



**İSTANBUL ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ**

YÜKSEK LİSANS TEZİ

**AKILLI ELEKTRİK ŞEBEKELERİNDE SIP
PROTOKOLÜNÜN KULLANIMI**

Gökhan OKYAY

Elektrik-Elektronik Mühendisliği Anabilim Dalı

Danışman

Prof.Dr. Aydın Akan

Haziran, 2012

İSTANBUL



**İSTANBUL ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ**

YÜKSEK LİSANS TEZİ

**AKILLI ELEKTRİK ŞEBEKELERİNDE SIP
PROTOKOLÜNÜN KULLANIMI**

Gökhan OKYAY

Elektrik-Elektronik Mühendisliği Anabilim Dalı

Danışman

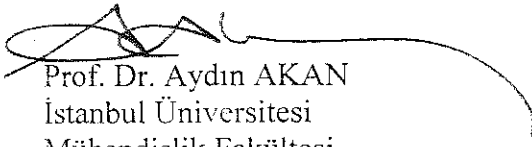
Prof.Dr. Aydın Akan

Haziran, 2012

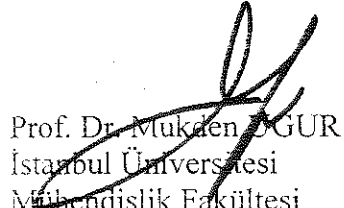
İSTANBUL

2601080067 Öğrenci numaralı Gökhan OKYAY tarafından hazırlanan bu çalışma 25/06/2012 tarihinde aşağıdaki jüri tarafından Elektrik-Elektronik Mühendisliği Anabilim Dalı Elektrik-Elektronik Mühendisliği programında Yüksek Lisans Tezi olarak kabul edilmiştir.

Tez Jürisi


Prof. Dr. Aydın AKAN
İstanbul Üniversitesi
Mühendislik Fakültesi


Prof. Dr. İlhan KOCAARSLAN
İstanbul Üniversitesi
Mühendislik Fakültesi


Prof. Dr. Mukden DOĞUR
İstanbul Üniversitesi
Mühendislik Fakültesi


Doç. Dr. Hakan DOĞAN
İstanbul Üniversitesi
Mühendislik Fakültesi


Yrd. Doç. Dr. Yalçın ÇEKİÇ
Bahçeşehir Üniversitesi
Mühendislik Fakültesi

ÖNSÖZ

Lisans ve yüksek lisans öğrenimim sırasında ve tez çalışmalarım boyunca gösterdiği her türlü destek ve yardımdan dolayı çok değerli hocam Prof.Dr.Aydın Akan'a en içten dileklerimle teşekkür ederim.

Bu çalışma boyunca desteklerini esirgemeyen çalışma arkadaşlarıma ve hayatımın her aşamasında maddi ve manevi beni destekleyen aileme teşekkürü borç bilirim.

Mayıs, 2012

Gökhan Okyay

İÇİNDEKİLER

ÖNSÖZ	i
İÇİNDEKİLER	ii
KISALTMALAR	v
ŞEKİL LİSTESİ	vii
ÖZET	ix
SUMMARY	x
1. GİRİŞ	1
1.1. AMAÇ	5
2. GENEL KISIMLAR	6
2.1. SIP	6
2.1.1. İstek Metotları	10
2.1.1.1 Invite	10
2.1.1.2 Register	11
2.1.1.3 Bye	12
2.1.1.4 Ack	13
2.1.1.5 Cancel	14
2.1.1.6 Options	15
2.1.1.7 Subscribe	16
2.1.1.8 Notify	18
2.1.1.9 Message	19
2.1.2. Yanıtlar (Cevaplar)	21
2.2. AKILLI ŞEBEKE (SMART GRID)	22
2.2.1. Akıllı Şebeke Konsept Referans Modeli	24
2.2.1.1 Pazar	25
2.2.1.2 Servis Sağlayıcı	26

2.2.1.3 Operasyon.....	26
2.2.1.4 Toplu Üretim.....	26
2.2.1.5 İletim.....	27
2.2.1.6 Dağıtım.....	27
2.2.1.7 Müşteri.....	27
2.3. AKILLI ŞEBEKE’DE OTURUM BAŞLATMA PROTOKOLÜNÜN KULLANIMI	28
2.3.1. Ev/Bina Otomasyonu	28
2.3.2. Gelişmiş Sayaç Altyapısı.....	31
2.3.3. Phev	32
2.3.1. Otomatik Talep-Cevap	34
3. MALZEME VE YÖNTEM	36
3.1. SIP TABANLI UZAKTAN EV OTOMASYONU VE OLAY/DURUM BİLDİRİMİ YAZILIMI	36
3.1.1. Sistem Mimarisi.....	37
3.1.2. Yazılım Mimarisi.....	39
3.1.2.1 İstemci.....	40
3.1.2.2 Ev Otomasyon Sunucusu.....	42
3.1.3. SIP Sinyalleşmesi Mimarisi.....	43
3.2. SIP-ZIGBEE İŞBİRLİĞİYLE UZAKTAN EV OTOMASYONU VE OLAY/DURUM BİLDİRİMİ	48
3.2.1. Sistem Mimarisi.....	48
3.2.2. Yazılım Mimarisi.....	49
4. BULGULAR	51
4.1. UZAKTAN EV OTOMASYONU SİSTEMİNİN SİMÜLE EDİLMESİ	51
4.1.1. İstemci ve Ev Otomasyon Sunucusunun Vekil Sunucuya Kaydı	51
4.1.2. Ev Otomasyon Sunucusunun Sorgulanması.....	53
4.1.3. Komutların Uygulanması	56
4.1.4. Olay/Durum Bildirimi	59
4.2. UZAKTAN EV OTOMASYONUNUN SIP-ZIGBEE İŞBİRLİĞİYLE GERÇEKLENMESİ	63
4.2.1. Ev Otomasyonu Sunucusunun Sorgulanması	63
4.2.2. Komutların Uygulanması	64

4.2.3. Olay/Durum Bildirimi	65
5. TARTIŞMA VE SONUÇ	67
KAYNAKLAR	69
EKLER	73
ÖZGEÇMİŞ	124

KISALTMALAR

AMI	: Advanced Metering Infrastructure
AMR	: Automatic Meter Reading
ANSI	: American National Standards Institute
API	: Application Programming Interface
CEN	: European Committee for Standardization
COSEM	: Companion Specification for Energy Metering
DLMS	: Device Language Message Specification
DNS	: Domain Name System
DoS	: Denial of Service
EPRI	: Electric Power Research Institute
ETSI	: European Telecommunication Standards Institute
EV-VPP	: Electric Vehicle Virtual Power Plant
FQDN	: Fully Qualified Domain Name
HAN	: Home Area Network
HTTP	: Hypertext Transfer Protocol
IANA	: Internet Assigned Numbers Authority
IEC	: International Electrotechnical Commission
IEEE	: Institute of Electrical and Electronics Engineers
IETF	: Internet Engineering Task Force
IMS	: IP Multimedia Subsystem
IP	: Internet Protocol
IPv6	: Internet Protocol version 6
ISUP	: ISDN User Part
ITU	: International Telecommunication Union
LAN	: Local Area Network
LED	: Light Emitting Diode
MIME	: Multipurpose Internet Mail Extensions
NAT	: Network Address Translation
NIST	: National Institute of Standards and Technology
OSI	: Open Systems Interconnection
PHEV	: Plug-in Hybrid Electric Vehicle
QoS	: Quality of Service
RFC	: Request For Comments
RTP	: Real-Time Transport Protocol
SBC	: Session Border Controller
SDP	: Session Description Protocol
SG-CG	: Smart Grid Coordination Group
SGIP	: Smart Grid Interoperability Panel
SIMPLE	: Session Initiation Protocol for Instant Messaging and Presence Leveraging Extensions
SIP	: Session Initiation Protocol

SOAP	: Simple Object Access Protocol
STUN	: Simple Traversal of UDP through NATs
TCP	: Transmission Control Protocol
TLS	: Transport Layer Security
UDP	: User Datagram Protocol
UML	: Unified Modeling Language
URI	: Uniform Resource Identifier
URL	: Uniform Resource Locator
VoIP	: Voice over Internet Protocol
XML	: Extensible Markup Language

ŞEKİL LİSTESİ

Şekil 1.1	: Akıllı Şebeke Mimarisi (Topolojisi).....	3
Şekil 2.1	: OSI ve TCP/IP referans modelinde SIP	6
Şekil 2.2	: Temel SIP sinyalleşmesi	7
Şekil 2.3	: Örnek INVITE isteği (SDP içermeyen)	8
Şekil 2.4	: Örnek REGISTER isteği	12
Şekil 2.5	: Örnek BYE isteği	13
Şekil 2.6	: Örnek ACK isteği	13
Şekil 2.7	: Uçtan uca ve adım adım gerçekleştirilen isteklerin farkı.....	14
Şekil 2.8	: Örnek CANCEL isteği	15
Şekil 2.9	: Örnek OPTIONS isteği	16
Şekil 2.10	: SUBSCRIBE – NOTIFY çağrı akışı.....	17
Şekil 2.11	: Örnek SUBSCRIBE isteği	18
Şekil 2.12	: Örnek NOTIFY isteği	19
Şekil 2.13	: MESSAGE çağrı akışı	20
Şekil 2.14	: Örnek MESSAGE isteği	21
Şekil 2.15	: Akıllı Şebeke Mimari Konsept Modeli.....	25
Şekil 2.16	: Ağ Bağlantılı Eşyalar için SIP taslağındaki mimari	29
Şekil 2.17	: Paketleyici Yazılım Katmanı	30
Şekil 2.18	: AMR SIP mesajlaşması	32
Şekil 2.19	: Şarj Oturumunun Kurulması.....	33
Şekil 2.20	: Şarj Oturumunun Sonlandırılması	34
Şekil 2.21	: Otomatik Talep-Cevap olay bildirimleri.....	35
Şekil 3.1	: SIP kullanarak ev otomasyonu topolojisi.....	37
Şekil 3.2	: Ev otomasyon sunucusunun arayüzleri.....	39
Şekil 3.3	: İstemci sınıf hiyerarşisi UML diyagram gösterimi	41
Şekil 3.4	: İstemci arayüzü	41
Şekil 3.5	: Ev otomasyon sunucusu sınıf hiyerarşisi UML diyagram gösterimi	43
Şekil 3.6	: Ev otomasyon sunucusu arayüzü	44
Şekil 3.7	: Vekil sunucuya kayıt sinyalleşmesi akışı.....	45
Şekil 3.8	: İstemci ve sunucu arasındaki örnek sorgulama sinyalleşmesi akışı	45
Şekil 3.9	: Örnek otomasyon sinyalleşmesi akışı.....	46
Şekil 3.10	: Örnek olay bildirimi sinyalleşmesi akışı.....	47
Şekil 3.11	: Ev otomasyon sunucusu sisteminin fotoğrafı	49
Şekil 3.12	: XBEE ve MSP-EXP430G2 modüllerinin pin diyagramları	50
Şekil 4.1	: OPTIONS isteğiyle sorgulamadaki arayüz ekran çıktıları.....	55
Şekil 4.2	: MESSAGE isteğiyle komutların uygulanmasındaki arayüz ekran çıktıları.....	58
Şekil 4.3	: Olay bildirimi arayüz ekran çıktıları.....	61
Şekil 4.4	: SIP OPTIONS ve ZIGBEE ile sorgulama çıktıları.....	64

Şekil 4.5	: SIP MESSAGE ve ZIGBEE ile otomasyon çıktıları 65
Şekil 4.6	: SIP SUBSCRIBE/NOTIFY ve ZIGBEE ile olay bildirimi çıktıları 66

ÖZET

AKILLI ELEKTRİK ŞEBEKESİNDE SIP PROTOKOLÜNÜN KULLANIMI

Akıllı Elektrik Şebekesi daha güvenilir, daha verimli ve daha otomatik hareket eden bir elektrik şebekesi elde edilmesi için sayısal teknolojileri kullanan yeni nesil elektrik şebekesidir. Akıllı Elektrik Şebekesi projesi elektrik enerjisi sağlayıcılar/dağıtıcılar ile tüketiciler arasında çift yönlü enerji ve bilgi alışverişine izin verir ve böylece yenilenebilir enerji kaynaklarının daha sık kullanılmasına ve elektrik sağlayıcıları, tüketiciler ve diğer katılımcılar için yeni servisler oluşturulmasına imkan verir. Bu altyapı bahsedilen hedeflerin gerçekleştirilmesi için bir iletişim ağı mimarisine ihtiyaç duyar. Dolayısıyla, bir çok organizasyon ve kuruluş Akıllı Elektrik Şebeke'sinin ihtiyaçlarına istinaden iletişim ağı mimarisinin oluşturulması için standartların, protokollerin ve teknolojilerin üzerinde çalışmaktadır. SIP, Akıllı Elektrik Şebekesi'nde kullanılması muhtemel protokoller arasında listelenmiştir.

SIP, internet üzerinden ses iletimi yapılan ağlarda multimedya oturumlarının başlatılması ve yönetilmesi amacıyla sıkça kullanılmaktadır. SIP ayrıca anlık mesajlaşma protokolü olarak kullanılabilinecek yeteneğe sahiptir. Bu çalışma SIP protokolünün Akıllı Elektrik Şebekesi'ndeki olası kullanımını Akıllı Elektrik Şebekesi Konsept Referans Modeli'nin Müşteri alanına odaklanarak, özellikle Enerji Yönetim Sistemi'nin bir parçası olarak ev alan ağı (HAN) içerisinde son kullanıcıya yönelik uygulama ve servisler için incelemektedir. Bu doğrultuda SIP'in kullanılabilirliğinin analiz edilmesi amacıyla, SIP sinyalleşmesine dayalı bir uzaktan ev otomasyonu simüle edilmiş ve SIP-ZIGBEE işbirliğiyle ev otomasyon sisteminin gerçekleştirilmesi tamamlanmıştır.

SUMMARY

THE USE OF SIP PROTOCOL IN SMART GRID NETWORKS

Smart Grid is the new generation power grid which uses digital technologies to achieve more reliable, efficient and automated grid. Smart Grid project allows two-way exchange of energy and information between utilities and consumers, thus enables widely use of renewable energy resources and creating new services for utilities, consumers and other participants. This infrastructure requires communications network architecture to achieve these mentioned goals. Therefore, several organizations work on identifying standards, protocols and technologies to build communications network architecture based on needs of Smart Grid. SIP is listed as one of the protocols that can possibly be used in Smart Grid's communications network architecture.

SIP is widely used in voice over IP networks to initiate and manage multimedia sessions. SIP also has the capability to be used as an instant messaging protocol. This work investigates the possible use of SIP in Smart Grid which focuses on Customer domain of Smart Grid Conceptual Reference Model, especially the use in HAN (Home Area Network) for end user applications and services as a part of Energy Management System. Accordingly, a remote home automation service based on SIP signaling is simulated and a realization of SIP-ZIGBEE interworking home automation system is completed to analyze the use of SIP.

1. GİRİŞ

Günümüze kadar elektrik şebekeleri, elektriğin üretildiği santrallerden tüketiciye doğru enerjinin iletildiği, sayısal (dijital) teknolojilerdeki gelişmelere rağmen son yıllara kadar bu gelişmelerden yeteri kadar faydalanamayan bir yapı içerisindeydi. Bu teknolojik gelişmelerin katkısıyla elektrik şebekelerinin daha akıllı ve daha verimli yapılar haline gelmesi için birçok çalışma yapılmakta, dünya çapında birçok kurum, organizasyon ve şirket bu çalışmalara katkı sağlayıp gerekli standartların oluşturulmasında görev yapmaktadır. EPRI (Electric Power Research Institute), NIST (National Institute of Standards and Technology), IEC (International Electrotechnical Commission), ETSI (European Telecommunications Standard Institute) ve CEN (European Committee for Standardization) akıllı elektrik şebekeleri çalışmalarına ve standartlarına yön veren organizasyonların başında gelmektedir.

Temel olarak akıllı elektrik şebekesi (smart grid) kavramı, uçtan uca gelişmiş iletişim altyapısını elektrik şebekesine entegre ederek, kullanıcılara elektrik kullanımlarıyla ilgili gerçek zamanlı bilgi verebilecek, arz ve talebe göre enerji tüketimi fiyatlandırılmasının yapılabilineceği, kullanıcının akıllı ev eşyalarına sahip olup enerji tüketimini optimize ederken, bu eşyaları kontrol edebileceği yapıyı tanımlamaktadır [1]. Akıllı bir şebeke yenilikçi ürün ve servisleri akıllı takip etme, kontrol etme, haberleşme, kendini iyileştirmeye bir araya getirerek şunları sağlar [2];

- Farklı teknoloji ve büyüklükte üreticilerin (jeneratörlerin) daha iyi bir şekilde şebekeye bağlanması ve kullanılmasını daha iyi sağlamak.
- Kullanıcıların sistemin yürütülmesinde rol oynamasını sağlamak.
- Kaynağın (üreticinin) seçilmesinde kullanıcıya daha çok seçenek ve bilgi vermek.

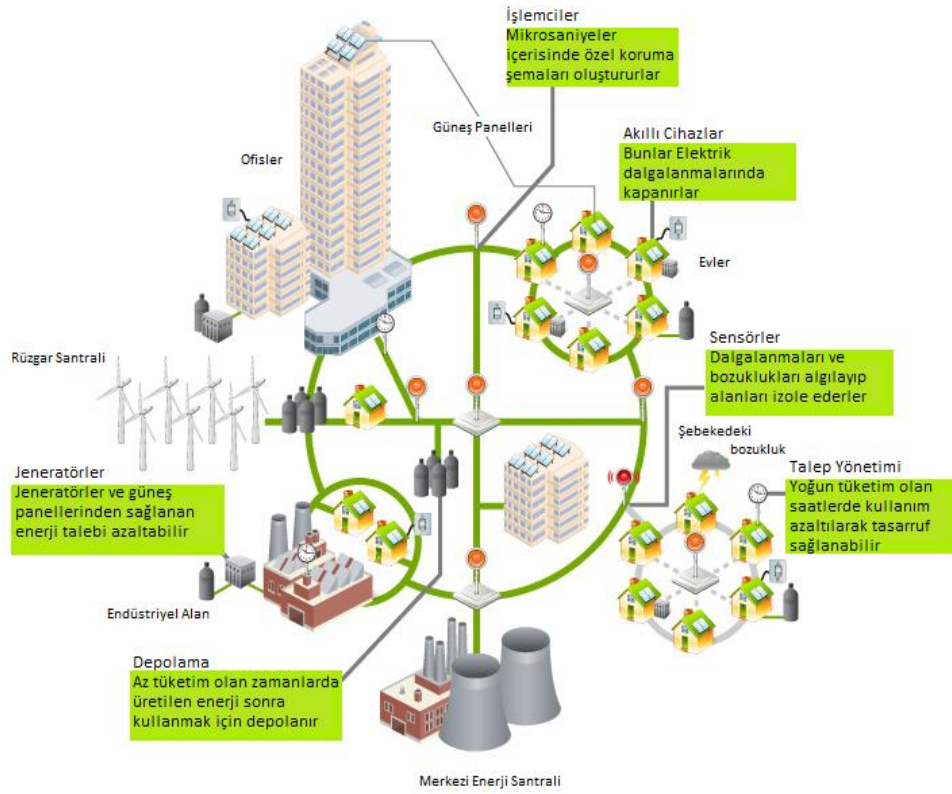
- Elektrik şebekesinin doğaya verdiği zararı en aza indirmek.
- Kaynak güvenilirliği, kalitesi ve güvenliğini arttırmak.
- Var olan servisleri daha verimli kullanılacak şekilde geliştirmek.

Akıllı elektrik şebekesinin oluşturulması güneş ve rüzgar gibi yenilenebilir enerji kaynaklarının kullanımını daha verimli kılmaktadır. Çünkü akıllı elektrik şebekesi klasik elektrik şebekesi gibi tek yönlü bir dağıtım yerine çift yönlü bir dağıtıma sahiptir [2, 3]. Bu sayede kullanıcılar sahip oldukları yenilenebilir enerji kaynaklarından enerji ihtiyaçlarını karşılayabilir, yenilenebilir kaynaklar yetmediği takdirde elektrik şebekesinden enerji talep edebilir veya yenilenebilir kaynaklardan sağlanan enerji tamamen tüketilmediği durumlarda bu enerji şebekenin de talep etmesi durumunda şebekeye verilebilir. Böylelikle farklı enerji kaynakları verimli bir şekilde kullanılmış olur.

Talep ve arzın düzgün bir şekilde ölçülmesiyle zaman temelli tarifeler ve fiyatlandırmalar oluşturulabilir. Akıllı ev eşyaları kullanılarak kullanıcının elektrik tüketiminin optimize etmesi ve bu eşyalar üzerinde uzaktan bile kontrolü tam anlamıyla sağlanabilir. Kullanıcı bütün bunlardan faydalanarak elektrik tüketiminde tasarruf sağlayabilir.

Aynı zamanda şebekede dalgalanma ve bozukluklar (arızalar) oluşması durumunda otomatik aksiyonlar alınarak, farklı kaynakların kullanımı ve kendini iyileştirme senaryoları oluşturulabilir. Şekil 1.1 akıllı elektrik şebekesinin örnek mimarisini (topoloji) göstermektedir.

Akıllı elektrik şebekesi, uçtan uca gelişmiş iletişim (telekomünikasyon) altyapısının elektrik şebekesine entegre etme çabası olduğundan elektrik enerjisiyle ilgilenen şirket ve organizasyonlarının yanı sıra telekomünikasyon şirket ve organizasyonlarının desteğini de gerektirmektedir. Bu konuda NIST ve IEC'nin yanı sıra ITU (International Telecommunication Union), IETF (Internet Engineering Task Force) gibi kurumlar çalışma grupları oluşturarak gerekli noktalarda çalışmalara destek vermektedir.



Şekil 1.1 Akıllı Şebeke Mimarisi (Topolojisi) [4]

İnternet protokollerini belirleyen kurum olarak IETF, akıllı elektrik şebekesinde internet protokollerinin kullanımıyla ilgili çalışmaları yürütmekte ve tavsiyelerini oluşturmaktadır. IP temelli iletişimin gerekli olduğu noktalarda hangi protokollerin kullanılması gerektiğini ve var olan protokollerin gereksinimleri karşılamadığı durumlarda gerekli çalışmaların yapılarak yeni bir protokol kümesi oluşturulmasını sağlamaktadır.

6lowpan (IPv6 over low power wireless personal area network) çalışma grubu kablosuz teknolojiler üzerinde çok düşük güç tüketimine sahip sensör ve kontrol ağlarında kullanılabilecek IPv6 temelli protokol kümesinin oluşturulması için çalışmaktadır. Bu protokol kümesi ev alan ağlarında (HAN) otomasyon, durum bildirimi gibi uygulama alanlarını IEEE 802.15.4 standardı üzerinde sağlayabilecek yapıyı oluşturacaktır [5, 6].

Bu çalışma grubunun gerçekleştirdiği çalışmalarının yanı sıra IETF tarafından yayınlanan bir internet taslağıyla akıllı elektrik şebekesi konusunda kullanılması muhtemel internet protokolleri, akıllı şebeke konusu üzerine çalışanları bilgilendirmek ve gerektiği yerlerde bu protokollerden faydalanmaları için listelenmiş ve açıklanmıştır [7]. Belirtilen protokollerin içerisinde bu çalışmanın konusu olan SIP [8] de yer almaktadır.

SIP [8], IETF tarafından tanımlanmış olan bir uygulama katmanı protokolüdür. Multimedya iletişiminin gerçekleştiği oturumların kurulması, yönetilmesi ve değiştirilmesi için yaygın olarak kullanılan bir standarttır. Bu multimedya oturumları ses, görüntü, anlık ileti (mesajlaşma), paylaşılan veri ve/veya belli olayların (durumların) üyeliği olabilir. SIP; TCP, TLS kullanarak TCP, UDP gibi iletim (transport) katmanı protokolleri üzerinde çalışabilir. SIP altında yatan iletim (transport) katmanı protokolünden ve IPv4/v6 protokolünden bağımsızdır. Hatta SIP mesajları SIP birimleri üzerinden geçerken iletim katmanı protokolü değişebilir [7].

SIP; oturumun ses mi video mu olacağına kendisi karar vermez, hatta oturumun gerçekleşeceği uç noktaların adreslerini (IP) de kendisi belirtmez. Bu amaçlar için SDP (Oturum Açıklama Protokolü) [9] kullanılır. SIP ile taşınan SDP içerisinde medya (codec) bilgisi önerilir ve kabul edilir/reddedilir, uç noktalar arasında iletilmesi gereken RTP (Real-Time Transport Protocol) [10] paketlerinin iletileceği IP adres ve port numarasını barındırır [7].

Kablosuz haberleşmenin hızla geliştiği ve yer bağımsız servisler üzerinde çalışmaların yoğunlaştığı günümüzde, SIP'in IMS (IP Multimedia Subsystem) altyapısında, sabit ve mobil ağlar (sağlayıcılar) arasında yakınsamayı sağlayacak uygulama katmanı protokolü olarak çalışması öngörülmüştür [11]. Dolayısıyla SIP yeni nesil haberleşme teknolojilerinin kullandığı temel protokoller arasında yerini almaktadır. Akıllı elektrik şebekesinin; uçtan uca gelişmiş haberleşme altyapısını elektrik şebekesiyle entegre etmek olduğu düşünüldüğünde, SIP akıllı elektrik şebekesi mimarisinin standartları arasında yerini alamasa bile, telekomünikasyon sektörünün ve uygulamalarının kurulan akıllı elektrik şebekesine yakınsaması gerektiği senaryolarda SIP'e ihtiyaç duyulabilir.

1.1 AMAÇ

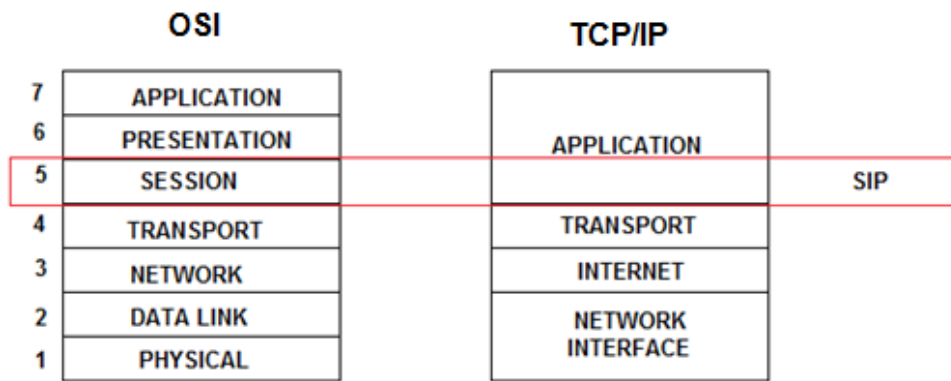
Bu tez çalışmasında SIP protokolünün akıllı elektrik şebekelerindeki kullanımı araştırılacak ve protokolün akıllı elektrik şebekesi içerisinde kullanım bulabileceği alanlar irdelenecektir. Akıllı elektrik şebekesinin birçok farklı katılımcı ve etmenden oluşan geniş bir platform olması ve SIP'in daha çok son kullanıcıya yönelik uygulamalarda sıkça kullanılması nedeniyle, SIP'in akıllı evlerde uzaktan otomasyon amacıyla kullanılmasına yönelik bir simülasyon ve SIP ile ZIGBEE [12] protokollerinin beraber kullanımıyla bir uygulama gerçekleştirilecektir. Sonuç olarak SIP protokolünün akıllı elektrik şebekesindeki kullanımı, başarısı, avantajları ve dezavantajları irdelenecektir.

2. GENEL KISIMLAR

2.1 SIP

Oturum başlatma (kurma) protokolü (SIP) internet üzerinden multimedia oturumları kurmak ve değiştirmek (modifiye etmek), durum bildirme ve isteme ile anlık mesajlaşma için tasarlanmış yeni bir sinyalleşme, durum ve anlık mesajlaşma protokolüdür. SIP, IETF tarafından TCP, TLS, UDP, IP, DNS ve diğer internet protokolleriyle bütünleşik çalışabilecek şekilde tasarlanmıştır [13]. SIP üzerinde çalıştığı bütün bu protokollerden bağımsızdır [7].

SIP, OSI (Open System Interconnection) modelinin oturum katmanında (5. Katman) yer alır. TCP ve UDP gibi farklı iletim katmanı (4. katman) protokolleriyle çalışabilmektedir. UDP paket kayıplarına karşı güvenilir bir protokol olmadığından SIP standardı içerisinde bu kayıplara karşı yeniden gönderim mekanizmaları ve zamanlamaları ilgili RFC'lerde anlatılmıştır [8].

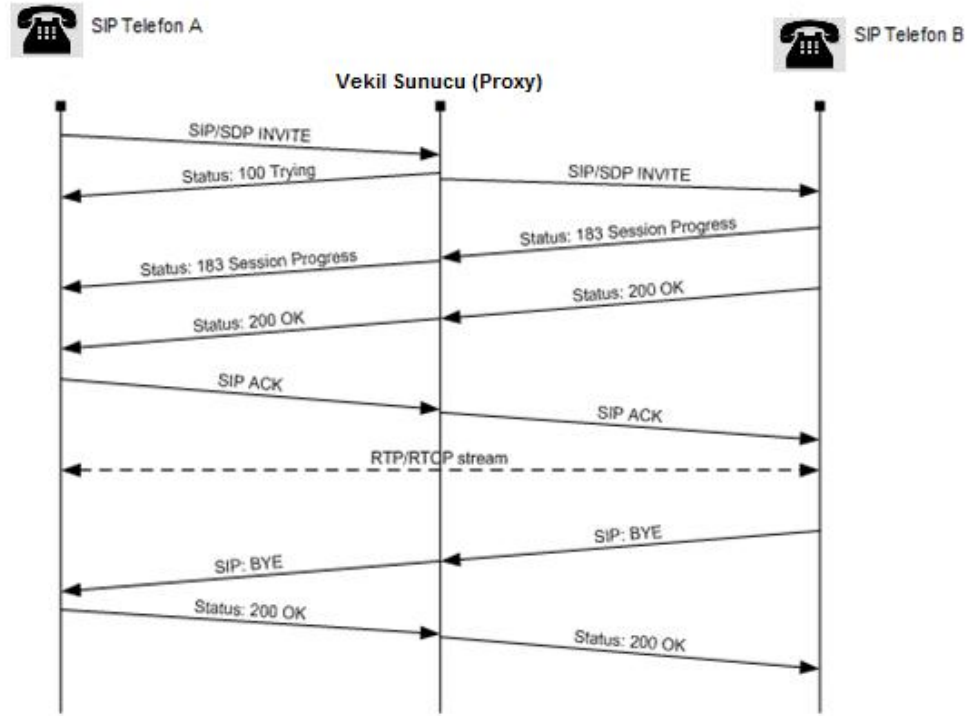


Şekil 2.1 OSI ve TCP/IP referans modelinde SIP

SIP, HTTP'den ilham alınarak oluşturulmuştur [8]. Dolayısıyla istek ve bu isteklere verilen cevaplardan oluşur. Internet ortamında yaygın olarak VoIP (Voice over IP), internet üzerinden ses iletimi uygulamalarında kullanılır. Kablosuz haberleşmenin hızla

geliştiđi ve yer bağımsız servisler üzerinde çalışmaların yoğunlaştığı günümüzde, SIP'in IMS (IP Multimedia Subsystem) altyapısında, sabit ve mobil ağlar (sağlayıcılar) arasında yakınsamayı sağlayacak uygulama katmanı protokolü olarak çalışması öngörülmüştür [11].

Şekil 2.2 temel SIP sinyalleşmesini diyagram olarak göstermektedir.



Şekil 2.2 Temel SIP sinyalleşmesi

A kullanıcısı oturumu başlatmak için INVITE isteğinde bulunmalıdır. INVITE içerisinde isteği yapanın ve istek yapılanın SIP URI'leri (Unified Resource Identifier) bulunur. İsteğin yapıldığı B kullanıcısına SIP Proxy (SIP vekil sunucusu) aracılığıyla DNS adreslemesi kullanılarak bu istek ulaştırılır. İsteğe cevap olarak gönderilen 100-199 arasındaki yanıtlar, kurulum aşaması yanıtlarıdır. Bu yanıtlar isteğin alındığını ve isteğin yerine getirilmesi üzerinde çalışıldığını, istek yapan tarafa bildirmek için kullanılan opsiyonel cevaplardır [8]. Şekil 2.3'te A kullanıcısının örnek INVITE isteği gösterilmektedir.

```

INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bK776asdhds
Max-Forwards: 70
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710@pc33.atlanta.com
CSeq: 314159 INVITE
Contact: <sip:alice@pc33.atlanta.com>
Content-Type: application/sdp
Content-Length: 142
(A kullanıcısının SDP'si gösterilmemektedir)

```

Şekil 2.3 Örnek INVITE isteği (SDP içermeyen) [9]

İsteğin ilk satırında isteğin türü belirtilmiştir. İsteğin içerdiği başlıkları (header) kısaca açıklayacak olursak;

“Via” başlığı A kullanıcısının bu isteğe karşı cevap almayı beklediği adresi içerir ve istek işlemini belirten (ayıran) ”branch” parametresini içerir.

“To” başlığı isteğin yönlendirilmesi gereken birimin SIP URI adresini içerir.

“From” başlığı isteği üreten birimin SIP URI adresini ve diyalogun tanımlanmasında kullanılmak üzere birim tarafından rastgele üretilen “tag” parametresini içerir.

“Call-ID” başlığı isteği yapan birim tarafından rastgele üretilen bir dizi ve birimin URI veya IP’sinden oluşan global eşsiz tanımlayıcıyı içerir. Call-ID, From başlığı tag’i ve To başlığı tag’inin kombinasyonu, A kullanıcısıyla B kullanıcısı arasında diyalog adı verilen uçtan uca SIP bağlantısını tanımlar.

“CSeq” başlığı diyalog içerisindeki isteklerin sıra numarasıdır ve diyalog içerisindeki her bir yeni istekte arttırılır.

“Contact” başlığı A kullanıcısına doğru direk yönlendirmeyi belirten SIP URI’sini içerir. Genel olarak kullanıcı adı ve FQDN’in (Fully Qalified Domain Name) bütünleştirilmesinden oluşur ama her sistemin bir FQDN’i olmadığından IP kullanımına da izin verilir. “Via” başlığı diyalogun bileşenlerine cevapların nereye gönderilmesi gerektiğini belirtirken, “Contact” başlığı diyalogun ilerleyen sürecinde yapılması olası isteklerin hangi adrese yapılması gerektiğini gösterir.

“Max-Forwards” başlığı isteğin yapılacağı birime kadar maksimum kaç noktadan geçilebileceğini belirler ve her bir noktada bir azaltılır.

“Content-Type” başlığı mesaj içeriğinin açıklamasını içerir. (SDP/Text)

“Content-Length” başlığı ise mesaj içeriğinin octet (byte) sayısını içerir [8].

Kurulacak gerçek zamanlı oturumun detayı SIP mesajının içinde bulunmaz. Bunun yerine SIP mesajı bu bilgileri başka protokoller şeklinde barındırır. SDP’yi (oturum açıklama protokolü) buna örnek olarak verebiliriz.

Bu örnekte B bu isteğe olumlu cevap vermek istiyor. Dolayısıyla B A’ya 200 OK yanıtını göndermektedir. 200 OK yanıtı B tarafının oturumu kurmak istediği medya parametrelerini belirtecek şekilde SDP içermektedir. Bu şekilde 2 aşamalı SDP alışverişi A’dan B’ye INVITE, B’den A’ya 200 OK ile gerçekleşmiş olur. Eğer B bu isteğe olumlu olarak cevap vermek istemeseydi; örneğin meşgul yanıtı vermek isteseydi; 200 OK yerine hata yanıtlarından birini kullanmak durumundaydı.

Oturum kurulma aşamasında son olarak A tarafı 200 OK gerçek yanıtını aldığını doğrulayacak ACK mesajını B’ye gönderir ve bu mesaj INVITE/200 OK/ACK mesajlaşma üçlüsünün tamamlanmasını sağlar.

Bu andan itibaren A ile B arasında medya oturumu paylaşılan SDP’ler aracılığıyla anlaşılan formatta başlamış olur.

Oturumun sonunda B tarafı çağrışı sonlandırmak üzere BYE isteğini gönderir. A tarafı BYE mesajını aldığını onaylamak üzere 200 OK cevabını verir ve bu oturum işlemini sonlandırır [8].

2.1.1 İstek Yöntemleri

SIP farklı amaçlarda kullanılmak üzere farklı istek yöntemleri tanımlamaktadır. Gerçek zamanlı multimedya oturumlarının kurulması için tasarlanan SIP, son zamanlardaki çalışmalarla anlık mesajlaşma, bir olaya üye olup olay hakkında uyarılar alma gibi sistemlere sahip olmuştur.

2.1.1.1 Invite

INVITE yöntemi kullanıcı araçları arasında medya oturumu kurulması için kullanılır. INVITE'lara verilen cevaplar her zaman ACK yöntemi ile doğrulanmak zorundadır.

INVITE istekleri genel olarak oturum bilgisini içeren mesaj içeriğine (SDP) sahip olurlar. Mesaj içeriği ayrıca QoS veya güvenlik gibi diğer oturum bilgilerini içerebilir. Eğer bir INVITE medya bilgisi içermiyorsa, istemci kullanıcı aracının medya bilgisini ACK yöntemi taşımalıdır. Eğer ACK içerisindeki medya bilgisi kabul edilebilir değilse, istek yapılan taraf oturumu iptal etmek için BYE yöntemini kullanmalıdır. Bunun sebebi oturumun zaten 200 OK ile kurulmuş olmasından dolayı CANCEL yönteminin kullanılamamasıdır. İstemci kullanıcı aracıyla sunucu kullanıcı aracı arasındaki medya oturumu INVITE, 200 OK, ACK mesajlarının karşılıklı değiş tokuşundan sonra başlar. Başarılı bir INVITE isteği iki kullanıcı arasında bir diyalog oluşturur ve bu diyalog taraflardan biri tarafından BYE isteği gönderilip oturum sonlandırılana kadar devam eder.

Bir diyalog oluşturmak üzere INVITE oluşturan bir istemci kullanıcı aracı, oturumun tanımlanmasını sağlayan global olarak eşsiz Call-ID değerini üretir. Aynı Call-ID'ye sahip her yeni istekte, arttırılmak üzere bir CSeq değeri oluşturulur. To ve From başlıkları uzak ve yerel adresleri içerecek şekilde oluşturulur. Diyalogun belirlenmesi amacıyla INVITE içerisindeki From başlığının içine Tag parametresi eklenir ve sunucu kullanıcı aracı da her cevabında To headeri içerisine Tag parametresini ekler. INVITE isteğine verilen 200 OK yanıtındaki To başlığı Tag parametresi, istemci kullanıcı tarafından gönderilen ACK yönteminde ve diyalog içerisindeki daha sonraki bütün

isteklerde kullanılır. To başlığındaki Tag parametresi, From başlığındaki Tag parametresi ve Call-ID başlığının kombinasyonu eşsiz bir diyalogu tanımlar.

Call-ID, From Tag ve To Tag parametreleriyle tanımlanmış var olan bir diyalog için gönderilen yeni bir INVITE isteği re-INVITE olarak isimlendirilir ve oturumun karakteristik parametrelerinin değiştirilmesi veya diyalogun durumunun yenilenmesi için kullanılır. CSeq değeri arttırılmış olacağından istek yapılan taraf bunun orijinal INVITE'ın tekrar gönderimlerinden ayırabilir. Bir istemci kullanıcı aracı ilk INVITE'a gerçek bir yanıt almadan re-INVITE göndermemelidir ve INVITE yerine UPDATE yöntemini gönderebilir.

INVITE isteği içerisindeki Expires başlığı bu isteğin ne kadar süre ile geçerli olacağını belirtir. Örneğin, sunucu kullanıcı aracı cevaplanmamış bir INVITE isteğini Expires başlığında tanımlanan süre kadar ekranda gösterebilir. Oturum kurulduktan sonra Expires başlığının bir anlamı yoktur. Eğer kurulacak oturumun belli bir zamanla sınırlandırılması isteniyorsa Session-Expires başlığı kullanılabilir.

Bu yöntem için gerekli başlıklar dışında opsiyonel Subject başlığı kullanılabilir. Ayrıca Request-Uri başlığı SIP URI'yi barındırır ve SIP URI telefon numaralarını destekler [13].

2.1.1.2 Register

REGISTER yöntemi bir kullanıcı aracı tarafından SIP ağını şu anki iletişim bilgisiyle (Contact başlığındaki URI aracılığıyla) uyarmak/kaydetmek için kullanılır. Kayıt işlemi kullanıcı aracın vekil sunucuyu kullanarak istek göndermesini sağlamak için değil kullanılan alan adını sunan vekil sunucuya iletişim bilgisiyle kaydolup gelen istekleri alabilmek içindir. Standartlarda kullanımı belirtilmemesine rağmen REGISTER isteği mesaj içeriğine sahip olabilir. Contact ve Expires başlıklarının kullanımına bağlı olarak kayıt sunucuları farklı aksiyon alırlar. Herhangi bir Expires başlığı ve Expires parametresi olmaması durumunda SIP URI'nin 1 saat içinde süresi dolacaktır. Herhangi bir expires parametresi olmadan Expires başlığının varlığı bağlantıların geçerlilik süresini belirler. Expires parametresi varsa sadece o bağlantının geçerlilik süresini belirler.

CSeq değeri her REGISTER isteği için artırılır. Request-URI, To, From ve Call-ID başlıklarının kullanımı diğer istek türlerine göre oldukça farklıdır. Request-URI başlığı kayıt sunucusunun herhangi bir kullanıcı bilgisi olmadan sadece alan adını içerir. REGISTER isteği o alan adı için yetkili kayıt sunucusuna ulaşana kadar yönlendirilebilir. To başlığı kayıt olmak isteyen kullanıcı aracın adres kaydının SIP URI'sini içerir. From başlığı isteği gönderenin SIP URI'sini içerir ve genel olarak To başlığı ile aynıdır. Bir kullanıcı aracı tarafından yapılan bütün kayıt isteklerinde aynı Call-ID kullanılması tavsiye edilir.

REGISTER istekleri kaydın yapılması gereken adresi içerecek şekilde Contact başlığına sahip 3xx yönlendirme veya 4xx hata yanıtları alabilir [13]. Örnek bir REGISTER isteği şekil 2.4'te gösterilmektedir.

```
REGISTER sip:registrar.athens.gr SIP/2.0
Via: SIP/2.0/UDP 201.202.203.204:5060;branch=z9hG4bK313
Max-Forwards:70
To: sip:euclid@athens.gr
From: <sip:secretary@academy.athens.gr>;tag=543131
Call-ID: 2000-July-07-23:59:59.1234@201.202.203.204
CSeq: 1 REGISTER
Contact: sip:euclid@parthenon.athens.gr
Contact: mailto:euclid@geometry.org
Content-Length: 0
```

Şekil 2.4 Örnek REGISTER isteği [13]

2.1.1.3 Bye

BYE yöntemi kurulmuş olan bir medya oturumunu sonlandırmak için kullanılır. Bir oturumun kurulmuş olması için INVITE isteğine 200 OK cevabı alınmalıdır veya ACK gönderilmiş olmalıdır. BYE yöntemi sadece oturumun katılımcıları tarafından gönderilebilir, asla vekil sunucular veya harici bileşenler tarafından gönderilemez. Uçtan uca bir yöntem olduğundan sadece diğer kullanıcı aracı tarafından gönderilebilir. Üzerinde olmayan bir diyalog için BYE yöntemi alan bir kullanıcı aracı “481 Dialog/Transaction Does Not Exist” cevabını dönmelidir [13]. Şekil 2.5'te örnek bir BYE isteği gösterilmektedir.

```

BYE sip:info@hypotenuse.org SIP/2.0
Via: SIP/2.0/UDP port443.hotmail.com:5060;branch=z9hG4bK312bc
Max-Forwards:70
To: <sip:info@hypotenuse.org>;tag=63104
From: <sip:pythag42@hotmail.com>;tag=9341123
Call-ID: 34283291273@port443.hotmail.com
CSeq: 47 BYE
Content-Length: 0

```

Şekil 2.5 Örnek BYE isteği [13]

2.1.1.4 Ack

ACK yöntemi INVITE isteklerine gelen gerçek yanıtları onaylamak için kullanılır. Diğer isteklerin hiçbirine verilen gerçek yanıtlar onaylanmaz. Gerçek yanıtlar 2xx, 3xx, 4xx, 5xx veya 6xx sınıfı olan yanıtlardır. Bir ACK yöntemi için CSeq değeri hiç bir zaman arttırılmaz, sadece CSeq yöntemi ACK olarak değiştirilir. Böylece sunucu kullanıcı aracı ilgili INVITE'ın CSeq değeriyle ACK'inkini eşleyebilir.

ACK SDP mesaj içeriği taşıyabilir. Buna ilk INVITE bir SDP mesaj içeriğine sahip olmadığı durumda izin verilmektedir. Eğer INVITE içerisinde mesaj içeriği varsa, ACK mesaj içeriğine sahip olmayacaktır. ACK medya parametrelerinin değiştirilmesi için kullanılamaz, bunun yerine re-INVITE bu amaçla kullanılmalıdır [13]. Şekil 2.6'da örnek bir ACK isteği bulunmaktadır.

```

ACK sip:laplace@mathematica.org SIP/2.0
Via: SIP/2.0/TCP 128.5.2.1:5060;branch=z9hG4bK1834
Max-Forwards:70
To: Marquis de Laplace <sip:laplace@mathematica.org>;tag=90210
From: Nathaniel Bowditch <sip:n.bowditch@salem.ma.us>;tag=887865
Call-ID: 152-45-N-32-23-W@128.5.2.1
CSeq: 3 ACK
Content-Type: application/sdp
Content-Length: 143

v=0
o=bowditch 2590844326 2590944532 IN IP4 salem.ma.us
s=Bearing
c=IN IP4 salem.ma.us
t=0 0
m=audio 32852 RTP/AVP 0
a=rtpmap:0 PCMU/8000

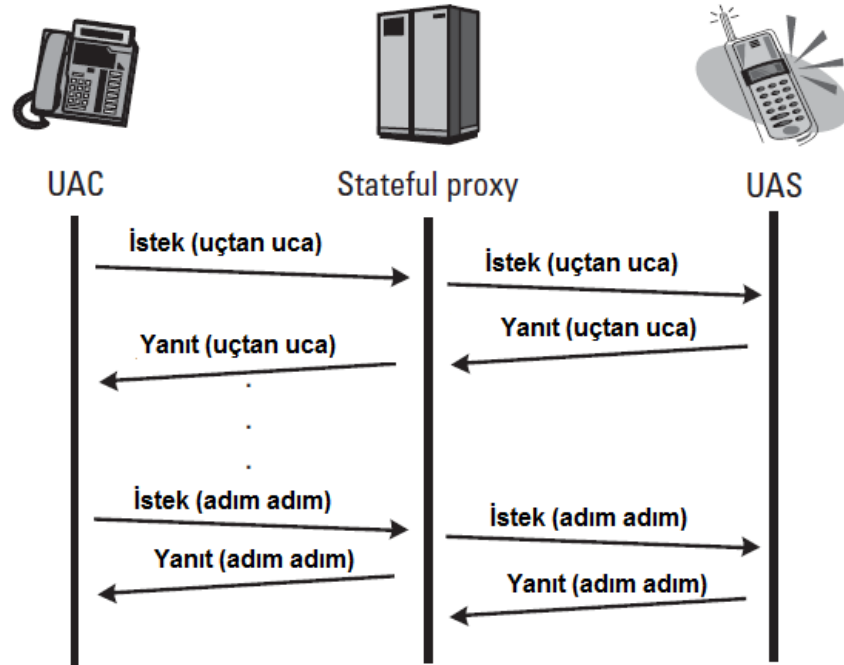
```

Şekil 2.6 Örnek ACK isteği [13]

2.1.1.5 Cancel

CANCEL yöntemi bekleyen çağrı girişimlerini sonlandırmak için kullanılır. İçinde tag parametresi bulunan herhangi bir 1xx yanıtı alan fakat gerçek yanıt almamış olan kullanıcı aracı veya vekil sunucusu tarafından üretilebilir. Kullanıcı aracı bu yöntemi daha önce yaptığı INVITE isteğini iptal etmek için kullanır. Vekil sunucu ise çatal sunucu senaryosunda, başarılı yanıt istemci kullanıcı araca ilettikten sonra, paralelde bekleyen isteklerin iptali için bu yöntemi kullanır. CANCEL adım adım iletilen bir istektir ve yanıt sonraki durum bilgisine sahip eleman tarafından oluşturulur. Uçtan uca olan bir istekle adım adım olan bir istek arasındaki fark şekil 2.7’de gösterilmiştir. Bu yöntem için CSeq değeri arttırılmaz. Böylece kullanıcı araçları ve vekil sunucular CANCEL’in CSeq değeri ile bekleyen INVITE isteğinin CSeq değerini eşleyebilirler.

CANCEL isteğinin “branch” değeri iptal ettiği INVITE isteğinkiyle eşleşmelidir. CANCEL isteği sadece INVITE isteği için anlamlıdır çünkü INVITE harici bütün istekler anında yanıt almaktadır.



Şekil 2.7 Uçtan uca ve adım adım gerçekleştirilen isteklerin farkı

CANCEL isteği alan bir vekil sunucu bu isteği bekleyen ilk INVITE isteğinin gönderildiği bütün adreslere iletir. Vekil sunucu iletilen CANCEL isteklerine yanıt beklemez ama kendisi anında yanıt verir. Kullanıcı aracı iptal işlemini 200 OK ile onaylar ve INVITE’a “487 Request Terminated” yanıtı döner.

Eğer bir gerçek yanıt ulaşırsa, kullanıcı aracı BYE yöntemi göndererek oturumu sonlandırır [13]. Şekil 2.8’de örnek bir CANCEL isteği görülmektedir.

```
CANCEL sip:i.newton@cambridge.edu.gb SIP/2.0
Via: SIP/2.0/UDP 10.downing.gb:5060
    ;branch=z9hG4bK3134134
Max-Forwards:70
To: Isaac Newton <sip:i.newton@cambridge.edu.gb>
From: Rene Descartes <sip:visitor@10.downing.gb>;tag=034323
Call-ID: 42@10.downing.gb
CSeq: 32156 CANCEL
Content-Length: 0
```

Şekil 2.8 Örnek CANCEL isteği [13]

2.1.1.6 Options

OPTIONS yöntemi bir kullanıcı aracın veya sunucunun yeteneklerini sorgulamak için veya şu anki durumunu öğrenmek için kullanılır. İsteğe verilen cevap kullanıcı aracın veya sunucunun yeteneklerini listeler. Vekil sunucular asla OPTIONS isteği oluşturmazlar. Kullanıcı araçlar veya sunucular bu isteğe bir INVITE’a verdikleri türden yanıtlar vermelidir. 2xx sınıfından bir yanıt yetenekleri gösterecek şekilde Allow, Accept, Accept-Encoding, Accept-Language ve Supported başlıklarını içerebilir.

OPTIONS isteği mesaj içeriği barındıramaz. Vekil sunucular OPTIONS isteğinin kendilerine mi yoksa başka vekil sunucu veya kullanıcı aracı için olup olmadığını Request-URI başlığını kontrol ederek anlarlar. Eğer Request-URI başlığı vekil sunucunun adresini içeriyorsa istek o vekil sunucu içindir aksi takdirde bu başka bir cihaz içindir ve istek iletilir [13]. Şekil 2.9’da örnek bir OPTIONS isteği gösterilmektedir.

```

OPTIONS sip:user@carrier.com SIP/2.0
Via: SIP/2.0/UDP cavendish.kings.cambridge.edu.uk
    ;branch=z9hG4bK1834
Max-Forwards:70
To: <sip:user@proxy.carrier.com>
From: J.C. Maxwell <sip:james.maxwell@kings.cambridge.edu.uk>
    ;tag=34
Call-ID: 9352812@cavendish.kings.cambridge.edu.uk
CSeq: 1 OPTIONS
Content-Length: 0

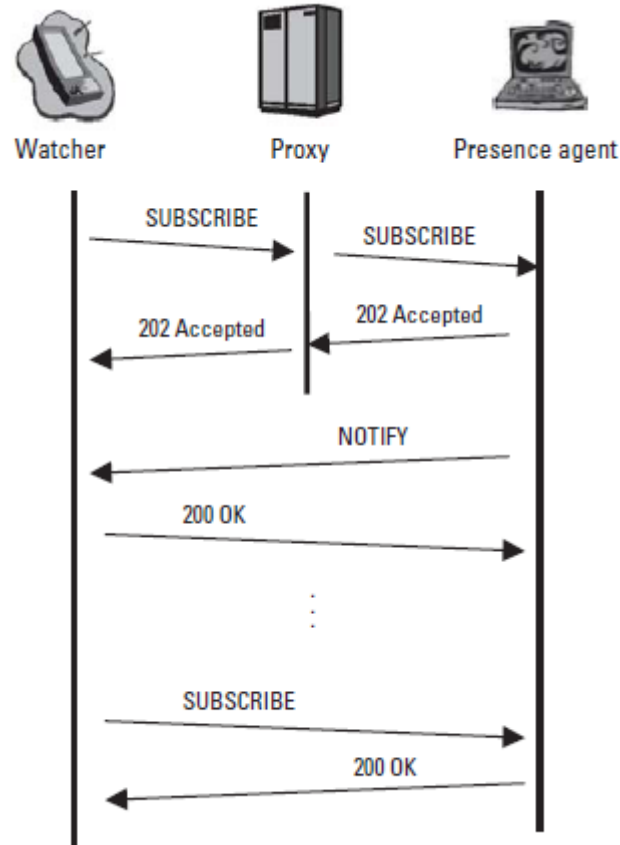
```

Şekil 2.9 Örnek OPTIONS isteği [13]

2.1.1.7 Subscribe

SUBSCRIBE yöntemi bir kullanıcı aracı tarafından belli bir olay ile ilgili uyarılar alabilmek amacıyla üyelik oluşturmak üzere kullanılır. Başarılı bir üyelik isteği istemci kullanıcı aracı ile sunucu kullanıcı aracı arasında bir diyalog oluşturur. Üyelik isteği üyeliğin geçerli olacağı süreyi gösterecek şekilde Expires başlığı bulundurur. Bu belirtilen zaman aşıldığında üyelik otomatik olarak sonlandırılır. Üyelik geçerlilik süresi dolmadan diyalog içerisinde yeni bir SUBSCRIBE isteği gönderilerek yenilenebilir. Üyeliği kabul eden sunucu bir Expires başlığı içerecek şekilde 200 OK yanıtını döner. Buradaki geçerlilik süresi istekteki ile aynı olabilir veya sunucu süreyi kısaltabilir fakat süreyi uzatamaz. Eğer üyelik sonlandırılmak isteniyorsa diyalog içerisinde Expires başlığındaki değer “0” olacak şekilde SUBSCRIBE isteği gönderilmesi gerekir. Hem sonlandırma isteğiyle hem de geçerlilik süresi dolduğundan bir üyelik sonlandırıldığında üyeliğin sonlandırıldığını belirten son NOTIFY isteği gönderilir.

202 Accepted yanıtı SUBSCRIBE isteğinin onaylandığını ve yetkilendirildiğini göstermez, isteğin sunucu tarafından anlaşıldığını belirtir.



Şekil 2.10 SUBSCRIBE – NOTIFY çağrı akışı

Temel çağrı akışı şekil 2.10’da gösterilmiştir. İstemci başarılı olan bir SUBSCRIBE isteği göndermiştir ve izlenmek istenen olaylar oluştuğunda NOTIFY yöntemleri almıştır.

Üye olunmak istenen olay SUBSCRIBE isteğinin Event başlığında belirtilir. SIP Event çerçevesinin her uygulaması bir paket tanımlar. Bir sunucu desteklediği Event paketlerini Allow-Events başlığı altında listelemelidir [13].

Şekil 2.11’de örnek bir SUBSCRIBE isteği gösterilmiştir.

Uyaran taraf Event başlığında belirtilen Event paketini anladığını kontrol etmelidir. Eğer anlamazsa; istek içerisinde belirtilen Event paketinin anlaşılmadığını belirtmek üzere 489 Bad Event yanıtını dönmelidir [14].

```

SUBSCRIBE sip:ptolemy@rosettastone.org SIP/2.0
Via SIP/2.0/UDP proxy.elasticity.co.uk:5060
    ;branch=z9hG4bK348471123
Via SIP/2.0/UDP parlour.elasticity.co.uk:5060
    ;branch=z9hG4bKABDA ;received=192.0.3.4
Max-Forwards: 69
To: <sip:Ptolemy@rosettastone.org>
From: Thomas Young <sip:tyoung@elasticity.co.uk>;tag=1814
Call-ID: 452k59252058dkfj349241k34
CSeq: 3412 SUBSCRIBE
Allow-Events: dialog
Contact: <sip:tyoung@parlour.elasticity.co.uk>
Event: dialog
Content-Length: 0

```

Şekil 2.11 Örnek SUBSCRIBE isteği [14]

2.1.1.8 Notify

NOTIFY yöntemi belli bir olayın oluşması bilgisinin iletilmesi için kullanıcı araçlar tarafından kullanılır. NOTIFY yöntemi üye olan ve uyararı taraf arasında bir üyelik olduğu sürece diyalog içerisinde kullanılır. Yine de SIP anlamlarının dışında üyelikler kurulabilir ve başka bir SIP isteği tarafından üyelik zorlanılabilir. (örneğin REFER yöntemi) Diyalog içerisinde gönderileceğinden NOTIFY yöntemi diyalogun To Tag, From Tag ve Call-ID değerlerine sahip olacaktır. Şekil 2.10'da temel çağrı akışı gösterilmiştir.

Normalde NOTIFY isteği ulaştığını belirtecek şekilde 200 OK yanıtı alır. Eğer 481 Dialog/Transaction Does Not Exist yanıtı alırsa üyelik otomatik olarak sonlandırılmalı ve başka NOTIFY yöntemi gönderilmemelidir.

NOTIFY istekleri Event paketini belirtmek üzere Event başlığı ve üyeliğin durumunu belirtmek üzere Subscription-State başlığı bulundurulur. Subscription-State başlığı "active", "pending" veya "terminated" değerlerinden birini alır.

Bir üyeliğin daima başında ve sonunda NOTIFY gönderilir. Eğer bir NOTIFY isteği artan bir durum bilgisi içeriyorsa, mesaj içeriği gönderilen her NOTIFY için bir

arttırılan durum versiyonu numarasını içermelidir. Bu sayede isteği alan taraf olay sırasının doğru olup olmadığını anlayabilir [13].

Şekil 2.12’de örnek bir NOTIFY isteği gösterilmektedir.

```
NOTIFY sip:tyoung@parlour.elasticity.co.uk SIP/2.0
Via SIP/2.0/UDP cartouche.rosettastone.org:5060
;branch=z9hG4bK3841323
Max-Forwards: 70
To: Thomas Young <sip:tyoung@elasticity.co.uk>;tag=1814
From: <sip:ptolemy@rosettastone.org>;tag=5363956k
Call-ID: 452k59252058dkfj349241k34
CSeq: 3 NOTIFY
Contact: <sip:Ptolemy@cartouche.rosettastone.org>
Event: dialog
Subscription-State: active
Allow-Events: dialog
Content-Type: application/xml+dialog
Content-Length: ...

(XML Message body not shown...)
```

Şekil 2.12 Örnek NOTIFY isteği [13]

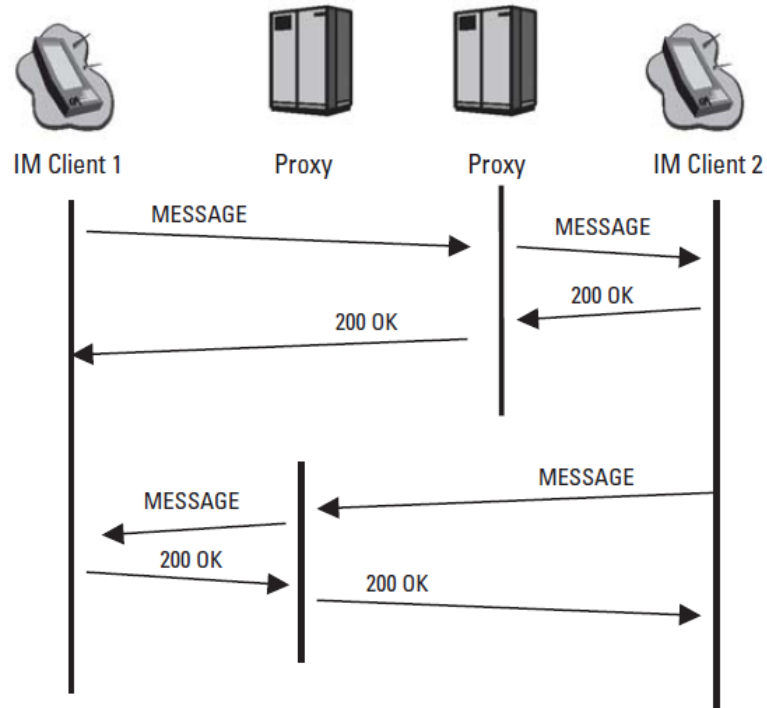
2.1.1.9 Message

MESSAGE yöntemi SIP kullanılarak anlık mesajlar iletilmesi için kullanılır. Anlık mesajlaşma bir konuşmadaki katılımcıların genel olarak yaklaşık gerçek zamanlı kısa mesaj değiş tokuşu olarak değerlendirilir. MESSAGE yöntemi diyalog içerisinde veya diyalog dışında kullanılabilir. Yalnız MESSAGE yöntemi kendi başına bir diyalog oluşturmaz. Gerçek mesajlar MIME eklentisi olarak mesaj içeriğinde yer alır. MESSAGE yöntemini destekleyen bütün kullanıcı araçları plain/text formatını desteklemek zorundadır ve message/cpim veya text/html gibi diğer formatları da destekleyebilirler.

Normal koşullarda bir MESSAGE isteği mesajın istenen son noktaya iletilildiğini belirtecek şekilde 200 OK yanıtı alır. 200 OK içerisinde anlık mesaj içeriği

gönderilmemelidir, bunun yerine ayrı bir MESSAGE yöntemiyle gönderilmelidir. 202 Accepted cevabı yöntemin depolayıp gönderim yapan bir ara cihaza ulaştığını belirtir ve muhtemelen uç noktaya iletilecektir. Her iki durumda da 2xx yanıtı mesaj içeriğinin kullanıcıya gönderildiğini onaylar.

Şekil 2.13'te örnek bir çağrı akışı gösterilmektedir.



Şekil 2.13 MESSAGE çağrı akışı [13]

Her ne kadar istemci adresi bir SIP veya SIPS URI adresine çözmeye çalışacak olsa da, MESSAGE isteği Request-URI içerisinde anlık mesajlaşma URI şemasını kullanabilir [13].

Şekil 2.14'te örnek bir MESSAGE yöntemi gösterilmektedir.

Normal kullanımda birçok SIP isteği oturum kurmak veya oturum parametrelerini değiştirmek için kullanılır. Katılımcılar arasındaki gerçek iletişim, isteklerin içinde değil; medya oturumunda gerçekleşir. MESSAGE yöntemi bu genel kuralı değiştirmektedir ve bu istek katılımcılar arasındaki gerçek iletişimi üzerinde

taşımaktadır. Bu durum MESSAGE isteği üzerine diğer SIP isteklerine nazaran daha fazla güvenlik ihtiyacı getirir [15].

```
MESSAGE sip:editor@rcs.org SIP/2.0
Via SIP/2.0/UDP lab.mendeleev.org:5060;branch=z9hG4bK3
Max-Forwards: 70
To: <editor@rcs.org>
From: "D. I. Mendeleev" <dmitry@mendeleev.org>;tag=1865
Call-ID: 93847197172049343
CSeq: 5634 MESSAGE
Subject: First Row
Contact: <sip:dmitry@lab.mendeleev.org>
Content-Type: text/plain
Content-Length: 5
```

H, He

Şekil 2.14 Örnek MESSAGE isteği [13]

2.1.2 Yanıtlar (Cevaplar)

Yukarıda belirtilen istekler için SIP HTTP benzeri nümerik yanıtlar kullanır. Bu yanıtlar ifade ettikleri anlamlara göre gruplanabilir.

1xx şeklindeki yanıtlar kurulum aşaması cevaplarıdır. İsteğin alındığı ve oturumun kurulma aşamasının devam ettiği gibi mesajlar içerirler. “100 Trying” ve “183 Session In Progress” yanıtlarını örnek olarak gösterebiliriz.

2xx şeklindeki yanıtlar isteğin başarılı bir şekilde işlendiğini ve oturumun kurulduğunu belirtirler. “200 OK” yanıtını örnek olarak gösterebiliriz.

3xx şeklindeki yanıtlar yönlendirmeler için kullanılır ve isteğin tamamlanabilmesi için başka işlemlerin yapılması gerektiğini belirtir. “302 Moved Temporarily” yanıtını örnek olarak gösterebiliriz.

4xx şeklindeki yanıtlar istemci (istek) temelli hataları ifade eder. İstek yapılan cihaz (sunucu) isteği karşılayamıyor veya istek hatalar barındırıyor olabilir. “404 Not Found” yanıtını örnek olarak gösterebiliriz.

5xx şeklindeki yanıtlar sunucu temelli hataları belirtir. İstek yapılan sunucunun isteği karşılayamayacak durumda olduğunu ifade eder. “503 Service Unavailable” yanıtını örnek olarak gösterebiliriz.

6xx şeklindeki yanıtlar global hataları belirtir. İsteği hiç bir sunucunun işleyemediğini anlatır. “604 Does Not Exist Anywhere” yanıtını örnek olarak gösterebiliriz [8].

2.2 AKILLI ŞEBEKE (SMART GRID)

Son 70 yıldır kurulan elektrik güç sistemleri, elektrik enerjisini büyük merkezi jeneratörlerden yüksek gerilim transformatörleri aracılığıyla beslemekte ve bu iletim şebekesi olarak adlandırılmaktaydı. İletim şebekesi elektrik enerjisinin dikkate değer mesafelerde birçok dağıtım transformatörü üzerinden geçerek son müşteriye ulaştırılması için kullanılmaktadır.

Yükü besleyen dağıtım sistemi, çok büyük kapsamlı fakat sınırlı bir iletişim ve yerel kontrol mekanizmalarına sahip; neredeyse bütünüyle pasif bir yapıdadır. Talep edildiğinde yük enerjisinin sağlanması dışında yükler ve enerji sistemi arasında çok az etkileşim vardır.

İletişim sistemlerinin günümüzdeki; özellikle internet tarafından ivmelenen gelişimi enerji sistemini daha verimli, esnek ve daha düşük operasyon maliyetine sahip hale getirecek şekilde sistem boyunca daha fazla izleme ve kontrol olanağı getirmekte. Akıllı Şebeke elektrik enerjisi sisteminin geliştirilmesi için yeni bilgi ve iletişim teknolojilerinin kullanımı için bir fırsattır.

Akıllı Şebeke konsepti çok sayıda teknolojiyi, son kullanıcılar için çözümünü bir araya getirir ve çok sayıda ilke ile düzenleyiciyi (kural koyucuyu) adresler. Tek bir açık tanıma sahip değildir.

Avrupa Teknoloji Platformuna göre Akıllı Şebeke:

“Akıllı Şebeke verimli, sürdürülebilir, ekonomik ve güvenilir elektrik kaynağı sağlanması için ona bağlı olan kullanıcıların - üreticilerin, tüketicilerin ve hem üretilen hem tüketenlerin - hareketlerinin akıllıca entegre edildiği elektrik ağıdır [16].”

Tanımlamalar ve terminoloji değişse de, her ne kadar “Smart,” “smart,” “smarter,” veya “supersmart” şeklinde isimlendirilse de, bütün bu gösterimler çok çeşitli bilgi işlem, iletişim teknolojisinin ve enerji teslimi altyapısıyla servislerin eklendiği ve entegre edildiği gelişmiş bir 21. Yüzyıl enerji şebekesini belirtir. Enerjinin iki yönlü akışı, iki yönlü iletişim ve kontrol yetenekleri ev ve işyerleri için akıllı sayaçlar ve ötesinde uygulama imkanları sağlayacaktır. Akıllı Şebeke’nin belirleyici özellikleri ise:

- Elektrik şebekesinin güvenilirliğini, güvenliğini ve verimini arttıracak şekilde artan sayısal bilgi ve kontrol teknolojilerinin kullanımı.
- Tam anlamıyla siber güvenliği sağlayarak; şebeke operasyonları ve kaynaklarının dinamik optimizasyonu.
- Yenilenebilir enerji kaynakları dahil dağıtık kaynakların ve üretimin kurulumu ve entegrasyonu.
- Talep cevap, talep tarafı kaynaklar ve enerji verimliliği kaynaklarının dahil edilmesi ve kurulumu.
- Sayaç ve şebeke operasyonlarıyla ilgili iletişim ile durum ve dağıtım otomasyonunun akıllı teknolojilerle kurulumu.
- Akıllı ev eşyaları ve tüketici cihazlarının entegrasyonu.
- Elektrik şebekesine bağlanan ev eşyaları ve ekipmanın şebekeyi sağlayan altyapıyı içerecek şekilde uyumluluk ve iletişim standartlarının oluşturulması.

Akıllı Şebeke bütün dünyada global olarak önem teşkil etmektedir. Birçok ülke bu konudaki araştırma ve çalışmalarını yerel kaynaklarıyla gerçekleştirmektedir. Dolayısıyla bütün bu Akıllı Şebeke çalışmalarının eşgüdümlü bir şekilde uluslararası birliktelik altında harmanlanması büyük önem taşımaktadır. Bu eşgüdümün en temel elemanı ise uluslararası standartların oluşturulmasıdır [17].

Uluslararası standartların oluşturulmasında NIST, ETSI, CEN, CEPRI gibi yerel kurum ve yükleniciler SGIP, SG-CG gibi uluslararası platform ve çalışma grupları altında eşgüdüm içerisinde standartların oluşturulmasında çalışmaktadır.

Akıllı Şebeke’de enerjinin ve iletişimin iki taraflı yapılması yenilenebilir enerji kaynaklarının daha etkili ve verimli bir şekilde sisteme entegre edilmesine olanak tanımaktadır. Bu sayede tüketiciler yenilenebilir enerji kaynaklarını kullanarak kendi elektrik enerjilerini üretebilir hatta şebekenin ihtiyacı halinde üretim fazlası enerjiyi şebekeye verebilir.

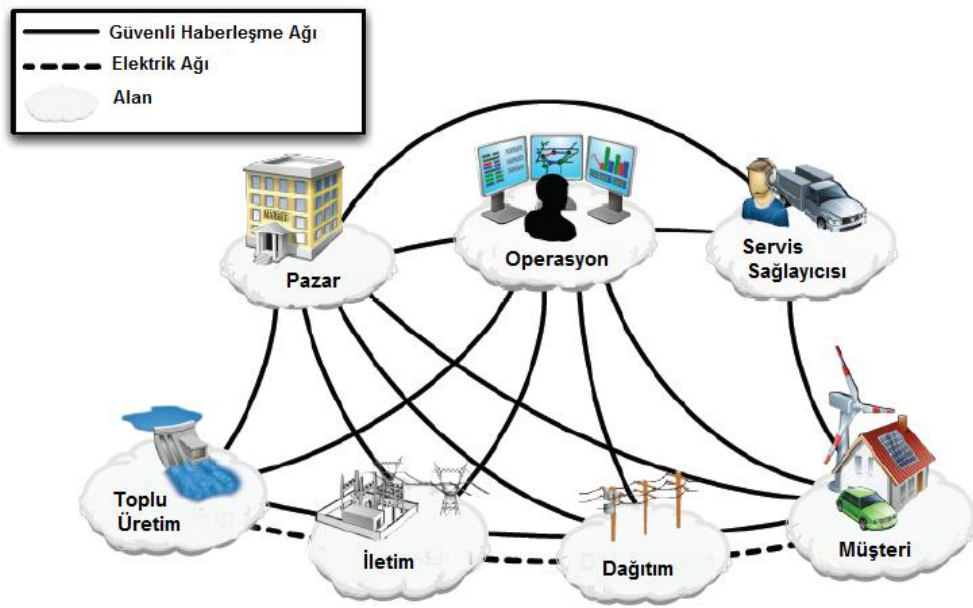
2.2.1 Akıllı Şebeke Konsept Referans Modeli

Akıllı Şebeke karmaşık sistemlerin bir araya gelerek oluşturduğu bütünleşik bir sistem olduğundan NIST tarafından farklı teknik alanların ayrıldığı konsept bir mimari referans model oluşturulmuştur. Bu mimaride NIST Akıllı Şebeke’yi 7 farklı alana bölme yaklaşımını kabul etmiştir. Bu teknik alanlar her ne kadar ayrılmış olsa da Akıllı Şebeke altında hepsi birlikte bir bütün olarak çalışmalıdır. Böylesine büyük ölçekli farklı parçalardan oluşan bir sistemde uyumluluk sağlanabilmesi için böyle bir mimari rehber ihtiyacı vardır.

Her bir alan ve onun alt alanı Akıllı Şebeke aktörlerini ve uygulamalarını ihtiva eder. Aktörler akıllı sayaçlar, güneş enerjisi jeneratörleri ve kontrol sistemleri gibi cihazları ve sistemleri, programları ve gerekli uygulamaları gerçekleştirmek üzere karar veren ve bilgi alışverişi yapan hisse sahiplerini kapsar. Uygulamalarsa alan içerisinde bir veya daha fazla aktör tarafından gerçekleştirilen görevlerdir. Uygulamalara örnek olarak ev otomasyonu, güneş enerjisi üretimi ve enerji depolama ile yönetimini verebiliriz.

Genel olarak ayrı alan içindeki aktörler benzer amaçlara sahiptir. Yine de aynı alan içindeki haberleşmeler farklı nitelikler gösterebilirler ve uyumluluğun elde edilmesi için farklı gereksinimleri karşılayabilirler.

Akıllı Şebeke'yi mümkün kılmak için farklı alandaki aktörler diğer alanlardaki aktörlerle etkileşebilirler ve dahası belli alanlar diğer alanların bileşenlerini içerebilirler [17]. Şekil 2.15'te NIST'in kabul ettiği mimari konsept modeli ve bu modeldeki alanların birbiriyle etkileşimi resmedilmiştir.



Şekil 2.15 Akıllı Şebeke Mimari Konsept Modeli [17]

2.2.1.1 Pazar

Pazar şebeke değerlerinin alınıp satıldığı yerdir. Oluşturulan Pazar Akıllı Şebeke'nin geleceğini tanımlamada yararlı olacaktır. Pazar alanı içerisindeki aktörler enerji sistemi dahilinde fiyat alışverişi yapar ve talep ile tedariği dengelerler. Pazar alanının sınırları kontrolün olduğu Operasyon alanı, değerleri sağlayan alanlar (Toplu Üretim, İletim vs.) ve Müşteri alanının sınırlarını kapsar [17].

Pazar alanı enerji servislerinin pazar yönetimini, toptan satışını, perakende satışını ve ticaretini sağlar. Pazar alanı diğer bütün alanlarla arayüz oluşturur ve onların rekabetçi pazar ortamı içerisinde eşgüdümlü çalışmalarını garantiler [18].

2.2.1.2 Servis Sağlayıcısı

Servis Sağlayıcısı alanındaki aktörler enerji sistemi üreticileri, dağıtıcıları ve müşterilerinin ticari süreçlerini destekleyecek servisler sağlarlar. Bu ticari süreçler faturalandırma ve müşteri hesabı yönetimi gibi klasik servislerden, enerji kullanımı yönetimi ve evde enerji üretimi gibi gelişmiş müşteri servislerine kadar değişen bir aralıktadır.

Servis sağlayıcısı var olan veya gelişen servisleri sağlarken elektrik enerjisinde siber güvenlik, güvenilirlik, süreklilik, bütünlük ve emniyet açısından ödün vermemelidir.

Servis sağlayıcılar gelişen Akıllı Şebeke'nin yeni ihtiyaçları ve fırsatlarını karşılayacak şekilde yenilikçi servisler ve ürünler geliştirecektir. Ticari modellere göre servisler elektrik servis sağlayıcısı, var olan taşeronlar veya yeni katılımcılar tarafından gerçekleştirilebilir [17].

2.2.1.3 Operasyon

Operasyon alanındaki aktörler elektrik sisteminin düzgün operasyonundan sorumludur. Bugün bu fonksiyonların çoğunluğu yetkilendirilen hizmet sağlayıcısının sorumluluğundadır. Akıllı Şebeke bu fonksiyonlarının çoğunluğunun servis sağlayıcılarına (taşeron olarak) verilmesini mümkün kılacaktır. Servis Sağlayıcısı ve Pazar alanlarının gelişimden bağımsız olarak da servis sağlama noktasında planlama ve operasyon için temel fonksiyonlara ihtiyaç olacaktır.

İletim operasyonlarında Enerji Yönetim Sistemleri enerji iletim sisteminin güvenilir ve verimli bir şekilde çalışması ve analiz edilmesi için kullanılırken, dağıtım operasyonlarında benzer olarak dağıtımın yönetilmesi ve analiz edilmesi için Dağıtım Yönetim Sistemi kullanılacaktır [17].

2.2.1.4 Toplu Üretim

Toplu Üretim alanındaki uygulamalar müşterilere elektriğin sağlanmasındaki ilk süreçlerdir. Elektrik üretimi diğer çok çeşit enerji türlerinden elektriğin oluşturulduğu süreçtir. Toplu Üretim alanının sınırı İletim alanıdır. Toplu Üretim alanı elektriksel açıdan İletim alanına bağlanmıştır ve Operasyon, Pazar ve İletim alanlarıyla arayüzler paylaşır [17].

2.2.1.4 İletim

İletim elektrik enerjisinin birçok trafo aracılığıyla üretim kaynaklarından dağıtım toplu transferidir. Bu alanın aktörlerine örnek olarak trafo sayaçları, güç kalite monitörlerini, faz hesaplama ünitelerini ve trafo kullanıcı arayüzlerini gösterebiliriz [17].

2.2.1.5 Dağıtım

Dağıtım alanı İletim alanı, Müşteri alanı ve sayaç noktaları arasındaki elektriksel bağlantıdır. Elektriksel dağıtım sistemi çok farklı şekillerde oluşturulabilir: radyal, döngüsel ve örgüsel. Dağıtım sisteminin güvenilirliği kurulmuş olan yapıya, içerdiği aktörlere ve onların birbirleriyle ve diğer alanlardaki aktörlerle haberleşme seviyesine bağlı olarak değişmektedir.

Tarihsel olarak dağıtım sistemleri radyal konfigürasyona ve çok az bilgi toplama mekanizmasına (telemetry) sahip, neredeyse bütün iletişimin insanlar aracılığıyla yapıldığı bir sistemdir. Bu alanda kurulan ilk algı merkezi telefonu olan bir müşteridir ve onun çağrısıyla enerjinin tekrar verilmesi için sahaya bir ekip gönderilir. Her ne kadar şu anda her iki yönde çalıştığı kabul edilse de bu alandaki birçok iletişim arayüzü hiyerarşik ve tek yönlüdür. Hatta elektriksel bağlantılar da çift yönlü akışı desteklemeye başlamıştır [17].

2.2.1.6 Müşteri

Bu alan elektriğin tüketildiği alandır. Müşteri alanındaki aktörler müşterilerin enerji tüketim ve üretimlerini yönetebilmelerini sağlar. Bazı aktörler ayrıca müşteri ve diğer alanlar arasında bilgi ve kontrol veri akışını sağlar. Müşteri alanının sınırları hizmet sağlayıcı sayacı ve Enerji Servis Arayüzü olarak değerlendirilebilir. Enerji Servis Arayüzü hizmet sağlayıcısı ile tüketicinin etkileşimi için güvenli bir arayüz oluşturur. Enerji Servis Arayüzü, Bina Otomasyon Sistemi veya Enerji Yönetim Sistemi gibi bina temelli sistemlere erişimde köprü görevi yapabilir.

Müşteri alanı genel olarak ev, ticari/bina ve endüstriyel olarak alt alanlara ayrılmıştır. Bu alt alanlar enerji kullanımlarına istinaden şekillendirilmiştir. Her bir alt alan birçok aktör ve uygulamaya sahiptir ve bunlar diğer alt alanlarda da kullanılabilir. Her bir alt alan bir sayaç ve sayacın üzerinde, Enerji Yönetim Sistemi içerisinde veya bir uç cihazda yer alabilecek Enerji Servis Arayüzü'ne sahiptir. Enerji Servis Arayüzü Müşteri

alanının ana/birincil servis arayüzüdür. Enerji Servis Arayüzü diğer alanlarla Gelişmiş Sayaç Altyapısı aracılığıyla veya Internet gibi başka yapılarla haberleşebilir. Enerji Servis Arayüzü müşteri bölgesindeki cihazlara ve sistemlere direk olarak HAN veya LAN aracılığıyla arayüz sağlar [17].

2.3 AKILLI ŞEBEKE'DE OTURUM BAŞLATMA PROTOKOLÜNÜN KULLANIMI

Uygulama katmanı protokolü olan SIP ile Akıllı Şebeke alanında yapılan çalışmalar [19, 20, 21, 22, 23] daha çok Akıllı Şebeke Konsept Modeli'nin Müşteri alanı içerisinde yer alan uygulamalardan ev/bina otomasyonu, Gelişmiş Sayaç Altyapısı ve PHEV konuları üzerinde yoğunlaşmaktadır.

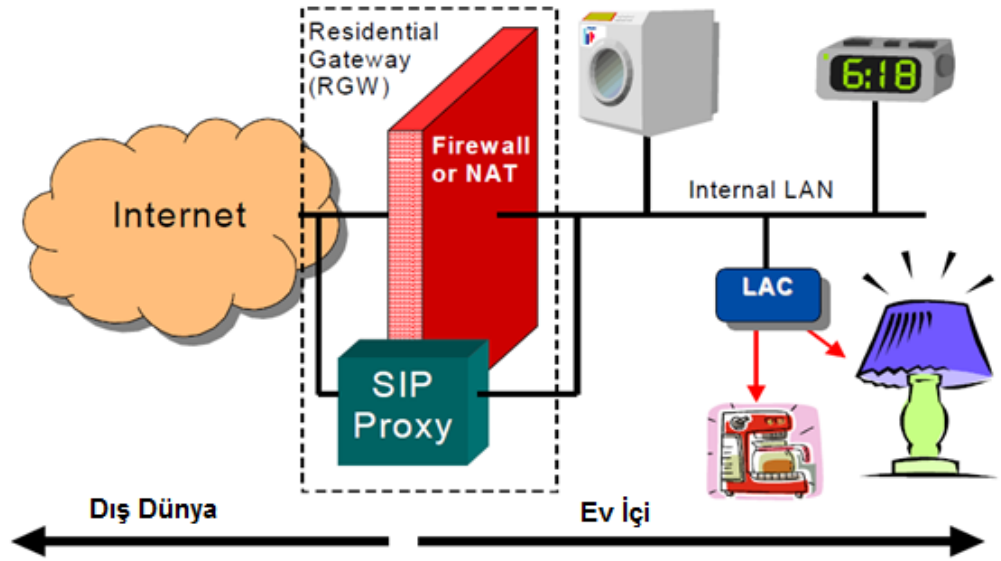
Operasyon, Dağıtım ve Müşteri alanlarının ortak kapsama alanına giren Otomatik Talep-Cevap sistemlerinde SIP kullanımına yönelik öneriler de [24] yapılmıştır.

2.3.1 Ev/Bina Otomasyonu

SIP ile ev otomasyonu fikri Akıllı Şebeke çalışmalarından bağımsız olarak ilk versiyonu 2000 yılında ve en son versiyonu 2001 yılında yayınlanan ve bir ağ ile birbirine ve internete bağlanmış ev eşyalarının gündeme gelmesiyle sunulmuş olan internet taslağıyla [25] ortaya çıkmıştır.

Bu taslak [25] çalışmasında her bir ev eşyası ağ bağlantısına sahip, işlemcili tüketici cihazları olarak kabul edilmiştir. SIP ise bu tüketici cihazlarının birbirleri arasında ve internet aracılığıyla da dış dünya ile iletişimini sağlayacak protokol olarak önerilmiştir.

Çalışma dahilinde, iletişim ev içi ve ev dışı olmak üzere ikiye ayrılmıştır. Cihazlar için örnek SIP adreslemeleri paylaşılmış ve güvenlik sebebiyle bu adreslerin şifrelenmesi tavsiye edilmiş ve bir örnek paylaşılmıştır. Her bir cihaz ayrı birer SIP birimi olarak düşünülmüştür. Aynı zamanda Cihaz Mesajlaşma Protokolü isminde yeni bir MIME türü önerilmiştir. Şekil 2.16'da bu çalışmanın ele aldığı örnek mimari gösterilmektedir.



Şekil 2.16 Ağ Bağlantılı Eşyalar için SIP taslağındaki mimari [26]

Günümüzde kullandığımız ev eşyaları düşünüldüğünde birçok elektronik ev eşyası temel düzeyde programlanabilir elektronik ekipmanla donatılmış olsa da bu tarz bir veri işlemesi ve iletimi yapacak ürünler üretilmemiştir.

Kwak [19] ise yaptığı çalışmada SIP ile beraber X.10 kablosuz haberleşme protokolünü kullandığı bir model üzerinde çalışmıştır. Bu iki iletişim protokolünün beraber kullanımı ile yer bağımsız servisler elde etmeyi amaçlamıştır.

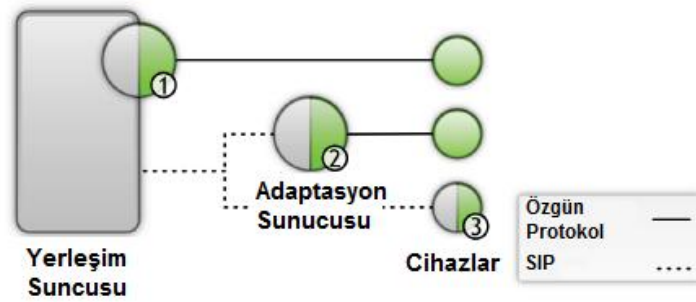
Kullanıcı aracı Yer Bağımsız Servis Yöneticisi ismi verilen birime kayıt olarak multimedia servisi, ev otomasyon servisi gibi servislere ulaşabilir. Yer Bağımsız Servis Yöneticisi cihaz listesini ve açıklama bilgisini her bir cihaz için yönetir ve cihaz listesi değiştiğinde güncellenmiş versiyonunu kullanıcılara gönderir. Bu cihaz listesi aynı zamanda servislerin listesini ve servislerin durum bilgisini de içerir.

SIP kullanıcılarını içeren servis birimi SIP isteği başlatır yada alınan bir SIP isteğine cevap verir. Bu servis birimi her statik gömülü cihazda ve kullanıcının mobil cihazında yüklüdür ve cihaz listesini yer bağımsız servis yöneticisinden alır.

Sonuç olarak SIP'in mobilite ve X.10 protokolünün cihaz kontrolü özelliklerini kullanarak yer bağımsız servisler elde edilmiştir [19].

Bertran ve diğ. [20] ev içerisindeki her eşyayı bir SIP birimi olarak düşünüp, ev alanı içerisindeki bir yerleşim sunucusu aracılığıyla ev otomasyonunun gerçekleştirilmesini önermişlerdir. Ayrıca kendine özgü protokollere sahip olup SIP desteklemeyen cihazlar için paketleyici yazılım katmanı üretmişlerdir. Bu yazılım katmanı, kendine özgü protokolü SIP'e dönüştürmektedir. Şekil 2.17 yerleşim sunucusunu ve diğer cihazları göstermektedir. Şekilden görülebileceği üzere, paketleyiciler yerleşim sunucusunun bir modülü olabilir, bir kaç cihaza servis sunan adaptasyon sunucusu üzerinde bulunabilir yada mümkünse direk cihazın üzerinde bulunabilir [20].

Çalışmada [20] SIP mesaj içeriğinde SOAP kullanarak ev otomasyonu için gerekli adaptasyonun sağlanması planlanmıştır. Senaryo dahilindeki işlemleri; kayıt ve keşif, komutlar, olaylar ve oturumlar olarak 4 gruba ayırmışlardır. Kayıt ve keşif amacıyla bu yaklaşımda birimler bir SIP proxy'e kayıt olmakta ve böylece URI/network bilgileri elde edilmektedir. URI bilgisi cihazın özellikleri ve durumu konusunda bir bilgi vermeyeceğinden OPTIONS yöntemi kullanarak cihazın özelliklerinin elde edilmesi önerilmiştir.



Şekil 2.17 Paketleyici Yazılım Katmanı [21]

Komutlar MESSAGE isteğiyle yapılmaktadır. MESSAGE isteklerinin mesaj içeriğinde SOAP mesajları taşınarak gerçekleştirilmektedir.

Olaylar başlığı altında SIP durum bildirimi modeli kullanılmış ve böylece belli olaylara üye olunarak; olayın gerçekleşmesi durumunda üyelerin olaydan haberdar edilmeleri gerçekleşmiştir.

Oturumlar ise multimedya gibi canlı veri akışı olacak durumlar için kullanılmıştır. Yaklaşımda oturumlar içinde SOAP mesajlaşması mesaj içeriği olarak kullanılarak genelleştirilmiştir [20].

Akıllı Şebeke içerisinde ev otomasyonu bölüm 2.2.1.6’da bahsedildiği gibi Enerji Servis Arayüzü ile etkileşim içerisinde. Bu servisler sayacın kendi üzerinde olabileceği gibi sisteme entegre başka bir cihaz üzerinde de yer alabilir.

2.3.2 Gelişmiş Sayaç Altyapısı

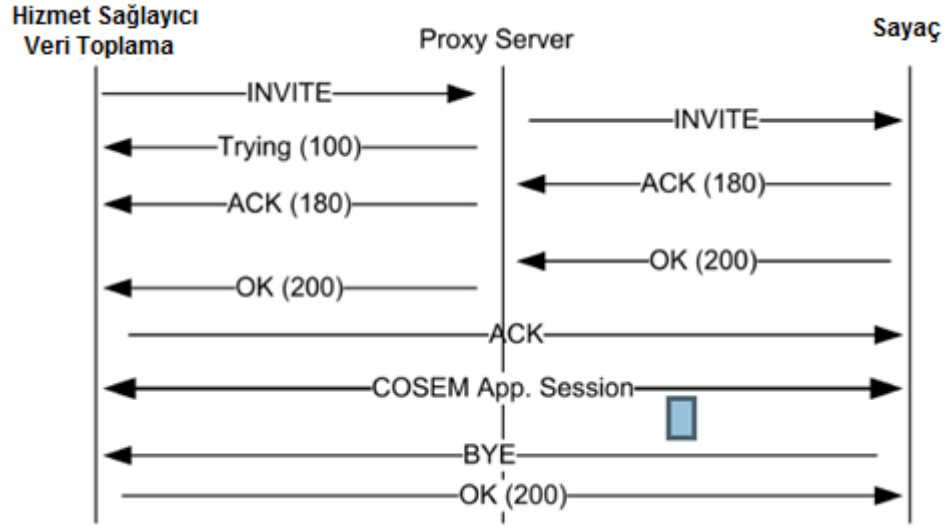
Gelişmiş Sayaç Altyapısı sayaçların uzaktan okunabildiği, Akıllı Şebeke’nin ihtiyacı gereği çift yönlü elektrik akışına izin veren ve buna göre hesaplamalar gerçekleştirebilen, dinamik fiyatlandırmalara olanak tanıyan teknolojidir.

Gelişmiş Sayaç Altyapısı’nda sayacın kendisi Müşteri alanı içerisinde bulunsa dahi Operasyon, Dağıtım gibi alanlarla etkileşim halindedir. Sayaç üzerinde Enerji Servis Arayüzü’nü barındırabilir.

Khalifa ve diğ. [21] DLMS/COSEM standardı ile SIP’in beraber kullanıldığı bir modeli otomatik sayaç okuma için önermişlerdir. DLMS/COSEM standardını kullanarak bir iletişim oturumu oluşturulabilir. Ancak, oturumların yönetilmesi (kurulması, değiştirilmesi ve sonlandırılması) diğer protokollerin kullanılmasını gerektirir. SIP, oturum kurulmasını ve olay üyeliği ile bildirim gerektiren her durumda tam anlamıyla kullanılabilir. AMR sistemi iletişim oturumlarının oluşturulmasını, değiştirilmesini ve sonlandırılmasını gerektirir. Böylece SIP sayaçlarla merkezi veri toplama sistemi cihazlarının iletişiminin kurulmasında potansiyele sahiptir ve veri toplama merkezi cihazları daha esnek ve iyi kontrol edilebilir.

Olası bir mesajlaşma SIP’in temel fonksiyonlarının gösterildiği şekil 2.18’de görülebilir. Uç noktalar (sayaç ve veri toplama cihazı), INVITE ve ACK mesajları

aracılığıyla oturum parametrelerini belirler. Proxy her iki taraftan gelen isteklerin iletilmesine yardım eder. Daha sonra COSEM uygulama oturumu başlar ve taraflardan birinden sonlandırma mesajı gelene kadar devam eder. Taraflardan herhangi biri SIP oturumunu başlatabilir ve bu AMR'a yüksek bir esneklik sağlar [21].



Şekil 2.18 AMR SIP mesajlaşması [22]

Wang ve Leung [22] IP temelli Akıllı Şebeke AMI ağı için SIP ve ANSI C12.22 protokollerini inceleyip karşılaştırmıştır. Bu karşılaştırma sonucunda ANSI C12.22 protokolünü SIP'e nazaran daha avantajlı olduğu düşüncesini belirtmişlerdir. Bunun sebebi olarak SIP'in firewall ve NAT'ların üzerinden geçerken oluşan sorunlar ve bu sorunların çözülürken getirdiği karmaşıklıktan dolayı olduğunu belirtmişlerdir [22].

2.3.3 Phev

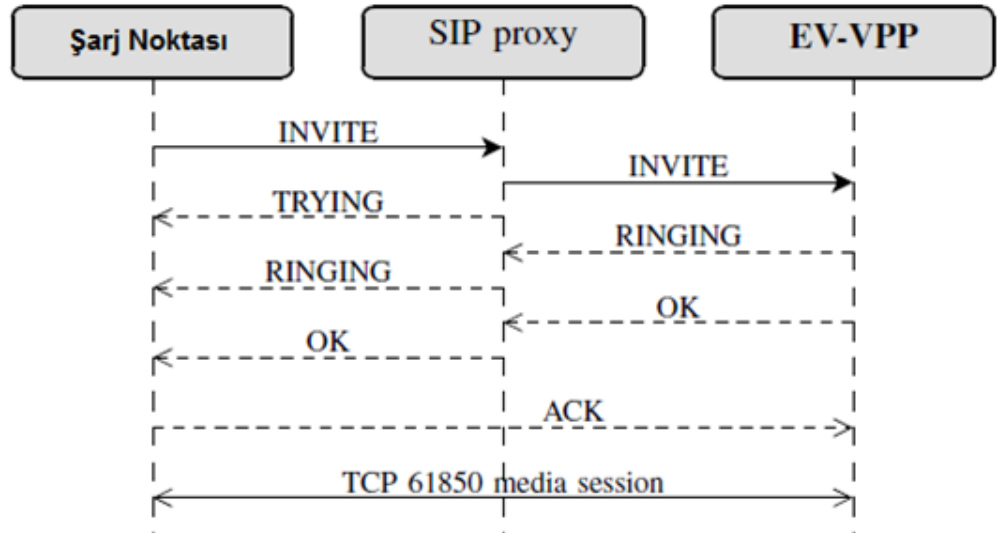
PHEV, Akıllı Şebeke içerisindeki şarj edilebilir elektrikli araçlardır. Son yıllarda bu tarz elektrikli araçların kullanımı ve üretimi artmıştır. Genel olarak araba türünde araçlar kullanılsa da artık kamyonet, kamyon, otobüs gibi büyük boyutlu araçların da şarj edilebilir elektrikli versiyonları üretilmeye başlanmıştır.

Jansen ve diğ. [23] çalışmalarında şarj edilebilir elektrik araçlarının enerji sistemine dahil edilmesine yönelik bir model önermişlerdir. Bu model içerisinde SIP şarj noktasıyla hizmet sağlayıcı arasında SIP oturumu kurulmasını ve bu oturum aracılığıyla

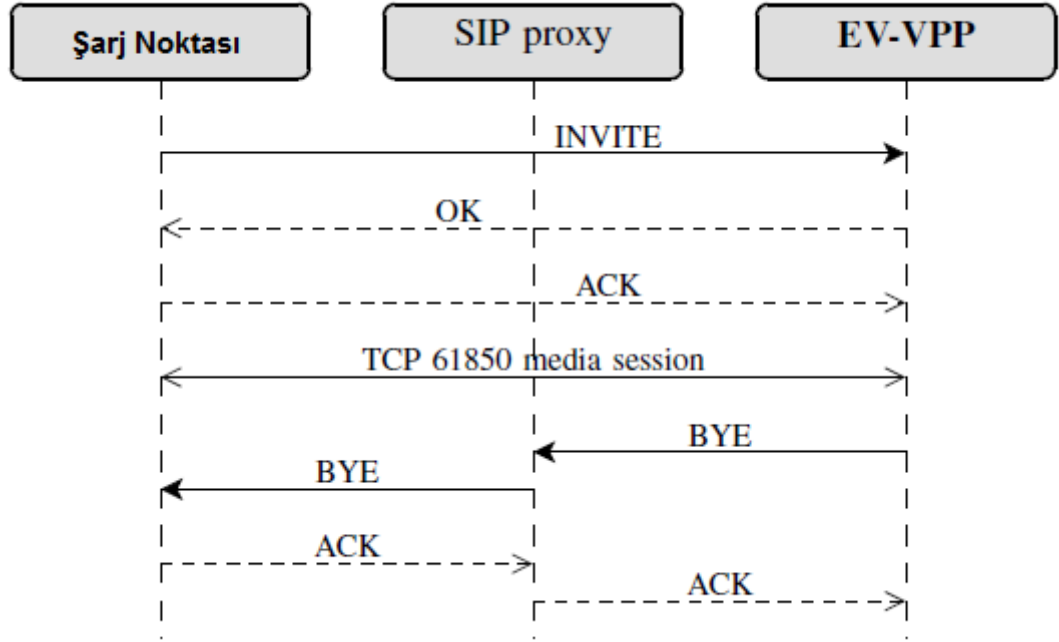
TLS/IEC 61850 veri akışının taraflar arasında gerçekleşmesi önerilmiştir. Burada firewall ve NAT probleminin aşılması için STUN yöntemi kullanılmıştır.

Şarj noktası kendisine bir elektrikli araç bağlandığında EV-VPP birimiyle oturum kurmak üzere bir INVITE isteği gönderir. Bu oturum şarj oturumu devam ettiği sürece açık kalacaktır. INVITE öncesi STUN tarafından algılanan NAT temelinde, SDP gövdesinde hangi tarafın şarj noktası ile EV-VPP birimi arasındaki TLS oturumunu başlatacağı tanımlanır. TLS oturumu yolculuk, pil ve sayaç bilgisi alışverişinin yanı sıra şarj etmenin başlatılması ve şarj süresinin iletilmesi için de kullanılır.

Şarj oturumunun bitirilmesi için elektrikli araç sürücüsü şarj noktasından TLS oturumu kurulması için EV-VPP'ye INVITE göndermesini talep etmelidir. Daha sonra oturumun bitirilmesi isteği güvenli olarak TLS haberleşmesiyle gerçekleşecektir. EV-VPP sayacı okuyacaktır ve elektrik bağlantısını kesecektir. Bütün sonlandırma görevleri tamamlandığında EV-VPP BYE isteği göndererek TLS oturumunun sonlandırılmasını sağlayacaktır [23]. Şekil 2.19'da oturumun kurulması ve Şekil 2.20'de oturumun sonlandırılmasına yönelik çağrı akışı gösterilmiştir.



Şekil 2.19 Şarj Oturumunun Kurulması [23]



Şekil 2.20 Şarj Oturumunun Sonlandırılması [23]

2.3.4 Otomatik Talep-Cevap

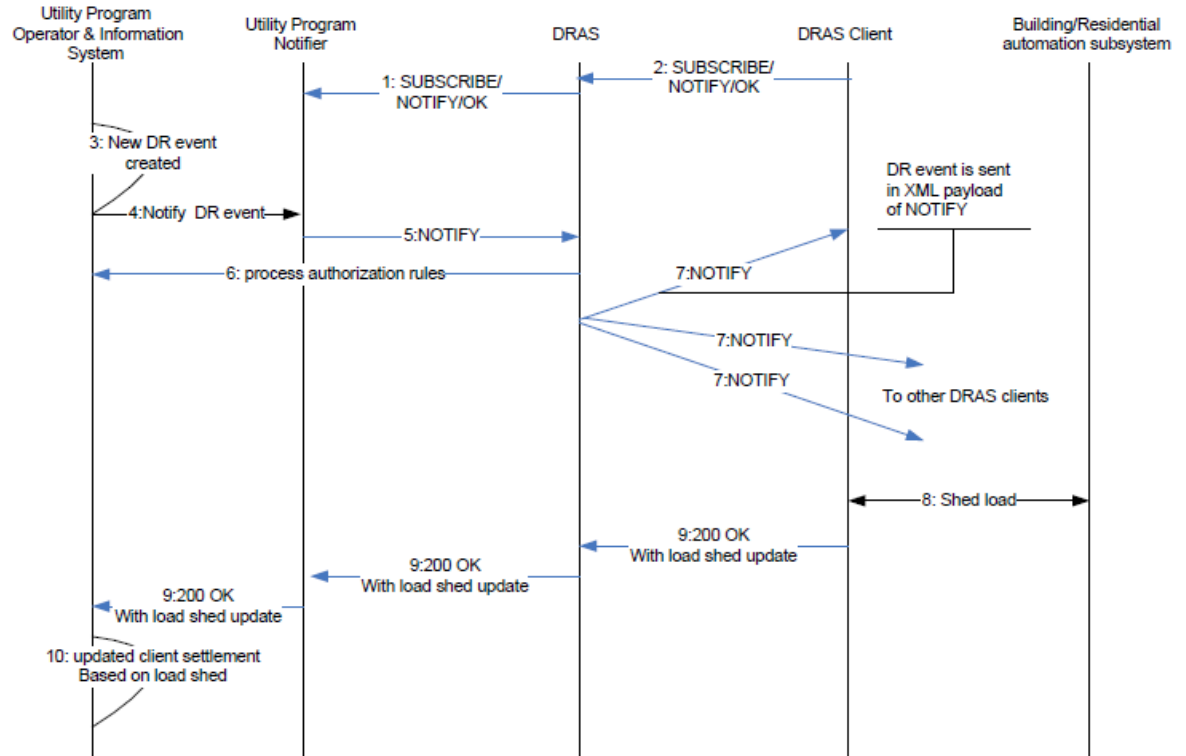
Bir Talep-Cevap programı kullanıcılara daha iyi enerji yönetimi/kullanımı için daha verimli dağıtım yapılmasını sunan kritik bir bileşendir ve tipik olarak elektrik dağıtım şirketleri tarafından talep edilir veya tasarlanır. Tipik katılımcılar küçük veya büyük şirketleri de kapsar. Bir Talep-Cevap programı elektrik için katılımcıların değişik faktörlere bağlı olarak bir dağıtım şirketine talebini arttırıp azaltabilecek şekilde bir dinamik fiyatlandırma şemasına uyar. Bu katılımcılar için tüketim maliyetlerini azaltmasını sağlarken elektrik dağıtım şirketlerine kendi maliyetlerini azaltmalarını ve elektrik dağıtımını daha iyi yönetebilmelerini sağlar.

Geleneksel olarak Talep-Cevap'ın birçok bileşeni olan manuel süreçlerdir. Genel olarak endüstriyel bir yönetici Talep-Cevap olaylarını izler ve manuel olarak maksimum kullanım fiyatlandırmaları belirtilerine tepki verir. Bu iş için ayrılmış personeli olmayan şirketlerde, gözleme işini yaparak sistem yükü tepe noktasına ulaşp kritik noktaya yaklaştığında müşteriyi kullanımı azaltmasına yönelik bilgilendirebilecek elektrik dağıtım şirketi ve müşteri arasında konuşlandırılmış bazı toplayıcılar bulunmaktadır. Bu

durumda bir kısım otomasyon olmasına rağmen halen daha çok fazla insan müdahalesi gerekmektedir. Sıklıkla, bilgilendirmeler müşterilere telefon yoluyla aktarılmaktadır. Dahası, Talep-Cevap için şu anda takip edilen bir standart bulunmamaktadır [24].

Talep-Cevap programlarının otomasyonu için Roychowdhury [24] SIP kullanmayı önermiştir. IETF SIMPLE çalışma grubu tarafından yapılan çalışmalarla SIP'e kazandırılan olay-durum modeli ve anlık mesajlaşma yeteneklerini kullanarak Talep-Cevap program dahiindeki olayların gerçekleşmesi durumunda olay bildirimlerinin SIP ile sağlanabileceğini göstermiştir.

İstemcilerin Talep-Cevap Otomasyon Sunucusu'na belli olaylar için gönderdikleri SUBSCRIBE isteğiyle bu olayların durumuna üye olmaları ve olay gerçekleştiğinde sunucunun otomatik olarak NOTIFY yöntemiyle istemcileri uyarması esasına dayanmaktadır [24]. Şekil 2.21 bir olay bildirimi için gerekli sinyalleşmeyi göstermektedir.



Şekil 2.21 Otomatik Talep-Cevap olay bildirimleri [24]

3. MALZEME VE YÖNTEM

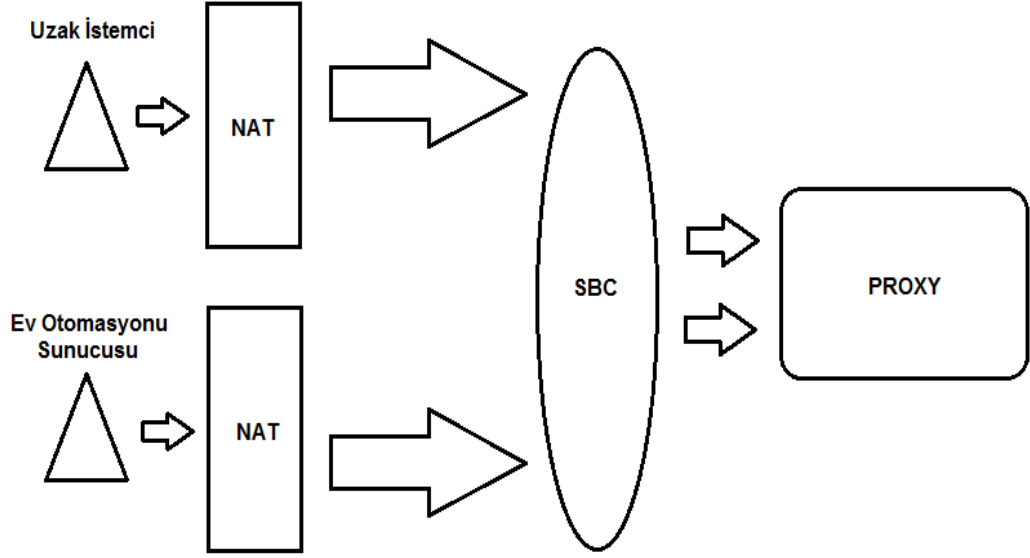
Ev/Bina otomasyonu NIST'in Akıllı Şebeke standartları referans modelinde [17] Müşteri alanının bir parçası olarak yer almaktadır. Bu modelde Müşteri alanında tanımlanan Enerji Servis Arayüzü, hizmet sağlayıcısı ile tüketicinin etkileşimi için güvenli bir arayüz oluşturur. Enerji Servis Arayüzü, Bina Otomasyon Sistemi veya Enerji Yönetim Sistemi gibi bina temelli sistemlere erişimde köprü görevi yapabilir. Ev otomasyonu hizmetini elektrik dağıtım hizmetini sağlayan kuruluş/şirket veya ayrı bir servis hizmeti sağlayan servis sağlayıcısı sağlayabilir.

Bu bölümde SIP tabanlı bir uzaktan ev otomasyonu ve olay/bildirimi simülasyonu ile gerçekleştirilmesi için tasarlanan yazılım ve sistemler anlatılacaktır.

3.1 SIP TABANLI UZAKTAN EV OTOMASYONU VE OLAY/DURUM BİLDİRİMİ SİMÜLASYONU YAZILIMI

SIP kullanılarak ev otomasyonu yapılmasıyla ilgili yapılan önceki çalışmalarda [19, 20, 25] ev alanı içerisinde olan bir ev sunucusu tanımı yapılmıştır. Bu çalışmada da ev sunucusu ve bu sunucudan otomasyon isteği yapan istemci ve istemcilerin yer aldığı bir topoloji benimsenmiştir. Bu topolojide hem güvenlik hem de NAT çözümlerinin başarıyla yapılabilmesi için SBC cihazları önerilmiş ve kullanıldığı varsayılmıştır. SBC'ler internet üzerinden ses taşınmasında son yıllarda büyük önem kazanmış ve sık olarak kullanılan cihazlardır. IETF SBC'lerin kullanımı ile ilgili RFC 5853'ü [26] yayınlamıştır. RFC 5853'te de belirtildiği üzere SBC erişim senaryosunda, vekil sunucusunun (proxy) önünde konumlandırılarak vekil sunucusu olası saldırılara karşı korunmuş olur. İstemciler SBC'nin erişim adresini "outbound proxy" adresi olarak tanımlayabilir ve vekil sunucusuna SBC aracılığıyla adres kayıtlarını gerçekleştirebilirler. SBC'ler günümüzde aynı zamanda NAT arkasından gelen istemcilerin çözümlerini de yapmaktadırlar. Böylelikle SBC NAT kaynaklı çıkabilecek problemler için Bertran ve diğ. [20] belirttiği çözümlere ek olarak gösterilebilecek bir çözümdür. Üstelik SBC'nin asıl amacı olan ve ev otomasyonu için

çok büyük önem taşıyan güvenlik de arttırılmış olacaktır. Bu çalışmada benimsenen uzaktan SIP kullanarak ev otomasyonu gerçekleyen sisteminin topolojisi şekil 3.1’de gösterilmiştir.



Şekil 3.1 SIP kullanarak ev otomasyonu topolojisi

Uzak istemci ve ev otomasyon sunucusu SBC aracılığıyla vekil sunucuya kayıt olur. Kayıt işlemi sayesinde vekil sunucu ve SBC, tarafların yerel adreslerini çözer ve kayıt altına alır. Bu andan itibaren her iki tarafın birbirine SIP URI’leri aracılığıyla göndereceği istekler SBC ve vekil sunucusu tarafından çözülerek sorunsuzca iletilecektir.

Bu çalışmada belirtilen mimarinin gerçekleştirilmesi için Genband S3 SBC [27] ve Kamailio SIP [28] sunucusu kullanılmıştır.

3.1.1 Sistem Mimarisi

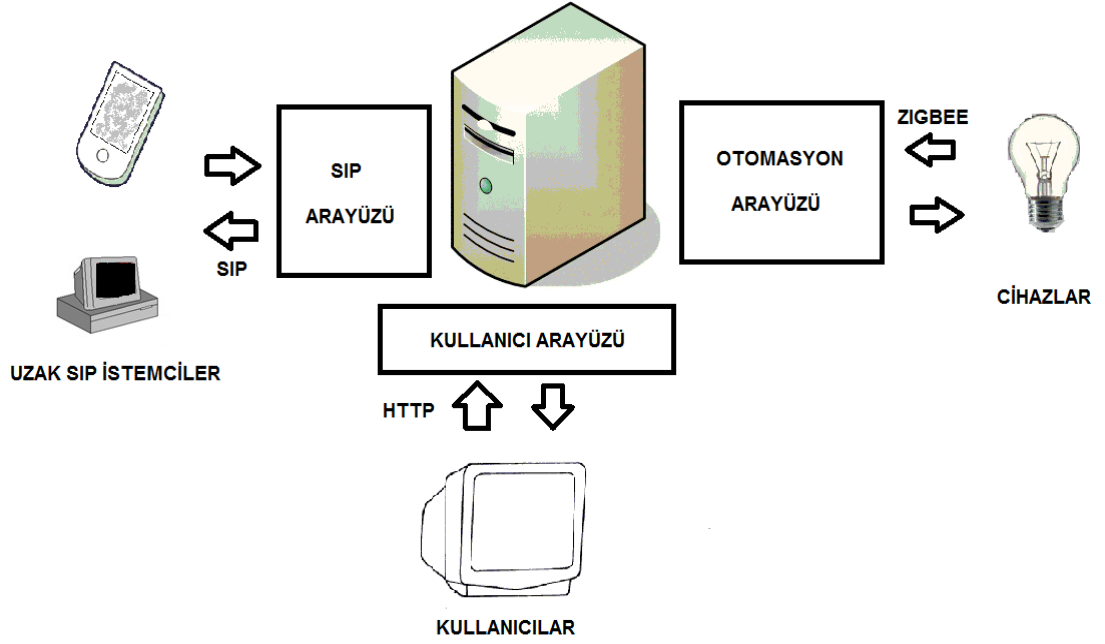
Ev içerisindeki cihazlar üzerinde otomasyon işlemleri ve cihazlara ait olay/durum takibinin yapılabilmesi için SIP sinyalleşmesi kullanılabilir. Kwak [19] ile Bertran ve diğ. [20] yaptıkları çalışmalarda belirttikleri üzere SIMPLE çalışma grubunun

tanımlamış olduğu anlık mesajlaşma ve olay/durum bildirimi modelleri bu kullanım için uygundur.

Ev içerisinde konuşlandırılmış bir otomasyon sunucusu evdeki cihazlara, onları yönetebileceği bir arayüz/platform üzerinden erişir. Bu arayüz cihazlar üzerinde çalışabilecek; aşırı elektrik tüketimine sebep vermeden cihazların ev sunucusundan aldığı komutları uygulayabileceği bir yapıda olmalıdır. Ev sunucusu ve cihazların arasındaki iletişim bu arayüzün kendine özgün protokollerini kullanabilir. Kwak [19] çalışmasında X.10 protokolünü kullanmışken, Tomic ve Teodorova [29] ZIGBEE protokolünü kullanmışlardır.

Sonuç olarak SIP kullanarak uzaktan otomasyon yapılmasını sağlayan bir ev otomasyon sunucusunun farklı arayüzleri olması gerekir. Bunlardan ilki SIP protokolünü kullanan uzak istemci ile haberleşecek SIP arayüzüdür. Bu arayüz gelen SIP isteklerine göre hareket eder ve bu isteklere cevap verir. Diğer arayüz evdeki cihazlarla iletişim kuracağı ve otomasyon işlemini gerçekleştireceği arayüzdür. Bu arayüzde bahsedilen X.10, ZIGBEE veya başka bir protokol kullanılabilir. Burada kullanılacak olan protokolün standartlaşmış ve evdeki farklı cihazlar tarafından desteklenecek şekilde olması gerekir. Bunların dışında otomasyon sisteminin kullanıcılarının arzu ettikleri işlemleri tanımlayabilecekleri veya engelleyebilecekleri, gerekli ayarları yapabilecekleri bir kullanıcı arayüzüne sahip olması gerekebilir. Gerçek bir ev otomasyon sunucusunun arayüzleri ve iletişim kurduğu noktalar şekil 3.2'deki gibi olabilir.

Bu simülasyonda ise uzak bir SIP istemcinin SIP istek yöntemlerini kullanarak ev otomasyon sunucusu aracılığıyla ev otomasyonunu gerçekleştirmesi simüle edilecektir. Bunun için bir kişisel bilgisayar üzerinde çalışabilecek istemci ve ev otomasyon sunucusu yazılımı yapılmıştır. İstemci ve bu istemcinin isteklerini algılayıp, isteklere göre arayüz üzerinde konuşlandırılmış bir ampul simülasyonunu yakıp söndürecek ev otomasyon sunucusu farklı kişisel bilgisayarlar üzerinde çalıştırılacaktır.



Şekil 3.2 Ev otomasyon sunucusunun arayüzleri

3.1.2 Yazılım Mimarisi

Bu bölümde gerçekleştirilen simülasyonun yazılım mimarisi iki temel yapıdan oluşmaktadır. SIP mesajlarını iletim katmanından alıp işleyen “SIP stack” ve SIP stack’ten aldığı bilgileri ekrana döken arayüzdür. Her iki bileşenin yazımında da Java programlama dili kullanılmıştır.

“SIP stack” yazılım şirket ve kuruluşlarının çalışmalarıyla IETF’in yayınladığı RFC’ler rehberliğinde hazırlanmış program yığınlarıdır. Bu program yığınları içerisinde tanımlanan fonksiyonlar sayesinde uygulama yazıcılara kolaylık sağlar. Bu simülasyonda Java programlama dili için yazılmış “JAIN SIP” [30] API kullanılmıştır.

Simülasyonun arayüzünün oluşturulmasında ise Java Swing [31] ve AWT [32] API bileşen ve fonksiyonlarından faydalanılmıştır. Swing ve AWT API’leri arayüz ekranı, yazı girdi ve çıktıları ile tanımlanan fonksiyonları tıklandığında çağırarak butonların gerçekleşmesinde kullanılmaktadır.

Hem istemci hem de sunucu yazılımı üç sınıftan oluşmaktadır. İstemci “SIPAutoClient” paketi SIPLayerAutoClient, MessageProcessorClient ve SIPAutoClient sınıflarından oluşur. Sunucu ise benzer şekilde “SIPAuto”paketi SIPLayerAuto, MessageProcessor ve SIPAuto sınıflarından oluşur.

3.1.2.1 İstemci

SIPAutoClient sınıfı arayüzün oluşturulması için Swing API’sinin bir sınıfı olan JFrame’den türetilmiştir ve MessageProcessorClient arayüzünü uygular. SIPAutoClient ana fonksiyonu içerir ve programın başlatılmasında kullanıcı tarafından girilmesi gereken kullanıcı adı ve port bilgisini şart koyar. Kullanıcı bu bilgileri program başlatılırken sağlar ve sınıf SIPLayerAutoClient sınıfını bu değişkenlerle çağırır. Böylece istemcinin kullanıcı adı, IP ve port değerleri SIPLayerAutoClient sınıfına yani SIP katmanına iletilmiş olunur.

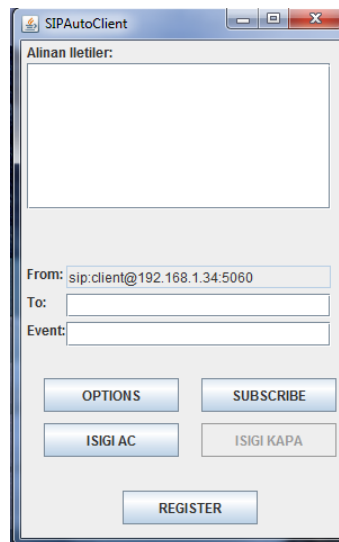
SIPLayerAutoClient sınıfı JAIN SIP API’sinin bir arayüzü olan SipListener arayüzünü uygular. Böylelikle API’nin SIP istek ve yanıtlarını oluşturan ve algılayan fonksiyonlarını kullanır. MessageProcessorClient arayüzünü çağırarak gelen SIP isteklerinin kimden geldiğini ve mesaj içeriklerini yayınlar. MessageProcessorClient arayüzünü uygulayan SIPAutoClient sınıfı böylelikle isteklerin kimden geldiği ve mesaj içeriklerinin ne olduğu bilgisine ulaşır. Böylece bu bilgileri ekrana yazdırabilir veya bu bilgilere istinaden başka fonksiyonları çağırabilir.

Sınıfların hiyerarşik yapısını gösteren UML diyagramı şekil 3.3’te gösterilmiştir.

Program başlatıldığında arayüz ve “SIP stack” başlatılır. “SIP stack” belirtilen porta gelen istek ve cevapları dinler. Arayüz ise isteklerin kime gönderileceğinin ve hangi olaya üye olunacağını belirtmesi gereken iki adet metin girişi alanına sahiptir. Bu metin alanlarına yazılan bilgi, istek gönderimi fonksiyonları çağrıldığında SIP katmanına gönderilir. Böylece SIP katmanı bu bilgilere istinaden istekleri gönderir. Arayüz üzerinde bulunan butonlar da SIP isteği gönderen ilgili fonksiyonları çağırır. İstemci üzerinde beş adet buton bulunmaktadır. Bunlar OPTIONS, SUBSCRIBE, AMPUL AC, AMPUL KAPA ve REGISTER şeklinde isimlendirilmişlerdir. AMPUL AC ve AMPUL KAPA butonlarından biri daima aktif değildir. Bu butonların aktifliği ev otomasyon sunucusu üzerindeki ampulün durumuna göre değişmektedir. OPTIONS



Şekil 3.3 İstemci sınıf hiyerarşisi UML diyagram gösterimi



Şekil 3.4 İstemci arayüzü

butonu ev otomasyon sunucusunun sorgulanması için kullanılırken SUBSCRIBE sunucu üzerinde tanımlanmış bir olaya üye olmak için kullanılır. AMPUL AC ve AMPUL KAPA butonları sunucu üzerindeki ampulün açılmasını ve kapanmasını sağlar. REGISTER butonu ise istemcinin vekil sunucuya alan adıyla kayıt olması için kullanılır. Bunların dışında istemci gelen istek ve yanıtların ekrana yansıtıldığı bir metin çıktı bölümünü içermektedir. Şekil 3.4'te istemci arayüzünün ekran çıktısı gösterilmektedir.

İstemci yazılımına ait Java kodları ise Ek-A içerisinde paylaşılmıştır.

3.1.2.2 Ev Otomasyon Sunucusu

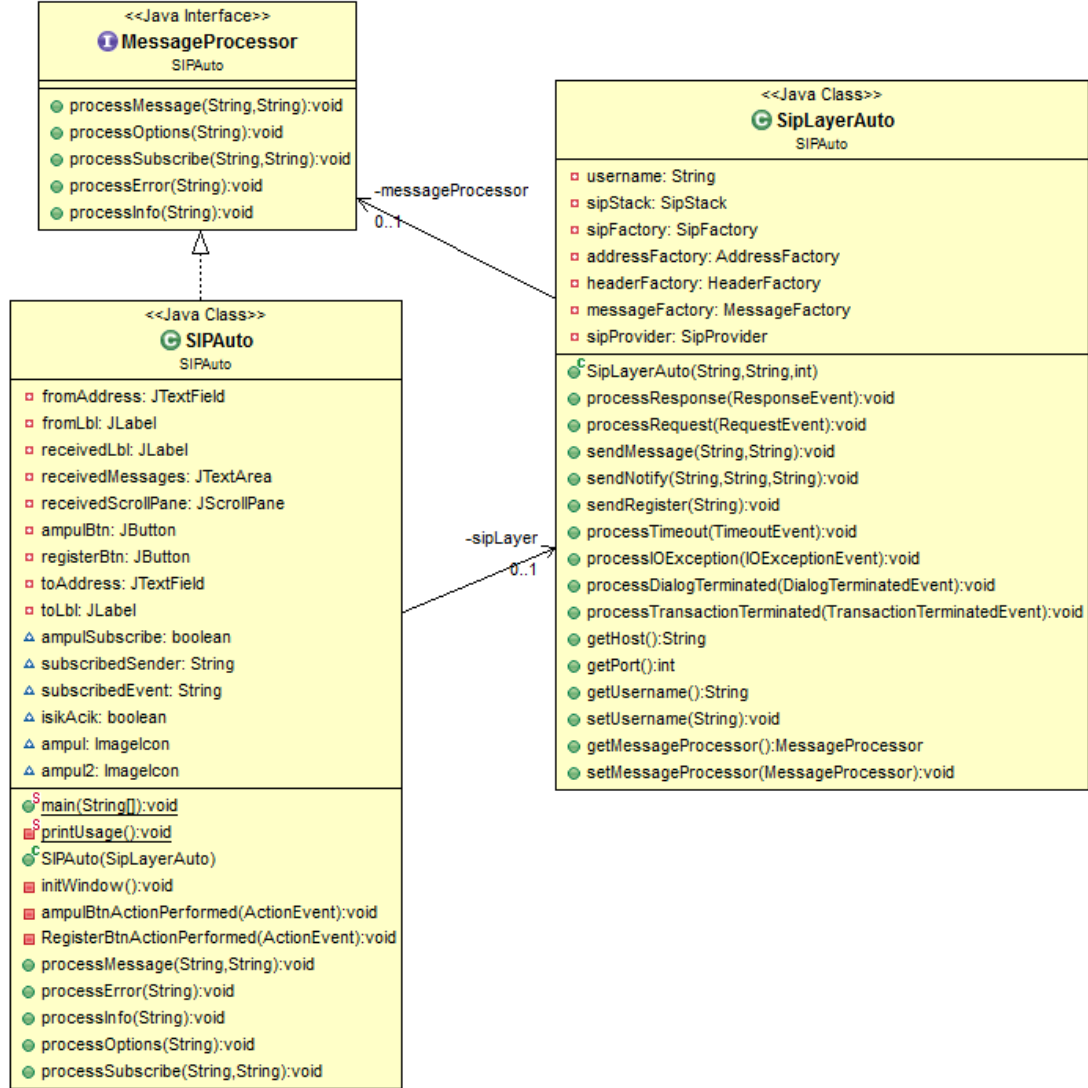
Ev otomasyon sunucusunun yazılım mimarisi de istemciyle aynı yapıya sahiptir. İstemci gibi üç sınıftan oluşmaktadır. Bu sınıflar SIPAuto, MessageProcessor ve SIPLayerAuto olarak isimlendirilmişlerdir. Yazılım temelde SIP katmanı ve bir arayüzden oluşmaktadır. İstemciden farklı olarak sunucu sınıfları bazı farklı fonksiyonlar kullanmakta ve SIP katmanından aldığı bilgilere göre farklı aksiyonlar gerçekleştirmektedir. Sınıfların hiyerarşik yapısı ve birbirleriyle etkileşimleri istemci ile aynıdır. Şekil 3.5 ev otomasyon sunucusunun sınıflarının hiyerarşik yapısını UML diyagramı şeklinde göstermektedir.

Sunucu arayüzünde gelen istek ve yanıtların ekrana yansıtıldığı bir metin çıktı bölümü bulunmaktadır. Bu bölümün altına büyük bir buton konuşlandırılmıştır. Bu butona iki adet resim eşlenmiştir. Bu resimlerden ilki sönmüş durumda olan bir ampulü, ikincisi ise yanar durumda olan bir ampulü tasvir etmektedir. Sunucu istemciden gelen isteklere istinaden bu butonla eşlenen resmi değiştirerek otomasyon işlemini simüle etmektedir. Ayrıca butona tıklayarak da butonla eşlenen ampul resimleri değiştirilip ev içerisindeki bir kullanıcının cihaza direk etkisi simüle edilir. Sunucunun vekil sunucuya kayıt olacağı alan adının girilmesi için bir metin girdi alanı bulunur. Bu alanın üstüne kayıt isteğinin tetiklenmesini sağlayan bir buton yerleştirilmiştir. Buton REGISTER şeklinde isimlendirilmiştir. Sunucunun kayıt olması gereken alan adı metin girişi bölümüne yazıldıktan sonra bu buton kullanılarak otomasyon sunucusunun vekil sunucuya kaydı gerçekleştirilebilir. Ev otomasyon sunucusu arayüzü Şekil 3.6'da gösterilmektedir.

Ev otomasyon sunucusuna ait Java kodları ise Ek-B içerisinde paylaşılmıştır.

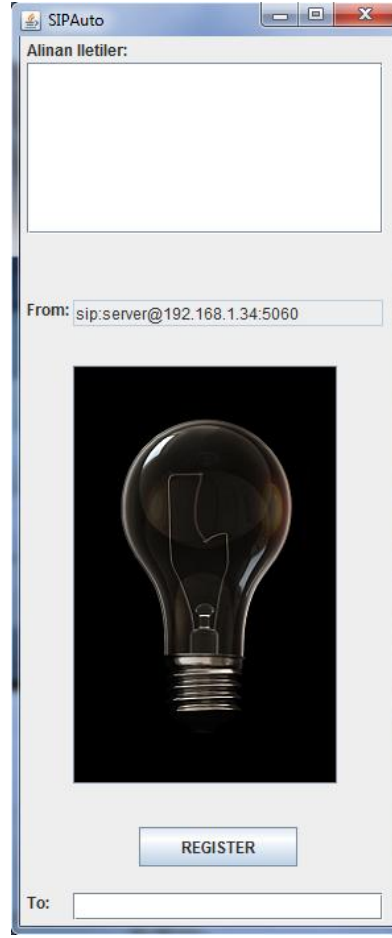
3.1.3 SIP Sinyalleşmesi Mimarisi

İstemci ve ev otomasyon sunucusunun birbirlerine SIP URI'leri aracılığıyla SIP istekleri gönderebilmeleri için öncelikle vekil sunucuya kayıt olmaları gerekmektedir. Bu çalışmada güvenlik açısından daha doğru bir yapı sağlamak amacıyla SIP isteklerinin SBC aracılığıyla vekil sunucuya iletilmesi önerilmiştir.



Şekil 3.5 Ev otomasyon sunucusu sınıf hiyerarşisi UML diyagram gösterimi

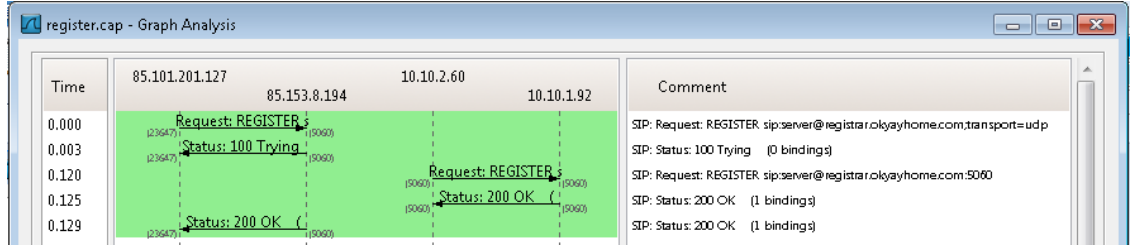
Şekil 3.7 Wireshark [33] paket analiz programı aracılığıyla ev otomasyon sunucusunun vekil sunucuya kaydı sırasında SBC üzerinde toplanan paketler için çizdirilen sinyalleşme akışı göstermektedir.



Şekil 3.6 Ev otomasyon sunucusu arayüzü

Ev ortamında otomasyon işlemini düşündüğümüzde dikkate almamız gereken en önemli konulardan biri ortamın dinamik oluşudur. Ev ortamını uzaktan yönetebilmek için önceden yönetebileceğimiz cihazları ve bu cihazların o anki durumunu bilmemiz gerekmektedir. Ev içerisindeki kullanıcılar cihazları o anda değişik fonksiyonları aracılığıyla kullanıyor olabilir. Bu sebeple uzaktan yapılacak bir otomasyonun öncesinde cihazın çalışıp çalışmadığına ve cihazın yetenekleri ile o anki durumuna yönelik bir bilgiye sahip olunması gerekir.

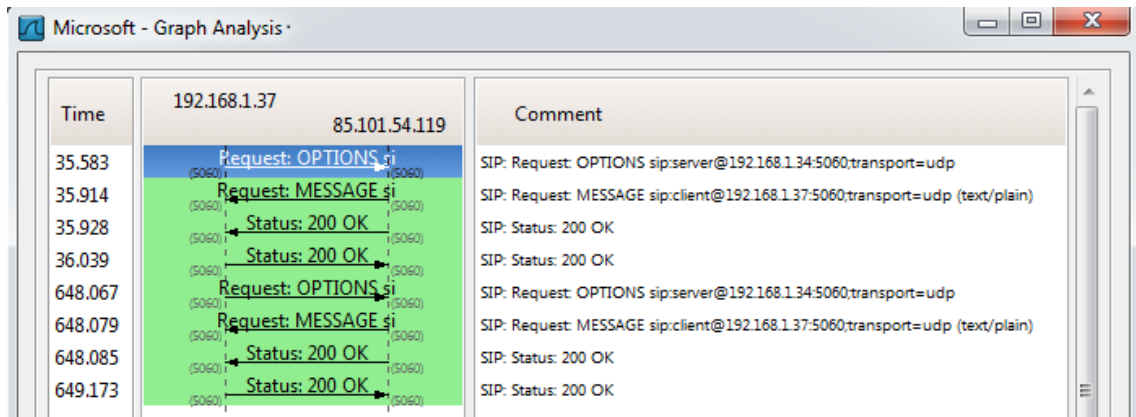
Bir kullanıcı aracının, başka bir kullanıcı aracının veya bir vekil sunucusunun yeteneklerini sorgulaması için OPTIONS isteği önerilmiştir. Bu sayede kullanıcı aracı iletişim kuracağı noktanın desteklediği SIP istek yöntemleri, mesaj içerikleri veya medya türlerini öğrenebilir [8]. OPTIONS alan bir kullanıcı aracı bu bilgileri verdiği yanıt içerisinde paylaşır.



Şekil 3.7 Vekil sunucuya kayıt sinyalleşmesi akışı

Uzak istemciler OPTIONS isteğini ev otomasyon sunucusundan ev içerisinde otomasyon kapasitesi olan cihazların bir listesini, şu anki durumları ve yetenekleriyle beraber paylaşmasını istemek için kullanabilir. OPTIONS isteğini alan bir ev otomasyon sunucusu bu bilgileri OPTIONS isteğine verdiği 200 OK cevabında paylaşabilir yada 200 OK cevabını isteği aldığını ve onayladığını belirtmek için kullanır, hemen akabinde istemciye bu bilgileri MESSAGE isteğinin mesaj içeriğinde gönderebilir. Bu bilgiler düz metin, XML veya farklı şekillerde kodlanmış olabilir. Bu tez çalışmasında düz metin şeklindeki mesaj içerikleri kullanılmıştır.

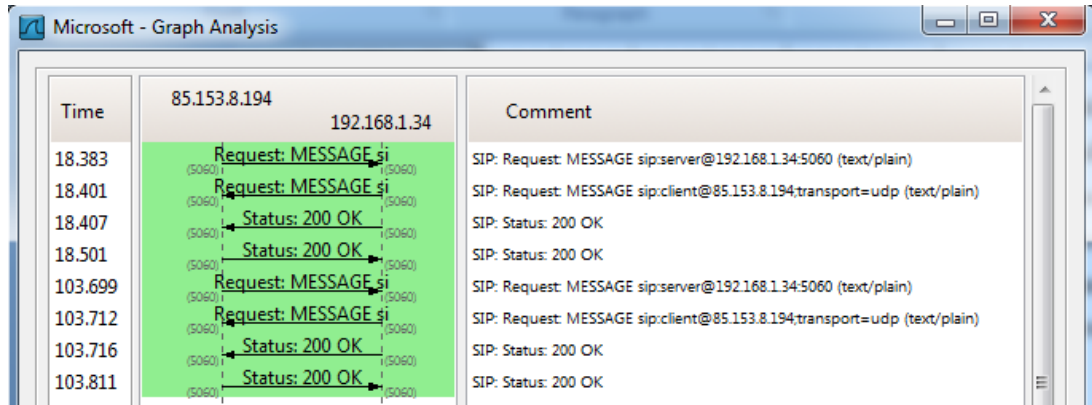
Şekil 3.8’de Wireshark [33] paket analiz programıyla istemcinin ve ev otomasyon sunucusunu sorgulama amacıyla gönderdiği OPTIONS isteğiyle gerçekleşen örnek bir sinyalleşmeye ait yakalanan SIP paketlerinin akışı çizdirilmiştir.



Şekil 3.8 İstemci ve sunucu arasındaki örnek sorgulama sinyalleşmesi akışı

Sunucunun kontrol edebildiği cihazların listesini, yeteneklerini ve şu anki durumunu alan istemci bu bilgiler ışığında MESSAGE isteğini kullanarak otomasyon isteğini gönderebilir. Bu isteğe ait bilgiler mesaj içeriğinde taşınacaktır. Bu durumda istemci sunucuya MESSAGE isteğiyle otomasyon talebinde bulunacak, sunucu bu talebi yerine getirmek istiyorsa işlemi gerçekleştirip 200 OK yanıtını verecektir. Eğer işlemin gerçekleştirilmesiyle ilgili bir sorun varsa veya sunucunun bu isteği reddetmesi gerekiyorsa uygun SIP yanıtlarından birini kullanabilir.

Wireshark paket analiz programıyla örnek bir otomasyon talebi için otomasyon sunucusu üzerinde yakalanan SIP paketlerinin sinyalleşme akış grafiği Şekil 3.9'daki gibi olmaktadır.



Şekil 3.9 Örnek otomasyon sinyalleşmesi akışı

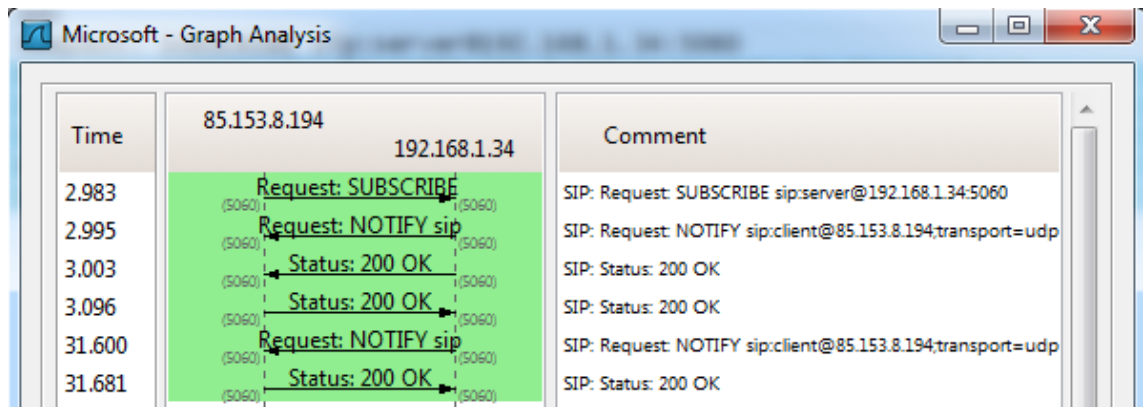
İstemci otomasyon gerçekleştirdiği cihazların son durumundan haberdar olmak isteyebilir. Ev ortamının dinamik bir ortam olması ve ev içerisindeki kullanıcıların cihazların durumlarına direk etki edebilmelerinden dolayı uzak istemcinin cihazların son durumuyla ilgili bilgi alması faydalı olacaktır. IETF tarafından yayınlanan RFC 3265 [14] ile SIP protokolü olay/durum bildirimi için bir yapıya sahip olmuştur. Bu yapı istemcinin evdeki herhangi bir cihaz veya olayı takip etmesi için kullanılabilir.

Bu yapıda herhangi bir olayın değişimlerinden haberdar olmak isteyen bir kullanıcı aracının olaya üye olması gerekir. Bunun için olayın gelişimlerinden kendisini haberdar edecek kullanıcı aracına SUBSCRIBE isteğini gönderir. Eğer bildirici taraf bu isteği kabul ederse 200 OK ile yanıt verir ve hemen arkasından olayın o anki durumunu

NOTIFY isteğiyle üye olan tarafa gönderir. SUBSCRIBE isteği içerisinde belirtilen zaman aşımı süresine kadar bildirici taraf olayın değişimlerini NOTIFY isteklerini kullanarak olaya üye olan kullanıcı aracına bildirir [14].

RFC 3265'e [14] göre bir bildirici tarafından raporlanabilecek olaylar grubu ve özellikleri bir olay paketi tarafından tanımlanır. Oluşturulan olay paketleri IANA tarafından onaylanır ve kayıt altına alınır. Kayıt altına alınan olay paketleri IANA tarafından yayınlanmaktadır. Bugüne kadar SIP olay bildirim modelinin akıllı şebeke ve uzaktan ev otomasyonuna yönelik bir olay paketi tanımlanmamıştır [34]. Dolayısıyla bu çalışmada olay bildiriminin gerçekleştirilmesi için “ampulEvent” adıyla bir olay tanımlanmış ve bu olaya kullanıcı aracının üye olmasıyla ışığın izlenmesi gerçekleştirilmiştir.

Işığın durum değişimlerinden haberdar olmak isteyen bir uzak kullanıcı “ampulEvent” olayına üye olacak şekilde ev otomasyon sunucusuna bir SUBSCRIBE isteği gönderir. Eğer sunucu bu isteği kabul ederse SUBSCRIBE isteği için 200 OK yanıtını gönderir ve olayın o anki durumunu NOTIFY isteğini kullanarak gönderir. Üyeliğin geçerli olduğu süre içerisinde olayda herhangi bir değişim olursa bu değişimi NOTIFY isteğiyle gönderir. Şekil 3.10 Wireshark [33] paket analiz programı kullanılarak hazırlanan olay bildirimi için örnek bir sinyalleşme akışını göstermektedir.



Şekil 3.10 Örnek olay bildirimi sinyalleşmesi akışı

3.2 SIP-ZIGBEE İŞBİRLİĞİYLE UZAKTAN EV OTOMASYONU VE OLAY/DURUM BİLDİRİMİ SİSTEMİ

Uzak istemciden SIP yoluyla gelen otomasyon istekleri, cihazlara erişerek onları kontrol etmeyi sağlayan bir arayüz aracılığıyla otomasyon sunucusu tarafından gerçekleştirilebilir. ZIGBEE [12] protokol yığını böyle bir arayüzün oluşturulmasını sağlamaktadır. ZIGBEE IEEE 802.15.4 kablosuz haberleşme standardını ağ ve güvenlik katmanları ile uygulama yapısı ekleyerek geliştirmektedir. Ev ve bina otomasyonu yapılarındaki standartlarla birlikte çalışmaya uyumludur [12].

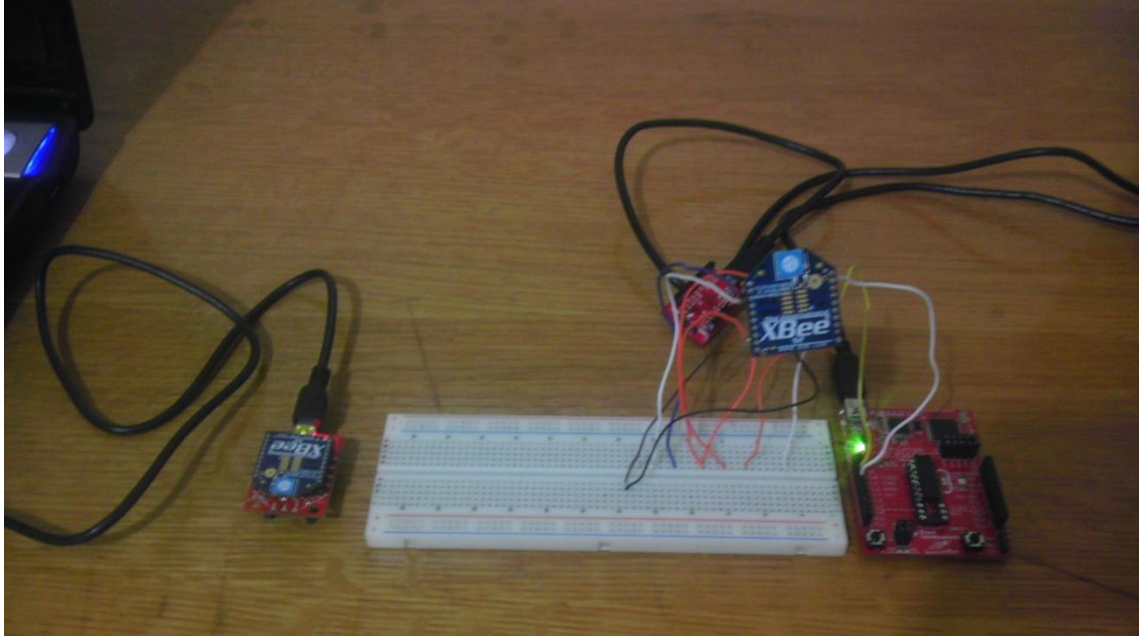
3.2.1 Sistem Mimarisi

Simülasyon uygulamasına benzer şekilde temelde uzak bir istemci ile ev içerisinde konuşlandırılmış bir otomasyon sunucusundan oluşmaktadır. Farklı olarak ev otomasyon sunucusu ZIGBEE arayüzünü kullanarak gerçek bir LED üzerinde otomasyon işlemi gerçekleştirebilmektedir.

Bu uygulamada, istemci ve ev otomasyon sunucusu yazılımı iki ayrı kişisel bilgisayar üzerinde konuşlandırılmıştır. Ev otomasyon sunucusunun bulunduğu kişisel bilgisayara, USB'den seri porta dönüşüm sağlayan bir ara ekipman aracılığıyla XBEE [35] modülü bağlanmıştır. Bu modül otomasyon sunucusundan aldığı komutlarla bir diğer XBEE modülüne ZIGBEE protokolünü kullanarak komutlar gönderir. ZIGBEE aracılığıyla bu komutları alan ikinci modül Texas Instruments MSP-EXP430G2 [36] modülüne bağlanmıştır ve komutlara istinaden belirli pinlerin lojik değerini değiştirerek MSP-EXP430G2 modülü üzerindeki LED'i MSP430 mikrodenetleyici aracılığıyla yakar veya söndürür. Aynı zamanda modül üzerindeki bir buton aracılığıyla da LED el ile yakılıp söndürülebilmektedir. Böylece LED'in durumu değiştirilerek bu durumun takibi ve bildirimi yapılabilir.

Şekil 3.11'de ev otomasyon sunucusu sistemin kurulmuş halinin bir fotoğrafı yer almaktadır. İki adet XBEE modülünden soldaki ev otomasyon sunucusu yazılımının çalıştığı kişisel bilgisayara USB'den seri porta dönüşüm yapan bir arabirim aracılığıyla bağlıdır. Bu XBEE modülüne USB tarafından hem enerji beslenmekte hem de seri port üzerinden veri alışverişi gerçekleştirilmektedir. Ev otomasyon sunucusu tarafından

alınan SIP isteklerine istinaden bu XBEE modülü aracılığıyla sağ tarafta yer alan XBEE modülüne komutlar ZIGBEE kablosuz haberleşme protokolü aracılığıyla iletilmektedir.



Şekil 3.11 Ev otomasyon sunucusu sisteminin fotoğrafı

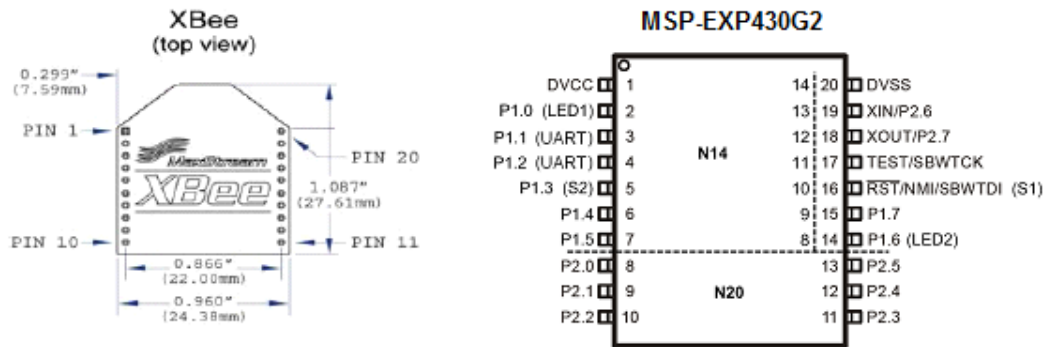
Sağ tarafta yer alan XBEE modülü ve MSP-EXP430G2 modülü için USB tarafından sadece enerji beslemesi yapılmaktadır. USB yerine herhangi bir enerji kaynağı kullanılabilir. Bu XBEE modülünün 19 ve 20 numaralı pinleri MSP-EXP430G2 modülüne bağlanmıştır. Bu pinlerin soldaki XBEE vasıtasıyla gönderilen komutlar aracılığıyla kontrol edilmesiyle otomasyon ve olay bildirimi işlemleri gerçekleştirilmektedir.

3.2.2 Yazılım Mimarisi

Bu uygulamada simülasyon yazılımı uygulamasında kullanılan istemci yazılımı aynı şekilde kullanılmaktadır. Ev otomasyon sunucusu yazılımında ise XBEE API [37] fonksiyonları yazılıma eklenerek, sunucu tarafından algılanan SIP isteklerine istinaden, kendisine bağlı olan XBEE aracılığıyla uzak XBEE üzerinde işlem gerçekleştirebilecek şekilde yetenek kazandırılmıştır. XBEE API'nin çağrıldığı SIPAutoRealize.java sınıfı eklerde Ek-C içerisinde paylaşılmıştır. Sınıf hiyerarşisi ve mimarisi olarak hem istemci hem de sunucu için simülasyon uygulamasına göre bir değişiklik bulunmamaktadır.

Uzak XBEE'nin bağlı olduğu MSP-EXP430G2 modülü üzerindeki MSP430 mikrodenetleyicisi ise C programlama dili kullanılarak programlanmıştır. Yazılım mikrodenetleyicinin “MSP-EXP430G2231.h” isimli kütüphanesi eklendikten sonra, XBEE'nin 20 numaralı pininin bağlı olduğu 1.2 numaralı pin kontrol edilmektedir. XBEE tarafından bu pin değeri lojik 1 veya 0 yapıldığında, mikrodenetleyici MSP-EXP430G2 modülü üzerindeki LED'i yakmakta veya söndürmektedir. Ayrıca modül üzerindeki butonun bağlı olduğu 1.3 numaralı pindeki değişimleri takip ederek her değişimde LED'i yakıp söndürmekte ve bu işlemi XBEE'nin 19 numaralı pininin bağlı olduğu 1.1 numaralı pin değerini değiştirerek LED'in durumundan XBEE'nin haberdar olmasını sağlamaktadır. Mikrodenetleyici yazılımına ait kodlar Eklerde Ek-D içerisinde paylaşılmıştır.

Şekil 3.12'de XBEE ve MSP-EXP430G2 modüllerinin pin diyagramları gösterilmektedir.



Şekil 3.12 XBEE ve MSP-EXP430G2 modüllerinin pin diyagramları [35, 36]

4. BULGULAR

Bu bölümde SIP protokolünün uzaktan ev otomasyonu ve olay/durum bildirimi için kullanımının başarısı açıklanmıştır.

4.1 UZAKTAN EV OTOMASYONU SİSTEMİNİN SİMÜLE EDİLMESİ

Başlangıç olarak, SIP'in uzaktan ev otomasyonu ve olay/durum bildirimi başarısını gözlemlemek için yapılan ev otomasyonu sunucusunun gerçek bir elektronik otomasyon işlemi gerçekleştirmesi yerine arayüz üzerinde otomasyon işlemini simüle etmesi sağlanmıştır.

Bu simülasyonda 192.168.1.37 yerel ağ adresine ve client@registrar.okyayhome.com SIP URI'sine sahip bir istemci ile 192.168.1.34 yerel ağ adresine ve server@registrar.okyayhome.com SIP URI'sine sahip ev otomasyon sunucusu kullanılmaktadır.

4.1.1 İstemci ve Ev Otomasyon Sunucusunun Vekil Sunucuya Kaydı

NAT arkasında bulunan istemci ve ev otomasyon sunucusunun birbirleriyle SIP URI'leri aracılığıyla haberleşebilmeleri için bu URI'lerin gerçek adreslerini çözebilecek bir vekil sunucuya kayıt olmaları gerekir. Her iki kullanıcı aracı önce kendi alan adlarına kayıt olur.

Wireshark paket analiz programı kullanılarak ev otomasyon sunucusu üzerinde kayıt işlemi sırasında toplanan paketler:

1. Ev otomasyon sunucusundan SBC aracılığıyla vekil sunucuya

REGISTER sip:server@registrar.okyayhome.com;transport=udp SIP/2.0

Call-ID: 9293608d3bc50c75fd2c947e5dd28348@192.168.1.34

CSeq: 1 REGISTER

From: "server" <sip:server@192.168.1.34:5060>;tag=SIPAutoClient1.0
 To: "server" <sip:server@registrar.okyayhome.com>
 Via: SIP/2.0/UDP 192.168.1.34:5060;branch=z9hG4bK683ea
 Max-Forwards: 70
 Contact: "server" <sip:server@192.168.1.34:5060>
 Expires: 3600
 Content-Length: 0

2. Vekil sunucudan SBC aracılığıyla otomasyon sunucusuna

SIP/2.0 100 Trying
 Via: SIP/2.0/UDP 192.168.1.34:5060;received=85.101.201.127;branch=z9hG4bK683ea
 From: "server" <sip:server@192.168.1.34:5060>;tag=SIPAutoClient1.0
 To: "server" <sip:server@registrar.okyayhome.com>
 Call-ID: 9293608d3bc50c75fd2c947e5dd28348@192.168.1.34
 CSeq: 1 REGISTER
 Content-Length: 0

3. Vekil sunucudan SBC aracılığıyla otomasyon sunucusuna

SIP/2.0 200 OK
 Via: SIP/2.0/UDP 192.168.1.34:5060;received=85.101.201.127;branch=z9hG4bK683ea
 To: "server" <sip:server@registrar.okyayhome.com>;tag=3547735317-411297
 From: "server" <sip:server@192.168.1.34:5060>;tag=SIPAutoClient1.0
 Call-ID: 9293608d3bc50c75fd2c947e5dd28348@192.168.1.34
 CSeq: 1 REGISTER
 Allow: CANCEL, ACK, INVITE, BYE, OPTIONS, REGISTER, NOTIFY, INFO,
 REFER, SUBSCRIBE, PRACK, UPDATE, MESSAGE, PUBLISH
 Contact: <sip:server@192.168.1.34:5060>;expires=899
 Expires: 899
 Content-Length: 0

4.1.2 Ev Otomasyon Sunucusunun Sorgulanması

İstemci, ev otomasyon sunucusuna OPTIONS isteği göndererek sunucunun kendisine ışığın şu anki durumuyla ilgili bilgi göndermesini sağlayabilir. Gelen bilgiye istinaden istemci ışığı açmak ve kapamak için kullanılacak butonları aktifleştirir veya devre dışı bırakır. Böylece örneğin ev içerisinde zaten açık olan bir ışığın veya herhangi bir cihazın tekrar açılmaya çalışılması önlenmiş olur. Elbette cihaz çalışırken özellikleriyle ilgili bir değişiklik yapılacaksa dizayn buna göre yapılmalıdır. Bu tarz kompleks senaryolarda mesaj içeriği olarak XML veya daha farklı formatlar kullanılması gerekebilir. Bu çalışmada sadece bir tek ampulün otomasyonu yapılacağından otomasyon sisteminin herhangi bir liste paylaşmasına gerek yoktur. Sadece ışığın durumunu bildirmesi yeterli olacaktır.

Genband S3 SBC [27] adresleri cihaz üzerinde tanımlanmadan; REGISTER isteği aracılığıyla kayıt olan dinamik kullanıcı cihazlar tarafından gönderilen OPTIONS yönteminin vekil sunucusuna iletimini güvenlik sebebiyle izin vermemektedir. Dolayısıyla ev otomasyon sunucusunun sorgulanması senaryosunda, Genband S3 SBC [27] topolojiden çıkarılarak OPTIONS istekleri doğrudan vekil sunucu görevi yapan Kamailio SIP sunucusuna [28] gönderilmiştir.

Şekil 4.1’de yazılım arayüzlerinin çıktısı aracılığıyla ışık önce kapalı sonra açık olduğu durumda istemci tarafından OPTIONS isteğiyle sorgulanması gösterilmektedir. İstemci sunucuya sorgu amacıyla bir OPTIONS isteği gönderir. Işık bu sırada kapalı olduğundan sunucu bu bilgiyi MESSAGE isteğiyle istemciye bildirir ve istemciye OPTIONS isteğini aldığını ve isteği onayladığını belirtecek şekilde 200 OK yanıtını verir. İstemci ise MESSAGE isteğini aldığını belirtmek üzere sunucuya 200 OK yanıtını gönderir. Aynı işlem ışık açıldıktan sonra yapıldığında ise bu kez sunucu ışığın açık olduğu bilgisini aynı sinyalleşmeyi tekrarlayarak iletir. Böylece istemci otomasyon sunucusunun kontrol edebildiği bir ışık olduğunu ve bu ışığın kapalı veya açık olduğunu başarıyla algılar.

Wireshark paket analiz programıyla sorgulama sinyalleşmesi sırasında sunucu üzerinde yakalanan SIP paketleri ışık kapalı olduğu durum için şu şekilde olmuştur:

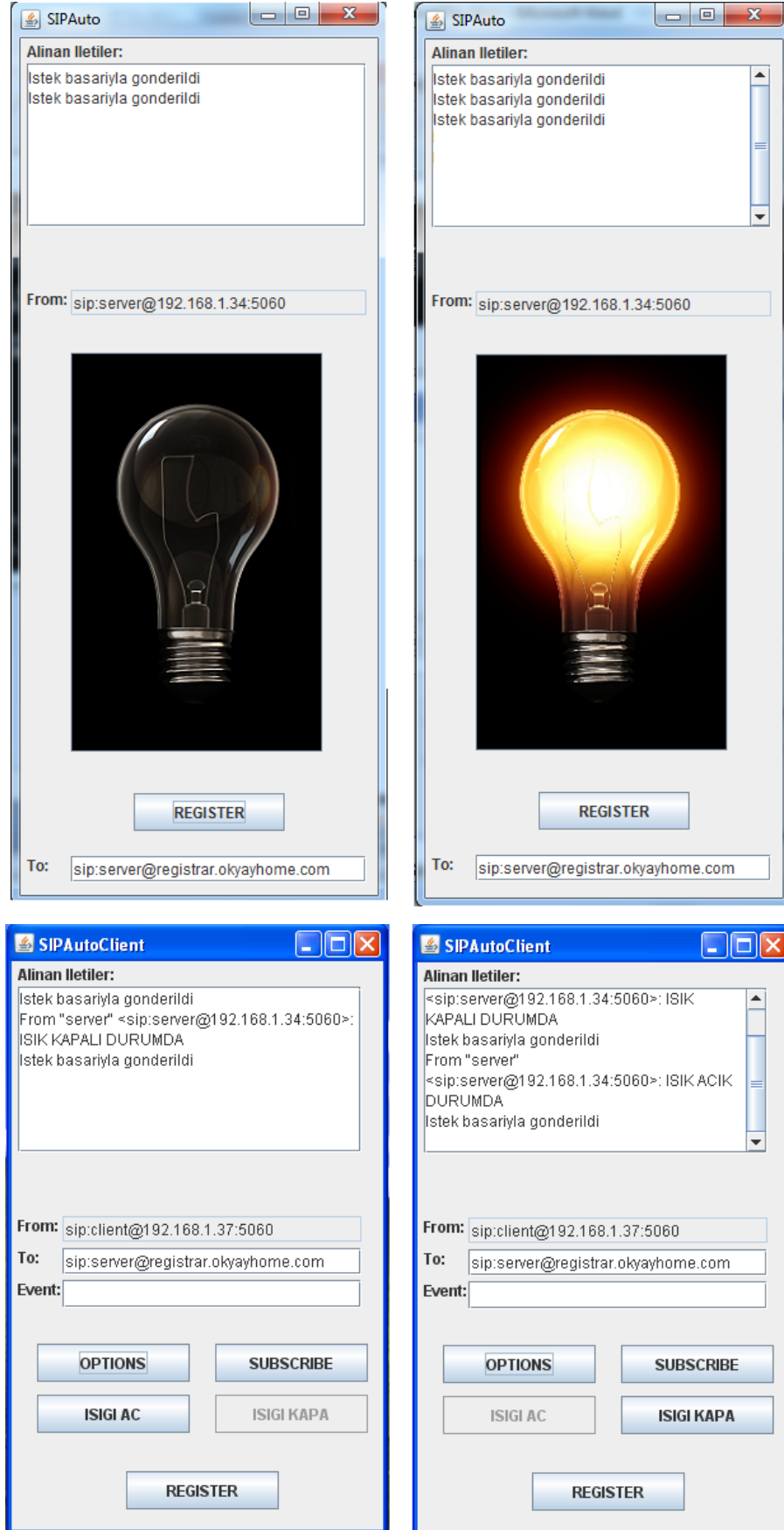
1. İstemciden vekil sunucu aracılığıyla otomasyon sunucusuna;

OPTIONS sip:server@192.168.1.34:5060 SIP/2.0
 Call-ID: 5577e490cf1b329cb42ade16e8495149@192.168.1.37
 CSeq: 1 OPTIONS
 From: "client" <sip:client@192.168.1.37:5060>;tag=SIPAutoClient1.0
 To: "server" <sip:server@registrar.okyayhome.com>
 Via: SIP/2.0/UDP 85.153.8.107;branch=z9hG4bK7fb.76d50f5.0
 Via: SIP/2.0/UDP
 192.168.1.37:5060;rport=23182;received=85.101.201.127;branch=branch1
 Max-Forwards: 69
 Contact: "client" <sip:client@85.101.201.127:23182>
 Content-Length: 0

2. Otomasyon sunucusundan istemciye;

MESSAGE sip:client@192.168.1.37:5060;transport=udp SIP/2.0
 Call-ID: fa18f47b9a2662e0c0a9d684a900925d@192.168.1.34
 CSeq: 1 MESSAGE
 From: "server" <sip:server@192.168.1.34:5060>;tag=SIPAuto1.0
 To: "client" <sip:client@192.168.1.37:5060>
 Via: SIP/2.0/UDP 192.168.1.34:5060;branch=branch1
 Max-Forwards: 70
 Contact: "server" <sip:server@192.168.1.34:5060>
 Content-Type: text/plain
 Content-Length: 19

ISIK KAPALI DURUMDA



Şekil 4.1 OPTIONS isteğiyle sorgulamadaki arayüz ekran çıktıları

4.1.3 Komutların Uygulanması

Komutların uygulanması istemcinin ev otomasyon sunucusuna mesaj içeriğinde ışığı açmak istediğini belirten bir MESSAGE isteği göndermesiyle gerçekleşmektedir. Bu isteği kabul eden sunucu isteğe 200 OK yanıtını verir ve arayüz ekranındaki resmi değiştirerek ışığı açar. Yanıtın hangi isteğe verildiğinin karışmaması ve istemcinin doğru bir şekilde işlemin gerçekleştirildiğinden kesinlikle haberdar olması için sunucu istemciye ışığın açıldığını belirtecek şekilde MESSAGE isteği gönderir. İstemci bu isteği alınca, isteği aldığını belirtmek üzere 200 OK yanıtını gönderir.

Şekil 4.2 istemciden sunucuya önce ışığın açılması daha sonra ışığın kapatılması isteği gönderildiğindeki durum için arayüz ekran çıktılarını göstermektedir. Sunucu isteklere olumlu olarak cevap vermiş ve arayüz üzerindeki ampul resmini gelen isteklere göre değiştirerek ışığın açılmasını ve kapatılmasını başarılı bir şekilde simüle etmiştir. Ayrıca ışığın açılıp kapandığına dair geri bildirim yaparak, istemciyi son durumdan haberdar etmiştir. Böylece istemci ışığın durumuna göre “ISIGI AC” ve “ISIGI KAPA” butonlarını devreye almış veya devreden çıkarmıştır. Aynı zamanda olaylar ve bildirimler ekrana bilgi amaçlı yazdırılmıştır.

Wireshark ile toplanan paketleri incelediğimizde ışığın başarıyla açıldığı durumda istemci ve sunucu arasındaki haberleşmenin şu şekilde olduğu gözlemlenmiştir:

1. İstemciden SBC ve vekil sunucu aracılığıyla otomasyon sunucusuna;

MESSAGE sip:server@192.168.1.34:5060 SIP/2.0

Max-Forwards: 67

To: "server" <sip:dynamic-3547734614.669467.121@85.153.8.194:5060>

From: "client" <sip:client@85.153.8.194>;tag=3547735755-199633

Call-ID: 289-3547735755-189332@lab-ist-msx2.mydomain.com

CSeq: 1 MESSAGE

Allow: CANCEL, ACK, INVITE, BYE, OPTIONS, REGISTER, NOTIFY, INFO, REFER, SUBSCRIBE, PRACK, UPDATE, MESSAGE, PUBLISH

Via: SIP/2.0/UDP

85.153.8.194:5060;branch=z9hG4bKf833a293a9e345c93410237c381bf5f9

Contact: <sip:client@85.153.8.194:5060>

Expires: 0

Content-Type: text/plain

Accept: text/plain

Content-Length: 8

AMPUL AC

2. Otomasyon sunucusundan SBC ve vekil sunucu aracılığıyla istemciye;

MESSAGE sip:client@85.153.8.194;transport=udp SIP/2.0

Call-ID: 9e28a4820395c311c37a7ea2a7b750a3@192.168.1.34

CSeq: 1 MESSAGE

From: "server" <sip:server@192.168.1.34:5060>;tag=SIPAuto1.0

To: "client" <sip:client@85.153.8.194>

Via: SIP/2.0/UDP 192.168.1.34:5060;branch=branch1

Max-Forwards: 70

Contact: "server" <sip:server@192.168.1.34:5060>

Content-Type: text/plain

Content-Length: 11

ISIK ACILDI

3. Otomasyon sunucusundan SBC ve vekil sunucu aracılığıyla istemciye;

SIP/2.0 200 OK

To: "server" <sip:dynamic-3547734614.669467.121@85.153.8.194:5060>;tag=789

From: "client" <sip:client@85.153.8.194>;tag=3547735755-199633

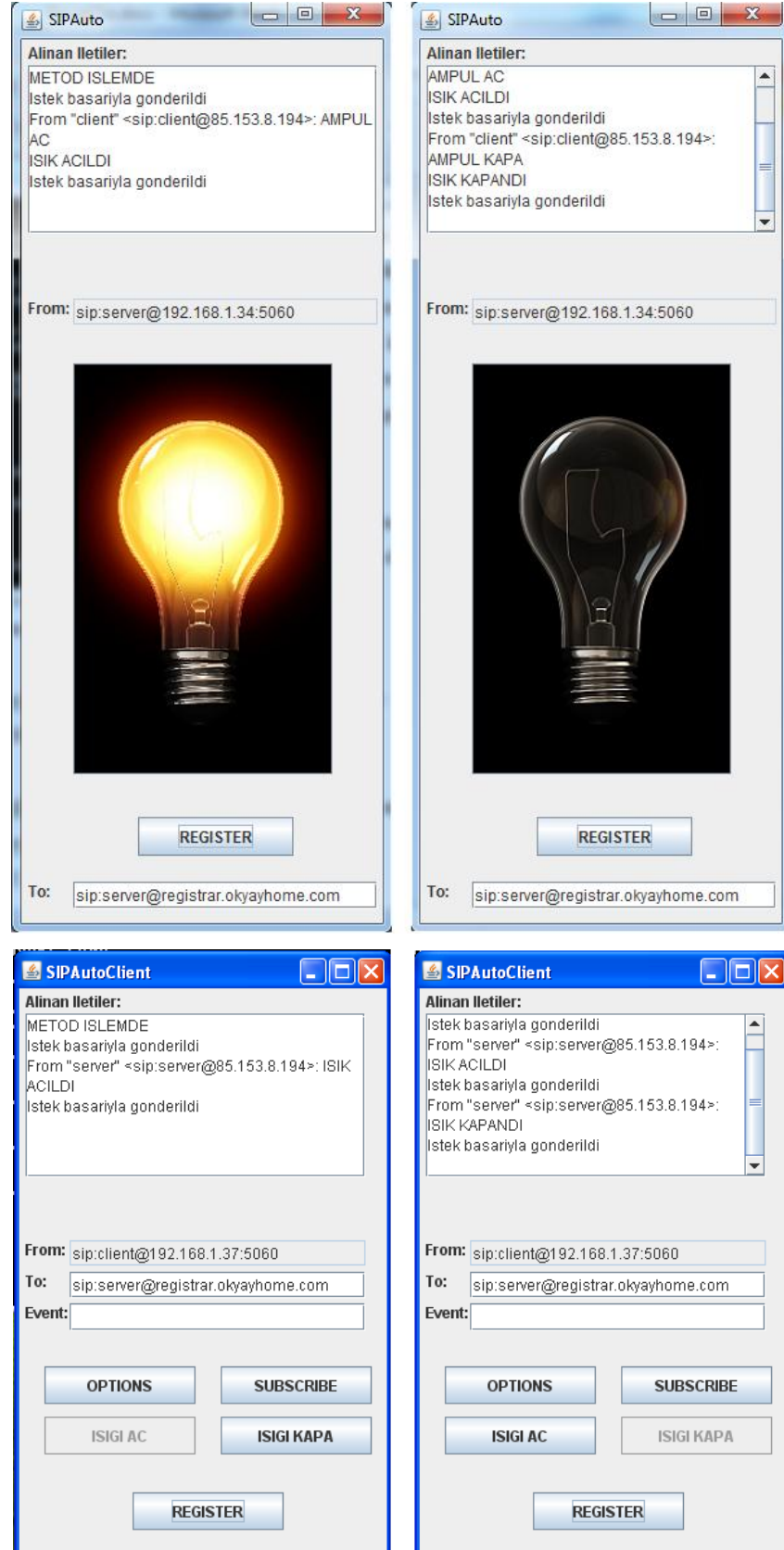
Call-ID: 289-3547735755-189332@lab-ist-msx2.mydomain.com

CSeq: 1 MESSAGE

Via: SIP/2.0/UDP

85.153.8.194:5060;branch=z9hG4bKf833a293a9e345c93410237c381bf5f9

Content-Length: 0



Şekil 4.2 MESSAGE isteğiyle komutların uygulanmasındaki arayüz ekran çıktıları

4. İstemciden SBC ve vekil sunucu aracılığıyla otomasyon sunucusuna;

SIP/2.0 200 OK

Via: SIP/2.0/UDP 192.168.1.34:5060;received=85.101.201.127;branch=branch1

To: "client" <sip:client@85.153.8.194>;tag=3547735756-544040

From: "server" <sip:server@192.168.1.34:5060>;tag=SIPAuto1.0

Call-ID: 9e28a4820395c311c37a7ea2a7b750a3@192.168.1.34

CSeq: 1 MESSAGE

Allow: CANCEL, ACK, INVITE, BYE, OPTIONS, REGISTER, NOTIFY, INFO, REFER, SUBSCRIBE, PRACK, UPDATE, MESSAGE, PUBLISH

Content-Length: 0

4.1.4 Olay/Durum Bildirimi

İstemcinin “ampulEvent” olayına üye olmasından sonra, sunucu arayüzü üzerindeki buton kullanılarak ışığın açılması ve kapatılmasıyla; istemcinin başarılı bir şekilde bu olaylardan haberdar olması sağlanmıştır.

Şekil 4.3, istemcinin “ampulEvent” olayına üye olduktan sonra, sunucu üzerindeki buton aracılığıyla ışığın durumunun değiştirilmesi sonucu oluşan, olay/durum bildirimlerine ait arayüz ekran çıktıları göstermektedir. Olaya ilk üye olunduktan sonra ışığın kapalı olduğu istemciye bildirilmiştir. Daha sonra ışığın her açılıp kapanmasındaki durum değişimleri yine başarıyla istemciye iletilmiştir.

Olaya üye olunması ve ilk başta kapalı olan ışığın açılması durumunda; haberleşmedeki paketlerin yapısı şu şekilde olmuştur:

1. İstemciden SBC ve vekil sunucu aracılığıyla otomasyon sunucusuna;

SUBSCRIBE sip:server@192.168.1.34:5060 SIP/2.0

Max-Forwards: 67

To: "server" <sip:dynamic-3547737647.721637.124@85.153.8.194:5060>

From: "client" <sip:client@85.153.8.194>;tag=3547738518-906527

Call-ID: 346-3547738518-906397@lab-ist-msx2.mydomain.com

CSeq: 1 SUBSCRIBE

Allow: CANCEL, ACK, INVITE, BYE, OPTIONS, REGISTER, NOTIFY, INFO, REFER, SUBSCRIBE, PRACK, UPDATE, MESSAGE, PUBLISH

Via: SIP/2.0/UDP

85.153.8.194:5060;branch=z9hG4bKe5d3f08e4faac2f4ad889ea072d2b0ed

Contact: <sip:client@85.153.8.194:5060>

Event: ampulEvent

Expires: 3600

Content-Length: 0

2. Otomasyon sunucusundan SBC ve vekil sunucu aracılığıyla istemciye;

NOTIFY sip:client@85.153.8.194;transport=udp SIP/2.0

Call-ID: b4f1c72f29f0077a3beb4ce237980494@192.168.1.34

CSeq: 1 NOTIFY

From: "server" <sip:server@192.168.1.34:5060>;tag=SipAuto1.0

To: "client" <sip:client@85.153.8.194>

Via: SIP/2.0/UDP 192.168.1.34:5060;branch=branch1

Max-Forwards: 70

Contact: "server" <sip:server@192.168.1.34:5060>

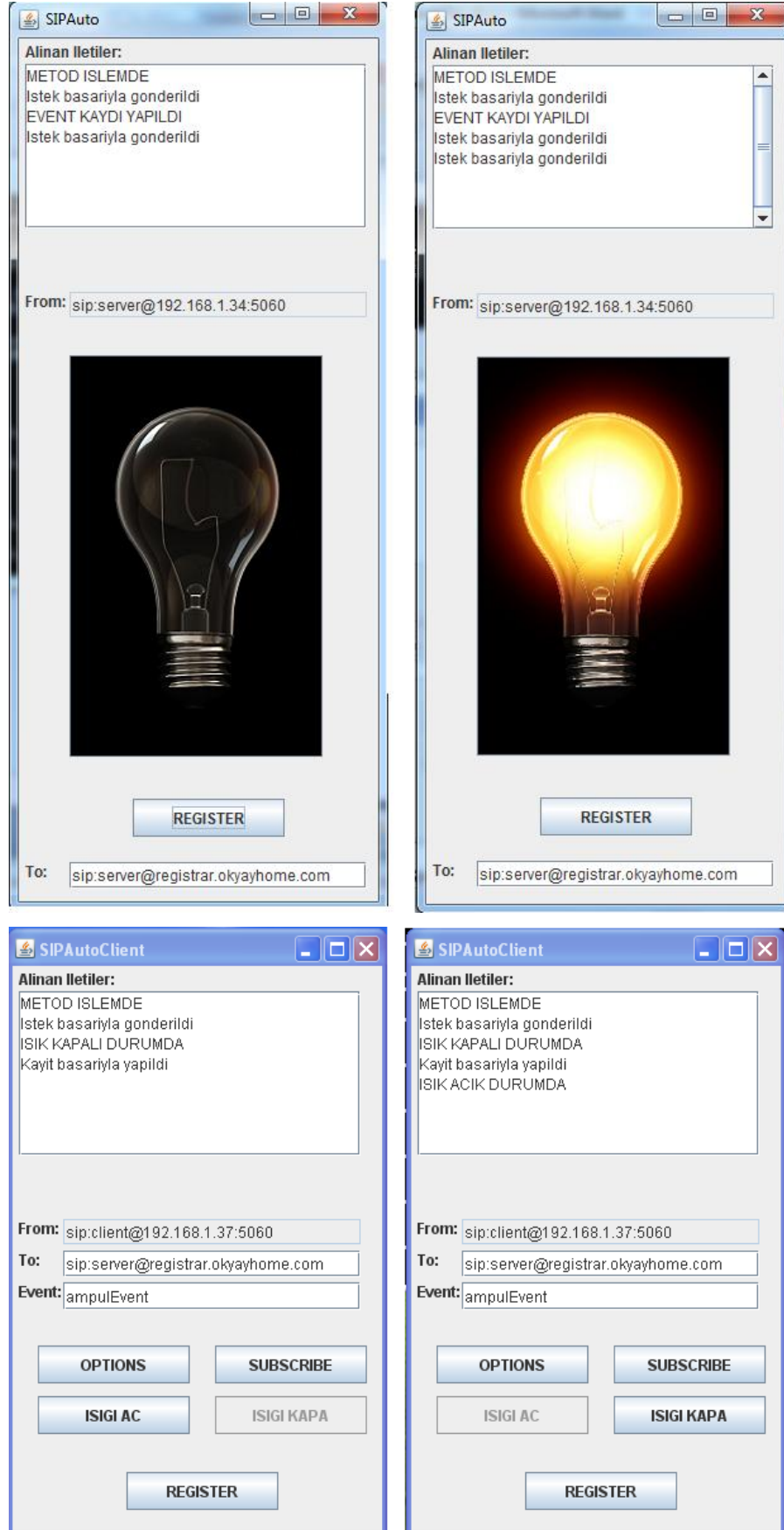
Subscription-State: active

Event: ampulEvent

Content-Type: text/plain

Content-Length: 19

ISIK KAPALI DURUMDA



Şekil 4.3 Olay bildirimi ekran çıktıları

3. Otomasyon sunucusundan SBC ve vekil sunucu aracılığıyla istemciye;

SIP/2.0 200 OK

To: "server" <sip:dynamic-3547737647.721637.124@85.153.8.194:5060>;tag=789

From: "client" <sip:client@85.153.8.194>;tag=3547738518-906527

Call-ID: 346-3547738518-906397@lab-ist-msx2.mydomain.com

CSeq: 1 SUBSCRIBE

Via: SIP/2.0/UDP

85.153.8.194:5060;branch=z9hG4bKe5d3f08e4faac2f4ad889ea072d2b0ed

Expires: 0

Content-Length: 0

4. İstemciden sunucuya SBC ve vekil sunucu aracılığıyla otomasyon sunucusuna;

SIP/2.0 200 OK

Via: SIP/2.0/UDP 192.168.1.34:5060;received=85.101.201.127;branch=branch1

To: "client" <sip:client@85.153.8.194>;tag=3547738519-24918

From: "server" <sip:server@192.168.1.34:5060>;tag=SipAuto1.0

Call-ID: b4f1c72f29f0077a3beb4ce237980494@192.168.1.34

CSeq: 1 NOTIFY

Allow: CANCEL, ACK, INVITE, BYE, OPTIONS, REGISTER, NOTIFY, INFO, REFER, SUBSCRIBE, PRACK, UPDATE, MESSAGE, PUBLISH

Contact: <sip:client@85.153.8.194:5060>

Content-Length: 0

5. Otomasyon sunucusundan SBC ve vekil sunucu aracılığıyla istemciye;

NOTIFY sip:client@85.153.8.194;transport=udp SIP/2.0

Call-ID: 93afa93252cc865c4bc67314ceffc6cd@192.168.1.34

CSeq: 1 NOTIFY

From: "server" <sip:server@192.168.1.34:5060>;tag=SipAuto1.0

To: "client" <sip:client@85.153.8.194>

Via: SIP/2.0/UDP 192.168.1.34:5060;branch=branch1

Max-Forwards: 70

Contact: "server" <sip:server@192.168.1.34:5060>

Subscription-State: active

Event: ampulEvent

Content-Type: text/plain

Content-Length: 17

ISIK ACIK DURUMDA

6. İstemciden SBC ve vekil sunucu aracılığıyla otomasyon sunucusuna;

SIP/2.0 200 OK

Via: SIP/2.0/UDP 192.168.1.34:5060;received=85.101.201.127;branch=branch1

To: "client" <sip:client@85.153.8.194>;tag=3547738526-249979

From: "server" <sip:server@192.168.1.34:5060>;tag=SipAuto1.0

Call-ID: 93afa93252cc865c4bc67314ceffc6cd@192.168.1.34

CSeq: 1 NOTIFY

Allow: CANCEL, ACK, INVITE, BYE, OPTIONS, REGISTER, NOTIFY, INFO, REFER, SUBSCRIBE, PRACK, UPDATE, MESSAGE, PUBLISH

Contact: <sip:client@85.153.8.194:5060>

Content-Length: 0

4.2 UZAKTAN EV OTOMASYONU SİSTEMİNİN SIP-ZIGBEE İŞBİRLİĞİYLE GERÇEKLENMESİ

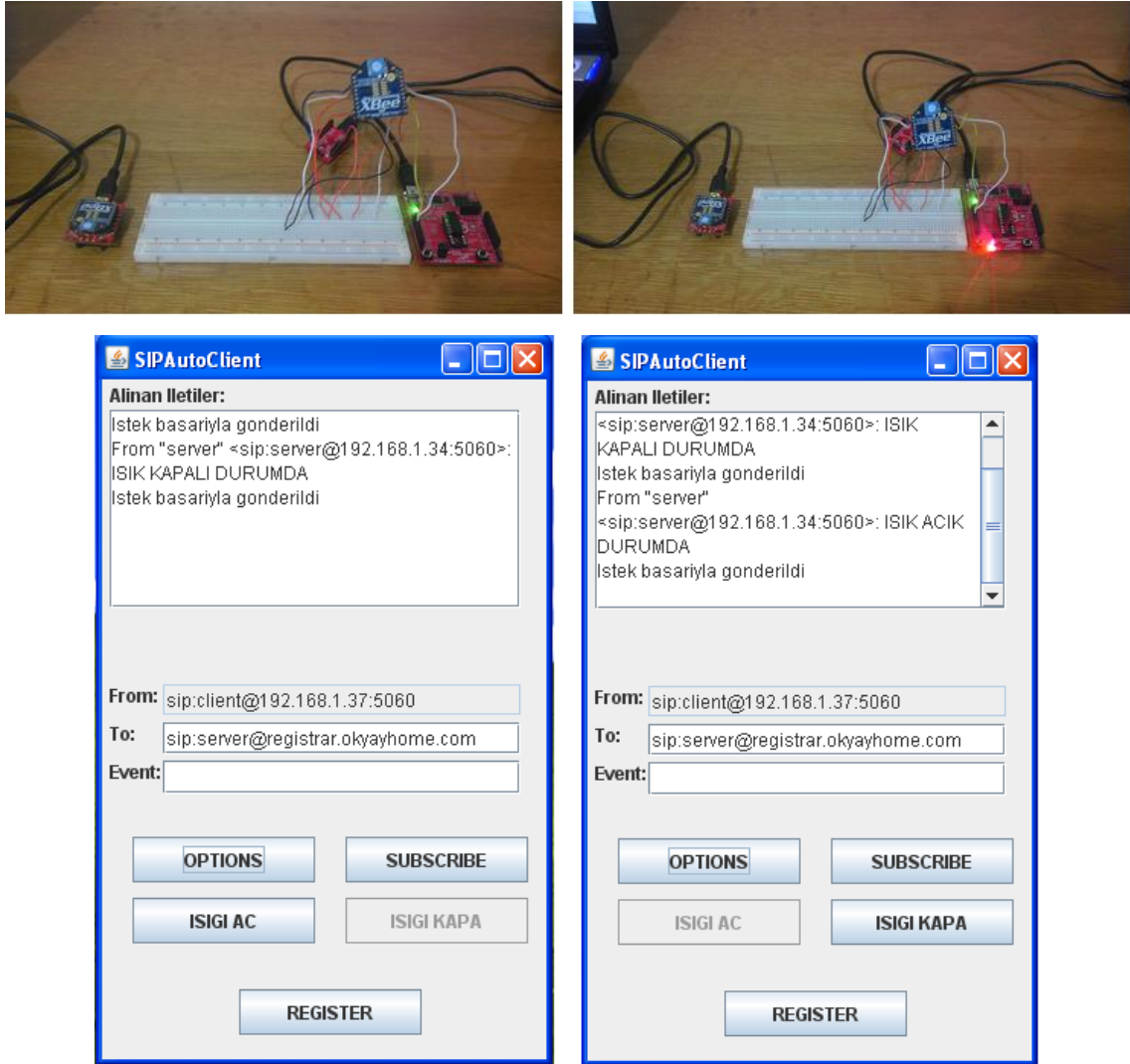
Ev otomasyon sunucusunun kablosuz haberleşmeyle gerçek bir otomasyon yapılması XBEE modülleri aracılığıyla olmaktadır. Tasarlanan bu sistem, yapılan simülasyon uygulamasıyla ile aynı SIP sinyalleşmesine sahipken, sunucuya entegre edilen ZIGBEE arayüzü sayesinde gerçek bir LED'in yakılıp söndürülmesi sağlanmaktadır.

4.2.1 Ev Otomasyon Sunucusunun Sorgulanması

Ev otomasyon sunucusu, istemciden OPTIONS isteğini aldıktan sonra, ZIGBEE sinyalleşmesiyle uzak XBEE'ye MSP-EXP430G2 modülü aracılığıyla bağlı olan

LED'in durumunu kontrol eder. Bunun sağlanması için MSP430 mikrodenetleyicisi LED'i kontrol eder ve LED'in o anki durumuna göre XBEE'nin 19 numaralı pinine dijital veri girişi yapar. Sunucu bu dijital veri girişine istinaden LED'in durumunu anlamakta ve istemciye SIP MESSAGE isteğiyle bu durumu başarıyla bildirmektedir.

Şekil 4.4 LED'in kapalı ve açık olduğu durumlar için sunucu sisteminin resimlerini ve istemci arayüzünün çıktılarını göstermektedir.



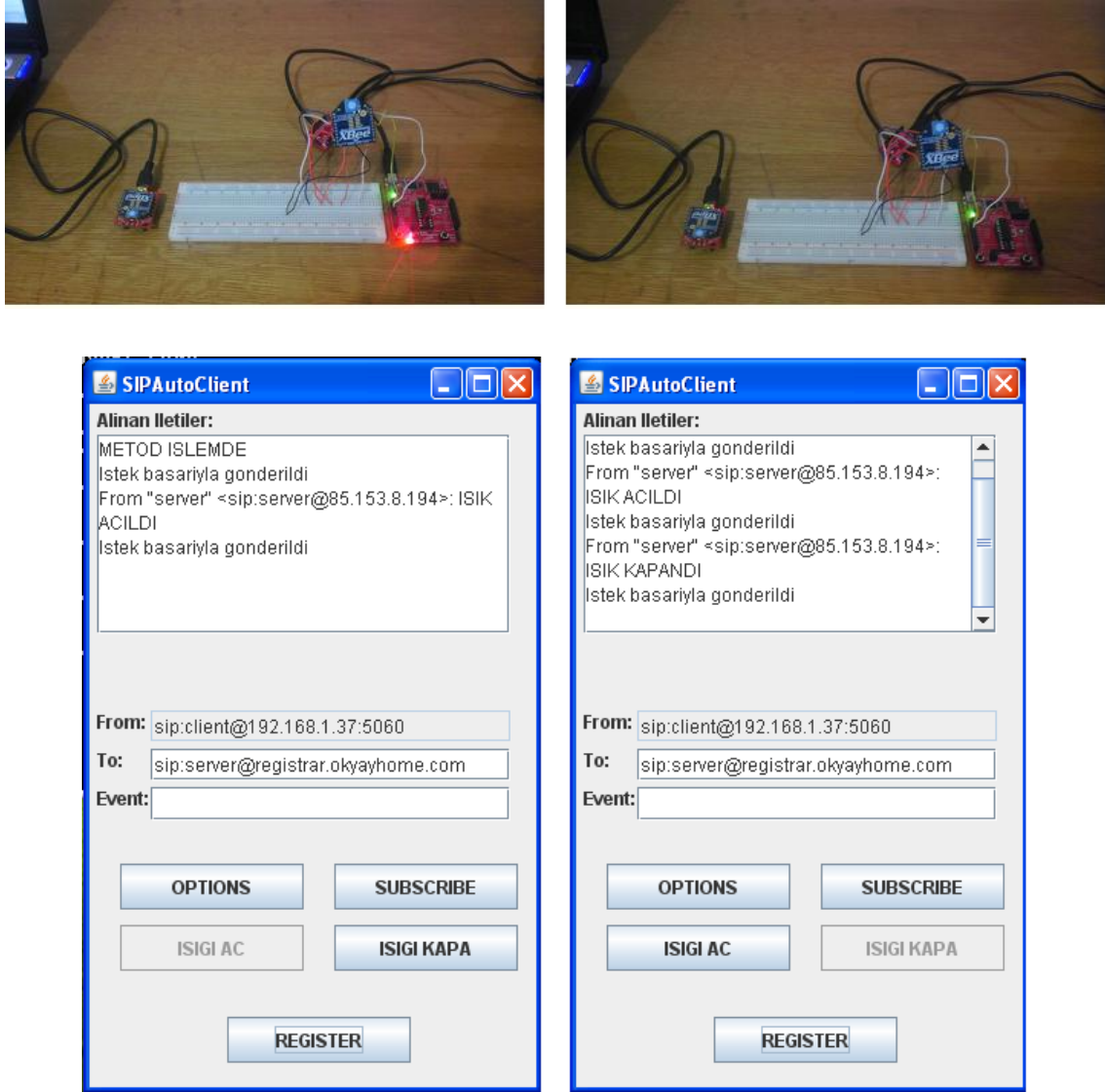
Şekil 4.4 SIP OPTIONS ve ZIGBEE ile sorgulama çıktıları

4.2.2 Komutların Uygulanması

İstemci LED'i yakmak veya söndürmek istediğinde sunucuya bunu belirten MESSAGE isteğini gönderir. Bu isteği alan sunucu, kendisine seri port olarak bağlı olan XBEE modülüyle haberleşir. Bu XBEE modülünü kullanarak, ZIGBEE protokolü aracılığıyla

uzaktaki XBEE modülünün 20 numaralı pininde dijital bir veri çıkışı yapılmasını sağlar. Bu veri çıkışına istinaden MSP430 mikrodenetleyicisi MSP-EXP430G2 modülü üzerindeki LED'i başarıyla yakmakta veya söndürmektedir.

Şekil 4.5 sunucu sistemindeki LED'in yakılıp söndürülmesi ve bunun sonucunda istemciye geri bildirimlerin yapılmasıyla oluşan ekran çıktıları paylaşılmıştır.



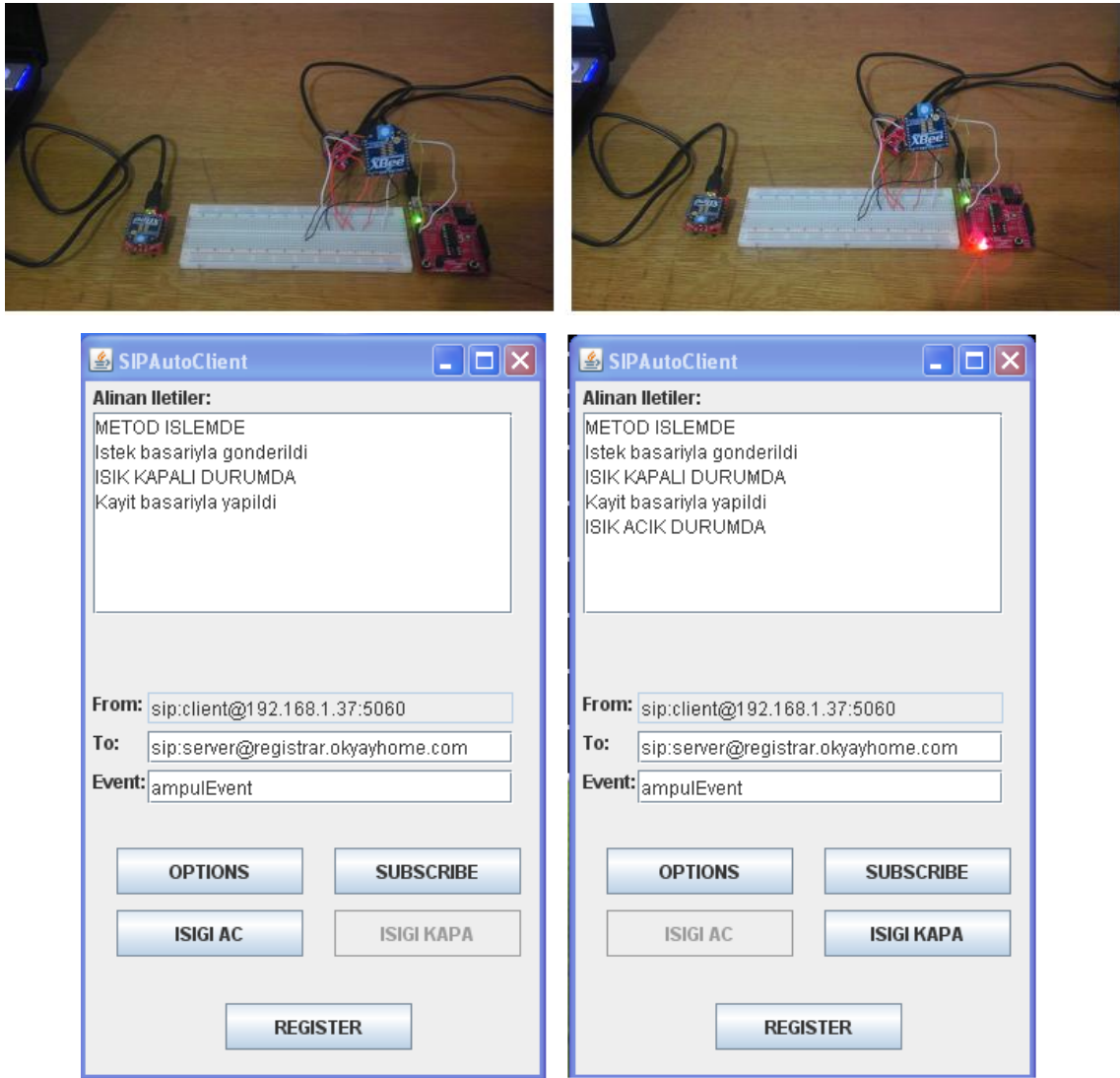
Şekil 4.5 SIP MESSAGE ve ZIGBEE ile otomasyon çıktıları

4.2.3 Olay/Durum Bildirimi

İstemcinin SUBSCRIBE isteğiyle “ampulEvent” olayına kaydolmasının ardından, sunucu XBEE modülleri aracılığıyla MSP-EXP430G2 modülü üzerindeki LED'in durumunu takip etmeye başlar. LED el ile yakılıp söndürüldüğünde MSP430

mikrodenetleyicisi kendine bağı olan XBEE modülünün 19 numaralı pinine dijital veri girişi yapar. Bu veri girişi ev otomasyon sunucusu üzerindeki XBEE'ye aktarılır ve böylece sunucu bunu algılayarak NOTIFY isteğiyle istemciye LED'in durumunu bildirir.

Şekil 4.6 LED kapalı durumdayken istemcinin üye olmasından sonra, LED'in el ile yakılıp söndürülmesi sonucu istemci arayüzünde gözlemlenen bildirimleri göstermektedir.



Şekil 4.6 SIP SUBSCRIBE/NOTIFY ve ZIGBEE ile olay bildirimi çıktıları

5. TARTIŞMA VE SONUÇ

Bu tez çalışması kapsamında yapılan araştırmalar göstermektedir ki SIP, Akıllı Şebeke konsept modelinin [17] Müşteri alanı dahilinde olan yapılarda ve daha çok son kullanıcıyı ilgilendiren AMI veya ev/bina otomasyonu gibi konularda çalışma alanı bulma potansiyeline sahiptir.

SIP her ne kadar canlı veri akışının gerçekleştiği multimedya oturumlarının kurulması ve yönetilmesi için tasarlanmış olsa da, SIMPLE çalışma grubu tarafından kazandırılan anlık mesajlaşma ve olay bildirimi yapıları [14, 15] SIP'in Akıllı Elektrik Şebekesi'nde farklı konularda kullanımını mümkün kılmaktadır. Bu tez çalışması kapsamında SIP ve ZIGBEE işbirliğiyle gerçekleştirilen ev otomasyon yazılım ve sistemiyle, bu yapıların ev otomasyonunun gerçekleştirilmesinde başarılı olduğu gözlemlenmiştir. Bu tarz bir sistem Akıllı Elektrik Şebekesi konsept modeline [17] istinaden elektrik enerjisi dağıtıcısı tarafından veya telekomünikasyon şirketleri gibi Akıllı Şebeke'nin diğer katılımcıları tarafından sunulan bir servis olabilir.

SIP'in tasarlanma amacının multimedya oturumları kurulması ve yönetilmesi olması, Akıllı Elektrik Şebekesi yapılarına özel tasarlanmamış olması, ona dezavantajlar getirmektedir. Bunlara örnek olarak NAT problemlerini, güvenlikle ilgili elde edilmesi gerekenleri ve VoIP temelli yapıdan kaynaklı çeşitli zorlukları verebiliriz. SIP'in NAT mimarisindeki sorunları asıl olarak mesaj içeriğinde SDP gibi multimedya akışı için IP adres ve portu belirtildiği durum için gerçekleşmektedir. Jansen ve diğ. [23] önerdikleri PHEV mimarisinde SIP'in firewall ve NAT problemi için STUN yöntemini kullanmayı tercih etmişlerdir. Ev otomasyonu sisteminde genel olarak multimedya akışını gerektiren bir durum olmadığından, bu sorunla karşılaşılma olasılığı düşüktür. Yine de bu problemlerin aşılması için bu çalışmada vekil sunucu önüne konuşlandırılacak NAT çözümü yapabilen bir SBC önerilmiştir. Böylece SBC'nin ana görevlerinden olan, DoS gibi saldırılara karşı da güvenlik sağlanmış olur. Ayrıca günümüzde birçok SIP

sunucu NAT çözümünü gerçekleştirecek özelliklere sahiptir. Bu çalışmada kullanılan Kamailio SIP sunucusu da NAT çözümü yapabilmektedir.

SIP'in temelde multimedia oturumları kurulmasını sağlayan protokol olması dezavantajlar getirmesinin yanında avantajlar da getirmektedir. Örneğin ev otomasyon sistemi evdeki görüntülü interkom sistemi ile entegre edilebilir. Bu sayede evin sahipleri evde olmadığı zaman otomasyon sistemi INVITE isteğiyle uzaktaki bir mobil istemciyle multimedia oturumu kurabilir. Bu sayede evden uzakta olan bir kişi evine gelen bir kişiden haberdar olur ve üstelik onunla görüntülü konuşma imkanına sahip olur.

Ev otomasyonu gibi kötü kullanıma açık ve kişinin özel hayatını yakından ilgilendiren bir konuda elbette güvenlik açısından değerlendirmeler çok kritik rol oynamaktadır. SIP kimlik doğrulama ve yetkilendirme yapılarına sahiptir [8]. Yine de vekil sunucu ve SBC gibi servis sağlayıcı tarafından yönetilmesi gereken ekipmanlar güvenlik açıkları oluşturmaktadır. Kullanıcılar ve otomasyon sistemleri vekil sunuculara güvenli bir şekilde kayıt olsalar bile, bu cihazlara erişim yapanlar sistem üzerinde kayıtlı şifre gibi güvenlik parametrelerine erişebilirler. Dolayısıyla gerçek kimlik doğrulama ve yetkilendirmenin evdeki otomasyon sunucusu tarafından yapılması gerekir. Evdeki cihazların başkaları tarafından kontrol edilebilmesi kritik durumlar arz edebileceğinden buradaki güvenlik tasarımı dikkatli bir çalışmanın yapılmasını gerektirmektedir.

Ev ortamının dinamik ve karmaşık yapısı sebebiyle de SIP istekleri tarafından taşınacak olan mesaj içeriklerinin yapısı önem kazanmaktadır. Mesaj içeriği olarak, yapılan çalışmalarda [19, 20] genel olarak XML tabanlı mesaj içerikleri kullanılmıştır. Bu çalışmada düz metin mesaj içerikleri kullanılmıştır. Bu tarz mesaj içerikleri güvenlik açısından risk taşımaktadırlar. İleriki çalışmalarda, ev otomasyonu için en uygun mesaj içeriği formatı araştırılabilir ve bu formatın standartları belirlenerek SIP'in ev otomasyonu yapılarında tam anlamıyla başarılı bir şekilde çalışması sağlanabilir.

Bu çalışmada gösterildiği gibi SIP gerekli çalışmalar yapıldığı takdirde Akıllı Şebeke içerisinde farklı şekillerde kendine çalışma alanı bulabilecek kapasiteye sahiptir.

KAYNAKLAR

1. *What is a Smart Grid* [online], <http://collaborate.nist.gov/twiki-sggrid/bin/view/SmartGrid/WebHome> [Ziyaret Tarihi: 4 Ocak 2012].
2. *Smart Grid* [online], http://en.wikipedia.org/wiki/Smart_grid [Ziyaret Tarihi: 4 Ocak 2012].
3. *Information Videos about Smart Grid* [online], <http://www.energy.siemens.com/hq/en/energy-topics/smart-grid/smart-grid-information/> [Ziyaret Tarihi: 4 Ocak 2011].
4. *Iteres Smart Grid: Filling the Gap for New Technology* [online], <http://www.iteresgroup.com/services/smart-grid/> [Ziyaret Tarihi: 5 Nisan 2012].
5. *IPv6 over low power wireless personal area network* [online], <http://datatracker.ietf.org/wg/6lowpan/charter/> [Ziyaret Tarihi: 7 Ocak 2012].
6. *6LoWPAN* [online], <http://en.wikipedia.org/wiki/6LoWPAN> [Ziyaret Tarihi: 7 Ocak 2012].
7. BAKER, F. and MEYER, D., 2011, *Internet Protocols for the Smart Grid* [online], draft-baker-ietf-core-15, Internet Engineering Task Force, <http://tools.ietf.org/html/draft-baker-ietf-core-15> [Ziyaret Tarihi: 8 Ocak 2012].
8. ROSENBERG, J.; SCHULZRINNE, H.; CAMARILLO, G.; JOHNSTON, A. R.; PETERSON, J.; SPARKS, R.; HANDLEY, M.; and SCHOOLER, E., 2002, *SIP: Session Initiation Protocol* [online], RFC 3261, Internet Engineering Task Force, <http://www.ietf.org/rfc/rfc3261.txt> [Ziyaret Tarihi: 15 Ocak 2012].
9. HANDLEY, M.; JACOBSON, V. and PERKINS, C., 2006, *SDP: Session Description Protocol* [online], RFC 4566, Internet Engineering Task Force, <http://www.ietf.org/rfc/rfc4566.txt> [Ziyaret Tarihi: 15 Ocak 2012].
10. SCHULZRINNE, H.; CASNER, S.; FREDERICK, R. and JACOBSON, V., 2003, *RTP: A Transport Protocol for Real-Time Applications* [online], RFC 3550, Internet Engineering Task Force, <http://www.ietf.org/rfc/rfc3550.txt> [Ziyaret Tarihi: 15 Ocak 2012].

11. JOSEPH, V.C.; LUCKY, K.K. and MOHAN RAO, G.N.S., 2006, *SIP as an enabler for convergence in future wireless communication networks*, 2006 IFIP International Conference on Wireless and Optical Communications Networks, Samsung India Software Oper., Bangalore.
12. Zigbee Specification Overview [online], <http://zigbee.org/Specifications/ZigBee/Overview.aspx> [Ziyaret Tarihi: 6 Nisan 2012]
13. JOHSNTON, A. B., 2004, *SIP Understanding the Session Initiation Protocol*, Boston, Artech House, 1-58053-655-7.
14. ROACH, A., 2002, *Session Initiation Protocol (SIP)-Specific Event Notification* [online], RFC 3265, Internet Engineering Task Force, <http://www.ietf.org/rfc/rfc3265.txt> [Ziyaret Tarihi: 13 Mart 2012].
15. CAMPBELL, B., et al., 2002, *Session Initiation Protocol (SIP) Extension for Instant Messaging* [online], RFC 3428, Internet Engineering Task Force, <http://www.ietf.org/rfc/rfc3428.txt> [Ziyaret Tarihi: 15 Mart 2012].
16. EKANAYAKE, J.; LIYANEKE, K.; JIANZHONG, W.; YOKOYAMA, A. and JENKINS, N., 2012, *Smart Grid Technology And Applications*, UK, Wiley, 978-0-470-97409-4.
17. US NIST, 2012, *NIST Framework And Roadmap for Smart Grid Interoperability Standards* [online], http://www.nist.gov/smartgrid/upload/NIST_Framework_Release_2-0_corr.pdf [Ziyaret Tarihi: 23 Mart 2012].
18. IEEE Smart Grid, 2012, *Smart Grid Conceptual Model* [online], <http://smartgrid.ieee.org/ieee-smart-grid/smart-grid-conceptual-model> [Ziyaret Tarihi: 25 Mart 2012].
19. KWAK, J., 2007, *Ubiquitous Services Sytem Based on SIP*, IEEE Transactions on Consumer Electronics, 53 (3), 938-944.
20. BERTRAN, B.; CONSEL, C; KADIONIK, P. and LAMER, B., 2009, *A SIP-Based Home Automation Platform*, 13th International Conference on Intelligence in Next Generation Networks, Cite Mondiale, Bordeaux, France , Oct. 26-29 2009.
21. KHALIFA, T.; NAIK, K. and NAYAK, A., 2011, *A Survey of Communication Protocols for Automatic Meter Reading Applications*, IEEE Communications Surveys & Tutorials, 13 (2), 168-182.
22. WANG, J. and LEUNG, V.C.M., 2011, *A Survey of Technical Requirements and Consumer Application Standards for IP-based Smart Grid AMI Network*, International Conference on Information Networking, Jan. 26-28 2011.

23. JANSEN, B.; BINDING, C.; SUNDSTÖRM, O. and GANTENBEIN, D., 2010, *Architecture and Communication of an Electric Vehicle Virtual Power Plant*, 2010 Proc. 1st IEEE Int'l Conf. on Smart Grid Communications "IEEE SmartGridComm 2010," Gaithersburg, Maryland (IEEE, October 2010) 149-154.
24. ROYCHOWDHURY, A., 2009, *Realizing the OpenADR Demand and Response Specification using SIP* [Online], 2009, http://www.sipforum.org/component/option,com_docman/task,cat_view/gid,95/Itemid,261/ [Ziyaret Tarihi: 26 Mart 2012].
25. MOYER, S.; MARPLES, D.; TSANG, S.; KATZ, Y.; GURUNG, P.; CHENG, T.; DUTTA, A.; SCHULZRINNE, H. ve ROYCHODHURY, A., 2001, *Framework Draft for Networked Appliances using the Session Initiation Protocol* [online], <http://tools.ietf.org/html/draft-moyer-sip-appliances-framework-02> [Ziyaret Tarihi: 26 Mart 2012].
26. HAUTAKOPRI, J.; CAMARILLO, G.; PENFIELD, R.; HAWRYLYSHEN, A. and BHATIA, M., 2010, *Requirements from Session Initiation Protocol (SIP) Session Border Control (SBC) Deployments*, RFC 5853, Internet Engineering Task Force, <http://tools.ietf.org/html/rfc5853> [Ziyaret Tarihi: 9 Nisan 2012].
27. S3 Session Border Controller [online], <http://www.genband.com/products/sbc-and-security/s3-session-border-controller> [Ziyaret Tarihi: 10 Nisan 2012].
28. *Welcome to Kamailio (Openser) – the Open Source SIP Server* [online], <http://www.kamailio.org/w/> [Ziyaret Tarihi: 10 Nisan 2012].
29. TOMIC, S. and TEODOROVA, P., *SIP meets ZigBee*, 16th Mobile and Wireless Communications Summit, Budapest Hungary, 1-5 July, 2007.
30. O'DOHERTY, P., 2003, *SIP and the Java Platforms* [online], <http://www.oracle.com/technetwork/java/sip-and-java-137921.html> [Ziyaret Tarihi: 10 Nisan 2012].
31. *Trail: Creating a GUI with JFC/Swing* [online], <http://docs.oracle.com/javase/tutorial/uiswing/> [Ziyaret Tarihi: 12 Nisan 2012].
32. *Abstract Window Toolkit* [online], <http://docs.oracle.com/javase/6/docs/technotes/guides/awt/> [Ziyaret Tarihi: 12 Nisan 2012].
33. *Wireshark's User's Guide for Wireshark 1.7* [online], http://www.wireshark.org/docs/wsug_html_chunked/ [Ziyaret Tarihi: 16 Nisan 2012].
34. *Session Initiation Protocol (SIP) Event Types Namespace* [online], <http://www.iana.org/assignments/sip-events/sip-events.xml> [Ziyaret Tarihi: 17 Nisan 2012].

35. Digi International, 2012, *XBEE /XBEE PRO ZB RF Modules* [online], http://ftp1.digi.com/support/documentation/90000976_K.pdf [Ziyaret Tarihi: 18 Nisan 2012].
36. Texas Instruments, 2012, *MSP-EXP430G2 LaunchPad Experimenter Board User's Guide* [online], <http://www.ti.com/lit/ug/slau318b/slau318b.pdf> [Ziyaret Tarihi: 18 Nisan 2012].
37. *Xbee API : A Java API for Digi XBee/XBee-Pro OEM RF Modules* [online], <http://code.google.com/p/xbee-api/> [Ziyaret Tarihi: 18 Nisan 2012].

EKLER

Ek-A

SIPAutoClient.java

```
package SIPAutoClient;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.net.InetAddress;
import java.net.UnknownHostException;
import java.text.ParseException;
import java.util.ArrayList;
import java.util.Properties;
import javax.sip.InvalidArgumentException;
import javax.sip.ListeningPoint;
import javax.sip.RequestEvent;
import javax.sip.ResponseEvent;
import javax.sip.SipException;
import javax.sip.SipFactory;
import javax.sip.SipListener;
import javax.sip.SipProvider;
import javax.sip.SipStack;
import javax.sip.TimeoutEvent;
import javax.sip.address.Address;
import javax.sip.address.AddressFactory;
import javax.sip.address.SipURI;
import javax.sip.header.CSeqHeader;
import javax.sip.header.CallIdHeader;
import javax.sip.header.ContactHeader;
import javax.sip.header.ContentTypeHeader;
import javax.sip.header.FromHeader;
import javax.sip.header.HeaderFactory;
import javax.sip.header.MaxForwardsHeader;
import javax.sip.header.ToHeader;
import javax.sip.header.ViaHeader;
import javax.sip.message.MessageFactory;
import javax.sip.message.Request;
import javax.sip.message.Response;
import javax.swing.JButton;
```

```

import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;

public class SIPAutoClient
    extends JFrame
    implements MessageProcessorClient
{
    private SipLayerAutoClient sipLayer;
    private JTextField fromAddress;
    private JLabel fromLbl;
    private JLabel receivedLbl;
    private JTextArea receivedMessages;
    private JScrollPane receivedScrollPane;
    private JTextField toAddress;
    private JLabel toLbl;
    private JLabel subsEventLbl;
    private JTextField subsEvent;
    private JButton optBtn;
    private JLabel optLbl;
    private JButton subsBtn;
    private JLabel subsLbl;
    private JButton ampulAcBtn;
    private JLabel ampulAcLbl;
    private JButton ampulKapaBtn;
    private JLabel ampulKapaLbl;

    public static void main(String[] args)
    {
        if(args.length != 2) {
            printUsage();
            System.exit(-1);
        }

        try
        {
            String username = args[0];
            int port = Integer.parseInt(args[1]);
            String ip = InetAddress.getLocalHost().getHostAddress();

            SipLayerAutoClient sipLayer = new
SipLayerAutoClient(username, ip, port);
            SIPAutoClient SIPAc = new SIPAutoClient(sipLayer);
            sipLayer.setMessageProcessor(SIPAc);
            SIPAc.show();
        }
        catch (Throwable e) {

```

```

        System.out.println("SIP stack'in baslatilmasinda problemle karsilasildi");
        e.printStackTrace();
        System.exit(-1);
    }
}

private static void printUsage()
{
    System.out.println("Kullanim:");
    System.out.println(" java -jar SIPAutoClient.jar <kullanici adi> <port>");
    System.out.println("<kullanici adi> kullanicinin SIP kullanici adidir");
    System.out.println("ve <port> kullanilacak port numarasidir");
    System.out.println("Ornegin:");
    System.out.println(" java -jar SIPAutoClient.jar 5060");
}

public SIPAutoClient(SipLayerAutoClient sip)
{
    super();
    sipLayer = sip;
    initWindow();
    String from = "sip:" + sip.getUsername() + "@" + sip.getHost() + ":" +
sip.getPort();
    this.fromAddress.setText(from);
}

private void initWindow() {
    receivedLbl = new JLabel();
    receivedScrollPane = new JScrollPane();
    receivedMessages = new JTextArea();
    fromLbl = new JLabel();
    fromAddress = new JTextField();
    toLbl = new JLabel();
    toAddress = new JTextField();
    subsEventLbl = new JLabel();
    subsEvent= new JTextField();
    optBtn= new JButton();
    subsBtn = new JButton();
    ampulAcBtn = new JButton();
    ampulKapaBtn = new JButton();

    getContentPane().setLayout(null);

    setTitle("SIPAutoClient");
    addWindowListener(new WindowAdapter() {
        public void windowClosing(WindowEvent evt) {
            System.exit(0);
        }
    });
}

```

```

receivedLbl.setText("Alinan Iletiler:");
receivedLbl.setAlignmentY(0.0F);
receivedLbl.setPreferredSize(new java.awt.Dimension(25, 100));
getContentPane().add(receivedLbl);
receivedLbl.setBounds(5, 0, 136, 20);

```

```

receivedMessages.setAlignmentX(0.0F);
receivedMessages.setEditable(false);
receivedMessages.setLineWrap(true);
receivedMessages.setWrapStyleWord(true);
receivedScrollPane.setViewportView(receivedMessages);

```

```

receivedScrollPane.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLL
LBAR_NEVER);

```

```

getContentPane().add(receivedScrollPane);
receivedScrollPane.setBounds(5, 20, 270, 130);

```

```

fromLbl.setText("From:");
getContentPane().add(fromLbl);
fromLbl.setBounds(5, 200, 35, 15);

```

```

getContentPane().add(fromAddress);
fromAddress.setBounds(40, 200, 235, 20);
fromAddress.setEditable(false);

```

```

toLbl.setText("To:");
getContentPane().add(toLbl);
toLbl.setBounds(5, 225, 35, 15);

```

```

getContentPane().add(toAddress);
toAddress.setBounds(40, 225, 235, 21);

```

```

subsEventLbl.setText("Event:");
getContentPane().add(subsEventLbl);
subsEventLbl.setBounds(5 ,250 ,35, 15);

```

```

getContentPane().add(subsEvent);
subsEvent.setBounds(40, 250, 235, 22);

```

```

optBtn.setText("OPTIONS");
optBtn.addActionListener( new ActionListener() {
    public void actionPerformed(ActionEvent evt) {
        optBtnActionPerformed(evt);
    }
}
);

```

```

getContentPane().add(optBtn);
optBtn.setBounds(20, 300, 120, 30);

```

```

subsBtn.setText("SUBSCRIBE");
subsBtn.addActionListener( new ActionListener() {
    public void actionPerformed(ActionEvent evt) {
        subsBtnActionPerformed(evt);
    }
});

```

```

getContentPane().add(subsBtn);
subsBtn.setBounds(160, 300, 120, 30);

```

```

ampulAcBtn.setText("ISIGI AC");
ampulAcBtn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent evt) {
        ampulAcBtnActionPerformed(evt);
    }
});

```

```

getContentPane().add(ampulAcBtn);
ampulAcBtn.setBounds(20, 340, 120, 30);
if (!isikAcik){
    ampulAcBtn.setEnabled(true);
}
else {ampulAcBtn.setEnabled(false);
}

```

```

ampulKapaBtn.setText("ISIGI KAPA");
ampulKapaBtn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent evt) {
        ampulKapaBtnActionPerformed(evt);
    }
});

```

```

getContentPane().add(ampulKapaBtn);
ampulKapaBtn.setBounds(160, 340, 120, 30);
if (!isikAcik){
    ampulKapaBtn.setEnabled(false);
}

```

```

else {ampulKapaBtn.setEnabled(true);
}

    java.awt.Dimension                screenSize                =
java.awt.Toolkit.getDefaultToolkit().getScreenSize();
    setBounds((screenSize.width-290)/2, (screenSize.height-320)/2, 300, 480);
}

boolean isikAcik=false;

private void optBtnActionPerformed(ActionEvent evt) {

    try
    {
        String to = this.toAddress.getText();
        sipLayer.sendOptions(to);

    } catch (Throwable e)
    {
        e.printStackTrace();
        this.receivedMessages.append("HATA: Options Istegi Gonderilemedi " +
e.getMessage() + "\n");
    }

}

private void subsBtnActionPerformed(ActionEvent evt) {

    try
    {
        String to = this.toAddress.getText();
        String event = this.subsEvent.getText();
        sipLayer.sendSubscribe(to, event);

    } catch (Throwable e)
    {
        e.printStackTrace();
        this.receivedMessages.append("HATA: Subscribe Istegi Gonderilemedi "
+ e.getMessage() + "\n");
    }

}

private void ampulAcBtnActionPerformed(ActionEvent evt) {
    try
    {

```

```

String to = this.toAddress.getText();
String message = "AMPUL AC";
sipLayer.sendMessage(to, message);

} catch (Throwable e)
{
    e.printStackTrace();
    this.receivedMessages.append("HATA: Ampul Acma Istegi Gonderilemedi " +
e.getMessage() + "\n");
}

}

private void ampulKapaBtnActionPerformed(ActionEvent evt) {

    try
    {
        String to = this.toAddress.getText();
        String message = "AMPUL KAPA";
        sipLayer.sendMessage(to, message);

    } catch (Throwable e)
    {
        e.printStackTrace();
        this.receivedMessages.append("HATA:      Ampul      Kapama      Istegi
Gonderilemedi " + e.getMessage() + "\n");
    }

}

public void processMessage(String sender, String message)
{
    this.receivedMessages.append("From " +
sender + ": " + message + "\n");

    if(message.matches("ISIK ACILDI")){

        ampulAcBtn.setEnabled(false);
        ampulKapaBtn.setEnabled(true);
        isikAcik=true;

    }
}

```

```

else if(message.matches("ISIK KAPANDI")){

    ampulAcBtn.setEnabled(true);
    ampulKapaBtn.setEnabled(false);
    isikAcik=false;

}

else if(message.matches("ISIK ACIK DURUMDA")){

    ampulAcBtn.setEnabled(false);
    ampulKapaBtn.setEnabled(true);
    isikAcik=true;

}

else if(message.matches("ISIK KAPALI DURUMDA")){

    ampulAcBtn.setEnabled(true);
    ampulKapaBtn.setEnabled(false);
    isikAcik=false;

}

}

public void processError(String errorMessage)
{
    this.receivedMessages.append("ERROR: " +
        errorMessage + "\n");
}

public void processInfo(String infoMessage)
{
    this.receivedMessages.append(
        infoMessage + "\n");
}

public void processNotify(String sender, String event, String eventContent) {

    this.receivedMessages.append(eventContent + "\n");

    if (eventContent.matches("ISIK KAPALI DURUMDA")){

        isikAcik=false;
        ampulKapaBtn.setEnabled(false);
        ampulAcBtn.setEnabled(true);

    }
}

```

```

else if(eventContent.matches("ISIK ACIK DURUMDA")){

    isikAcik=true;
    ampulKapaBtn.setEnabled(true);
    ampulAcBtn.setEnabled(false);

}

}

}

```

MessageProcessor.java

```

package SIPAutoClient;

public interface MessageProcessorClient
{
    public void processMessage(String sender, String message);
    public void processNotify(String sender, String event, String eventContent);
    public void processError(String errorMessage);
    public void processInfo(String infoMessage);
}

```

SIPAutoLayerClient.java

```

package SIPAutoClient;

import java.net.InetAddress;
import java.text.ParseException;
import java.util.ArrayList;
import java.util.Properties;
import java.util.TooManyListenersException;
import javax.sip.ClientTransaction;
import javax.sip.Dialog;
import javax.sip.DialogTerminatedEvent;
import javax.sip.IOExceptionEvent;
import javax.sip.InvalidArgumentException;
import javax.sip.ListeningPoint;
import javax.sip.ObjectInUseException;
import javax.sip.PeerUnavailableException;
import javax.sip.RequestEvent;
import javax.sip.ResponseEvent;
import javax.sip.ServerTransaction;

```

```

import javax.sip.SipException;
import javax.sip.SipFactory;
import javax.sip.SipListener;
import javax.sip.SipProvider;
import javax.sip.SipStack;
import javax.sip.TimeoutEvent;
import javax.sip.TransactionTerminatedEvent;
import javax.sip.TransportNotSupportedException;
import javax.sip.address.Address;
import javax.sip.address.AddressFactory;
import javax.sip.address.SipURI;
import javax.sip.header.CSeqHeader;
import javax.sip.header.CallIdHeader;
import javax.sip.header.ContactHeader;
import javax.sip.header.ContentLengthHeader;
import javax.sip.header.ContentTypeHeader;
import javax.sip.header.EventHeader;
import javax.sip.header.FromHeader;
import javax.sip.header.Header;
import javax.sip.header.HeaderFactory;
import javax.sip.header.MaxForwardsHeader;
import javax.sip.header.ToHeader;
import javax.sip.header.ViaHeader;
import javax.sip.message.MessageFactory;
import javax.sip.message.Request;
import javax.sip.message.Response;

public class SipLayerAutoClient implements SipListener {

    private MessageProcessorClient messageProcessor;

    private String username;

    private SipStack sipStack;

    private SipFactory sipFactory;

    private AddressFactory addressFactory;

    private HeaderFactory headerFactory;

    private MessageFactory messageFactory;

    private SipProvider sipProvider;
    public SipLayerAutoClient(String username, String ip, int port)
        throws PeerUnavailableException, TransportNotSupportedException,
        InvalidArgumentException, ObjectInUseException,
        TooManyListenersException {

```

```

        setUsername(username);
        sipFactory = SipFactory.getInstance();
        sipFactory.setPathName("gov.nist");
        Properties properties = new Properties();
        properties.setProperty("javax.sip.STACK_NAME", "SIPAutoClient");
        properties.setProperty("javax.sip.IP_ADDRESS", ip);

        properties.setProperty("gov.nist.javasip.TRACE_LEVEL", "32");
        properties.setProperty("gov.nist.javasip.SERVER_LOG",
            "SIPAuto.txt");
        properties.setProperty("gov.nist.javasip.DEBUG_LOG",
            "SIPAutodebug.log");
        properties.setProperty("gov.nist.javasip.DELIVER_UNSOLICITED_NOTIFY",
            "true");

        sipStack = sipFactory.createSipStack(properties);
        headerFactory = sipFactory.createHeaderFactory();
        addressFactory = sipFactory.createAddressFactory();
        messageFactory = sipFactory.createMessageFactory();

        ListeningPoint tcp = sipStack.createListeningPoint(port, "tcp");
        ListeningPoint udp = sipStack.createListeningPoint(port, "udp");

        sipProvider = sipStack.createSipProvider(tcp);
        sipProvider.addSipListener(this);
        sipProvider = sipStack.createSipProvider(udp);
        sipProvider.addSipListener(this);

    }

    public void sendMessage(String to, String message) throws ParseException,
        InvalidArgumentException, SipException {

        SipURI from = addressFactory.createSipURI(getUsername(), getHost()
            + ":" + getPort());
        Address fromNameAddress = addressFactory.createAddress(from);
        fromNameAddress.setDisplayName(getUsername());
        FromHeader fromHeader = headerFactory.createFromHeader(fromNameAddress,
            "SIPAutoClient1.0");

        String username = to.substring(to.indexOf(":") + 1, to.indexOf("@"));
        String address = to.substring(to.indexOf("@") + 1);

        SipURI toAddress = addressFactory.createSipURI(username, address);
        Address toNameAddress = addressFactory.createAddress(toAddress);
        toNameAddress.setDisplayName(username);
        ToHeader toHeader = headerFactory.createToHeader(toNameAddress, null);
    }

```

```
SipURI requestURI = addressFactory.createSipURI(username, address);
requestURI.setTransportParam("udp");
```

```
ArrayList viaHeaders = new ArrayList();
ViaHeader viaHeader = headerFactory.createViaHeader(getHost(),
    getPort(), "udp", "branch1");
viaHeaders.add(viaHeader);
```

```
CallIdHeader callIdHeader = sipProvider.getNewCallId();
```

```
CSeqHeader cSeqHeader = headerFactory.createCSeqHeader(1,
    Request.MESSAGE);
```

```
MaxForwardsHeader maxForwards = headerFactory
    .createMaxForwardsHeader(70);
```

```
Request request = messageFactory.createRequest(requestURI,
    Request.MESSAGE, callIdHeader, cSeqHeader, fromHeader,
    toHeader, viaHeaders, maxForwards);
```

```
SipURI contactURI = addressFactory.createSipURI(getUsername(),
    getHost());
contactURI.setPort(getPort());
Address contactAddress = addressFactory.createAddress(contactURI);
contactAddress.setDisplayName(getUsername());
ContactHeader contactHeader = headerFactory
    .createContactHeader(contactAddress);
request.addHeader(contactHeader);
```

```
ContentTypeHeader contentTypeHeader = headerFactory
    .createContentTypeHeader("text", "plain");
request.setContent(message, contentTypeHeader);
```

```
sipProvider.sendRequest(request);
```

```
}
```

```
public void sendOptions(String to) throws ParseException,
    InvalidArgumentException, SipException {
```

```
    SipURI from = addressFactory.createSipURI(getUsername(), getHost()
        + ":" + getPort());
    Address fromNameAddress = addressFactory.createAddress(from);
    fromNameAddress.setDisplayName(getUsername());
    FromHeader fromHeader =
        headerFactory.createFromHeader(fromNameAddress,
            "SIPAutoClient1.0");
```

```
    String username = to.substring(to.indexOf(":") + 1, to.indexOf("@"));
```

```

String address = to.substring(to.indexOf("@") + 1);

SipURI toAddress = addressFactory.createSipURI(username, address);
Address toNameAddress = addressFactory.createAddress(toAddress);
toNameAddress.setDisplayName(username);
ToHeader toHeader = headerFactory.createToHeader(toNameAddress,
null);

SipURI requestURI = addressFactory.createSipURI(username, address);
requestURI.setTransportParam("udp");

ArrayList viaHeaders = new ArrayList();
ViaHeader viaHeader = headerFactory.createViaHeader(getHost(),
    getPort(), "udp", "branch1");
viaHeaders.add(viaHeader);

CallIdHeader callIdHeader = sipProvider.getNewCallId();

CSeqHeader cSeqHeader = headerFactory.createCSeqHeader(1,
    Request.OPTIONS);

MaxForwardsHeader maxForwards = headerFactory
    .createMaxForwardsHeader(70);

Request request = messageFactory.createRequest(requestURI,
    Request.OPTIONS, callIdHeader, cSeqHeader, fromHeader,
    toHeader, viaHeaders, maxForwards);

SipURI contactURI = addressFactory.createSipURI(getUsername(),
    getHost());
contactURI.setPort(getPort());
Address contactAddress = addressFactory.createAddress(contactURI);
contactAddress.setDisplayName(getUsername());
ContactHeader contactHeader = headerFactory
    .createContactHeader(contactAddress);
request.addHeader(contactHeader);

sipProvider.sendRequest(request);
}

public void sendSubscribe(String to,String event) throws ParseException,
InvalidArgumentException, SipException {

    SipURI from = addressFactory.createSipURI(getUsername(), getHost()
        + ":" + getPort());
    Address fromNameAddress = addressFactory.createAddress(from);
    fromNameAddress.setDisplayName(getUsername());

```

```

        FromHeader fromHeader =
headerFactory.createFromHeader(fromNameAddress,
        "SipAutoClient1.0");

String username = to.substring(to.indexOf(":") + 1, to.indexOf("@"));
String address = to.substring(to.indexOf("@") + 1);

SipURI toAddress = addressFactory.createSipURI(username, address);
Address toNameAddress = addressFactory.createAddress(toAddress);
toNameAddress.setDisplayName(username);
ToHeader toHeader = headerFactory.createToHeader(toNameAddress,
null);

SipURI requestURI = addressFactory.createSipURI(username, address);
requestURI.setTransportParam("udp");

ArrayList viaHeaders = new ArrayList();
ViaHeader viaHeader = headerFactory.createViaHeader(getHost(),
        getPort(), "udp", "branch1");
viaHeaders.add(viaHeader);

CallIdHeader callIdHeader = sipProvider.getNewCallId();

CSeqHeader cSeqHeader = headerFactory.createCSeqHeader(1,
        Request.SUBSCRIBE);

MaxForwardsHeader maxForwards = headerFactory
        .createMaxForwardsHeader(70);

Request request = messageFactory.createRequest(requestURI,
        Request.SUBSCRIBE, callIdHeader, cSeqHeader, fromHeader,
        toHeader, viaHeaders, maxForwards);

SipURI contactURI = addressFactory.createSipURI(getUsername(),
        getHost());
contactURI.setPort(getPort());
Address contactAddress = addressFactory.createAddress(contactURI);
contactAddress.setDisplayName(getUsername());
ContactHeader contactHeader = headerFactory
        .createContactHeader(contactAddress);
request.addHeader(contactHeader);
EventHandler eventHeader = headerFactory.createEventHandler(event);
request.addHeader(eventHeader);

Header extensionHeader = headerFactory.createHeader("Expires",
"3600");
request.addHeader(extensionHeader);

```

```

        sipProvider.sendRequest(request);

    }

    public void sendRegister(String to) throws ParseException,
        InvalidArgumentException, SipException {

        SipURI from = addressFactory.createSipURI(getUsername(), getHost()
            + ":" + getPort());
        Address fromNameAddress = addressFactory.createAddress(from);
        fromNameAddress.setDisplayName(getUsername());
        FromHeader fromHeader = headerFactory.createFromHeader(fromNameAddress,
            "SIPAutoClient1.0");

        String username = to.substring(to.indexOf(":") + 1, to.indexOf("@"));
        String address = to.substring(to.indexOf("@") + 1);

        SipURI toAddress = addressFactory.createSipURI(username, address);
        Address toNameAddress = addressFactory.createAddress(toAddress);
        toNameAddress.setDisplayName(username);
        ToHeader toHeader = headerFactory.createToHeader(toNameAddress,
            null);

        SipURI requestURI = addressFactory.createSipURI(username, address);
        requestURI.setTransportParam("udp");

        ArrayList viaHeaders = new ArrayList();
        ViaHeader viaHeader = headerFactory.createViaHeader(getHost(),
            getPort(), "udp", "branch1");
        viaHeaders.add(viaHeader);

        CallIdHeader callIdHeader = sipProvider.getNewCallId();

        CSeqHeader cSeqHeader = headerFactory.createCSeqHeader(1,
            Request.REGISTER);

        MaxForwardsHeader maxForwards = headerFactory
            .createMaxForwardsHeader(70);

        Request request = messageFactory.createRequest(requestURI,
            Request.REGISTER, callIdHeader, cSeqHeader, fromHeader,
            toHeader, viaHeaders, maxForwards);

        SipURI contactURI = addressFactory.createSipURI(getUsername(),
            getHost());
        contactURI.setPort(getPort());
    }

```

```

Address contactAddress = addressFactory.createAddress(contactURI);
contactAddress.setDisplayName(getUsername());
ContactHeader contactHeader = headerFactory
    .createContactHeader(contactAddress);
request.addHeader(contactHeader);

Header extensionHeader = headerFactory.createHeader("Expires",
    "3600");
request.addHeader(extensionHeader);

ClientTransaction registerTid =
sipProvider.getClientTransaction(request);

registerTid.sendRequest();

Dialog dialog = registerTid.getDialog();

}

public void processResponse(ResponseEvent evt) {
    Response response = evt.getResponse();
    int status = response.getStatusCode();
    CSeqHeader CSeq = (CSeqHeader)response.getHeader("CSeq");
    String method = CSeq.getMethod();

    if(method.equals("SUBSCRIBE")){

        if ((status >= 200) && (status < 300)) {
            messageProcessor.processInfo("Kayit basariyla yapildi");
            return;
        }

        else if(status==489){
            messageProcessor.processInfo("Belirtilen olay bulunamadi");
            return;
        }
        else if (status == 100){
            messageProcessor.processInfo("METOD ISLEMDE");
            return;
        }
        messageProcessor.processError("Bilinmeyen yanit : " + status);
    }
}

```

```

        if ((status >= 200) && (status < 300)) {
            messageProcessor.processInfo("Istek basariyla gonderildi");
            return;
        }

        messageProcessor.processError("Istek gonderilemedi: " + status);
    }

    public void processRequest(RequestEvent evt) {
        Request req = evt.getRequest();
        String method = req.getMethod();
        if (method.equals("MESSAGE")) {

            FromHeader from = (FromHeader) req.getHeader("From");
            messageProcessor.processMessage(from.getAddress().toString(),
                new String(req.getRawContent()));
            Response response = null;
            try {
                response = messageFactory.createResponse(200, req);
                ToHeader toHeader = (ToHeader) response.getHeader(ToHeader.NAME);
                toHeader.setTag("987");
                ServerTransaction st = sipProvider.getNewServerTransaction(req);
                st.sendResponse(response);
            } catch (Throwable e) {
                e.printStackTrace();
                messageProcessor.processError("OK cevabi gonderilemedi");
            }
        }

        else if (method.equals("NOTIFY")){

            FromHeader from = (FromHeader) req.getHeader("From");
            EventHeader event = (EventHeader) req.getHeader("Event");
            messageProcessor.processNotify(from.getAddress().toString(),
            event.getEventType(),new String(req.getRawContent()));

            Response response = null;
            try {
                response = messageFactory.createResponse(200, req);
                ToHeader toHeader = (ToHeader) response.getHeader(ToHeader.NAME);
                toHeader.setTag("987");
                ServerTransaction st = sipProvider.getNewServerTransaction(req);
                st.sendResponse(response);
            } catch (Throwable e) {

```

```

        e.printStackTrace();
        messageProcessor.processError("OK cevabi gonderilemedi");
    }

}

else{
    messageProcessor.processError("Yanlis istek metodu: " + method);
    return;
}
}

public void processTimeout(TimeoutEvent evt) {
    messageProcessor
        .processError("Bir onceki istek gonderilemedi: " + "sure asimi");
}

public void processIOException(IOExceptionEvent evt) {
    messageProcessor.processError("Bir onceki istek gonderilemedi: "
        + "I/O Exception");
}

}

public void processDialogTerminated(DialogTerminatedEvent evt) {
}

}

public void processTransactionTerminated(TransactionTerminatedEvent evt) {
}

}

public String getHost() {
    int port = sipProvider.getListeningPoint().getPort();
    String host = sipStack.getIPAddress();
    return host;
}

}

public int getPort() {
    int port = sipProvider.getListeningPoint().getPort();
    return port;
}

}

public String getUsername() {
    return username;
}

}

public void setUsername(String newUsername) {
    username = newUsername;
}

}

```

```
public MessageProcessorClient getMessageProcessor() {  
    return messageProcessor;  
}  
  
public void setMessageProcessor(MessageProcessorClient newMessageProcessor) {  
    messageProcessor = newMessageProcessor;  
}  
}
```

Ek-B**SIPAuto.java**

```

package SIPAuto;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.net.InetAddress;
import java.net.UnknownHostException;
import java.text.ParseException;
import java.util.ArrayList;
import java.util.Properties;
import java.util.StringTokenizer;
import javax.sip.InvalidArgumentException;
import javax.sip.ListeningPoint;
import javax.sip.RequestEvent;
import javax.sip.ResponseEvent;
import javax.sip.SipException;
import javax.sip.SipFactory;
import javax.sip.SipListener;
import javax.sip.SipProvider;
import javax.sip.SipStack;
import javax.sip.TimeoutEvent;
import javax.sip.address.Address;
import javax.sip.address.AddressFactory;
import javax.sip.address.SipURI;
import javax.sip.header.CSeqHeader;
import javax.sip.header.CallIdHeader;
import javax.sip.header.ContactHeader;
import javax.sip.header.ContentTypeHeader;
import javax.sip.header.FromHeader;
import javax.sip.header.HeaderFactory;
import javax.sip.header.MaxForwardsHeader;
import javax.sip.header.ToHeader;
import javax.sip.header.ViaHeader;
import javax.sip.message.MessageFactory;
import javax.sip.message.Request;
import javax.sip.message.Response;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;

```

```

public class SIPAuto extends JFrame implements MessageProcessor
{
    private SipLayerAuto sipLayer;

    private JTextField fromAddress;
    private JLabel fromLbl;
    private JLabel receivedLbl;
    private JTextArea receivedMessages;
    private JScrollPane receivedScrollPane;
    private JButton ampulBtn;

    public static void main(String[] args)
    {
        if(args.length != 2) {
            printUsage();
            System.exit(-1);
        }

        try
        {
            String username = args[0];
            int port = Integer.parseInt(args[1]);
            String ip = InetAddress.getLocalHost().getHostAddress();

            SipLayerAuto sipLayer = new SipLayerAuto(username,
ip, port);

            SIPAuto sa = new SIPAuto(sipLayer);
            sipLayer.setMessageProcessor(sa);
            sa.show();
        } catch (Throwable e)
        {
            System.out.println("SIP      stack'in      baslatilmasinda      problemle
karsilasildi.");
            e.printStackTrace();
            System.exit(-1);
        }
    }

    private static void printUsage()
    {
        System.out.println("Kullanim:");
        System.out.println(" java -jar SIPAuto.jar <kullanici adi> <port>");
        System.out.println("<kullanici adi> kulanicinin SIP adidir");
        System.out.println("<port> kulanilacak olan port numarisidir");
        System.out.println("Ornegin:");
        System.out.println(" java -jar SIPAuto.jar server 5060");
    }
}

```

```

public SIPAuto(SipLayerAuto sip)
{
    super();
    sipLayer = sip;
    initWindow();
    String from = "sip:" + sip.getUsername() + "@" + sip.getHost() + ":" +
sip.getPort();
    this.fromAddress.setText(from);
}

private void initWindow() {
    receivedLbl = new JLabel();
    receivedScrollPane = new JScrollPane();
    receivedMessages = new JTextArea();
    fromLbl = new JLabel();
    fromAddress = new JTextField();
    ampulBtn = new JButton();

    getContentPane().setLayout(null);

    setTitle("SIPAuto");
    addWindowListener(new WindowAdapter() {
        public void windowClosing(WindowEvent evt) {
            System.exit(0);
        }
    });

    receivedLbl.setText("Alinan Iletiler:");
    receivedLbl.setAlignmentY(0.0F);
    receivedLbl.setPreferredSize(new java.awt.Dimension(25, 100));
    getContentPane().add(receivedLbl);
    receivedLbl.setBounds(5, 0, 136, 20);

    receivedMessages.setAlignmentX(0.0F);
    receivedMessages.setEditable(false);
    receivedMessages.setLineWrap(true);
    receivedMessages.setWrapStyleWord(true);
    receivedScrollPane.setViewportView(receivedMessages);

    receivedScrollPane.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLL
    LBAR_NEVER);

    getContentPane().add(receivedScrollPane);
    receivedScrollPane.setBounds(5, 20, 270, 130);

    fromLbl.setText("From:");
    getContentPane().add(fromLbl);
    fromLbl.setBounds(5, 200, 35, 15);

```

```

        getContentPane().add(fromAddress);
        fromAddress.setBounds(40, 200, 235, 20);
        fromAddress.setEditable(false);

        getContentPane().add(ampulBtn);
        ampulBtn.setIcon(ampul);
        ampulBtn.setBounds(40, 250, 200, 317);
        ampulBtn.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent ampulEvt) {
                ampulBtnActionPerformed(ampulEvt);
            }
        });

        java.awt.Dimension screenSize =
        java.awt.Toolkit.getDefaultToolkit().getScreenSize();
        setBounds((screenSize.width-288)/2, (screenSize.height-310)/2, 300, 740);
    }

    boolean ampulSubscribe=false;
    String subscribedSender= "uyeYok";
    String subscribedEvent= "olayYok";
    boolean isikAcik=false;

    ImageIcon ampul = new ImageIcon("ampul.png");
    ImageIcon ampul2 = new ImageIcon("ampul2.png");

    private void ampulBtnActionPerformed(ActionEvent ampulEvt) {

        if (!isikAcik){
            ampulBtn.setIcon(ampul2);
            isikAcik=true;
            if(ampulSubscribe==true){

                try {
                    sipLayer.sendNotify(subscribedSender,
subscribedEvent, "ISIK ACIK DURUMDA");
                } catch (ParseException |
InvalidArgumentException
                    | SipException e) {
                    e.printStackTrace();
                }
            }
        }
    }

```

```

else {

    ampulBtn.setIcon(ampul);
    isikAcik=false;

    if(ampulSubscribe==true){

        try {
            sipLayer.sendNotify(subscribedSender,
subscribedEvent, "ISIK KAPALI DURUMDA");
        } catch (ParseException |
InvalidArgumentException
                | SipException e) {

            e.printStackTrace();

        }

    }

}

}

public void processMessage(String sender, String message)
{
    this.receivedMessages.append("From " +
        sender + ": " + message + "\n");

    if (message.matches("AMPUL AC")) {
        if (!isikAcik){
            try
            {
                this.processInfo("ISIK ACILDI");
                isikAcik=true;
                String to = sender;
                String cevap = "ISIK ACILDI";
                sipLayer.sendMessage(to, cevap);
                this.ampulBtn.setIcon(ampul2);

            } catch (Throwable e)

            {
                e.printStackTrace();
                this.receivedMessages.append("Mesaji gonderirken hata olustu: " +
e.getMessage() + "\n");
            }

        }

    }
    else {

```

```

        try
        {
            this.processInfo("ISIK ZATEN ACIK");
            String to = sender;
            String cevap = "ISIK ZATEN ACIK";
            sipLayer.sendMessage(to, cevap);
        } catch (Throwable e)
        {
            e.printStackTrace();
            this.receivedMessages.append("Mesaji gonderirken hata olustu: " +
e.getMessage() + "\n");
        }

    }

    }else if(message.matches("AMPUL KAPA")){
        if (isikAcik){

            try
            {
                this.processInfo("ISIK KAPANDI");
                isikAcik=false;
                String to = sender;
                String cevap = "ISIK KAPANDI";
                sipLayer.sendMessage(to, cevap);
                this.ampulBtn.setIcon(ampul);
            } catch (Throwable e)
            {
                e.printStackTrace();
                this.receivedMessages.append("Mesaji gonderirken hata olustu: " +
e.getMessage() + "\n");
            }

        }

    }
    else {

        try
        {
            this.processInfo("ISIK ZATEN KAPALI");
            String to = sender;
            String cevap = "ISIK ZATEN KAPALI";
            sipLayer.sendMessage(to, cevap);
        } catch (Throwable e)
        {
            e.printStackTrace();
            this.receivedMessages.append("Mesaji gonderirken hata olustu: " +
e.getMessage() + "\n");
        }

    }

```

```

    }
}

public void processError(String errorMessage)
{
    this.receivedMessages.append("ERROR: " +
        errorMessage + "\n");
}

public void processInfo(String infoMessage)
{
    this.receivedMessages.append(
        infoMessage + "\n");
}

@Override
public void processOptions(String sender) {

    if (!isikAcik){

        String capability="ISIK KAPALI DURUMDA";
        try {
            sipLayer.sendMessage(sender,capability);
        } catch (ParseException | InvalidArgumentException
            | SipException e) {

            e.printStackTrace();
        }
    }

    else {

        String capability="ISIK ACIK DURUMDA";
        try {
            sipLayer.sendMessage(sender,capability);
        } catch (ParseException | InvalidArgumentException
            | SipException e) {

            e.printStackTrace();
        }
    }

}

@Override
public void processSubscribe(String sender, String event) {

```

```

        if (event.matches("ampulEvent")){

            ampulSubscribe=true;
            subscribedSender= sender;
            subscribedEvent=event;
            this.processInfo("EVENT KAYDI YAPILDI");

            if(!isikAcik){

                try {
                    sipLayer.sendNotify(sender, event, "ISIK
KAPALI DURUMDA");
                } catch (ParseException |
InvalidArgumentException | SipException e) {
                    e.printStackTrace();
                }
            }
            else{
                try {
                    sipLayer.sendNotify(sender, event, "ISIK
ACIK DURUMDA");
                } catch (ParseException |
InvalidArgumentException | SipException e) {
                    e.printStackTrace();
                }
            }

        }

        else{

            ampulSubscribe=false;

        }

    }

}

```

MessageProcessor.java

```
package SIPAuto;

public interface MessageProcessor
{
    public void processMessage(String sender, String message);
    public void processOptions(String sender);
    public void processSubscribe(String sender, String event);
    public void processError(String errorMessage);
    public void processInfo(String infoMessage);
}
```

SIPLayerAuto.java

```
package SIPAuto;

import java.net.InetAddress;
import java.text.ParseException;
import java.util.ArrayList;
import java.util.Properties;
import java.util.TooManyListenersException;

import javax.sip.ClientTransaction;
import javax.sip.Dialog;
import javax.sip.DialogTerminatedEvent;
import javax.sip.IOExceptionEvent;
import javax.sip.InvalidArgumentException;
import javax.sip.ListeningPoint;
import javax.sip.ObjectInUseException;
import javax.sip.PeerUnavailableException;
import javax.sip.RequestEvent;
import javax.sip.ResponseEvent;
import javax.sip.ServerTransaction;
import javax.sip.SipException;
import javax.sip.SipFactory;
import javax.sip.SipListener;
import javax.sip.SipProvider;
import javax.sip.SipStack;
import javax.sip.TimeoutEvent;
import javax.sip.TransactionTerminatedEvent;
import javax.sip.TransportNotSupportedException;
import javax.sip.address.Address;
import javax.sip.address.AddressFactory;
import javax.sip.address.SipURI;
import javax.sip.header.CSeqHeader;
import javax.sip.header.CallIdHeader;
import javax.sip.header.ContactHeader;
import javax.sip.header.ContentTypeHeader;
```

```

import javax.sip.header.EventHeader;
import javax.sip.header.ExpiresHeader;
import javax.sip.header.FromHeader;
import javax.sip.header.Header;
import javax.sip.header.HeaderFactory;
import javax.sip.header.MaxForwardsHeader;
import javax.sip.header.SubscriptionStateHeader;
import javax.sip.header.ToHeader;
import javax.sip.header.ViaHeader;
import javax.sip.message.MessageFactory;
import javax.sip.message.Request;
import javax.sip.message.Response;

public class SipLayerAuto implements SipListener {

    private MessageProcessor messageProcessor;

    private String username;

    private SipStack sipStack;

    private SipFactory sipFactory;

    private AddressFactory addressFactory;

    private HeaderFactory headerFactory;

    private MessageFactory messageFactory;

    private SipProvider sipProvider;

    public SipLayerAuto(String username, String ip, int port)
        throws PeerUnavailableException, TransportNotSupportedException,
        InvalidArgumentException, ObjectInUseException,
        TooManyListenersException {
        setUsername(username);
        sipFactory = SipFactory.getInstance();
        sipFactory.setPathName("gov.nist");
        Properties properties = new Properties();
        properties.setProperty("javax.sip.STACK_NAME", "SIPAuto");
        properties.setProperty("javax.sip.IP_ADDRESS", ip);

        properties.setProperty("gov.nist.java.sip.TRACE_LEVEL", "32");
        properties.setProperty("gov.nist.java.sip.SERVER_LOG",
            "SIPAuto.txt");
        properties.setProperty("gov.nist.java.sip.DEBUG_LOG",
            "SIPAuto.log");
    }

```

```

sipStack = sipFactory.createSipStack(properties);
headerFactory = sipFactory.createHeaderFactory();
addressFactory = sipFactory.createAddressFactory();
messageFactory = sipFactory.createMessageFactory();

ListeningPoint tcp = sipStack.createListeningPoint(port, "tcp");
ListeningPoint udp = sipStack.createListeningPoint(port, "udp");

sipProvider = sipStack.createSipProvider(tcp);
sipProvider.addSipListener(this);
sipProvider = sipStack.createSipProvider(udp);
sipProvider.addSipListener(this);
}

public void processResponse(ResponseEvent evt) {
    Response response = evt.getResponse();
    int status = response.getStatusCode();

    if ((status >= 200) && (status < 300)) {
        messageProcessor.processInfo("Istek basariyla gonderildi");
        return;
    }
    else if (status == 100){
        messageProcessor.processInfo("METOD ISLEMDE");
        return;
    }

    messageProcessor.processError("Istek gonderilemedi: " + status);
}

public void processRequest(RequestEvent evt) {
    Request req = evt.getRequest();

    String method = req.getMethod();
    if (method.equals("MESSAGE")) {

        FromHeader from = (FromHeader) req.getHeader("From");
        messageProcessor.processMessage(from.getAddress().toString(),
            new String(req.getRawContent()));
        Response response = null;
    }
}

```

```

try {
    response = messageFactory.createResponse(200, req);
    ToHeader toHeader = (ToHeader) response.getHeader(ToHeader.NAME);
    toHeader.setTag("789");
    ServerTransaction st = sipProvider.getNewServerTransaction(req);
    st.sendResponse(response);
} catch (Throwable e) {
    e.printStackTrace();
    messageProcessor.processError("OK cevabi gonderilemedi.");
}

}

```

```

else if(method.equals("OPTIONS")){
    FromHeader from = (FromHeader) req.getHeader("From");
    messageProcessor.processOptions(from.getAddress().toString());
    Response response = null;

```

```

try {

    response = messageFactory.createResponse(200, req);
    ToHeader toHeader = (ToHeader) response.getHeader(ToHeader.NAME);
    toHeader.setTag("789");
    ServerTransaction st = sipProvider.getNewServerTransaction(req);
    st.sendResponse(response);
} catch (Throwable e) {
    e.printStackTrace();
    messageProcessor.processError("OK cevabi gonderilemedi.");
}

}

```

```

else if(method.equals("SUBSCRIBE")){

```

```

    FromHeader from = (FromHeader) req.getHeader("From");
    EventHeader eventHeader = (EventHeader) req.getHeader("Event");
    messageProcessor.processSubscribe(from.getAddress().toString(),
eventHeader.getEventType().toString());
    Response response = null;

```

```

if (eventHeader.getEventType().toString().equals("ampulEvent")) {

    try {
        response = messageFactory.createResponse(200, req);
        ExpiresHeader expiresHeader = headerFactory.createExpiresHeader(0);
        response.addHeader(expiresHeader);
        ToHeader toHeader = (ToHeader) response.getHeader(ToHeader.NAME);
        toHeader.setTag("789");
        ServerTransaction st = sipProvider.getNewServerTransaction(req);
        st.sendResponse(response);

    } catch (Throwable e) {
        e.printStackTrace();
        messageProcessor.processError("OK cevabi gonderilemedi");
    }

}

else {

    try {
        response = messageFactory.createResponse(489, req);
        ToHeader toHeader = (ToHeader) response.getHeader(ToHeader.NAME);
        toHeader.setTag("789");
        ServerTransaction st = sipProvider.getNewServerTransaction(req);
        st.sendResponse(response);

    } catch (Throwable e) {
        e.printStackTrace();
        messageProcessor.processError("Bad Event cevabi gonderilemedi");
    }

}

else{
    messageProcessor.processError("Yanlis istek: " + method);
    return;
}

}

```

```

public void sendMessage(String to, String message) throws ParseException,
    IllegalArgumentException, SipException {

    SipURI from = addressFactory.createSipURI(getUsername(), getHost()
        + ":" + getPort());
    Address fromNameAddress = addressFactory.createAddress(from);
    fromNameAddress.setDisplayName(getUsername());
    FromHeader fromHeader = headerFactory.createFromHeader(fromNameAddress,
        "SIPAuto1.0");

    String username = to.substring(to.indexOf(":") + 1, to.indexOf("@"));
    String address = to.substring(to.indexOf("@") + 1);

    SipURI toAddress = addressFactory.createSipURI(username, address);
    Address toNameAddress = addressFactory.createAddress(toAddress);
    toNameAddress.setDisplayName(username);
    ToHeader toHeader = headerFactory.createToHeader(toNameAddress, null);

    SipURI requestURI = addressFactory.createSipURI(username, address);
    requestURI.setTransportParam("udp");

    ArrayList<ViaHeader> viaHeaders = new ArrayList<>();
    ViaHeader viaHeader = headerFactory.createViaHeader(getHost(),
        getPort(), "udp", "branch1");
    viaHeaders.add(viaHeader);

    CallIdHeader callIdHeader = sipProvider.getNewCallId();

    CSeqHeader cSeqHeader = headerFactory.createCSeqHeader(1,
        Request.MESSAGE);

    MaxForwardsHeader maxForwards = headerFactory
        .createMaxForwardsHeader(70);

    Request request = messageFactory.createRequest(requestURI,
        Request.MESSAGE, callIdHeader, cSeqHeader, fromHeader,
        toHeader, viaHeaders, maxForwards);

    SipURI contactURI = addressFactory.createSipURI(getUsername(),
        getHost());
    contactURI.setPort(getPort());
    Address contactAddress = addressFactory.createAddress(contactURI);
    contactAddress.setDisplayName(getUsername());
    ContactHeader contactHeader = headerFactory
        .createContactHeader(contactAddress);
    request.addHeader(contactHeader);

    ContentTypeHeader contentTypeHeader = headerFactory

```

```

        .createContentTypeHeader("text", "plain");
request.setContent(message, contentTypeHeader);

sipProvider.sendRequest(request);
}

public void sendNotify(String to,String event,String eventContent) throws
ParseException,
InvalidArgumentException, SipException {

    SipURI from = addressFactory.createSipURI(getUsername(), getHost()
        + ":" + getPort());
    Address fromNameAddress = addressFactory.createAddress(from);
    fromNameAddress.setDisplayName(getUsername());
    FromHeader fromHeader = headerFactory.createFromHeader(fromNameAddress,
        "SipAuto1.0");

    String username = to.substring(to.indexOf(":") + 1, to.indexOf("@"));
    String address = to.substring(to.indexOf("@") + 1);

    SipURI toAddress = addressFactory.createSipURI(username, address);
    Address toNameAddress = addressFactory.createAddress(toAddress);
    toNameAddress.setDisplayName(username);
    ToHeader toHeader = headerFactory.createToHeader(toNameAddress,
null);

    SipURI requestURI = addressFactory.createSipURI(username, address);
    requestURI.setTransportParam("udp");

    ArrayList viaHeaders = new ArrayList();
    ViaHeader viaHeader = headerFactory.createViaHeader(getHost(),
        getPort(), "udp", "branch1");
    viaHeaders.add(viaHeader);

    CallIdHeader callIdHeader = sipProvider.getNewCallId();

    CSeqHeader cSeqHeader = headerFactory.createCSeqHeader(1,
        Request.NOTIFY);

    MaxForwardsHeader maxForwards = headerFactory
        .createMaxForwardsHeader(70);

    Request request = messageFactory.createRequest(requestURI,
        Request.NOTIFY, callIdHeader, cSeqHeader, fromHeader,
        toHeader, viaHeaders, maxForwards);

```

```

        SipURI contactURI = addressFactory.createSipURI(getUsername(),
            getHost());
        contactURI.setPort(getPort());
        Address contactAddress = addressFactory.createAddress(contactURI);
        contactAddress.setDisplayName(getUsername());
        ContactHeader contactHeader = headerFactory
            .createContactHeader(contactAddress);
        request.addHeader(contactHeader);

        SubscriptionStateHeader subscriptionStateHeader =
headerFactory.createSubscriptionStateHeader("active");
        request.addHeader(subscriptionStateHeader);

        EventHeader eventHeader = headerFactory.createEventHeader(event);
        request.addHeader(eventHeader);

        ContentTypeHeader contentTypeHeader = headerFactory
            .createContentTypeHeader("text", "plain");
        request.setContent(eventContent, contentTypeHeader);

        sipProvider.sendRequest(request);
    }

    public void sendRegister(String to) throws ParseException,
        InvalidArgumentException, SipException {

        SipURI from = addressFactory.createSipURI(getUsername(), getHost()
            + ":" + getPort());
        Address fromNameAddress = addressFactory.createAddress(from);
        fromNameAddress.setDisplayName(getUsername());
        FromHeader fromHeader =
headerFactory.createFromHeader(fromNameAddress,
            "SIPAutoClient1.0");

        String username = to.substring(to.indexOf(":") + 1, to.indexOf("@"));
        String address = to.substring(to.indexOf("@") + 1);

        SipURI toAddress = addressFactory.createSipURI(username, address);
        Address toNameAddress = addressFactory.createAddress(toAddress);
        toNameAddress.setDisplayName(username);
        ToHeader toHeader = headerFactory.createToHeader(toNameAddress,
null);

        SipURI requestURI = addressFactory.createSipURI(username, address);
        requestURI.setTransportParam("udp");

        ArrayList viaHeaders = new ArrayList();

```

```

ViaHeader viaHeader = headerFactory.createViaHeader(getHost(),
    getPort(), "udp", "branch1");
viaHeaders.add(viaHeader);

CallIdHeader callIdHeader = sipProvider.getNewCallId();

CSeqHeader cSeqHeader = headerFactory.createCSeqHeader(1,
    Request.REGISTER);

MaxForwardsHeader maxForwards = headerFactory
    .createMaxForwardsHeader(70);

Request request = messageFactory.createRequest(requestURI,
    Request.REGISTER, callIdHeader, cSeqHeader, fromHeader,
    toHeader, viaHeaders, maxForwards);

SipURI contactURI = addressFactory.createSipURI(getUsername(),
    getHost());
contactURI.setPort(getPort());
Address contactAddress = addressFactory.createAddress(contactURI);
contactAddress.setDisplayName(getUsername());
ContactHeader contactHeader = headerFactory
    .createContactHeader(contactAddress);
request.addHeader(contactHeader);

Header extensionHeader = headerFactory.createHeader("Expires",
    "3600");
request.addHeader(extensionHeader);

ClientTransaction                                registerTid                =
sipProvider.getClientTransaction(request);

registerTid.sendRequest();

Dialog dialog = registerTid.getDialog();

}

public void processTimeout(TimeoutEvent evt) {
    messageProcessor
        .processError("Previous message not sent: " + "timeout");
}

public void processIOException(IOExceptionEvent evt) {
    messageProcessor.processError("Previous message not sent: "
        + "I/O Exception");
}

```

```
public void processDialogTerminated(DialogTerminatedEvent evt) {
}

public void processTransactionTerminated(TransactionTerminatedEvent evt) {
}

public String getHost() {
    int port = sipProvider.getListeningPoint().getPort();
    String host = sipStack.getIPAddress();
    return host;
}

public int getPort() {
    int port = sipProvider.getListeningPoint().getPort();
    return port;
}

public String getUsername() {
    return username;
}

public void setUsername(String newUsername) {
    username = newUsername;
}

public MessageProcessor getMessageProcessor() {
    return messageProcessor;
}

public void setMessageProcessor(MessageProcessor newMessageProcessor) {
    messageProcessor = newMessageProcessor;
}

}
```

Ek-C**SIPAutoRealize.java**

```
package SIPAuto;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.net.InetAddress;
import java.net.UnknownHostException;
import java.text.ParseException;
import java.util.ArrayList;
import java.util.Properties;
import java.util.StringTokenizer;
import javax.sip.InvalidArgumentException;
import javax.sip.ListeningPoint;
import javax.sip.RequestEvent;
import javax.sip.ResponseEvent;
import javax.sip.SipException;
import javax.sip.SipFactory;
import javax.sip.SipListener;
import javax.sip.SipProvider;
import javax.sip.SipStack;
import javax.sip.TimeoutEvent;
import javax.sip.address.Address;
import javax.sip.address.AddressFactory;
import javax.sip.address.SipURI;
import javax.sip.header.CSeqHeader;
import javax.sip.header.CallIdHeader;
import javax.sip.header.ContactHeader;
import javax.sip.header.ContentTypeHeader;
import javax.sip.header.FromHeader;
import javax.sip.header.HeaderFactory;
import javax.sip.header.MaxForwardsHeader;
import javax.sip.header.ToHeader;
import javax.sip.header.ViaHeader;
import javax.sip.message.MessageFactory;
import javax.sip.message.Request;
import javax.sip.message.Response;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JScrollPane;
```

```

import javax.swing.JTextArea;
import javax.swing.JTextField;
import com.rapplogic.xbee.api.ApiId;
import com.rapplogic.xbee.api.PacketListener;
import com.rapplogic.xbee.api.RemoteAtRequest;
import com.rapplogic.xbee.api.RemoteAtResponse;
import com.rapplogic.xbee.api.XBee;
import com.rapplogic.xbee.api.XBeeAddress64;
import com.rapplogic.xbee.api.XBeeException;
import com.rapplogic.xbee.api.XBeeResponse;
import com.rapplogic.xbee.api.XBeeTimeoutException;
import com.rapplogic.xbee.api.zigbee.ZNetRxIoSampleResponse;
import com.rapplogic.xbee.examples.RemoteAtExample;
import org.apache.log4j.Logger;
import org.apache.log4j.PropertyConfigurator;

public class SIPAuto extends JFrame implements MessageProcessor
{
    private SipLayerAuto sipLayer;

    private JTextField fromAddress;
    private JLabel fromLbl;
    private JLabel receivedLbl;
    private JTextArea receivedMessages;
    private JScrollPane receivedScrollPane;
    private JButton ampulBtn;

    private final static Logger log = Logger.getLogger(RemoteAtExample.class);

    public static void main(String[] args)
    {
        if(args.length != 2) {
            printUsage();
            System.exit(-1);
        }

        try
        {
            String username = args[0];
            int port = Integer.parseInt(args[1]);
            String ip = InetAddress.getLocalHost().getHostAddress();

            SipLayerAuto sipLayer = new SipLayerAuto(username,
ip, port);

            SIPAuto sa = new SIPAuto(sipLayer);
            sipLayer.setMessageProcessor(sa);
            sa.show();
        } catch (Throwable e)
        {

```

```

        System.out.println("SIP      stack'in      baslatilmasinda      problemle
karsilasildi.");
        e.printStackTrace();
        System.exit(-1);
    }
    PropertyConfigurator.configure("log4j.properties");
}

private static void printUsage()
{
    System.out.println("Kullanim:");
    System.out.println(" java -jar SIPAuto.jar <kullanici adi> <port>");
    System.out.println("<kullanici adi> kulanicinin SIP adidir");
    System.out.println("<port> kulanilacak olan port numarisidir");
    System.out.println("Ornegin:");
    System.out.println(" java -jar SIPAuto.jar server 5060");
}
public SIPAuto(SipLayerAuto sip)
{
    super();
    sipLayer = sip;
    initWindow();
    String from = "sip:" + sip.getUsername() + "@" + sip.getHost() + ":" +
sip.getPort();
    this.fromAddress.setText(from);
}

private void initWindow() {
    receivedLbl = new JLabel();
    receivedScrollPane = new JScrollPane();
    receivedMessages = new JTextArea();
    fromLbl = new JLabel();
    fromAddress = new JTextField();
    ampulBtn = new JButton();

    getContentPane().setLayout(null);

    setTitle("SIPAuto");
    addWindowListener(new WindowAdapter() {
        public void windowClosing(WindowEvent evt) {
            System.exit(0);
        }
    });

    receivedLbl.setText("Alinan Iletiler:");
    receivedLbl.setAlignmentY(0.0F);
    receivedLbl.setPreferredSize(new java.awt.Dimension(25, 100));
    getContentPane().add(receivedLbl);
    receivedLbl.setBounds(5, 0, 136, 20);

```

```

receivedMessages.setAlignmentX(0.0F);
receivedMessages.setEditable(false);
receivedMessages.setLineWrap(true);
receivedMessages.setWrapStyleWord(true);
receivedScrollPane.setViewportView(receivedMessages);

receivedScrollPane.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLL
LBAR_NEVER);

getContentPane().add(receivedScrollPane);
receivedScrollPane.setBounds(5, 20, 270, 130);

fromLbl.setText("From:");
getContentPane().add(fromLbl);
fromLbl.setBounds(5, 200, 35, 15);

getContentPane().add(fromAddress);
fromAddress.setBounds(40, 200, 235, 20);
fromAddress.setEditable(false);

getContentPane().add(ampulBtn);
ampulBtn.setIcon(ampul);
ampulBtn.setBounds(40, 250, 200, 317);
ampulBtn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent ampulEvt) {
        ampulBtnActionPerformed(ampulEvt);
    }
});

java.awt.Dimension screenSize =
java.awt.Toolkit.getDefaultToolkit().getScreenSize();
setBounds((screenSize.width-288)/2, (screenSize.height-310)/2, 300, 740);
}

boolean ampulSubscribe=false;
    String subscribedSender= "uyeYok";
    String subscribedEvent= "olayYok";
boolean isikAcik=false;

ImageIcon ampul = new ImageIcon("ampul.png");
ImageIcon ampul2 = new ImageIcon("ampul2.png");

private void ampulBtnActionPerformed(ActionEvent ampulEvt) {

```

```

        if (!isikAcik){
            ampulBtn.setIcon(ampul2);
            isikAcik=true;
            if(ampulSubscribe==true){

                try {
                    sipLayer.sendNotify(subscribedSender,
subscribedEvent, "ISIK ACIK DURUMDA");
                } catch (ParseException |
InvalidArgumentException| SipException e) {

                    e.printStackTrace();
                }
            }
        }
        else {

            ampulBtn.setIcon(ampul);
            isikAcik=false;

            if(ampulSubscribe==true){

                try {
                    sipLayer.sendNotify(subscribedSender,
subscribedEvent, "ISIK KAPALI DURUMDA");
                } catch (ParseException |
InvalidArgumentException
                    | SipException e) {

                    e.printStackTrace();
                }
            }
        }
    }
}

XBee xbee = new XBee();

private void xbeeAmpulAc(){

    try {

        xbee.open("COM12", 9600);

```

```

        XBeeAddress64 addr64 = new XBeeAddress64(0, 0x13, 0xa2, 0, 0x40,
0x71, 0x25, 0xd4);

        RemoteAtRequest request = new
RemoteAtRequest(addr64, "D0", new int[] {5});

        RemoteAtResponse response =
(RemoteAtResponse) xbee.sendSynchronous(request, 10000);

        if (response.isOk()) {
            log.info("successfully turned on pin 20
(D0)");
        } else {
            throw new RuntimeException("failed to
turn on pin 20. status is " + response.getStatus());
        }

        xbee.close();

    } catch (XBeeTimeoutException e) {
        log.error("request timed out. make sure you
remote XBee is configured and powered on");
    } catch (Exception e) {
        log.error("unexpected error", e);
    }
}

private void xbeeAmpulKapa(){

    try {

        xbee.open("COM12", 9600);

        XBeeAddress64 addr64 = new XBeeAddress64(0,
0x13, 0xa2, 0, 0x40, 0x71, 0x25, 0xd4);
        RemoteAtRequest request = new
RemoteAtRequest(addr64, "D0", new int[] {4});

        RemoteAtResponse response =
(RemoteAtResponse) xbee.sendSynchronous(request, 10000);

```

```

if (response.isOk()) {
    log.info("successfully turned off pin 20
(D0)");
} else {
    throw new RuntimeException("failed to
turn on pin 20. status is " + response.getStatus());
}
xbee.close();

} catch (XBeeTimeoutException e) {
    log.error("request timed out. make sure you
remote XBee is configured and powered on");
} catch (Exception e) {
    log.error("unexpected error", e);
}
}

public void ZNetIoSample() throws Exception {

    try {
        xbee.open("COM12", 9600);
        xbee.addPacketListener(this);

        synchronized(this) { this.wait(); }
    } finally {
        xbee.close();
    }
}

    int xtrue = 1;

    int xfalse = 1;

    @Override
    public void processResponse(XBeeResponse response) {

        if (response.getApiId() == ApiId.ZNET_IO_SAMPLE_RESPONSE) {
            ZNetRxIoSampleResponse ioSample = (ZNetRxIoSampleResponse)
response;

            log.debug("received i/o sample packet. contains analog is " +
ioSample.containsAnalog() + ", contains digital is " + ioSample.containsDigital());

```

```

log.debug("pin 19 (D1) digital is " + ioSample.isD1On());

if (ioSample.isD1On() == true) {
    isikAcik = true;

    try {

        this.sipLayer.sendNotify(subscribedSender, subscribedEvent, "ISIK ACIK
DURUMDA");

        if (xtrue == 1) {

            XBeeAddress64
            addr64 = new XBeeAddress64(0, 0x13, 0xa2, 0, 0x40, 0x71, 0x25, 0xd4);

            RemoteAtRequest
            request = new RemoteAtRequest(addr64, "D0", new int[] {5});

            RemoteAtResponse
            responseAt = (RemoteAtResponse) xbee.sendSynchronous(request, 10000);

            if
            (responseAt.isOk()) {

                log.info("successfully turned off pin 20 (D0)");

            } else {
                throw new
                RuntimeException("failed to turn on pin 20. status is " + responseAt.getStatus());
            }

            xtrue++;
            xfalse = 1;
        }

        } catch (ParseException |
        InvalidArgumentException | SipException e) {

            e.printStackTrace();
        } catch (XBeeTimeoutException e)

            e.printStackTrace();
        } catch (XBeeException e) {

            e.printStackTrace();
        }
    }
}

```

```

    }

    else {

        try {

            this.sipLayer.sendNotify(subscribedSender, subscribedEvent, "ISIK KAPALI
DURUMDA");

            isikAcik=false;

            if (xfalse==1){
                XBeeAddress64
addr64 = new XBeeAddress64(0, 0x13, 0xa2, 0, 0x40, 0x71, 0x25, 0xd4);

                RemoteAtRequest
request = new RemoteAtRequest(addr64, "D0", new int[] {4});

                RemoteAtResponse
responseAt = (RemoteAtResponse) xbee.sendSynchronous(request, 10000);

                if
(responseAt.isOk()) {

                    log.info("successfully turned off pin 20 (D0)");

                } else {
                    throw new
RuntimeException("failed to turn on pin 20. status is " + responseAt.getStatus());
                }
                xfalse++;
                xtrue = 1;
            }

        } catch (ParseException |
InvalidArgumentException| SipException e) {

            e.printStackTrace();
        } catch (XBeeTimeoutException e)

        {

            e.printStackTrace();
        } catch (XBeeException e) {

            e.printStackTrace();
        }

    }

}
}
}

```



```

        this.processInfo("ISIK KAPANDI");
        isikAcik=false;
        String to = sender;
        String cevap = "ISIK KAPANDI";
        sipLayer.sendMessage(to, cevap);
        this.ampulBtn.setIcon(ampul);
    } catch (Throwable e)
    {
        e.printStackTrace();
        this.receivedMessages.append("Mesaji gonderirken hata olustu: " +
e.getMessage() + "\n");
    }

    }
    else {

        try
        {
            this.processInfo("ISIK ZATEN KAPALI");
            String to = sender;
            String cevap = "ISIK ZATEN KAPALI";
            sipLayer.sendMessage(to, cevap);
        } catch (Throwable e)
        {
            e.printStackTrace();
            this.receivedMessages.append("Mesaji gonderirken hata olustu: " +
e.getMessage() + "\n");
        }

    }

}

}

public void processError(String errorMessage)
{
    this.receivedMessages.append("ERROR: " +
        errorMessage + "\n");
}

public void processInfo(String infoMessage)
{
    this.receivedMessages.append(
        infoMessage + "\n");
}

@Override
public void processOptions(String sender) {

```

```

        if (!isikAcik){

            String capability="ISIK KAPALI DURUMDA";
            try {
                sipLayer.sendMessage(sender,capability);
            } catch (ParseException | InvalidArgumentException
                    | SipException e) {

                e.printStackTrace();
            }
        }

        else {

            String capability="ISIK ACIK DURUMDA";
            try {
                sipLayer.sendMessage(sender,capability);
            } catch (ParseException | InvalidArgumentException
                    | SipException e) {

                e.printStackTrace();
            }
        }
    }
    @Override
    public void processSubscribe(String sender, String event) {

        if (event.matches("ampulEvent")){

            ampulSubscribe=true;
            subscribedSender= sender;
            subscribedEvent=event;
            this.processInfo("EVENT KAYDI YAPILDI");

            try {
                this.ZNetIoSample();
            } catch (Exception e1) {

                e1.printStackTrace();
            }

            if(!isikAcik){

```

```

        try {
            sipLayer.sendNotify(sender, event, "ISIK
KAPALI DURUMDA");
        } catch (ParseException | SipException e) {
            e.printStackTrace();
        }
    } else {
        try {
            sipLayer.sendNotify(sender, event, "ISIK
ACIK DURUMDA");
        } catch (ParseException | SipException e) {
            e.printStackTrace();
        }
    }
}

else {
    ampulSubscribe=false;
}

}

}

```

Ek-D

```

#include <msp430g2231.h>

void main(void)
{
    int flag=0,i=0;

    WDTCTL = WDTPW + WDTHOLD;
    P1DIR |= 0x01;
    P1DIR |= 0x02;
    P1DIR &= ~0x04;
    P1DIR &= ~0x08;
    P1OUT = 0x00;

    while(1)
    {

        if(P1IN & 0x04)

            P1OUT |= 0x03;
        else
            P1OUT &= ~0x03;

        if(!(P1IN & 0x08))
        {

            while(!(P1IN & 0x08));
            ++i;
            flag = i%2 ;

            if(flag)
            {
                P1OUT |= 0x03;
                while(!(P1IN & 0x04));
            }
            else
            {
                P1OUT &= ~0x03;
                while(P1IN & 0x04);
            }
        }
    }
}

```

ÖZGEÇMİŞ

Gökhan Okyay 1986 yılında Balıkesir’de doğdu. 2008 yılında İstanbul Üniversitesi Elektrik Elektronik Mühendisliği Bölümü’nden mezun olduktan sonra aynı yıl yine bu bölümde yüksek lisans öğrenimine başladı. 2009 yılında Verscom Solutions bünyesinde teknik destek mühendisi olarak başladığı kariyerini halen aynı pozisyonda devam ettirmektedir.