KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ELEKTRİK - ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

UZAKTAN HASTA TAKİP SİSTEMİ İÇİN IEEE 802.15.6 ESASLI KABLOSUZ VÜCUT ALAN AĞI HABERLEŞMESİNİN GERÇEKLEŞTİRİLMESİ

YÜKSEK LİSANS TEZİ

EleKtrik - Elektronik Müh. Hasan Yavuz ÖZDERYA

HAZİRAN 2017
TRABZON

KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

UZAKTAN HASTA TAKİP SİSTEMİ İÇİN IEEE 802.15.6 ESASLI KABLOSUZ VÜCUT ALAN AĞI HABERLEŞMESİNİN GERÇEKLEŞTİRİLMESİ

Hasan Yavuz ÖZDERYA

Karadeniz Teknik Üniversitesi Fen Bilimleri Enstitüsünce
"ELEKTRONİK YÜKSEK MÜHENDİSİ"
Unvanı Verilmesi İçin Kabul Edilen Tezdir.

Tezin Enstitüye Verildiği Tarih : 30/05/2017
Tezin Savunma Tarihi : 15/06/2017

Tez Danışmanı : Doç. Dr. İsmail KAYA

Trabzon 2017

KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

Elektrik-Elektronik Mühendisliği Anabilim Dalında
Hasan Yavuz ÖZDERYA tarafından hazırlanan

UZAKTAN HASTA TAKİP SİSTEMİ İÇİN IEEE 802.15.6 ESASLI KABLOSUZ VÜCUT ALAN AĞI HABERLEŞMESİNİN GERÇEKLEŞTİRİLMESİ

başlıklı bu çalışma, Enstitü Yönetim Kurulunun 31 / 05 / 2017 gün ve 1704 sayılı kararıyla oluşturulan jüri tarafından yapılan sınavda
YÜKSEK LİSANS TEZİ
olarak kabul edilmiştir.

Jüri Üyeleri

Başkan : Doç. Dr. İsmail KAYA .....

Üye : Doç. Dr. Kadir TÜRK .....

Üye : Doç. Dr. Birol SOYSAL .....

Prof. Dr. Sadettin KORKMAZ
Enstitü Müdürü

ÖNSÖZ

Bu çalışma uzaktan hasta takip sistemlerinde kullanılmak üzere, hasta üzerindeki taşınabilir ölçüm cihazlarının kablosuz bağlantısını sağlayacak bir Vücut Alan Ağı haberleşme sisteminin, IEEE 802.15.6 standardı Ortam Erişim Katmanı temel alınarak, donanım ve yazılımının geliştirilmesini amaçlamaktadır.

Yüksek lisans tezimde, danışmanlığımı üstelenerek maddi ve manevi desteğini esirgemeyen saygıdeğer hocam Doç. Dr. İsmail KAYA'ya teşekkürü borç bilirim. Çalışmamda yol gösteren ve destek olan hocam Prof. Dr. Temel KAYIKÇIOĞLU'na ve Dr. Yusuf BALTACI'ya yardımlarından ötürü teşekkür ederim. Eğitim öğretim hayatım boyunca daima arkamda duran ve desteğini esirgemeyen babama, ağabeyime ve bütün aileme sevgilerimi sunarım.

Bu çalışma 114E452 nolu 1003 projesi kapsamında TÜBİTAK tarafından desteklenmiştir.

TEZ ETİK BEYANNAMESİ

Yüksek Lisans Tezi olarak sunduğum "Uzaktan Hasta Takip Sistemi için IEEE 802.15.6 Esaslı Kablosuz Vücut Alan Ağı Haberleşmesinin Gerçekleştirilmesi" başlıklı bu çalışmayı baştan sona kadar danışmanım Doç. Dr. İsmail KAYA'nın sorumluluğunda tamamladığımı, verileri/örnekleri kendim topladığımı, deneyleri/analizleri ilgili laboratuvarlarda yaptığımı/yaptırdığımı, başka kaynaklardan aldığım bilgileri metinde ve kaynakçada eksiksiz olarak gösterdiğimi, çalışma sürecinde bilimsel araştırma ve etik kurallara uygun olarak davrandığımı ve aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul ettiğimi beyan ederim. 15/06/2017

Hasan Yavuz ÖZDERYA

İÇİNDEKİLER

Sayfa No

Yüksek Lisans

ÖZET

UZAKTAN HASTA TAKİP SİSTEMİ İÇİN IEEE 802.15.6 ESASLI KABLOSUZ VÜCUT ALAN AĞI HABERLEŞMESİNİN GERÇEKLEŞTİRİLMESİ

Hasan Yavuz ÖZDERYA

Karadeniz Teknik Üniversitesi
Fen Bilimleri Enstitüsü
Elektrik - Elektronik Anabilim Dalı
Danışman: Doç. Dr. İsmail KAYA
2017, 69 Sayfa, 4 Sayfa Ek

Kablosuz iletişim teknikleri ve batarya teknolojilerinin gelişmesiyle birlikte kablosuz cihazlar hayatımızın her alanına girmiştir. Ancak tıbbi kullanım alanlarında kablosuz iletişim teknolojileri yaygınlaşmamıştır. Var olan kablosuz iletişim teknolojileri tıbbi cihazlardan beklenen güvenilirliği sunamamaktadır. Bluetooth, Zigbee gibi teknolojiler beklenen iletişim güvenilirliği için gerekli yapıları tanımlamamakta veya güç tüketimi açısından verimli çalışmamaktadırlar. Wi-Fi, 4G gibi teknolojiler yüksek bant genişliği sunsalar da alıcı verici entegrelerinin büyük olmaları ve yüksek güç tüketimleri sebebiyle hasta üzerindeki sensörlerde kullanılmaya uygun değildirler. Bu tezde uzaktan hasta takip sistemlerinde kullanılmak üzere, kablo eşdeğeri güvenilirlik sunacak, kesintisiz aktarım için optimize edilmiş bir kablosuz haberleşme sistemi geliştirilmesi hedeflenmiştir.

Bu çalışma hasta vücudu üzerine yerleştirilecek ölçüm cihazları ve kablosuz erişim cihazı için kullanılacak haberleşme donanımı ve yazılımlarının geliştirilmesini kapsamaktadır. IEEE tarafından 2012 yılında yayınlanan 802.15.6 Wireless Body Area Networks (Kablosuz Vücut Alanı Ağları) standardı temel alınarak kablosuz haberleşme protokolünün Ortam Erişim Katmanı (MAC) gerçekleştirilmiştir. Ortam erişim katmanının yanı sıra algılayıcı cihazı üzerindeki farklı donanımların yönetimini ve iletişimini sağlayan, modüler bir servis katmanı Makineden Makineye İletişim (M2M) prensiplerine uygun olarak geliştirilmiştir.

Anahtar Kelimeler: Kablosuz Haberleşme, 802.15.6, M2M, Teletıp

Master Thesis

SUMMARY

IMPLEMENTATION OF A WIRELESS BODY AREA NETWORK BASED ON IEEE 802.15.6 FOR REMOTE PATIENT MONITORING SYSTEM

Hasan Yavuz ÖZDERYA

Karadeniz Technical University
The Graduate School of Natural and Applied Sciences
Electrical and Electronics Engineering Program
Supervisor: Assoc. Prof. İsmail KAYA
2017, 69 Pages, 4 Pages Appendix

Developments in wireless and battery technologies has been putting wireless equipment every part of our lives. But widespread use of wireless medical equipment has yet to come. Existing wireless communication technologies can not provide the reliability expected from a medical equipment. Technologies such as Bluetooth and ZigBee has not been designed with extreme reliability and security in focus and they are not ideal from a power efficiency perspective. Although Wi-Fi and 4G can provide reliable and high bandwidth connections, the fact that they require relatively big circuitry and their high power usage, eliminates them from use of on body patient sensors. In this thesis, development of a wireless communication system that should provide wire equivalent reliability, capable of uninterrupted streaming for use in remote patient monitoring applications is aimed.

This work encompasses the development of a wireless communication module that will be placed on the patient body and a gateway device hardware along with its wireless communication stack. Implementation of the Medium Access Control Layer is based on the 802.15.6 Wireless Body Area Networks specification that is released by IEEE organization in 2012. Along with MAC layer, a modular service layer that manages different sensors and features of monitoring devices according to Machine to Machine (M2M) principles has been developed.

Key Words: Wireless Communication, 802.15.6, M2M, BAN, Telemedicine

ŞEKİLLER DİZİNİ

Sayfa No

TABLOLAR DİZİNİ

Sayfa No

SEMBOLLER DİZİNİ

ACK	:	Acknowledgment (Alındı)
BAN	:	Body Area Network (Vücut Alan Ağı)
CA	:	Collision Avoidance (Çakışma Önleme)
CSMA/CA	:	Carrier Sense Multiple Access / Collision Avoidance (Taşıyıcı Algılamalı Çoklu Erişim / Çakışma Önleme)
DBPSK	:	Differential Binary Phase Shift Keying (Diferansiyel İkili Faz Kaymalı Anahtarlama)
DQPSK	:	Differential Quadrature Phase Shift Keying (Diferansiyel Dörtlü Faz Kaymalı Anahtarlama)
D8PSK	:	Differential Eight Phase Shift Keying (Diferansiyel Sekizli Faz Kaymalı Anahtarlama)
FCC	:	Federal Communications Commission (Federal İletişim Komisyonu – Amerika Birleşik Devletleri)
FCS	:	Frame Check Sequence (Paket Kontrol Dizisi)
FEC	:	Forward Error Correction (Önden Hata Düzeltme)
FIFO	:	First In First Out (İlk Giren İlk Çıkar Bellek)
GATT	:	Generic Attribute Profile (Genel Öznitelik Profili)
GMSK	:	Gaussian Minimum Shift Keying (Gauss Minimum Kaydırmalı Anahtarlama)
GTK	:	Group Temporal Key (Geçici Grup Anahtarı)
HAL	:	Hardware Abstraction Layer (Donanım Ayırma Katmanı)
HBC	:	Human Body Communications (İnsan Vücudu İletişimi)
I2C	:	Inter-Integrated Circuit
ID	:	Identification (Kimlik Numarası)
IoT	:	Internet of Things (Nesnelerin İnterneti)

MAC	:	Medium Access Control (Ortam Erişim Kontrolü)
MAP	:	Managed Access Phase (Yönetilen Erişim Fazı)
OO-QPSK	:	Offset Orthogonal – Quadrature Phase Shift Keying (Ofsetli Dikey – Dörtlü Faz Kaydırmalı Anahtarlama)
PHY	:	Physical (Fiziksel Katman)
PLL	:	Phase Locked Loop (Faz Kenetleme Döngüsü)
PLCP	:	Physical Layer Convergence Protocol (Fiziksel Katman Yakınsama Protokolü)
PSDU	:	Physical Layer Service Data Unit (Fiziksel Katman Servis Katman Veri Birimi)
PTK	:	Pair Temporal Key (Geçici Eşleşme Anahtarı)
QoS	:	Quality of Service (Servis Kalitesi)
RAP	:	Random Access Phase (Rastgele Erişim Fazı)
RT	:	Realtime (Gerçek Zamanlı)
RTOS	:	Realtime Operating System (Gerçek Zamanlı İşletim Sistemi)
SPI	:	Serial Peripheral Interface (Seri Donanım Arayüzü)
SRRC	:	Square-Root Raised Cosine (Yükseltilmiş Kosinüsün Kare Kökü)
UUID	:	Universally Unique ID (Evrensel Özgün Kimlik Numarası)
UWB	:	Ultra Wide Band -(Çok Geniş Bant)

1. GENEL BİLGİLER

1.1. Giriş

Elektronik komponentlerin küçülmesi ve batarya teknolojisi alanındaki gelişmeler kablosuz haberleşmeyi hayatımızın bir çok parçasına sokmaya başlamıştır. Her geçen gün yeni bir kablosuz cihazın ya da var olan bir cihazın kablosuz modelinin icat edildiğine veya piyasaya sürüldüğüne şahit oluyoruz. Kablosuz teknolojisindeki bu ivmeye rağmen tıbbi cihaz sektörü kablosuz ekipmanlara karşı temkinli yaklaşmaktadır. Bunun temel sebebi olarak var olan teknolojilerin tıbbi ekipmanlardan beklenen güvenilirliği garanti edememesi gösterilebilir. Ancak yaşlanmakta ve gelir seviyesi yükselmekte olan dünya nüfusunun tıbbı ihtiyaçları da değişmektedir. Yaşlanan nüfusla birlikte, kalp hastalıkları diyabet gibi sürekli kontrol altında tutulması gereken hastalıklar yaygınlaşmaktadır. Hastanelerin kapasitesi bu ihtiyacı karşılayacak hızda artmadığı için, alternatif teknikler araştırılmaktadır. Bunlardan biri hastanın internet ve kablosuz iletişim teknikleri kullanılarak evinden takip edilmesidir. Bu da kablosuz teknolojiler arasında, tıbbi kullanım odaklı bir teknolojinin geliştirilmesi ihtiyacını doğurmaktadır.

1.2. Uzaktan Hasta Takip Sistemleri

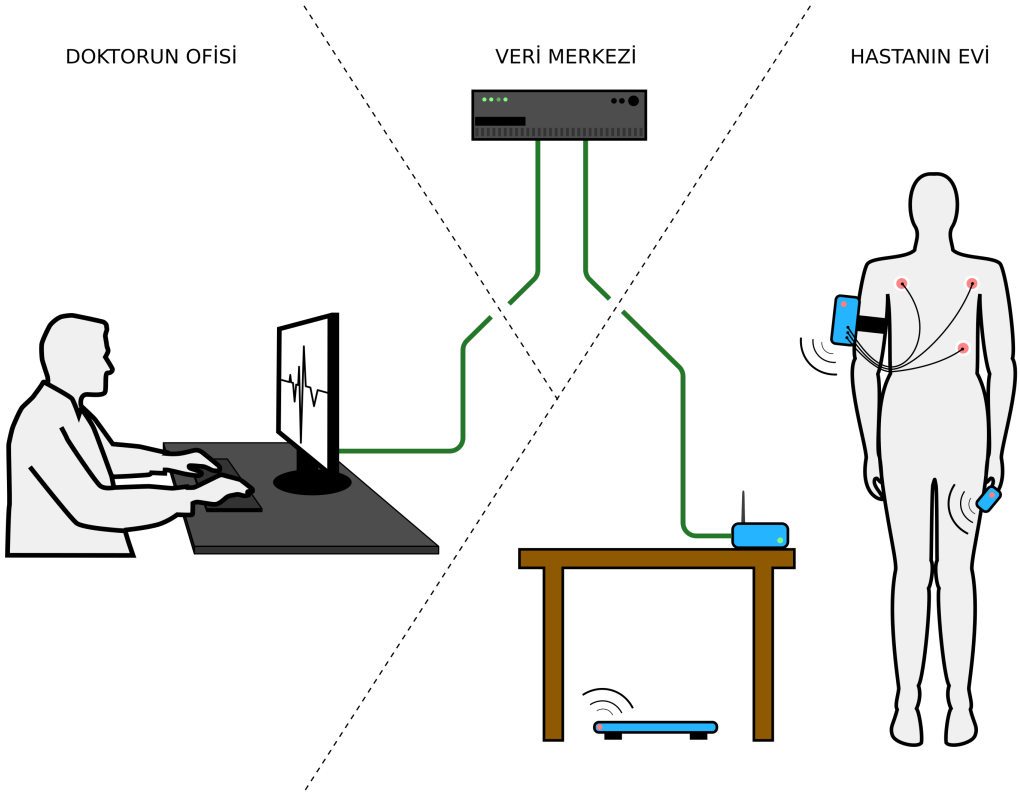
Günümüzün en popüler tıbbi ilerleme konularından biri uzaktan hasta takip sistemleridir. Diyabet ve kalp hastalıkları gibi modern rahatsızlıklar doğaları itibariyle hasta için çok acil bir durum temsil etmemekle birlikte, dikkatli bir şekilde takip edilmedikleri taktirde ani ölümle sonuçlanabilmektedir.

Tıptaki diğer ilerlemeler ve modern yaşam tarzının sonucu olarak yaşlı nüfus artmaktadır ve sonuç olarak bu şekildeki hastalıkların miktarı da artmaktadır. Sadece hastanın yakında takibi ile kontrol altında tutulabilecek ve ölümcül vakaların engellenebileceği bu hastalıkların hastaneler üzerinde oluşturduğu yük artmaktadır.

Bu probleme karşı sunulan bir çözüm uzaktan hasta takip sistemidir. Bu sistemlerde hasta gerekli ölçüm cihazlarıyla birlikte evine gönderilir. Bu cihazlar bazen doğrudan hastanın üzerine yerleştirilir. Cihaz sürekli olarak hasta üzerinden ilgili ölçümleri yapar.

Diğer durumlarda hasta periyodik olarak cihazı kullanarak ölçüm yapar. Sürekli ölçüm yapan bir cihaza örnek olarak EKG cihazı, hastanın manuel olarak ölçüm yaptığı bir cihaza örnek olarak ise tansiyon ölçüm cihazı veya tartı verilebilir. Bu ölçümler otomatik olarak internet üzerinden hastane tarafından yönetilen veri merkezine gönderilir. Veri merkezinde toplanan veriler doktorlar tarafından incelenerek hastanın durumu takip altında tutulur. Hatta veri merkezine bağlı bilgisayarlarda çalıştırılan otomatik algoritmalar vasıtasıyla, kalp krizi gibi kritik durumların otomatik olarak, hızlı bir şekilde tespit edilmesi dahi mümkündür.

1.2.1. Uzaktan Hasta Takip Sisteminin Bileşenleri



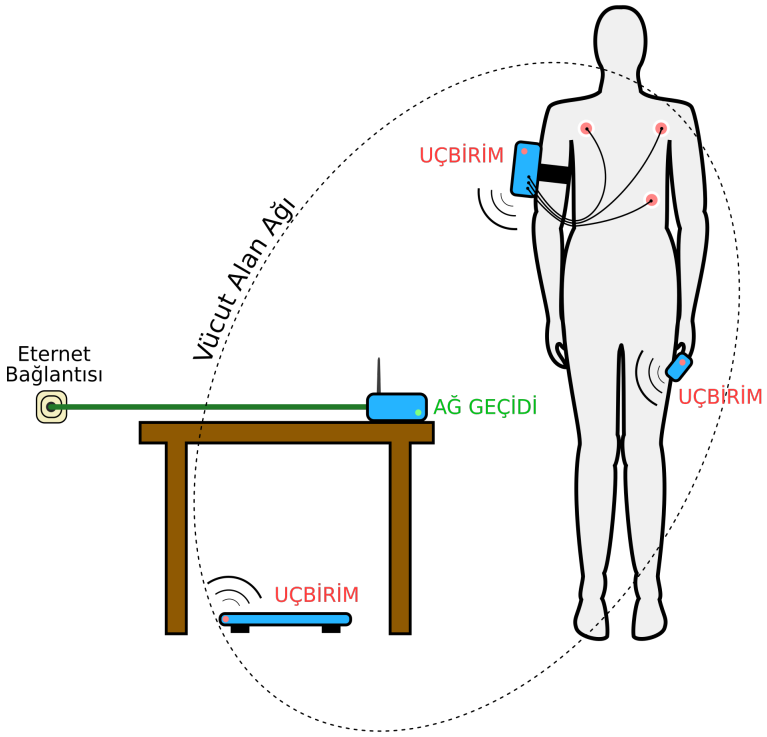
Şekil 1. Uzaktan hasta takip sisteminin genel yapısı

Şekil 1'de bir uzaktan hasta takip sisteminin temel bileşenleri verilmiştir. Verilerin toplanması hastanın üzerine yerleştirilen sensörler ile gerçekleştirilmiştir. Bu diyagramda kablosuz sensörler gösterilmiştir. Bu sensörlerden toplanan bilgi hastanın hemen yakınındaki kablosuz ağ geçidine gönderilir. Ağ geçidinde toplanan bu bilgiler internet ağı

üzerinden veri merkezine iletilir. Doktor kendi ofisindeki bir terminal (kişisel bilgisayar gibi) aracılığıyla veri merkezinde toplanan hasta bilgilerine erişebilir. Hastanın durumunu anlık olarak kontrol edebilir. Bu sistemde hasta üzerindeki sensörler ile ağ geçidi arasında kurulan ağa vücut alanı ağı (BAN : Body Area Network) denilmektedir.

1.2.2. Vücut Alan Ağının Yapısı

Şekil 2'de vücut alan ağının temel öğeleri detaylı olarak gösterilmiştir. Şekilde görüldüğü gibi ağın merkezinde bir ağ geçidi cihazı bulunmaktadır. Düğüm cihazları bu ağ geçidine kablosuz olarak bağlanırlar.



Şekil 2. Kablosuz vücut alan ağının temel öğeleri

Ağ geçidinin görevi ağı kurmak ve düğümlerin bağlantısını yönetmektir. Ağ geçidi hasta üzerinden toplanan tıbbi verinin internet üzerinden veri merkezine gönderilmeden önce toplandığı noktadır. Uçbirim ise ağa bağlanan herhangi bir sensör cihazıdır. Bu cihazlar genelde hasta üzerinde olup hastadan sürekli bilgi toplamaktadır. Ancak bunun yanı sıra bazı cihazlar sürekli olarak hasta üzerinde, yakın (6 metre, aynı oda) olup yine de vücut alan ağının bir parçası olabilir.

Ağ geçidi, yüksek bant genişliği sunan bir ağa direkt veya dolaylı olarak bağlıdır. Bu şekilde bir bağlantı (Wi-Fi, 3G/4G) genelde yüksek enerji tüketimi gerekmektedir ve boyutları açısından hasta üzerine yerleştirmeye uygun değildir.

1.2.3. Vücut Alan Ağının Uzaktan Hasta Takibindeki Yeri

Vücut alan ağı hasta üzerindeki sensörlerin kablosuz ve küçük olmasını sağlar. Kablosuz sensörler hastayı kablo karmaşasından kurtardığı ve kısmi de olsa hastanın hareket etmesine izin verdiği için hastayı rahatsız etmeden 24 saat takip etme imkanı sunar. Kablosuz sensörlerin hastaların sistemi kabul oranını arttırarak uzaktan hasta takip sisteminin yaygınlaşmasını tetiklemesi beklenmektedir.

1.3. IEEE 802.15.6 Protokolü

802.15.6 standardı IEEE organizasyonu tarafından insan vücudu yakınında veya vücut içerisine yerleştirilmiş sensörlerin kablosuz haberleşmesi için tasarlanmış bir haberleşme protokolüdür. [1]

802.15.6, var olan kablosuz haberleşme protokollerinin, tıbbi uygulamalarda ihtiyaç duyulan bant genişliği ve gecikme şartlarını yerine getirememelerinden, servis kalitesi kontrolü (QoS: Quality of Service) ve çakışmasız haberleşme sunamadıkları için ortaya çıkan boşluğu doldurmayı hedeflemektedir.

Bu standart haberleşmenin fiziksel katmanı (*PHY Layer*), ve ortam erişim kontrol katmanını (*MAC Layer*) tanımlar.

1.3.1. Fiziksel Katman (PHY Layer)

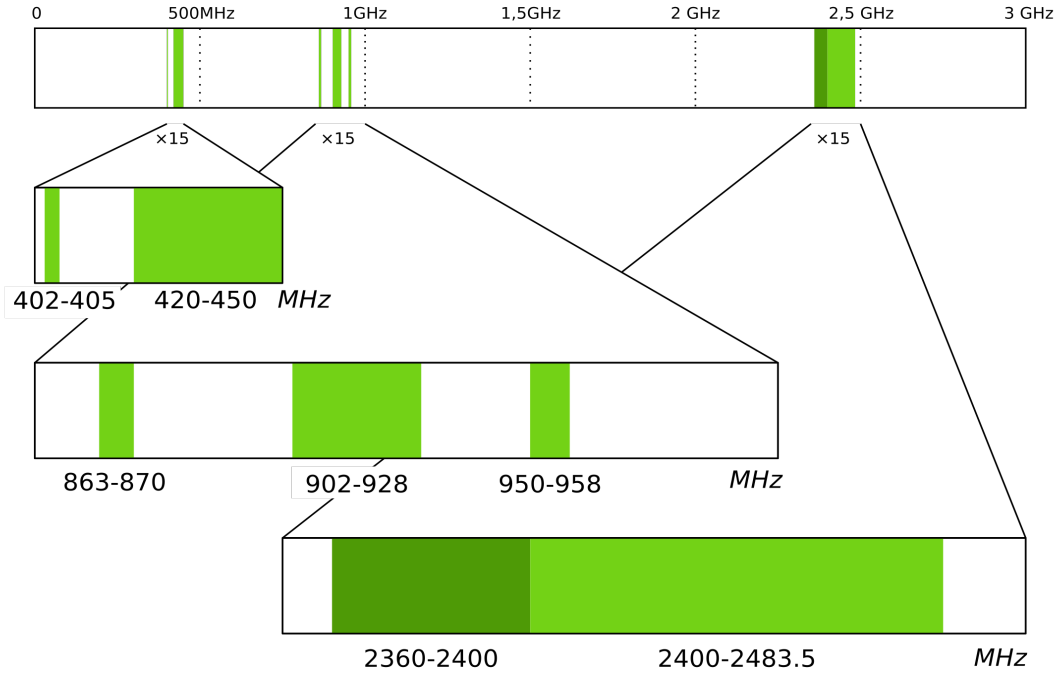
Fiziksel katman veri paketlerinin kablosuz kanal üzerinde nasıl ve hangi yöntemlerle iletildiğini tanımlar. Kablosuz kanal etrafımızı saran hava olduğu gibi, insan vücudu da haberleşme kanalı olarak kullanılabilir.

802.15.6 protokolü birbirinden bağımsız 3 farklı fiziksel katman standardı tanımlar.

1. Dar Bant (*Narrow Band*)
2. Çok Geniş Bant (*Ultra Wide Band*)

3. İnsan Vücudu Haberleşme Bandı (*Human Body Communication*)

Dar bant fiziksel katman, farklı frekans bantlarında ve birden fazla modülasyon çeşidi için tanımlanmıştır. Şekil 3'te kullanılabilecek frekans bantları gösterilmiştir.



Şekil 3. IEEE 802.15.6 dar bant fiziksel katmanı frekans bantları

Bu diyagramda koyu renkle gösterilen 2360-2400MHz bandı bu proje açısından özel önem arz etmektedir. Bu frekans bandı yakın zamanda Amerika Birleşik Devletleri FCC teşkilatı tarafından tıbbi uygulamalar için tahsis edilmiştir [2]. Bu bant tıp kuruluşları tarafından izne tabi olarak sadece bina içinde olmak üzere diğer kullanım alanlarıyla çakışma olmadan kullanılabilir. Bu bandın üst 10 MHz'lik kısmı (2390-2400MHz) herhangi bir izne tabi olmadan, hastahane dışındaki binalarda (evde bakım gibi), tıbbi uygulamalar için kullanılabilir. Diğer ülkelerin de benzer şekilde bu bandı tıbbi uygulamalara tahsis etmeleri beklenmektedir. Bu bandın, tıbbi uygulamalar için ayrılan diğer bantlardan daha ilgi çekici olmasının sebebi, frekansın nispeten yüksek olmasının daha küçük antenlerin tasarlanmasına, dolayısıyla daha küçük sensör cihazlarının geliştirilmesine imkan sunmasıdır.

Standardın tanımladığı modülasyon çeşitleri aşağıda listelenmiştir.

- π/2-DBPSK
- π/4-DQPSK

- π/8-D8PSK
- GMSK

Farklı çalışma bantları için farklı modülasyon parametreleri tanımlanmıştır. 2360-2400 MHz bandı için tanımlanan parametreler Tablo 1'de verilmiştir.

Tablo 1. Dar bant fiziksel katman parametreleri

Paket Kısmı	Modülasyon	Sembol Hızı	Dağıtma Oranı	Kod Oranı	Darbe Şekli	Hız (kbps)
PLCP Başlığı	π/2-DBPSK	600	4	19/31	SRRC	91.9
PSDU	π/2-DBPSK	600	4	51/63	SRRC	121.4
PSDU	π/2-DBPSK	600	2	51/63	SRRC	242.9
PSDU	π/2-DBPSK	600	1	51/63	SRRC	485.7
PSDU	π/4-DQPSK	600	1	51/63	SRRC	971.4

Hata düzeltme kodlaması olarak BCH kodlama tekniği tanımlanmıştır[1]. Bu kodlama tekniği bir önden hata düzeltme (FEC) tekniğidir. Bu teknikle paket kodlanarak gönderilir. Bu paketin boyutunun büyümesine sebep olur. Eğer paket transferi sırasında bazı bitler yanlış alınırsa, alıcı kodlama bilgisinden faydalanarak paketteki bozulmayı hatta bozulmanın yerini tespit edebilir. Böylece bir paketin bir noktaya kadar düzeltilmesi mümkündür. Bu şekilde küçük miktardaki bozukluklardan ötürü paketin tekrar gönderilmesine gerek kalmaz. İleri kodlama tekniğinin kullanımıyla iletişimin güvenilirliği artar, hatta kötü kanal koşullarında verimliliği de yükselir.

Çok geniş bant standardı, henüz geliştirilme aşamasında olup, 500 MHz genişliğinde bant kullanır. Bu bantlar 3,5 GHz-10 GHz aralığında tanımlanmıştır.

İnsan vücudu üzerinden iletim standardında, isminden anlaşılabileceği gibi insan vücudu kablosuz iletim ortamı olarak kullanılır. Haberleşme frekansı olarak 21MHz, bant genişliği olarak 5 MHz kullanılır.

1.3.2. Ortam Erişim Kontrol Katmanı (MAC Layer)

Vücut ağına bağlanan cihazların haberleşmesini zamanlama ve iletişim sırası açısından düzenleyen katman, ortam erişim kontrol katmanıdır. Bu katmanda genel ağ

yapısı, ağ kurulumunun nasıl yapılacağı, düğümlerin haberleşme yaparken uyacağı kurallar, paket yapısı, paketlerin zamanlaması gibi konular tanımlanır.

1.3.2.1. Ağ Öğeleri

1.3.2.1.1. Ağ Geçidi (Hub)

Ağ geçidi, ağı kuran ve uç birim düğümlerini yöneten cihazdır. Genel olarak ağ geçidi, uç birim düğümlerine göre daha fazla güç kapasitesine ve işlem gücüne sahiptir. Ağ geçidi güç kaynağı şebeke (direkt veya dolaylı olarak) olabileceği gibi, akıllı telefon gibi nispeten büyük batarya kapasitesine sahip bir cihaz da olabilir. Ağ geçidi cihazı ayrıca yüksek bant genişlikli bir ağa (internet gibi) bağlıdır. Bu ağ bağlantısı, ağ geçidinin bir kişisel bilgisayara bağlanması durumunda Wi-Fi veya Ethernet, akıllı telefona bağlanması durumunda 3G/4G üzerinden sağlanıyor olabilir.

1.3.2.1.2. Uç Birim Düğümü (End Device Node)

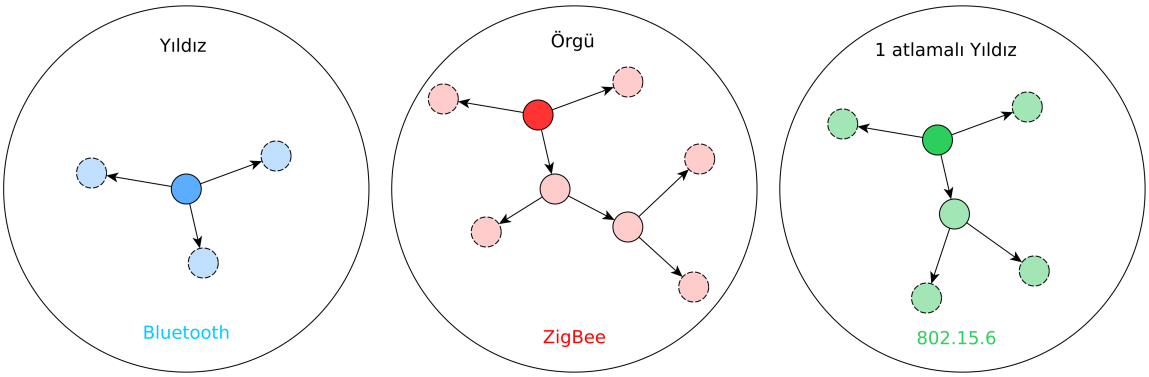
Uç birim cihazları, genelde insan vücudu üzerine yerleştirilen, küçük boyutlu olması gereken ve düşük güç tüketimi gerektiren, ölçüm cihazlarıdır. Bu cihazlar hasta üzerinden tıbbi bilgileri toplar ve vücut alan ağı üzerinden ağ geçidine gönderirler.

1.3.2.2. Ağ Yapısı

802.15.6 standardı yıldız topolojisinde bir ağ tanımlar. Bu yapıda ağı kuran ve kontrol eden bir cihaz vardır. Bu cihaza ağ geçidi (hub) denir. Diğer cihazlar direkt olarak, ağ geçidine bağlanırlar. Cihazların birbirleriyle iletişimi söz konusu değildir.

Ancak iletişimin güvenilirliğini arttırmak için tek atlamalı bir röle desteklenir. Ağ geçidine direkt olarak erişemeyen düğümler, röle özelliğini destekleyen bir düğüm üzerinden ağ geçidine, dolayısıyla ağa bağlanabilirler. Röle düğümlerinin varlığı, ağ geçidinin menzilini arttırabileceği gibi, araya giren nesneler veya insan uzuvları nedeniyle erişim sağlayamadığı düğümlere erişmesini sağlar.

Şekil 4'te yıldız topolojisi, örgü topolojisi ve röle destekli yıldız topolojisi gösterilmiştir. Örgü topolojisinde, diyagramda görüldüğü gibi düğümler birbirleri ile de haberleşme kurabilirler. Bluetooth teknolojisi yıldız, ZigBee teknolojisi ise örgü topolojisi kullanmaktadır.



Şekil 4. Kablosuz haberleşmede kullanılan ağ yapıları

1.3.2.3. Erişim Kontrolü

802.15.6 standardı birden fazla erişim tekniği tanımlar.

1. İşaretli bölünmüş zamanlı mod (beacon mode with superframes)
2. İşaretsiz bölünmüş zamanlı mod (non-beacon mode with superframes)
3. İşaretsiz ve zaman bölümlemesiz mod (non-beacon mode without superframes)

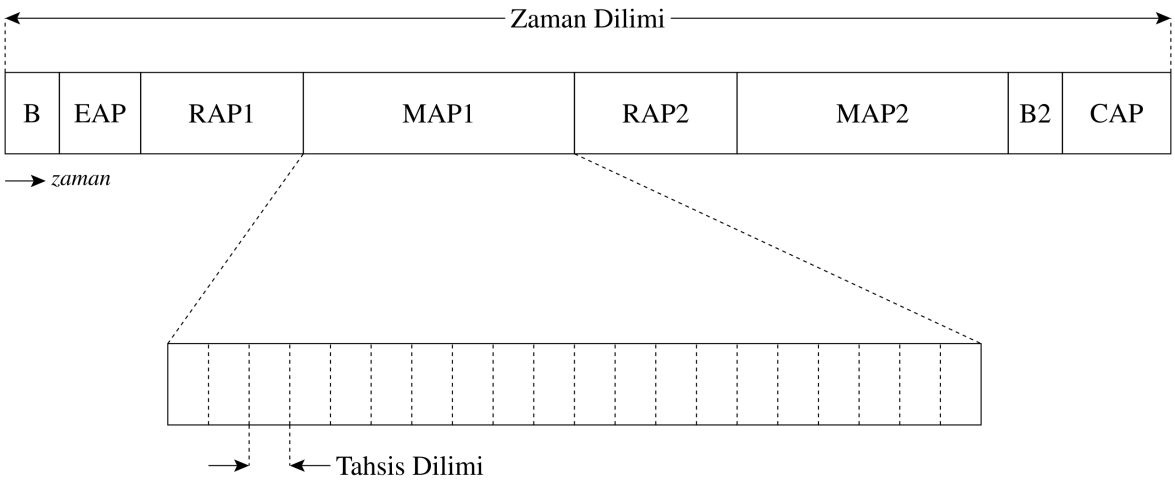
1. modda, çalışma zamanı ağ geçidi tarafından dilimlere bölünür (*superframe*). Her zaman diliminin başında bir işaret paketi gönderilir. Bu işaret paketi ağ bilgilerini ve zaman diliminin detaylarını taşır. Ağa bağlanmak isteyen düğümler bu işaret paketini aldıktan sonra, paketteki bilgileri kullanarak ağa bağlanma isteği gönderirler. Ayrıca işaret paketi zaman diliminin başlangıcını işaretlediği için, düğümler tarafından ağ geçidine senkronize olmak için kullanılır.

2. modda, çalışma zamanı yine dilimlere bölünür ancak işaret paketi gönderilmez. Bu modda ağ geçidinin yoklama paketleri göndererek düğümlerin ağa bağlanmasını sağlaması gerekir. Zaman senkronizasyonunun sağlanabilmesi için bu yoklama paketlerinin zaman bilgisi de içermesi gerekir.

3. modda, zaman dilimlere bölünmez ve iletişim ağ geçidi tarafından yoklama paketleri ile yürütülür.

1.3.2.3.1. Zaman Diliminin Yapısı

Zaman dilimi tahsis dilimlerine bölünmüştür. Bu dilimler farklı amaçlarla kullanılmak üzere gruplandırılmıştır. Bu gruplara erişim fazı denir. Bir zaman diliminin genel yapısı Şekil 5'te verilmiştir. Bütün fazlar tercihidir. Ağın tasarımına göre hangi fazların var olduğu ve boyutları değişebilir. Zaman diliminin yapısı, fazların uzunlukları, zaman diliminin başında gönderilen her işaret paketinin içerisinde bulunur. Ağ geçidine senkronize olmak isteyen düğümler, zaman diliminin yapısını işaret paketinden alır.



B: İşaret Paketi (Beacon Frame)
EAP: Acil Erişim Fazı (Emergency Access Phase)
RAP: Düzensiz Erişim Fazı (Random Access Phase)
MAP: Yönetilen Erişim Fazı (Managed Access Phase)
B2: Beacon 2 Paketi
CAP: Çekişmeli Erişim Fazı (Contention Access Phase)

Şekil 5. Zaman diliminin yapısı

a. Acil Erişim Fazı (Emergency Access Phase)

Bu faz işaret paketinden hemen sonra başlar. Yüksek öncelikli cihazların iletişimi ve acil öneme sahip paketlerin aktarılması için tahsis edilmiştir. Bu fazın amacı hayati önem taşıyan bilgi paketlerinin hızlı ve güvenilir bir şekilde iletilmesidir. Bir zaman diliminde iki

farklı acil erişim fazı bulunabilir. Acil iletişim fazında iletişim, rastgele erişim fazında olduğu gibi çekişmeli erişim teknikleri ile yürütülür.

b. Rastgele Erişim Fazı (Random Access Phase)

Bu fazda, düğümler, ağ geçidi ile olan iletişimlerini rastgele erişim tekniklerini kullanarak yürütürler. Ağa bağlı olan düğümlerin yanı sıra, ağa bağlanmak isteyen düğümler de bu fazda iletişim kurabilirler.

Ağ geçidi tarafından herhangi bir tahsis işlemi yapılmadığı için, iki farklı düğüm tarafından gönderilen paketlerin çakışma olasılığı vardır. Aynı anda haberleşme yapmak isteyen iki düğüm arasındaki çakışmayı çözebilmek için çekişmeli erişim teknikleri kullanılır. Zaman diliminde iki farklı rastgele erişim fazı bulunabilir. Bunlar RAP1 ve RAP2 olarak isimlendirilir.

c. Yönetilen Erişim Fazı (Managed Access Phase)

Bu fazda, tahsis dilimleri, ağ geçidi tarafından düğümlere ihtiyaçlarına ve öncelik sıralarına göre atanır. Bir düğüm kendisine atanan zaman aralığında ağa bağlı diğer düğümlerle çakışma riski olmadan iletişim kurabilir.

Tahsis dilimleri 3 farklı işlev için atanabilir.

1. İndirme (down-link)
2. Yükleme (up-link)
3. İki Yönlü (bi-link)

İndirme için atanan dilimlerde, ağ geçidinden uçbirim düğümüne giden paketler aktarılır. Yükleme dilimlerinde, düğümden ağ geçidine giden paketler aktarılır. İndirme dilimlerinde düğüm, yükleme dilimlerinde ağ geçidi, sürekli dinlemede olmalıdır. İki yönlü iletişim için atanan dilimlerde, her iki yönde de paket aktarımı yapılabilir. Çakışma olmaması için, bu şekilde tahsis edilen dilimlerde paket aktarımı, ağ geçidinin kontrolü altında gerçekleştirilir. İki farklı yönetilen erişim fazı bulunabilir, bunlar MAP1 ve MAP2 olarak isimlendirilir.

d. B2 Paketi ve Çekişmeli Erişim Fazı (Contention Access Phase)

Ağ geçidi zaman diliminin sonundan bir kısmı çekişmeli erişim için tahsis edebilir. B2 paketi bu dilimin başlangıcını işaretler ve uzunluk bilgisini taşır. B2 paketi, grup alındı özelliği, pasif zaman dilimi gibi diğer bazı özelliklerin de gerçekleştirilmesi için kullanılır.

1.3.2.3.2. Çekişmeli Erişim Teknikleri

802.15.6 iki farklı çekişmeli erişim tekniği tanımlar.

1. CSMA/CA : Carrier Sense Multiple Access/Collision Avoidance
2. Slotted Aloha

Bunlardan Slotted Aloha, geniş bant radyosu için, CSMA/CA dar bant radyosu için tanımlanmıştır.

CSMA/CA erişim tekniği, iki esasa dayalıdır. Gönderme işlemi yapmadan önce kablosuz kanal dinlenerek, o an başka bir düğümün iletişim yapıp yapmadığı tespit edilir. Bunun için taşıyıcı sinyalinin tespit edilmesi yeterlidir. Dolayısıyla bir paket alımı için gerekli olan de-modülasyon işlemleri yapılmaz, sadece enerji ölçüm tekniği kullanılır. Diğer esas şöyledir; eğer bir taşıyıcı işareti tespit edilirse veya iletişim bilinmeyen bir sebepten ötürü başarısız olursa (alındı paketinin alınamaması), düğüm gönüllü olarak belli bir süre kanaldan geri çekilir (*backoff*). Bu geri çekilmenin ne kadar olacağı rastgele olarak belirlenir.

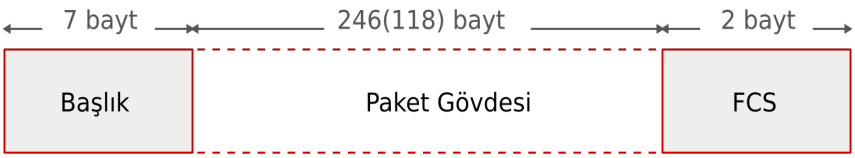
Bu teknikle iletişim gerçekleştirilirken, ilgili zaman aralığı, küçük zaman dilimlerine bölünür. Bu zaman diliminin süresi dar bant fiziksel katman için 125 μs olarak tanımlanmıştır. Düğüm, CSMA/CA için tanımlanan kurallar çerçevesinde kendi kendisine bir tahsis yapmaya çalışır. 802.15.6 standardında detaylı bir şekilde açıklanan kurallara burada sadece özetle değinilecektir. Tahsis yapmak için, bir çekişme penceresi uzunluğu seçilir (CW: Contention Window). Bu uzunluk, küçük zaman dilimi adedi cinsinden 1-64 arası bir sayıdır. Ancak hangi aralıklardaki sayıların seçilebileceği düğümün öncelik numarasına göre ve rastgele olarak seçilen bir sayıyla belirlenir. Yüksek öncelikli bir düğümün çekişme penceresi daha küçük, dolayısıyla iletişim kurma ihtimali daha yüksek olacaktır. Kendisine çekişme penceresi seçen düğüm, geri çekilme sayacını bu sayıya ayarlar. Her çekişme zaman diliminde kablosuz kanaldaki enerji seviyesini ölçer. Eğer bir iletişim tespit edilmezse, sayacı bir azaltır. Eğer iletişimin varlığı tespit edilirse, sayaç azaltılmaz. Sayacın sıfıra ulaşmasıyla, düğüm kendisine bir iletişim hakkı tahsis etmiş olur. Eğer iletişim başarısız olursa (çakışma gerçekleşirse), düğüm tahsis işlemini yeniden tekrarlar. Ancak her tekrarda kendisine daha uzun bir çekişme penceresi seçer. Çekişme penceresinin maksimum uzunluğuna ulaşıldığında, daha büyük bir pencere seçilmez.

Slotted Aloha erişim tekniği, bir çok yönüyle CSMA/CA tekniğine benzer. Ancak bu teknikte kanal ölçümü yapılmaz. Düşük trafikli kanallarda enerji tüketimi açısından

CSMA/CA'dan daha verimli olabilir [3]. Slotted Aloha sadece geniş bant radyosu için kullanılır.

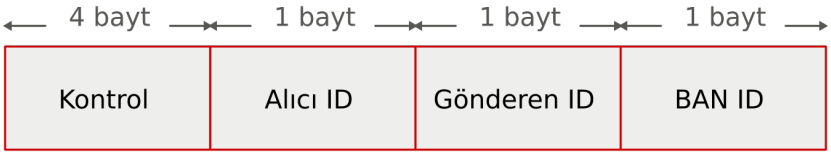
1.3.2.4. Paket Yapısı

Şekil 6'da erişim katmanı paket yapısı verilmiştir. Paketin ilk 7 baytı başlık bilgisidir. Başlık her pakette bulunur, ancak işlevi paket türüne ve düğüme göre değişiklik gösterir. Başlığın ardından paket gövdesi gelir. Burası asıl bilginin taşındığı alandır. Paket gövdesinin uzunluğu fiziksel katmana göre değişir. Paketin tam uzunluğu fiziksel katman tarafından tanımlanır.



Şekil 6. 802.15.6 erişim kontrol katmanı paketi genel yapısı

802.15.6 dar bant standardı için tam paket uzunluğu 255 bayt olarak tanımlandığı için, paket gövdesinin maksimum uzunluğu 246 (*255-7-2*) bayttır. Ancak bu çalışmada 802.15.4 alıcı vericisi kullanıldığı, ve bu entegrenin maksimum paket boyutu 127 bayt olduğu için, maksimum paket gövdesi 118 bayttır. Paket gövdesinin ardından FCS (*Frame Check Sequence*) gelir. Bu paketin bozulup bozulmadığının kontrol edilmesini sağlayan 2 baytlık bir CRC değeridir.



Şekil 7. Paket başlığının kısımları

Başlık kısmının detayları şekil 7'de verilmiştir. Başlık kısmında, kontrol, paketin alıcısı düğümün ID'si, gönderen düğümün ID'si, bağlı olunan vücut alan ağı'nın ID'si bulunur. Kontrol bilgisi, paket çeşidine ve işlevine göre çok farklı bilgiler taşıyabilir.

1.3.2.5. Paket Çeşitleri

Paket çeşitleri 3 gruba ayrılmıştır.

- Yönetim (*Management*)
- Kontrol (*Control*)
- Veri (*Data*)

Tablo 2. Yönetim türü paketler

Paket	**İşlevi**
İşaret (*Beacon*)	Ağ geçidi tarafından zaman diliminin başlangıcında gönderilir
Güvenlik Birleşmesi (*Security Association*)	Şifreli bağlantı kurma
Güvenlik Çözmesi (*Security Disassociation*)	Şifreli bağlantıyı bozma
PTK	Şifreli bağlantı için kullanılır
GTK	Şifreli bağlantı için kullanılır
Bağlantı İsteği (*Connection Request*)	Bağlantı kurma
Bağlantı Atama (*Connection Assignment*)	Bağlantı kabul etme/reddetme
Bağlantı Kesme (*Disconnection*)	Bağlantı kesme

Yönetim paketleri Tablo 2'de, kontrol paketleri Tablo 3'te verilmiştir. Veri paketleri düğümün öncelik numarasına göre sınıflandırılır. Yönetim paketleri, ağ kurulumu ve kesilmesi sırasında kullanılan iletişim paketleridir. Kontrol paketleri ise veri paketlerinin iletimi sırasında gönderilen, ortam erişimini kontrol etmek ve düzenlemek için kullanılan paketlerdir.

Tablo 3. Kontrol türü paketler

Paket	**İşlevi**
Anında Alındı (*I-Ack*)	Bir paket alındıktan hemen sonra gönderilir
Blok Alındı (*B-Ack*)	Birden fazla paketin alındığını bildirir
Anında Alındı + Yoklama (*I-Ack+Poll*)	Paketin alındığını ve yeni paket gönderilebileceğini belirtir
Blok Alındı + Yoklama (*B-Ack+Poll*)	Birden fazla paketin alındığını ve yeni paket gönderilebileceğini bildirir
Yoklama (*Poll*)	Paket gönderilebileceğini belirtir
Zamanlı-Yoklama (*T-Poll*)	Zaman dilimsiz erişim modunda yoklama paketi olarak kullanılır
Uyandırma (*Wakeup*)	Düğümü uyandırmak için gönderilir

1.4. Literatür

Kablosuz vücut alan ağlarında kullanılmaya uygun bazı erişim kontrol protokollerinin incelendiği bir çalışma kaynak [4]'te verilmiştir. Zaman paylaşımlı çoklu erişim protokollerinin enerji verimliliği açısından üstünlüğü belirtilmiş ancak esneklik ve senkronizasyon problemleri olduğuna da değinilmiştir.

IEEE 802.15.4 protokolü kullanılarak uzaktan hasta takip sisteminde kullanılmak üzere bir kablosuz EKG cihazı kaynak [5]'te gerçekleştirilmiştir. Bu çalışmada kablosuz haberleşme modülü olarak XBee ZigBee haberleşme modülü kullanılmıştır. Çalışmada kendine özgün bir FEC (İleri Hata Kodlama) tekniği geliştirilmiştir. Yapılan çalışma 802.15.4 protokolünün, 802.11 (Wi-Fi) bağlantıları tarafından doyuma ulaşmış kanallarda yeterli performansa ulaşamadığını göstermiştir. ZigBee protokolünün kullanıldığı bir vücut alan ağı uygulaması kaynak [6]'da gerçekleştirilmiştir. Aynı ortamda bulunan 802.11 ağının, paket kayıplarını ciddi oranda arttırdığı ve bağlantı kalitesini kötü yönde etkilediğini gösteren bulgular sunulmuştur. 802.15.4'ü kullanan bir diğer kablosuz EKG uygulaması kaynak [7]'de gerçekleştirilmiştir. Çalışmada cihazın uzaktan kontrolü için tasarlanan bir paket protokolü verilmiştir.

Bluetooth protokolünü kullanan ve bir akıllı telefonun ağ geçidi olarak görev yaptığı bir kablosuz hasta takip sensörü ise kaynak [8]'de gerçekleştirilmiştir. Bluetooth'un kullanıldığı benzer bir çalışma da kaynak [9]'da yapılmıştır. Bu çalışmada cep telefonu ağ geçidi olarak kullanıldığı gibi, aynı zamanda hasta'dan alınan EKG işaretlerinin analizini de gerçekleştirmektedir. Yine Bluetooth'un kullanıldığı bir çalışma kaynak [10]'da verilmiştir. Bu çalışmada ağ geçidi olarak bir COM (Computer On Module) bilgisayar modülü kullanılmıştır. Bluetooth protokolü EKG verisi aktarımı için özel bir profil tanımlamadığı için *SerialPort* profili üzerinden aktarım yapan özel bir protokol tasarlanmıştır. Bluetooth'un hasta üzerinden toplanan PPG verisinin aktarımı için kullanıldığı bir çalışma kaynak [11]'de verilmiştir. Bu çalışmada bir kişisel bilgisayar ağ geçidi olarak kullanılmıştır. Tıbbi cihazlar için düşük güç tüketimli bir kablosuz aktarım cihazının, bir Bluetooth modülü kullanılarak geliştirilmesi kaynak [12]'de verilmiştir. Bluetooth'un, sensör cihazlarından ağ geçidi cihazında toplanan verileri yakındaki bir akıllı telefona göndermek için kullanıldığı bir çalışma kaynak [13]'te verilmiştir.

Kablosuz haberleşmede paket kayıpları kaçınılmazdır. Tıbbi uygulamalarda veri kaybı kabul edilemeyeceğinden dolayı, paketlerin yeniden gönderilmesi ilk yapılacak işlemdir. Bu yeniden gönderme işleminin verimli bir şekilde gerçekleştirilmesini araştıran bir çalışma kaynak [14]'te verilmiştir. Bu çalışmada ayrıca Bluetooth'un servis kalitesi kontrol (QoS) ve erişim kontrol mekanizmalarının verimli iletişim için yeterli olmadığına değinilmiştir.

IEEE 802.15.6 protokolü ile ilgili bir çok simülasyon odaklı çalışma yayınlanmıştır. Bir Ağ geçidi cihazına bağlı iki uç birim cihazının bulunduğu, bütün zaman dilimini rastgele erişim fazı olarak tahsis eden, 802.15.6 tabanlı bir erişim kontrol katmanı simülasyon çalışması kaynak [15]'te verilmiştir. Çekişmeli erişim tekniğinin kullanıldığı bu çalışmada, idealde daha verimli olması gereken daha büyük paket boyutlarının sistemin bant genişliğini düşürdüğü gözlemlenmiştir. Bu da uzun paketlerin çakışma ihtimalinin daha yüksek olmasından kaynaklanmaktadır. 802.15.6 farklı erişim tekniklerine imkan sunan esnek bir protokoldür. Hangi tekniğin hangi şartlar altında daha verimli olacağını inceleyen, yoklamalı, yönetimli ve çekişmeli erişim tekniklerinin karşılaştıran bir çalışma kaynak [16]'da verilmiştir. Enerji verimliliğini arttırmayı hedefleyen 802.15.6 tabanlı bir erişim katmanının tasarımı ve simülasyonu kaynak [17]'de verilmiştir. Söz konusu çalışmada, zaman diliminin bir kısmı yoklama paketleri ile yönetilmektedir. Bu sürenin uzunluğu ağın ihtiyaçlarına göre dinamik olarak belirlenmektedir. Vücut alan ağları

arasındaki çakışmayı inceleyen ve çakışma çözümleme için özel bir protokol sunan bir çalışma kaynak [18]'de verilmiştir. Yoklamalı erişim ve çekişmeli erişim tekniklerinin karşılaştırıldığı analitik ve simülasyon odaklı bir çalışma kaynak [19]'da gerçekleştirilmiştir. Yoklamalı erişimin enerji verimliliği açısından üstünlüğü belirtilmiştir.

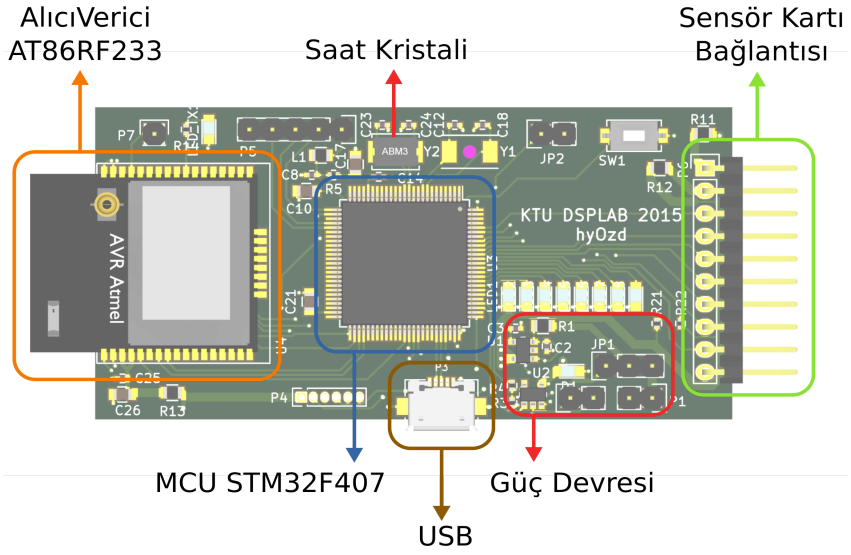
IEEE 802.15.6 erişim tekniklerinin 802.15.4 standardı ile karşılaştırıldığı bir simülasyon çalışması kaynak [20]'de verilmiştir. 802.15.6'nın daha düşük güç tüketimi olduğu belirtilmiştir. Bu iki protokolü zaman dilimi yapıları yönünden karşılaştıran bir diğer çalışma kaynak [21]'de verilmiştir. 802.15.4'ün daha yüksek bant genişliği sunduğu ancak 802.15.6'nın daha düşük gecikmeye sahip olduğu belirtilmiştir. Ayrıca 802.15.6'nın düğüm önceliği yönetim özelliklerinin, kritik bilgi taşıyan paketlerin daha erken iletilmesini sağlayacağı belirtilmiştir.

IEEE 802.15.6 dar bant fiziksel katmanını gerçekleyen, enerji verimliliği yüksek bir fiziksel alıcı verici yongası kaynak [22]'de gerçekleştirilmiştir. Tasarlanan yonga sadece π/4-DQPSK modülasyonunu desteklemektedir. π/2-DQPSK modülasyonunu da destekleyen benzer bir çalışma kaynak [23]'te verilmiştir. Dar bant fiziksel katmanın FPGA üzerinde gerçeklendiği bir çalışma kaynak [24]'te verilmiştir.

2. YAPILAN ÇALIŞMALAR

2.1. Donanım

Kablosuz ağın gerçekleştirilmesi için bir haberleşme modülü donanımı gerçekleştirilmiştir. Bu modül üzerinde bir mikrodenetleyici ve alıcı verici entegresi bulunmaktadır. Modül hem ağ geçidi cihazı hem uçbirim cihazı olarak kullanılmıştır. Şekil 8'de tasarlanan modül görülmektedir.



Şekil 8. Haberleşme modülü

Haberleşme modülünü ağ geçidi olarak kullanmak için, ağ geçidi yazılımını yüklemek ve USB ile bir kişisel bilgisayara bağlamak yeterlidir.

Uçbirim cihazı ise haberleşme modülüne sensör devrelerinin üzerinde bulunduğu bir ek kart bağlanarak gerçekleştirilmektedir. Bu konfigürasyonda haberleşme modülü, hem sensör komponentlerinin kontrolünü hem kablosuz haberleşmeyi sağlar. Modül enerjisini bir bataryadan sağlar. Aynı zamanda USB bağlantısı üzerinden enerji sağlamak da mümkündür. Cihazın pili de bu bağlantı üzerinden şarj edilir.

Haberleşme modülünün temel parçaları şöyledir:

1. Mikrodenetleyici

2. Alıcı verici entegresi
3. USB bağlantısı
4. Besleme voltaj düzenleyici
5. Şarj kontrol entegresi (uçbirim için)
6. Sensör bağlantı konnektörü (uçbirim için)
7. Düğme ve LED'ler
8. Programlama konnektörü

2.1.1. Mikrodenetleyici

Haberleşme modülünün kontrolü ve haberleşme protokolünün gerçekleştirilmesi bu işlemci üzerinde gerçekleştirilmiştir. Mikrodenetleyici olarak STMicroelectronics firmasının STM32F407 mikroişlemcisi kullanılmıştır. Bu işlemcinin temel özellikleri şöyle listenelebilir [25]:

- ARM Cortex-M4F çekirdek
- 168 MHz azami saat hızı
- 1 MB Flash (program) belleği
- 192 KB RAM belleği
- 1.8V-3.3V besleme
- Dahili USB
- Zamanlayıcı (Timer)
- 12-bit ADC
- SPI, I2C, UART
- Rastgele Sayı Üreteci
- 96 bit özgün ID

Listede mikrodenetleyicinin bu çalışmada kullanılan özelliklerine değinilmiştir. Bu mikrodenetleyicinin seçilmesindeki ana sebep yüksek hızı ve bellek kapasitesi olmuştur. Özellikle ağ geçidi modülünün birden fazla cihazla hızlı ve düşük gecikmeli bir şekilde haberleşme kurabilmesi için yüksek hızlı bir işlemci gerekeceği öngörülmüştür.

2.1.2. Alıcı Verici Modülü

Halihazırda 802.15.6 dar bant fiziksel katmanını gerçekleyen bir alıcı verici entegresi ticari olarak mevcut değildir. Bu sebeple 802.15.6 erişim katmanının, kablosuz vücut alan ağı prensiplerine yakın bir şekilde gerçekleştirilebileceği bir alıcı verici entegresi arayışına gidilmiştir. Kablosuz alıcı verici olarak Atmel[1] firmasının AT86RF233 entegresini içeren ATZB-RF-233-1-C[26] kablosuz haberleşme modülü seçilmiştir. Bu modül aslen 802.15.4 ve ZigBee haberleşme için geliştirilmiş olsa da, araştırma amaçlı kullanıma ve özel protokollerin gerçekleştirilmesine de imkan sunmaktadır.

Bu modülün seçilmesindeki en önemli etken MBAN için ayrılan 2360-2400 MHz bandında haberleşme kurabilmesidir.

AT86RF233 modülünün temel özellikleri şöyle listelenebilir [27]:

- 802.15.4 modülasyon: OO-QPSK
- 1.8V-3.3V besleme voltajı
- -17 dbm ve +4 dbm arası ayarlanabilir çıkış gücü
- 2.36 GHz - 2.485 GHz çalışma frekansı
- -101 dbm giriş hassasiyeti
- 2 mbps veri iletim hızı
- SPI üzerinden kontrol
- Paket fifosu
- RSSI (giriş sinyal seviyesi ölçümü)

Modül ZigBee gibi 802.15.4 tabanlı haberleşme protokollerinin gerçekleştirilmesi için yardımcı, paket filtreleme, adres kontrol etme gibi özellikler sunmaktadır. Ancak entegrenin bu özellikleri kullanılmamıştır.

2.1.3. Besleme

Haberleşme modülü'nün çalışma gerilimi 3,3V'tur. Bu voltajın ana kaynağı, haberleşme modülü ağ geçidi olarak çalıştırıldığında 5 Volt USB beslemesidir. Kablosuz

---

1 Atmel firması 2016 yılında Microchip firması tarafından satın alınmıştır.

uçbirim cihazı olarak çalıştırıldığında ise besleme Lityum bataryadan sağlanmaktadır. Batarya seviyesi 4.2V-3.5V arasında değişmektedir. Her iki kullanım konfigürasyonunda da voltaj seviyesinin 3,3V'a düşürülmesi gerekmektedir. Bunun için ST firmasının LD3985 voltaj düzenleyici entegresi kullanılmıştır.

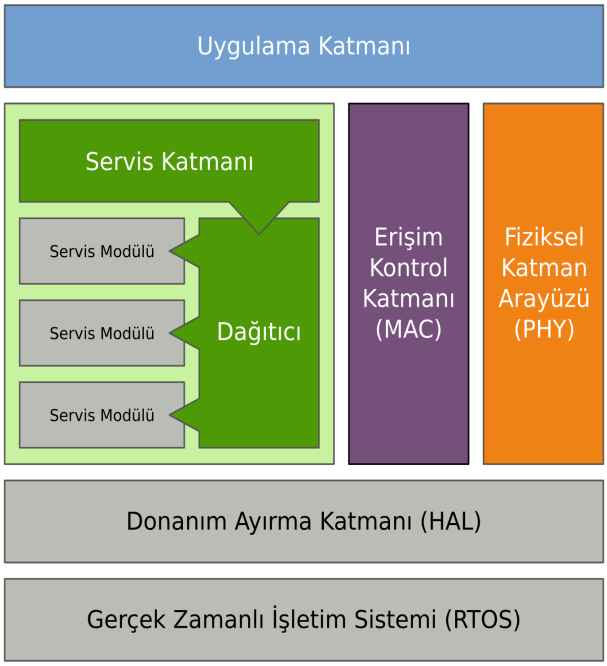
2.1.4. Batarya Sarj Devresi

Düğüm devresinin bataryasının sarj edilmesi için modül üzerinde bir de sarj devresi bulunmaktadır. Bunun için Microchip firmasının MCP73831 entegresi kullanılmıştır. Bu entegre 5 Volt USB beslemesinin Lityum batarya için düzenler ve batarya sarj olana kadar beslemeyi sağlar. Batarya voltajını periyodik olarak kontrol ederek, pilin sarj olduğuna karar verdiğinde sarj işlemini keser.

2.2. Yazılım

2.2.1. Genel Yapı

İletişim sisteminin yazılımı 3 ana katmandan oluşmaktadır. Fiziksel katman arayüzü, alıcı verici entegresinin kontrolünü sağlayan arayüzü sunmaktadır. Erişim kontrol katmanı, IEEE 802.15.6 uyumlu erişim kontrol protokolünü gerçekleştirir. Servis katmanı, cihaz üzerindeki sensörlerin uzaktan yönetilmesi ve bunlara erişim için modüler bir yapı olarak gerçekleştirilmiştir. Uygulama katmanı bütün bu katmanları kontrol eder ve aralarındaki senkronizasyonu sağlar. Şekil 9’da bu yapıları genel olarak gösteren bir çizim verilmiştir.



Şekil 9. Yazılımın genel yapısı

2.2.2. Geliştirme Platformu

STM32F4 mikroişlemcisinin programlanması için C programlama dili kullanılmıştır. ARM Cortex-M ailesinden bu mikrodenetleyicinin, dahili donanımlar açısından zenginliği ve bir çok kullanım şeklinin mümkün olması sebebiyle herhangi bir kütüphane kullanılmadan programlanması son derece güçtür. ARM Cortex-M işlemcilerin programlanması için ARM tarafından sunulan CMSIS[28] veya bizzat ST Micro tarafından STM32CubeF4[29] yazılımının bir parçası olarak sunulan HAL (*Hardware Abstraction Layer*) kullanılabilir. Ancak planlanan projenin karmaşıklığından ötürü bir gerçek zamanlı işletim sistemi kullanılması ön görülmüştür. İşletim sistemi olarak ChibiOs RTOS işletim sistemi kullanılmasına karar verilmiştir. Bu işletim sistemi aynı zamanda ST firmasının mikrodenetleyicileriyle çok iyi entegre olan bir HAL kütüphanesi sunmaktadır.

2.2.3. ChibiOs İşletim Sisteminin Temel Özellikleri

ChibiOs işletim sistemi birden fazla kısımdan oluşmaktadır [30].

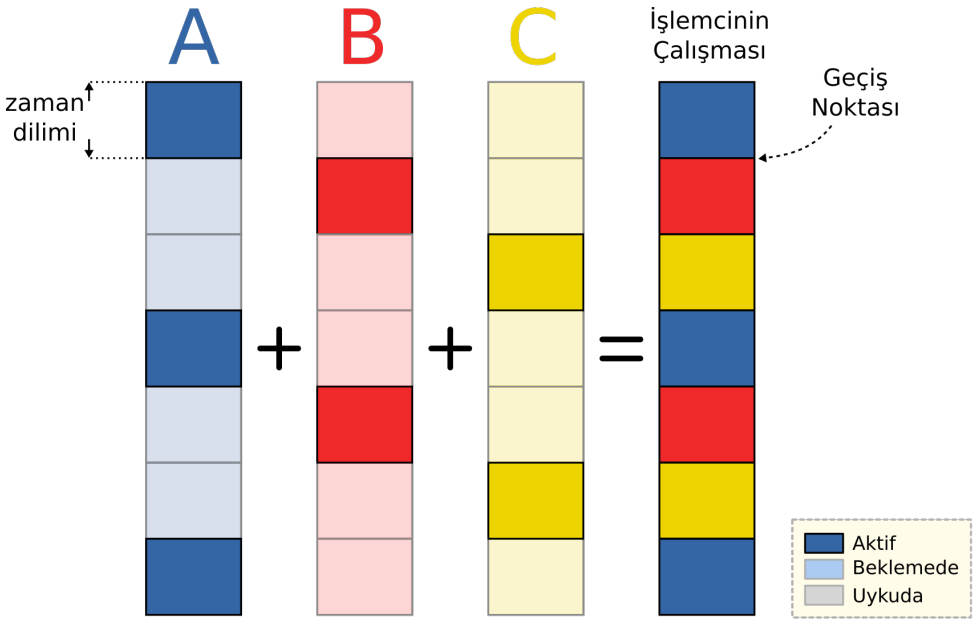
- *RT* : gerçek zamanlı işletim sistemi çekirdeği
- *HAL* : donanım ayırma katmanı (işlemci ailesine özel)

- *nil* : RT'ye göre daha hafif bir alternatif çekirdek

2.2.3.1. RT Çekirdek

İşletim sistemi çekirdeği, gerçek zamanlı bir işletim sisteminin temel özelliklerini sağlar. İşletim sistemi çekirdeği tarafından sunulan en önemli özellik aynı anda birden fazla programın çalışmasını sağlayan izlek (*thread*) çalıştırma özelliğidir.

Gerçekte tek bir işlemci aynı anda iki programı çalıştıramaz. Bu ancak çok çekirdekli işlemcilerde mümkündür. Ancak sırayla bir programdan diğer programa geçiş yapılarak sanki iki program da aynı anda çalışıyormuş gibi bir davranış sağlanabilir. Bu geçişi periyodik olarak sağlamak işletim sisteminin görevidir. Şekil 10'da bu prensip basitçe gösterilmiştir.

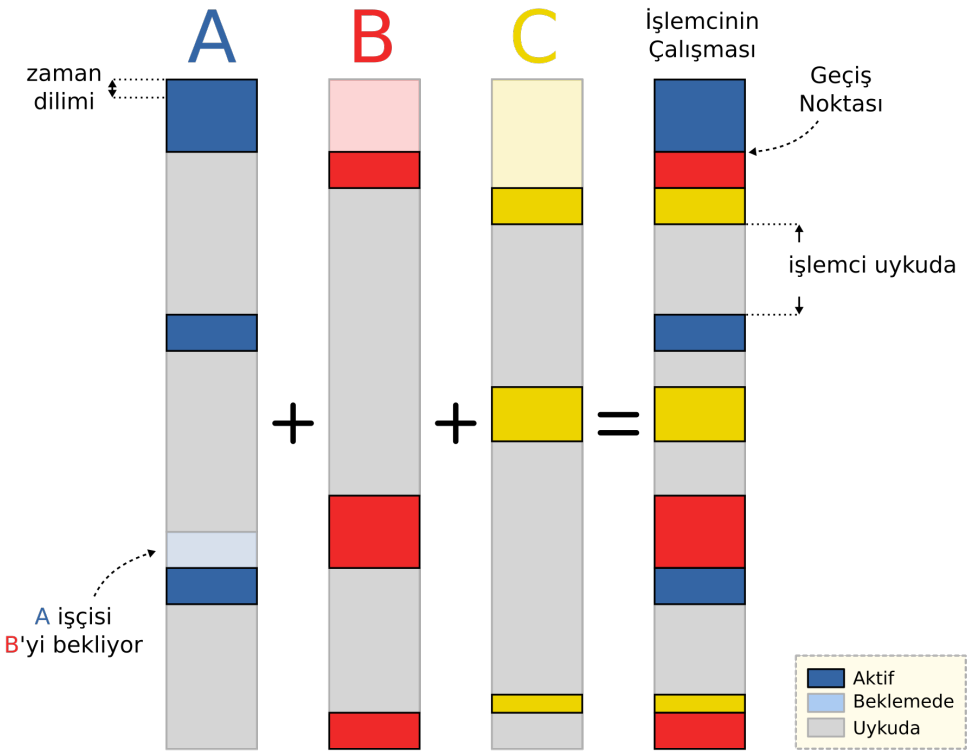


Şekil 10. Birden fazla izleği çalıştırılmasının basitleştirilmiş gösterimi

Diyagramı kısaca özetlemek gerekirse. Bu işlemci 3 (A, B, C) farklı programı aynı anda çalıştırmaktadır. Bunu gerçekleştirebilmek için zamanı dilimlere böler. Her dilimin sonunda bir kesme (*interrupt*) oluşturulur, bu kesme ile işlemci diğer programı çalıştırmaya geçer. Eğer zaman dilimi yeterince kısa tutulursa (<10ms) dışarıdaki bir gözlemciye göre bu üç program da aynı anda çalışıyormuş görüntüsü oluşur. Tabi aslında bir program

çalışırken diğer programlar çalışmaz, yani beklemededirler. Örneğin diyagramın ilk satırına bakıldığında A programı aktiftir dolayısıyla B ve C programları beklemededir. İkinci satırda B programı aktiftir, A ve C programları beklemededir. Görüldüğü gibi işlemci programları sırayla aktifleştirmektedir. Bu tekniğe sırayla çalıştırma (round-robin) denir. Pratikte işletim sistemi, programların ihtiyaçlarına daha hızlı yanıt verebilmek için daha gelişmiş bir geçiş sistemi kullanır.

Bu teknik ile pratik uygulamaların nasıl çalıştığının anlayabilmek için, gömülü sistemlerde işlemcinin çoğu zaman beklemede olduğunu (uyuduğunu) hatırlatmak gerekir. Dolayısıyla Şekil 11'de verilen diyagram biraz daha karışık olsa da, daha gerçekçi bir çalışma senaryosu sunmaktadır.

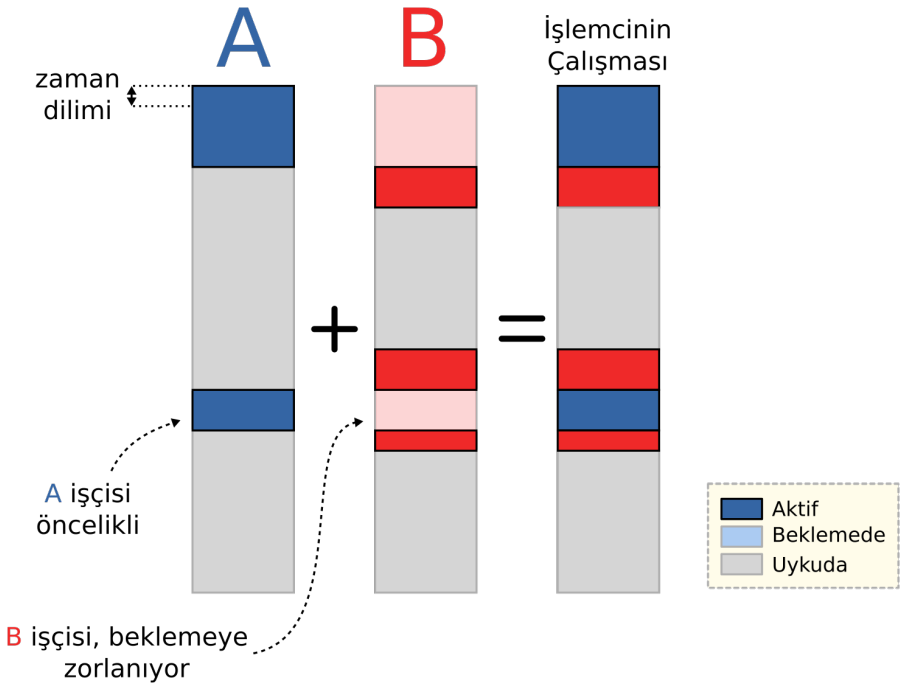


Şekil 11. Birden fazla izleğin düzensiz bir sıra ile çalışması

Bu diyagramda görüldüğü gibi izlekler sırayla çalıştırılmak zorunda değildir. İzlekler aktif olmaları gerektiğinde işletim sistemi tarafından uyandırılarak çalıştırılır. Ancak diyagramda örnek olarak bir çakışma senaryosu gösterilmiştir. A programı, B programı aktif iken çalışmak istediğinde işletim sistemi tarafından beklemeye alınmıştır. B programı çalışmasını bitirdiğinde, A programının çalışması devam ettirilmiştir. Bu senaryo öncelik

yönetiminin gerekliliğini göstermektedir. Zira işlemci zamanlama açısından kritik (acil) olmayan bir iş programını çalıştırırken, daha acil olan bir program beklemede kalabilir.

Şekil 12'de iş parçacıklarının önceliklerini bilen ve buna göre geçiş yapan bir çekirdeğin çalışması gösterilmiştir. Bu örnekte A işçisi, B işçisine göre daha yüksek bir önceliğe sahiptir. Bu bilgi işletim sistemine verilmiştir. Görüldüğü gibi B işçisi aktif olmasına rağmen, A işçisi çalışmak istediğinde, düşük öncelikli B işçisi beklemeye zorlanarak A işçisi aktif edilmiştir. A işçisi işini bitirdikten sonra, B işçisi aktif edilerek çalışması devam ettirilmiştir. Bir işletim sistemi kullanıldığında, farklı işlerden sorumlu programları bir birinden bağımsız olarak yazmak ve çalıştırmak mümkündür.

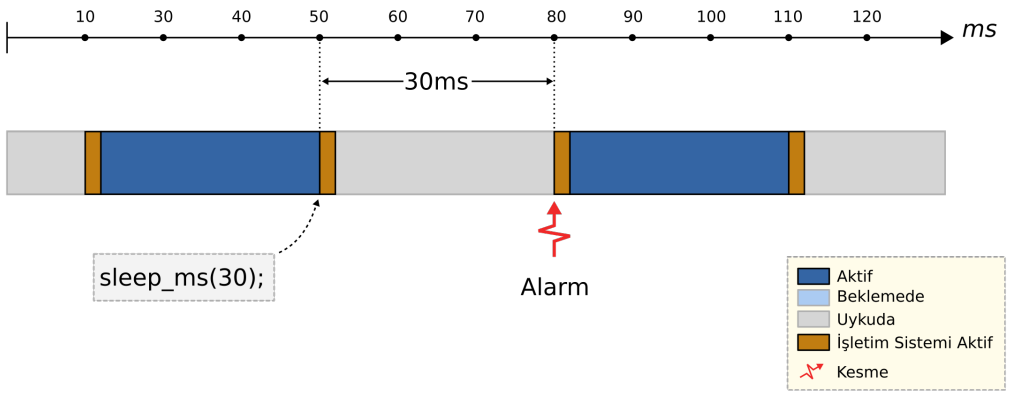


Şekil 12. Yüksek öncelikli bir izleğin, diğer izleği bekletmesi

2.2.3.1.1. Zaman Dilimsiz İşletim Sistemi

ChibiOs dilimsiz (*tickless*) çalışma modunu desteklemektedir. Bu teknikte işletim sisteminin çalışan programı periyodik olarak durdurması söz konusu değildir. Bunun yerine işletim sistemi, durum geçişlerini kesme olaylarında sağlar. Örneğin bir izleği belli bir zamanda uyandırmak için bir zamanlayıcı alarmı kurar. Belirlenen an geldiğinde işlemci zamanlayıcı kesmesi ile uyanır ve işletim sistemi kontrolü ele alır. İşletim sistemi, beklemekte olan izleklerin listesine bakarak hangisinin çalışması gerektiğine karar verir ve bu izleğe durum geçişi yapar.

Şekil 13'te kurmalı operasyonun bir örneği gösterilmiştir. İzlek programında *chThdSleepMilliseconds(30)* kodunun çalıştırılmasıyla birlikte izlek durdurulur. Bu fonksiyon işletim sisteminin bir parçasıdır. İşletim sistemi bu noktada kontrolü ele alır. İşletim sistemi şimdiki zaman + bekleme zamanı'nı hesaplayarak, bir sonraki uyanma noktasını belirler. Bu noktada uyanmak üzere zamanlayıcıyı ayarlar - alarm kurar. Uyanma anı geldiğinde, zamanlayıcı işlemciyi bir kesme ile uyandırır. Her kesmede ilk olarak işletim sistemi çalışır ve işletim sistemi uyumakta olan izleklerin listesini gözden geçirir. Bu listede uyanma vakti gelmiş olan izleği aktif eder ve izleğe durum geçişiş yaparak programın çalışmasını devam ettirir.



Şekil 13. İşletim sisteminin izleği zamanlayıcı kesmesi ile uyandırması

Bu tekniğin en güçlü yanlarından biri, izleğin zamanlayıcının yanı sıra donanımsal kesmelerle de uyandırılabilmesidir. Bir izlek örneğin bir pin kesmesinde beklediğini işletim sistemine bildirerek uykuya girdiğinde, işletim sistemi bu isteği kaydeder ve kesme olayı gerçekleştiğinde beklemekte olan izleği uyandırır. Böylece izlek çalışmaya, beklenen olay gerçekleştikten hemen sonra devam edebilir. Ancak zaman dilimli teknikte izleklerin uyandırılması ve izlekler arasında geçiş sadece zaman diliminin başında gerçekleştiği için, potansiyel olarak olarak zaman diliminin uzunluğu boyunca bir gecikme gerçekleşebilmektedir.

Zaman dilimli metotta işletim sisteminin, izlekler arasında geçişi sağlayabilmek için her zaman diliminin başında bir kesme ile çalışmakta olan izleği durdurması gerekmektedir. Hatta bütün izlekler uyuyor olsa dahi, işletim sisteminin periyodik olarak uyanarak sistemi kontrol etmesi gerekir. Dilimsiz metotta, bu periyodik uyanmaya gerek

olmaz. Dolayısıyla izleklerin çalışması bir kesme oluşmadığı müddetçe durdurulmaz ve işlemcinin uzun süren uyku olayları sırasında periyodik olarak uyandırılmadığı için uyku süresinde işlemcinin güç tüketimi düşük tutulabilir.

2.2.3.2. Donanımsal Soyutlama Katmanı (HAL)

ChibiOs işletim sistemi çekirdeğinin yanı sıra bu çekirdek ile çok iyi bir şekilde entegre olabilen bir donanımsal programlama kütüphanesi de sunmaktadır. Bu kütüphane STM32 mikrodenetleyicisinin dahili donanımlarının konfigüre edilmesi ve yönetilmesi için gerekli fonksiyonları sunar.

ChibiOs HAL'i modüler bir programlama arayüzü sunmaktadır. Genel olarak işlemcinin her bir donanımı için bir sürücü (*driver*) sunulmaktadır. Ancak bazı donanımların farklı kullanım şekilleri için farklı sürücüler vardır. Ayrıca bazı sürücüler direkt donanımsal olmayıp, diğer sürücülerin üstünde çalışan bir katman olarak gerçekleştirilmiştir.

ChibiOs işletim sistemi tarafından sunulan ve proje kapsamında kullanılan sürücüler şöyledir:

- *ADC Driver* : analog dijital dönüştürücü sürücüsü
- *EXT Driver* : pin kesme yönetim (External Interrupt) sürücüsü
- *GPT Driver* : genel amaçlı zamanlayıcı (General Purpose Timer) sürücüsü
- *I2C Driver* : I2C sürücüsü
- *PAL Driver* : pin (giriş çıkış) kontrol sürücüsü (Pin Abstraction Layer)
- *RTC Driver* : gerçek saat sürücüsü (Real Time Clock)
- *Serial Driver* : genel amaçlı seri data aktarım sürücüsü
- *SPI Driver* : SPI sürücüsü
- *USB Driver* : USB sürücüsü
- *Serial over USB Driver* : USB UART (CDC) protokol sürücüsü

Projenin gerçekleştirilmesi sırasında donanımlara erişim her zaman işletim sisteminin sunduğu bu sürücüler aracılığıyla gerçekleştirilmiştir. Zira işletim sisteminin doğru bir şekilde çalışması için donanımlar üzerinde tam hakimiyeti olması gerekmektedir.

Yalnız MAC katmanının zamanlayıcısı özel ihtiyaçlarından dolayı özel olarak gerçekleştirilmiştir (bkz. 2.2.5.3.1). Geliştirilen bu sürücünün, işletim sistemiyle

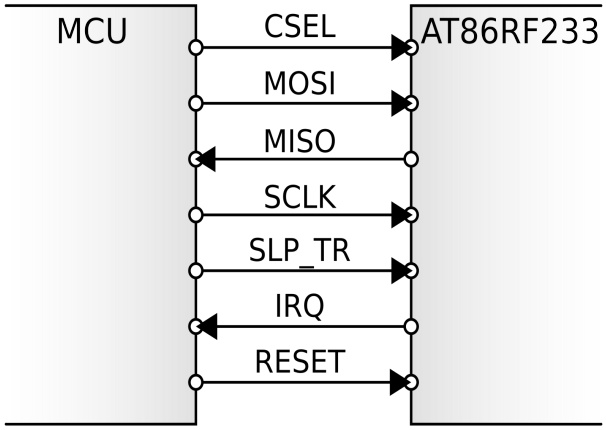
entegrasyonunun sağlanması için ChibiOs tarafından tanımlanan bazı kurallara uymak ve özel fonksiyonların kullanılması gerekmiştir.

2.2.4. Alıcı Verici Arayüzü

AT86RF233 alıcı verici entegresi ile paket transferinin gerçekleştirilmesi için geliştirilen program modülüdür. Bu modülün görevi entegrenin konfigüre edilmesi, gönderilecek paketlerin yazılması ve gelen paketlerin entegreden okunması olarak özetlenebilir.

Entegre ile iletişim işlemcinin SPI protokolü üzerinden yapılmaktadır. Bunun için işlemcinin SPI donanımlarından biri AT86RF233 entegresine ayrılmış ve haberleşme modülü kartı üzerinde uygun bağlantı yapılmıştır.

2.2.4.1. Pin Bağlantıları



Şekil 14. Alıcı verici entegresi pin bağlantıları

Şekil 14'te mikrodenetleyici ile alıcı verici entegresi arasındaki bağlantı gösterilmiştir. CSEL pini, erişilmek istenen entegreyi seçmek için kullanılır. Bu pin düşük seviye (0) iken entegre SPI arayüzünü aktif eder. Yüksek seviye (1) iken entegre SPI arayüzü devre dışıdır. SPI arayüzü üzerinden herhangi bir transfer yapılmadan önce bu pinin düşük seviyeye getirilmesi gerekir. Mikrodenetleyiciden alıcı verici entegresine giden veriler, MOSI pininden seri olarak gönderilir. Alıcı verici entegresinden

mikrodenetleyiciye giden veriler, MISO pininden yine seri olarak gönderilir. SPI protokolü senkron bir iletişim protokolüdür. Veriler MOSI ve MISO hattından her bir saat darbesinde aktarılır. Söz konusu saat darbesi SCLK pininden mikrodenetleyici tarafından sağlanır. Bu saat darbesi ayrıca entegre ile olan iletişimin hızını da belirler.

AT86RF233 entegresi 8MHz SPI saat hızını desteklemektedir. Aynı zamanda her iki bayt arasında 250ns süre olması gerekmektedir. STM32F4 işlemcisinin SPI donanımı DMA desteği ile baytları peş peşe, işlemci çekirdeğini uyandırmadan yazabilmektedir. Ancak SPI bu şekilde kullanıldığında baytlar arasına herhangi bir gecikme sokmak mümkün değildir. Dolayısıyla pratikte kullanılabilecek maksimum saat hızı 4 MHz'dir (1/250ns). İşlemcinin ana saat hızını düşürmeden erişilebilecek en yüksek saat hızı olan 3.125 MHz SPI çalışma hızı olarak seçilmiştir.

SLP_TR pini, iki fonksiyonlu bir pindir. Entegre konfigürasyon modunda iken bu pini düşük seviyeye çekmek alıcı verici entegresini uykuya sokar. Entegre gönderme (TX) modunda iken bu pini düşük seviyeye çekmek entegrenin gönderme işlemini başlatmasını tetikler. Şekil 17'de bu fonksiyon gösterilmiştir. Bu pin sürülmeden önce gönderilecek paketin FIFO'ya yazılmış olması gerekir.

IRQ (Interrupt) pini, entegrenin genel amaçlı kesme pinidir. Bildirim pini olarak çalışır. Entegre kendi içinde gerçekleşen bazı olayları denetimciye bildirmek için bu pini düşük seviyeye çeker. Mikroişlemci bu pin düşük seviyeye geldiğinde entegreden kesmenin sebebini okumalı ve buna göre gerekli işlemleri gerçekleştirmelidir.

Kesme pininden bildirilen olaylar şöyledir:

- *PLL_LOCK* : dahili PLL'in hedef frekansa kilitlenmesi
- *PLL_UNLOCK* : PLL'nin hedef frekanstan sapması
- *RX_START* : paket alınmaya başlandı
- *TRX_END* : paket gönderilmesi veya alınması tamamlandı
- *CCA_ED_DONE* : kanal meşgul testi tamamlandı
- *AMI* : adresi kabul edilen bir paket alındı (ZigBee modu)
- *TRX_UR* : paket belleğine erişim hatası
- *BAT_LOW* : besleme voltajı düşük uyarısı

Bu pinin en sık kullanılan fonksiyonu, paket dinleme modunda bir paket alındığında ve paket gönderme işlemi tamamlandığında oluşturduğu paket alındı (TRX_END) kesmesidir.

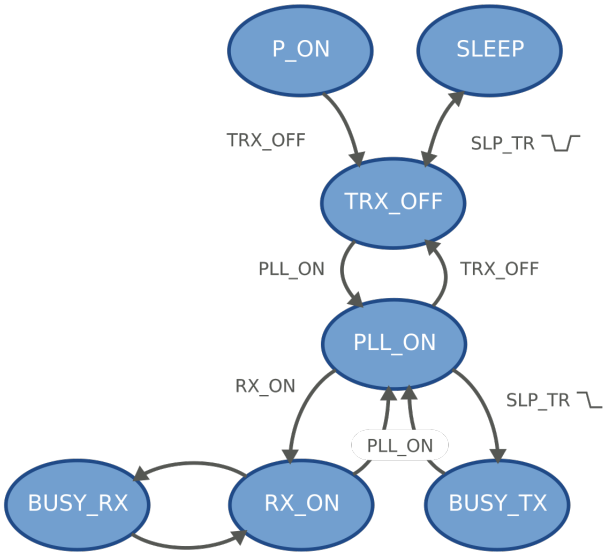
Reset pini, entegreyi sıfırlamak için kullanılır. Bu pinin düşük seviyeye çekilmesiyle birlikte entegre kendisini sıfırlar. Entegre böylece sanki güç yeni verilmiş gibi başlangıçta olması gereken konfigürasyona geçer. Bu özellik mikrodenetleyici sıfırlandığı zaman entegreyi de sıfırlamak için kullanılır.

2.2.4.2. Saklayıcı Erişimi

Entegre dahili saklayıcılar üzerinden kontrol edilmektedir. Entegrenin konfigürasyonu SPI üzerinden bu saklayıcılara gerekli bitler yazılarak yapılır. Entegrenin durumunu kontrol etmek için yine SPI arayüzü üzerinden saklayıcılar okunur.

2.2.4.3. Entegrenin Çalışma Evreleri

Şekil 15'te AT86RF233'ün çalışma evreleri basitleştirilerek gösterilmiştir. Evrelerin yanı sıra, evreler arasında geçiş yapmak için gerekli komutlar ve pin hareketleri de gösterilmiştir.



Şekil 15. AT86RF233 entegresinin çalışma evreleri

Entegre besleme ilk olarak verildiğinde P_ON (Power On) evresinde çalışmaya başlar. Bu evrede entegrenin giriş çıkış pinleri efektif olarak devre dışıdır. İlk iş olarak bu

evreden TRX_OFF evresine geçilmesi gerekir. Entegre sıfırlandığında bu evreden başlamaz, TRX_OFF evresinden başlar.

Entegre SLEEP evresinde uykudadır. Entegre uykudan çıkana kadar herhangi bir iletişim söz konusu olamaz. Bu evreye girme (uyuma) ve çıkış (uyanma) SLP_TR pini ile kontrol edilir.

TRX_OFF (Transmit Receive Off), Entegrenin bekleme evresidir. Bu evrede entegre ile SPI üzerinden haberleşme yapmak ve konfigürasyon mümkündür. Radyo saati (*PLL Clock*) aktif değildir.

PLL_ON evresine girilmesiyle, radyo taşıyıcı frekansını üreten dahili PLL (Phase Locked Loop) aktif edilir. Ancak bu evrede transfer gerçekleştirilmez. Bu moddan RX_ON ve BUSY_TX moduna geçilebilir.

RX_ON (Receive On) evresinde, entegre paket almaya hazırdır ve sürekli dinlemededir. Bu moda PLL_ON evresinden RX komutuyla geçilir. Bir paket alınmaya başlandığında BUSY_RX moduna geçilir.

Kablosuz bir paket alınırken entegre BUSY_RX evresindedir. Paket alınmaya başlandığında RX_START, paket alınması tamamlandığında TRX_END kesmesi oluşturulur. Paket alınması tamamlandığında otomatik olarak RX_ON evresine dönülür.

BUSY_TX evresinde kablosuz paket gönderilir. Bu evreye PLL_ON evresinden TX komutuyla geçilir ve gönderme işleminin tamamlanmasıyla PLL_ON evresine otomatik olarak geri dönülür.

2.2.4.4. Konfigürasyon ve Başlatma

Entegrenin başlangıç konfigürasyonu için uygulanan adımlar aşağıda verilmiştir.

1. SPI bağlantısı test edilir
2. Entegrenin kullanılmayan saat çıkışı devre dışı bırakılır
3. Kesme pini konfigüre edilir
4. Entegre TRX_OFF moduna sokulur
5. Entegre TRX_OFF moduna girene kadar beklenir

2.2.4.5. Radyo Kanalı Seçimi ve Güç Ayarı

AT86RF233 entegresi 2360 MHz - 2485 MHz bandında çalışabilmektedir. Frekans seçimi iki saklayıcı üzerinden yapılır. *CC_BAND* saklayıcısı ile iki farklı çalışma bandı seçilir. *CC_NUMBER* saklayıcısı ise taşıyıcı frekansının 0.5 MHz çözünürlük ile seçilmesini sağlar. Tablo 4'te frekans seçimi hesabı verilmiştir. *CC_BAND* ve *CC_NUMBER* saklayıcılarının değerlerini hesaplayan bir yalancı kod Şekil 16'da verilmiştir.

Tablo 4. AT86RF233 frekans kanalı seçimi

CC_BAND	CC_NUMBER	Band Aralığı	Taşıyıcı Frekansı
0x08	0x20-0xFF	2322-2433.5	2306 + 0.5 * CC_NUMBER
0x09	0x20-0xFF	2434-2527	2434 + 0.5 * CC_NUMBER

```
void calc_freq (unsigned fc_mhz)
{
    if (fc_mhz < 2435)
    {
        cc_band = 0x08;
        cc_number = (fc_mhz - 2306)*2;
    }
    else
    {
        cc_band = 0x09;
        cc_number = (fc_mhz - 2434)*2;
    }
}
```

Şekil 16. AT86RF233 kanal saklayıcılarını hesaplanması

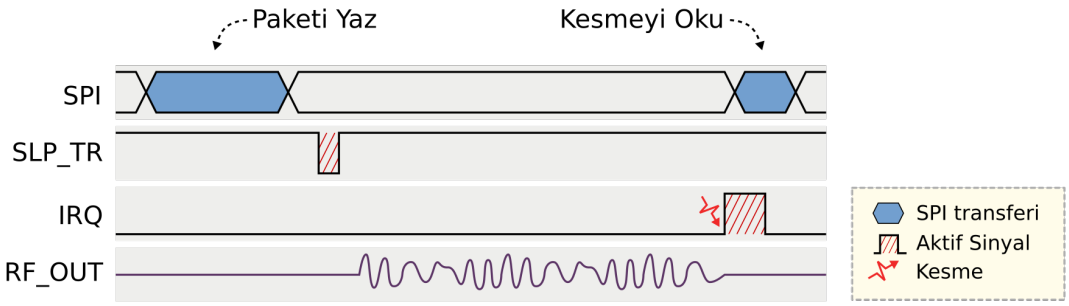
Çıkış gücü *PHY_TX_PWR* saklayıcısına yazılan 4 bitlik bir değer ile 16 kademe olarak ayarlanabilir. -17 dBm ile 4 dBm arasında bir değer seçilebilir.

2.2.4.6. Paket Gönderme

Şekil 17'de bir transfer sırasında alıcı verici hattında gerçekleşen olaylar gösterilmiştir. Paket gönderme için takip edilen adımlar şöyledir.

1. Entegreyi PLL_ON evresine sok.

2. Paketi yaz
3. SLP_TR pinini tetikle
4. IRQ_TRX_END kesmesini bekle

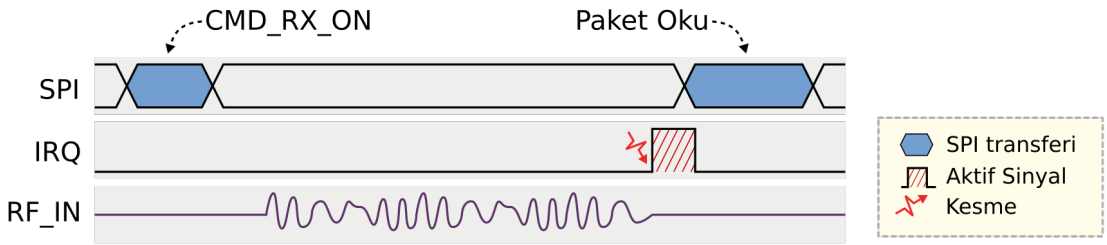


Şekil 17. AT86RF233 entegresi ile paket gönderme işlemi

2.2.4.7. Paket Dinleme ve Alma

Dinleme işleminin başlatılması için şu adımlar takip edilir. Şekil 18'de bu adımlar görsel olarak gösterilmiştir.

1. Entegreyi *PLL_ON* evresine sok
2. *CMD_RX_ON* komutunu gönder
3. *IRQ_TRX_END* kesmesini bekle
4. Alınan paketi oku



Şekil 18. AT86RF233 entegresi ile paket alma

Senkronizasyon işlemleri için bazı paketlerin alındığı zamanın hassas bir şekilde bilinmesi gerekir. Bunun için IRQ pin hattı erişim katmanı zamanlayıcısının girişine bağlanmıştır. Kesme gerçekleştiği andaki zamanlayıcı değeri donanımsal olarak yakalanır ve bu yakalanan değer bellekte belirlenen değişkene kaydedilir.

2.2.4.8. Enerji Seviyesi Ölçümü

AT86RF233 entegresi, kanal çakışmasını önlemek için kullanılmak üzere temiz kanal belirleme (CCA : *Clear Channel Assessment*) özelliğine sahiptir. Bu özellik aktifken amaç kanalda başka bir vericinin transfer yapıp yapmadığını belirlemektir.

CCA, *PHY_CC_CCA* saklayıcısına yapılan bir yazma işlemi ile başlatılır. 180µs sonra, kanaldaki enerji seviyesini gösteren bir sonuç üretilir. İşlemin tamamlandığı *CC_ED_DONE* kesmesi ile bildirilir. Sonuç *PHY_ED_LEVEL* saklayıcısından okunur.

2.2.4.9. Programlama Arayüzü (API)

phy_init fonksiyonu programlama arayüzünün başlangıç konfigürasyonunu yapar. Sadece bir defa çağrılabilir. *phy_start* fonksiyonu alıcı vericinin çalışmasını başlatır. *phy_stop* fonksiyonu ise alıcı vericiyi durdurur.

phy_transmit(char data, size_t size)* fonksyionu bir paketi transfer eder. *data* paket içeriğinin bulunduğu dizidir. *size* paketin uzunluğunu bildirir, maksimum 127 olabilir. *phy_receive(char* data, systime_t timeout)* fonksiyonu ise bir paket alır. Eğer alıcı dinleme modunda değil ise ilk önce alıcıyı dinleme moduna sokar ve paket alındıktan sonra dinleyici modundan çıkarır. Eğer alıcı daha önceden *phy_start_listening* fonksiyonu ile dinleme moduna sokulmuş ise, dinleme modundan çıkılmaz. Benzer şekilde *data* alınan paketin yazılacağı bellektir. *timeout* maksimum dinleme süresidir. Geri dönüş değeri alınan paketin uzunluğudur. 0 dönülmesi, zaman aşımı gerçekleştiğini bildirir.

phy_receive_timed fonksiyonu, *phy_receive* fonksiyonuna benzer şekilde çalışır ancak ekstra bir *timestamp* parametresi kabul eder. Alınan paketin, alınma zamanı bu parametrenin belirttiği değişkene kaydedilir. Bu fonksiyon "MAC Timer" ile birlikte çalışır. Kullanılmadan önce "MAC Timer"ın başlatılmış olması gerekir.

phy_start_listening, *a*lıcıyı dinleme moduna sokar ve hemen döner, paket alımını beklemez. Alınan paketin okunması için phy_receive fonksiyonun çağrılması gerekir. *phy_stop_listenin*, alıcıyı dinleme modundan çıkarır. *phy_is_listening*, alıcının dinleme modunda olup olmadığını bildirir.

phy_cca fonksiyonu, CCA (Clear Channel Assessment) operasyonunun başlatır ve ölçülen enerji seviyesini döner. phy_frame_transmission_time fonksiyonu, Bir paketin iletim süresini hesaplar. Paketin uzunluğu parametre olarak verilir ve dönüş değeri

mikrosaniye cinsinden iletim süresidir. *phy_set_tx_power* fonksiyonu ile, çıkış gücü ayarlanır.

phy_timer_cb, fonksiyonu erişim katmanı zamanlayıcısı (bkz. 2.2.5.3.1) tarafından IRQ pininde bir kesme gerçekleştiğinde, zaman bilgisi ile birlikte çağrılır. Alıcı vericinin erişim zamanlayıcısı ile entegrasyonu bu fonksiyon ile sağlanır. *phy_receive_timed* fonksiyonunun çalışması için gereklidir.

2.2.5. Ortam Erişim Kontrol Katmanı (MAC Layer)

802.15.6 uyumlu erişim kontrol katmanının gerçekleştirildiği modüldür. Ağ geçidi ve uç birim cihazları için iki farklı modül olarak geliştirilmiştir. Paket işleme, alma ve gönderme fonksiyonları gibi bazı temel fonksiyonlar iki modül için de ortaktır.

Erişim kontrol tekniği olarak "İşaretli bölünmüş zamanlı mod (*beacon mode with superframes*)" (bkz. 1.3.2.3) gerçekleştirilmiştir. Sadece rastgele erişim fazı ve yönetilen erişim fazı içeren bir zaman dilimi yapısı kullanılmıştır. Herhangi bir gizlilik ve güvenlik mekanizması gerçekleştirilmemiştir.

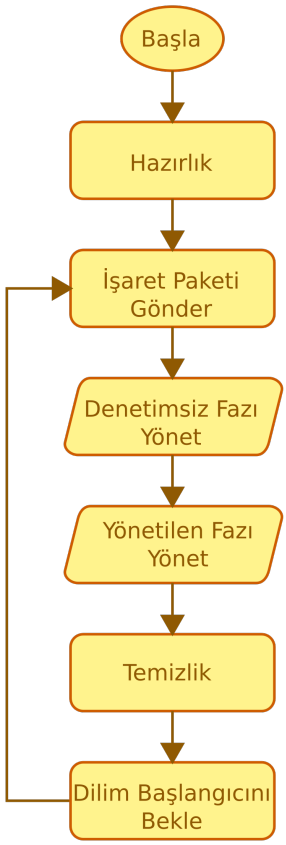
2.2.5.1. Ağ Geçidi Modülü

Ağ geçidi modülünün sorumlulukları ve işlevleri şöyle özetlenebilir.

- Ağ kurma
- Zaman dilimini yönetme
- İşaret paketini gönderme
- Bağlantı isteklerini kabul etme ve düğümleri yönetme
- Yönetilen erişim fazı için zaman tahsislerini yönetme
- Düğümlerden gelen paketleri alıp, servis katmanına iletme
- Servis katmanından gelen paketleri, düğümlere iletme

2.2.5.1.1. Program Ana Döngüsü

Ağ geçidinin ağı yönetmek için kullandığı program ana döngüsü Şekil 19'da verilmiştir. Hazırlık aşamasında, Ağ geçidinin çalışması için gerekli başlangıç ayarlamaları yapılır. Zamanlayıcı konfigüre edilir ve başlatılır. Fiziksel katman başlatılır. Her zaman dilimi için döngü bir defa tamamlanır.



Şekil 19. Ağ geçidinin zaman dilimini yönetimi

Döngünün başında ilk olarak işaret paketi gönderilir. İşaret paketinin gönderilme zamanı düğümler tarafından zaman diliminin başlangıcı olarak alınacaktır. Ardından rastgele erişim fazını yöneten alt program çağrılır. Bu alt programın rastgele erişim fazının süresi bitmeden hemen önce dönmesi beklenir. Daha sonra yönetilen fazı yönetecek alt program çağrılır. Bu alt programın da zaman dilimi süresi bitmeden bir müddet önce dönmesi beklenir. Bu kalan kısa sürede bazı temizlik işlemleri yapılır. Örneğin bağlantısı kesilen düğümlerin bilgilerinin silinmesi gibi. Ardından, döngünün zamanlayıcıya

senkronize olması için bir sonraki zaman diliminin başlangıcı beklenir ve döngü tekrarlanır.

2.2.5.1.2. İşaret Paketinin Yapısı

İşaret paketinin yapısı zaman diliminin yapısını ve ağ geçidinin zamanı nasıl yönettiğini anlamak açısından önemlidir. Şekil 20'de işaret paketinin içeriği verilmiştir.

Şekil 20. İşaret paketinin yapısı

Başlığın kontrol bölümünde, işaret paketine özel olarak bulunan bilgiler şöyledir.

- ACK_POLICY_NACK : alındı paketi gerektirmez
- SECURITY_LEVEL_NONE : şifreleme yok
- EAP yok
- Paket türü: işaret
- B2 yok
- İşaret sırası
- Birliktelik (*coexistance*) yok
- Pasif zaman dilimi yok

Başlığın alıcı ID kısmına, bütün bağlı ve bağlanmamış düğümleri hedefleyen özel adres (0x00) yazılır. Gönderici kısmına ağ geçidinin düğüm düğüm kimlik numarası (node ID) yazılır. BAN ID kısmına ağın kimlik numarası yazılır. Bu çalışmadaki yazılım, aynı

kanalda birden fazla ağı desteklemediği için ağ geçidi kimlik numarası ve BAN kimlik numarası kısımları sabittir. EUI-48 kısmına ağ geçidi cihazının 48 bitlik (6 bayt) eşsiz kimlik numarası (*unique ID*) yazılır.

Tahsis dilimi adedi kısmına zaman diliminin kaç tahsis dilimine bölümlendiği yazılır. Tahsis dilimi uzunluğu kısmına, tahsis diliminin süresi yazılır. Tahsis dilimi süresi mikro saniye cinsinden şöyle hesaplanır:

$$T_{us} = T \times 500\,\mu s + 500\,\mu s \tag{2.2.1}$$

Burada T, pakete yazılan sayıdır. Zaman diliminin süresi mikro saniye cinsinden, tahsis dilimi sayısı ve süresi kullanılarak şu şekilde belirlenebilir:

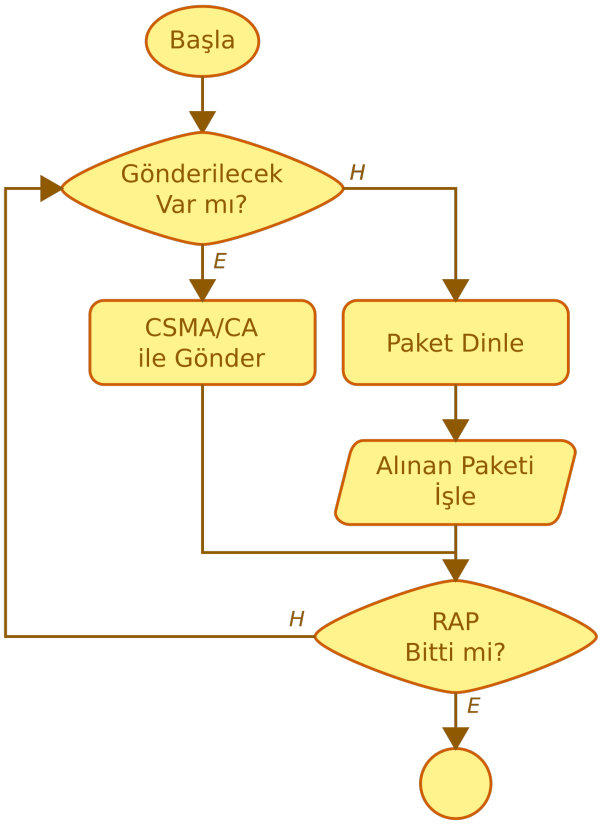
$$Z_{us} = T_{us} \times T_n \tag{2.2.2}$$

"RAP1 Sonu", RAP1 fazının bittiği tahsis diliminin sıra numarasıdır. "RAP2 Başlangıcı", RAP2 fazının başlangıcını belirtir, RAP2 fazı kullanılmadığı için 0'dır."RAP2 Sonu", RAP2 fazının sonunu belirtir, aynı sebeple 0'dır.

"MAC Özellikleri", erişim katmanının, desteklediği tercihe bağlı özellikleri bildirmek için kullanılır. "PHY Özellikleri", fiziksel katmanın özelliklerini (modülasyon çeşidi, kodlama bilgileri vs.) bildirmek için kullanılır. Bu çalışmada farklı sistemlerle uyumluluk beklenmediği için bu alanlar kullanılmamıştır.

2.2.5.1.3. Rastgele Erişim Fazı

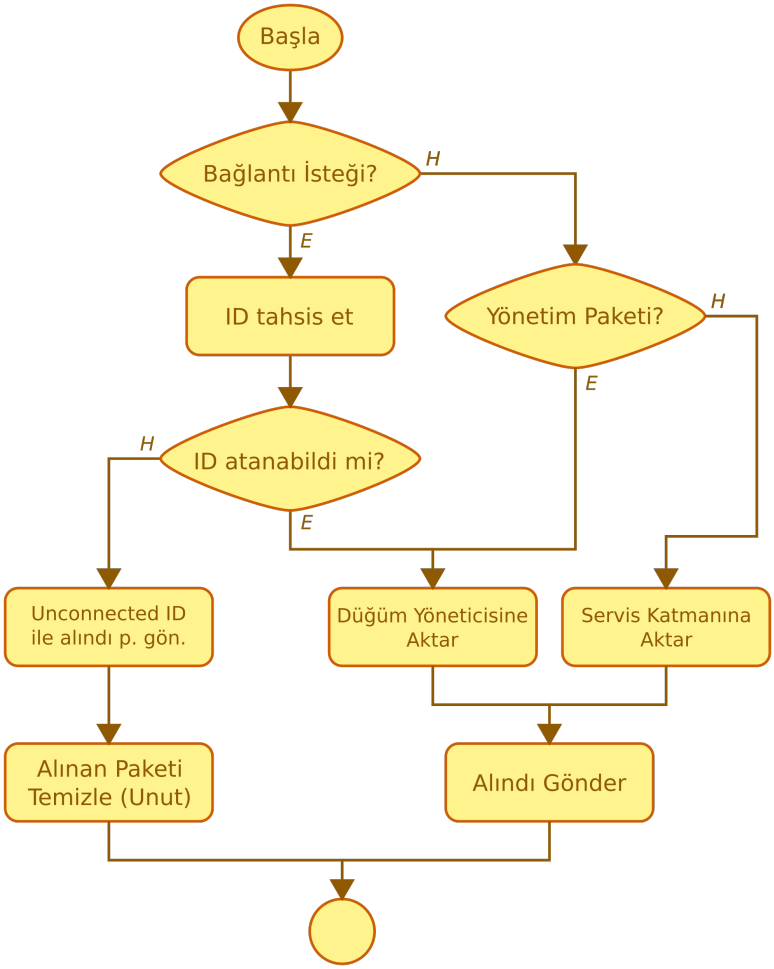
Rastgele erişim fazını (RAP – Random Access Phase) yöneten alt programın akış diyagramı Şekil 21'de verilmiştir. Bu alt programa işaret paketinin gönderilmesinden hemen sonra girilir ve rastgele erişim fazı sona erdiğinde çıkılır. Bu fazda henüz ağa bağlanmamış düğümler ile ve bağlanmış düğümler ile iletişim kurulur. Ağa bağlanmamış düğümlerden sadece bağlantı isteği paketi kabul edilir.



Şekil 21. Ağ geçidinin rastgele erişim fazını yönetimi

Fazın başlangıcında, gönderilmesi gereken paket olup olmadığı kontrol edilir. Eğer varsa, ağ geçidi ilk önce kendi paketlerini göndermeye çalışır. Ancak bunun için CSMA/CA tekniği kullanılır. Ağ geçidi elindeki paketlerin hepsini gönderdikten sonra, kalan zamanda gelen paketleri dinler. Alınan paketlerin işlenmesi için çağrılan alt programın akış diyagramı Şekil 22'de verilmiştir.

Bir bağlantı isteği (*Connection Request*) paketi alındığında, ilk olarak bir ID ataması yapılır. ID atanabilmesi düğümle olan haberleşmenin devam edebilmesi için önemlidir. 802.15.6 protokolü bağlantı kurulması aşamasında, herhangi bir ID kurulmadan iletişim yapmayı desteklemektedir. Ancak daha uzun EUI-48 ID ile çalışmayı gerektirir. Eğer ID tahsisi başarısız olursa, düğüme *Unconnected ID* ile bir alındı paketi gönderilir. Bu özel olarak ayrılmış bir ID numarasıdır. Bunu alan düğüm, bağlantı isteğinin reddedildiğini farz edecektir. ID ataması ancak, ağ geçidine bağlanan düğüm sayısı sınırı aşılırsa başarısız olur. Ancak ID atamasının başarılı olması, bağlantı isteğinin kabul edildiği anlamına gelmez. Bağlantı isteğinin işlenmesi için, alınan paket düğüm yöneticisine aktarılır.



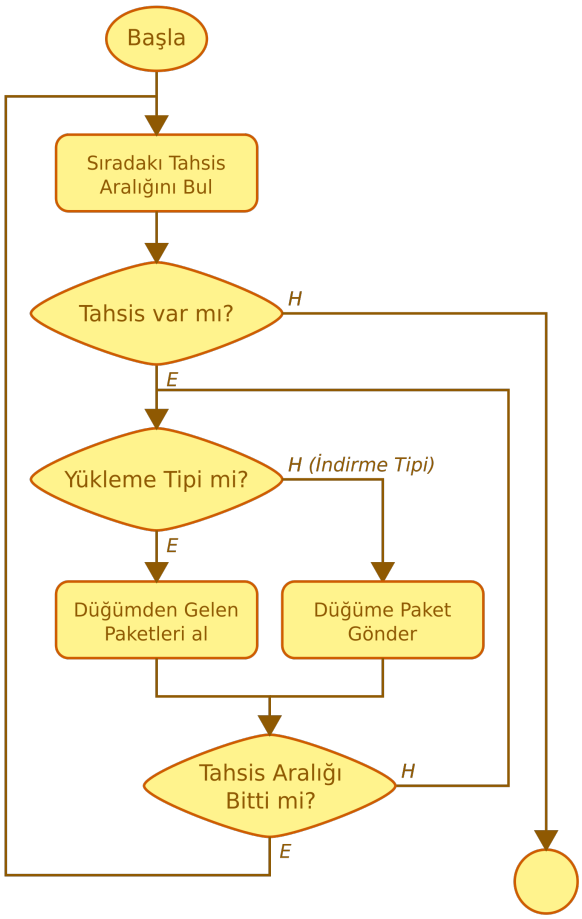
Şekil 22. Ağ geçidinin RAP'ta alınan bir paketi işlemesi

Bağlantı isteği olmayan yönetim paketleri (bağlantı kesme gibi), direkt olarak düğüm yöneticiye aktarılır. Yönetim paketi olmayan paketler, veri paketleridir. Bunlar da, bilgisayara aktarılmak üzere servis katmanına yönlendirilirler. Paket aktarıldıktan sonra bir alındı paketi gönderilir.

2.2.5.1.4. Yönetilen Erişim Fazı

Bu faz, tahsis aralıkları bazında yönetilir. Düğüm bağlantı isteğinde hangi amaçla (yükleme-indirme) kaç adet tahsis dilimi talep ettiğini bildirir. Bu taleplere göre tahsis edilen dilimlere, tahsis aralığı denir. MAP boyunca, ağ geçidi bu tahsis aralıklarının listesini sırayla tarayarak, ilgili zaman aralığı boyunca gerekli işlemleri yapar. Bu da eğer tahsis aralığı yükleme için tahsis edilmişse, düğümden gelen paketleri alıp servis

katmanına aktarmak, indirme için tahsis edilmişse, servis katmanından gelen paketleri düğüme aktarmaktdır. İlgili tahsis aralığının sonuna gelindiğinde bir sonraki tahsis aralığına geçilir. Tahsis aralıklarının hepsi işlendiğinde MAP fazının yönetimi bitmiştir. Şekil 23'te MAP fazının yönetimini özetleyen bir akış diyagramı verilmiştir.



Şekil 23. Ağ geçidinin yönetilen erişim fazını kontrol etmesi

2.2.5.1.5. Bağlantı ve Tahsislerin Yönetimi

Uçbirimlerden gelen bağlantı ve tahsis dilimi isteklerinin yönetimi ayrı bir izlekte gerçekleştirilir. Bağlantı isteği ve bağlantı isteği kesme paketleri bu birime aktarılır ve işlenir. Burada amaç, erişim katmanını yöneten ağ geçidi izleğinin çalışmasını yavaşlatmamaktır.

Bağlantı yönetimi modülünün ilk görevi, uç birime bir ID atamaktır. Ancak bu işlemin hızlı bir şekilde gerçekleşmesi gerektiği için ana yönetim izleğinden yapılır. Eğer ID atama başarısız olursa, uç birime *Unconnected ID* ile bir alındı paketi gönderilir. Bu ancak azami bağlantı sayısına erişildiğinde gerçekleşir.

Bağlantı yönetim modülü, gelen bağlantı isteği paketlerini inceler. Eğer uç birimin verdiği bilgiler kabul edilirse, istenen adette tahsis dilimi atanır. Eğer talep edilen uzunlukta yeterince tahsis dilimi yoksa, düğümün bağlantısı reddedilir. Bağlantının kabul edilmesi durumunda, atanan tahsis dilimlerinin sıra numaralarını içeren bir bağlantı atama paketi gönderilir. Bağlantı isteğinin reddedilmesi durumunda bağlantı atama paketi, bağlantının neden reddedildiğini bildiren sebep koduyla birlikte gönderilir.

Bir düğümün bağlantısının kesilmesi de bu modül tarafından gerçekleştirilir. Bağlantı kesilmesi iki sebeple gerçekleşebilir.

1. Üst katmandan gelen bağlantı kesme komutu
2. Uç birimden gelen bağlantı kesme paketi

Üst katmandan gelen bir bağlantı kesme komutuyla, düğümün bağlantısı kesilmiş olur ve gerekli temizlik işlemleri yapılır. Aynı şekilde düğümden gelen bir bağlantı kesme paketi de bu temizlik işlemlerini başlatır. Temizlik işlemleri için, düğüme atanmış olan ID ve tahsis dilimleri serbest olarak işaretlenir ve düğüme ait paket kuyrukları temizlenir.

2.2.5.1.6. Programlama Arayüzü

mac_hub_start fonksiyonu, ağ geçidi erişim katmanını başlatır. Bu fonksiyonun sadece bir defa çağrılması beklenir. Oluşturulan iş parçacığına bir işaretçi döner. *mac_hub_started* fonksiyonu, modülün başladığını bildiren bir fonksiyondur. *true* veya *false* değerlerini döner.

mac_hub_is_connected fonksiyonu verilen kimlik numarasına sahip bir bağlı düğümün bulunup bulunmadığını bildirir. *mac_hub_get_connected*, bağlı olan düğümlerin bir listesini verir.

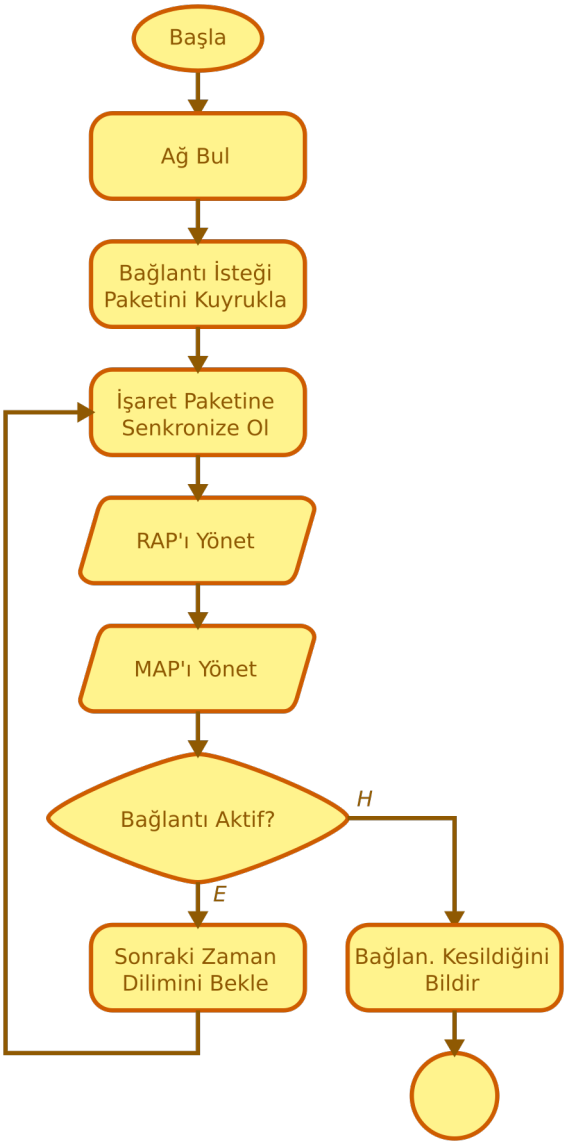
mac_hub_queue fonksiyonu, ile bir düğüme gönderilecek bir paket kuyruklanır. Verilen paket içeriği ayrı bir paket kuyruğuna kopyalanır ve gerekli başlık bilgileri eklenir. Eğer kuyruk dolu ise işlem başarısız olur ve fonksiyon *false* değerini döner. *mac_hub_disconnect* fonksiyonu ile verilen kimlik numarasına ait düğümün bağlantısı kesilir. Bağlantı kesilmesi için ilk olarak bir bağlantı kesme paketi oluşturulur ve paket

iletim kuyruğuna konulur. Ayrıca düğümün bağlantısının kesilmekte olduğu işaretlenir. Bağlantı kesilme paketi, düğüm tarafından alındığında bağlantı kesme işlemi tamamlanmış olacaktır.

2.2.5.2. Düğüm Modülü

Uçbirim düğümünün ağ bağlantısını yönetmek için gerçekleştirilen modüldür. Bu modülün sorumlulukları şöyle listelenebilir.

- ağ bulmak
- işaret paketine senkronize olup, RAP, MAP fazlarını yönetmek
- bağlantı isteğini göndermek
- bağlantı kesme paketini göndermek
- servis katmanından aktarılan paketleri ağ geçidine göndermek
- ağ geçidinden gelen paketleri servis katmanına aktarmak



Şekil 24. Uçbirim düğümün ağı yönetmesi

Şekil 24'te ağın yönetimini genel hatlarıyla gösteren bir akış diyagramı verilmiştir. Bu diyagramda görüldüğü gibi, ağ bağlantısı ilk olarak bir ağın bulunması ile başlar. Ağın bulunması için ağ geçidi tarafından yayınlanan bir işaret paketi aranır. Bu işaret paketinin alınmasıyla birlikte bir bağlantı isteği paketi kuyruk sırasına konur ve zaman diliminin yönetilmesine başlanır. Görüldüğü gibi, düğüm bu noktada henüz ağa bağlanmış olmasa da ağa bağlı imiş gibi zaman dilimine senkronizedir. Ancak bağlantı isteği kabul edilinceye kadar, düğüm sadece bağlantı kurmak için gerekli olan haberleşmeyi yapar. Bu da bağlantı isteği paketinin gönderilmesi ve bağlantı atama paketinin kabul edilmesidir. Bu işlemler, rastgele erişim fazında (RAP) gerçekleştirilir. Bağlantı isteğinin kabul edilmesinin

ardından düğüm, eğer kendisine yapılan atamalar varsa, bu tahsis dilimlerinde (MAP sürecinde) paketlerini gönderir ve alır.

Tahsis dilimi atanmamış bir düğümün sürekli olarak RAP'ta iletişim kurması gerekecektir. Ancak RAP'taki iletişim enerji tüketimi açısından verimli olmadığı için, bu çalışmada gerçekleştirilen düğüm yönetim yazılımı her zaman en az 1 yükleme ve en az 1 indirme dilimi tahsis edilmesini talep eder. Dolayısıyla rastgele erişim fazı sadece bağlantının kurulması sırasında kullanılır.

Yönetilen erişim fazının yönetimi, ağ geçidi için gerçekleştirilen programla hemen hemen aynıdır. Sadece iletişimin yönü farklıdır. Daha fazla bilgi için bölüm 2.2.5.1.4'e bakılabilir.

2.2.5.2.1. Programlama Arayüzü

mac_init fonksiyonu modülü hazırlar. Sadece bir defa, diğer fonksiyonlar kullanılmadan önce çağrılmalıdır. *mac_node_start* düğümün erişim katmanını yönetecek iş parçacığını başlatır. Bu işlemle birlikte düğüm bir bağlantı aramaya başlayacaktır. Bu fonksiyon tekrar çağrılmadan önce *mac_node_stop* fonksiyonunun çağrılması şarttır. *mac_node_stop* iş parçacığının çalışmasını sonlandırmak ve temizlik işlemlerini yapmak için çağrılır. Bu fonksiyon ancak bağlantı kesildikten sonra çağrılmalıdır. *mac_node_disconnect* düğümün bağlantısını kesme işlemini başlatır. Bir bağlantı kesme (Disconnect) paketi oluşturulur ve ağ geçidine gönderilmek üzere kuyruğa koyulur. Bağlantının kesilmesi ancak bu paketin alınması onaylandıktan sonra tamamlanmış olacaktır.

mac_node_transmit(char data, unsigned size)* fonksiyonu verilen paketi ağ geçidine göndermek üzere kuyruğa koyar. *data* paket içeriğinin başlangıcını, *size* uzunluğunu gösterir. Veri kuyruğa koyulmadan önce kopyalanacaktır. *mac_node_allocate_transmit* paket kuyruğuna koyulmak üzere, paket havuzundan bir paket ayırır. Eğer paket kalmamışsa *NULL* döner. Verilen dizinin uzunluğu maksimum paket uzunluğundadır. Bu şekilde ayrılan bir paketin *mac_node_queue_transmit* ile kuyruğa koyulması gerekir. *mac_node_queue_transmit* daha önce *mac_node_allocate_transmit* ile ayrılan bir paketi, iletim kuyruğuna koyar.

mac_node_conchange_hook fonksiyonu, erişim katmanının uygulama katmanı ile olan entegrasyonu için gereklidir. Erişim katmanında gerçekleşen bağlantı değişikliği

olaylarını bildirmek için bu fonksiyon çağrılır. Bu fonksiyon erişim katmanında tanımlanmaz, uygulama katmanında tanımlanması gerekir.

En önemli işlevi, bağlantı sağlandığında ve bağlantı kesildiğinde bunun uygulama katmanına bildirilmesidir. Bağlantı kurulduğunda, servis katmanı başlatılır, bağlantı kesildiğinde de servis katmanı durdurulur ve *mac_node_stop* fonksiyonu çağrılır.

Fonksiyonun parametreler şöyledir:

- *unsigned devid*
- *MAC_NODE_STATUS status*

devid düğüme atanan kimlik numarasıdır. *status* düğümün o anki durumunu bildiren bir durum kodudur. Şu değerleri alabilir.

- *IDLE* : boşta (başlangıç değeri)
- *SEARCHING* : ağ aranıyor
- *CONNECTING* : bağlantı isteği gönderildi
- *CONNECTED* : bağlantı isteği kabul edildi
- *REJECTED* : bağlantı isteği reddedildi
- *NOT_FOUND* : ağ bulunamadı (zaman aşımı)
- *LOST* : bağlantı kaybedildi
- *DISCONNECTED* : bağlantı kesildi (istek üzerine)

2.2.5.3. Ortak Özellikler

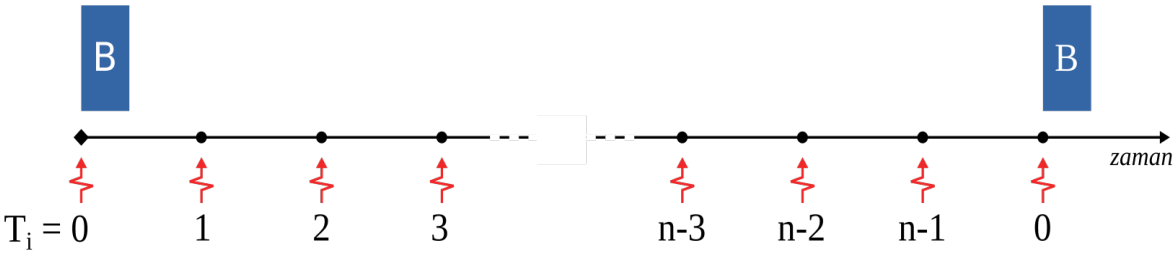
2.2.5.3.1. Zamanlayıcı

Erişim katmanının en önemli işlevlerinden biri, paketlerin belli bir zaman aralığında gönderilmesi ve alınmasıdır. Bu çalışmada gerçekleştirilen 802.15.6 protokolü, zaman bölümlemeli çoklu erişim (TDMA) prensiplerine göre çalıştığı için zamanın hassas bir şekilde yönetilmesi özel önem arz etmektedir. Bunun gerçekleştirilebilmesi için, STM32 mikrodenetleyicisinin donanımsal zamanlayıcılarından biri erişim kontrolü için ayrılmıştır.

Bu zamanlayıcının görevleri tahsis dilimlerinin başlangıcını ve uzunluğunu belirlemek, zaman diliminin başlangıcını ve uzunluğunu kontrol etmek, gelen paketlerin alınma zamanını belirlemek, işaret paketine senkronize olmaktır.

Zamanlayıcı olarak, mikrodenetleyicinin sunduğu 16 bitlik zamanlayıcılardan biri kullanılmıştır. Zamanlayıcı frekansı 1 MHz olarak ayarlanmıştır. Yani zamanlayıcı çözünürlüğü 1µs'dir.

Zamanlayıcı, sayacının uzunluğu tahsis diliminin uzunluğuna göre ayarlanır. Örneğin 50 ms tahsis süresi için, sayaç limiti 50000 değerine ayarlanır. Sayaç bu değere ulaştığında sıfırlanır ve bir kesme oluşur. Bu kesmede tahsis dilimi sayacı bir arttırılır. Tahsis dilimi sayacı bir bakıma ana zaman diliminin hangi noktasında bulunulduğunu belirtir. Şekil 25'te sayacın kesmelerle arttırılması gösterilmiştir.



Şekil 25. Erişim katmanı zamanlayıcısının tahsis dilimlerini sayması

Zamanlayıcının en önemli işlevlerinden biri, zaman dilimini işaret paketinin başlangıcına senkronize edebilmesidir. Bunun için, alıcı verici entegresinin kesme pini zamanlayıcının yakalama pinlerinden birine bağlanmıştır. Bu pinde oluşan bir yükselen kenar işaretinin zamanı, o anki sayaç değeri olarak kaydedilir. Daha sonra paket süresi ve kesmenin gecikme süresi hesaba katılarak, kayma süresi hesaplanır. Bu kayma süresi, aynı zamanda zaman diliminin başlangıcı olan işaret paketinin başlangıç anına olan uzaklığı vermektedir. Zamanlayıcı bu müddet kadar kaydırıldığında, ağ geçidinin kontrol ettiği zaman dilimine senkronize olunmuştur. İdeal şartlarda bu işlemin sadece bir defa gerçekleştirilmesi yeterli olur. Ancak pratikte kullanılan saat kristallerinin frekansları toleranslara tabi olduğu için, her zaman diliminde bir miktar kayma gerçekleşmektedir. Bu da uç birim ve ağ geçidi arasındaki senkronizasyonun bozulmasına sebep olur. Bunu engellemek için her zaman diliminin başında bu işlem gerçekleştirilir.

2.2.5.3.2. Paket Kuyrukları

Paket kuyrukları (*queue*), iletişim paketlerinin farklı katmanlar ve modüller arasındaki geçişlerini sağlamak ve senkronize etmek için oluşturulmuştur. Paket kuyrukları, ilk önce giren ilk çıkar (FIFO: First in First Out) prensibi ile çalışırlar. Farklı işleyiciler arasında paketlerin güvenli bir şekilde aktarılabilmesi için işletim sistemi tarafından sunulan *MailBox* arayüzü kullanılmıştır.

a. Ağ Geçidi Cihazı İçin Kullanılan Kuyruklar

a.i. Erişim Katmanı Giden Paket Kuyrukları

Ağ geçidi cihazına bağlı olan her uçbirim düğümü için ayrı bir giden paket kuyruğu bulunmaktadır. Bunun sebebi, düğümlerin iletişiminin ayrı zaman dilimlerinde gerçekleştirilmesidir. Yönetilen erişim fazında bir indirme tahsis diliminde gönderilecek paketler, o tahsis diliminin tahsis edildiği düğümün giden paket kuyruğundan alınır.

a.ii. Rastgele Erişim Fazı Giden Paket Kuyruğu

Rastgele erişim fazında, herhangi bir düğüme paket gönderilebileceği için, bu fazda gönderilecek paketler düğümden bağımsız olarak tek bir kuyrukta toplanır. Erişim katmanı, rastgele erişim fazında göndereceği paketleri bu kuyruktan alır.

a.iii. Gelen Paket Kuyruğu

Uç birim düğümlerinden gelen bütün paketler tek bir kuyrukta toplanır. Bu kuyruktan paketler, servis katmanı tarafından, herhangi bir işleme tabi tutulmadan bilgisayara gönderilir.

a.iv. Servis Katmanı Gelen Paket Kuyruğu

Bilgisayardan gelen paketler servis katmanında işlenmek üzere bu kuyrukta toplanırlar. Eğer servis paketi, ağ geçidi cihazına adreslenmişse servis katmanı tarafından işlenirler. Ancak bağlı olan bir düğüme adreslenmiş paketler, o düğüme aktarılmak üzere ilgili erişim katmanı giden paket kuyruğuna aktarılırlar.

b. Uçbirim Cihazında Kullanılan Kuyruklar

b.i. Giden Paket Kuyruğu

Ağ geçidi cihazına gönderilecek paketler bu kuyrukta toplanırlar. Düğüm kendisine tahsis edilen yükleme diliminde göndereceği paketleri bu kuyruktan alır.

b.ii. Gelen Paket Kuyruğu

Ağ geçidi cihazından gelen paketler bu kuyrukta toplanırlar. Servis katmanı, bu kuyruğa gelen paketleri alır, işler ve cevaplarını giden paket kuyruğuna aktarır.

2.2.6. Servis Katmanı

802.15.6 protokolü tam bir iletişim sisteminin gerçekleştirilmesi için gerekli bütün katmanları tanımlamamaktadır. Örneğin Bluetooth protokolü, farklı uygulamaların geliştirilmesi için gerekli profiller ve bunların sürücülerini tanımlar. Bu profillere uygun olarak geliştirilen bir Bluetooth cihaz herhangi bir bilgisayar veya akıllı telefona bağlanarak, herhangi bir sürücü veya özel yazılım kurulumu gerektirmeden kullanılabilmektedir. 802.15.4 protokolü de, 802.15.6'ya benzer bir konumdadır. Ancak 802.15.4'ü temel alan ZigBee, 6LoWPAN gibi iletişim sistemleri farklı organizasyonlar tarafından geliştirilmiştir ve kullanımdadır.

Bu çalışma kapsamında, 802.15.6 protokolünü tamamlamak için hasta takip sensör sistemlerinde kullanılmak üzere, modüler ve kolaylıkla genişletilebilir bir servis katmanı oluşturulmuştur. Bu sistemin ana prensibi farklı sensörler ve işlevler için farklı servis modüllerinin gerçekleştirilmesidir. Bu modüller birbirlerinden tamamen bağımsız olarak gerçekleştirilir ve alıcı sistemin bağlandığı kablosuz uçbirim cihazındaki servis modüllerini önceden bilmesi gerekmez. Makineden Makineye İletişim (M2M) prensiplerine uygun olarak tasarlanan bu sistemde, bir cihaza bağlandıktan sonra o cihazdaki modüllerin listesini sorgulayıp ona göre cihazı konfigüre etmek, çalıştırmak ve iletişim kurmak mümkündür.

Servis katmanı da kendi içerisinde 3 katmandan oluşur.

- Yönetici
- Dağıtıcı
- Modüller

2.2.6.1. Modüller

Modüller, servis katmanında asıl işlemlerin gerçekleştirildiği katmandır. Bir sistemde birden fazla modül bulunabilir. Bir uç birim cihazındaki, her farklı özellik için bir modül bulunabilir. Örneğin, uzaktan hasta takip için kullanılan bir kablosuz EKG cihazında şu modüller bulunabilir.

- *ECG* : EKG sinyallerinin toplanması
- *Temperature* : Sıcaklık sensörü

- *Humidity* : Ortam nem sensörü
- *Motion* : Hasta hareketlerinin takibi

Bu özelliklerin hepsi ayrı bir modül olarak gerçekleştirilir. Bu modüllerin sunduğu arayüz, ortak bir modül tanıtım dosyasında tanımlanır. Ayrıca bir uç cihazda, aynı tür modülden birden fazla bulunabilir. Örneğin birden fazla sıcaklık sensörüne erişim sunmak istendiğinde, tek bir sıcaklık sensörü modülü tanımlanıp, bunun birden fazla olduğunu belirtmek mümkündür (bakınız 2.2.6.2.1).

2.2.6.1.1. Modül Tanıtım Dosyası

Modül tanıtım dosyası YAML [31] formatındadır. Modüllerin bir listesinden oluşur. Her modül için 3 özellikler tanımlanabilir.

- *name* : modülün ismi (benzersiz olmalı)
- *attributes* : öz nitelik
- *methods* : yöntemler
- *signal* : sinyaller

name modülün ismidir. Açıklayıcı ve benzersiz olması dışında bir şart yoktur. Modül için otomatik oluşturulan fonksiyonlarda bu isim kullanılır.

attributes modül için tanımlanan öz niteliklerin bir listesidir. Öznitelikler programlamada kullanılan değişkenler gibi düşünülebilir. Öznitelikler genelde sayı türünde olurlar. Okunabilir ve yazılabilirler. Bir modülün konfigürasyonunda veya modülden basit bir bilginin (sıcaklık değeri gibi) alınmasında kullanılırlar.

methods modül için tanımlanan yöntemlerin bir listesidir. Yöntemler, nesne yönelimli programlamadaki tanıma çok benzerdirler. Uzaktan çağrılabilen bir fonksiyon gibi davranırlar. Bir modülün parametreleri ve dönüş değerleri isimleri ve türleri ile tanımlanabilir. Yöntemler genelde modüle veri göndermek, konfigüre etmek için kullanılırlar.

signals modül için tanımlanan sinyallerin bir listesidir. Sinyaller, yöntemlere benzer bir şekilde sadece bir parametre listesi ile tanımlanırlar. Sinyaller, uç birim cihazı tarafından herhangi bir anda bir cevap beklemeksizin gönderilirler. Genelde periyodik olarak toplanan ölçüm sonuçlarını bildirmek için kullanılırlar. Bu mekanizma sayesinde,

merkezin sürekli uç birim cihazından sonuçları okumak için komut göndermesine gerek kalmaz. Prensip olarak, henüz başlatılmamış bir modülün sinyal göndermesi beklenmez.

Modül tanıtım dosyasında bir modülün hangi sırada tanımlandığı önemlidir. Bu sıra modülün genel kimlik numarası (ID) olarak kullanılır.

```
- name: OrnekModul1
  attributes:
    - {name: x , type: Integer}
    - {name: y , type: Integer}
    - {name: z , type: Integer}
  methods:
    - name: baslat
      params:
        - {name: hiz, type: Integer}
      returns:
        - {name: basarili, type: Bool}
    - {name: durdur}
  signals:
    - name: pozisyon
      params:
        - {name: x, type: Integer}
        - {name: y, type: Integer}
        - {name: z, type: Integer}

- name: OrnekModul2
  attributes:
    - {name: isim, type: String}
    - {name: id, type: Byte}

- name: OrnekModul3
  signals:
    - name: olay
```

Şekil 26. Örnek bir modül tanıtım dosyası

Şekil 26'da sadece 3 modülden oluşan örnek bir modül tanıtım dosyası verilmiştir. Ek 1'de projede kullanılan modül tanıtım dosyasının tamamı verilmiştir. Bu dosyada 3 farklı modül tanımlanmıştır. Bu modüllerin kimlik numaraları şu şekilde olur:

- OrnekModul1 : 0
- OrnekModul2 : 1
- OrnekModul3 : 2

OrnekModul1 3 adet öz niteliğe sahiptir. Bunlar tamsayı (*Integer*) cinsinden olup, isimleri *x*, *y*, *z* dir. Modül kimlik numarasına benzer şekilde her öz nitelik, yöntem ve sinyalinde bir kimlik numarası vardır. Tanımlama sıra numarası kimlik numarası olarak kullanılır. Modülün farklı özelliklerine uzaktan yapılan çağrılarda bu kimlik numaraları kullanılır. Bu modülün ayrıca *baslat* isminde bir yöntemi tanımlanmıştır. *baslat* yöntemi

tek bir parametre kabul etmektedir ve işlemin başarılı olup olmadığını cevap olarak dönmektedir. Ayrıca *durdur* metodu tanımlanmıştır. Görüldüğü gibi bu metod herhangi bir parametre kabul etmez ve cevap parametresi de tanımlanmamıştır.

Dosyanın devamında sadece iki parametresi olan *OrnekModul2*, ve sadece 1 sinyali tanımlanan *OrnekModul3* modülleri tanımlanmıştır. Görüldüğü gibi bu sinyalin herhangi bir parametre taşıması gerekmez. Bu da, sinyalin sadece bir olayın gerçekleştiğini bildirmek için kullanılabileceğini gösterir.

Modül tanıtımında kullanılabilecek değişken türleri şöyledir:

- *Boolean* : doğruluk değeri *true*, *false*
- *Integer* : 32-bit tamsayı
- *String* : metin dizini
- *Byte* : bayt
- *uint8* : 8-bit işaretsiz tamsayı
- *uint16* : 16-bit işaretsiz tamsayı
- *uint32* : 32-bit işaretsiz tamsayı
- *uint64* : 64-bit işaretsiz tamsayı
- *int8* : 8-bit tamsayı
- *int16* : 16-bit tamsayı
- *int32* : 32-bit tamsayı
- *int64* : 64-bit tamsayı
- *float* : 4 bayt ondalık sayı (C tipi)

2.2.6.1.2. Modül Tanıtım Dosyasının İşlenmesi

Modül tanıtım dosyası işlenerek, modül programlama arayüzünü tanımlayan bir şablon otomatik olarak oluşturulur. Bu dosya aslında C fonksiyonlarını içeren bir *header* dosyasıdır. Oluşturulan dosyada bulunacak fonksiyonlar şöyledir.

- *start* ve *stop* fonksiyonları
- öz nitelik *get* ve *set* fonksiyonları
- metot çağırma fonksiyonları

start ve *stop* fonksiyonları her modül için tanımlanır. Bu fonksiyonlar, servis katmanı başlatıldığında ve durdurulduğunda modülün çalışmasını başlatmak ve durdurmak için çağrılır.

get ve *set* fonksiyonları, modülün her öz niteliği için bir çift olmak üzere tanımlanır. *get* fonksiyonu, öz niteliğin değerini okumak için, *set* fonksiyonu, değerini değiştirmek için çağrılır.

Her yöntem için bir işleyici (*handler*) fonksiyon tanımlanır. Bu fonksiyon parametrelerin bulunduğu ve cevabın yazılacağı dizinin adresini fonksiyon parametresi olarak alır. Yöntem parametrelerinin listesini doğru formatta okuyup, cevabı doğru formatta verilen adrese yazmak işleyici fonksiyonun sorumluluğundandır.

Şekil 27'de örnek bir modül tanıtımı ve bu modül için oluşturulan programlama arayüzü Şekil 28'de verilmiştir.

```
- name: OrnekModul
  attributes:
    - {name: nitelik1, type: Integer}
  methods:
    - name: baslat
      params:
          - {name: deger, type: Integer}
```

Şekil 27. Örnek bir modülün arayüz tanıtımı

Oluşturulan başlık dosyasının sadece fonksiyon isimleri verilmiştir. Çalışmada kullanılan sıcaklık modülü için otomatik olarak oluşturulan başlık dosyasının tamamı Ek 2'de bulunabilir.

```
void mtl_start_module_ornekmodul(unsigned mindex);
void mtl_stop_module_ornekmodul(unsigned mindex);
void mtl_module_ornekmodul_get_attr_nitelik1(unsigned mindex,
                                             int* value);
void mtl_module_ornekmodul_set_attr_nitelik1(unsigned mindex,
                                             int value);
void mtl_module_ornekmodul_baslat(
    unsigned mindex,
    char* params, unsigned params_size,
    char* returns, unsigned* returns_size);
```

Şekil 28. Örnek modül için otomatik olarak oluşturulan arayüz

Başlık dosyası oluşturulduktan sonra, bu fonksiyonların geliştirici tarafından gerçeklenmesi gerekmektedir. Bu genelde her modül için bir C dosyası oluşturularak yapılır. Bu C dosyası o modülü içeren cihazın programına eklenmelidir.

2.2.6.2. Dağıtıcı

Dağıtıcının işlevi, servis katmanına gelen istekleri inceleyerek ilgili modüle yönlendirmek ve modülden gelen cevabı da servis katmanına aktarmaktır. Burada yönlendirme işlemi, basitçe modülde ilgili işlevi gerçekleyen C fonksiyonunun çağrılmasıdır. Çağrılan fonksiyon parametreleri okur, bu parametrelere göre gerekli işlemleri gerçekleştirir, cevabı (*returns*) yine ilgili diziye yerleştirir.

2.2.6.2.1. Cihaz Tanıtım Dosyası

Modüller birbirlerinden bağımsız olarak gerçekleştirilse de, dağıtıcı gerçekleştirilen uç birim cihazına özel olarak oluşturulur. Zira gerçekleştirilen cihazda bütün modüller bulunmayabilir. Cihazda bulunan modüllerin listesi cihaz tanıtım dosyasını oluşturur. Bu dosya YAML formatında hazırlanır ve basitçe, cihaz tip ismi ve içerdiği modüllerin isimlerini listeler. Şekil 29'da örnek bir cihaz tanıtım dosyası verilmiştir.

```
- name: Cihaz1
  modules:
    - name: Modul1
    - name: Modul2
    - name: Modul3
- name: Cihaz2
  modules:
    - name: Modul3
      number: 2
```

Şekil 29. Örnek bir cihaz tanıtım dosyası

Bu örnek dosyada, sadece iki cihaz tanımlanmıştır, "Cihaz1" ve "Cihaz2". Cihaz1, 3 modül, Cihaz2 tek bir modül türünden 2 adet içermektedir. Bu modüllerin tanıtımının ayrı bir modül tanıtım dosyasında (bkz. 2.2.6.1.1) verilmesi gerekir.

2.2.6.2.2. Dağıtıcının Yapısı

Dağıtıcı, her cihaz tanımı için bir fonksiyon olarak gerçekleştirilir. Bu fonksiyonun ismi cihazdan bağımsız olarak *mtl_dispatch_handler*'dır. Ancak her cihazın dağıtıcı fonksiyonu ayrı bir C dosyasında gerçeklenmiştir. Bu dosya dağıtıcı oluşturulurken otomatik olarak oluşturulur. Dağıtıcı fonksiyonun kabul ettiği parametreleri tanımlayan fonksiyon prototipi Şekil 30'da verilmiştir.

```
MTL_STATUS mtl_dispatch_handler(
    unsigned module,
    unsigned mindex,
    unsigned method,
    char* params,
    unsigned params_size,
    char* returns,
    unsigned* returns_size)
```

Şekil 30. Dağıtıcı fonksiyonun prototipi

Bu fonksiyonun parametrelerinin işlevi şöyledir:

- *module* : ilgili modülün kimlik numarası
- *mindex* : ilgili modülün sıra numarası (birden fazla modül yoksa hep 0)
- *method* : ilgili yöntemin kimlik numarası
- *params* : metod parametrelerinin bulunduğu bellek adresi
- *params_size* : metod parametrelerinin toplam uzunluğu
- *returns* : metod cevabının yazılacağı bellek adresi
- *returns_size* : metod cevabının toplam uzunluğu

Fonksiyon, hiyerarşik bir şekilde iç içe geçmiş *switch case* ifadeleriyle oluşturulmuştur. Bu kodun basitleştirilmiş bir hali Şekil 31'de verilmiştir.

```
switch (module) // modül ID'si
{
    case MODULE_ID_X:
        switch (method) // method ID'si
        {
            case MODULE_X_METHOD_A:
                mtl_module_x_a(...) // işleyici fonksiyonu çağır
                return;
        }
}
```

Şekil 31. Dağıtıcının kodunun basitleştirilmiş bir gösterimi

2.2.6.3. Servis Katmanı Yönetimi

Servis katmanının başlatılması, durdurulması, erişim katmanından gelen paketlerin ilk kontrollerinin yapılması bu katmanda gerçekleşir. Kontrol edilen paketler, dağıtıcı fonksiyonun çağrılmasıyla ilgili servis modülüne yönlendirilir. Servis modülünden gelen cevap yine bir pakete yerleştirilerek erişim katmanına aktarılır. Yönetim katmanı, servis katmanının uygulamanın diğer kısımları tarafından görülen programlama arayüzünü gerçekleştirir.

2.2.6.3.1. Servis Katmanı Paketleri

Servis katmanının işleyişi için 3 paket çeşidi kullanılır.

1. İstek paketi (*request*)
2. Cevap paket (*response*)
3. Sinyal Paketi (*signal*)

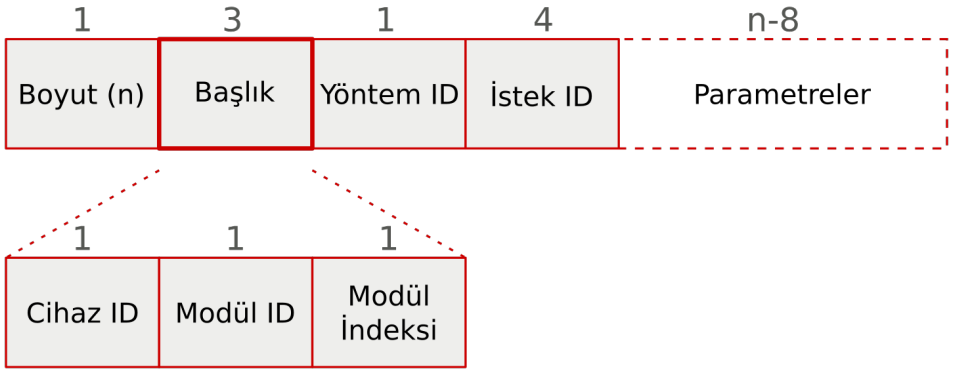
İstek paketleri, bilgisayar tarafından oluşturulup, uç birim cihazındaki fonksiyonları yönetmek için gönderilir. Cevap paketi, istek paketine cevap olarak gönderilen pakettir. Her istek paketine karşılık, mutlaka bir cevap paketi gönderilir. Sinyal paketleri modüller tarafından herhangi bir anda gönderilebilirler.

Servis katmanı paketleri, erişim katmanında veri türünde bir paketin, paket gövdesi kısmı (bkz. Şekil 6) olarak taşınırlar.

a. İstek Paketi

İstek paketleri, bir servis modülünün herhangi bir yöntemine çağrıda bulunmak için gönderilir. İstek paketinin yapısını gösteren bir diyagram Şekil 32'de verilmiştir. *Cihaz ID*,

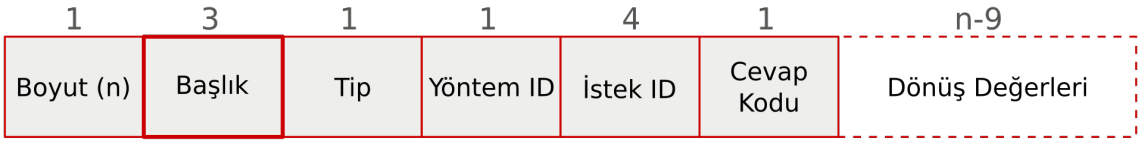
isteğin gönderildiği cihazın bağlantı kimlik numarasıdır. *Modül ID* ilgili modülün kimlik numarasıdır. *Modül İndeksi*, kaçıncı modüle erişilmek istendiğini belirtir. Genelde bir modülden sadece tek bir tane bulunduğu için bu indeks 0 olur. *Yöntem ID*, erişilmek istenen yöntemin kimlik numarasıdır. *İstek ID*, istek paketlerini birbirlerinden ayırt etmek için kullanılan bir sıra numarasıdır. Bir İstek paketine gönderilen cevapta aynı numaranın bulunması gerekmektedir. Bu bilgilerin ardından ilgili metodun parametreleri tanımlandıkları sırada gelirler. Bu paketi okuyan modül işleyici fonksiyonunun parametre yapısına göre gerçekleştirilmiş olması gerekmektedir.



Şekil 32. Servis katmanı istek paketinin yapısı

b. Cevap Paketi

İstek paketine, cevap olarak gönderilen pakettir. Yöntemin döneceği verileri taşır. Şekil 33'te, bir istek paketinin yapısı verilmiştir. Başlık kısmı istek1 paketi ile aynıdır (bakınız Şekil 32).



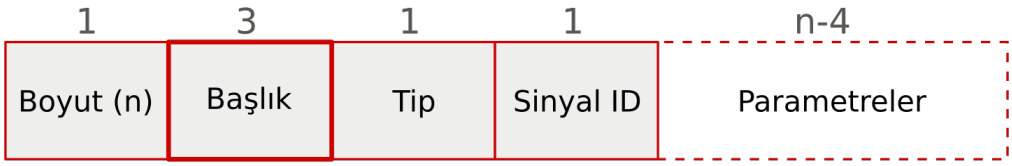
Şekil 33. İstek paketine cevap olarak gönderilen paketin yapısı

Tip, cevap paketlerini sinyal paketlerinden ayırt etmek için kullanılır. Değeri her zaman 1'dir. *Yöntem ID*, cevabın gönderildiği yöntemin kimlik numarasıdır. *İstek ID*, bu cevabın gönderildiği pakette gelen *İstek ID* numarasının aynısıdır. Servis katmanı sadece

aynı numarayı kopyalar herhangi bir şekilde inceleme yapmaz. Her istekte farklı bir numaranın kullanılması isteğin oluşturulduğu bilgisayar sürücüsünün sorumluluğundadır. *Cevap Kodu*, yöntemin başarılı bir şekilde çalıştırıldığını veya bir hata oluştuğunu gösteren bir koddur. Her yöntem için bir cevap kodu gönderilecektir. Paketin kalan kısmına, istek paketine benzer şekilde, dönüş değerleri sıralı olarak yerleştirilir.

c. Sinyal Paketi

Servis modüllerinden gönderilen sinyalleri taşımak için oluşturulan paket yapısıdır. Şekil 34'te paketin yapısını gösteren bir şekil verilmiştir. Bu paket de diğerlerine benzer bir başlık yapısındadır. *Tip*, sinyal paketini cevap paketinden ayırt etmek için kullanılır. Değeri her zaman 2'dir. *Sinyal ID*, gönderilen sinyalin kimlik numarasıdır. *Parametreler* kısmında, sinyalin taşıdığı veriler sıralı olarak yerleştirilmiştir.



Şekil 34. Servis modüllerinden gönderilen sinyal paketinin yapısı

2.2.6.3.2. Programlama Arayüzü

mtl_start(unsigned id) fonksiyonu, servis katmanını başlatır. Bağlantı sağlandıktan sonra çağrılması gerekir. Kablolu bir cihaz için bağlantının sağlanması, bilgisayara USB ile bağlanma, kablosuz bir uç birim cihazı için ise, erişim katmanının ağ geçidi ile bağlantıyı sağlamasıdır. *id* parametresi cihaza atanan kimlik numarasıdır. Kablolu bir cihaz için bu her zaman "0"dır. *mtl_stop* fonksiyonu ise, servis katmanını durdurur. Bağlantı kesildikten sonra, temizlik işlemlerini yapmak için çağrılması gerekir. *mtl_id* fonksiyonu, cihaza atanan kimlik numarasını döner. *mtl_start* fonksiyonuna verilen parametre ile aynı değerdir.

mtl_send_signal fonksiyonu, bir sinyal paketi göndermek için çağrılır. Bu fonksiyon sadece modüller tarafından kullanılır. Fonksiyona sağlanması gereken parametreler şöyledir:

1. module_id : modülün ID numarası
2. module_index : modül sıra numarası

3. signal_id : sinyal ID numarası
4. signal_params : sinyal parametrelerinin bulunduğu bir dizi
5. params_size : parametrelerin bayt uzunluğu

2.2.6.4. Özel Modüller

2.2.6.4.1. DeviceInfo

Her cihazda bulunan, cihazın servis katmanıyla ilgili bilgileri sağlayan modüldür. Bu modülün sunduğu en önemli bilgi, cihazda bulunan modüllerin listesidir. Makineden makineye iletişim prensiplerinin gerçekleştirilmesindeki en önemli nokta, otomatik olarak cihazdaki özelliklerin taranmasına imkan sunan bu arayüzdür. Modülün YAML formatındaki tanımı Şekil 35'te verilmiştir.

```
- name: DeviceInfo
  methods:
    - name: type
      returns:
        - {name: type, type: Integer}
    - name: name
      returns:
        - {name: name, type: String}
    - name: id
      returns:
        - {name: id, type: Integer}
    - name: uid
      returns:
        - {name: uid, type: Byte, array: True}
    - name: modules
      returns:
        - {name: modules, type: Integer, array: True}
```

Şekil 35. DeviceInfo modülünün arayüz tanımı

type metodu, cihazın tür kimlik numarasını döner. Bu cihaz tanıtım dosyasındaki sıra numarasıdır. *name* cihazın ismini döner, cihaz tanıtım dosyasında cihaza verilen isimdir. *id* cihaza atanan bağlantı kimlik numarasını döner. Kablolu bir cihaz için bu her zaman "0"dır. Kablosuz bir uç birim cihazı için bu değer, erişim katmanı tarafından atanan bağlantı kimlik numarasıdır. *uid* cihazın eşsiz kimlik numarasını döner. Bu her cihaz için farklı olan kimlik numarasıdır. Bu çalışmada, STM32 işlemcisine gömülü olan eşsiz ID kullanılmıştır.

modules cihazdaki modüllerin listesini döner. Bu liste sırayla modülün kimlik numarasını ve o modülden kaç adet bulunduğunu içerir. Modül listesi *DeviceInfo* modülünü içermez. Çünkü bu modül her cihazda bulunmak zorundadır.

Merkez, uzaktaki bir uç birim cihazına bağlandığında ilk olarak bu modüle erişerek, cihazla ilgili bilgileri almalıdır. Daha sonraki işlemler cihazın içerdiği modüllere göre gerçekleştirilir. Örneğin cihaz EKG modülü içeriyorsa, EKG modülünün çalışması başlatılır, vs.

2.2.6.4.2. Hub

Ağ geçidi cihazının merkez tarafından yönetilmesi için oluşturulan modüldür. Ağ geçidi cihazının, kablo üzerinden bağlanması beklense de modüler yapının bir parçasıdır. *Hub* modülünün tanımı YAML formatında Şekil 36'da verilmiştir.

```
- name: Hub
  attributes:
    - {name: ban_on, type: Boolean}
  methods:
    - name: connectedDevices
      returns:
        - {name: ids, type: Integer, array: True}
    - name: disconnect
      params:
        - {name: id, type: Integer}
  signals:
    - name: deviceConnected
      params:
        - {name: id, type: Integer}
    - name: deviceDisconnected
      params:
        - {name: id, type: Integer}
```

Şekil 36. Ağ geçidi modülünün arayüz tanımı

ban_on niteliği, ağ geçidinin operasyonunu başlatır ve durdurur. *connectedDevices* ağ geçidine bağlı olan uç birim cihazlarının kimlik numaralarını liste olarak döner. *disconnect* bağlı olan bir uç birimin bağlantısını keser. Ağ geçidine bir cihaz bağlandığında, *deviceConnected* sinyali gönderilir. Benzer şekilde bir cihazın bağlantısı kesildiğinde, *deviceDisconnected* sinyali gönderilir.

2.2.7. Sürücü

Bilgisayardan, ağ geçidi cihazına ve uzaktaki uçbirim cihazına erişimi sağlayan programlama arayüzüdür. Sürücü Python programlama dili ile geliştirilmiştir. Sürücünün görevi, USB üzerinden servis katmanına yapılacak çağrıları oluşturup göndermek, gelen servis katmanı paketlerini inceleyip, programda kullanılabilecek bir formata ayrıştırmaktır. Servis katmanı modüllerinin arayüzüne uygun bir programlama arayüzü sunmak için, modül tanıtım dosyalarını kullanarak anında uygun arayüzü oluşturan bir yazılım kütüphanesi geliştirilmiştir. Dinamik programlama arayüzü oluşturması çok kolay olduğu için, Python dili seçilmiştir.

2.2.7.1. Connection Sınıfı

Sürücü kütüphanesinin ana arayüzü bu sınıftır. Bu sınıf, USB-UART üzerinden ağ geçidine direkt bağlantı sağlar. Kablosuz düğümlere olan bağlantı ağ geçidi üzerinden dolaylı olarak sağlanır. Uçbirim cihazının kablolu olarak bilgisayara bağlanması durumunda, direkt olarak uçbirim cihazına bağlantı da mümkündür.

Cihazla bağlantı kurmak için bir *Connection* nesnesi oluşturulması gerekir. Bu işlemin bir örneği Şekil 37'de verilmiştir. Bu örnekte "*dev/ttyACM0*", cihazın bağlı olduğu USB-seri port bağlantısıdır. Cihaza erişmek için *Connection* nesnesinin, *device* metodu kullanılır. Verilen örnekte "0", erişilmek istenen cihazın kimlik numarasıdır. "0" özel bir numara olup, kablo ile bağlanmış cihaza işaret eder. Bu cihaz genelde ağ geçidi cihazı olacaktır. Kablosuz olarak ağ geçidine bağlanmış olan uçbirim düğümlerinin kimlik numaralarının ağ geçidinden sorgulanması gerekir. *dev* objesi *DeviceProxy* (bkz. 2.2.7.2) türündendir ve cihazın servis modüllerine erişim sağlar.

```
con = Connection("/dev/ttyACM0")
dev = con.device(0)
```

Şekil 37. Cihazla bağlantı kurma örneği

2.2.7.2. DeviceProxy Sınıfı

Bu sınıf, cihazın servis modüllerine erişim sunar. Modülün ismi Şekil 38'de gösterildiği gibi kullanılarak bir *ModuleProxy* (bkz. 2.2.7.3) nesnesi elde edilir.

```
hubModule = con.device(0).Hub
temp0 = con.device(0).Temperature[0]
```

Şekil 38. Modül erişim nesnesinin oluşturulması

Şekil 38'de modül seçiminin bir örneği verilmiştir. Burada *hubModule* objesi, "0" kimlik numaralı (ağ geçidi) cihazın, "Hub" servis modülüne erişim sunmaktadır. Eğer cihazda aynı modülden birden fazla bulunduruyorsa, bu teknikle bir listeye erişilir. Liste indekslenerek istenilen modül seçilebilir. Bu örnekte cihazın birinci sıcaklık modülüne (*Temperature*) erişim sunan bir nesne oluşturulmuştur.

2.2.7.3. ModuleProxy Sınıfı

Bir cihazın servis modülü tarafından sunulan arayüze erişim sunar. Bu objenin üyeleri, servis modülünün metotları ve öz nitelikleridir. Bu metot ve öz niteliklere isimleriyle Şekil 39 örneğinde gösterildiği gibi erişilebilir.

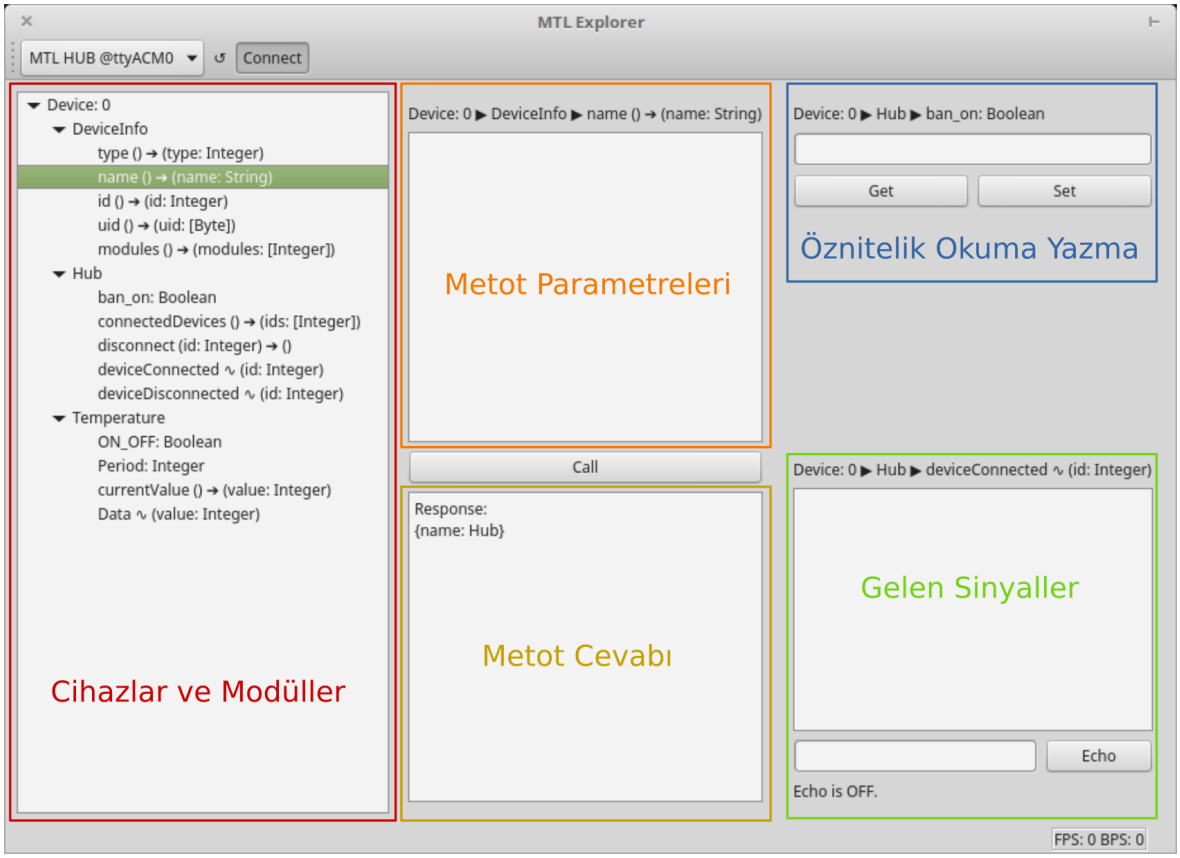
```
module = con.device(0).ModuleName
module.start()
module.setting = 3
```

Şekil 39. Modül metotları ve özniteliklerine erişim

Şekil 39'da *ModuleName* adında bir modülün, *start* metodu çağrılmakta ve *setting* niteliğine 3 yazılmaktadır. Bu iki işlem içinde birer servis katmanı paketi oluşturulacak, bu paket cihaza transfer edilecek, cevaplar işlenip kontrol edilecektir. Görüldüğü gibi bütün bu işlemler programcıdan gizli bir şekilde gerçekleştirilmektedir. Herhangi bir hata oluşması veya cihazdan bir hata kodu dönmesi durumunda bu bir *Exception* oluşturacaktır.

2.2.7.4. Sürücü Grafik Arayüzü

Servis katmanının geliştirilmesi ve test edilmesi için, geliştirilen sürücüyü kullanan bir grafik arayüz geliştirilmiştir. Bu yazılım ile kablo ile bağlı bir cihazın yanı sıra kablosuz olarak uzaktan bağlanan uç birim cihazlarına erişmek, bunların modüllerini listelemek, metotlara çağrı yapmak, öz nitelikleri ayarlamak, gelen sinyalleri görüntülemek mümkündür.

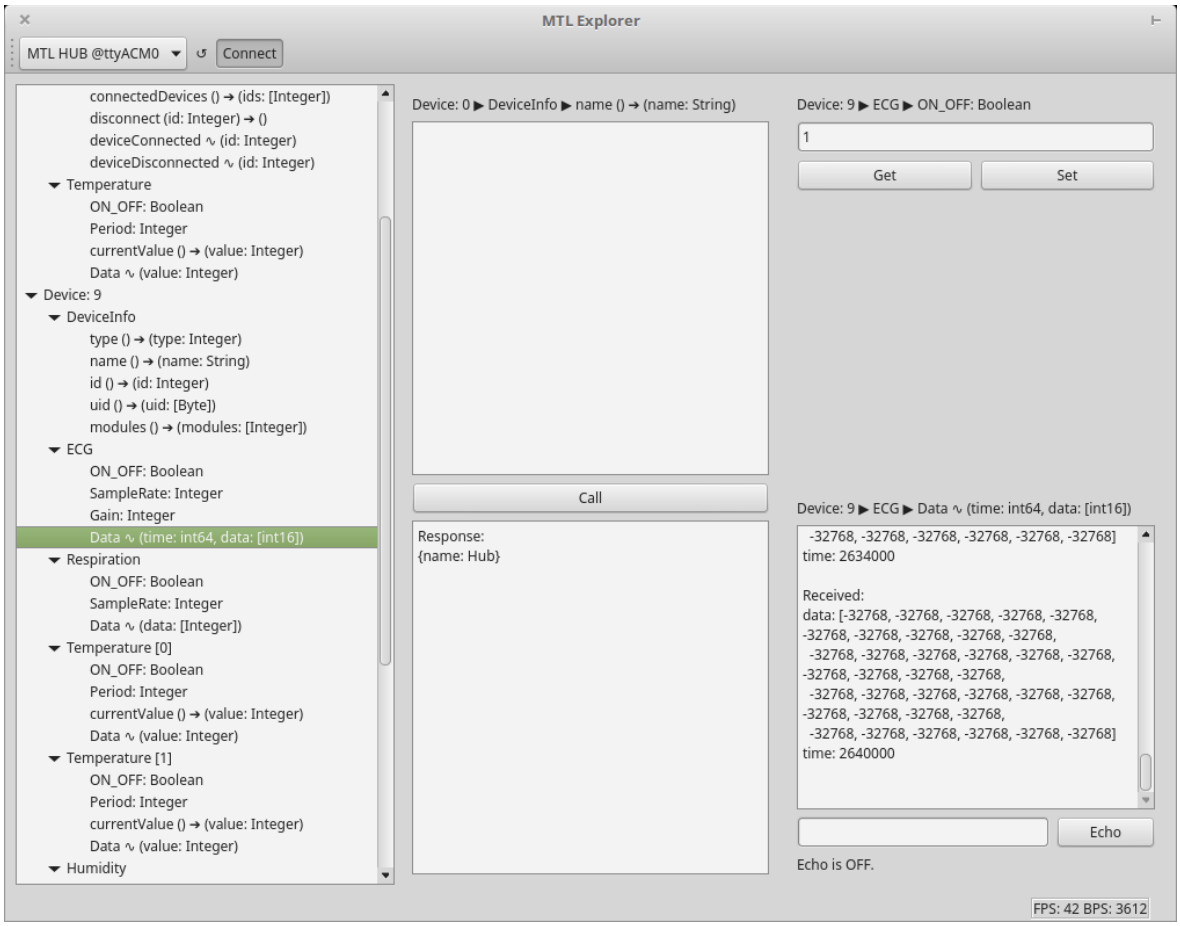


Şekil 40. Servis katmanı keşif arayüzü

Şekil 40'ta geliştirilen yazılımın farklı panellerinin işaretlendiği bir ekran görüntüsü verilmiştir. Cihazlar ve modüller panelinde, bağlanan cihazların bir listesi, cihazın içerdiği modüllerin listesi ve bu modüllerin detayları bulunur. Bu panel kablosuz bir cihazın bağlanmasıyla otomatik olarak güncellenir. Şekilde görüldüğü gibi panelde, modüllerin öz nitelikleri, metotları ve sinyalleri listelenmektedir. Metot parametreleri kısmına seçilen bir metoda gönderilmek istenen parametreler girilmektedir. *Call* düğmesine basılmasıyla metot çağrılır ve cevabı metot cevabı kısmında gösterilir. Öznitelik okuma yazma paneli

kullanılarak, seçilen bir özniteliğin değeri, *Get* düğmesi ile okunur ve *Set* düğmesi ile yazılır. Sinyaller panelinde ise, kablo ile bağlı veya uzaktaki cihazdan gelen sinyal paketleri ve içerikleri anlık olarak gösterilir.

Şekil 41'de uzaktaki bir EKG cihazına bağlanan keşif programının, cihazdan gelen sinyal paketlerini listelemesini gösteren bir ekran görüntüsü verilmiştir. Burada dikkat edilmesi gereken nokta, ana bağlantının ağ geçidi cihazı üzerinden sağlanıyor oluşudur.



Şekil 41. Keşif arayüzünün uzaktaki bir cihazdan gelen paketleri gösterimi

3. BULGULAR

Ağ geçidi cihazının program belleği (Flash) ve ara bellek (RAM) kullanımı Tablo 5'te verilmiştir. Haberleşme modülü üzerinde gerçekleştirilen bir kablosuz EKG uçbirim cihazının bellek kullanım değerleri Tablo 6'da gösterilmiştir. Bu cihazda, EKG'nin yanı sıra sıcaklık sensörü ve hareket sensörü servis modülleri de gerçekleştirilmiştir.

Tablo 5. Ağ geçidi cihazı bellek kullanımı

	Flash (bayt)	**RAM (bayt)**
Erişim Katmanı	5956	17224
Fiziksel Katman Arayüzü	2132	7
Servis Katmanı	2868	4745
Uygulama (*main.c*)	604	520
HAL	16344	2590
ChibiOs	6772	376
Diğer	12252	953
Toplam	46928	26415

Tablo 6. Uçbirim cihazı bellek kullanımı

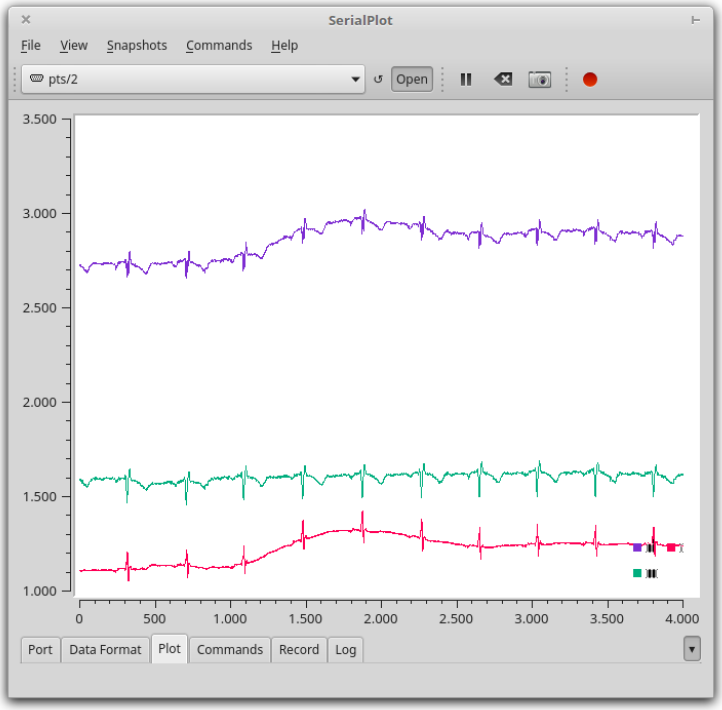
	Flash (bayt)	**RAM (bayt)**
Erişim Katmanı	5424	19706
Fiziksel Katman Arayüzü	2132	7
Servis Katmanı	6496	2934
Uygulama (*main.c*)	1556	851
HAL	18516	2698
ChibiOs	9088	384
Diğer	33724	138
Toplam	76936	26718

Kablosuz EKG cihazının ortalama akım tüketim bilgileri Tablo 7'de verilmiştir. Aktif değerler, 250sps 16 bit EKG sinyalinin sıkıştırılmadan, maksimum kablosuz iletim

gücünde aktarılması için verilen değerlerdir. Sistemin beklemedeki yüksek akım tüketimi kullanılan STM32F4 işlemcisinin, nispeten performans odaklı bir işlemci olmasından kaynaklanmaktadır. Uç birim cihazı için düşük güçlü, ARM Cortex M0 veya MSP430 ailesinden bir mikrodenetleyicisinin kullanılmasının daha uygun olacağı öngörülmektedir. Yine de sistemin 1000mAh'lik nispeten küçük bir lityum batarya ile 24 saatten daha uzun süre çalışabileceği tespit edilmiştir. Şekil 42'de aktarılan sinyal görülmektedir.

Tablo 7. Kablosuz EKG cihazının akım tüketimi

	Beklemede	Aktif
Alıcı Verici	0 ma	6,5 ma
EKG Sensörü	0 ma	4 ma
Diğer	24 ma	26 ma
Sistem Toplamı	24 ma	36,5 ma



Şekil 42. Kablosuz olarak aktarılan 3 kanal EKG sinyalinin görüntüsü

4. SONUÇLAR VE ÖNERİLER

Bu çalışmada, IEEE 802.15.6 standardı tarafından tanımlanan ortam erişimi kontrol katmanını temel alan bir vücut alan ağı haberleşme sistemi, böyle bir sistemin çalışması için gerekli tüm donanım ve yazılım komponentleriyle gerçekleştirilmiştir. Geliştirilen sistemde 802.15.6 standardı ortam erişim kontrol katmanının zaman dilimlemeli modları kullanılmıştır. Birden fazla uçbirim düğümüne aynı anda bağlanıp, zaman dilimi tahsisiyle yönetebilen, bir ağ geçidi cihazının donanım ve yazılımı geliştirilmiştir. Sistem zaman çoklamalı erişim tekniklerinin yanı sıra rastgele erişim ve çekişmeli erişim tekniklerini de başarıyla uygulayabilmektedir. Fiziksel katman için, 802.15.6 dar bant uyumlu bir ticari alıcı vericinin entegresinin yokluğundan ötürü, AT86RF233 802.15.4 alıcı verici entegresi, 802.15.6 ortam erişim parametrelerine mümkün olduğunca uyularak kullanılmıştır.

Ortam erişim kontrol katmanının yanı sıra, nesnelerin internetinde (IoT) en önemli konulardan biri olan Makineden Makineye İletişim (M2M) prensipleri hedeflenerek, kablosuz sensörlerin geliştirilmesini kolaylaştıracak bir servis katmanı ve bu sistemin bilgisayar sürücüsü geliştirilmiştir. Modüler yapıya sahip bu servis katmanında, yeni yazılımsal özellikleri, sistemde mevcut olan özellikler ile çakışmadan kolaylıkla eklemek ve sisteme tanıtmak mümkündür. Servis katmanının kendisini tanıtabilme özelliği sistemin taşınabilirliğini arttırmaktadır.

Bu çalışmanın devamı olarak, geliştirilen sistemin güvenilirlik ve enerji tüketimi testlerinin gerçek şartları temsil eden ortamlarda yapılması gerekmektedir. Enerji verimliliğini arttırmak için IEEE 802.15.6 ortam erişim kontrol katmanının zaman dilimlemeli modu kullanılmıştır. Ancak bu modun parametrelerinin optimize edilmesi için detaylı deneysel çalışmaların yapılması gerekmektedir. Dahası, ağa bağlanan düğümlerin ihtiyaçlarına ve davranışlarına göre kendisini adapte edebilen bir zaman dilimi yapısı da araştırılabilir. Servis katmanının, benzer bir sistem olan Bluetooth GATT Profili ile karşılaştırıldığında bir eksik yönü, özgün kimlik numaralarının yoksunluğudur. Modül kimlik numaraları olarak sıra numarası yerine, UUID kullanımı sistemin taşınabilirlik yönünü güçlendirecektir. Ayrıca servis katmanının paket yapısının basitleştirilerek optimize edilmesi sistemin efektif veri hızını arttıracaktır.

5. KAYNAKLAR

1. IEEE Standard for Local and metropolitan area networks - Part 15.6: Wireless Body Area Networks, IEEE Std 802.15.6-2012, (2012) 1–271.

2. https://www.fcc.gov/document/fcc-dedicates-spectrum-enabling-medical-body-area-networks FCC Dedicates Spectrum Enabling Medical Body Area Networks | Federal Communications Commission. 23 Aralık 2016.

3. Cavallari, R., Martelli, F., Rosini, R., Buratti, C., ve Verdone, R., A Survey on Wireless Body Area Networks: Technologies and Design Challenges, IEEE Communications Surveys Tutorials, 16,3 (2014) 1635–1657.

4. Ullah, S., Shen, B., Islam, S.M.R., Khan, P., Saleem, S., ve Kwak, K.S., A Study of MAC Protocols for WBANs, Sensors (Basel, Switzerland), 10,1 (2009) 128–145.

5. Kritsotaki, A., Perakis, K., ve Koutsouris, D., Evaluation of 802.15.4 WPAN for patient monitoring, 10th IEEE International Conference on Information Technology and Applications in Biomedicine, Kasım 2010, Corfu, 1–4.

6. Mitko Petrov Shopov, Galidiya Ivanova Petrova, ve Grisha Valentinov Spasov, Evaluation of Zigbee-based Body Sensor Networks, ANNUAL JOURNAL OF ELECTRONICS, (2011).

7. Simunic, D., Tomac, S., ve Vrdoljak, I., Wireless ECG monitoring system, 2009 1st International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace Electronic Systems Technology, Mayıs 2009, Aalborg, Denmark, 73–76.

8. Jhuang, J.-W. ve Ma, H.-P., A patch-sized wearable ECG/respiration recording platform with DSP capability, 2015 17th International Conference on E-health Networking, Application Services (HealthCom), Ekim 2015, Dalian, China, 298–304.

9. Chan, C.C., Chou, W.C., Chen, C.W., Ho, Y.L., Lin, Y.H., ve Ma, H.P., Energy efficient diagnostic grade mobile ECG monitoring, 10th IEEE International NEWCAS Conference, Haziran 2012, Montréal, Canada, 153–156.

10. Wang, X., Design of ECG acquisition system based on Bluetooth wireless communication, 2014 IEEE 5th International Conference on Software Engineering and Service Science, Haziran 2014, Beijing, China, 1019–1022.

11. Zainal, N.I., Rodzi, M.Z.M., Khan, S., Habaebi, M.H., ve Gunawan, T.S., Design and development of wireless PPG data acquisition for health monitoring application using Bluetooth module, 2016 IEEE Student Conference on Research and Development (SCOReD), Aralık 2016, Putrajaya, Malaysia, 1–6.

12. Babusiak, B. ve Borik, S., Low energy wireless communication for medical devices, 2015 38th International Conference on Telecommunications and Signal Processing (TSP), Temmuz 2015, Prague, Czech Republic, 444–447.

13. Harvey, P., Woodward, B., Datta, S., ve Mulvaney, D., Data acquisition in a wireless diabetic and cardiac monitoring system, 2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Ağustos 2011, Boston, MA, USA, 3154–3157.

14. Tseng, H., Wu, R., ve Wu, Y., An Efficient Cross-Layer Reliable Retransmission Scheme for the Human Body Shadowing in IEEE 802.15.6-Based Wireless Body Area Networks, IEEE Sensors Journal, PP,99 (2016) 1–1.

15. Gu, N., Jiang, Y., Zhang, J., ve Zheng, H.-T., An Implementation of WBAN Module Based on NS-2, 2013 International Conference on Computer Sciences and Applications (CSA), Aralık 2013, Wuhan, China, 114–118.

16. Al-Mazroa, A. ve Rikli, N.-E., Performance evaluation of IEEE 802.15.6 MAC in WBANs - A case study, 2014 International Conference on Information Technology and Multimedia (ICIMU), Kasım 2014, Dubai, UAE, 66–71.

17. Choi, J. ve Kim, J.G., An energy efficient MAC protocol for WBAN through flexible frame structure, 2013 Fifth International Conference on Ubiquitous and Future Networks (ICUFN), Temmuz 2013, Da Nang, Vietnam, 476–480.

18. Shakshuki, E.M., Lu, Z., Khan, Z., ve Iqbal, M.A., A New Coexistence Mechanism for Baseline BAN MAC (802.15.6) of Body Area Networks, Procedia Computer Science, 19 (2013) 950–955.

19. Jacob, A.K., Kishore, G.S., ve Jacob, L., Contention versus polling access in IEEE 802.15.6: Delay and lifetime analysis, 2015 Twenty First National Conference on Communications (NCC), Şubat 2015, Mumbai, 1–6.

20. Martelli, F., Buratti, C., ve Verdone, R., On the performance of an IEEE 802.15.6 Wireless Body Area Network, 17th European Wireless 2011 - Sustainable Wireless Technologies, Nisan 2011, Vienna, Austria, 1–6.

21. Azhari, M.E., Toumanari, A., ve Latif, R., Performance analysis of IEEE 802.15.6 and IEEE 802.15.4 for wireless body sensor networks, 2014 International Conference on Multimedia Computing and Systems (ICMCS), Nisan 2014, Marrakesh, Morocco, 910–915.

22. Rahman, M., Elbadry, M., ve Harjani, R., An IEEE 802.15.6 Standard Compliant 2.5 nJ/Bit Multiband WBAN Transmitter Using Phase Multiplexing and Injection Locking, IEEE Journal of Solid-State Circuits, 50,5 (2015) 1126–1136.

23. Kopta, V., Thirunarayanan, R., Pengg, F., Le Roux, E., ve Enz, C., A 2.4-GHz low complexity polar transmitter using dynamic biasing for IEEE 802.15.6, 2015 IEEE International Symposium on Circuits and Systems (ISCAS), Mayıs 2015, Lisbon, Portugal, 1686–1689.

24. Mathew, P., Augustine, L., Kushwaha, D., Desalphine, V., ve Selvakumar, A.D., Implementation of NB PHY transceiver of IEEE 802.15.6 WBAN on FPGA, 2015 International Conference on VLSI Systems, Architecture, Technology and Applications (VLSI-SATA), Ocak 2015, Bangalore, India, 1–6.

25. http://www.st.com/resource/en/datasheet/stm32f407vg.pdf STM32F407 Datasheet. 26 Mayıs 2017.

26. http://www.atmel.com/images/atmel-42192-wireless-zigbit-atzb-rf-233-1-c_datasheet.pdf ATZB-RF-233-1-C ZigBit 2.4GHz Wireless Module Datasheet. 28 Mayıs 2017.

27. http://www.microchip.com/wwwproducts/en/AT86RF233 AT86RF233 - Wireless Modules. 28 Mayıs 2017.

28. https://www.arm.com/products/processors/cortex-m/cortex-microcontroller-software-interface-standard.php CMSIS - Cortex Microcontroller Software Interface Standard - ARM. 28 Mayıs 2017.

29. http://www.st.com/en/embedded-software/stm32cubef4.html STM32CubeF4 - Embedded software for STM32F4 series (HAL, Low-Layer APIs and CMSIS (CORE, DSP, RTOS), USB, TCP/IP, File system, RTOS, Graphic - coming with examples running on ST boards: STM32 Nucleo, Discovery kits and Evaluation boards) - STMicroelectronics - STMicroelectronics. 28 Mayıs 2017.

30. http://chibios.org/ ChibiOS free embedded RTOS - ChibiOS Homepage. 29 Mayıs 2017.

31. http://www.yaml.org/ The Official YAML Web Site. 26 Mayıs 2017.

6. EKLER

Ek 1. modules.yaml dosyasının tamamı

```
- name: DeviceInfo
  methods:
    - name: type
      returns:
        - {name: type, type: Integer}
    - name: name
      returns:
        - {name: name, type: String}
    - name: id
      returns:
        - {name: id, type: Integer}
    - name: uid
      returns:
        - {name: uid, type: Byte, array: True}
    - name: modules
      returns:
        - {name: modules, type: Integer, array: True}

- name: Hub
  attributes:
    - {name: ban_on, type: Boolean}
  methods:
    - name: connectedDevices
      returns:
        - {name: ids, type: Integer, array: True}
    - name: disconnect
      params:
        - {name: id, type: Integer}
  signals:
    - name: deviceConnected
      params:
        - {name: id, type: Integer}
    - name: deviceDisconnected
      params:
        - {name: id, type: Integer}

- name: ECG
  attributes:
    - {name: ON_OFF     , type: Boolean}
    - {name: SampleRate , type: Integer}
    - {name: Gain       , type: Integer}
  signals:
    - name: Data
      params:
        - {name: time, type: int64}
        - {name: data, type: int16, array: True}

- name: Respiration
  attributes:
    - {name: ON_OFF     , type: Boolean}
- {name: SampleRate , type: Integer}
```

EK 1'in devamı

```
  signals:
    - name: Data
      params:
        - {name: data, type: Integer, array: True}

- name: Temperature
  attributes:
    - {name: ON_OFF, type: Boolean}
    - {name: Period, type: Integer}
  methods:
    - name: currentValue
      returns:
        - {name: value, type: Integer}
  signals:
    - name: Data
      params:
        - {name: value, type: Integer}

- name: Humidity
  attributes:
    - {name: ON_OFF, type: Boolean}
    - {name: Period, type: Integer}
  methods:
    - name: currentValue
      returns:
        - {name: value, type: Integer}
  signals:
    - name: Data
      params:
        - {name: value, type: Integer}

- name: Motion
  attributes:
    - {name: ON_OFF         , type: Boolean}
    - {name: SampleRate     , type: Integer}
    - {name: AlarmThreshold , type: Integer}
  methods:
    - name: currentValue
      returns:
        - {name: x, type: Integer}
        - {name: y, type: Integer}
        - {name: z, type: Integer}
  signals:
    - name: Data
      params:
        - {name: x, type: Integer}
        - {name: y, type: Integer}
        - {name: z, type: Integer}

- name: Battery
  attributes:
    - {name: AlarmThreshold, type: Integer}
  methods:
    - name: status
      returns:
- {name: percentage, type: Integer}
```

EK 1'in devamı

```
        - {name: voltage, type: float}
        - {name: charging, type: Boolean}
```

```
  signals:
    - name: Alarm
      params:
        - {name: percentage, type: Integer}

- name: RTC
  attributes:
    - {name: time, type: int64}

- name: Scale
  methods:
    - name: currentValue
      returns:
        - {name: value, type: Integer}
  signals:
    - name: measurement
      params:
        - {name: value, type: Integer}
```

Ek 2. Örnek Sıcaklık Modülü Arayüzü

```
// AUTO GENERATED handler declarations for 'Temperature'

#ifndef __MTL_MODULE_TEMPERATURE_HANDLERS__
#define __MTL_MODULE_TEMPERATURE_HANDLERS__

#include <stdbool.h>
#include <stdint.h>

#include "mtl_status.h"

void mtl_start_module_temperature(void);

void mtl_stop_module_temperature(void);

/**
 * `Temperature` module `ON_OFF` attribute get handler.
 */
MTL_STATUS  mtl_module_temperature_get_attr_on_off(unsigned  mindex,
bool* value);

/**
 * `Temperature` module `ON_OFF` attribute set handler.
 */
MTL_STATUS  mtl_module_temperature_set_attr_on_off(unsigned  mindex,
bool value);

/**
 * `Temperature` module `Period` attribute get handler.
 */
MTL_STATUS  mtl_module_temperature_get_attr_period(unsigned  mindex,
int* value);

/**
 * `Temperature` module `Period` attribute set handler.
 */
```

EK 2'nin devamı

```
MTL_STATUS  mtl_module_temperature_set_attr_period(unsigned  mindex,
int value);


/**
 * `Temperature` module `currentValue` method handler.
 */
MTL_STATUS    mtl_module_temperature_currentvalue(unsigned    mindex,
char*  params,  unsigned  params_size,  char*  returns,  unsigned*
returns_size);


#endif // __MTL_MODULE_TEMPERATURE_HANDLERS__
```

ÖZGEÇMİŞ

Hasan Yavuz ÖZDERYA, 1990 yılında Trabzon'un Araklı ilçesinde doğdu. Lise öğrenimini Araklı Anadolu Öğretmen Lisesinde tamamladı. 2008 yılında Zonguldak Karaelmas Üniversitesi Elektrik-Elektronik Mühendisliği bölümünü kazandı. 1 yıl Zonguldak'ta eğitim gördükten sonra, Trabzon Karadeniz Teknik Üniversitesi'ne yatay geçiş yaptı. 2012 yılında Karadeniz Teknik Üniversitesi Fen Bilimleri Enstitüsü Elektronik Ana Bilim Dalı alanında yüksek lisans eğitimine başladı. 2013-2014 yılları arasında, 1 yıl Cypress Semiconductor (Gebze) şirketinde Gömülü Sistem Yazılımcısı olarak çalıştı. 2016 yılından itibaren Altu Teknoloji şirketinde AR-GE Mühendisi olarak çalışmaktadır. İngilizce bilmektedir.