'ya ve TOBB ETÜ Elektrik-Elektronik Mühendisliği Bölümü öğretim üyelerine sonsuz teşekkürlerimi sunarım.

Ayrıca manevi desteklerinden ötürü sevgili annem Safiye İNANÇ'a ve tez yazım sürecinde bana yardımcı olan Abdullah SOYLU ve Seyfullah KILIÇ'a da teşekkürü borç bilirim.

İÇİNDEKİLER

ŞEKİL LİSTESİ

ÇİZELGE LİSTESİ

KISALTMALAR

KAA : Kablosuz Algılayıcı Ağ
GPS : Global Positioning System
eTPU : Enhanced Time Processor Unit
PTP : Precision Time Protocol
NTP : Network Timing Protocol
MCU : Microcontroller Unit
ADC : Analog to Digital Converter
TCP/IP : Transmission Control Protocol/Internet Protocol
PPM : Parts Per Million
TCXO : Temperature Controlled Crystal Oscillator
OCXO : Oven Controlled Crystal Oscillator
CSMA/CA : Carrier Sense Multiple Access/Collision Avoidance
MAC : Medium Access Control
LBT : Listen Before Talk
RAM : Random Access Memory
ARM : Advanced Risc Machine
BPS : Bits Per Second
TI : Texas Instruments
RTOS : Real Time Operating System
CRC : Cyclic Redundancy Check
RF : Radio Frequency
GFSK : Gaussian Frequency Shift Keying
PHY : Phsical Layer
USB : Universal Serial Bus

SEMBOL LİSTESİ

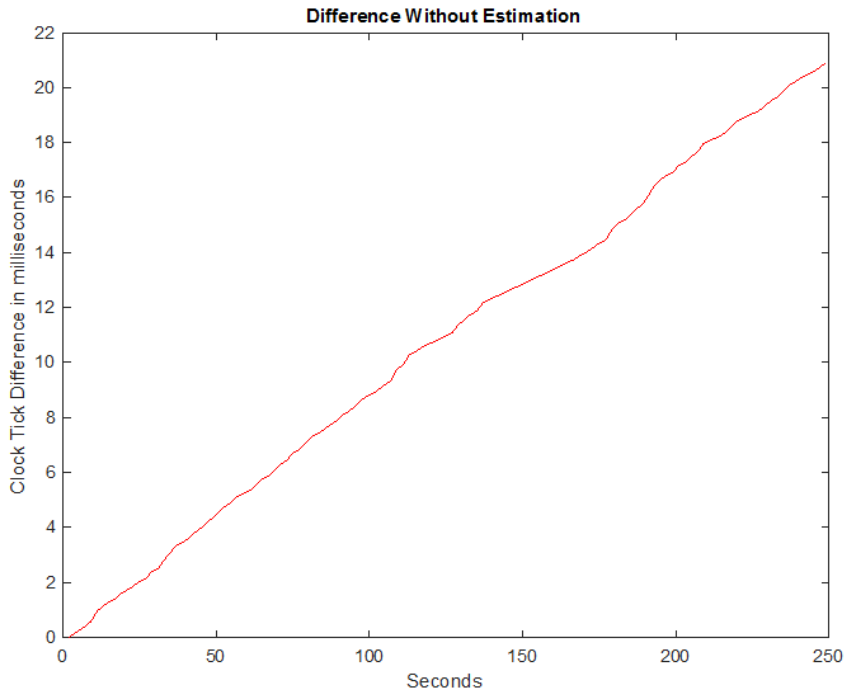
Bu çalışmada kullanılmış olan simgeler açıklamaları ile birlikte aşağıda sunulmuştur.

Simgeler	Açıklama
t_a	Algılayıcı-1'in eşzamanlama işlemini başlattığı zaman bilgisi
t_b	Algılayıcı-2'nin algılayıcı-1'in gönderdiği paketi aldığı zaman bilgisi
t_c	Algılayıcı-1'in algılayıcı-2'nin gönderdiği paketi aldığı zaman bilgisi
t_x	Yüksek öncelikli diğer işlemlerin süresi
t_1	Algılayıcı-1'in yerel zamanı
t_2	Algılayıcı-2'nin yerel zamanı
α_{12a}	Algılayıcı-1 ile Algılayıcı-2 arasındaki ofset değerinin alt sınırı
α_{12b}	Algılayıcı-1 ile Algılayıcı-2 arasındaki ofset değerinin üst sınırı
α_{12}	Algılayıcı-1 ile Algılayıcı-2 arasındaki ofset değerinin ortalaması
α_{12avg}	Nihai ofset değeri
β_{12a}	Algılayıcı-1 ile Algılayıcı-2 arasındaki kayma değerinin alt sınırı
β_{12b}	Algılayıcı-1 ile Algılayıcı-2 arasındaki kayma değerinin alt sınırı
β_{12}	Algılayıcı-1 ile Algılayıcı-2 arasındaki kayma değerinin ortalaması
β_{12avg}	Nihai kayma değeri
t_{b-est}	Algılayıcı-2'nin tahmini yerel zaman değeri
d_{TX}	Paket gönderme süresi
p	Gönderilen bit sayısı
s	Gönderim hızı(bps)
N	Örnek değeri

1. GİRİŞ

Kablosuz Algılayıcı Ağlar (KAA) henüz olgunlaşma safhasını bütünüyle tamamlamamıştır. Buna rağmen, tüm dünyada KAA kullanımı özellikle güvenlik amaçlı olarak oldukça yaygın bir durumdadır (örneğin savaş alanı izlemesi, kritik altyapı koruması, izleme). KAA'ların verimli ve yüksek başarımlı iş görebimeleri için eşzamanlama en önemli servislerden birisidir. KAA'ların küçük boyutlu ve sınırlı kapasiteli platformlardan oluşmasının yanısıra kullanılabilecek olan frekans bandı ve pil enerjisinin de sınırlı olması nedenleriyle KAA'lar için tasarlanacak olan eşzamanlama protokollerinin bu oldukça sıkı kısıtlar içerisinde verimli şekilde çalışmasını sağlayacak tarzda tasarlanması gerektirmektedir [1].

KAA'larda eşzamanlama 10 yılı aşkın süredir incelenmekte olan bir konudur [2, 3]. Bu konudaki çalışmaların böylesi uzun soluklu olmasının temel nedenleri hem konun önemi hem de çözülmeye çalışılan problemin zor olan doğasıdır [4–6]. Pek çok KAA uygulamasında, eşzamanlama, amaçlanan işlevselliği gerçekleştirebilmek adına yaşamsal öneme haizdir. Örneğin, nesne takibi veya olay tespiti gibi uygulamalarda çok miktarda algılayıcıdan alınan verinin birleştirilmesi önem arz etmektedir. Böylesi işlevleri yüksek başarımla yerine getirebilmek için eşzamanlama gereklidir. Yaşam süresini uzatmak amacıyla KAA algılayıcıları sıklıkla düşük enerji harcayacakları uyku durumuna geçerler. Fakat, süregelen temel algılama ve algılanan verinin baz istasyonuna (doğrudan veya çok atlamalı olarak) eriştirilmesi için algılayıcıların belirli bir çizelgeye göre uyanmaları birbirleriyle etkileşime geçmeleri ve iletişim gerçekleştirmeleri gerekmektedir. Böylesi bir çizelgeye uyabilmek için komşu algılayıcılar arası eşzamanlama gereklidir. Şimidiye kadar anlatılan kurgular KAA'larda eşzamanlamının önemini vurgulamak amacıyla sunulan az sayıda örnekten ibarettir fakat KAA'larda daha pek çok durumda eşzamanlama son derece önemli bir servistir [7–9].



Şekil 1.1: İki algılayıcı arasında oluşan zaman farkı.

Şekil 1.1'de görüldüğü üzere bir KAA'da iki algılayıcı arasında, 4 dakikada yaklaşık 20 milisaniye zaman farkı oluşmaktadır. Çoğu kablosuz ağ sistemi için 20 milisaniye fark, yüksek bir değer olarak kabul edilmektedir. Bu sebeple, bu tip bir ağda bulunan kablosuz algılayıcıların, yaklaşık her 4 dakika veya daha az bir sürede zaman senkronizasyonu yapması gerekmektedir. Bu zaman senkronizasyonu ihtiyacından dolayı kablosuz algılayıcılar, çok sık aralıklarla ağda aktif kalacağından dolayı, kablosuz ağ trafiğini arttırmakta ve ağ performansını azaltmaktadırlar.

Ayrıca, genellikle kablosuz algılayıcılar pil ile çalıştığı için kısa aralıklarla veri gönderimi ve alımı yaparak sürekli enerji tüketeceklerdir. Bu ise, algılayıcıların pil ömrünü büyük oranda azaltmaktadır.

Genel kabul saat zaman kaymasını değişmez bir sabit almak yönündedir. Aslında göreceli olarak kısa sürelerde bu varsayımın doğruluğu pek çok deneysel çalışmada teyid edilmiştir. Böylece, çözülmesi gereken problem, evrensel bir referansa göre kayma ve ofset değerlerinin doğru bir şekilde tahmin edilmesine indirgenebilmektedir.

Literatürde deneysel olarak doğrulanmış çok miktarda ofset ve kayma tahmini yapabilen çalışma sunulmuştur [10–17]. Ofset ve kayma parametrelerini tahmin için iki yollu bir tokalaşma mekanizması kullanılabilir. Ayrıca eşzamanlama için oluşan fazla trafiği düşürmek amacıyla mevcut paketler üzerine bindirme yöntemi de kullanılabilir çünkü standart bağ seviyesi veri ve teyid mesajları eşzamanlama amaçlı tokalaşmayı gerçekleştirebilmek için mükemmel bir altlık oluşturmaktadır.

Bu çalışmada, çok atlamalı KAA'larda dağıtık çalışabilen ve düşük karmaşıklıklı bir eşzamanlama yönteminin tasarım ve gerçeklenmesi sunulmaktadır. Tasarladığımız eşzamanlama yönteminin deneysel incelemesi doğrudan deney yöntemiyle gerçekleştirilmiş ve deney sonuçları sunulmuştur.

Bu tez çalışması, aşağıda anlatıldığı şekilde hazırlanmıştır.

2. Bölümde Kalbosuz Algılayıcı Ağlar(KAA) tanımlanmıştır. KAA'ların ve algılayıcı düğümlerin özelliklerinden, üzerlerinde bulunabilecek sensör cihazlarından ve yazılım katmanlarından bahsedilmiştir. Ayrıca algılayıcı düğümlerin kalbi olan kristal osilatörlerin algılayıcı düğümlerdeki önemli rolü anlatılmıştır.

3. Bölümde kullanılan eşzamanlama algoritması detaylarıyla birlikte anlatılmıştır. Doğrunun eğimi denkleminin algılayıcı düğümlerin zamanlarının tahmininde nasıl kullanılabileceği formüllerle ve örneklerle gösterilmiştir.

4. Bölümde sistem modelinden bahsedilmiştir. Bir KAA'da eşzamanlamanın hangi türlerde yapılabileceği anlatılmıştır. Tek atlamalı veya çok atlamalı yöntemlerin ikisinde de önemli rol oynayan ortam trafiğinin eşzamanlama algoritmasını ne şekillerde etkileyebileceğinden bahsedilmiştir.

5. Bölümde sistem tasarımı detaylı olarak açıklanmıştır. Bu bölümde, hangi mikrodenetleyicinin ve kablosuz haberleşme biriminin kullanıldığından, bu birimlerin teknik özelliklerinden ve bu birimlerdeki iç gecikmelerin tespitinin Bölüm 3'te anlatılan eşzamanlama algoritmasının doğru çalışabilmesine olan etkisinden bahsedilmiştir.

6. Bölümde gerçek testlerden elde edilen deneysel çıktılar incelenmiştir. Hesaplanan örnek sayısının eşzamanlama algoritmasına olan önemli etkisi bu bölümde verilen çıktılarla daha net görülmüştür.

7. Bölümde tez çalışmasının sonuçları değerlendirilmiştir.

1.1 Eşzamanlama Konusu ile İlgili Literatürdeki Çalışmalar

Literatürde KAA'lar için çeşitli eşzamanlama protokolleri önerilmiştir. Bunların bazıları yanlıca simülasyon yöntemi ile test edilmiş[2, 7] , bazıları da doğrudan donanım üzerinde olarak teste tabi tutulmuştur[10–17]. Fakat, KAA'larda MAC katmanı içerisinde çalışabilen çok-atlamalı eşzamanlama konusu, deneysel olarak hiç incelenmemiştir.

Sichitiu ve arkadaşları yapmış oldukları çalışmada[14] 802.11 ad-hoc ağları için tek ve çok atlamalı eşzamanlama konusunu incelemişlerdir. Deneylerinde klasik tokalaşma mekanizması gerçeklenmiş ve atlama sayısına bağlı olarak ofset değerinin de arttığı gözlemlenmiştir.

Dai ve arkadaşları yapmış oldukları çalışmada[17] TSync adında küçük ve iki yönlü bir eşzamanlama algoritması geliştirmişlerdir. Deneylerini hem tek atlamalı hem de çok atlamalı olarak gerçekleştirip en doğru sonuca ulaşabilmek için tüm algılayıcıların zaman bilgilerini GPS alıcısından aldıkları zaman bilgisi ile kıyaslamışlarıdır. Atlama sayısının artmasıyla ofset değerinin de arttığını gözlemlemişlerdir.

Mirabella ve arkadaşları yapmış oldukları çalışmada[8] 802.15.4 ZigBee ağları için eşzamanlama konusunu incelemişlerdir. Algılayıcıların ortam şartlarına bağlı olarak zamanının değişebildiğinden ve bunun giderilmesi için kullanılabilecek birkaç farklı teknikten bahsetmişlerdir. Fakat bu yaklaşımların güç tüketimi açısından uygun olmadığını gözlemlemişlerdir. Bu sebeple büyük KAA'lar için eşzamanlama yapılırken az enerji harcayan yeni bir metod geliştirmişlerdir.

Cho ve arkadaşları yapmış oldukları çalışmada[12] IEEE 1588 PTP(*ing*. Precision Time Protocol) standardını deneysel olarak incelemişlerdir. Deneylerinde zaman bilgilerinin hassas elde edilebilmesi için birkaç farklı ek çevrebirim kullanmışlarıdır(eTPU). Bu çalışmayı 802.15.4 ağları için deneysel olarak test edip ve oldukça düşük ofset değerleri gözlemlemişlerdir.

Maroti ve arkadaşları yapmış oldukları çalışmada[15] herhangi bir KAA'da kullanılabilecek bir eşzamanlama modeli tasarlamışlardır. Bu model, algılayıcılar arasında eşzamanlama mesajları gönderilirken oluşabilecek gecikmeleri de içermektedir. İki algılayıcı arasında yaklaşık 30 dakikalık bir örnekleme işleminden sonra ofset değerinin hızla arttığı gözlemlenmiştir.

Elson ve arkadaşları yamış oldukları çalışmada[1] NTP(*ing*. Network Timing Protocol) yöntemini incelemişlerdir. NTP üzerinden bir kez eşzamanlama işleminden sonra ofset değerinin ne şekilde arttığı gözlemlenmiş ve bundan dolayı gelecek çalışmalarında ofset değerinin minimize eden NTP tabanlı yeni bir yaklaşım geliştirmeyi hedeflemişlerdir.

Bu çalışmalar içerisinde çok-atlamalı olarak on mikrosaniyeler mertebesinde sonuç elde edilen hiçbir deneysel çalışmaya rastlanılamamıştır.

2. KABLOSUZ ALGILAYICI AĞLAR

Kablosuz Algılayıcı Ağ(KAA), üzerinde bir takım algılayıcıların bulunduğu ve kablosuz haberleşme kabiliyeti olan cihazlar gurubudur. Genellikle bu tür algılayıcılar sıcaklık, nem, ışık değeri, basınç, titreşim, akım ve gerilim gibi birçok farklı değeri ölçmek ve görüntülemek amacıyla kullanılırlar.

Bir KAA, birden fazla algılayıcı düğüm(ing. *sensor node*) ve genellikle bir adet baz istasyonu içerir. Temel amaç; düğümlerin topladıkları sensör verilerini baz istasyonuna göndermeleridir. KAA içerisindeki algılayıcı düğümler, diğer düğümlerden etkilenmeksizin kendi haberleşme rotasını oluşturabilirler. Yani birbirlerine en yakın düğüm üzerinden çok atlamalı olarak baz istasyonuna erişim sağlarlar. Herhangi bir/birkaç düğüm ulaşılmaz duruma geldiğinde diğer düğümler yeni bir haberleşme rotası oluştururlar(ing. *self healing*).

Algılayıcı düğümlerin en önemli özelliği küçük ve taşınabilir olmasıdır. Bu düğümler, tipik olarak üzerinde mikro-işlemci, sensör ekipmanları, kablosuz haberleşme birimi ve batarya barındırırlar. Güç kaynağı batarya olduğu için enerji tüketimi KAA'larda çok önemlidir. Genellikle bir algılayıcı düğümün batarya değişimine ihtiyaç kalmadan uygulama sahasında en az 5 yıl çalışması beklenir.

KAA'lar, teknolojinin gelişmesiyle birlikte günümüzde birçok alanda yaygın olarak kullanılmaktadır. Ve her geçen gün yeni ve farklı uygulama alanları keşfedilmektedir. Aşağıda bu uygulama alanlarından bazıları görülebilir[18].

2.1 Uygulama Alanları

* Askeri Uygulamalar
  Keskin nişancı tespit sistemi
  İnsansız hava araçları
  Nükleer saldırı tespiti

* Sağlık Uygulamaları
  Yapay retina
  Hasta takip sisemleri
  Acil çağrı sistemleri

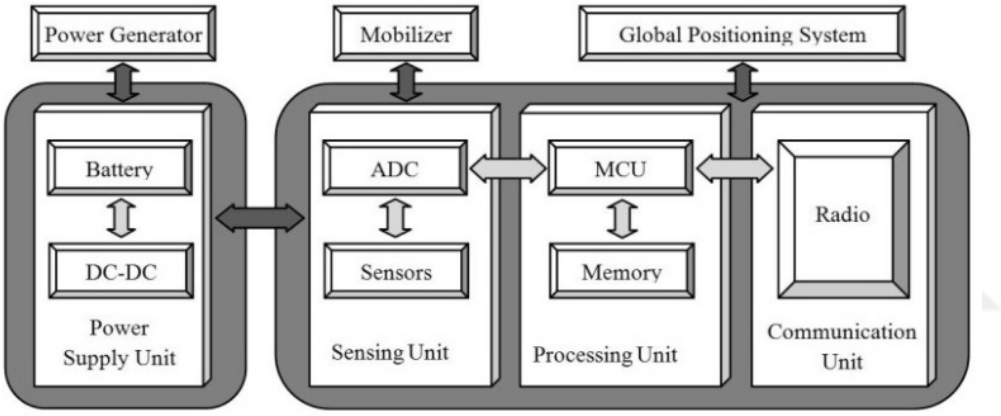
* Ev Uygulamaları
  Su, elektrik tüketim takibi
  Akıllı ev sistemleri

* Çevresel Uygulamalar
  Mikroklima
  Yanardağ takibi
  Doğal afet(sel, yangın) tespiti

* Endüstriyel Uygulamalar
  Akıllı üretim bantları
  Arıza önleme sistemleri
  Barajlarda su seviyesi tespiti

2.2 Algılayıcı Düğümlerin Yapısı

Algılayıcı düğümlerin temel donanım yapısı Şekil 2.1'de görülmektedir. Duruma göre faklı sensörler, çevre birimler ve hatta güneş panelleri de bu donanımlara entegre edilebilir.

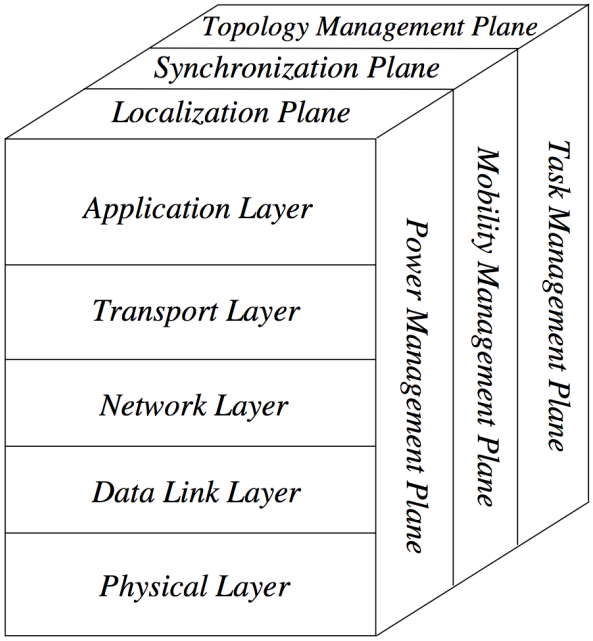


Şekil 2.1: Tipik bir algılayıcı düğüm yapısı

Şekil 2.2'de ise algılayıcı düğümlerde kullanılan yazılım katmanları görülmektedir. Bu katmanların işlevi TCP/IP protokolündeki katmanların işlevi ile neredeyse aynıdır. Fakat algılayıcı düğümlerin kısıtlı hafıza ve işlem yeteneklerinden dolayı daha hafifleştirilmiş(ing. *lightweight*) hali algılayıcı düğümler üzerinde çalışmaktadır.

Ağ yazılım katmanları ve kütüphanelerinin kullanımı bazı durumlarda avantajlı, bazı durumlarda da dezavantajlı olabilmektedir. Örneğin; yukarıda bahsedildiği gibi ağdaki bir düğümün pasif hale gelmesi durumunda algılayıcıların kendi aralarında oluşturmuş olduğu rota da geçersiz olacaktır. Bu durumda oluşan yeni bir rota belirleme ihtiyacını Şekil 2.2'de görülen Data Link Layer içerisindeki MAC katmanı üstlenecektir. Dolayısıyla en üst katmanda geliştirilen uygulama için bu işlem yükü ortadan kalkacaktır.

Dezavantajlarından en önemlisi ise güç tüketimidir. Yukarıdaki örneğin gerçeklenmesi için ağdaki algılayıcı düğümlerin çok sık aralıklara uyanmaları ve kablosuz haberleşme yapmaları gerekmektedir. Bu sebeple algılayıcı düğümlerin güç tüketimi artmaktadır.



Şekil 2.2: Algılayıcı düğümlerdeki yazılım katmanları

Diğer önemli dezavantajı ise, kritik zamanlama gerektiren durumlarda bu katmanlarda oluşan gecikmeler, uygulamayı olumsuz yönde etilemektedirler.

Bu sebeple bazı durumlarda yanlızca uygulamaya yönelik geliştirimiş basit bir yazılım katmanı kullanmak daha avantajlı olabilmektedir. Piyasada her iki seçenek de tercih edilmektedir.

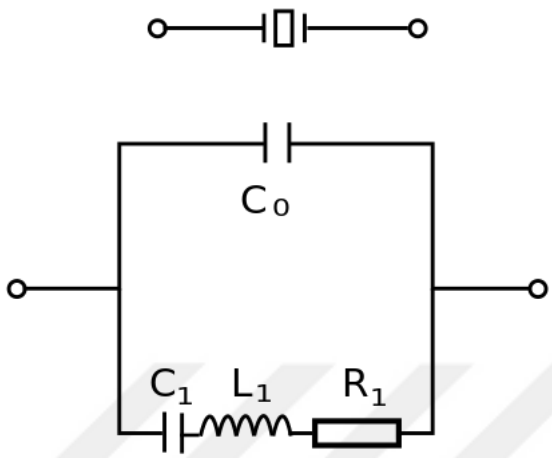
Şekil 2.3'de piyasada bulunan popüler algılayıcı düğümlerin listesi ve teknik özellikleri görülmektedir[19].

Platform	CPU	Clock (MHz)	RAM/Flash/ EEPROM	Radio Transceiver	BW (kbps)	Freq. MHz	OS	Year
WeC	Atmel AT90LS8535	4	512/8K/32K	RFM TR1000	10	916.5	TinyOS	1998
Rene 1	Atmel AT90LS8535	4	512/8K/32K	RFM TR1000	10	916.5	TinyOS	1998
AWAIRS 1	Intel StrongARM SA1100	59-206	1M/4M	Conexant RDSSS9M	100	900	MicroC/ OS	1999
µAMPS	Intel StrongARM SA1100	59-206	1M/4M	National LMX3162	1000	2400	µOS	1999
Rene 2	Atmel Atmega 163	8	1K/16K/32K	RFM TR1000	10	916.5	TinyOS	2000
Dot	Atmel Atmega 163	8	1K/16K/32K	RFM TR1000	10	916.5	TinyOS	2000
Mica	Atmel Atmega 128L	4	4K/128K/512K	RFM TR1000	40	916.5	TinyOS	2001
BT Node	Atmel Atmega 128L	8	4K/128K/4K	ZV4002 BT/ CC1000	1000	2400	TinyOS	2001
SpotON	Dragonball EZ	16	2M/2M	RFM TR1000	10	916.5		2001
Smart-its	PIC 18F252	8	3K/48K/64K	Radiometrix	64	433	Smart-its	2001
Mica2	Atmel Atmega 128L	8	4K/128K/512K	Chipcon CC1000	38.4	900	TinyOS	2002
Mica2Dot	Atmel Atmega 128L	4	4K/128K/512K	Chipcon CC1000	38.4	900	TinyOS	2002
iBadge	Atmel Atmega 128L	8	4K/128K	Ericsson ROK101007 BT	1000	2400	Palos	2002
CENS Medusa MK2	Atmel Atmega 128L/ Atmel AT91FR4081	4/40	4K/32K 136K/1M	RFM TR1000	10	916	Palos	2002
iMote	Zeevo ZV4002 (ARM)	12-48	64K/512K	Zeevo BT	720	2400	TinyOS	2003
U3	PIC 18F452	0.031-8	1K/32K/256	CDC-TR-02B	100	315	Pavenet	2003
Spec	8-bit AVR-like RISC	4-8	3K	FSK Transmitter	100		TinyOS	2003
RFRAIN	Chipcon CC1010 (8051)	3-24	2K/32K	Chipcon CC1010	76.8	0.3 - 1000	RFRAIN Libraries	2003
Nymph	Atmel Atmega 128L	4	4K/128K/512K	Chipcon CC1000	38.4	900	Mantis	2003
Telos	TI MSP430F149	8	2K/60K/512K	Chipcon CC2420	250	2400	TinyOS	2004
MicaZ	Atmel Atmega 128L	8	4K/128K	Chipcon CC2420	250	2400	TinyOS	2004
CIT Sensor Node	PIC 16F877	20	368/8K	Nordic nRF903	76.8	868	TinyOS	2004
BSN node	TI MSP430F149	8	2K/60K/512K	Chipcon CC2420	250	2400	TinyOS	2004
MITes	nRF24E1 (8051)	16	4K/512	Nordic nRF24E1	1000	2400		2004
AquisGrain	Atmel Atmega 128L	4	4K/128K/512K	Chipcon CC2420	250	2400		2004
RISE	Chipcon CC1010 EM (8051)	3-24	2K/32K	Chipcon CC1010 EM	76.8	0.3 - 1000	TinyOS	2004
Particle2/29	PIC 18F6720	20	4K/128K/512K	RFM TR1001	125	868.35	Smart-its	2004
Pluto	TI MSP430F149	8	4K/60K/512K	Chipcon CC2420	250	2400	TinyOS	2004
DSYS25	Atmel Atmega 128L	4	4K/128K	Nordic nRF2401	1000	2400	TinyOS	2004
EnOcean TCM120	PIC 18F452	10	1.5K/32K/256	Infineon TDA 5200	120	868	TinyOS	2005
eyesIFXv2	TI MSP430F1611	8	10K/48K	Infineon TDA 5250	64	868	TinyOS	2005
iMote2	Intel PXA 271	13-104	256K/32M	Chipcon CC2420	250	2400	TinyOS	2005
uPart0140 ilmt	rfPIC 16F675	4	64/1K	rfPIC 16F675	19.2	868	Smart-its	2005
TelosB/ Tmote Sky	TI MSP430F1611	8	10K/48K/1M	Chipcon CC2420	250	2400	TinyOS	2005
Ember RF Module	Atmel Atmega 128L	8	4K/128K	Ember 250	250	2400	EmberNet	2005
XYZ sensor node	OKI ML67Q500x (ARM/THUMB)	1.8-57.6	4K/256K/512K	Chipcon CC2420	250	2400	SOS	2005
Ant	TI MSP430F1232	8	256/8K	Nordic nRF24AP1	1000	2400	Ant	2005
ProSpeckz II	Cypress CY8C2764	12	256/16K	Chipcon CC2420	250	2400	Speckle net	2005
Fleck	Atmel Atmega 128L	8	4K/128K/512K	Nordic nRF903	76.8	902-928	TinyOS	2005
Sun Spot	Atmel AT91FR40162S	75	256K/2M	Chipcon CC2420	250	2400	Squawk VM (Java)	2005
ECO	nRF24E1 (8051)	16	4K/512/32K	Nordic nRF24E1	1000	2400		2006
SHIMMER	TI MSP430F1611	4/8	10K/2G	WML-C46A BT/ CC2420	250	2400	TinyOS	2006
IRIS	Atmel ATmega 1281	8	8K/640K/4K	Atmel ATRF230	250	2400	TinyOS	2007

Şekil 2.3: Piyasadaki popüler algılayıcı düğümlerin listesi

2.3 Kristal Osilatörler

Fiziksel yapısı ve bileşenleri gereği kristal osilatörler ortam şartlarına bağlı olarak her zaman stabil bir durumda çalışamazlar, bir hata değeri üretirler. Bu hata değeri, kristalin istenen frekanstan bir miktar sapmasına sebep olur.



Şekil 2.4: Kuvars kristalinin eşdeğer şeması

Örneğin 32.768kHz'lik bir saat kristalinin bir algılayıcı düğümde saniyeleri saymak amacıyla kullanıldığını düşünelim; Teorik olarak bu kristalin, donanımın içindeki çeşitli birimlerden geçtikten sonra her 1 saniyede 1 pals üretmesi beklenir. Fakat yukarıda bahsedilen sebepten dolayı bu kristaller her zaman tam olarak 1 saniyede 1 pals üretmezler(1 Hz). Kristalin ürettiği frekans bazı durumlarda 0.99999 Hz olabilirken, bazı durumlarda da 1.00001 Hz olabilir. Bu hata parametresi, kristal osilatörlerde PPM(*Parts Per Million*) olarak adlandırılmaktadır. Genellikle standart bir saat kristalinin hata değeri 10ppm veya 20ppm'dir. 20ppm lik bir kristalin 1 günde kaç saniye hata payı üreteceğini aşağıdaki örnekle daha iyi anlayabiliriz.

$$\text{hata} = 20\text{ppm} = 20/10^6$$
$$\text{gün} = 86400 \text{ saniye}$$
$$\text{1 gündeki hata} = \text{gün} * \text{hata} = 1.73 \text{ saniye}$$

2.3.1 Saat kristalleri

Günümüzde birçok elektronik cihaz içerisinde saati saymak amacıyla kullanılırlar. Genellikle hata payları 10-20ppm arasındadır.

2.3.2 Isı kontrollü kristaller(TCXO)

Bu tür osilatörlerin içerisinde ısı kontrol mekanizması bulunur. Bu ısı kontrol mekanizması, ortam sıcaklığına göre kristalin içindeki bir takım ayarlanabilir direnç veya kondansatörlerin değerini değiştirerek üretilen frekansın sürekli belirlenen değerde kalmasını sağlarlar. TCXO(ing. Temperature controlled crystal oscillator) ve OCXO(ing. oven controlled crystal oscillator) bu tür osilatör çeşitlerine örnektir.

2.3.3 Atomik saat

Dünyadaki hata payı en düşük saat çeşitidir. Atomların içerisindeki elektronların salınımı ile oluşan osilasyon frekansının tespiti ile elde edilen saat değeridir. GPS uydusu gibi hassas zaman ölçmesi istenen sistemlerde çok kararlı bir frekans çıktısı vermesinden dolayı atomik saatler kullanılırlar. Örnek olarak Caesyum atomik saatinin bir gündeki hata payı 8.46 nanosaniyedir.

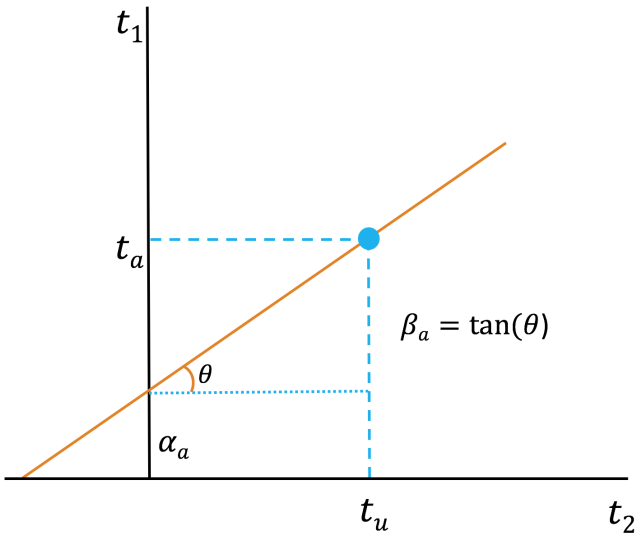
Çizelge 2.1'de bazı osilatör çeşitlerine ilişkin hata değerleri ve özellikler görülebilir.

Çizelge 2.1: Osilatör çeşitleri

Türü	**PPM Değeri**	**Doğruluk**	**Yıllanma(10 yıl-20 yıl)**
Kristal	10ppm-100ppm	10^{-5}	10-20ppm - $10x10^{-6}$
TCXO	1ppm	10^{-6}	3ppm - $3x10^{-6}$
OCXO(10MHz)	0.02ppm	$2x10^{-8}$	0.2ppm - $0.2x10^{-6}$
OCXO(100MHz)	0.5ppm	$5x10^{-7}$	0.01ppm - 10^{-8}
Ribidyum	10^{-6}ppm	10^{-12}	0.005ppm - $5x10^{-9}$

3. EŞZAMANLAMA ALGORİTMASI

Herhangi iki KAA algılayıcısı arasındaki zaman ilişkisini kestirebilmek için doğrunun eğimi denkleminden faydalanılabilir.



Şekil 3.1: Doğrunun eğimi

$$t_a = \alpha_a + \beta_a t_u \tag{3.1}$$

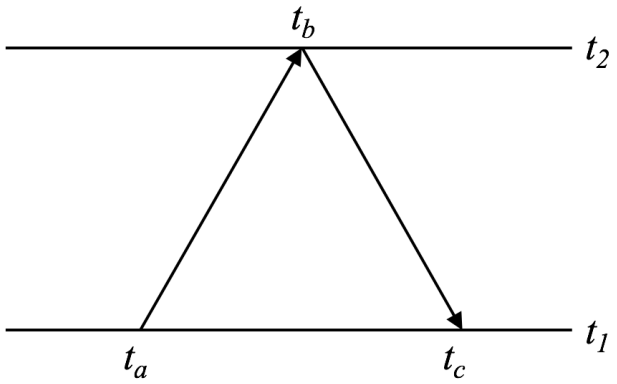
Bu denklemde, t_a bir KAA algılayıcısının (algılayıcı-a) saat değeri, α_a ofset, β_a algılayıcı-a'nın saatinin kayma değeri ve t_u da evrensel zamandır [20–24]. Genel kabul saat zaman kaymasını değişmez bir sabit almak yönündedir. Aslında göreceli olarak kısa sürelerde bu varsayımın doğruluğu pek çok deneysel çalışmada teyid edilmiştir. Böylece, çözülmesi gereken problem, evrensel bir referansa göre kayma ve ofset değerlerinin doğru bir şekilde tahmin edilmesine indirgenebilmektedir.

Şekil 3.1'de görüldüğü gibi α_a ve β_a bilinirse herhangi bir t_u anında t_a zamanı, Denklem 3.1 ile hesaplanabiir. Bu denklemi iki algılayıcı arasındaki ilişkiyi temsil edecek şekilde değiştirdiğimizde ortaya çıkan yeni denklem aşağıdaki gibi olacaktır.

$$t_1 = \alpha_{12} + \beta_{12} t_2 \tag{3.2}$$

Bu denklemde, t_1 ve t_2 algılayıcı-1 ve algılayıcı-2'nin yerel saatlerini, α_{12} algılayıcı-1 ve algılayıcı-2'nin saatleri arasındaki göreceli ofseti ve β_{12} de algılayıcı-1 ve algılayıcı-2'nin saatleri arasındaki göreceli kaymayı ifade etmektedir.

Örneğin, algılayıcı-1'in algılayıcı-2'nin iç saatini tahmin etmek istediğini varsayarsak, Şekil 3.2'deki eşzamanlama şemasında verilen algoritma ve denklem 3.2'den yararlanarak α_{12} ve β_{12} değerlerini elde edebiliriz.



Şekil 3.2: Eşzamanlama Şeması.

Şekil 3.2'de görüleceği üzere algılayıcı-1, t_a anında algılayıcı-2'ye kendi zaman bilgisini içeren bir paket gönderir. Algılayıcı-2 bu mesaja kendi zaman bilgisini(t_b) ekleyip tekrar algılayıcı-1'e gönderir. Algılayıcı-1, t_c anında algılayıcı-2'ye ait zaman bilgisini(t_b) elde etmiş olur. Bu aşamadan sonra aşağıdaki eşitlikleri hesaplayabiliriz.

$$t_a = \alpha_{12} + \beta_{12} t_b \tag{3.3}$$

$$\alpha_{12} = t_a - t_b \tag{3.4}$$

$$\beta_{12} = \frac{t_a - \alpha_{12}}{t_b} \tag{3.5}$$

Şekil 3.1'de verilen t_u değerini t_b olarak düşünürsek, sıfır noktasından t_b zamanına kadar geçen süre, α noktasından t_a noktasına kadar geçen süreye eşit olacaktır. Çünkü bu birim zamandır ve her iki taraf için eşittir. Bu sebeple, $t_a - \alpha_{12} = t_b$ eşitliğinden faydalanarak denklem 3.14'ü yazabiliriz.

Denklem 3.5'e bakıldığında ise teorik olarak β_{12}'nin her zaman 1 değerini aldığı açıkça görülebilir. Fakat pratikte bu değer tam olarak 1 değerini vermemektedir. Bunun sebebi, Bölüm 2.3'de bahsedilen saat kristalinin karakteristik özelliği ile ilgilidir. Bu sebeple, şekil 3.1'de verilen doğrunun eğimini tek bir örnekle tam olarak kestirebilmek mümkün olmamaktadır. β_{12} ile kestirilecek doğru için ne kadar fazla nokta tespit

edilirse eğim ölçüde gerçeğe yakın olacaktır. Bunun için algılayıcı-1 ve algılayıcı-2 arasında bir miktar örnekleme toplama işlemi gerekmektedir.

Yeteri kadar örnek verisi(*timestamp*) toplandıktan sonra algılayıcılar, birbirleri arasındaki kayma ve ofset değerlerini kestirebilir. Bunun için aşağıda verilen denklemlerden yararlanılmaktadır[25].

$$\beta_{12a}(i) = (t_a(i) - t_a(i-1))/(t_b(i) - t_b(i-1)) \tag{3.6}$$

$$\beta_{12b}(i) = (t_c(i) - t_c(i-1))/(t_b(i) - t_b(i-1)) \tag{3.7}$$

Denklem 3.6 ve denklem 3.7, kayma(β_{12}) değeri için sırasıyla alt ve üst sınırları ifade etmektedir. $\beta_{12a}(i)$, t_a ve t_b'nin, $\beta_{12b}(i)$ ise t_b ve t_c'nin i zamanında alınan örneği için hesaplanan değeri ifade etmektedir.

$$\alpha_{12a}(i) = t_a(i) - \beta_{12a}(i)t_b(i) \tag{3.8}$$

$$\alpha_{12b}(i) = t_c(i) - \beta_{12b}(i)t_b(i) \tag{3.9}$$

Denklem 3.6 ve denklem 3.7'ten yararlanarak algılayıcılar arasındaki ofset değeri bulunabilir. Denklem 3.8 ve denklem 3.9, ofset(α_{12}) değeri için sırasıyla alt ve üst sınırları ifade etmektedir.

$$\beta_{12}(i) = [\beta_{12a}(i) + \beta_{12b}(i)]/2 \tag{3.10}$$

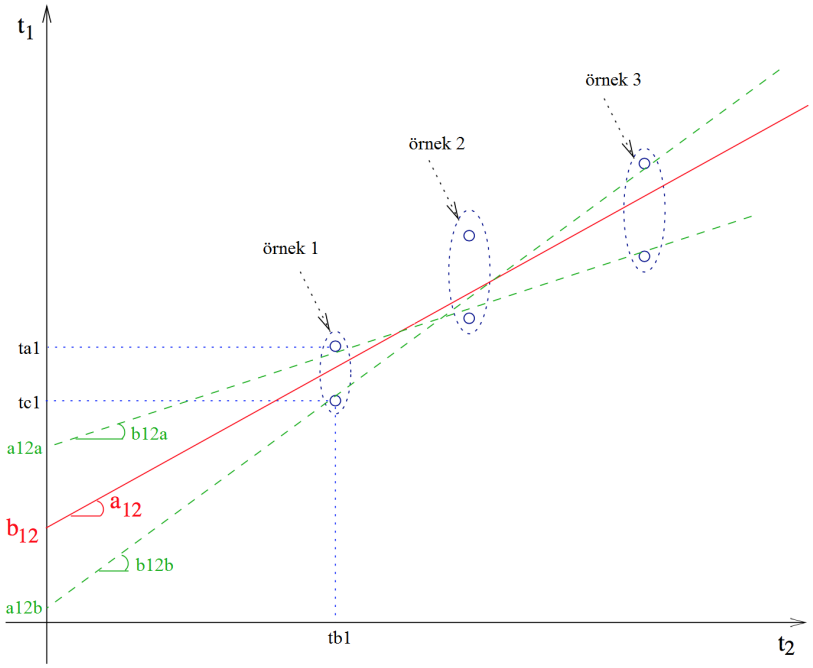
$$\alpha_{12}(i) = [\alpha_{12a}(i) + \alpha_{12b}(i)]/2 \tag{3.11}$$

Alt ve üst sınırlar belirlendikten sonra bu iki değerin ortalamasını alarak ikisi arasında gerçeğe daha yakın bir değer elde edilmektedir.

$$\beta_{12avg} = \frac{1}{N}\sum_{i=1}^{N}\beta_{12}(i) \tag{3.12}$$

$$\alpha_{12avg} = \frac{1}{N}\sum_{i=1}^{N}\alpha_{12}(i) \tag{3.13}$$

Son olarak toplanan tüm örneklerin (N adet) ortalaması aldındığında gerçeğe en yakın β_{12} ve α_{12} değerini kestirmek mümkün olmaktadır.



Şekil 3.3: Örnek sayısının zaman kestirimine etkisi

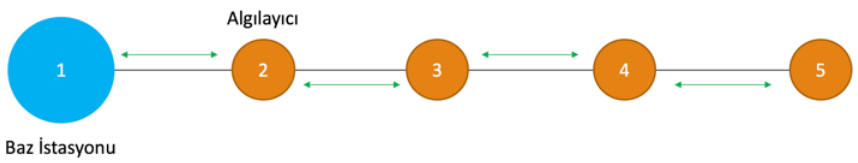
Şekil 3.3'de görüleceği gibi alınan her bir örnek, gerçek β_{12} değerine yaklaşabilmek için katkı sağlamaktadır.

Örnekleme işlemi tamamlandıktan sonra algılayıcı-1, herhangi bir t_a anında Denklem 3.14 aracılığıyla algılayıcı-2'nin zamanını(t_{b-est}) tahmin edilebilir.

$$t_{b-est} = \frac{t_a - \alpha_{12avg}}{\beta_{12avg}} \tag{3.14}$$

4. SİSTEM MODELİ

Bir Kablosuz Algılayıcı Ağında algılayıcılar arasında zaman senkronizasyonu yapmak temel olarak iki şekilde mümkündür.



Şekil 4.1: Tek/Çok atlamalı eşzamanlama modeli.

1. Algılayıcı-1 ve algılayıcı-2'nin zaman senkronizasyonu yapması.
2. Algılayıcı-1 ve algılayıcı-5'in çok atlamalı olarak zaman senkronizasyonu yapması.

4.1 Tek Atlamalı Model

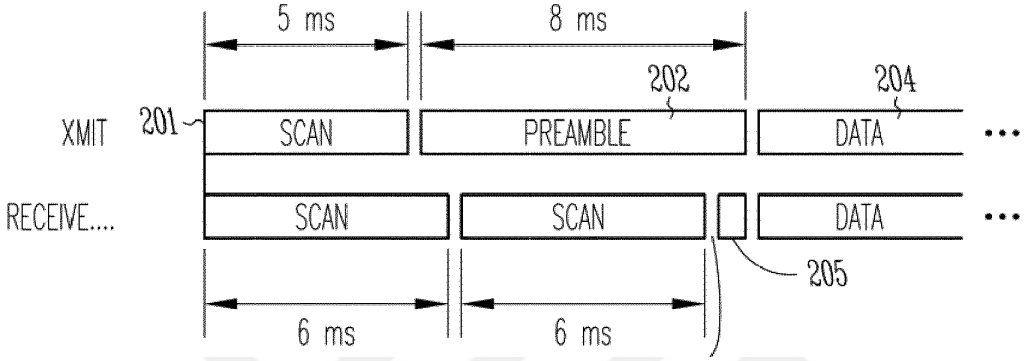
Bu modelde Şekil 4.1'de görüleceği gibi algılayıcılar kendi arasında zaman senkronizasyonu yapar. Örneğin, baz istasyonu(algılayıcı-1) ile algılayıcı-2 kendi aralarında zamanlarını senkronize ettikten sonra alıglayıcı-2 ve algılayıcı-3 de kendi aralarında zaman senkronizasyonu yaparlarsa, baz istasyonu ile algılayıcı-3 senkronize olmuş olur. Bu modeli ağdaki tüm algılayıcılarda ikili olarak uygulamak mümkündür.

4.2 Çok Atlamalı Model

Bu modelde herhangi bir algılayıcı diğer algılayıcılardan biriyle çok atlamalı bir şekilde ikili olarak zaman senkronizasyonu yapmaktadır. Örneğin algılayıcı-1'in algılayıcı-5 ile zaman senkronizasyonu işlemi yapmak istediğini düşünürsek, zaman bilgisini içeren paketler sırayla algılayıcı-2-3-4 üzerinden algılayıcı-5'e iletilecektir. Aynı şekilde algılayıcı-5 de göndermek istediği paketi sırayla algılayıcı-4-3-2 üzerinden algılayıcı-1'e gönderecektir.

4.3 Ağ Trafiği

Gerek tek atlamalı gerekse çok atlamalı modelde karşılaşılabilecek en önemli problemlerden biri kablosuz kanaldaki ağ trafiğidir. Çünkü ortamdaki yoğun ağ trafiği, zaman bilgisi içeren paketlerin karşı tarafa doğru bir şekilde iletilmesini engelleyebilir. Örneğin biz bir paketi karşı tarafa göndermek istediğimizde kullandığımız kablosuz kanalda başka bir cihaz aktif ise, veya kanal gürültülü ise bu paket karşı tarafa tamamıyla aktarılamayabilir. Bu durumda göndermek istediğimiz paketi yeniden göndermemiz gerekir.



Şekil 4.2: *Listen Before Talk* gönderme/alma diyagramı.

Pratikte paketler arasındaki gecikmeler arttıkça algoritmanın zamanı doğru tahmin edebilme ihtimali azalmaktadır. Örneğin, bu algoritmanın 802.11 Wi-Fi altyapısı üzerinde çalıştırıldığı düşünülürse, gönderimlerin başarısız olmasından dolayı bazı gecikmeler meyana gelecektir. Algılayıcılar t_a zamanında paket gönderim komutu verseler bile CSMA/CA[26] implementasyonunun 802.11 MAC katmanı içerisinde gömülü olarak gelmesinden dolayı, önlenemeyen, tespit edilemeyen ve müdahalesi mümkün olmayan gecikmeler meydana gelecektir. Ve bu gecikmeler ortamdaki algılayıcı sayısına bağlı olarak artacaktır. Bu ise pratikte algoritmanın hata payının belirgin şekilde artmasına sebep olmaktadır. Fakat bu çalışmada herhangi bir ağ yazılım kütüphanesi ve katmanı kullanılmadığı için yukarıda belirtilen durumlarda oluşan istenmeyen gecikmeler olsa bile bu gecikmeler tespit edilebilmektedir. Yukarıda bahsedilen gecikmelerin tespit edilme aşamaları aşağıdaki gibidir:

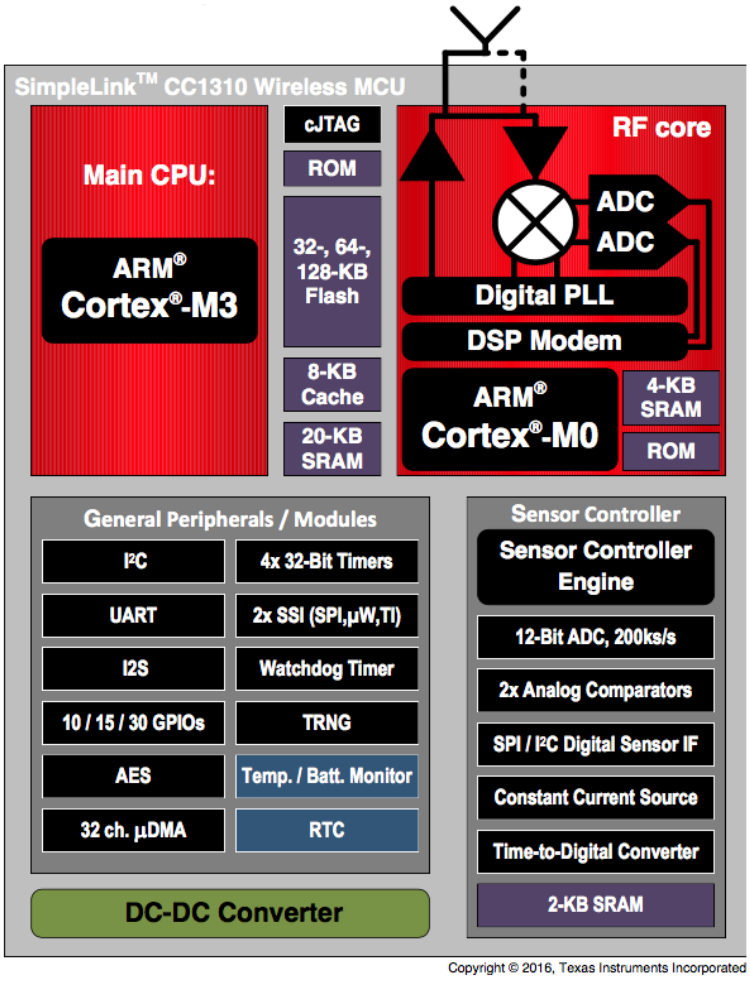
1. t_a zamanında paket göndermek üzere bir zamanlayıcı çalıştır.
2. t_a zamanı geldiğinde kablosuz kanalı kontrol et.
3. Eğer kanal meşgul değilse paketi gönder.
4. Eğer kanal meşgul ise 2. adıma git ve arada geçen süreyi pakete ekle.

2 ve 3. adımlar aslında CSMA/CA yönteminin aşamaları içinde de bulunmaktadır. Fakat bu yöntemin implementasyonu elle yapıldığı için(buna *Listen Before Talk*[27] özelliği denilmektedir) araya 1 ve 4. adımlar da katılarak bu özellik yararlı bir hale dönüştürülmüştür(Şekil 4.2).

Yukarıda belirtilen adımların gerçekleştirilmesi ile ortamda yoğun veri trafiği olsa bile veya saati ayarlanmak istenen algılayıcılar arasında birden fazla algılayıcı olsa bile saat tahmin algoritması yüksek doğrulukla çalışmaktadır.

5. SİSTEM TASARIMI

Deneyleri gerçekleştirmek için Texas Instruments firmasının CC1310 ARM Cortex-M3 mikrodenetleyicisi kullanılmıştır (Şekil 5.1). CC1310 mikrodenetleyicisi, 128KB flash hafıza, 20KB RAM, IEEE 802.15.4g uyumlu 868 MHz kablosuz alıcı/verici birimini ve bu birimi kontrol eden ARM Cortex-M0 işlemcisini üzerinde barındırmaktadır.



Şekil 5.1: CC1310 Mikrodenetleyicisi ve çevrebirimleri.

CC1310 mikrodenetleyicisinin bu deneyin minimum hata payı ile gerçekleştirilmesine büyük katkı sağlayan 2 önemli özelliği vardır. Bunlar;

1. Tam olarak belirlenen zamanda paket gönderme özelliği.
2. Paket alımı sırasında otomatik *timestamp* kaydetme özelliği.

Bu özelliklerin deneylere olan katkısını anlamak için Şekil 3.2'yi tekrar gözden geçirelim.

Şekil 3.2'ye bakıldığında $t_a - t_b$ arasında geçen bir süre olduğu görülmektedir. Bu süre, paketin karşı tarafa iletimi için geçen süredir(*time of flight*). Bu süreyi d_{TX} olarak adlandıralım. d_{TX} süresi farklı yollarla tespit edilebilir. Bunlar;

a) d_{TX} = *gönderilen bit sayısı / haberleşme hızı(bps)* formülü ile,

b) Gönderimin başladığı ve bittiği anlarda bir pini 1-0 yapıp bu süreyi osiloskop ile ölçerek,

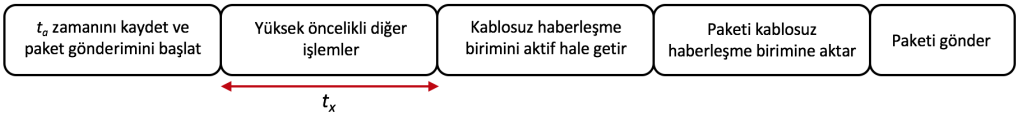
c) Eğer varsa kullanılan RF çipinin donanım özellikleri yardımıyla,

d_{TX} süresi ölçülebilir. Bu yöntemleri, bölümün ilerleyen kısımlarında daha detaylı olarak değerlendireceğiz.

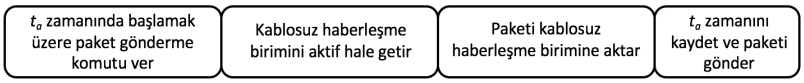
d_{TX} süresi normalde Denklem 3.8 - 3.13 arasında verilen hiçbir denklemde hesaba katılmamıştır. Çünkü bu süre her paket için aynı olacağından(paket uzunluğu sabit varsayıldığında) dolayı bu değeri bir kez ölçüp kestirilen t_b zamanına(t_{b-est}) eklemek yeterlidir. Fakat burada d_{TX} süresinin minimum hata ile tespit edilmesi çok önemlidir. Aksi halde t_{b-est} değeri hatalı çıkacaktır. Yukarıda bahsedilen iki özellik sayesinde (1 ve 2. maddeler), d_{TX} süresinin çok hassas bir şekilde tespit edilmesi sağlanmıştır.

Tam olarak belirlenen zamanda paket gönderme özelliği şu şekilde çalışmaktadır: Şekil 3.2'de paketi tam olarak t_a zamanında gönderdiğimizi varsayıyoruz. Fakat aslında t_a zamanında mikrodenetleyiciye *"paketi gönder"* komutu veriliyor. Şeikil 5.2'de görüleceği gibi mikrodenetleyici bu komutu aldıktan sonra bir takım işlemlere devam ediyor. Komutun işletilmesi, kablosuz birimin aktif hale getirilmesi, paketin kablosuz birimin işlemicisine aktarılması gibi işlemler t_a anında *"paketi gönder"* komutu verildikten sonra gerçekleşiyor. Örneğin bir gerçek zamanlı işletim sistemi kullanılıyorsa(*RTOS*), *"paketi gönder"* komutu verildikten sonra araya daha yüksek öncelikli bazı işlemler(*task*) girebilir. Bu sebeple t_a zamanından sonra duruma bağlı olarak bazı gecikmeler oluşabilir. Bu gecikmeler Şekil 5.2'de t_x olarak belirtilmiştir. Dolayısıyla t_x bilinmeyen değerinin de t_a süresine eklenmesi gerekir. Fakat t_x süresinin tespiti, pratikte mikrodenetleyici içerisinde mümkün olsa bile yazılım karmaşıklığını ciddi boyutta artıracaktır. Bu aşamada *"t_a zamanında paketi gönder"* şeklinde çalışmak beraberinde birçok riski getirecektir.

CC1310 mikrodenetleyicisi bize *"paketi t_a zamanında gönder"* şeklinde bir özellik sunmaktadır. Bu özellik sayesinde yukarıda belirtilen gecikmelerin tamamından izole



Şekil 5.2: Paket gönderimi için standart yöntem



Şekil 5.3: Paket gönderimi için gecikmelerden etkilenmeyen yöntem

olmak mümkündür. Örneğin $t_a - 5(ms)$ anında *"paketi t_a zamanında gönder"* komutu verildiğini düşünelim. Bu 5 milisaniyelik süre içerisinde CC1310, paketin gönderilmesi için gerekli olan komutu işleme alacak, kablosuz birimi aktif edecek, paketi kablosuz birimin işlemcisine aktaracak ve t_a zamanı geldiğinde paketi gönderecektir.

Sonuç olarak yukarada belirtilen 'a' maddesindeki yöntemle sadece matematiksel bir hesap yapılmaktadır. Çünkü kullanılan kablosuz haberleşme entegresi ve mikrodenetleyiciye göre değişiklik gösterebilen birçok süre parametresi bu yöntemde göz ardı edilmiştir.
'b' maddesindeki yöntemle ölçülen süre ise Şekil 5.2'de belirtilen t_x süresinden dolayı uygulamaya veya farklı koşullara bağlı olarak değişiklik gösterebilecektir.
'c' maddesindeki yöntem ise uygulamaya bağlı olmayıp dinamik olarak her zaman daha kararlı sonuç verecektir.

İkinci özellik olan *Paket alımı sırasında otomatik timestamp kaydetme özelliği* ise şu şekilde çalışmaktadır: Şekil 5.4'deki paket formatına dikkat edilirse *Sync Word* bilgisinin de paket içerisinde olduğu görülmektedir. Bu sistem, kısaca aşağıdaki şekilde çalışmaktadır:

1. Algılayıcı-1 göndermek istediği paketi t_a zamanında göndermek üzere kablosuz haberleşme birimini aktif hale getirir.
2. t_a zamanında algılayıcı-1 önce *Preamble* bitleri göndererek algılayıcı-2'yi uyandırır.
3. Algılayıcı-2 uyandığı anda algılayıcı-1'den *Sync Word* bilgisini bekler.
4. Algılayıcı-1 *Sync Word* bilgisini gönderir.
5. Algılayıcı-2 *Sync Word* bilgisini alır. Bu bilgi eğer doğru ise (yani paket kendine ait ise) paketin devamını almak üzere kablosuz haberleşme devam eder.

Bu özellik sayesinde haberleşmenin iki taraf arasında başarılı bir şekilde başladığı andaki zaman değerini(*timestamp*) elde etmek mümkündür. Bu değeri t_{b-rs} olarak ad-

Şekil 5.4: Paket formatı.

landıralım. Paket alımı bittiğindeki zamanlayıcı değerini de t_{b-re} olarak adlandıralım. $t_{b-rs} - t_{b-re}$ değeri bize paketin karşı tarafa aktarımı için geçen süreyi net olarak verecektir. Bu sürenin matematiksel hesabı Şekil 5.4'deki paket formatının uzunluğu bilinerek yapılabilir.

- Preamble = 8 byte
- Sync Word = 4 byte
- Length = 1 byte
- Packet = 20 byte
- CRC = 2 byte
- Toplam **35** byte = 35x8 bit

$$d_{TX} = p/s \tag{5.1}$$

Denklem 5.1 p bitin s hızında gönderilme süresini(milisaniye cinsinden) göstermektedir. p bit/saniye cinsinden gönderilen bit miktarını temsil etmektedir. s değeri ise kilo-bit/saniye cinsinden haberleşme hızını temsil etmektedir. p değeri, toplam paket boyutu hesabından 280 bit olarak bulunabilir. s değeri ise sabittir ve 50 kbps olarak alınabilir. p ve s değeri Denklem 5.1'de yerine konulduğunda 35 byte lık bir paketin 50 kbps hızında gönderilme süresi d_{TX} = 5.6 milisaniye olarak çıkmaktadır.

Hesaplanan d_{TX} değeri, aslında $t_{b-rs} - t_{b-re}$ değerine eşittir. Fakat d_{TX} değeri teorik olarak her zaman aynı çıksa da pratikte her zaman aynı çıkmamaktadır. Çünkü 2. maddede belirtilen *Preamble* gönderimi aşamasında genellikle uzun bit dizileri gönderilmektedir. Örneğin 64 bitlik bir *Preamble* bit dizisi gönderildiğini düşünelim. Algılayıcı-2 bu bit dizisinin başında, ortasında veya sonlarında uyanabilir. Bu sebeple algılayıcı-2'nin toplamda aldığı bit sayısı, yani d_{TX} değişiklik gösterebilir. Yani 8 byte lık bir *Preamble* dizisinin son 2 byte ında uyanma durumuna geçen bir alıcı, toplam 6 byte eksik alacaktır. 6 byte ın d_{TX} süresine etkisi; $6x8/50 = 0.96$ milisaniyedir. Sabit paket süresi hesaplanılırsa aslında geçmemiş olan bu 0.96 milisaniyelik süre de hesaba katılmaktadır. Bu sebeple sabit paket süresi hesaplamak yerine bu yöntemi kullanmak d_{TX} süresini neredeyse hatasız olarak tespit etmeyi sağlamaktadır.

5.1 Eşzamanlama Örneklerinin İncelenmesi

Çizelge 5.1'de somut örnek olması açısından algılayıcı-1 ve algılayıcı-2'den alınan 10 örnek için t_a, t_b, α_{12} ve β_{12} değerleri verilmiştir. t_a ve t_b değerleri mikrosaniye cinsindendir.

Çizelge 5.1: Algılayıcılardan alınan örnekler ve hesaplanan değerler

i	t_a	t_b	β_{12}	α_{12}
1	500244	46063750	0	0
2	1000244	46563696	1.00010801166526	-45568481
3	1500244	47063645	1.00010201040506	-45568201
4	2000244	47563594	1.00010201040506	-45568201
5	2500244	48063544	1.00010001000100	-45568106
6	3000244	48563488	1.00011201254541	-45568683
7	3500244	49063436	1.00010401081713	-45568295
8	4000244	49563385	1.00010201040506	-45568196
9	4500244	50063335	1.00010001000100	-45568098
10	5000244	50563284	1.00010201040506	-45568197

$$i >= 2, i <= 10 \tag{5.2}$$

$$\beta_{12}(i) = \frac{t_a(i) - t_a(i-1)}{t_b(i) - t_b(i-1)} \tag{5.3}$$

$$\alpha_{12}(i) = t_a(i) - \beta_{12}(i) t_b(i) \tag{5.4}$$

5.2 Yazılım

Bölüm 2.2'de bahsedildiği gibi, ağ yazılım kütüphanesi kullanımı ekstra gecikmelere neden olabilmektedir. Yapılan deneylerde zamanlama, çok kritik bir rol oynadığı için bu çalışmada CC1310 mikrodenetleyicisi üzerinde herhangi bir ağ yazılım kütüphanesi(ing. *software network stack*) kullanılmamıştır. Bunun yerine bazı yapılar doğrudan alt seviye kodlar olarak gerçeklenmiştir(ing. *low level implementation*). Aşağıda hiyerarşik olarak gerçeklenen katman sıralaması görülmektedir. Yazılım, Texas Instruments TI-RTOS gerçek zamanlı işletim sistemi üzerine inşa edilmiştir.

- Listen Before Talk Algoritması
- CC1310 RF haberleşme birimi sürücü yazılımı
- IEEE 802.15.4g 50 kbps GFSK fiziksel katmanı(PHY)

5.3 Donanım

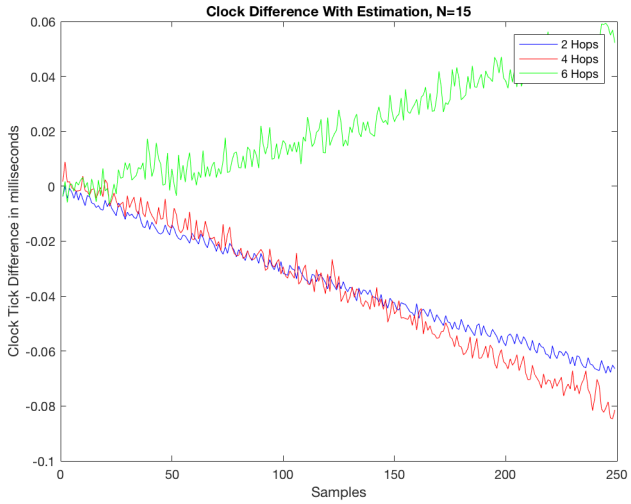
Deneyleri gerçekleşitrmek ve kablosuz haberleşmede atlama sayısını arttırabilmek amacıyla küçük boyutlu ek bir devre tasarımı gerçekleştirilmiştir. Bu tasarıma *nodETU* adı verişmiştir. *nodETU*, her cihaza rahatlıkla entegre olabilecek şekilde tasarlanmıştır. Harici anten gereksinimi olmaması, 10ppm 25 MHz kristal osilatör ile çalışması ve 2x3 cm boyutunda olması *nodETU* nün önemli özelliklendendir.

Ayrıca, deneylerde örnek toplama işleminin uzun sürmesinden dolayı baz istasonu olarak çalışan algılayıcı modül, RaspberryPi tek kart bilgisayarına USB üzerinden bağlanmıştır. RaspberryPi ise Wi-Fi üzerinden internete bağlanmıştır. Böylece, toplanan örnekler, uzaktan erişim vasıtasıyla doğrudan RapsberryPi üzerinden okunmuştur. Bu yöntem, çalışmaları önemli ölçüde kolaylaştırmıştır. Kullanılan donanımlara ilişkin resimler ve çizimler EK 1-2-3-4'te verilmiştir.

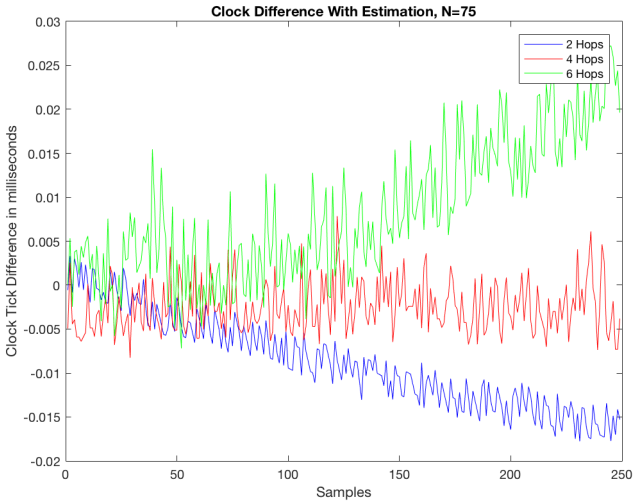
Ortam trafiği oluşturmak için ise, 1 adet CC1310 modülü, 1-50ms rastgele seçilen zaman aralığında test düzeneği ile aynı kablosuz kanalda çalışacak şekilde programlanmış ve test ortamına eklenmiştir. Rastgele üretilen trafiğin zaman aralıklarını temsil eden grafik çıktısı Şekil 6.8'de görülmektedir.

6. DENEYSEL İNCELEME

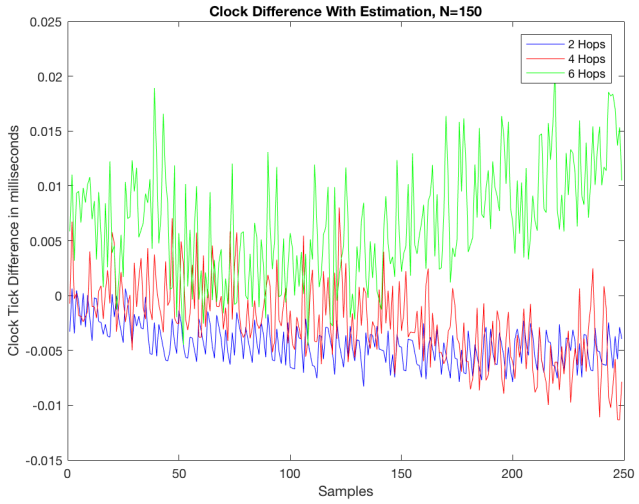
Farklı örnek(N) ve atlama sayıları için elde edilen grafikler Şekil 6.1 - 6.6'da gösterilmiştir.



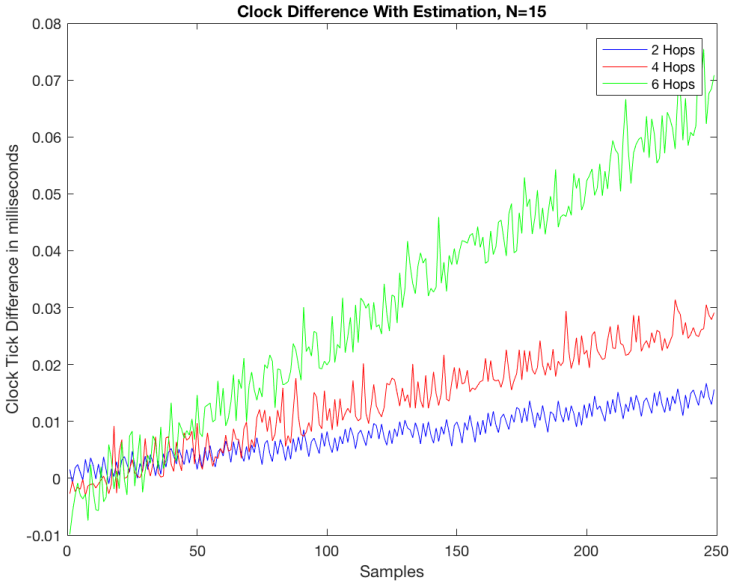
Şekil 6.1: N=15 için ofset miktarı



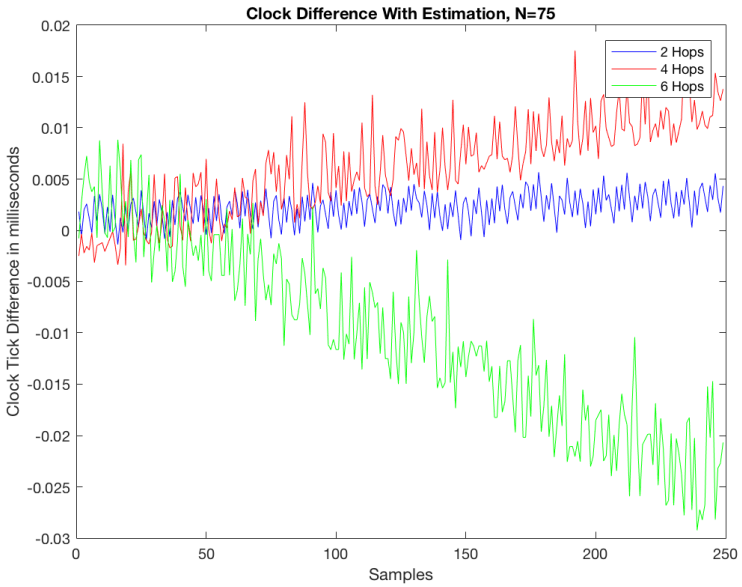
Şekil 6.2: N=75 için ofset miktarı



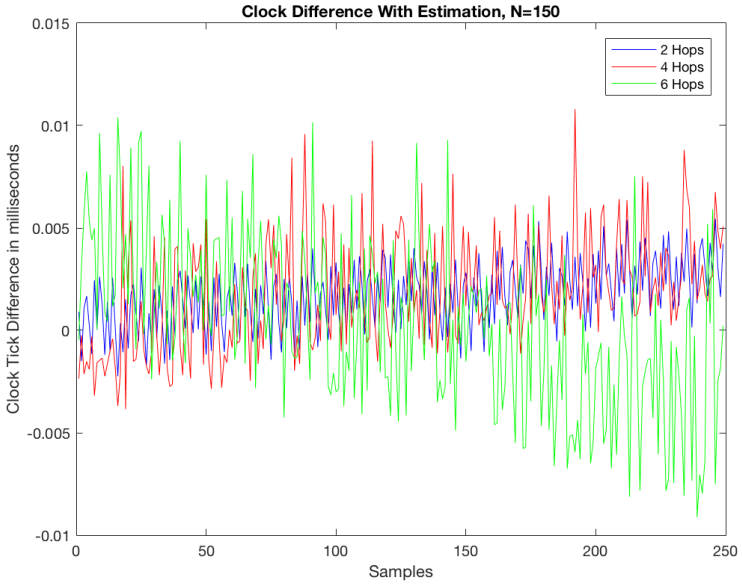
Şekil 6.3: N=150 için ofset miktarı



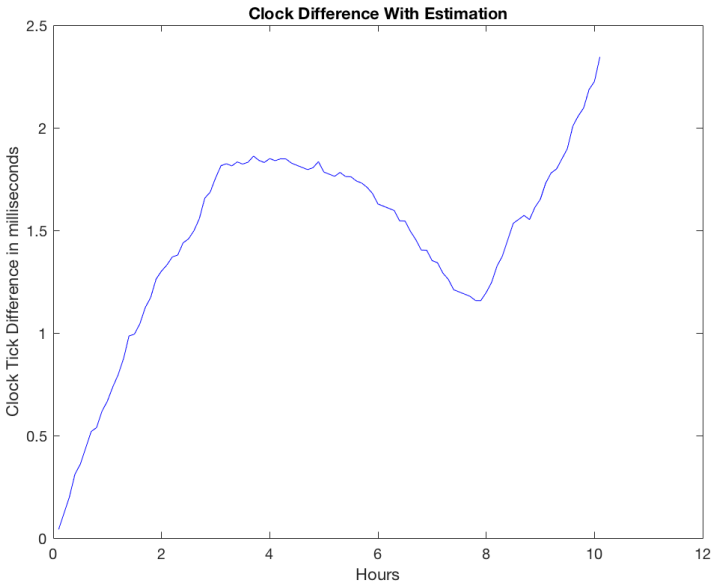
Şekil 6.4: Ağ trafiği altında N=15 için ofset miktarı



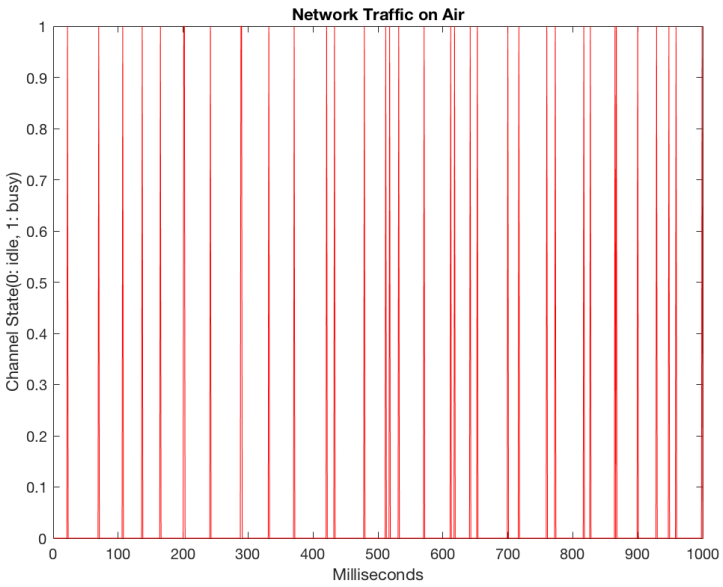
Şekil 6.5: Ağ trafiği altında N=75 için ofset miktarı



Şekil 6.6: Ağ trafiği altında N=150 için ofset miktarı



Şekil 6.7: 10 saat sonraki ofset miktarı



Şekil 6.8: Havadaki ağ trafiği

7. SONUÇLAR

Bu çalışmada KAA'larda eşzamanlama işleminin düşük karmaşıklıkta bir yöntemle gerçeklenmesi amaçlanmıştır. Bu amaçla, çok atlamalı KAA'larda çalışmak üzere bir yöntem tasarlanmış ve gömülü bir sistem üzerinde gerçeklenmiştir. Yöntem deneysel olarak irdelenmiş ve mikrosaniyeler seviyesinde zamanlama hatalarına kadar inilebildiği gözlemlenmiştir. Literatürde çok atlamalı KAA'larda harici trafik altında deneysel olarak eşzamanlama üzerine ilk çalışma olma özelliği çalışmamızın oldukça özgün bir yanıdır.

Ayrıca deneyler saniyede 1 örnek alınarak gerçekleştirilmiştir. Grafik çıktılarından da görüleceği üzere Şekil 1.1'de 250 saniyedeki(örnek) 20 milisaniyelik fark, tasarlanan sistemin hassas zamanlama tespit edebilmesinden dolayı mikrosaniyeler mertebesine indirgenmiştir. Ayrıca örnekleme miktarının kullanılan sistemlerdeki kristalin karakteristiğini tespit edebilmeye olan büyük etkisi de gözlemlenmiştir. Şekil 6.6'dan da anlaşılacağı üzere, bu algoritmanın kullanılacağı sistemlerde doğru örnekleme miktarı seçilirse, atlama sayısı ve ağ trafiğine bağımlılığın büyük ölçüde göz ardı edilebileceği gözlemlenmiştir.

KAYNAKLAR

[1] **J. Elson and D. Estrin,** Time Synchronization for wireless sensor networks. In Proceedings of the 15th International Parallel & Distributed Processing Symposium, IPDS '01, pages 186-193, 2001.

[2] **F. Sivrikaya and B. Yener,** Time synchronization in sensor networks: a survey. IEEE Network, vol. 18, no. 4, pp. 45-50, July 2004.

[3] **I.-K. Rhee, J. Lee, J. Kim, E. Serpedin, and Y.-C. Wu,** Clock syn- chronization in wireless sensor networks: An overview, Sensors, vol. 9, no. 1, pp. 56–85, 2009.

[4] **J. Elson and K. Römer,** "Wireless sensor networks: A new regime for time synchronization," ACM SIGCOMM Computer Commununications Review, vol. 33, no. 1, pp. 149–154, Jan. 2003.

[5] **L. Schenato and G. Gamba,** "A distributed consensus protocol for clock synchronization in wireless sensor network," in Decision and Control, 2007 46th IEEE Conference on, Dec 2007, pp. 2289–2294.

[6] **L. Schenato and F. Fiorentin,** "Average timesynch: A consensus-based protocol for clock synchronization in wireless sensor networks," Auto- matica, vol. 47, no. 9, pp. 1878 – 1886, 2011.

[7] **T. Kunz and E. McKnight-MacNeil,** "Clock synchronization in wsn: Si- mulation vs. implementation," in Wireless Communications and Mobile Computing Conference (IWCMC), 2011 7th International, July 2011, pp. 980–985.

[8] **O. Mirabella, M. Brischetto, A. Raucea, and P. Sindoni,** "Dynamic con- tinuous clock synchronization for ieee 802.15.4 based sensor networks," in Industrial Electronics, 2008. IECON 2008. 34th Annual Conference of IEEE, Nov 2008, pp. 2438–2444.

[9] **Z. Yang, L. Cai, Y. Liu, and J. Pan,** "Environment-aware clock skew estimation and synchronization for wireless sensor networks," in INFO- COM, 2012 Proceedings IEEE, March 2012, pp. 1017–1025.

[10] **T. Schmid, R. Shea, Z. Charbiwala, J. Friedman, M. B. Srivastava, and Y. H. Cho,** "On the interaction of clocks, power, and synchronization in duty-cycled embedded sensor nodes," ACM Transactions on Sensor Networks, vol. 7, no. 3, pp. 24:1–24:19, Oct. 2010.

[11] **Z. Zhong, P. Chen, and T. He,** "On-demand time synchronization with predictable accuracy," in INFOCOM, 2011 Proceedings IEEE, April 2011, pp. 2480–2488.

[12] **H. Cho, J. Jung, B. Cho, Y. Jin, S.-W. Lee, and Y. Baek,** "Precision time synchronization using ieee 1588 for wireless sensor networks," in Computational Science and Engineering, 2009. CSE '09. International Conference on, vol. 2, Aug 2009, pp. 579–586.

[13] **F. Ren, C. Lin, and F. Liu,** "Self-correcting time synchronization using reference broadcast in wireless sensor network," Wireless Communica- tions, IEEE, vol. 15, no. 4, pp. 79–85, Aug 2008.

[14] **M. Sichitiu and C. Veerarittiphan,** "Simple, accurate time synchroni- zation for wireless sensor networks," in Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE, vol. 2, March 2003, pp. 1266–1273 vol.2.

[15] **M. Maróti, B. Kusy, G. Simon, and A. Lédeczi,** "The flooding time synchronization protocol," in Proc. International Conference on Em- bedded Networked Sensor Systems, ser. SenSys '04, 2004, pp. 39–49.

[16] **S. Yoon, C. Veerarittiphan, and M. L. Sichitiu,** "Tiny-sync: Tight time synchronization for wireless sensor networks," ACM Transactions on Sensor Networks, vol. 3, no. 2, pp. 8:1–8:34, Jun. 2007.

[17] **H. Dai and R. Han,** "Tsync: A lightweight bidirectional time synchroni- zation service for wireless sensor networks," ACM SIGMOBILE Mobile Computing and Communications Review, vol. 8, no. 1, pp. 125–139, Jan. 2004.

[18] **M. Healy, T. Newe, and E. Lewis.** Wireless sensor node hardware: A review. In 2008 IEEE Sensors, pages 621–624, Oct 2008.

[19] **I. F. Akyıldız and M. Vuran,** Wireless Sensor Networks. Wiley, 2009.

[20] **L. Ferrigno, V. Paciello, and A. Pietrosanto,** "Experimental characteri- zation of synchronization protocols for instrument wireless interface," Instrumentation and Measurement, IEEE Transactions on, vol. 60, no. 3, pp. 1037–1046, March 2011.

[21] **S. Lee, U. Jang, and J. Park,** "Fast fault-tolerant time synchronization for wireless sensor networks," in Object Oriented Real-Time Distributed Computing (ISORC), 2008 11th IEEE International Symposium on, May 2008, pp. 178–185.

[22] **J. Chen, Q. Yu, Y. Zhang, H.-H. Chen, and Y. Sun,** "Feedback-based clock synchronization in wireless sensor networks: A control theoretic approach," Vehicular Technology, IEEE Transactions on, vol. 59, no. 6, pp. 2963–2973, July 2010.

[23] **J. Elson, L. Girod, and D. Estrin,** "Fine-grained network time synchroni- zation using reference broadcasts," ACM SIGOPS Opereration Systems Review, vol. 36, no. SI, pp. 147–163, Dec. 2002.

[24] **F. Goncalves, L. Suresh, R. Lopes Pereira, J. Trindade, and T. Vazao,** "Light-weight time synchronization for wireless sensor networks," in Future Internet Communications (CFIC), 2013 Conference on, May 2013, pp. 1–8.

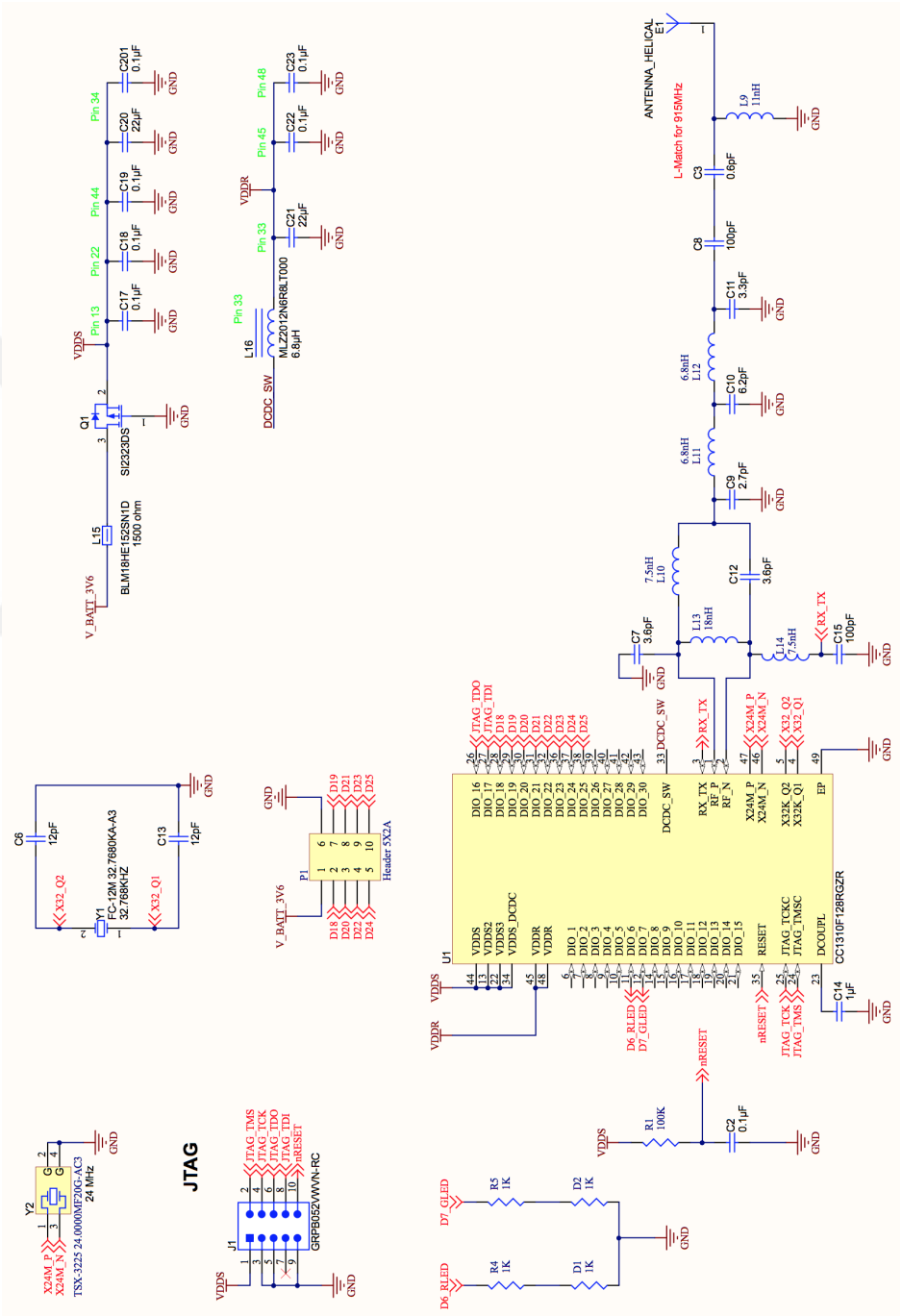
[25] **G. Bam, E. Dilcan, B. Dogan, B. Dinc and Bulent Tavli**. Dlwts: Distributed light weight time synchronization for wireless sensor networks. IEEE International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS), 2015.

[26] **Z. Dahham, A. Sali, B. M. Ali, and M. S. Jahan.** An efficient csma-ca algorithm for ieee 802.15.4 wireless sensor networks. In 2012 International Symposium on Telecommunication Technologies, pages 118–123, Nov 2012.

[27] **H.D.P. Ferguson.** Listen before talk frequency agile radio synchronization, Janu- ary 2 2013. EP Patent App. EP20,120,174,512.

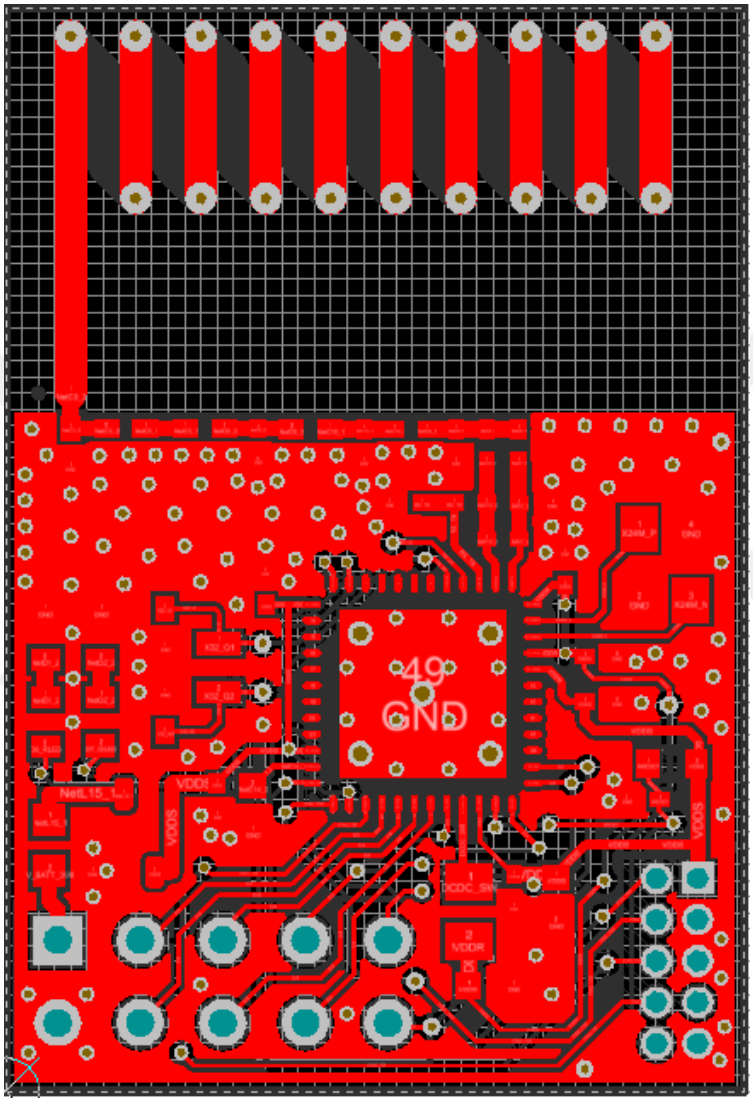
EKLER

EK 1



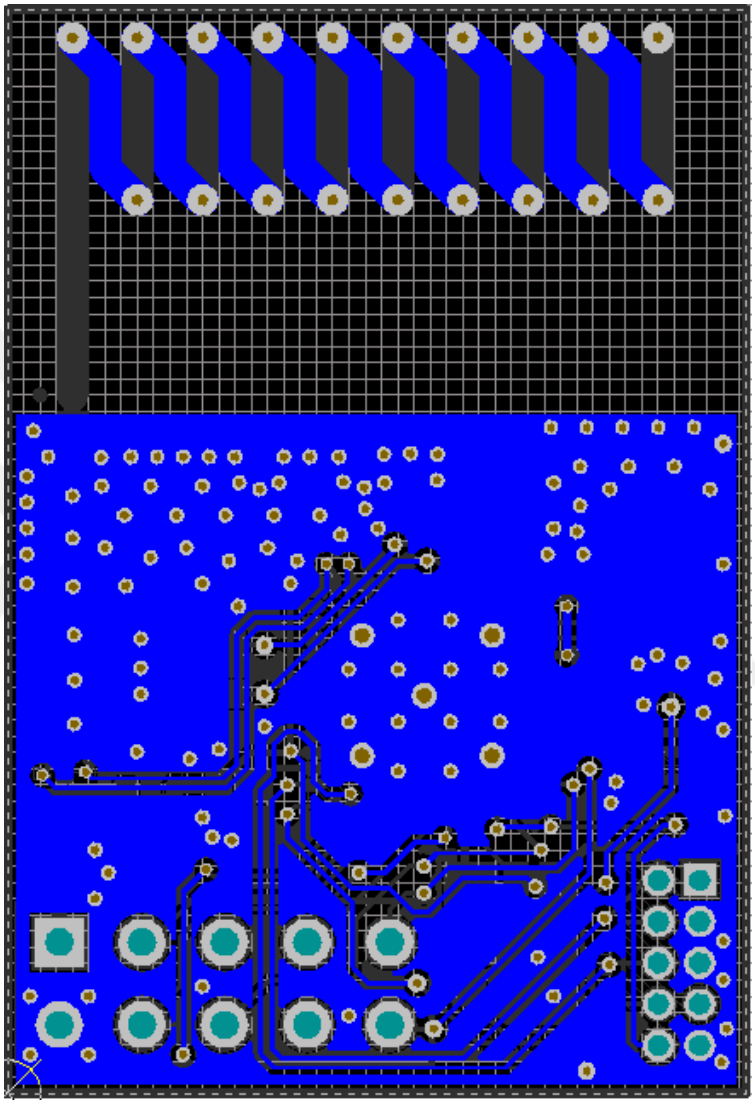
Şekil Ek.1: nodETU Devre şeması

EK 2



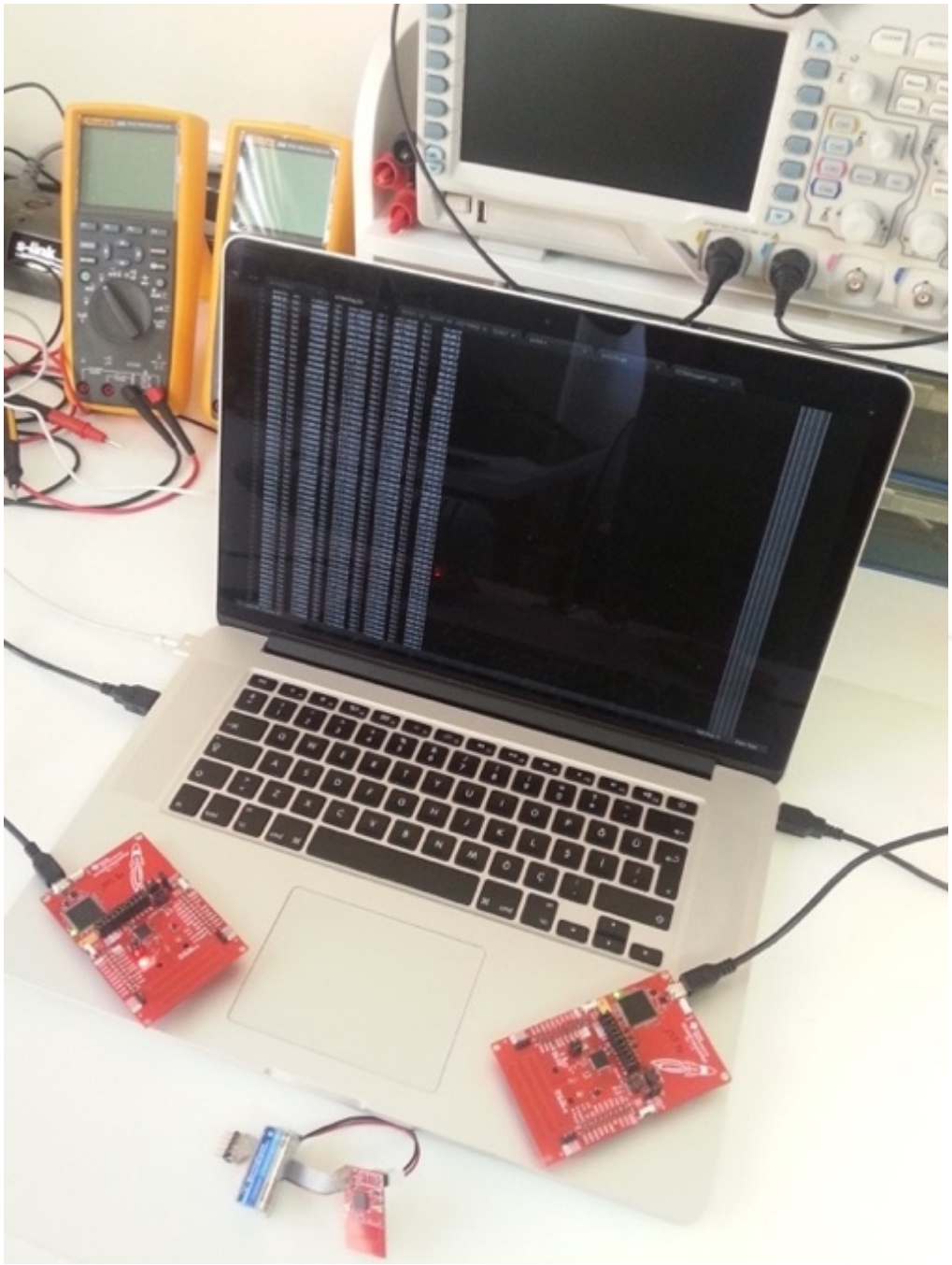
Şekil Ek.2: nodETU PCB üstten görünüm

EK 3



Şekil Ek.3: nodETU PCB alttan görünüm

EK 4



Şekil Ek.4: Test düzeneği

ÖZGEÇMİŞ

Ad-Soyad : Muhammed Fatih İnanç

Uyruğu : T.C.

Doğum Tarihi ve Yeri : 1989, ANKARA

E-posta : fatihinanc@etu.edu.tr

ÖĞRENİM DURUMU:

- **Lisans** : 2012, Selçuk Üniversitesi, Bilgisayar Sistemleri Öğretmenliği
- **Yüksek Lisans** : 2017, TOBB ETÜ, Elektrik ve Elektronik Mühendisliği

MESLEKİ DENEYİM VE ÖDÜLLER:

Yıl	**Yer**	**Görev**
2013	Metlab Metroloji	Gömülü Yazılım Mühendisi
2017	CTech Bilişim	Gömülü Yazılım Mühendisi

YABANCI DİL: İngilizce

TEZDEN TÜRETİLEN YAYINLAR, SUNUMLAR VE PATENTLER:

- **İnanç, M.F., Tavlı, B.,** "Çok Atlamalı Kablosuz Algılayıcı Ağlar için Dağıtık Bir Eşzamanlama Tekniğinin Tasarımı ve Deneysel İncelenmesi." 25. Sinyal İşleme ve İletişim Uygulamaları Kurultayı (SİU 2017)