T.C.
PAMUKKALE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

RESTFUL API İLE IOT DESTEKLİ GERÇEK ZAMANLI ENERJİ İZLEME, KONTROL VE BİLDİRİM SİSTEMİ

YÜKSEK LİSANS TEZİ

BARANSEL AYDIN

DENİZLİ, MAYIS - 2025

T.C.
PAMUKKALE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

RESTFUL API İLE IOT DESTEKLİ GERÇEK ZAMANLI ENERJİ İZLEME, KONTROL VE BİLDİRİM SİSTEMİ

YÜKSEK LİSANS TEZİ

BARANSEL AYDIN

DENİZLİ, MAYIS - 2025

Bu tezin tasarımı, hazırlanması, yürütülmesi, araştırmalarının yapılması ve bulgularının analizlerinde bilimsel etiğe ve akademik kurallara özenle riayet edildiğini; bu çalışmanın doğrudan birincil ürünü olmayan bulguların, verilerin ve materyallerin bilimsel etiğe uygun olarak kaynak gösterildiğini ve alıntı yapılan çalışmalara atfedildiğine beyan ederim.

BARANSEL AYDIN

ÖZET

RESTFUL API İLE IOT DESTEKLİ GERÇEK ZAMANLI ENERJİ İZLEME, KONTROL VE BİLDİRİM SİSTEMİ

YÜKSEK LİSANS TEZİ

BARANSEL AYDIN

PAMUKKALE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

(TEZ DANIŞMANI:PROF. DR. ABDULLAH TAHSİN TOLA)

DENİZLİ, MART - 2025

Günümüzde Nesnelerin İnterneti (Internet of Things – IoT) teknolojileri, enerji yönetimi ve akıllı otomasyon sistemleri açısından büyük önem taşımaktadır. Elektrik kesintileri ve dalgalanmalar, ev ve iş yerlerinde çeşitli sorunlara yol açarken, bu kesintilerin anlık olarak tespit edilmesi ve kullanıcıya bildirim gönderilmesi büyük bir gereklilik haline gelmiştir. Bu tez çalışmasında, *Raspberry Pi* tabanlı bir sistem geliştirilerek elektrik kesintilerinin tespit edilmesi, güç tüketiminin izlenmesi ve enerji yönetiminin sağlanması amaçlanmıştır.

Geliştirilen sistemde, UPS *Module 3S* kullanılarak kesintisiz güç sağlanmış, *DHT11* sensörü aracılığıyla sıcaklık ve nem verileri takip edilmiştir. *Shelly Plug S* akıllı prizi ile klima gibi cihazların uzaktan kontrolü gerçekleştirilmiş, çekilen akım ve güç tüketimi ölçülmüştür. Verilerin işlenmesi ve analiz edilmesi için çok aşamalı filtreleme algoritması (*Multi-Stage Filtering Algorithm - MSFA*) uygulanmış, böylece sensörlerden alınan verilerin doğruluğu artırılmıştır. *RESTful* API mimarisi kullanılarak geliştirilen sistem, *Python Flask* tabanlı bir arayüz ile veri paylaşımını sağlarken, *C# Web* API ile entegrasyon gerçekleştirilmiştir. Ölçüm verileri, *SQL Server* veri tabanında saklanarak enerji tüketimi analiz edilmiştir. Sistem, e-posta bildirim mekanizması sayesinde elektrik kesintisi ve geri gelme durumlarında kullanıcılara anlık uyarılar göndermektedir.

Enerji yönetimini optimize etmek adına akıllı priz aracılığıyla klima gibi cihazların otomatik olarak kontrol edilmesi sağlanmıştır. Test sonuçlarına göre, geliştirilen sistemin güç tüketimi izleme ve bildirim konusunda güvenilir sonuçlar verdiği gözlemlenmiştir.

ANAHTAR KELİMELER: IoT, Raspberry Pi, RESTful API, Enerji Yönetimi, Akıllı Priz, Elektrik Kesintisi İzleme, Sensör Verisi İşleme.

ABSTRACT

REAL-TIME ENERGY MONITORING, CONTROL, AND NOTIFICATION SYSTEM WITH IOT **SUPPORTED RESTFUL** API

MSC THESIS

BARANSEL AYDIN

PAMUKKALE UNIVERSITY INSTITUTE OF SCIENCE

ELECTRİCAL AND ELECTRONİCS ENGİNEERİNG

(SUPERVISOR:PROF. DR. ABDULLAH TAHSİN TOLA)

DENİZLİ, MARCH 2025

Nowadays, Internet of Things - IoT technologies play a crucial role in energy management and smart automation systems. Power outages and fluctuations cause various problems in houses and workplaces, making it essential to detect these interruptions instantly and notify users. This thesis aims to develop a Raspberry Pi-based system to detect power outages, monitor power consumption, and ensure energy management.

In the developed system, uninterrupted power was provided using the UPS Module 3S, while temperature and humidity data were monitored via the DHT11 sensor. Remote control of devices such as air conditioners was achieved using the Shelly Plug S smart plug, which also measured current draw and power consumption. A Multi-Stage Filtering Algorithm (MSFA) was applied to process and analyze the data, improving the accuracy of sensor readings. The system utilized a RESTful API architecture, where data sharing was facilitated through a Python Flask-based interface, and integration was performed with a C# Web API. Measurement data were stored in an SQL Server database, enabling energy consumption analysis.

To optimize energy management, the system automatically controlled devices like air conditioners via the smart plug. Test results demonstrated that the developed system reliably monitored power consumption and provided accurate notifications.

KEYWORDS: IoT, Raspberry Pi, RESTful API, Energy Management, Smart Plug, Power Outage Monitoring, Sensor Data Processing

İÇİNDEKİLER

Sayfa

ŞEKİL LİSTESİ

Sayfa

TABLO LİSTESİ

Sayfa

SEMBOL LİSTESİ

GHz	**:**	Gigahertz
GB	**:**	Gigabyte
GB	**:**	Gigabit
mm	**:**	milimetre
Kb/s	**:**	Kilobit/saniye
MT/s	**:**	MegaTransfers per second
Gbps	**:**	Gigabits per second
Mbps	**:**	Megabits per second
mA	**:**	Miliamper
V	**:**	Voltaj
kHz	**:**	Kilohertz
VDD	**:**	Voltage Drain Drain
kΩ	**:**	Kiloohm
DC	**:**	Direct Current
W	**:**	Watt
kWh	**:**	Kilovat-saat
Wh	**:**	Watt-saat
VCC	**:**	Voltage Common Collector

AÇIKLAMALAR LİSTESİ

ACK : Acknowledgment
ARM : Advanced RISC Machine
AV : Audio/Video
API : Application Programming Interface
BLE : Bluetooth Low Energy
BSC : Bus Serial Controller Module
CLK : Clock
CPU : Central Processing Unit
CRUD : Create, Read, Update, Delete
CSI-DSI : Camera Serial Interface - Display Serial Interface
CQRS : Command Query Responsibility Segregation
DHT11 : Digital Humidity and Temperature Sensor
DIN : Data In
DMA : Direct Memory Access
DOUT : Data Out
EF Core : Entity Framework Core
FIN : Finish
FS : Frame Sync
FTP : File Transfer Protocol
GND : Ground
GPIO : General-Purpose Input/Output
GPU : Graphics Processing Unit
gRPC : Google Remote Procedure Call
GUI : Graphical User Interface
HDMI : High-Definition Multimedia Interface
HTTP : Hypertext Transfer Protocol
HTTPS : Hypertext Transfer Protocol Secure
IBM : International Business Machines
I²C : Inter-Integrated Circuit
IMAP : Internet Message Access Protocol
IoT : Internet of Things
ISO : International Organization for Standardization
IP : Internet Protocol
JSON : JavaScript Object Notation
L1 : Locale
L2 : Timezone
L3 : Keyboard Mapping
MSFA : Multi-Stage Filtering Algorithm
MOSI : Master Out Slave In
MISO : Master In Slave Out
MVC : Model-View-Controller
OS : Operationg System
P2 : GPU Memory
P3 : Overlay File System
P4 : Fan Control
POE : *Power over Ethernet*
POP3 : Post Office Protocol 3

PWM	**:**	Pulse Width Modulation
RAM	**:**	Random Access Memory
REST	**:**	Representational State Transfer
SCLK	**:**	Serial Clock
SDRAM	**:**	Synchronous Dynamic Random-Access Memory
SMTP	**:**	Simple Mail Transfer Protocol
SOAP	**:**	Simple Object Access Protocol
SoC	**:**	System on Chip
SSH	**:**	Secure Shell
SSID	**:**	Service Set Identifier
SPI	**:**	Serial Peripheral Interface
SYN	**:**	Synchronize
SQL	**:**	Structured Query Language
TCP/IP	**:**	Transmission Control Protocol / Internet Protocol
UART	**:**	Universal Asynchronous Receiver-Transmitter
UDP	**:**	User Datagram Protocol
UI	**:**	Using Interface
UPS	**:**	Uninterruptible Power Supply
URL	**:**	Uniform Resource Locator
XML	**:**	Extensible Markup Language
W3C	**:**	World Wide Web Consortium
Wi-Fi	**:**	IEEE 802.11 Wireless Networking Standard

ÖNSÖZ

Bu süreçte, bilgi ve deneyimlerini benimle paylaşarak yol gösteren, çalışmamın her aşamasında destek ve rehberliğini esirgemeyen değerli hocam Prof. Dr. Abdullah Tahsin TOLA'ya en içten teşekkürlerimi sunarım. Sayın hocam, akademik gelişimime katkılarınız ve çalışmama sağladığınız yönlendirmeler, bu tezin başarıyla tamamlanmasında büyük bir rol oynamıştır. Bilimsel düşünme yeteneğimi geliştirmemde ve araştırma sürecinde karşılaştığım zorlukları aşmamda gösterdiğiniz anlayış ve teşvik için sonsuz minnettarım.

Ayrıca, bu süreçte desteğini hiçbir zaman esirgemeyen aileme, birlikte çalışmaktan büyük mutluluk duyduğum arkadaşlarıma ve bana ilham veren herkese teşekkür ederim. Umarım bu çalışma, ilgili alanda yeni bakış açıları kazandırarak gelecekte yapılacak araştırmalara katkı sağlar.

1. GİRİŞ

Günümüzde teknolojik gelişmeler bireylerin yaşam tarzını ve çevresiyle olan etkileşimini önemli ölçüde değiştirmektedir. Özellikle Nesnelerin İnterneti (Internet of Things- IoT) teknolojisinin yaygınlaşmasıyla birlikte ev otomasyon sistemleri ve akıllı cihazlar günlük hayatın ayrılmaz bir parçası haline gelmiştir. İnsanların konfor, güvenlik ve enerji tasarrufu beklentileri doğrultusunda geliştirilen akıllı ev sistemleri, kullanıcıların evde olmasalar bile cihazlarını uzaktan kontrol etmelerine, enerji tüketimini izlemelerine ve otomatik ayarlamalar yapmalarına olanak tanımaktadır.

Elektrikli cihazların kullanım oranı son yıllarda hızla artmıştır (Raporlar ve Serisi, 2020). Günlük hayatta kullandığımız buzdolabı, televizyon, bilgisayar, klima, su ısıtıcıları ve diğer elektrikli ev aletleri, toplam enerji tüketiminin büyük bir kısmını oluşturmaktadır. Ancak, bekleme konumunda çalışan cihazların gereksiz enerji tüketimi, elektrik kesintilerinin cihazlara verdiği zararlar ve kullanıcı farkındalığının düşük olması gibi faktörler, enerji yönetimi konusunda çeşitli sorunları da beraberinde getirmektedir. Bu noktada akıllı prizler ve IoT tabanlı enerji yönetim sistemleri enerji tüketimini optimize etmek, elektrik kesintisi durumlarında kullanıcıları bilgilendirmek ve uzaktan kontrol mekanizmaları sunmak açısından önemli çözümler arasında yer alabilir. Akıllı prizler, bağlı oldukları cihazların enerji tüketimini analiz edebilmekte, kullanıcıya anlık bildirimler gönderebilmekte ve otomatik açma-kapama fonksiyonlarıyla gereksiz enerji kaybının önüne geçebilir.

Elektrik kesintileri, beklenmedik güç dalgalanmaları ve şebeke arızaları nedeniyle birçok elektronik cihazın zarar görmesine veya çalışamaz hale gelmesine neden olabilmektedir. Özellikle uzun süreli kesintilerde veya ani voltaj değişimlerinde şalterin atması sonucu, evinden uzakta olan kişiler (örneğin tatile çıkan, seyahatte olan ya da işte bulunan bireyler) döndüklerinde buzdolabındaki tüm gıdaların bozulması gibi ciddi sorunlarla karşılaşabilmektedir. Bu tür durumlara karşı geliştirilen Kesintisiz Güç Kaynağı (UPS) sistemi, modeme enerji sağlayarak IoT tabanlı bildirim sisteminin çalışmasını sürdürür. Böylece, elektrik kesintisi algılandığında kullanıcıya e-posta

yoluyla bildirim gönderilir. Elektrik tekrar geldiğinde ise sistem, kullanıcıyı yeniden bilgilendirerek sürecin takibini kolaylaştırır.

Bu tez çalışmasında Raspberry Pi tabanlı bir akıllı ev otomasyon sistemi geliştirilerek, elektrik kesintisi tespiti ve anlık bildirim gönderme mekanizması oluşturulmuştur. Ayrıca, akıllı priz entegrasyonu ile kullanıcıların uzaktan cihaz kontrolü sağlaması ve sıcaklık ve nem sensörü ile ortam değişkenleri izlenerek tasarruf yapabilmesi amaçlanmaktadır. UPS voltaj dalgalanmalarını algılayarak batarya-adaptör modları arasındaki geçişleri yönetme sistemi geliştirilmiştir. Geliştirilen sistem, UPS voltajını ve akım dalgalanmalarını analiz ederek, bataryadan çalışmaya geçtiğinde kullanıcıya e-posta bildirimi göndermektedir. Örneğin, elektrik kesildiğinde "Elektrik Kesildi, Batarya Moduna Geçildi" e-postası gönderilirken, elektrik geri geldiğinde "Elektrik Geri Geldi, Adaptör Moduna Geçildi" bildirimi yapılmaktadır.

Tez kapsamında geliştirilen sistem mevcut enerji yönetimi çözümlerine kıyasla IoT tabanlı ve kullanıcı ihtiyaçlarına göre özelleştirilebilen bir yaklaşım sunmaktadır.

Sistem, enerji tüketim analizlerini veri tabanı üzerinden kaydedip işleyerek geçmiş kayıtlar üzerinden analiz yapmaya ve enerji yönetim stratejileri oluşturmayı sağlamaktadır. Sistemin modüler yapısı sayesinde ölçeklenebilir bir çözüm sunulmaktadır. Uzaktan erişim ve bildirim hizmetleri için ticari çözümler (örneğin, hosting, domain veya sms servisleri) gerekebilir, ancak sistem açık kaynak teknolojilerle geliştirildiği için yapılandırma konusunda esneklik sunmaktadır. Bu çalışma, enerji yönetimi, akıllı otomasyon ve uzaktan kontrol konularında uygulanabilecek yenilikçi bir IoT çözümü sunmaktadır.

1.1 Literatür Taraması

Enerji yönetimi ve akıllı ev sistemleri, günümüzde enerji tasarrufu sağlamak ve sürdürülebilir enerji kullanımı sağlamak amacıyla hızla gelişmektedir. Akıllı prizler, IoT tabanlı enerji ölçüm sistemleri ve akıllı ev otomasyonu, bu alandaki temel araştırma konularından bazılarıdır. Yapılan çalışmalar, IoT teknolojilerinin enerji

tüketimini izleme, yönetme ve optimize etme konusunda önemli katkılar sağladığını göstermektedir.

Akıllı prizler, geleneksel elektrik prizlerine entegre edilen ve enerji tüketimini gerçek zamanlı olarak izleme ve yönetme yeteneğine sahip cihazlardır. Yapılan araştırmalara göre, akıllı prizlerin kullanımı evlerde enerji tüketimini %5 ila %15 oranında azaltabilmektedir (Anilkumar, George, Aswin, Devika ve Jyothy, 2023). Akıllı prizler, Wi-Fi, *Zigbee* ve *Bluetooth* gibi kablosuz iletişim teknolojileri kullanarak kullanıcıların uzaktan cihazlarını açıp kapatmasına ve enerji tüketimini optimize etmesine olanak tanımaktadır.

Çalışmada geliştirilen akıllı prizler, *Zigbee* haberleşme protokolü kullanarak gerçek zamanlı enerji izleme ve uzaktan kontrol imkânı sunmaktadır. Bu sistemlerin, geleneksel prizlere kıyasla enerji verimliliğini artırdığı ve gereksiz tüketimin önüne geçtiği belirtilmiştir (Giray Eşref Kıral, 2014).

Günümüzde akıllı ev otomasyon sistemleri, IoT teknolojileriyle entegre edilerek enerji yönetimi, güvenlik ve konfor sağlamaktadır. *Prathyusha* ve *Bhowmik* 2023 yılında IoT tabanlı bir akıllı ev sisteminin enerji yönetimi ile nasıl optimize edilebileceğini incelemişlerdir. Çalışmada, IoT cihazlarının enerji tüketimi analiz edilerek, kullanıcıların enerji tasarrufu sağlamasına yardımcı olacak bir sistem tasarlanmıştır. Sonuçlar, akıllı ev sistemlerinin enerji verimliliğini artırarak uzun vadede maliyetleri düşürebileceğini göstermektedir (Prathyusha ve Bhowmik, 2023).

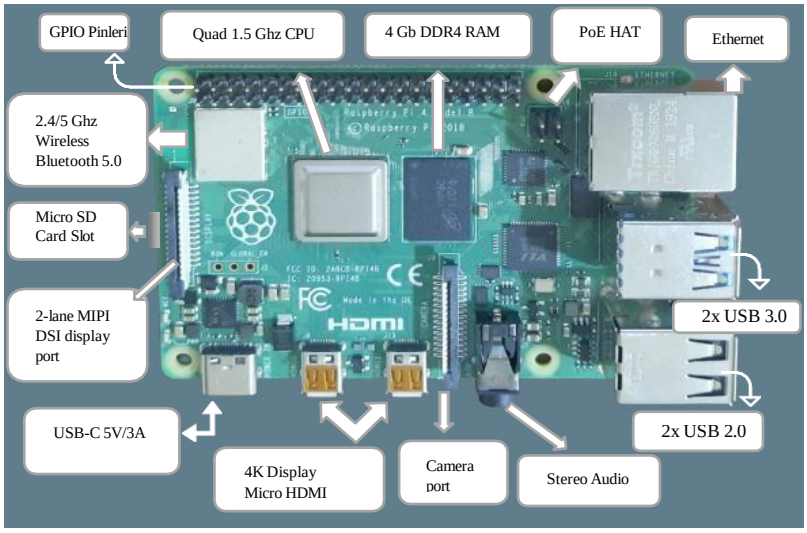
Gelişmekte olan ülkelerde enerji kesintileri, evlerde ve işyerlerinde ciddi sorunlara yol açmaktadır. Geleneksel UPS sistemleri, enerji kesintilerinde güç sağlamaya yardımcı olmakla birlikte, yüksek maliyetli ve verimsiz çözümler olabilir. Aftab ve arkadaşları 2021 yılında yapılan bir çalışmada IoT destekli akıllı enerji ölçüm sistemleri ve Soft-UPS konsepti incelenmiştir. Soft-UPS geleneksel UPS sistemlerine kıyasla daha verimli bir enerji yönetimi sağlamakta ve enerji kesintisi durumlarında dinamik bir yük yönetimi sunmaktadır. Bu tür çözümler, özellikle ani elektrik kesintilerinde güç yönetimini optimize ederek kullanıcı deneyimini iyileştirmektedir (Aftab ve diğerleri, 2021).

2. RASPBERRY Pİ 4 MODEL B

Bu bölümde, *Raspberry Pi* 4'ün donanım ve yazılım bileşenleri detaylı olarak incelenmiştir. *GPIO* pin yapısı ve çevre birimleri ile entegrasyonu, haberleşme protokolleri (I^2C, SPI, UART vb.) ve veri iletişimi, IoT sistemlerindeki rolü ve donanım-yazılım entegrasyonu, *RESTful* API ile etkileşimi ve sensör yönetimi, cihazın proje içindeki işlevselliği ele alınmıştır.

2.1 Raspberry Pi 4 Model B'nin Donanımsal İncelemesi

Raspberry Pi 4, gelişmiş donanım özellikleri sayesinde IoT projelerinde robotik sistemler ve gömülü yazılımlar için yaygın olarak kullanımı tercih edilen bir mini bilgisayardır.



Şekil 2.1: Raspberry Pi 4 Model B iç yapısı

Şekil 2.1'de gösterilen Raspberry Pi 4'ün iç yapısı:

- İşlemci: *Broadcom* BCM2711 çipi üzerinde çalışan dört çekirdekli 1.5 GHz *ARM Cortex-A72* CPU bulunur.
- Bellek (RAM): 4 GB *LPDDR4-3200* RAM kapasitesi ile çoklu görev işlemleri ve verinin yoğun olarak kullanıldığı uygulamalarda iyi performans sunar.
- Grafik İşlem Birimi (GPU): *Broadcom VideoCore VI* GPU, *OpenGL ES 3.x* desteği sunarak gelişmiş grafik işlemleri için uygundur.
- Depolama: İşletim sistemi ve veri saklama için *MicroSD* kart yuvası bulunur.

- USB Bağlantıları: 2 adet USB 3.0 ve 2 adet USB 2.0 portu ile veri aktarımı sağlar.
- Video Çıkışı: 2 adet *Micro* HDMI çıkışı, çift 4K ekran desteği sunar.
- Ağ Bağlantısı: 1 Gb *Ethernet* portu ve dahili 802.11ac Wi-Fi (2.4 GHz-5 GHz) kablosuz bağlantı desteği mevcuttur.
- *Bluetooth*: *Bluetooth* 5.0 (BLE) desteği, kablosuz iletişimi sağlar.
- GPIO Pinleri: 40 pinli GPIO (Genel Amaçlı Giriş/Çıkış) başlığı, sensörler ve harici modüller ile kolay uyum sağlar.
- Diğer Bağlantılar: CSI-DSI kamera bağlantısı, 3.5 mm analog AV Jak'ı ve PoE (*Power over Ethernet*) desteği bulunmaktadır.

Yüksek işlem gücü, geniş bağlantı modülleri, düşük güç tüketimi, programlama dilleriyle uyumluluğu sayesinde IoT ekosisteminde, gömülü sistemlerde ve sensörler yardımıyla veri toplama projelerinde çokça kullanılır.

2.1.1 Broadcom BCM2711 İşlemcisi

BCM2711, dört çekirdekli 64-*bit* ARM *Cortex-A72* işlemciye sahip bir sistem çipidir. 1.5 GHz hızında çalışır. Grafik birimi olarak *Broadcom VideoCore* VI GPU'yu kullanır ve *OpenGL ES* 3.1 desteği ile 4K çözünürlükte video çıkışı sağlar. 2 GB, 4 GB ve 8 GB LPDDR4 *RAM* seçenekleri ile gelir.

Bağlantı özellikleri ise *Gigabit Ethernet*, 2.4 *GHz* ve 5 GHz bantlarında çalışan dahili *802.11ac* Wi-Fi, *Bluetooth 5.0* ve *BLE* desteği bulunur. USB *3.0* ve USB *2.0* bağlantı noktalarına sahiptir. *MicroSD* kart yuvası, *CSI* kamera bağlantısı, *DSI* ekran bağlantısı, 3.5 mm analog *AV* Jak'ı (ses ve kompozit video çıkışı destekler) ve PoE desteği için ayrı olarak satılan bir PoE HAT bağlantısı bulunur. *Raspberry Pi* 4 Model B bu özellikleri ile gelişmiş bir mini bilgisayar olarak geniş kullanım alanlarına sahiptir.

BCM2711 çeşitli çevresel birimleri içeren bir sistemdir. Bu çevresel birimler şunlardır: *BSC* denetleyicisi I²C uyumlu haberleşme birimi, *DMA* kontrolörü, zamanlayıcılar, kesme kontrolörü, Genel Amaçlı G/Ç (GPIO), USB, PCM/I2S, I²C ana cihazları, SPI ana cihazları, PWM ve UART'lar.

BSC*, Philips*® I²C protokolü v2.1 ile uyumlu ana kontrolördür. Hızlı modda (400 Kb/s) çalışır ve yalnızca tek *controller* (ana cihaz) olarak çalışabilir. 7-*bit* ve 10-*bit* adresleme desteği sunar. BCM2711'de sekiz adet BSC modülü bulunmaktadır, sadece BSC2 ve BSC7 HDMI arayüzleri için ayrılmıştır.

DMA, işlemci müdahalesi olmadan veri transferlerini yönetir. DMA kontrolörü 16 kanala sahiptir. 4 tanesi DMA *Lite* düşük performanslı, 4 tanesi DMA4 yüksek performanslı ve geniş adresleme olarak yapılandırılmıştır. Kullanılmayan DMA kanalları, güç tasarrufu sağlamak için tamamen devre dışı bırakılabilir. DMA kontrolörü işlemci yükünü azaltarak verimli veri transferini gerçekleştirir. Bağımsız DNA kanalları çevresel birimlerle doğrudan veri alışverişi yapabilir.

Kesme kontrolörleri, işlemcinin farklı donanım birimlerinden gelen olayları yönetmesini ve işlemciye bildirmesini sağlayan donanımdır. Kesme kontrolörünün amacı, farklı donanım birimlerinden gelen kesme sinyallerini alır ve uygun bir şekilde CPU'ya yönlendirir. Birçok donanım birimlerini içeren sistemlerde, aynı anda birçok kesme oluşabilir. USB cihazı bağlandığında veya veri aktardığında, GPIO pinine sinyal geldiğinde, zamanlayıcı (*Timer*) sinyali oluşturduğunda, *Ethernet* üzerinden veri alışverişi gerçekleştiğinde, kamera modülü yeni bir görüntü yakaladığında gibi çeşitli donanım birimlerine örnek verilebilir. Eğer kesme kontrolcüsü olmazsa, işlemciye yük binecektir. İşlemci tüm donanımları kontrol etmek zorunda kalacaktır bu da büyük bir performans kaybına neden olur. Kesme kontrolcüsü, bu işlemleri optimize eder.

PCM (*Pulse Code Modulation*) / I2S (*Inter-IC Sound*), yüksek kaliteli ses sinyallerini dijital olarak iletmek için kullanılan bir seri haberleşme protokolüdür. Genellikle ses sistemlerinde ve telefon uygulamalarında karşımıza çıkar. 4 ana sinyal hat bulunur. PCM_CLK, bit saat sinyali, veri aktarım hızını belirler. PCM_FS, veri bloğu senkronizasyon sinyali yani ses verilerini belirli zaman aralıklarında veri setleri halinde iletir. Bu sayede ses verisinin hangi kanala ait olduğu belirlenebilir, sağ kanal ya da sol kanal. Sinyal düşük (*low*) olduğunda sol kanal verisi, yüksek (*high*) olduğunda sağ kanal verisi iletilir. PCM_DIN, mikrofon gibi çevresel birimden gelen veri girişidir, PCM_DOUT, hoparlöre giden ses çıkış verisidir.

PWM, bir sinyalin darbe genişliğini değiştirerek (*duty cycle*) analog çıkış oluşturmak için kullanılan bir yöntemdir. Bir döngü içinde yüksek ve düşük durumlar

arasında ne kadar zaman geçirildiği ayarlanarak farklı voltaj değerleri simüle edilir. Örneğin, bir LED'in parlaklığını ayarlamak için PWM yöntemi kullanılabilir. Eğer PWM sinyali yarı yarıya yüksek ve düşük durumda ise LED maksimum parlaklığının yarısında yanacaktır. PWM'in 2 özelliği vardır. Frekans ve *duty cycle*. Frekans, sinyalin ne kadar hızlı yanıp söndüğünü belirtirken, *duty cycle* sinyalin yüksek durumunda ne kadar süre kaldığını belirtir. Mekanik bir sürücü ya da motor hız kontrolü için de sıklıkla kullanılır (Broadcom Europe Ltd. and Raspberry Pi Ltd., 2022).

2.1.2 LPDDR4-3200 SDRAM Bellek

Raspberry Pi 4 Model B'de kullanılan LPDDR4-3200 SDRAM, düşük güç tüketimi ve yüksek bant genişliği sunarak sistemin performansını artırır. BCM2711, 16 GB'a kadar SDRAM adres alanını destekleyen bir bellek yönetimi sistemine sahiptir, ancak gerçekte kullanılan RAM miktarı cihazın modeline göre 2 GB, 4 GB veya 8 GB olarak belirlenmiştir.

Bellek erişimi ve adresleme ARM çekirdekleri bellek erişimi sırasında *Low Peripheral* (Düşük Çevresel) modunu kullanarak belirli bir adres aralığında çalışabilir. DMA (*Direct Memory Access*) motorları, RAM erişimi için özel olarak *Legacy Controller* adreslerini kullanır. RAM erişimi için tahsis edilen 1 GB'lık alan, ihtiyaç halinde 16 GB'lık *SDRAM* adres alanında kaydırılabilir. DMA *Lite* motorları (*DMA7*-10), SDRAM içerisinde 1 GB'lık ön belleksiz bir bellek aralığını taşımak için *PAGE* ve *PAGELITE* bitlerini kullanır.

VideoCore tarafından yapılan bellek erişimleri, 35-*bit* adresleme ile dönüştürülerek yürütülür ve *VideoCore,* bellek yönetimini *ARM* çekirdeklerinden bağımsız şekilde optimize edebilir.

Raspberry Pi 4'ün LPDDR4-3200 SDRAM'ı, 3200 MT/s (*megatransfer*/*saniye*) hızında çalışarak, geniş bant genişliği ve düşük güç tüketimi sağlar (Broadcom Europe Ltd. and Raspberry Pi Ltd., 2022).

2.1.3 MicroSD Depolama

Raspberry Pi 4'te *microSD* kart yuvası, cihazın temel depolama birimlerinden biridir. Bu yuva *Raspberry Pi*'ye işletim sistemi yüklenmesi ve veri depolanması için gereklidir. *MicroSD* kartlar, fiziksel olarak küçük olmalarına rağmen, günümüz teknolojisi sayesinde büyük kapasitelerde ve yüksek hızlarda üretilmektedir. Cihazın yan tarafına konumlandırılan *microSD* kart yuvası, kullanıcıların kartları hızlı ve pratik bir şekilde değiştirmesini sağlar.

Raspberry Pi, işletim sistemini harici bir *microSD* karttan başlatır, dolayısıyla sistemin çalışabilmesi için bu karta bir işletim sisteminin yazılması gerekir. Geliştiriciler depolama kapasitesini artırmak ya da farklı projeler için ayrı sistemler kullanmak amacıyla birden fazla *microSD* karttan yararlanabilirler. Örneğin, bir kartta *Linux* tabanlı bir işletim sistemi çalıştırılırken, farklı bir kartta gömülü sistemler için optimize edilmiş bir sürüm bulunabilir (Broadcom Europe Ltd. and Raspberry Pi Ltd., 2022).

MicroSD kart yuvası, *Secure Digital* (SD) kart ailesine aittir. Genellikle mobil cihazlar, dijital kameralar, tabletler, oyun konsolları ve diğer taşınabilir cihazlar için harici bir depolama ortamı olarak kullanılır.

2.1.4 USB 2.0 ve USB 3.0 Bağlantı Noktaları

Bugün neredeyse tüm cihazlarda USB bağlantısını görmek mümkün. USB hem veri hem de güç aktarımı yapabilen standart bir bağlantı noktasıdır. Yani sadece veri taşımakla kalmaz, aynı zamanda cihazları şarj etmek için de kullanılır. Telefonlardan dizüstü bilgisayarlara, klavyelerden harici disk sürücülerine kadar her yerde USB bağlantısını görmek mümkün. USB 3.0, *SuperSpeed* USB olarak da bilinir ve 5 Gbps'ye kadar veri aktarım hızı sağlar.

Genel olarak baktığımızda, *Raspberry Pi* 4 Model B'nin USB bağlantıları, cihazın işlevselliğini önemli ölçüde artırıyor. USB 3.0 sayesinde yüksek hızlarda veri aktarımı sağlanırken, USB 2.0 bağlantıları da temel çevre birimleri için kullanılabiliyor.

2.1.5 Ethernet Bağlantısı

Raspberry Pi 4 Model B, yüksek hızlı ve güvenilir ağ bağlantıları için tam gigabit *Ethernet* desteği sunmaktadır. Önceki modellerde *Ethernet* bağlantısı, dahili USB 2.0 veri yolu üzerinden sağlandığından, maksimum veri aktarım hızı yaklaşık 300 Mbps ile sınırlıydı. Ancak *Raspberry Pi* 4 Model B'de *Ethernet* portu doğrudan SoC'ye bağlanarak bu sınırlama ortadan kaldırılmış ve tam 1 Gbps (1000 Mbps) hızında veri aktarımı mümkün hale getirilmiştir (Raspberry Pi 4 Model B Specifications, 2025).

Ek olarak, *Raspberry Pi* 4 Model B'nin *Ethernet* portu, IEEE 802.3af standardına uygun *Power over Ethernet* (PoE) desteği sunmaktadır. Bu özellik, uygun bir PoE HAT (*Hardware Attached on Top*) eklentisi kullanılarak, cihazın *Ethernet* kablosu üzerinden güç almasını sağlar. Bu sayede, özellikle uzak veya erişimi zor noktalarda ek bir güç kaynağına ihtiyaç duymadan *Raspberry Pi*'yi çalıştırmak mümkün olur.

2.1.6 Kablosuz Bağlantı: Wi-Fi ve Bluetooth

Raspberry Pi 4 Model B, yüksek hızlı kablosuz bağlantı ihtiyaçları için çift bantlı 802.11ac Wi-Fi desteği sunmaktadır. Bu özellik, hem 2.4 GHz hem de 5 GHz frekanslarında kablosuz ağlara bağlanabilmeyi sağlar, böylece daha hızlı bir internet deneyimi sağlar (Raspberry Pi 4 Model B Specifications, 2025).

Raspberry Pi 4 Model B, *Bluetooth* 5.0 ve *Bluetooth Low Energy* (BLE) desteğiyle birlikte gelir. Bu teknoloji sayesinde düşük enerji tüketimiyle kablosuz cihazlarla iletişim kurabilir ve veri aktarımı gerçekleştirilebilir.

Raspberry Pi 3 modeli *Bluetooth* 4.1, *Raspberry Pi* 3 B+ modeli *Bluetooth* 4.2, *Raspberry Pi* 4 modeli ise *Bluetooth* 5.0 teknolojisini kullanmaktadır. Sürümler arasındaki farklar performans açısından önemli bir yer teşkil etmektedir. *Bluetooth* 4.1 24 Mbps hız, 100-300 metre mesafe, 2.4-2.485 GHz bant genişliği sağlarken, *Bluetooth* 5.0 48 Mbps hız, 985 metreye kadar mesafe, gelişmiş veri aktarımı ve daha düşük enerji tüketimi sunar (Raspberry Pi Foundation, 2025). *Raspberry Pi*

cihazlarında yerleşik *Bluetooth* modülü bulunur, kablosuz iletişim için rahatlıkla bu modül kullanılabilir. Bu *Bluetooth* modülü varsayılan olarak pasif durumda sunulur ve kullanılabilmesi için konfigüre edilerek aktif hale getirilmesi gerekmektedir.

Bluetooth teknolojisi, IoT sistemlerinde de önemli bir rol oynamaktadır. Örneğin, bir *Raspberry Pi* üzerinde bulunan *Bluetooth* modülü kullanılarak, DHT11 sıcaklık ve nem sensörü aracılığıyla ortam sıcaklık verileri ölçülerek *Bluetooth* üzerinden diğer cihazlara veriler aktarılabilir. Bu tür bir uygulamada, ilgili kütüphanelerin yüklenmesi ve *Bluetooth* protokollerinin kullanılması gerekebilir.

2.1.7 GPIO Pinleri (Genel Amaçlı Giriş/Çıkış)

Raspbbery Pi 4 Model B'nin GPIO (Genel Amaçlı Giriş/Çıkış) pinleri, harici cihazlarla etkileşim kurmak için kullanılan dijital sinyal bağlantılarıdır. Toplamda 3.3V, 5V ve GND (toprak) pinleri de dahil olmak üzere 40 pine sahiptir.

GPIO pinlerinin işlevi giriş modu ve çıkış modu olmak üzere ikiye ayrılır. Giriş modu (*Input*) sensörlerden veya cihazlardan gelen sinyalleri okumak için kullanılır.

Örneğin, bir butonun basılıp basılmadığını tespit etmek için giriş modu kullanılabilir. Çıkış modu (*Output*) cihazlara sinyal göndermek için kullanılır. Örneğin, bir LED'i yakmak veya bir motoru çalıştırmak için çıkış modu kullanılır.

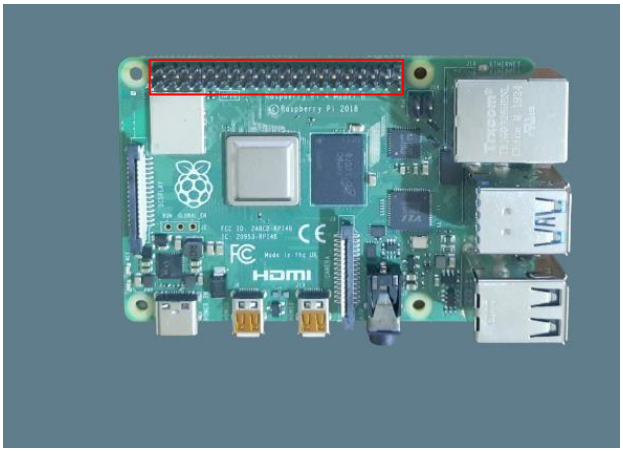
GPIO pinleri 3.3V seviyesinde çalışır ve maksimum 16 mA akım sağlayabilir. Toplam GPIO pinleri üzerinden çekilebilecek maksimum akım ise 50 mA ile sınırlıdır. GPIO pinleri, genellikle 0 ve 1 olarak ifade edilen dijital sinyalleri destekler. 0 düşük (*low*) voltaj ve 1 yüksek (*high*) voltaj anlamına gelir.

Bu model çeşitli programlama dilleriyle geliştirilebilir yapıya sahiptir. Özellikle *Python* dili için geliştirilen *RPi.GPIO, gpiozero* ve *pigpio* gibi kütüphaneler, pinlerin kolaylıkla yönetilmesini sağlar. Bu kütüphaneler, pinlerin giriş veya çıkış olarak ayarlanması ve veri okunup yazılması gibi işlemleri geliştiriciler için basitleştirir.

GPIO pinleri 3.3V ile çalıştığından, 5V'luk sinyaller doğrudan bağlandığında *Raspberry Pi*'ye zarar verebilir. Bu nedenle, voltaj seviyelerine dikkat etmek önemlidir. Doğru kullanım sağlandığında birçok farklı elektrik projesi ve IoT uygulamaları teknoloji dünyasına katkı sağlayabilir. Ancak yanlış kullanımı, yanlış bağlantılar veya voltaj seviyeleri çevre birimlerine veya *Raspberry Pi*'ye zarar verebilir. Bu nedenle, kullanmadan önce GPIO'yu bağlamak ve programlamak için gerekli bilgi ve becerilere sahip olmak önemlidir.

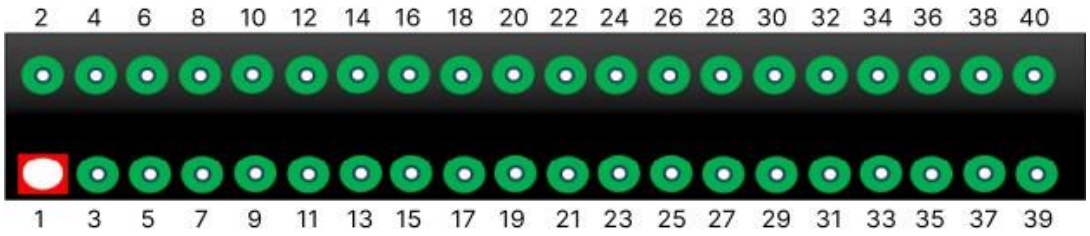
2.1.7.1 GPIO Pin Yapısı ve Temel Özellikler

Şekil 2.2' de *Raspberry Pi* 4'ün kart üzerinde GPIO pinlerinin yer aldığı alan gösterilmiştir.



Şekil 2.2: Raspberry Pi 4 Model B GPIO pinleri

Raspberry Pi 4 Model B üzerinde bulunan GPIO pinleri, 26'sı genel amaçlı giriş/çıkış için, 8 adet topraklama (GND), 2 adet 3.3V ve 2 adet 5V pin ve 2 adet donanımsal kontrol için ayrılmış olup, geliştiriciler tarafından doğrudan kullanılamayan pin bulunur.



Şekil 2.3: Raspberry Pi 4 GPIO pinleri

Şekil 2.3'te gösterilen Raspberry Pi 4 modeline ait GPIO pinlerinin işlevleri aşağıda açıklanmıştır.

1. 3.3V: Bu pin, 3.3 volt çıkış voltajı sağlar. Bu pin, düşük güçlü cihazları beslemek için kullanılabilir.
2. 5V: Bu pin, 5 volt çıkış voltajı sağlar. Bu pin, yüksek güçlü cihazları beslemek için kullanılabilir.
3. GPIO2 (I²C1 SDA): Bu pin, seri veri iletişimi için kullanılır. I²C cihazlarına bağlanmak için kullanılabilir.
4. 5V: Bu pin, 5 volt çıkış voltajı sağlar.
5. GPIO3 (I²C1 SCL): Bu pin, seri veri iletişimi için kullanılır. I²C cihazlarına bağlanmak için kullanılabilir.
6. GND: Bu pin, topraklama (GND) pini olarak kullanılır.
7. GPIO4: Bu pin, genel amaçlı bir I/O pinidir.
8. GPIO14 (TXD0): Bu pin, UART veri iletişimi için kullanılır.
9. GND: Bu pin, topraklama (GND) pini olarak kullanılır.
10. GPIO15 (RXD0): Bu pin, UART veri iletişimi için kullanılır.
11. GPIO17: Bu pin, genel amaçlı bir I/O pinidir.
12. GPIO18: Bu pin, genel amaçlı bir I/O pinidir.
13. GPIO27: Bu pin, genel amaçlı bir I/O pinidir.
14. GND: Bu pin, topraklama (GND) pini olarak kullanılır.
15. GPIO22: Bu pin, genel amaçlı bir I/O pinidir.
16. GPIO23: Bu pin, genel amaçlı bir I/O pinidir.
17. 3.3V: Bu pin, 3.3 volt çıkış voltajı sağlar.
18. GPIO24: Bu pin, genel amaçlı bir I/O pinidir.
19. GPIO10 (SPI0 MOSI-SDA): Bu pin, SPI veri iletişimi için kullanılır.
20. GND: Bu pin, topraklama (GND) pini olarak kullanılır.
21. GPIO9 (SPI0 MISO-SDO): Bu pin, SPI veri iletişimi için kullanılır.
22. GPIO25: Bu pin, genel amaçlı bir I/O pinidir.
23. GPIO11 (SPI0 SCLK): Bu pin, SPI veri iletişimi için kullanılır.
24. GPIO8 (SPI0(CS0=CE0)): Bu pin, SPI cihazlarına seçim yapmak için kullanılır.
25. GND: Bu pin, topraklama (GND) pini olarak kullanılır.
26. GPIO7 (SPI0 CS1): Bu pin, SPI cihazlarına seçim yapmak için kullanılır.

27. GPIO0: Bu pin, genel amaçlı bir I/O pinidir.
28. GPIO1: Bu pin, genel amaçlı bir I/O pinidir.
29. GPIO5: Bu pin, genel amaçlı bir I/O pinidir.
30. GND: Bu pin, topraklama (GND) pini olarak kullanılır.
31. GPIO6: Bu pin, genel amaçlı bir I/O pinidir.
32. GPIO12: Bu pin, genel amaçlı bir I/O pinidir.
33. GPIO13: Bu pin, genel amaçlı bir I/O pinidir.
34. GND: Bu pin, topraklama (GND) pini olarak kullanılır.
35. GPIO19: Bu pin, genel amaçlı bir I/O pinidir.
36. GPIO16 (SPI0 CS0): Bu pin, genel amaçlı bir I/O pinidir.
37. GPIO26: Bu pin, genel amaçlı bir I/O pinidir.
38. GPIO20 (SPI0 CS0): Bu pin, genel amaçlı bir I/O pinidir.
39. GND: Bu pin, topraklama (GND) pini olarak kullanılır.
40. GPIO21 (SPI0 SCLK): Bu pin, genel amaçlı bir I/O pinidir.

Raspberry Pi 4 Model B için 1,8V ile 3V arasındaki giriş yüksek voltaj olarak kabul edilir diğer yandan 1,8V'tan küçük değerler ise düşük voltaj olarak kabul edilir. Dikkat edilmesi gereken husus ise güç kaynağı voltajını 3V'un altında tutmaktır. *Raspberry Pi*'nin GPIO2 ve GPIO4 pinleri doğrudan 5V sağlar, bu pinler sadece güç sağlamak içindir, lojik sinyal üretmezler. Lojik seviyede çalışması 3.3V pinleri ile sağlanır. Bir GPIO pininin *HIGH* çıktısı 3.3V'u gösterir.

2.1.7.2 GPIO Pinleri Üzerinden Haberleşme Protokolleri

Raspberry Pi 4 Model B'nin GPIO pinleri, sadece temel dijital giriş ve çıkış işlemleri için değil, aynı zamanda çeşitli haberleşme protokollerinin uygulanması için de kullanılmaktadır. Bu protokoller, *Raspberry Pi*'nin harici cihazlar veya sensörler ile veri alışverişinde bulunmasını sağlar. Donanım seviyesinde en yaygın kullanılan haberleşme protokolleri arasında I²C (Inter-Integrated Circuit), SPI (*Serial Peripheral Interface*) ve UART (*Universal Asynchronous Receiver Transmitter*) protokolleri yer almaktadır.

2.1.7.2.1 I²C Protokolü

I²C protokolü *Philips Semiconductor* yeni adıyla NXP *Semiconductors* tarafından 1980'li yılların başında geliştirilmiş seri veri iletişim protokolüdür (NXP Semiconductors, 2021).

I²C protokolü, 100 kHz standart modda birkaç metre mesafeye kadar güvenli iletişim sağlarken, 400 kHz fast mode'da veri hızı artsa da iletişim mesafesi azalmaktadır (Leens, 2009). Mikrodenetleyiciler ile diğer IC'ler örneğin, sensörler *EEPROM* arsında veri alış-verişi yapmak için kullanılır.

I²C'nin başlıca özellikleri:

- Çift yönlü (*bidirectional*), hem veri gönderme hem veri alma yeteneği vardır.
- Tek bir I²C hattı üzerinden birden fazla cihazla iletişim kurulabilir.
- *Controller-Peripheral* yapısı, mikrodenetleyiciler *(controller)* diğer bileşenler örneğin sensörler *(peripheral)* olarak çalışır.
- Veriyi seri olarak bit bit gönderir.
- I²C, SDA (Serial Data – Seri Veri Hattı) ve SCL (Seri Saat Hattı) olmak üzere 2 hat kullanır. SCL, veri iletişiminin zamanlamasını kontrol eder, SDA üzerinden veri aktarılır.
- Cihazlara iletişim sırasında onları tanımlamak için adres atanır. Bu adres genellikle 7 *bittir*, bazı durumlarda 10 *bit* de olabilir.
- Senkron iletişim söz konusudur. Saat hattı (SCL) ile senkronize edilen veri hattı (SDA) aracılıyla veri transferi gerçekleşir.
- Saat sinyali bir kare dalga sinyali üretir. Her yüksekten düşüğe geçişte 1 *bit* aktarım sağlanır. Saniyede 100.000 *bit* hızında veri aktarılır.
- *Transmitter*, bir bilgi veya sinyali bir kaynaktan bir hedefe gönderen cihazdır (NXP Semiconductors, 2021). Örneğin, bir Wi-Fi yönlendirici (*router*) internetten aldığı dijital verileri radyo dalgalarına dönüştürerek çevresindeki Wi-Fi destekli cihazlara bilgisayar, telefon, tablet gibi gönderir. Bu cihazlar, veriyi alarak kullanıcının internete erişmesini sağlar.

Raspberry Pi 4 sahip olduğu GPIO pinleri ve desteklediği haberleşme protokolleri (I²C, SPI, *UART*) sayesinde, harici donanımlarla doğrudan iletişim kurabilen bir transmitter rolü üstlenebilir. Örneğin, bir sıcaklık sensöründen veri almak

isteyen *Raspberry Pi* 4, önce bu sensöre bir okuma komutu göndermek zorundadır. Bu komutu gönderdiği anda, *Raspberry Pi* 4 *transmitter* görevi yapmış olur. Sensörden gelen veriyi aldığında ise *receiver* (alıcı) rolünü üstlenir. I²C protokolünde *Raspberry Pi* 4 *controller* cihaz olarak çalışır ve haberleşme veri yolu üzerindeki *peripheral* (bağımlı) cihazlara veri gönderme görevini üstlenir. Bu süreçte *Raspberry Pi* 4 veri akışını başlattığı için *transmitter* olarak tanımlanır (NXP Semiconductors, 2021). *Raspberry Pi* 4, harici cihazlara komut gönderen ya da veri aktaran her durumda *transmitter* görevi üstlenir. Bu rol, sadece kablolu haberleşme protokollerinde değil, aynı zamanda Wi-Fi veya *Bluetooth* üzerinden yapılan kablosuz iletişimde de geçerlidir.

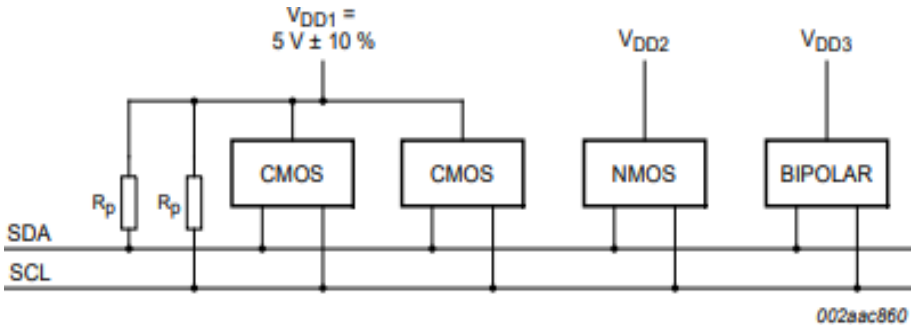
Receiver, bir haberleşme hattı (*bus*) üzerinden gelen veri veya sinyali alan ve bu veriyi işleyen cihazdır. Özellikle düşük hızlı haberleşme protokolleri olan I²C ve SPI gibi sistemlerde, *receiver* rolündeki cihaz, veriyolu üzerinden iletilen veriyi alarak işler ve gerektiğinde yanıtlar. *Transmitter* tarafından kodlanan bilginin alıcı tarafından doğru bir şekilde çözülmesini sağlar.

Raspberry Pi 4, GPIO pinleri ve haberleşme protokolleri sayesinde *receiver* rolünü de üstlenebilir. *Raspberry Pi* 4 üstlendiği role göre hem *transmitter* hem de *receiver* görevini üstlenebilir. Örneğin, *Raspberry Pi* 4'e bir sıcaklık sensörü bağlandığında ve bu sensör üzerinden veri alınmak istendiğinde, *Raspberry Pi* 4 önce sensöre bir okuma komutu gönderir. Bu aşamada Pi, *transmitter* rolündedir. Sensör, ölçtüğü sıcaklık verisini *Raspberry Pi* 4'e geri gönderdiğinde ise, *Raspberry Pi* 4 bu veriyi alan taraf olduğu için *receiver* rolünü üstlenir.

I²C haberleşme protokolü, düşük hızlı seri iletişim için geliştirilmiş, iki hat üzerinden çalışan senkron bir haberleşme sistemidir. Bu iki hat, SDA ve SCL olarak adlandırılır. Veriyolu üzerindeki cihazlar, bu iki hattı kullanarak veri alışverişi yapar (NXP Semiconductors, 2021).

I²C'nin donanımsal yapısında hem SDA hem de SCL hatları *open-drain* (açık drenaj) mimarisi kullanılarak tasarlanır. Bu tasarım, veri yolu üzerindeki her cihazın, hattı istediğinde aktif olarak çekebilmesini (*pull low*) ve istemediğinde serbest

bırakmasını sağlar. Hatların serbest bırakıldığı durumda, hatların belirsiz duruma düşmesini önlemek amacıyla, *pull-up* dirençleri (Rp) devreye girer ve hatları besleme voltajına (VDD) çeker. Şekil 2.4'te SDA ve SCL hatlarına bağlı *pull-up* dirençleri gösterilmiştir.



Şekil 2.4: I²C veriyolu üzerinde farklı teknolojilerle üretilmiş cihazların bağlantı yapısı ve pull-up dirençleri ile besleme hattı ilişkisi (NXP Semiconductors, 2021)

I²C haberleşme protokolünde SDA hattı veri iletimini, SCL hattı ise veri iletimi için zamanlama sinyali oluşturur. Bir cihaz veri göndermek istediğinde hattı aktif olarak 0 seviyesine çeker. Hattın serbest bırakıldığı durumda *pull-up* dirençleri tarafından VDD seviyesine çekilir. *Pull-up* direnç değerleri, hattın toplam kapasitansı ve haberleşme hızına göre uygun şekilde seçilmelidir. Tipik değerler 4.7kΩ ile 10kΩ arasında değişebilir. Direnç değeri ne kadar düşükse hattın 0 seviyesinden 1 seviyesine geçişi o kadar hızlı olur.

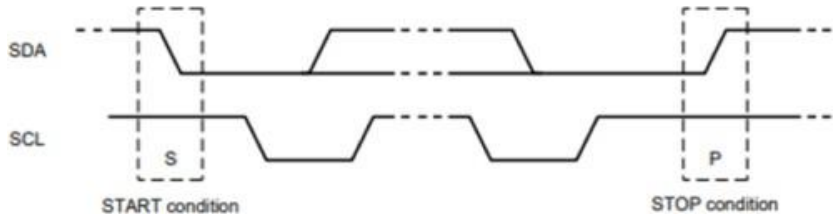
Farklı voltaj seviyelerinde çalışan cihazların aynı I²C hattına bağlanabilmesi için, her cihaz kendi besleme hattına bağlanır (VDD1, VDD2, VDD3 gibi). Bu sayede, voltaj seviyesinin uyumluluğu sağlanır ve karışık sistemlerde I²C haberleşmesi sorunsuz çalışabilir.

I²C haberleşme protokolü, çoklu cihaz destekleyen ve *controller-peripheral* mimarisi üzerine kurulu bir seri iletişim protokolüdür (Leens, 2009). Bu yapı sayesinde ana cihaz (*controller*) ve bağımlı cihaz (*peripheral*) arasında veri alışverişini yapmasını sağlar.

Controller cihaz, saat sinyali (SCL) üretir. Hangi cihazın konuşacağını *peripheral* belirler. *Peripheral* cihazlardan veri talep edebilir veya doğrudan veri

gönderebilir. *Peripheral* cihaza tanımlı bir adres atanır. *Controller* iletişim kuracağı zaman atanan bu adresi kullanır.

Peripheral cihaz, kontrolörden gelen sinyali dinler ve kendi sırası geldiğinde yani adres hedef olarak gösterildiğinde aktif hale geçer. *Controllerın* talebine göre sensör verisi gibi bilgileri gönderir ya da komut alır. Önemli olan husus *peripheral* cihazlar kendi saat sinyallerini üretmezler sadece kontrolörün sağladığı saat sinyaline senkron olurlar. Burada *controller Raspberry Pi*, *peripheral* ise sıcaklık sensörü olabilir. Şekil 2.5'te *START* ve *STOP* koşullarının geçiş seviyeleri gösterilmiştir.



Şekil 2.5: I²C protokolünde START ve STOP koşullarının sinyal diyagramı (NXP Semiconductors, 2021).

SDA hattının, SCL hattı yüksek seviyedeyken yüksekten alçak seviyeye (1'den 0'a) çekilmesiyle *START* koşulu oluşturulur. Bu koşul veri yolu üzerindeki tüm cihazlara kontrolörün yeni bir haberleşme başlattığını belirtir. SDA hattının, SCL hattı yüksek seviyedeyken alçaktan yüksek seviyeye (0'dan 1'e) geçmesiyle *STOP* koşulu oluşur. Bu koşul, haberleşmenin sona erdiğini ve veri yolunun serbest bırakıldığını bildirir.

I²C 8 bit veri transferinden sonra *receiver*, *transmitter* verinin başarıyla alınıp alınmadığını bildirmek için bir doğrulama (*acknowledgement*) sinyali gönderir (NXP Semiconductors, 2021).Bu doğrulama sürecinin iki durumu vardır. ACK durumu (Acknowledge – Onaylama), *receiver* gelen veriyi başarılı bir şekilde aldığını belirtmek için SDA hattını düşük seviyeye çeker. Böylelikle *transmittera* verinin sorunsuz alındığını bildirir ve bir sonraki 8 bitin gönderilmesine izin verir. Örneğin, *Raspberry Pi* 4 bir sıcaklık sensörüne komut gönderdiğinde, sensör bu komutu aldığını belirtmek için ACK sinyali gönderir.

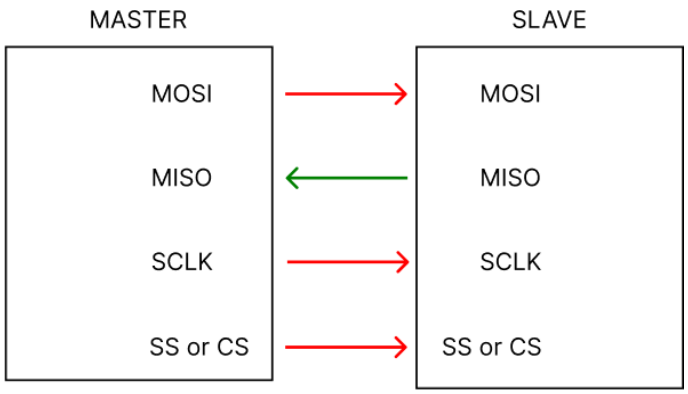
Bir diğer durum ise NACK (*Not Acknowledge*-Onaylamama), *receiver* kendisine gönderilen veriyi alamadığını ya da işlemin gerçekleştirilemeyeceğini göstermek için SDA hattını yüksek seviyede tutar. Böylelikle *transmittera* bir sorun olduğunu bildirir ve haberleşmeyi sonlandırır. Örneğin, *Raspberry Pi* 4, hafızasında kayıtlı olmayan bir sensör adresine veri gönderirse, ilgili cihaz yanıt vermez ve *bus* üzerinde NACK sinyali oluşur.

Raspberry Pi 4 ile I²C protokolü kullanarak bir sıcaklık sensöründen veri okuma işlemi, aşağıdaki adımlardan oluşan *pseudo code* (kaba kod) yapısıyla özetlenebilir:

1. *Raspberry Pi* üzerindeki *pigpio* servisi başlatılır.
2. Haberleşmenin sağlanacağı I²C hattı (*bus*) açılır.
3. İletişim kurulacak sensörün cihaz adresi belirlenir.
4. Sensörden veri okuma talebi gönderilir.
5. Sensörden gelen veri, bayt olarak okunur.
6. Okunan veri, sensörün özelliklerine uygun olarak sıcaklık değeri olarak dönüştürülür.
7. Elde edilen sıcaklık değeri, ekrana yazdırılır veya ilgili bir değişkene kaydedilir.
8. Kullanılan I²C hattı kapatılır.
9. *pigpio* servisi ile olan bağlantı sonlandırılarak haberleşme işlemi tamamlanır.

2.1.7.2.2 SPI Protokolü

Serial Peripheral Interface (SPI), gömülü sistemlerde ve bütünleşmiş devreler arasında senkron bir seri haberleşme protokolüdür. 1979 yılında Motorola tarafından geliştirilen bu protokol, özellikle mikrodenetleyiciler ve çevre birimleri arasında hızlı veri aktarımı sağlamak amacıyla tasarlanmıştır (Leens, 2009). SPI protokolü I²C protokolü gibi *controller-peripheral* mimarisi ile çalışır. Bir veya birden fazla *peripheral* cihaz *controller*'dan gelen komutlara göre veri gönderebilir veya alabilir. SPI haberleşmesi, dört temel hat üzerinden gerçekleştirilir.



Şekil 2.6: SPI Controller-Peripheral bağlantı şeması

Şekil 2.6, SPI haberleşme protokolünde *controller* ve *peripheral* cihazlar arasındaki temel bağlantıları göstermektedir. Verinin yönü, MOSI ve MISO hatları üzerinden belirlenirken, SS/CS hattı hangi peripheralın aktif olacağını kontrol etmektedir.

MOSI (*Master Out Slave In*): Verinin ana cihazdan (*controller*), *peripheral* cihaza veri gönderdiği hattır.

MISO (*Master In Slave Out*): Bu pin verinin *peripheral* cihazdan *controller* cihazına iletilmesini sağlayan hattır.

SCLK (*Serial Clock*): Verinin ne zaman okunup yazılacağını belirleyen saat sinyalini sağlar. *Controller* tarafından üretilir, tüm cihazlar bu saat sinyaline senkronize edilir. SPI'da veri aktarımı genellikle bu saat hattına eşzamanlı olarak gerçekleşmesini sağlar.

SS (*Peripheral Select*) *or CS* (*Chip Select*): Bu pin hangi *peripheral* cihazının aktif hale getirileceğini belirler. Bir SPI veri iletim hattında birden fazla *peripheral* cihaz bağlantı sağladığında her cihaz için ayrı bir SS/CS hattı bulunur. *Controller*, iletişim kurmak istediği *peripheral* cihazı seçmek için ilgili SS/CS hattını kullanır. Bir *peripheral* cihazın aktif hale gelmesi için bu hat düşük (0) seviyesine çekilir.

2.1.8 Güç Kaynağı

Raspberry Pi 4 Model B, yüksek performanslı işlemcisi ve gelişmiş donanımlarıyla uyumlu çalışabilmesi için uygun bir güç kaynağına ihtiyaç duyar. Cihazın enerji gereksinimlerini karşılamak ve olası performans sorunlarını önlemek için Raspberry Pi Vakfı 5.1V/3A USB-C güç adaptörünün kullanılmasını önerir (Raspberry Pi Foundation, 2024). Güç kaynağının özellikleri ise giriş voltajı 100-240V AC, 50/60Hz, 0.5 A, çıkış voltajı 5.1 V DC, çıkış akımı 3 A (3000 mA), bağlantı türü USB-C ve güç çıkışı 15.3 W sağlarken kablo uzunluğu ise 1.5 metredir, standart olarak beyaz renk adaptör kullanmaktadır. Bu adaptör, cihazın enerji ihtiyaçlarını karşılamak üzere özel olarak tasarlanmıştır. Üçüncü taraf adaptörlerin kullanımı, yetersiz güç sağlama veya voltaj dalgalanmaları nedeniyle cihazın stabilitesini olumsuz etkileyebilir.

2.2 Raspberry Pi 4 Model B'nin Yazılımsal Kurulumu

Bu bölümde, *Raspberry Pi* 4 Model B'nin yazılımsal kurulumu ve yapılandırma süreci detaylı olarak incelenmiştir. İşletim sistemi yükleme adımları, temel yapılandırmalar, uzaktan erişim yöntemleri ve sistem yönetimi ele alınarak, cihazın işlevselliğini artıran konfigürasyon seçenekleri anlatılmıştır.

2.2.1 Raspberry Pi İşletim Sistemi Kurulumu

Raspberry Pi için geliştirilen yerel işletim sistemi *Raspberry Pi* OS, *Debian* tabanlı bir sistemdir. İlk olarak 2012 yılında bağımsız geliştiriciler tarafından *Raspbian* adıyla oluşturulmuş ve 2015 yılında Raspberry Pi Vakfı tarafından resmi işletim sistemi olarak duyurulmuştur. 2020 yılında isim değişikliğine gidilerek *Raspberry Pi* OS adını almıştır. Açık kaynak kodlu olması, geniş bir kullanıcı topluluğuna sahip bulunması ve kütüphane desteğinin zenginliği nedeniyle diğer işletim sistemlerine kıyasla daha fazla kullanımı yaygın olabilir. Düşük güç tüketimi ve donanım ile tam uyumlu olacak şekilde optimize edilmiştir.

Raspbian, kullanımı kolay bir masaüstü ortamı sunarak *Raspberry Pi*'nin kullanımını kolaylaştırır. İşletim sistemi, Python programlama dili için geliştirme

araçları içerir, web tarayıcıları, medya oynatıcılar ve diğer kullanışlı *build-up* (yerleşik) yazılımlarıyla birlikte gelmektedir.

"OS işletim sisteminin kurulması için en az 8GB boyutunda Micro SD karta ihtiyaç duyulmaktadır" (Serhat Kağan Şahin ve Erdal Delebe, 2021).

Raspberry Pi Imager, çeşitli işletim sistemlerini *Raspberry Pi* tarafına kurulum programlarını kolaylaştırmak için Raspberry Pi Vakfı tarafından geliştirilmiş grafiksel kullanıcı arayüzdür (GUI). *Raspberry Pi* için sistemlere SD kartlara yazılmasını ve programlanmasını kolaylaştırır. Geliştiriciler tarafından programlanmış işletim sistemlerini listeden seçim yaparak doğrudan SD karta yazdırabilir.

Raspberry Pi Imager kullanımı şu adımları içerir:

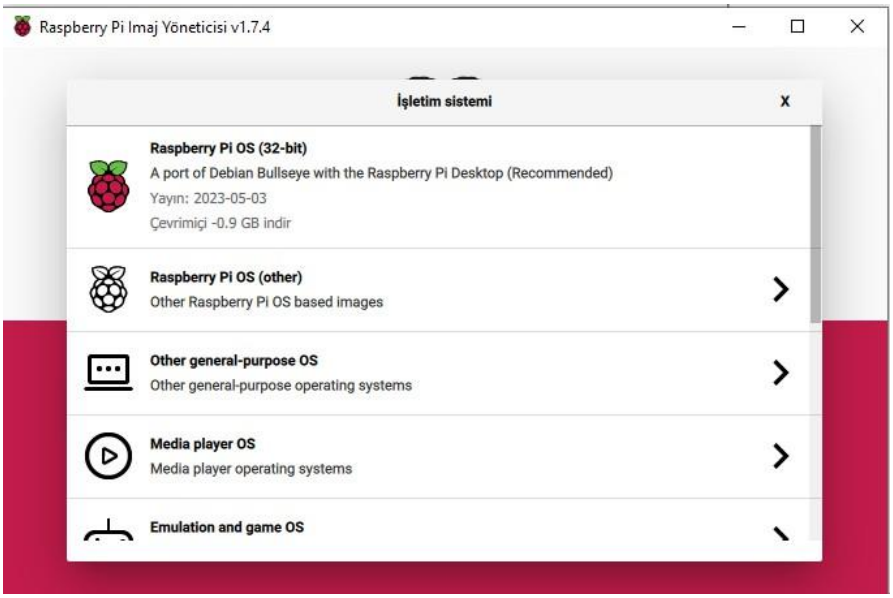
- *Raspberry Pi Imager* yazılımını indirilerek bilgisayara kurulur.
- Yazılım çalıştırılarak bir işletim sistemi seçilir (ör. *Raspbian, Ubuntu, LibreELEC* vb.).
- Boş bir SD kartı bilgisayara takılır ve *Raspberry Pi Imager* içinde bulunan SD kartı seçilir.

Şekil 2.7'de işletim sistemi seçim ekranı gösterilmiştir.

Şekil 2.7: Raspberry Pi işletim sistemi imaj yönetici ekranı

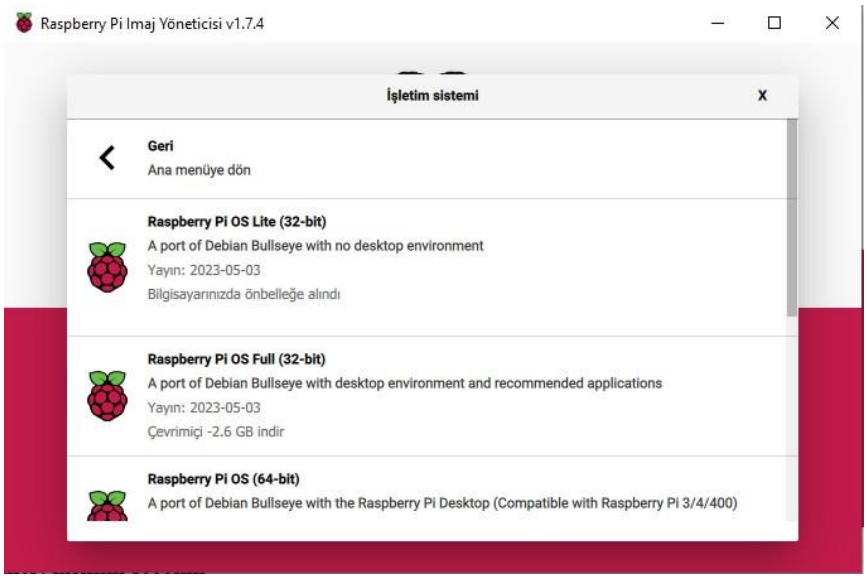
İŞLETİM SİSTEMİ SEÇİN butonu seçilerek işletim sistemi belirlenir.

Şekil 2.8'de *Raspberry Pi* OS *(other)* seçimi yapılacak görsel gösterilmiştir.



Şekil 2.8: Raspberry Pi işletim sistemi seçim ekranı

Şekil 2.9'da *Raspberry Pi Imager* arayüzü üzerinden *Raspberry Pi* OS *Lite* sürümünün seçim ekranı gösterilmektedir. Bu ekran üzerinden kullanıcı, masaüstü arayüzü içermeyen hafif işletim sistemini tercih edebilmektedir.



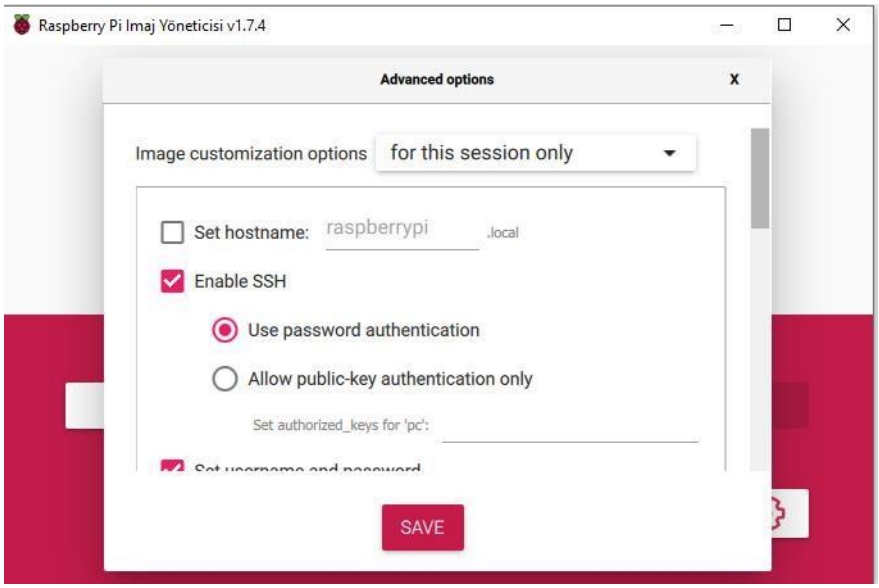
Şekil 2.9: Raspberry Pi OS Lite seçilen ekran

OS *Lite* grafiksel arayüz içermediğinden, SSH bağlantısı varsayılan olarak aktif gelmez ve yapılandırılması gerekir. *Raspberry Pi Imager* üzerinden SSH ayarları, ayarlar butonuna tıklanarak kolayca gerçekleştirilebilir. Bu yöntem, manuel olarak SSH dosyası oluşturmaya kıyasla daha hızlı ve pratiktir. Ayrıca, manuel olarak oluşturulan SSH dosyası *Raspberry Pi* tarafından her zaman algılanamayabilir, bu nedenle *Imager* üzerinden yapılandırma daha güvenilir bir seçenektir.

Şekil 2.10 *Raspberry Pi Imager* yazılımının ana ekranında yer alan ayarlar butonu gösterilmektedir. Bu buton aracılığıyla kullanıcı, işletim sistemi kurulumu öncesinde *hostname* belirleme, SSH etkinleştirme, kullanıcı oluşturma ve kablosuz ağ bağlantısı gibi ön ayarları yapabilmektedir.

Şekil 2.10: Ayarlar butonu

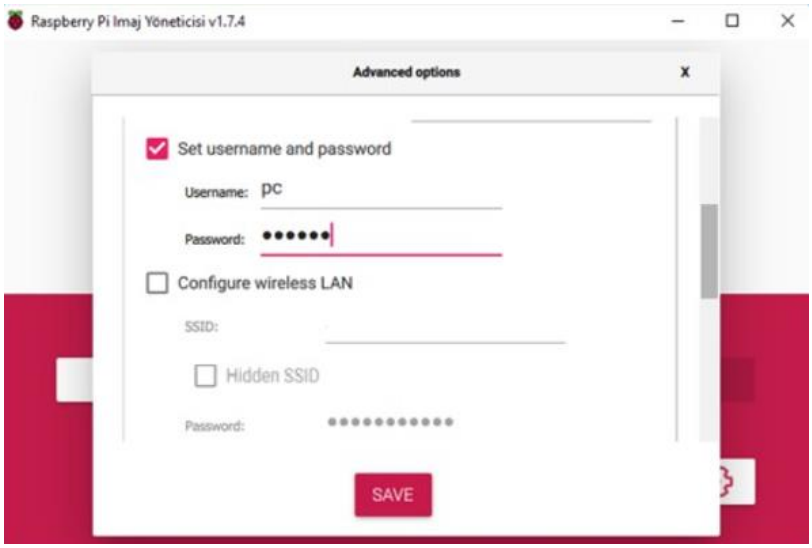
Şekil 2.11'de *Raspberry Pi Imager* yazılımında bulunan gelişmiş ayarlar menüsünden SSH yapılandırma seçenekleri gösterilmektedir.



Şekil 2.11: Gelişmiş Ayarlar (SSH yapılandırılması)

Use password authentication (Parola ile kimlik doğrulama), SSH bağlantısının kullanıcı adı ve parola ile yapılmasını sağlar. *Allow public-key authentication only* (Yalnızca genel anahtar ile kimlik doğrulama) SSH bağlantısının yalnızca belirli bir bilgisayara ait SSH anahtarı ile yapılmasına izin verir, daha güvenilir bir yöntemdir. Uzaktan erişimde yetkisiz bağlantıları önler. *Set authorized_keys for pc* (Ana bilgisayar adı belirleme) seçeneği ile *Raspberry Pi*'nin ağ üzerinde tanınacağı isim özelleştirilebilir. Böylece, statik IP gereksinimi olmadan cihaz belirlenen isimle erişilebilir hale gelir.

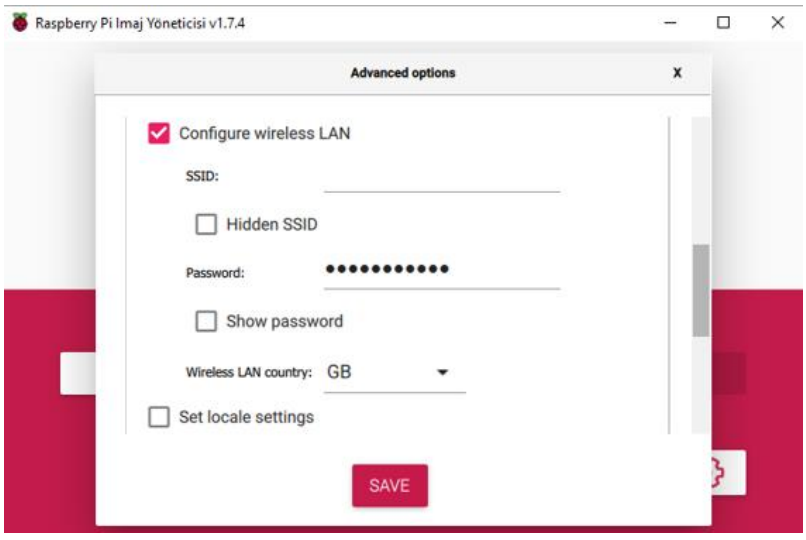
Şekil 2.12'de *Raspberry Pi Imager* yazılımı üzerinden kullanıcı adı ve şifre belirleme adımı gösterilmektedir.



Şekil 2.12: Kullanıcı adı ve şifre belirleme ekranı

Set username and password (Kullanıcı adı ve şifre belirleme) etkinleştirilerek kullanıcı adı ve şifre girilir. Varsayılan kullanıcı adı genellikle *pi* olarak atanır ancak burada özel bir isim verilebilir. Bu ayar yapıldığında *Raspberry Pi*'ye terminal veya SSH üzerinden oturumun açılmasını sağlar.

Şekil 2.13'te *Raspberry Pi Imager* yazılımı üzerinden kablosuz ağ (Wi-Fi) bağlantısının yapılandırıldığı ekran gösterilmektedir.



Şekil 2.13: Kablosuz Ağ yapılandırılması

Configure wireless LAN (Kablosuz ağ yapılandırma) seçeneği işaretlendiğinde *Raspberry Pi*'nin Wi-Fi ağına otomatik olarak bağlanması sağlanır. Masaüstü ortamı olmayan *OS Lite* sürümünde ağ bağlantısını baştan yapılandırmak iş yükünü azaltır. SSID bölümüne ağ adı (Wi-Fi ismi) girilir. *Hidden* SSID (Gizli SSID), Wi-Fi ağı SSID yayınlamıyorsa yani gizli ağsa bu kutucuk işaretlenmesi önerilir. *Password* (şifre) bölümüne ağ şifresi girilir. *Wireless LAN Country* (Kablosuz Ülke Kodu) *Raspberry Pi*'nin çalıştığı ülkeye göre Wi-Fi kanal ayarlarının belirlenmesini sağlar. Örneğin TR (Türkiye) ya da US (Amerika). Tüm ayarlar yapıldıktan sonra *SAVE* (kaydet) butonuna basıldığında *Write* (Yaz) düğmesine tıklayarak seçilen işletim sistemini SD karta yazdırmaya başlar. İşlem tamamlandığında, SD kartı çıkartılarak *Raspberry Pi* cihazına takılır. *Raspberry Pi* kurulan işletim sistemiyle kullanılmaya başlanır. *Raspberry Pi* başlatıldığında güncelleme yapılması önerilir.

2.2.2 Raspberry Pi OS Lite

Raspberry Pi Vakfı tarafından tasarlanmış, *Debian* ailesine ait olan *Linux* işletim sistemini kullanan *Raspbian'nın* hafif sürümüdür (Raspberry Pi Foundation,2025). *Raspberry Pi Imager* gibi bir GUI ve masaüstü uygulaması içermez. *Lite* ifadesi sadece komut satırı ile çalıştığını belirtir.

Raspberry Pi OS Lite, düşük güç tüketimi ve sınırlı işlem gücü gereksinimiyle çalışacak şekilde tasarlanmıştır. Sistem kaynaklarını verimli kullanarak geliştiricilere uygun bir çalışma ortamı sunar. Kullanıcı arayüzüne ihtiyaç duymayan sunucular, ağ yönetimi ve IoT projeleri için idealdir. Aynı zamanda *Lite* sürümü depolama alanından tasarruf sağlayarak sistemin daha hızlı ve verimli çalışmasını sağlar.

2.2.3 Raspberry Pi OS With Desktop

Raspberry Pi OS *Lite* gibi Linux işletim sistemine sahiptir. *Raspberry Pi* OS *With Desktop* grafiksel kullanıcı arayüzü içerir. Grafiksel kullanıcı arayüzü aracılıyla bir masaüstü ortamı ya da bir masaüstü uygulamalarıyla klavye, fare gibi araçları birlikte kullanabileceğiniz kişisel bir bilgisayar halinde kullanımına izin verir.

En son sürüm, 19 Kasım 2024 tarihinde yayınlanmış olup, 32-*bit* sistemler için tasarlanmıştır. Bu sürüm, *Debian* 12 (*bookworm*) tabanlıdır ve 6.6 sürüm numaralı *Linux* çekirdeğini içerir. İndirme boyutu yaklaşık 1,177 MB'dir (Raspberry Pi Foundation, 2025).

2.2.4 SSH (Secure Shell)

SSH 1995 yılında Helsinki Teknoloji Üniversitesi'nde araştırmacı olan Tatu Ylönen tarafından, üniversite ağında meydana gelen bir parola dinleme saldırısının ardından geliştirilmiştir. Ylönen, ilk SSH sürümünü Temmuz 1995'te ücretsiz olarak yayımlamış ve araç hızla popülerlik kazanmıştır. 1995 yılının sonunda, SSH kullanıcı sayısı elli ülkede 20.000'e ulaşmıştır. Aralık 1995'te Ylönen, SSH'yi pazarlamak ve geliştirmek amacıyla SSH *Communications Security* şirketini kurmuştur (SSH Communications Security, 2025).

Bu protokol, güvenli bir şekilde iletişim kurmayı amaçlayan 1995 yılında Finlandiya'da Helsinki Teknoloji Üniversitesinin geliştirmiş olduğu farklı bilgisayarlar üzerinden farklı sunuculara bağlantı kurmayı ve sunucu kimliğinin gizli kalmasını sağlayan bir kriptografik ağ protokolüdür (Tatu Ylonen ve Chris Lonvick, 2006). Sunucuları internet ortamından kontrol edilmesini sağlar. Sunucular tarafında değişiklik yapmayı sağlayan IoT projeleri için kullanışlı bir protokoldür.

SSH, kimlik güvenliği için güçlü bir algoritma kullanır ve veri iletimini korur. İzin verilen girişleri onaylar ve yetkilendirmeleri mümkün kılar.

2.2.5 Etcher

Etcher, resmi adıyla *balenaEtcher*, USB bellek veya SD kartlara işletim sistemi imaj dosyalarını (*.iso, .img*) yazdırmak için kullanılan ücretsiz ve açık kaynaklı bir uygulamadır. *Balena* tarafından geliştirilen bu araç, *Windows, macOS* ve *Linux* işletim sistemlerinde çalışır. *Raspberry Pi* gibi cihazlara işletim sistemi kurulumu yaparken tercih edilebilir.

İşletim sistemini sabit disk veya *SSD'ye* kurmadan *microSD* kart üzerine yazdırmak istersek *Etcher* manuel olarak bu kurulumu kolaylaştırır. *Raspberry Pi OS* işletim sistemi ISO dosyası olarak indirilir. *Etcher* açılarak indirilen ISO dosyası seçilir. Şekil 2.14'te balenaEtcher microSD kartına imaj kurulumu gösterilmiştir.

Şekil 2.14: Etcher kurulum

Yüklenecek olan *microSD* veya USB bellek belirtilir. *Flash* butonuna basarak işlem başlatılır. Alternatif olarak *Raspberry Pi Imager'da* kullanılabilir. *Etcher* farklı işletim sistemleri için de kullanılabilir.

2.2.6 Win32 Disk Imager

Bir işletim sisteminin imajını farklı programlarla almak mümkündür. İşletim sisteminin seçimine ve tercihlerine bağlı olarak farklı araçlar kullanılabilir.

Geliştiricilerin kullandığı bazı teknolojiler *Norton Ghost, Acronis True Image, Macrium Reflect Clonezilla, dd* komutu gibi araçlardır.

Win32 Disk Imager, Raspberry için SD kart imajı oluşturma noktasında işlevi yüksek olan ve tamamıyla ücretsiz olarak paylaşılan bir araçtır. SD kartın yedeğini alabilir, başka bir SD kartta bir *Raspberry Pi* işletim sistemi imajı için yazılabilir.

2.2.7 Git Bash Ortamı Üzerinden Raspberry Pi ile Bağlantı Kurma

Git Bash' e sırasıyla gönderilen kod satırları:

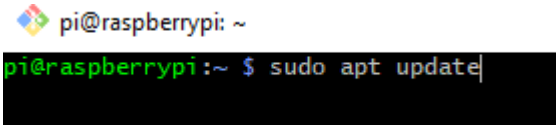
- *Git Bash* üzerinden SSH bağlantısı "*ssh pi@192.168.X.Y*" komutu ile gerçekleştirilir.

- Bağlantı sırasında sunucunun güvenlik kimliği doğrulamak için "*yes*" komutu girilir.

- *Password:* Kurulumda belirlenen şifreyle giriş sağlanır.

Last login ile başarılı bir giriş yapıldığı mesaj bilgisi ile anlaşılmaktadır.

İşletim sistemini her zaman güncel tutulması performans ve güvenlik açısından önemlidir Şekil 2.15'te gösterilen *sudo apt update* komutu çalıştırılarak sistem güncellenmesi sağlanır.

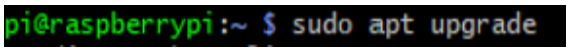


Şekil 2.15: İşletim sisteminin güncellenmesi için gereken kod satırı

sudo komutu, *Unix* ve *Linux* işletim sistemlerinde, bir kullanıcının başka bir kullanıcının yetkileriyle komut çalıştırmasını sağlayan bir komuttur. *sudo* kelimesi, *super user do* ifadesinden türetilerek oluşturulmuştur ve yönetici (*root*) yetkileriyle komut çalıştırmak için kullanılır.

apt (*Advanced Package Tool*) komutu, *Debian* tabanlı *Linux* sistemlerinde yazılım paketlerinin yüklenmesi, güncellenmesi ve kaldırılmasını yöneten bir paket yönetim aracıdır. Şekil 2.16'da bu paket kurulumu gösterilmiştir.

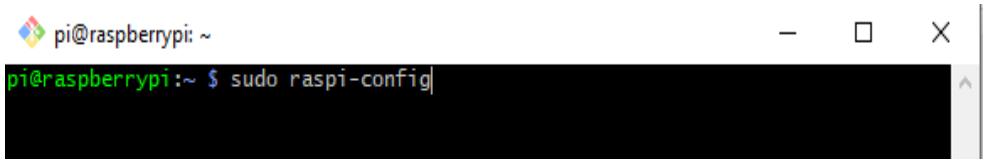
Raspberry Pi OS kurulumunun ardından sistemde yüklü olan paketlerin güncellenmesi gereklidir. Bu işlem, sistemin kararlılığı ve güvenliği açısından önem taşır. Paketlerin en son sürümlerine yükseltilmesi için terminal üzerinden *sudo apt upgrade* komutu kullanılmaktadır. Güncel olmayan paketler, performans kaybına ve güvenlik açıklarına neden olabileceğinden, imaj kurulumu sonrasında bu adımın uygulanması önerilmektedir.



Şekil 2.16: Paket yükseltme

İndirilen imaj tam güncel versiyonlarını içermez, içerisinde bulunan paketler farklı geliştiriciler tarafından yönetildiği için belirli aralıklarla güncellenir. Bu işlemleri ayda bir kere olacak şekilde gerçekleştirmek sistem kararlılığını korumak açısından faydalı olacaktır.

Şekil 2.17'de *Raspberry Pi'nin* sistem ayarlarının arayüzüne ulaşılacak komut satırı yer almaktadır.

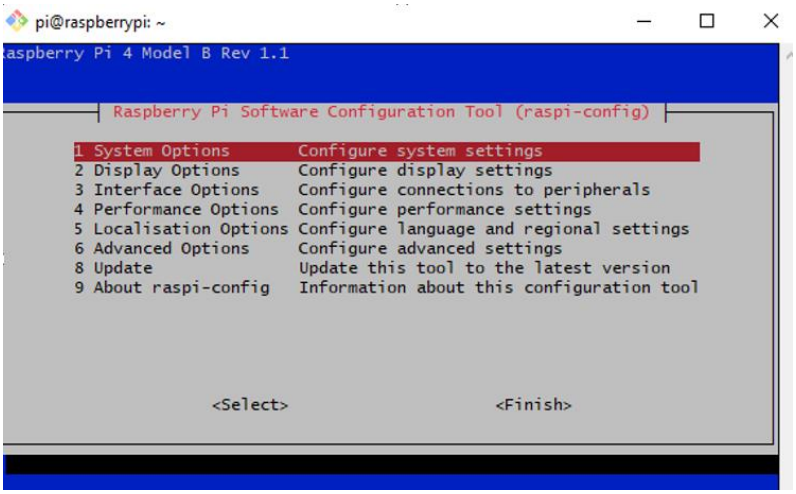


Şekil 2.17: Raspberry Pi yapılandırma komutu

sudo raspi-config ile şifre değiştirebilir, Wi-Fi ağının bağlantı konfigürasyonlarını değiştirebilir, *Boot* ayarlarını yönetebilir, I²C, SPI, ses, kamera gibi fonksiyonların aktif edilmesi gibi işlemler gerçekleştirilir.

2.2.8 Raspberry Pi Konfigürasyon Arayüzü ve Kullanımı

Bu bölümde, *raspi-config* arayüzünün tüm seçenekleri detaylı olarak ele alınarak *Raspberry Pi*'nin sistem ayarlarının nasıl değiştirilebileceği açıklanmıştır.



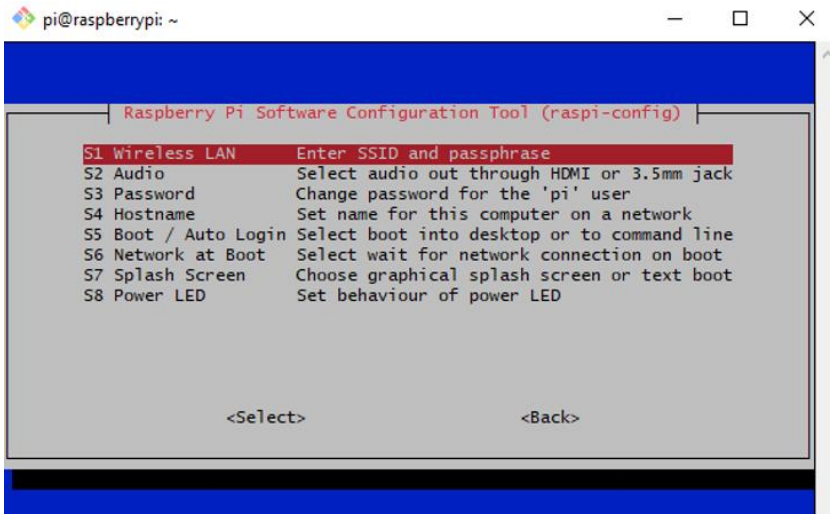
Şekil 2.18: Raspberry Pi yapılandırma menüsü

Şekil 2.18'de yer alan seçenekler şunlardır:

- *System Options*: Sistem ile ilgili temel ayarları yapmaya olanak tanır.
- *Display Options*: Görüntü ayarlarını yönetmek için kullanılır.
- *Interface Options*: Çevresel birimler ile bağlantı seçeneklerini yönetir.
- *Performance* Options: İşlemci, güç yönetimi ve performans optimizasyonlarını ayarlamak için kullanılır.
- *Localization Options*: Dil, saat dilimi ve klavye düzeni gibi ayarları içerir.
- *Advanced Options*: Daha detaylı sistem yapılandırmalarını içeren bölümdür.
- *Update*: *raspi-config* aracını en güncel sürüme yükseltmek için kullanılır.

2.2.8.1 Sistem Seçenekleri

Raspberry Pi sistem ayarlarına ilişkin temel yapılandırma seçenekleri, Şekil 2.19'da menü üzerinden kullanıcıya sunulmaktadır.



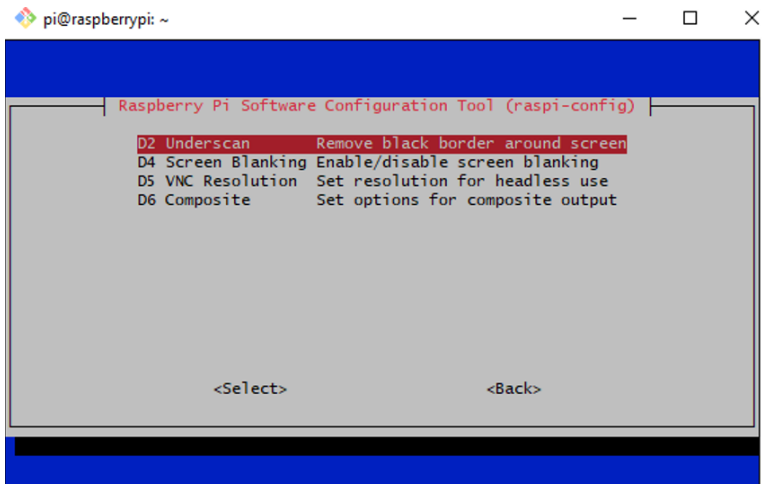
Şekil 2.19: Raspberry Pi sistem ayarları ana menü

1. S1 *Wireless LAN*: Wi-Fi ağına bağlantı kurmak için bir SSID ve şifre girilmesini ister. *Raspberry Pi*, belirtilen SSID ağı ayarlarıyla bağlantı kurar. Uzaktan bir ağ bağlantısı kullanmak hedefleniyorsa bu alan kullanışlıdır.
2. S2 *Audio*: Ses çıkış kaynağını belirleme seçeneğidir. *Raspberry Pi* 4'de bulunan mevcut ses çıkış yerleri HDMI ve ses Jak'ı üzerindendir. Bu seçenekte hangi ses çıkış noktası tercih edilecekse varsayılan olarak değiştirilebilir.
3. S3 *Password*: *Imager* kurulumu yapılan bölümde, oluşturulan kullanıcı adı ve şifre bu seçenek ile değiştirilebilmeyi sağlar.
4. S4 *Hostname*: Bu seçenek *Raspberry Pi'nin* ağdaki ismi değiştirilebilir.
5. S5 *Boot / Auto Login*: Cihazın grafik arayüz veya terminal modu ile başlatılması seçeneğidir. Bu seçenek ile Raspberry Pi'nin kullanım amacına uygun olarak başlangıç modu belirlenebilir. Grafik arayüz isteyen geliştiriciler için GUI modunu, uzaktan yönetim yapmak isteyen geliştiriciler için terminal modu tercih edilebilir.

6. S6 *Network at Boot: Raspberry Pi'nin* açılış sırasında ağ bağlantısı bekleyip beklemeyeceğini belirler.
7. S7 *Splash Screen*: Başlangıçta bir logo, sürüm bilgisi, tanıtım veya işletim sistemi bilgilerinin bir ekranda gösterilip gösterilmeyeceğini kontrol eden seçenektir. Açılış sürecinde işletim sistemi yüklenirken ekranda bir grafik arayüz veya metin tabanlı bilgiler görüntülenebilir.
8. S8 *Power* LED: Bu seçenek *Raspberry Pi'nin* üzerindeki güç (*power*) LED'inin çalışma durumunu kontrol etmeye yarar. Varsayılan olarak *Raspberry Pi* açıldığında güç LED'i sürekli yanar, ancak *Power* LED seçeneği ile değiştirilerek LED'in belirli koşullara göre yanıp sönmesi veya tamamen kapatılması sağlanabilir.

2.2.8.2 Ekran Seçenekleri

Raspberry Pi sisteminde, görüntüleme davranışlarını yapılandırmaya yönelik ayarlar Ekran Seçenekleri menüsü altında Şekil 2.20'de gösterilmiştir.



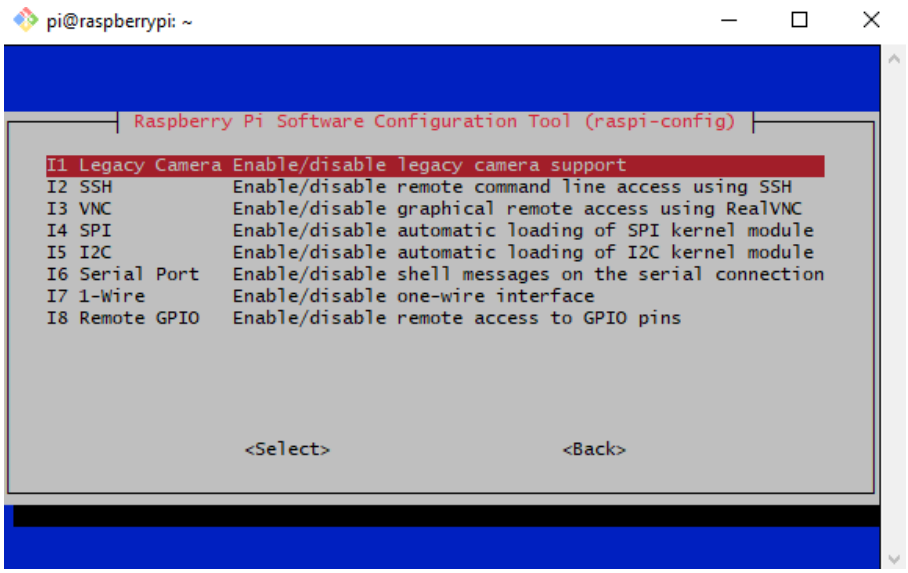
Şekil 2.20: Raspberry Pi ekran seçenek menü

1. D2 *Underscan*: Monitör optimize ayarının yapıldığı seçenektir. Genellikle ekran taşması veya tam ekran kaplanmıyorsa bu ayar ile düzeltilmeyi sağlar.

2. D4 *Screen Blanking*: *Raspberry Pi* belirtilen sürede bir işlem gerçekleştirmeden beklemeye geçtiğinde boş bir ekranın çıkartılıp çıkartılmamasını sağlayan kontroldür.
3. D5 VNC *Resolution*: VNC *(Virtual Network Computing) Resolution*, bilgisayardan bilgisayara erişimi sağlayan bir kontrol seçeneğidir. Ayrıca tek kart bilgisayarın ekran çözünürlüğü de ayarlanabilir.
4. D6 *Composite:* RCA video çıkışını kullanıp kullanılmayacağını kontrol eden seçenektir. HDMI yerine başka bir bağlantı kurmayı sağlar.

2.2.8.3 Arayüz Seçenekleri

Raspberry Pi cihazında, çeşitli donanım bileşenlerinin ve protokollerin etkinleştirilip devre dışı bırakılabildiği seçenekler, Arayüz Seçenekleri menüsü altında Şekil 2.21'de gösterilmiştir.



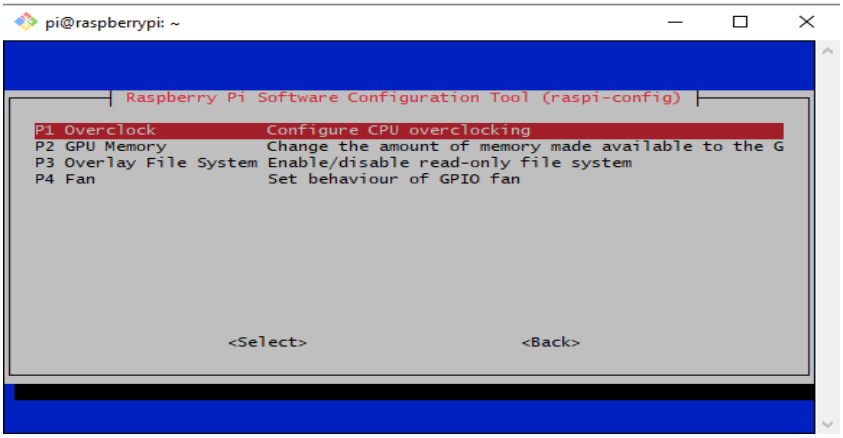
Şekil 2.21: Raspberry Pi arayüz seçenek menü

1. I1 *Legacy Camera*: Eski tip *Raspberry Pi* kamerasının kullanılıp kullanılmayacağını kontrol eder.
2. I2 SSH: Aynı ağa sahip olan *Rapberry Pi'yi* SSH etkinleştirildikten sonra komut satırı ile uzaktan erişim yapmayı sağlar. SSH sadece *terminal* üzerinden erişimi sağlar.

3. I3 VNC: *Raspberry Pi'ye* fiziksel erişim sağlayamadığımızda masaüstü ortamına uzaktan bağlantı kurarak cihazı kontrol etmeyi sağlayan bir protokoldür.
4. I4 SPI*:* Yakın menzil haberleşme için yararlanılan eşzamanlı bir seri haberleşme bağlantı noktası özelliği olan bu yapının bağlantı noktasının kullanılıp kullanılmayacağını kontrol eder.
5. I5 I²C*:* I²C bağlantı noktasının kullanılıp kullanılmayacağını kontrol eden seçenektir. Tez çalışmasında aktif hale getirilerek UPS modülünden veri çekme işlemi gerçekleştirilmiştir.
6. I6 *Serial Port:* Tek bir veri hattı yolunda tek bir bit veri göndermek için kullanılan seri portunun, hata ayıklama, uzaktan erişimi, düşük seviye donanım kontrolünü ve protokol desteği gibi özelliklerin kullanıp kullanılmayacağını kontrol eden seçenektir.
7. I7 *1-Wire: Raspberry Pi'nin* tek hat üzerinden düşük veri hızında cihazlarla haberleşmesini sağlayan protokoldür. Genellikle analog işlemi olan sıcaklık sensörü gibi basit aygıtlarla iletişim kurulup kurulmayacağını kontrol eden seçenektir.
8. I8 *Remote* GPIO*: Raspberry Pi* üzerinde bulunan GPIO pinlerinin ağ üzerinden uzaktan kontrol edilip edilemeyeceğini kontrol eden seçenektir.

2.2.8.4 Performans Seçenekleri

Raspberry Pi cihazlarında sistemin performansını etkileyen işlemci ve bellek ayarları, Performans Seçenekleri menüsü aracılığıyla yapılandırılabilir. Şekil 2.22'de menü gösterilmiştir.



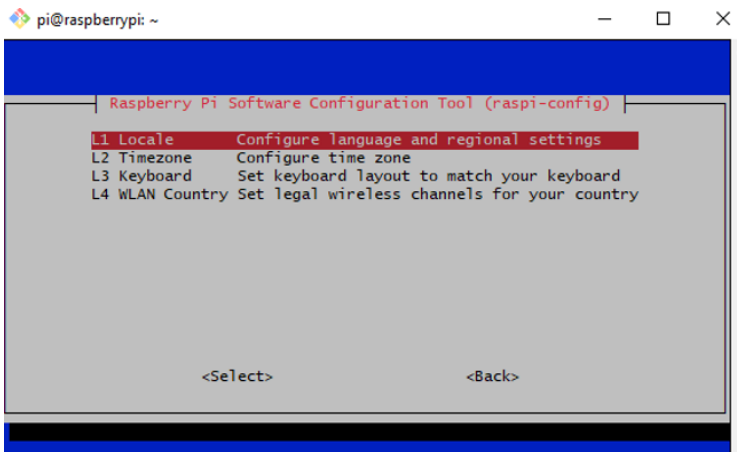
Şekil 2.22: Raspberry Pi performans seçenek menü

1. P1 *Overclock:* İşlemcinin varsayılan saniyede gerçekleştirdiği işlem sayısının artırılmasını sağlar. *Raspberry Pi* 1 ve *Raspberry Pi* 2 modellerinde işlemci hızı artırılarak performans kazanımı sağlanabilir. Varsayılan 700 MHz olarak ayarlanmıştır, ancak 1000 MHz'e kadar yükseltilebilir.
2. P2 GPU *Memory:* Grafik İşlemci Birimi Belleği Türkçe adıdır. *Raspberry Pi* hem CPU*'yu* hem de GPU destekler. GPU *Memory* seçeneği ile GPU'nun toplam sistem belleği içerisinde ne kadarına ulaşılabileceğini belirler. Daha yüksek bir GPU bellek seçeneği grafik ağırlıklı uygulamalar için daha çok tercih edilir. *Raspberry Pi'nin* GPU'su grafik hızlandırma ve video işlemi için tasarlanmıştır. Yapay zekâ uygulamalarını hızlandırmak için harici bir GPU veya *NVIDIA Jetson Nano* gibi hızlandırıcılar tercih edilir. Yapay zekâ modeli eğitiliyorsa bilgisayar veya bir sunucuda uygulayarak *Raspberry Pi'*de sadece çalıştırılması önerilir (Duman ve Şen, 2019). Daha düşük bir GPU bellek ayarı CPU*'ya* daha fazla RAM bırakır. Bu sayede video işleme, oyun geliştirme veya yapay zekâ ya da makine öğrenmesi uygulamalarında yaygın şekilde tercih edilebilir. CPU'yu bir yönetici olarak düşünürsek, GPU bu durumda bir grup işçidir. Yönetici karmaşık görevleri tek tek yaparken, işçiler aynı anda çok sayıda küçük işlemi hızlıca işler.

3. P3 *Overlay File System*: Dosya sisteminin yönetilme şeklini kontrol eder. *OverlayFS, upper layer* ve *lower layer* olmak üzere 2 katmana sahiptir. *Upper layer*, dosyaların veya verilerin üzerine yazıldığı örneğin kullanıcının ayarları, belgeleri ve değiştirildiği yerdir. *Lower layer*, salt okunur olan ve değiştirilmeyen örneğin *Linux* işletim sisteminin *imajı* katmanıdır.
4. P4 *Fan: Raspberry Pi 4* Model B ve diğer yüksek modellerde bir fan girişi bulunur. Bu fan girişinin kontrolünü bu seçenek ile kontrol etmek mümkündür. Fanın hangi zaman aralıklarında çalışacağı, ne kadar süre çalışacağı ayarlanır.

2.2.8.5 Yerelleştirme Seçenekleri

Raspberry Pi sisteminde tarih, saat, dil ve klavye düzeni gibi yerel ayarların yapılandırılması, Yerelleştirme Seçenekleri Şekil 2.23'te gösterilen menü üzerinden gerçekleştirilir.



Şekil 2.23: Raspberry Pi yerelleştirme seçenekleri menü

1. L1 *Locale: Raspberry Pi* bölge ve dil seçeneklerini kontrol eder.
2. L2 *Timezone:* Zaman Dilimi anlamına gel bu seçenek *ile Raspberry Pi'nin* saatini ve tarihini biçimlendirilmiş bir şekilde görüntülenmesini sağlar.
3. L3 *Keyboard*: Bu seçenek ile coğrafi bölgeye etken bir durumla değişecek şekilde klavyede bulunan karakter dizesinin karşılığının ne olacağı kontrol edilir.

4. L4 *WLAN Country*: Farklı ülkeler, kablosuz ağ frekansları ve kanallar üzerinde değişiklik göstereceği için bu seçenek ile *Raspberry Pi'nin* ağa ait olan ülke kodunun değiştirilmesi sağlanır. Örneğin Türkiye'de yaşayan bir kullanıcı için ülke kodu TR olarak girilmesi önerilir.

2.2.9 Pigpio Kütüphanesinin Kurulması

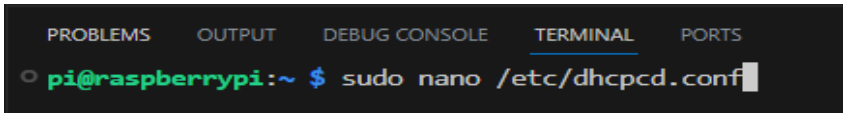
Pigpio geliştiriciler tarafından C programlama dili ile geliştirilmiştir. *Python* ile çalışabilen bir kütüphanedir. GPIO pinlerini kontrol etmek için kullanılır. Bu kütüphane sayesinde *Raspberry Pi'nin* donanımsal PMW, *servo* kontrolü, giriş/çıkış işlemleri ve haberleşme protokolleri olan SPI, I²C, UART gibi işlevleri yönetmek için kullanılır. Komut satırı olan *sudo apt install pigpio* ile kütüphanenin kurulumu sağlanır. İstenirse sanal ortamda yükleme işlemi gerçekleştirilebilir. Sanal ortam kullanılmak istenirse her projenin kendine özel bağımlılıklarına ve kütüphanelerine sahip olunması sağlanabilir. Bu sayede farklı *Python* kütüphanelerinin sürümlerinde karışıklılık olması engellenebilir.

Terminal ekranında *python -m venv Penv* komut satırı ile bir sanal ortam oluşturulur. *Penv* kısmına istenilen isim verilebilir. *source Penv /bin/activate* komut satırı ile sanal ortama geçiş sağlanır. *pip install pigpio* komut satırı ile *Penv* adlı sanal ortamda *pigpio* kütüphanesinin kurulumu sağlanır.

2.2.10 Statik IP Adresi

Bilgisayar ağlarında, istemciler genellikle Dinamik Ana Bilgisayar Yapılandırma Protokolü (Dynamic Host Configuration Protocol-DHCP) aracılığıyla yönlendirici veya bir DHCP sunucusu tarafından otomatik olarak atanan IP adreslerini kullanmaktadır. Ancak, Raspberry Pi gibi gömülü sistemler ve mikro bilgisayarlar için statik IP adresi atanması, belirli senaryolarda büyük bir gereklilik haline gelmektedir. *Raspberry Pi'ye* uzaktan erişim için SSH, VNC *(Virtual Network Computing)* veya uzak masaüstü bağlantıları kullanılmaktadır. Dinamik olarak atanan IP adresleri, her açılışta değişebildiğinden, kullanıcılar erişim sağlamak için sürekli olarak IP adresini kontrol etmek zorunda kalmaktadır. Statik bir IP adresi kullanımı, *Raspberry Pi'ye* her

zaman aynı IP adresi üzerinden erişim sağlanmasına olanak tanımaktadır. Şekil 2.24'te *dhcpcd.conf* bir metin düzenleyicide açmasını ve yapılandırılması istenilen komut satırlarının yazılmasını sağlayan görsel gösterilmiştir.



Şekil 2.24: Raspberry Pi DHCP metin düzenleyicisi komut satırı

Şekil 2.25 ve Şekil 2.26'da gösterilen şekilde açılan metin düzenleyicisinin en altına gitmek için CTRL+V tuş kombinasyonuna basarak gidilir.

```
interface eth0
static ip_address=<BELİRLENEN_IP>/24
static routers=<YÖNLENDİRİCİ_IP>
static domain_name_servers=<DNS_IP>
```

Şekil 2.25: dhcpcd.conf ile ethernet için statik ip konfigürasyonu

```
interface wlan0
static ip_address=<BELİRLENEN_IP>/24
static routers=<YÖNLENDİRİCİ_IP>
static domain_name_servers=<DNS_IP>
```

Şekil 2.26: dhcpcd.conf ile Wi-Fi için statik ip konfigürasyonu

eth0, Ethernet kablolu ağ bağlantısının yapılandırılması için kullanılırken *wlan0* kablosuz ağ bağlantı ayarlarının yapılandırılması için kullanılır. Hem *Ethernet* hem de Wi-Fi ağ yapılandırması için gerekli olan komut satırları girilmiştir. Değişiklikleri kaydetmek için CTRL+X ardından CTRL+Y ve ENTER tuşlarına basılır. Yapılan değişikliklerin etkili olması için *sudo reboot* ile *Raspberry Pi* yeniden başlatılır.

Raspberry Pi, IoT uygulamaları, ev otomasyonu ve akıllı cihazlar ile entegrasyon sağlamak amacıyla sıkça kullanılmaktadır. Bu tür sistemlerde, istemciler *Raspberry Pi'ye* önceden belirlenen IP adresi üzerinden veri göndermekte veya veri almakta olduğundan, dinamik IP kullanımı bağlantı sorunlarına yol açabilmektedir.

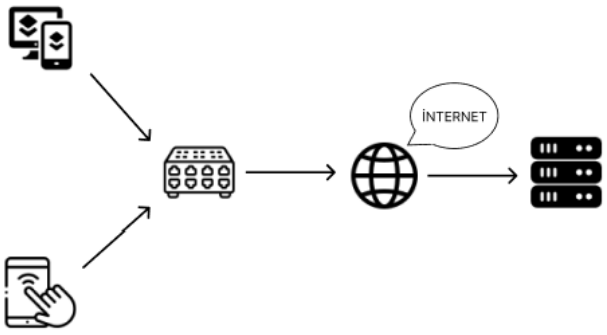
Statik IP atanması, cihazların *Raspberry Pi'yi* her zaman aynı ağ adresi üzerinden tanıyabilmesini ve veri iletişimini sürdürebilmesini sağlamaktadır.

3. AĞ VE İNTERNET HABERLEŞME PROTOKOLLERİ

Ağ üzerindeki cihazlar arasında veri alışverişini sağlayan kurallar ve standartlar ağ protokolleri olarak isimlendirilir. Birçok farklı kategoriye ayrılan ağ protokolleri arasından, bu çalışmada internet protokolleri ele alınmıştır.

3.1 TCP/IP

Transmission Control Protocol olarak adlandırılan internet protokolü ağ mimarisinin temel yapı taşıdır. İnternet trafiği cihazlar arasındaki iletişim TCP/IP protokolü vasıtasıyla sağlanır. Bağlantı odaklı iletişimi sağlayan TCP/IP protokolü güvenilir bir şekilde verilerin iletilmesini garanti eder. TCP, güvenilirlik sağlamak için hata algılama, onay mekanizmaları ve kaybolan paketlerin yeniden iletilmesi gibi yöntemler kullanır. Bu sayede, ağ tıkanıklığı ya da paket kaybı olsa bile veri bütünlüğü korunur. TCP çift yönlü iletişimi eş zamanlı olarak gerçekleştirir. Alıcı ve verici eş zamanda veriye ulaşır. TCP bağlantısı, üç aşamalı el sıkışma mekanizması ile kurulur. İlk olarak, istemci sunucuya bir SYN paketi gönderir. Sunucu bu mesajı alarak bir ACK paketi ile yanıt verir. İstemci son olarak sunucunun ACK mesajını alır ve bağlantıyı tamamlayarak veri transferine başlar. Bağlantı sonlandırma aşaması da benzer şekilde FIN paketi kullanılarak yapılır (Eddy, 2022). Şekil 3.1'de cihazların internet erişimi gösterilmektedir.



Şekil 3.1: Cihazların internet erişimi

IPv4, 1980'lerde TCP/IP protokolünün önemli bir parçası olarak tanıtıldı ve mantıksal adresler aracılığıyla cihazları tanımlamıştır. OSI modelinin ağ katmanında çalışarak verilerin yönlendirilmesini sağlar. 1970'lerde, *ARPANET* araştırmacıları IP adreslerinin uzunluğunu tartıştı ve 1977'de *Vint Cerf,* IPv4 adreslerinin 32 *bit* uzunluğunda olmasına karar verdi. IPv4 adresleri A, B, C, D ve E sınıflarına ayrılmıştır. E sınıfı gelecekteki kullanımlar için ayrılmıştır. IPv4, yaklaşık olarak 4,3 milyar benzersiz adresi sunmuştur.

Bir IPv4 adresi şu şekildedir: 209.31.201.249

İkili formatta gösterimi: 11010001.00011111.11001001.11111001 her biri 8 *bit* uzunluğunda dört *oktet* içerir ve toplamda 32 *bit* eder

A kümesinin ağ sayısı 126 *bit* ve ağ başına düşen *host* sayısı 16,777,214 iken C sınıfının ağ sayısı 2,097,150 ve ağ başına düşen *host* sayısı 254 uzunluğundadır. IPv4 adresleme sisteminde, alt ağ maskesi bir ağ adresi ile bir ana bilgisayarı ayırmak için kullanılır. IP adresi 192.168.0.103 ve 255.255.255.0 alt ağ maskesi kullanıldığında, bu adres 192.168.0.0 ağ adresini ve 103 numaralı ana bilgisayarı temsil eder. Alt ağ maskesi, IP adresindeki ağ ve cihaz kısımlarını ayırır. Bu durumda, 192.168.0 kısmı ağ kimliğini, 103 ise ana bilgisayar kimliğini gösterir.

Günümüzde, IPv4 adresleri tükenmek üzeredir (Yüce ve diğerleri, 2008). IPv4 yalnızca 4,3 milyar adres desteklerken, internete bağlı cihaz sayısı çok fazladır. Düzensiz IP tahsisi ve akıllı cihazların artışı bu sorunu büyütmüştür. Bu sorunu çözmek amacıyla geliştirilen IPv6, yaklaşık 3.4×10^{38} (2^{128}) adet adres kapasitesine sahiptir (Deering ve Hinden, 1998). Bu sayı, IPv4'ün sunduğu yaklaşık 4,3 milyar (2^{32}) adresin yaklaşık $7{,}9\times10^{28}$ katı büyüklüğündedir. Bu, her insan, cihaz ve nesne için neredeyse sınırsız IP adresi anlamına gelmektedir. Ancak, IPv6'ya geçiş süreci yavaş ilerlemektedir. Bunun sebepleri, mevcut altyapının IPv4 üzerine inşa edilmiş olması ve işletmelerin IPv6 uyumluluğuna geçiş için gereken zaman ve maliyetlerdir.

İnternet protokolünün 6. sürümü olan ve IPv4'ün alt yapısını kullanarak geliştirilmiş ve IPv4'ün yerini alması için geliştirmeler yapılan protokol IPv6, IPv4 ile aynı mantıkta çalışır. İnternet ile iletişim kurabilen cihazlara benzersiz sayısal IP adresleri atar. IPv4'ün adres alanı tükenmesi, güvenlik açıkları, otomatik yapılandırma

eksiklikleri haberleşme protokolünün IPv6 teknolojisinin kullanılışına yöneltebilir. IPv6'nın en büyük farkı 128 *bitlik* adresleme kullanmasıdır. Bir IPv6 adresi şu şekildedir: 67b7:925c:0032:d61f:abeb:1275:c02e:4f2e iki nokta üst üste (:) kullanılarak 8 gruba ayrılmıştır ve her grup 16 *bit* temsil etmektedir. Her onaltılık sayı sistemi 4 *bitlik* bilgi taşır (Hossain, Binti ve Uddin, 2024).

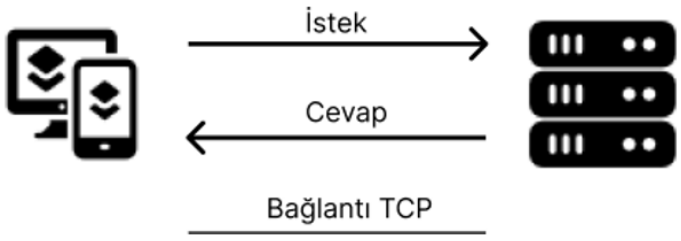
3.2 UDP-User Datagram Protocol

Türkçe karşılığı Kullanıcı Veri Bloğu İletişim Kuralları olan UDP, tek katmanlı bir aktarım protokolüdür. Kaynak cihaz UDP'yi kullanarak veriyi gönderir, hedef cihaz UDP kullanarak veriyi alır. TCP SYN ve ACK segmentleri kullanırken, UDP bağlantısızdır ve doğrudan veri paketleri gönderir. TCP'ye göre oldukça hızlıdır fakat verinin adrese ulaşıp ulaşmadığıyla ilgilenmez, bu yüzden hatalara tolerans gösterir, güvenirliliği ortadan kalkar. Bağlantı güvenirliliği düşük olduğundan sesli ve görüntülü iletişimde, video akışında, çevrim içi oyunlarda yaygın olarak kullanılır (Walugembe ve Kawuma, 2023).

3.3 HTTP/HTTPS

HTTP, *World Wide Web* (WWW) üzerinde iletişimi sağlar ve web sayfaların istemci-sunucu arasında aktarılmasına olanak tanır. TCP/IP protokolü üzerinde çalışarak istek-cevap modelini kullanır. İstemci, sunucuya kaynak talebi gönderir ve sunucu, bu isteğe cevap verir (Muller, 2014).

Http'nin temel özelliklerinden biri durumsuz bir protokol olmasıdır, her istek ve yanıt birbirinden bağımsızdır. Bir istemci *web* sunucusuna bir HTTP isteği gönderdiğinde sunucu bu isteği genellikle XML veya JSON formatında işler ve yanıt döndürür. HTTP protokolündeki durumsuz yapı gönderilen istekleri bir sonraki seferde hatırlamaz. HTTP, verileri açık metin olarak iletir, bu da verilerin ele geçirilmesini oldukça kolaylaştırır. 80 numaralı port HTTP için ayrılmıştır. Şekil 3.2'de istemci sunucu istek cevap bağlantısı gösterilmiştir.



Şekil 3.2: İstemci-Sunucu TCP bağlantısı

HTTPS, HTTP protokolüne eklenen bir güvenlik protokolüdür ve iletişimi şifreler. Temel özellikleri arasında şifreleme, dijital sertifikalar ve güven göstergeleri vardır. Şifreleme, yetkisiz kişilerin verilere erişimini engeller. Dijital sertifikalar, *web* sitelerinin kimliğini doğrular ve güvenli olmasını sağlar. Ayrıca, *web* tarayıcılarında kilit simgesi ve *https*:// öneki ile güvenli bağlantılar gösterilir.

HTTP, kullanıcı adı ve şifre gibi hassas bilgilerin güvenliğini sağlayamaz çünkü iletilen verileri şifrelemez. HTTPS ise TLS protokolü kullanarak bu bilgileri şifreler ve güvenli hale gelmesini sağlar. Örneğin, kullanıcı adı *baranselaydin* ve şifre 123 olan bir kullanıcı, belirli bir şifreleme algoritması ile şifrelendiğinde kullanıcı adı AiWRDOzYmd7FleP4nNkiYQ== şifresi ise wEcRsbL8EEwtUIOTcLo/8A== şeklinde olabilir. Bu şifreler kullanılan algoritma ve doğru anahtar ile tekrar çözümlenebilir. HTTPS kullandığında, kullanıcı adı ve şifre dahil olmak üzere tüm HTTP istekleri şifreli tünel (*encrypted tunnel*) üzerinden iletilir. Bu, saldırganların verileri ele geçirmesi halinde bile anlamlı hale getirememesini sağlar. HTTPS, bu şifrelemeyi kendisi sağlamaz, ancak şifrelenmiş veya düz metin verileri güvenli bir şekilde karşı tarafa ulaştırır.

REST'in HTTP ile entegrasyonu, IoT cihazlarının durum bilgilerini standart bir yöntemle paylaşmasını sağlar. CRUD işlemleri ile *HTTP* metotları arasında doğrudan bağlantılar kurar. *POST*, kaynak oluşturma; *GET*, kaynak okuma; *PUT*, kaynak güncelleme; *DELETE*, kaynak silme görevlerini yerine getirir. JSON ve XML veri formatları desteklenir, IoT sistemlerinde JSON tercih edilebilir.

HTTP'nin yaygın kullanımı, güçlü çerçeveler ve birçok hazır kütüphanenin REST API'leri desteklemesi sayesinde IoT cihazları için önemli avantajlar

sunmaktadır. HTTPS, IoT cihazlarının güvenliğini artırarak kötü niyetli kullanıcıların veri hırsızlığını önlemede etkili bir yöntemdir. Ancak, büyük veri iletimleri kaynak açısından kısıtlı cihazlarda gecikmelere yol açabilir ve gereksiz bir yük oluşturabilir. Ayrıca, HTTP'nin doğası gereği anlık bildirimler için uygun olmadığı da bilinmektedir. IoT tabanlı sistemlerde, sunucudan bir istek gelmeden cihazın doğrudan bildirim göndermesi mümkün değildir.

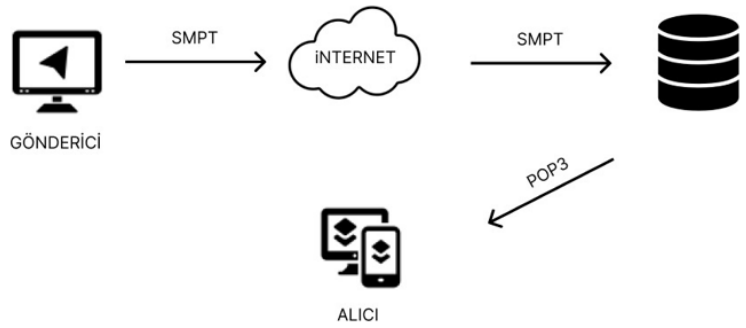
3.4 SMTP/IMAP/POP3

E-posta iletişimi, SMTP, POP3 ve IMAP gibi protokoller aracılığıyla sağlanır. Bu protokoller, e-posta gönderme, alma ve yönetme süreçlerini belirler. SMTP elektronik postaların sunucular arasında gönderilmesini sağlayan temel bir protokoldür. Standart bir iletim protokolü olup, güvenilir teslimat sağlar. Varsayılan olarak iletileri şifrelemez ve düz metin olarak aktarır. Bir e-posta ilgili adrese teslim edilmezse, teslim alan kişi hata mesajı alacaktır. Güvenlik ve şifre içermediği için ek güvenlik sağlanmazsa kötü niyetli geliştiriciler tarafından içerik okunacaktır. Geliştiriciler tarafından SMTPS kullanıma sunularak e-posta iletişiminin güvenliği artırılmıştır. SMTPS, güvenli bir bağlantıyı SSL veya TLS protokolleriyle şifreleyerek yapılandırır. Bu şifreleme yöntemi, saldırganların e-posta içeriğini okumasını ve verileri ele geçirmesini zorlaştırır (Chen ve Wanner, 2022).

Bir SMTP sunucusunu yalnızca bir bilgisayara yüklemek, e-posta hizmetinin çalışması için tek başına yeterli değildir. SMTP'nin düzgün çalışabilmesi için bağlantı noktası numarasının, güvenlik protokollerinin ve diğer sunucu ayarlarının doğru şekilde yapılandırılması gerekmektedir.

IMAP 1986 yılında Mark Crispin tarafından Stanford Üniversitesi'nde geliştirilmiştir. 1996 yılında ise Microsoft, IMAP'e alternatif olarak MAPI protokolünü sunmuştur. IMAP, e-postalara tüm cihazlardan erişim imkânı sağlayan bir protokoldür. En büyük avantajlarından biri, e-posta iletilerini cihazlara indirmeden doğrudan sunucu üzerinden görüntüleyebilmesidir. Kullanıcı, e-postayı indirmeye ihtiyaç duymadan sadece üzerine tıklayarak okuyabilir. Ayrıca, IMAP sayesinde elektronik postalar organize edilebilir ve klasörleme yapısı oluşturularak düzenli bir şekilde yönetilebilir. IMAP her erişim isteğinde verileri doğrudan sunucudan aldığı

için ağ trafiğini artırabilir ve sunucu kaynaklarını tüketebilir. Ayrıca, IMAP bağlantısına sahip bir istemcinin e-posta okuma ve yönetme işlemleri için aktif bir internet bağlantısına ihtiyacı vardır. Dolayısıyla, internet kesintisi durumunda e-postalara erişim sağlanamaz. Şekil 3.3'te elektronik posta gönderim ve alım süreci gösterilmiştir.



Şekil 3.3: E-Posta gönderim ve alım süreci

1984 yılında Amerika'da geliştirilen POP, iki yıl sonra POP3 sürümüne geçmiştir. POP3, tek yönlü bir iletişim protokolüdür ve e-postaların istemci cihazlara indirilerek yerel olarak saklanmasını sağlar (John G. Myers ve Marshall T. Rose, 1996).

E-posta gönderimi SMTP aracılığıyla gerçekleştirilirken, POP3, sunucudaki e-postaları almak için kullanılır. POP3 için varsayılan bağlantı noktası 110 numaralı porttur.

POP3, istemci cihazlara e-postaları indirerek çevrimdışı erişim imkânı sağlar. E- postalar indirilip cihazda saklandıktan sonra sunucudan silinebilir, bu da sunucu ile senkronizasyonun ortadan kalkmasına neden olur. POP3'ün kullanılabilmesi için istemci tarafında gerekli yapılandırmaların yapılması gerekmektedir.

3.5 Bluetooth

1994 yılında *Ericsson* tarafından geliştirilen ve kısa mesafede kablo gereksinimi olmadan iletişim sağlayan bir teknolojidir (Zeadally ve diğ., 2019). İki veya daha fazla cihazın kabloya ihtiyaç duymadan veri alışverişinde bulunmasını

olanak sağlayan *Bluetooth*, ilk olarak kablosuz fareler, klavyeler ve yazıcılar gibi cihazlarda yaygın olarak kullanılmıştır.

2010 yılında *Bluetooth* 4.0 standardının tanıtılmasıyla birlikte, düşük güç tüketimine sahip BLE teknolojisi hayatımıza hızlı bir şekilde girmiş ve özellikle akıllı saatler, *fitness* bileklikleri ve IoT tabanlı cihazlarda yaygın olarak kullanılmaya başlanmıştır (Zeadally ve diğ., 2019).

Bluetooth, Wi-Fi'ın aynı çalışma mantığında radyo frekanslarını kullanarak 2.4 GHz bandında iletişim sağlar. BLE 5.0 standardının geliştirilmesiyle birlikte, iletişim mesafesi 300 metreye kadar çıkartılmış, veri aktarım hızı artırılmıştır. Günümüzde kullanılan *Bluetooth* teknolojisi, veri transferinde 48 Mbps'ye kadar hız sunabilmektedir. Ancak, cihazın donanım özellikleri, çekim gücü, çevresel faktörler ve engeller gibi değişkenler mesafe ve bağlantı kalitesini etkileyebilir.

Raspberry Pi 3 modeli *Bluetooth* 4.1, *Raspberry Pi* 3 B+ modeli *Bluetooth* 4.2, *Raspberry Pi* 4 modeli ise *Bluetooth* 5.0 teknolojisini kullanmaktadır. Sürümler arasındaki farklar performans açısından önemli bir yer teşkil etmektedir. *Bluetooth* 4.1 24 Mbps hız, 100-300 metre mesafe, 2.4-2.485 GHz bant genişliği sağlarken, *Bluetooth* 5.0 48 Mbps hız, 985 metreye kadar mesafe, gelişmiş veri aktarımı ve daha düşük enerji tüketimi sunar (Raspberry Pi Foundation, 2025). *Raspberry Pi* cihazlarında yerleşik *Bluetooth* modülü bulunur, kablosuz iletişim için rahatlıkla bu modül kullanılabilir. Bu bluetooth modülü varsayılan olarak pasif durumda sunulur ve kullanılabilmesi için konfigüre edilerek aktif hale getirilmesi gerekmektedir.

Bluetooth teknolojisi, IoT sistemlerinde de önemli bir rol oynamaktadır. Örneğin, bir *Raspberry Pi* üzerinde bulunan *Bluetooth* modülü kullanılarak, DHT11 sıcaklık ve nem sensörü aracılığıyla ortam sıcaklık verileri ölçülerek *Bluetooth* üzerinden diğer cihazlara veriler aktarılabilir. Bu tür bir uygulamada, ilgili kütüphanelerin yüklenmesi ve *Bluetooth* protokollerinin kullanılması gerekmektedir.

Bluetooth teknolojisi kısa mesafeli kablosuz veri aktarımı için güvenilir, maliyeti düşük ve verimli bir çalışma prensibine sahip olup, IoT akıllı cihazları için düşük enerji tüketimiyle öne çıkmaktadır.

3.6 Web Servisleri ve API Haberleşme Protokolleri

Web servisleri ve uygulamalar arasında veri alışverişini sağlar.

3.6.1 SOAP (Simple Object Access Protocol)

SOAP'ın ortaya çıkışı, kurumsal yazılımların internet üzerinden etkileşimde bulunmasını kolaylaştırmak amacıyla olmuştur. İlk olarak *Microsoft* ve IBM gibi şirketlerin desteğiyle geliştirilmiş ve ardından W3C tarafından standartlaştırılmıştır (Tekli ve diğerleri, 2012). SOAP, ilk versiyonlarında yalnızca HTTP üzerinde çalışıyordu ancak zamanla SMTP, FTP gibi protokoller de desteklenmeye başlandı. *Web* servislerinin geniş bir kullanıcı kitlesine hitap etmesini sağlamak amacıyla XML tabanlı veri formatı benimsenmiştir böylece platformdan bağımsız bir mesajlaşma mekanizması oluşturulmuştur.

SOAP, zaman içinde REST ve *GraphQL* gibi alternatiflerle kıyaslandığında daha karmaşık ve ağır bir sistem olarak görülmeye başlandı (Tekli ve diğerleri, 2012). Güvenlik gereksinimlerinin yüksek olduğu ve güvenilir mesaj iletiminin kritik olduğu senaryolarda hala kullanılmaktadır.

Mesajlaşma yapısı genellikle 3 ana bileşenden oluşur:

Envelope (Zarf): Mesajın temel çerçevesini tanımlar ve SOAP mesajının başlangıcını belirtir.

Header (Başlık): Güvenlik, oturum yönetimi gibi ek bilgileri içerir.

Body (Gövde): Asıl verinin yer aldığı kısımdır; istemciden gelen talepler veya sunucudan dönen yanıtları içerir (Tekli ve diğerleri, 2012).

Mesajlar, HTTP, SMTP veya JMS (*Java Messaging System*) gibi farklı protokoller üzerinden taşınabilir. SOAP, genellikle HTTP üzerinden çalıştırıldığında *POST* metoduyla veri iletir ve yanıt alır.

SOAP, kurumsal çözümler ve güvenli haberleşme sistemlerini gerektiren uygulamalar için hala güçlü bir teknoloji olarak yerini korumaktadır. Banka sistemleri, finansal işlemlerde kullanılıyor olabilir. Ancak, mobil uygulamalar, IoT cihazları ve hafif web servisleri gibi alanlarda REST ve *GraphQL* gibi daha modern teknolojiler tercih edilebilir.

Gelecekte SOAP'ın daha optimize edilmiş ve hızlı çalışan bir versiyonu geliştirilmesi beklenmektedir. Belki de JSON-XML hibrit çözümler veya SOAP'ın içerdiği güvenlik mekanizmalarını REST ile entegre eden hibrit yaklaşımlar daha yaygın hale gelebilir.

3.6.2 RESTful API

RESTful API internet üzerinden istemciler ve sunucular arasında veri alışverişi yapmayı sağlayan hafif bir mimari teknolojisidir. REST, *Roy Fielding* tarafından 2000 yılında geliştirilmiştir, istemci-sunucu, *stateless* (durumsuz), önbelleklenebilirlik, katmanlı mimari, birbirinden bağımsız bileşenler gibi temel ilkeleri içerir (Malakhov, Kurgaev ve Velychko, 2018).

REST mimarisinin en önemli avantajlarından biri, hafif olması ve HTTP protokolünü kullanmasıdır. Bu sayede nesnelerin interneti gibi düşük bant genişliğine sahip ve hafif veri alışverişine ihtiyaç duyan sistemlerde etkili bir şekilde kullanılabilir. *RESTful* API'ler, mikro hizmetler ve bulut hizmetleri ile entegre olarak IoT cihazlarının veri alışverişini daha verimli hale getirir (Ehsan ve diğerleri, 2022). *RESTful* API'nin ana temel metotları ise, *GET*, Sunucudan verileri almak için kullanılır. *GET* istekleri tarayıcı yığınında görünür. IoT cihazının sıcaklık sensöründen veri çekmek için *GET* isteği yapılabilir. *GET http://localhost:5000/API/ Measurements / GetMeasurements/* içeriği ise Şekil 3.4'te gösterilen JSON formatında olabilir.

Şekil 3.4: JSON formatı

İstenirse gerekli katmanlarda metotlar yazılırak belirli bir sıcaklık değerini de getirilebilir. Burada ilgili sıcaklığın benzersiz kimliğini alarak getirme işleminin örneğini yapalım.

GET http://localhost:5000/API/ Measurements /GetMeasurementsById/1

POST: Yeni bir veri oluşturmak için kullanılır. Bir IoT cihazı, merkezi bir sunucuya sensör verilerini kaydetmek için *POST* isteği gönderebilir. *POST* yanıtları genellikle önbelleğe alınmaz, yer imlerine kaydedilemez. *POST http://localhost:5000/API/Measurements* metoduyla ilgili parametreleri girerek veri kaydetme işlemini gerçekleştirilir.

PUT: Var olan bir veriyi güncellemek için kullanılır. IoT cihazının parametrelerini güncellemek için kullanılabilir. *PUT http://localhost:5000/*API*/Measurements/{id}* isteğinde bulunulur. Gönderilen *id* bilgisine göre istenilen parametreleri değiştirmek mümkündür. *PUT id'yi* URL'de alır, ancak gövdede de olmalıdır.

200 OK durumu başarılı bir güncelleme anlamına gelir. Ancak gönderilen *id* parametresi sunucuda bulunamazsa 404 *Not Found* döndürecektir.

DELETE: Belirli bir veriyi silmek için kullanılır. Bir akıllı ev sisteminde bir cihaz kaydını silmek için kullanılabilir. *DELETE http://localhost:5000/*API*/Measurements/{id}* gönderilen *id* bilgisi değerine göre silme işlemini gerçekleştirecektir. İşlem başarılı olursa 200 OK ve bir başarı mesajı döndürmelidir. Ancak gönderilen *id* parametresi sunucu tarafında bir karşılığı yoksa 204 *No Content* mesajını döndürebilir, kullanıcı alışkanlığı açısından mesaj bilgisi döndürmek daha kullanışlıdır.

REST API'leri genellikle JSON formatında veri döndürür. JSON, hafif ve kullanıcı tarafından kolay bir şekilde okunabilir bir format olduğu için IoT cihazları arasında veri alışverişinde yaygın olarak tercih edilir. Akıllı Ev Sistemleri, akıllı termostatlar, lambalar ve güvenlik kameraları gibi cihazlar REST API'ler aracılığıyla bir merkezi kontrol birimiyle iletişim kurabilir. Endüstriyel IoT (IIoT) üretim hatlarındaki makineler, çalışma durumu ve bakım gereksinimleri hakkında bilgi göndermek için REST API'leri kullanabilir.

Dezavantajlarında ise REST'in *stateless* yapısı nedeniyle bazı durumlarda istemcinin her istekte gerekli bilgileri göndermesi gerekebilir. Anlık bildirim durum yapısı olmadığından istekte bulunulması gerekir. Bu, büyük ölçekli uygulamalarda ek bant genişliği tüketimine neden olabilir. IoT sistemlerinde verimliliği artırmak için *caching* (ön bellekleme) ve *rate limiting* (istek sınırlandırma) gibi yapıların kullanılması önerilmektedir (Rodríguez ve diğerleri, 2016).

Sonuç olarak, REST API'leri IoT'nin gelişimi için önemli bir teknolojidir. Güvenlik, test süreçleri ve performans optimizasyonu gibi konular üzerinde durulması gerekir. Özellikle büyük ölçekli IoT projelerinde, API'lerin iyi tasarlanmış ve optimize edilmiş olması sistemin başarısını doğrudan etkileyecektir.

3.6.3 GraphQL

Geleneksel REST tabanlı API çözümleri, IoT cihazlarının ihtiyacı olan verimliliği sağlamakta yetersiz kalabilir. *GraphQL'in* esnek veri sorgulama mekanizması sayesinde günümüzde kullanımını yaygınlaştırmıştır.

GraphQL, Facebook tarafından geliştirilmiş olup, veri erişimini optimize eden sorgu dili ve çalışma zamanı olarak tanımlanır. Geleneksel REST API'lerinin aksine, istemciler yalnızca ihtiyaç duydukları verileri sorgulayarak aşırı veri çekme (over-*fetching*) ve yetersiz veri çekme (*under-fetching*) sorunlarını minimize eder (Brito ve Valente, 2020). *GraphQL*, istemcilerin yalnızca ihtiyaç duyduğu verileri çekmesine izin vererek IoT ağlarında gereksiz veri trafiğini azaltır. IoT sistemlerinde farklı cihazlardan gelen verilerin yönetimi kritik bir konudur. *GraphQL*, tek bir uç nokta (*endpoint*) üzerinden çeşitli veri kaynaklarına erişmeyi mümkün kılar (Alzamzami ve

Azizah, 2022). *GraphQL*, JWT (*JSON Web Token*) ve kimlik doğrulama katmanları ile yetkilendirme süreçlerini kolaylaştırabilir. *GraphQL* tabanlı IoT sistemlerinde, REST API'lerine kıyasla %52'ye varan enerji tasarrufu sağlandığı gözlemlenmiştir (Khan ve Mian, 2020).

GraphQL API'leri, REST API'lerine kıyasla daha hızlı yanıt süresi sunar. Her ne kadar *GraphQL*, IoT sistemlerinde büyük avantajlar sağlasa da bazı dezavantajlar da içermektedir. *GraphQL* API, REST API'lerine kıyasla daha karmaşıktır ve optimize edilmesi gereklidir. Tüm sorguların tek bir uç noktaya yönlendirilmesi, sunucu tarafında ekstra işlem yükü oluşturabilir. *GraphQL'in* esnek yapısı, yetkilendirme kontrolleri iyi uygulanmazsa saldırganların hassas verilere erişmesine yol açabilir. Geleneksel REST mimarisine kıyasla, istemcinin yalnızca ihtiyacı olan verileri çekmesine olanak tanıması, gereksiz veri transferini azaltarak ağ kaynaklarının daha verimli kullanılmasını sağlar. Bu, büyük ölçekli IoT ağlarında özellikle pil ömrü sınırlı olan cihazlar için önemli bir avantajdır. Ancak, bu mimarinin etkin şekilde uygulanabilmesi için bazı performans iyileştirmelerinin ve güvenlik önlemlerinin geliştirilmesi gerekilebilir.

Sonuç olarak, *GraphQL*, IoT ekosisteminde veri akışını daha verimli hale getiren bir teknoloji olarak öne çıkmış olsa da mevcut sistemlerde tamamen REST'in yerini alması kısa sürede olağan görülmeyebilir. Bunun yerine, özellikle güvenlik gereksinimleri ve ölçeklenebilirlik açısından *GraphQL'in* avantajlarını REST'in sağlam yapısıyla birleştiren çözümler geliştirmek daha iyi olabilir. Gelecekte yapılacak çalışmalar, *GraphQL'in* performans ve güvenlik konularındaki eksikliklerini gidererek IoT için daha yaygın hale gelmesini sağlayabilir.

3.6.4 gRPC

gRPC, *Google* tarafından 2015 yılında açık kaynak olarak sunulan, yüksek performanslı bir Uzaktan Prosedür Çağrısı (RPC) sistemidir. HTTP/2 üzerine geliştirilen gRPC teknolojisi, bağlantılı ve verimli iletişim sağlamak için protokol arabellekleri (*Protocol Buffers-Protobuf*) kullanır. gRPC, istemci ve sunucu arasında bir kanal oluşturarak, çağrıların doğrudan uzaktan yürütülmesine olanak tanır (Kamiński ve diğerleri, 2022). Bu yapı sayesinde, istemciler REST gibi geleneksel

HTTP bazlı sistemlerden farklı olarak, gereksiz yükleri taşımadan direkt olarak belirli işlemleri çağırabilir. gRPC, dört temel iletişim modelini destekler (Wang ve Yi, 2022).

Unary RPC: İstemci, sunucuya bir istek gönderir ve tek bir yanıt alır.

Server Streaming RPC: İstemci sunucuya tek bir istek gönderir, ancak sunucu bir dizi yanıt gönderir.

Client Streaming RPC: İstemci, sunucuya birden fazla veri paketi gönderebilir, ancak sunucu yalnızca tek bir yanıt döndürür.

Bi-directional Streaming RPC: İstemci ve sunucu birbirlerine aynı anda mesaj gönderebilir ve bu mesajlar akış halinde işlenebilir.

gRPC, istemci ile sunucu arasındaki iletişimi hızlı, güvenilir ve verimli bir şekilde sağlayan modern bir *framework'tür*. Uzaktan Prosedür Çağrısı modelini temel alan bu yapı, istemcinin doğrudan sunucu tarafındaki fonksiyonları ve metotları çağırmasına olanak tanır.

İstemci ve sunucu arasındaki iletişim, önceden tanımlanmış bir sözleşme (*contract*) üzerinden gerçekleştirilir. Bu sözleşme, istemcinin sunucudan çağırabileceği prosedürleri, bu prosedürlere iletilebilecek parametreleri ve döndürülecek veri türlerini belirler. Bu yapı sayesinde, istemci ve sunucu bağımsız olarak geliştirilebilir ve farklı programlama dillerinde çalıştırılabilir.

Sözleşmeye dayalı bu sistem, istemci ve sunucunun otomatik olarak kendi dillerinde kod üretmesini sağlar. Oluşturulan bu kod, mesaj serileştirme, ağ çağrıları ve hata yönetimi gibi tüm iletişim ayrıntılarını kapsayarak, geliştiricilere hızlı ve güvenli bir entegrasyon sağlar. Bu yaklaşım, gRPC'yi *mikroservis* mimarileri, IoT sistemleri ve yüksek performans gerektiren uygulamalar için güçlü bir çözüm haline getirir.

REST ile gRPC'nin karşılaştırılması, veri aktarımı, performans ve *mikroservis* entegrasyonu açısından farklılıkları ortaya koymaktadır. Performans açısından gRPC, REST'ten daha hızlıdır, çünkü HTTP/2 kullanarak tek bir bağlantı üzerinden çoklu istekleri destekler (Bolanowski ve diğerleri, 2022).

JSON yerine *Protobuf* kullanması, gRPC'nin daha az veri boyutuyla işlem yapmasını sağlar. Bu da özellikle IoT cihazlarında ve kısıtlı bant genişliği olan ağlarda avantajlıdır (Nieman ve Sajal, 2023). REST, *stateless* bir yapıya sahipken, gRPC *persistent* (kalıcı) bağlantılar kurarak daha verimli bir iletişim sağlar. REST API'ler daha yaygın kullanımına sahip olup, HTTP'nin sunduğu geniş desteğe sahip olduğundan, web uygulamalarında hala büyük bir avantaja sahiptir.

Tablo 3.1: RESTful API,GraphQL ve gRPC karşılaştırması

Özellik	RESTful API	GraphQL	gRPC
Veri Alışverişi Modeli	HTTP tabanlı, her işlem için farklı uç noktalar kullanılır	Tek bir uç nokta üzerinden sorgu tabanlı veri çekme	HTTP/2 tabanlı, istemci doğrudan metot çağrısı yapar
Verimlilik	Veri aktarımı sırasında fazladan karakter çekebilir	İhtiyaç duyulan veriler çekildiği için daha verimli	Daha düşük veri yükü sayesinde yüksek verimlilik
İstemci Esnekliği	Sınırlıdır, istemci fazla veri çekebilir veya eksik veri alabilir	İstemci ihtiyacı olan verileri sorgulayabilir	İstemci ve sunucu arasındaki sözleşmeye göre çalışır
Veri Formatı	Genellikle JSON	JSON (metin tabanlı)	*Protobuf* (ikili format)
Ön bellekleme Desteği	Çok güçlüdür, HTTP ön bellekleme desteği vardır	Dahili ön bellekleme desteği zayıftır	Ön bellekleme desteği sınırlıdır
Gecikme Süresi	Yüksektir	Daha düşük, ancak kompleks sorgular gecikmeye sebep olur	En düşük gecikme süresine sahiptir
Güvenlik	Kimlik doğrulama JWT, OAuth vb. ile sağlanabilir	Yetkilendirme mekanizmaları ek olarak uygulanmalıdır	Dahili güvenlik protokolleri güçlüdür
IoT İçin Uygunluk	Orta seviyede uygundur, büyük veri iletimi gereken IoT cihazları için verimsiz olabilir	Bazı IoT sistemleri için uygundur, ancak sürekli veri akışı gereken durumlarda yetersiz kalabilir	IoT için en uygun seçenektir, düşük bant genişliği ile yüksek performans sunar
IoT Kullanım Alanları	Web servisleri, IoT cihazlarının bulut sistemlerine bağlanması	Veri odaklı IoT API'leri, cihaz yönetimi, akıllı şehir uygulamaları	Gerçek zamanlı IoT iletişimi, sensör veri aktarımı, akıllı şehirler, sağlık IoT sistemleri

Tablo 3.1, *RESTful* API, *GraphQL* ve *gRPC'nin* IoT sistemlerindeki performansını, avantajlarını ve dezavantajlarını kolay bir şekilde göstermek için

hazırlanmıştır. IoT sistemlerinde kullanılacak API seçimi uygulamanın gereksinimlerine ve performans hedeflerine göre yapılmalıdır.

Standart *web* servisleri ve cihaz yönetimi gerekiyorsa REST tercih edilebilir. Veri odaklı ve istemci tarafından özelleştirilebilir sorgular gerektiren durumlarda *GraphQL* seçilebilir. Gerçek zamanlı, düşük gecikmeli ve yüksek performans istenen IoT sistemlerinde gRPC en uygun tercih olacaktır. Ancak, gRPC'nin ve *Protobuf'un* tarayıcı desteğinin sınırlı olması, web tabanlı projelerde REST'in daha yaygın kullanılmasını sağlamaktadır. Tablo 3.1, farklı API yaklaşımlarını kıyaslayarak IoT geliştiricileri ve sistem mimarları için karşılaştırmalı bir bakış açısı sunmayı amaçlamaktadır. Bu kıyaslama, karar verme sürecinde yol gösterici nitelikte olabilir.

4. RASPBERRY PI İLE KESİNTİSİZ GÜÇ YÖNETİMİ VE BİLDİRİM SİSTEMİ

Günümüzde elektrik kesintileri, ev ve işletmelerdeki elektronik cihazlar için büyük bir sorun oluşturmaktadır. Kullanıcılar seyahat halindeyken veya evde olmadıklarında gerçekleşen kesintiler, özellikle beyaz eşya ve diğer kritik cihazların zarar görmesine neden olabilir.

Bu sistemin temel sorunu, kullanıcılar elektrik kesintisini anında fark edemezler, kesinti nedeniyle buzdolabındaki yiyecekler bozulabilir, akıllı cihazlar kapanabilir. Bu sebeple müdahale geç kalındığında kötü bir senaryo ile karşı karşıya kalınabilir.

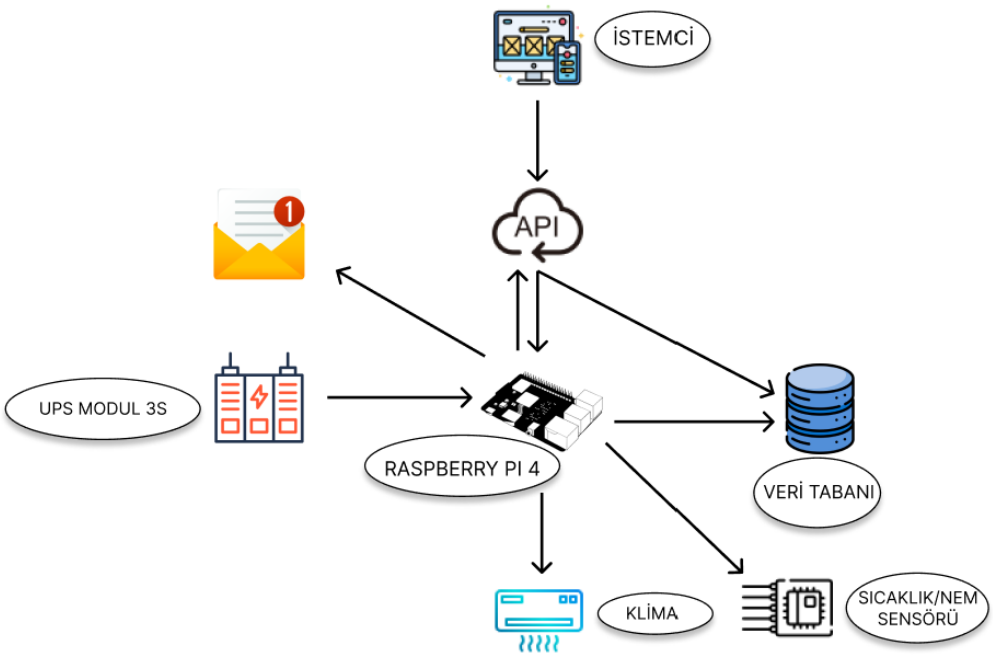
Mevcut UPS sistemleri, kesinti durumunda cihazlara enerji sağlasa da uzaktan bilgilendirme yapmaz ve gücün ne kadar süre devam edebileceği hakkında bilgi sunmaz.

Bu sorunu çözmek için *Raspberry Pi* tabanlı bir Kesintisiz Güç Yönetim ve Bildirim Sistemi geliştirildi. Bu sistem, UPS'in batarya veya adaptör modunda olup olmadığını tespit ederek, elektrik kesintisi durumunda kullanıcıya anında bildirim göndermektedir. Bu sistemde, Hareketli Ortalama Filtresi ve *Medyan* Filtresi kullanarak hibrit bir yaklaşım ile sensör verilerindeki anlık değişimleri dengeleyip, güç tüketimi ölçümlerini daha kararlı hale getirilir. Tüketilen güç miktarını hesaplayarak kullanıcıya detaylı bir enerji raporu sunar ve gerçek zamanlı olarak bataryanın mevcut şarj durumunu izler ve tüketim verilini kayıt altına alır.

Ek olarak akıllı olmayan bir klima, akıllı priz kullanımıyla uzaktan kontrol edilebilir hale gelmiştir. Kullanıcılar *web* arayüz üzerinden klimayı uzaktan açıp kapatabilir, enerji tüketimini izleyebilir, elektrik kesintisinin olup olmadığını kontrol edebilir ayrıca uzaktan *Raspberry Pi'nin* bulunduğu ortamın mevcut sıcaklığını gözlemleyebilir. Kullanıcının belirlediği sıcaklık eşiği ile klimanın otomatik olarak açılıp kapanması mümkün hale gelir.

4.1 Sistemin Tasarımı ve Genel Mimarisi

Raspberry Pi, UPS Modül 3S, DHT11 sensör, akıllı priz ve haberleşme protokolleri kullanarak gerçekleştirilen gerçek zamanlı bir projedir. Genel mimari, verinin *Raspberry Pi* tarafından işlenip bir *Flask* API aracılığıyla C# tarafına iletilmesini ve kullanıcı arayüzüne aktarılmasını kapsar.



Şekil 4.1: Sistem mimarisi ve veri akışı

Şekil 4.1'de gösterilen sistem mimarisinde veri akışının merkezi *Raspberry Pi* 4 cihazıdır. UPS *Modül 3S,* elektrik kesintisini kontrol eder ve *Raspberry Pi'ye* güç durumu verisini iletir. Elektrik kesintisi algılandığında, sistem SMTP haberleşme protokolü aracılığıyla kullanıcının e-posta adresine otomatik olarak bir bilgilendirme mesajı gönderir. Bu süreçte, *Raspberry Pi* UPS modülünden güç çekmeye devam eder ve elektrik tekrar geldiğinde UPS modülünden alınan ölçüm verisi yeniden *Raspberry Pi'ye* aktarılır. Sistem, kullanıcının e-posta adresine "Elektrik geri geldi" mesajını göndererek bilgilendirme yapar. Böylece kullanıcı, evde, işte, tatilde veya seyahat halindeyken elektrik bağlantı durumunu gerçek zamanlı olarak takip edebilir.

UPS Modülü, sadece bir güç kaynağı olarak değil, bir sensör gibi de davranarak I²C veri hattı üzerinden güç tüketimi, çekilen akım, voltaj ve batarya durum bilgilerini

Raspberry Pi'ye iletmektedir. Bu veriler işlenerek, kullanıcının anlık güç tüketimini ve toplam kWh bazında enerji kullanımını görmesine olanak tanır. Ayrıca, UPS Modülü modeme güç sağlayarak, bölgesel bir elektrik kesintisi olmadığı sürece *Raspberry Pi'nin* internet bağlantısını sürdürebilmesini garanti eder.

Tez çalışmasında kullanılan akıllı priz, akıllı olmayan bir klimayı uzaktan kontrol edilebilir hale getirmektedir. Kullanıcı, *web* arayüzü üzerinden klimayı açıp kapatabilir ve klimanın enerji tüketimini takip edebilir.

Ortam sıcaklığı ve nem değerlerinin ölçülmesi için DHT11 sensörü kullanılmıştır. Kullanıcı, belirlediği bir sıcaklık eşik değeri (örneğin 27°C) girdiğinde, sistem bu değeri referans alarak klimayı otomatik olarak kontrol eder. Eğer ortam sıcaklığı 27°C'nin üzerine çıkarsa, klima otomatik olarak açılır; sıcaklık bu eşik değerin altına düştüğünde ise klima kapanır. Böylece kullanıcı, manuel müdahale gerekmeksizin ortam sıcaklığını belirlenen seviyede tutabilir ve enerji tasarrufu sağlayabilir.

Tüm bu veriler veri tabanına kaydedilerek, kullanıcının geçmiş tüketim verilerini analiz etmesine olanak tanır. Böylece sistem, elektrik tüketiminin daha verimli yönetilmesine katkı sağlayan bir enerji izleme ve kontrol mekanizması sunar.

4.2 Kullanılan Teknolojiler ve Tasarım Desenleri

Bu tez çalışmasında, gerçek zamanlı veri akışı, sensör yönetimi ve uzaktan erişim gibi kritik işlemleri yönetebilmek için çeşitli yazılım ve donanım teknolojileri entegre edilmiştir. Sistem *Raspberry Pi* üzerinde çalışan bir *Flask* tabanlı *RESTful* API ile sensör verilerini toplar ve işler. İşlenen veriler, C# *Web* API aracılığıyla istemciye iletilir, böylece kullanıcılar gerçek zamanlı olarak sistem verilerini izleyebilir ve uzaktan kontrol sağlayabilir.

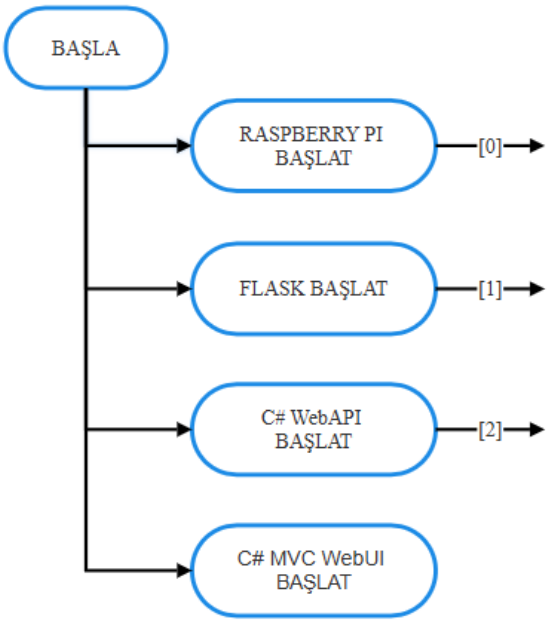
Veri tabanı yönetimi için *SQL Server* ve *Entity Framework Core* kullanılmış, iş mantığını daha modüler ve ölçeklenebilir hale getirmek amacıyla *MediatR* ve *Repository Design Pattern* gibi tasarım desenleri uygulanmıştır.

Ayrıca, UPS ve sensör verilerinin doğruluğunu artırmak için *Multi-Stage Filtering Algorithm (MSFA)* kullanılmıştır. *Raspberry Pi'nin* donanım seviyesinde yönetimi için *gpiod* kütüphanesi tercih edilmiştir, böylece donanım erişimi optimize edilmiş ve performans kayıpları en aza indirilmiştir.

Bu teknolojik kombinasyon, sistemin hem gerçek zamanlı hem de yüksek doğrulukta çalışmasını sağlarken, *RESTful* API yaklaşımıyla veri alışverişini standart ve güvenilir hale getirmektedir.

4.2.1 Yazılım Teknolojileri

Bu projede, sensör verilerinin gerçek zamanlı işlenmesi, veri güvenilirliğinin sağlanması ve sistem bileşenleri arasında hızlı iletişim kurulması için çeşitli yazılım teknolojileri kullanılmıştır. Proje kapsamında *RESTful* API mimarisi kullanılarak esnek ve ölçeklenebilir bir sistem oluşturulmuştur. Sistemin temel yazılım bileşenleri, *Flask, C# Web* API, *SQL Server, Entity Framework Core, MediatR, Repository Design Pattern* ve *Multi-Stage Filtering Algorithm (MSFA)* gibi teknolojilerden oluşmaktadır. Şekil 4.2' de akış diyagramı gösterilmiştir.



Şekil 4.2: Sistem başlatma akış diyagramı

4.2.1.1 RESTful API ve Veri Akışı

RESTful API, istemci ve sunucu arasındaki iletişimi sağlayan bir *web* servis mimarisidir. *RESTful* API'nin tercih edilmesinin sebepleri ise, platform bağımsızlığını sağlaması, ölçeklenebilirlik, standart veri yapısı ve esnek gibi avantajlar sunmasıdır.

Bağımsızlık, herhangi bir platform veya teknolojiye bağımlı olmadan geliştirilebilir. Bu çalışma kapsamında istemci tarafı MVC *Web* UI kullanılarak geliştirilmiştir. Ancak bağımsızlık ilkesi sayesinde, gelecekte *React, Angular* veya farklı bir *frontend* teknolojisine geçiş yapmak oldukça kolay olacaktır. API'nin ilerleyen teknolojilerle tekrar kullanılabilir olması, sistemin sürdürülebilirliğini artırmaktadır.

Ölçeklenebilirlik, yeni istemciler veya servisler eklendiğinde, mevcut sistemin mimarisi bozulmadan genişletilebilir. Örneğin, ilerleyen süreçte mobil uygulama entegrasyonu veya farklı cihazlardan veri erişimi gerektiğinde, mevcut *RESTful* API mimarisi ile kolayca uyarlanabilir.

Standart veri yapısı olan JSON formatını kullanarak verileri istemcilere sunar. JSON formatı hem geliştiriciler hem de sistemler tarafından kolayca işlenebilir ve hafif bir veri yapısı sunduğu için hızlı ve güvenilir bir iletişim sağlar.

Esneklik özelliği ise HTTP protokolü üzerinden çalıştığı için her platformda rahatlıkla kullanılabilir. İstemci tarafında herhangi bir ek yapılandırma gerektirmeden, *GET, POST, PUT* ve *DELETE* gibi standart HTTP istekleriyle veriler işlenebilir. Bu esneklik özelliği ile API'nin *web, mobil* ve IoT cihazlarıyla entegrasyonunu kolaylaştırmaktadır.

4.2.1.2 Flask Web Çatısı ve Raspberry Pi ile Kullanımı

"Flask; Web uygulamalarının geliştirilmesini sağlayan Python ile yazılmış bir micro framework'tür" (Serhat Kağan Şahin ve Erdal Delebe, 2021). *Framework*, belirli bir yazılım geliştirme alanında standart bir yapı ve kurallar sunan bir yazılım çerçevesi ya da çatısıdır. Geliştiricilere önceden tanımlanmış bir iskelet yapı sunarak belirli

kurallar çerçevesinde yazılımın geliştirilmesi için yönlendirme sağlar. Kütüphane (*library*) ise, belirli işlevleri yerine getiren bağımsız kod parçalarından oluşan bir bileşendir ve geliştirici tarafından ihtiyaç duyulduğunda çağrılarak kullanım sağlar. *NumPy* bir kütüphane iken *Flask* bir *frameworktür*. *Framework*, çalışma akışını kendisi yönetir, geliştirici belirli noktalarda müdahale eder ancak kütüphaneler geliştirici tarafından çağrılarak yönetilir, kontrol tamamen yazılımcıya aittir. *Flask* bir *framework* olduğu için çalıştırıldığında kendi sunucusunu başlatır ve gelen HTTP isteklerini yönlendirir. Geliştirici sadece belirli rotaları (*routes*) ve iş mantığını tanımlar ancak uygulamanın nasıl başlatılacağını ve yönetileceğini *Flask* belirler (Armin Ronacher, 2017).

RESTful API geliştirme ve mikro servis mimarileri için kullanılan *Flask* teknolojisi minimal, modüler ve genişletilebilir bir yapıya sahiptir. Bu yapısı sayesinde düşük kaynak tüketen gömülü sistemler ve hızlı API geliştirme süreçleri için tercih edilebilir.

Bu projede, *Flask* tabanlı bir *RESTful* API geliştirilerek *Raspberry Pi* üzerindeki sensörlerden veri alınması, bu verilerin işlenmesi ve istemciye sunulması sağlanmıştır. *Flask*, istemci ile sunucu arasında HTTP tabanlı bir veri alışverişi mekanizması oluşturarak sensör verilerinin gerçek zamanlı olarak yönetilmesine olanak tanımaktadır. *Flask frameworkü* aynı zamanda *Jinja2* şablon desteği sunarak *web* arayüzü ile kullanımını kolaylaştırır (Armin Ronacher, 2017).

Flask, Python'un pip paket yöneticisi kullanarak *pip install flask* komutu ile kolayca kurulabilir. *Flask* yapısı gereği içinde bazı kütüphaneleri barındırır. Örneğin, *Werkzeug* ve *Jinja* (Armin Ronacher, 2017). *Python'da flask frameworkü, from flask import Flask* komutu ile kullanılabilir. Bu şekilde *Flask* modülü projeye dahil edilerek bir uygulama nesnesi oluşturulur.

Flask'ın Route tanımlaması belirli bir URL'ye gelen http isteklerinin hangi fonksiyon tarafından işleneceği belirlenir. Örneğin *'/get-temperature'*. *Flask, Python'da* yerleşik olarak bulunan sözlük (*dictionary*) veri yapısını kullanarak, gelen ve dönen verileri JSON formatına otomatik olarak dönüştürmektedir. *Flask application/json* başlığını ekleyerek istemciye uygun formatta yanıt döndürmektedir. JSON formatında yanıt döndürme süreci*, Flask'in jsonify()* metodu ile daha optimize

bir hale getirilebilir. Bu metot, JSON yanıtlarının standart başlıklarla oluşturulmasını sağlayarak istemci tarafındaki veri işleme sürecini kolaylaştırmaktadır.

Flask uygulaması, ağ üzerinden erişilebilir olacak şekilde bir IP adresi ve port numarası atanarak çalıştırılır. *Flask* uygulaması, temel bir *Flask* nesnesi üzerinden başlatılmaktadır. *Python'da* özel bir değişken olan *__name__*, uygulamanın hangi modülde çalıştığını belirtmek için kullanılmaktadır.

Flask, Python'daki dekoratör (*decorator*) yapısını kullanarak URL yönlendirme işlemlerini gerçekleştirmektedir. Dekoratörler, bir fonksiyona ek özellikler kazandıran *Python* mekanizmalarıdır ve *Flask'in* çalışma mantığında önemli bir yer tutmaktadır.

Flask'in @app.route() dekoratörü kullanılarak, istemciden gelen HTTP isteklerinin belirli URL yollarına yönlendirilmesi sağlanmıştır. Bu kapsamda, HTTP *GET* metodunu kullanan bir istemcinin çağrısı ile sensör verilerine erişim sağlanmaktadır.

```
pi@raspberrypi:~ $ cat /etc/systemd/system/flask_app.service
[Unit]
Description=Raspberry Pi Flask UPS Service
After=network.target

[Service]
ExecStart=/home/pi/rasp_app/Penv/bin/python /home/pi/rasp_app/app.py
WorkingDirectory=/home/pi/rasp_app
StandardOutput=append:/var/log/flask_app.log
StandardError=append:/var/log/flask_app_error.log
Restart=always
User=pi

[Install]
WantedBy=multi-user.target
```

Şekil 4.3: Raspberry Pi Flask servis yapılandırması

Flask, Raspberry Pi üzerinde sürekli çalışmasını sağlamak için *systemd* servis yöneticisi ile arka planda çalışacak şekilde yapılandırılabilir. *Raspberry Pi'de Flask* API'nin arka planda servis olarak çalışması için */etc/systemd/system/flask_app.service* dosyası oluşturulmuştur. Şekil 4.3**Şekil 4.3**'te *Flask* servis yapılandırması gösterilmektedir.

Flask API'yi *Raspberry Pi* üzerinde her açılışta otomatik olarak başlatmak için şu *systemctl* komutları kullanılır.

sudo systemctl daemon-reload
sudo systemctl enable flask_app
sudo systemctl start flask_app

Komutları sırasıyla girilerek *Flask* API *Raspberry Pi* her açıldığında otomatik olarak başlatılacak ve sürekli çalışmasını sağlayacaktır. Servisi durdurmak için

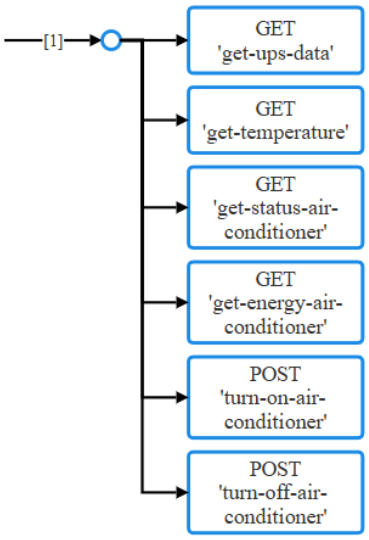
sudo systemctl stop flask_app.service

Servisin aktif olup olmadığını görmek için

sudo systemctl status flask_app.service

Komutu girilir. Geliştiriciler servisin aktif olduğu aralıkta kod düzenlemesi yaparsa, *sudo systemctl restart flask_app.service* komutunu girerek *reset* atmaları gerekebilir.

Flask, hafif ve modüler bir web çerçevesi olmasına rağmen, bazı dezavantajlara sahiptir, alternatif olarak *FastAPI, Django Rest fremeworkleri* kullanılabilir. Bu tez kapsamında minimal ve hızlı olması nedeniyle *Flask* tercih edilmiştir, daha büyük ölçekli projelerde *FastAPI* veya *Django Rest frameworkleri* kullanılabilir.



Şekil 4.4: Flask API istek akış diyagramı

Şekil 4.4, *Flask* tabanlı API servisinin istemciler tarafından çağrılabilen *GET* ve *POST* isteklerini göstermektedir. Bu API, UPS verilerini almak, sıcaklık ölçmek, klimanın durumunu ve enerji tüketimini sorgulamak, klimayı açıp kapamak gibi işlevleri desteklemektedir. *GET* metotları, belirli verileri sorgulamak için kullanılırken, *POST* metotları, sistem üzerinde değişiklik yapmak için kullanılmaktadır.

Bu API yapısı, *Raspberry Pi* üzerinde çalışan ve bağlı cihazlardan veri toplayan, ayrıca klima gibi sistemleri uzaktan yönetebilen bir *mikroservis* mimarisinin parçasıdır. *Flask framework'ü*, hafif ve hızlı bir *RESTful* API geliştirme süreci sağladığından, bu çalışmada kullanılmıştır.

4.2.1.3 C# Web API ve MVC Web Arayüzü

Bu çalışmada, veri erişim katmanını daha esnek ve yönetilebilir hale getirmek amacıyla *Entity Framework Core* (*EF Core*) ve *Repository Design Pattern* kullanılmıştır. *EF Core,* Microsoft tarafından geliştirilen ve *Object-Relational Mapping* (ORM) yaklaşımını benimseyen modern bir veri erişim kütüphanesidir. Bu yapı sayesinde, ilişkisel veri tabanları ile çalışırken SQL sorguları yazmaya gerek kalmadan, nesne yönelimli bir şekilde veri işlemleri gerçekleştirilebilmektedir. EF *Core'un* LINQ (*Language Integrated Query*) desteği, verilerin dinamik bir şekilde işlenmesini ve sorguların daha okunaklı hale gelmesini sağlamaktadır.

EF *Core*, bu çalışmada *Fluent* API ve *Data Annotations* kullanılarak konfigüre edilmiştir. *Code-First* yaklaşımı benimsenmiş olup, veri tabanı şeması doğrudan *C#* sınıflarından türetilerek oluşturulmuştur. Bu sayede, veri modeli üzerinde yapılan değişiklikler doğrudan veri tabanına yansıtılabilmekte ve yönetilebilirlik artırılmaktadır.

Veri erişim katmanının daha modüler ve sürdürülebilir hale getirilmesi amacıyla *Repository Design Pattern* uygulanmıştır. *Repository Pattern*, veri erişimini soyutlayarak, iş mantığı katmanının doğrudan EF *Core* ile etkileşime girmesini engeller ve veri işlemlerinin merkezi bir yapı üzerinden gerçekleştirilmesini sağlar. Bu

kapsamda, her bir veri modeli için ayrı bir *repository* sınıfı oluşturulmuş ve temel CRUD işlemleri tek bir yapı üzerinden yönetilmiştir.

Çalışmada *Generic Repository Pattern* kullanılarak, tekrar eden kod yazımının önüne geçilmiş ve veri erişim süreçleri optimize edilmiştir. Örneğin, tüm veri tabanı işlemlerini yöneten bir *GenericRepository<T>* sınıfı oluşturulmuş ve farklı modeller için ortak CRUD operasyonları bu yapı üzerinden sağlanmıştır. Böylece, her yeni model için ayrı veri erişim kodları yazmaya gerek kalmadan, merkezi bir yapı üzerinden yönetim sağlanmıştır.

EF *Core* ve *Repository Pattern* kullanımı sayesinde veri erişim sürecinde sürdürülebilirlik sağlanmış, bağımlılıklar azaltılmış ve sistemin ölçeklenebilirliği artırılmıştır. Bu yapı, ilerleyen dönemlerde farklı veri tabanları ile çalışma gerekliliği doğduğunda veya iş mantığında değişiklikler yapıldığında, minimum kod değişikliği ile sistemin uyarlanabilmesine olanak tanımaktadır.

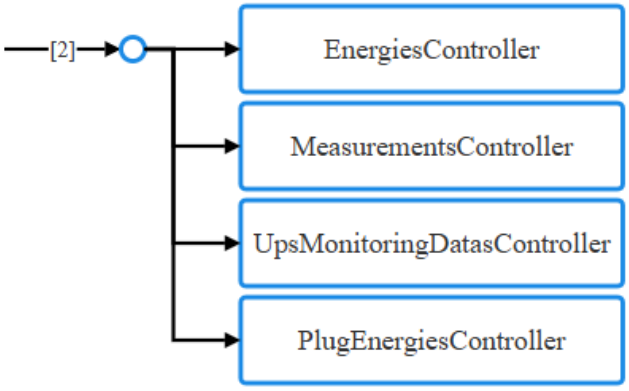
Bu çalışmada, iş mantığının daha modüler, okunabilir ve yönetilebilir hale getirilmesi amacıyla *MediatR* kütüphanesi ve CQRS mimari deseni kullanılmıştır. CQRS, veri yazma (komut) ve okuma (sorgu) işlemlerini birbirinden ayıran bir yazılım mimarisidir. Geleneksel veri yönetim sistemlerinde aynı servis veya *repository* üzerinden hem veri yazma hem de veri okuma işlemleri gerçekleştirilirken, CQRS ile bu işlemler ayrı katmanlar halinde ele alınmaktadır. Böylece, okuma işlemleri (*Query*) ve yazma işlemleri (*Command*) farklı iş mantıkları üzerinden yönetilir, bu da daha ölçeklenebilir ve performanslı bir sistem oluşturulmasını sağlar.

CQRS mimarisinde *Command* (Komut) ve *Query* (Sorgu) olmak üzere iki temel yapı bulunmaktadır: *Command* (komut) veri yazma, güncelleme ve silme işlemlerini gerçekleştiren işlemleri ifade eder. Komut işlemleri genellikle yan etkili (*side effect*) işlemler olup sistemde bir değişikliğe neden olur. Örneğin, bir kullanıcının enerji tüketim verisini güncellemesi *Command* işlemidir. *Query* (sorgu) ise veri okuma işlemlerini temsil eder ve yalnızca sistemden veri çekmek için kullanılır. *Query* işlemleri yan etkisizdir (*side effect-free*), yani veri tabanında herhangi bir değişiklik yapmazlar. Örneğin, kullanıcının mevcut güç tüketimini sorgulaması veya UPS batarya durumunu kontrol etmesi bir *Query* işlemidir.

CQRS sayesinde okuma ve yazma işlemlerinin birbirinden ayrılması, büyük ölçekli sistemlerde veri tutarlılığını artırmakta, bakım süreçlerini kolaylaştırmakta ve performansı optimize etmektedir. CQRS mimarisinin uygulanmasını kolaylaştırmak ve bağımlılıkları azaltmak amacıyla *MediatR* kütüphanesi kullanılmıştır. *MediatR, Mediator Design Pattern'ı* temel alarak, bileşenler arasındaki bağımlılığı azaltan ve modülerliği artıran bir mesaj yönlendirme kütüphanesidir. *MediatR, Controller'lar* ile iş mantığı katmanı arasındaki doğrudan bağımlılığı kaldırarak, işlemlerin merkezi bir *mediator* aracılığıyla yönlendirilmesini sağlar.

Bu çalışmada, CQRS ve *MediatR* entegrasyonu *Query* işlemleri için *IRequest<T>* arayüzü kullanılarak veri okuma talepleri modellenmiştir. *Command* işlemleri için *IRequest* arayüzü kullanılarak veri yazma, güncelleme ve silme işlemleri modellenmiştir. *Handler* sınıfları, gelen talepleri işleyerek ilgili servis veya *repository* üzerinden gerekli işlemleri gerçekleştirmiştir. Controller sınıfları, *MediatR* üzerinden uygun *Query* veya *Command* sınıflarını çağırarak veri işleme sürecini yönetmiştir.

Bu yöntemler, özellikle büyük ölçekli sistemlerde kompleks iş mantıklarının yönetilmesi, bağımsız servislerin oluşturulması ve modüler mimarinin sağlanması açısından önemli avantajlar sunabilir. Özellikle mikro servis mimarilerinde CQRS ve *MediatR* kullanımı, sistemin ölçeklenebilirliğini artırarak bağımsız bileşenlerin geliştirilmesine olanak tanıyabilir. Bu nedenle, CQRS ve *MediatR'ın* sağladığı avantajlar göz önüne alındığında, bu mimari yaklaşım büyük ölçekli sistemlerde veri yönetimi süreçlerinin daha etkili bir şekilde gerçekleştirilmesine önemli katkılar sağlayabilir.

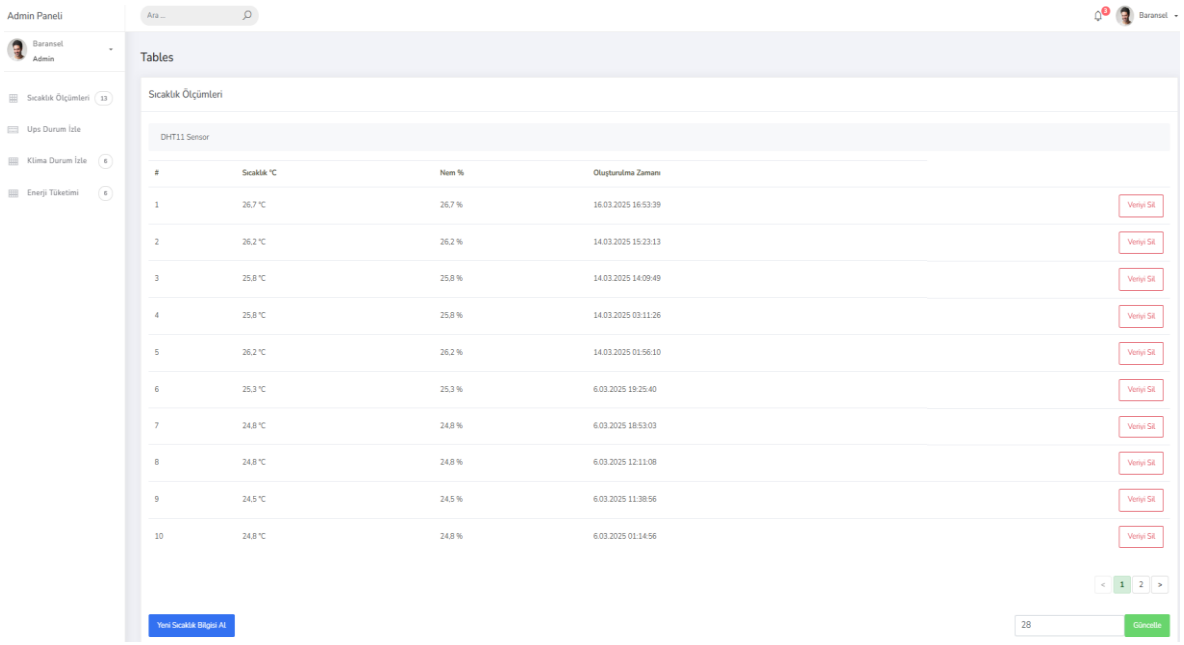


Şekil 4.5: C# Web API Controller Yapısı

Şekil 4.5, C# *Web* API tarafındaki Controller yapısını göstermektedir. Bu yapıda, farklı verileri yönetmek için kullanılan dört farklı Controller tanımlanmıştır:

- *EnergiesController* → Enerji tüketimi ve ilgili verileri işler.
- *MeasurementsController* → Ölçüm verilerini almak ve yönetmek için kullanılır.
- UPS*MonitoringDatasController* → UPS izleme verilerini toplar ve sistemin durumunu kontrol eder.
- *PlugEnergiesController* → Elektrik prizlerindeki enerji tüketimini izler ve klimanın açılıp kapanmasını yönetir.

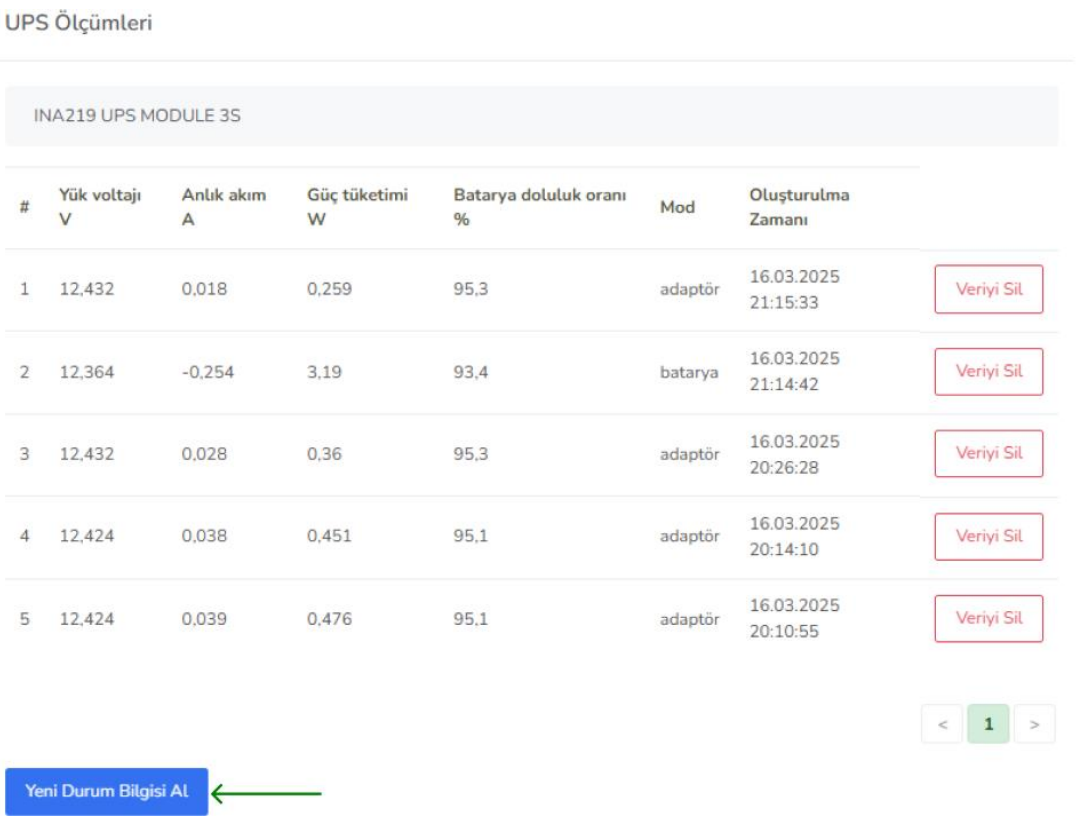
Bu yapı, *RESTful* API prensiplerine uygun olarak, istemcilerin belirli veri setlerine erişmesini ve kontrol etmesini sağlayan bir katmanlı mimari örneğidir. API'nin C# WebAPI ile geliştirilmesi, güçlü tip desteği, bağımlılık yönetimi ve genişletilebilirlik açısından avantaj sağlamaktadır. Şekil 4.6, C# MVC mimarisi kullanılarak geliştirilen Admin Paneli arayüzünü göstermektedir.



Şekil 4.6: C# MVC Web UI Sıcaklık Ölçümleri Yönetim arayüzü

Şekil 4.7: Sıcaklık eşiği

İstenilen sıcaklık eşiği bilgisi Şekil 4.7'de gösterilen alana girilerek klimanın otomatik olarak açılıp kapanması sağlanır. 28 °C eşiğini geçerse otomatik olarak klima açılacak aksi halde ortamın sıcaklığı 28 °C'den düşükse kapanacaktır.



Şekil 4.8: Gerçek zamanlı UPS ölçümleri

Şekil 4.8'de anlık olarak *Raspberry Pi'nin* enerji ölçümlerini takip edebileceğimiz MVC tarafında geliştirilmiş olan sayfa yapısını göstermektedir. Bu

sayfa üzerinden UPS modülüne ait güncel enerji tüketimi ve sistemin çalışma durumu takip edilebilir.

Tabloda yük voltajı *(Bus Voltage)*, anlık akım *(Current),* güç tüketimi (*Power*), batarya doluluk oranı (*Battery Percentage*), çalışma modu (*Mode*) ve oluşturulma zamanı (*Created At*) gibi önemli ölçümler gösterilmektedir.

Mod (*Mode*) sütunu, cihazın şu an adaptörle mi yoksa batarya ile mi çalıştığını belirtmektedir. Eğer bir elektrik kesintisi meydana gelirse, sistem batarya moduna geçtiğini algılar ve kullanıcıya e-posta ile bir bilgilendirme gönderilir.

Yük voltajı (*Bus Voltage*) sütunu, cihazın mevcut voltaj seviyesini gösterir. Bu değer, cihazın güç kaynağından aldığı voltajı izlemek için kullanılır. Adaptör bağlantısı aktif olduğunda, voltaj değeri genellikle sabit bir seviyede olur (örneğin, 12.4V – 12.6V aralığında). Eğer cihaz batarya moduna geçtiyse, voltaj seviyesi zamanla düşmeye başlar.

Anlık akım, cihazın o anda çektiği elektrik akımını mA cinsinden gösterir. Sistemin anlık yük durumuna bağlı olarak değişebilir. Daha fazla yük bindiğinde (örneğin, *Raspberry Pi'ye* bağlı ekstra donanımlar çalıştırıldığında), anlık akım artabilir. Adaptör bağlantısı varken genellikle sabit bir aralıkta seyreder (örneğin, 28mA – 120mA arasında değişebilir). Batarya modunda akım genellikle daha düşük olur, ancak batarya seviyesi azaldıkça akım dalgalanmaları olabilir.

UPS modülünde adaptör varken batarya şarj olur ve akım adaptörden bataryaya doğru akar (pozitif yönde). Elektrik kesildiğinde batarya artık güç kaynağına dönüşür. UPS modülü batarya ve adaptör arasında otomatik geçiş yapmak için *MOSFET* bazlı bir anahtarlama devresi kullanır.

MOSFET anahtarlama adaptör varken, *MOSFET'ler* adaptör yolunu açar ve batarya şarj olur. Akım adaptörden bataryaya doğru pozitif yönde akar. Elektrik kesilip batarya moduna geçtiğinde *MOSFET'ler* batarya yolunu açar ve adaptör devreden çıkar. Artık akım bataryadan çıkıp Raspberry Pi'ye doğru gider. Bu yön ters olduğu için akım negatif olarak ölçülür. *MOSFET* devresi akımın ters yönde akmasına sebep olur çünkü batarya artık güç kaynağıdır. UPS modülü batarya moduna geçtiğinde, akımın negatif olarak ölçülmesi, fiziksel olarak akımın ters aktığı anlamına gelmez.

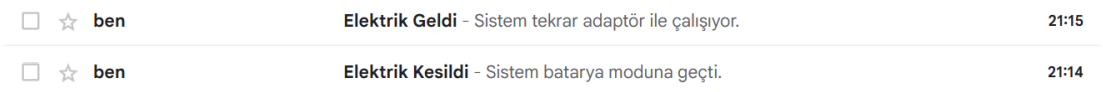
Akımın fiziksel yönü her zaman elektronların hareketine göre belirlenir. Ancak *INA219* gibi sensörler genellikle konvansiyonel akım yönüne (pozitif yük taşıyıcıların hareketine) göre ölçüm yapar. Bu durum, UPS devresinde *MOSFET* anahtarlama mekanizması ve *INA219'un* ölçüm referans noktası nedeniyle oluşmaktadır. Elektriksel olarak akımın yönü değişmese bile, UPS devresi *INA219*'a ters ölçüm yaptırdığı için negatif değerler çıkmaktadır.

Bu veri, sistem yükünü analiz etmek ve güç tüketimini optimize etmek için önemlidir.

Ayrıca, sayfa üzerinde "Yeni Durum Bilgisi Al" butonu bulunmaktadır. Kullanıcı bu butona tıklayarak anlık enerji ölçüm verilerini güncelleyebilir ve sistemin en güncel durumunu görüntüleyebilir.

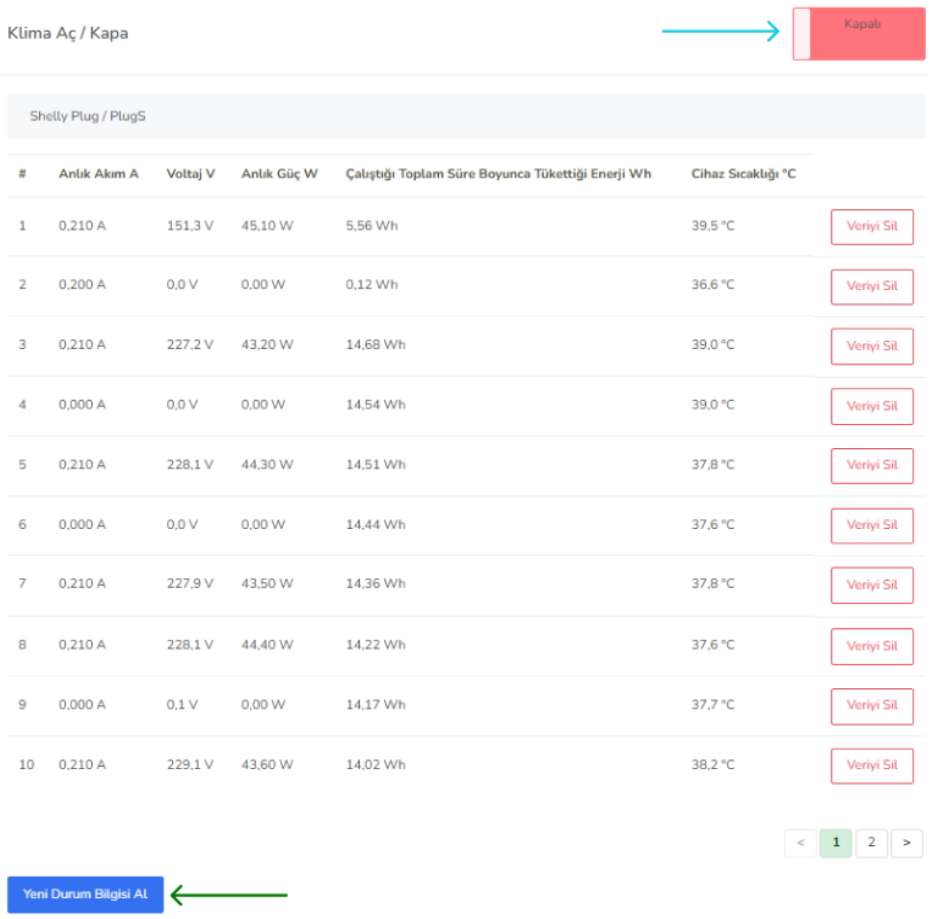
Veri tabanında kayıtlı ölçümler listelenmekte olup, "Veriyi Sil" butonu ile istenmeyen kayıtlar kullanıcı tarafından temizlenebilir.

Bu yapı sayesinde *Raspberry Pi'nin* güç tüketimi, adaptör veya batarya modunda çalışıp çalışmadığı ve anlık enerji ölçümleri izlenebilir, böylece sistemde herhangi bir anormallik olduğunda kullanıcı bilgilendirilebilir.



Şekil 4.9: Bildirim sisteminin e-posta üzerinden yapılması

Şekil 4.9'da bildirim sisteminin elektronik posta üzerinden teslim alınması gösterilmektedir.

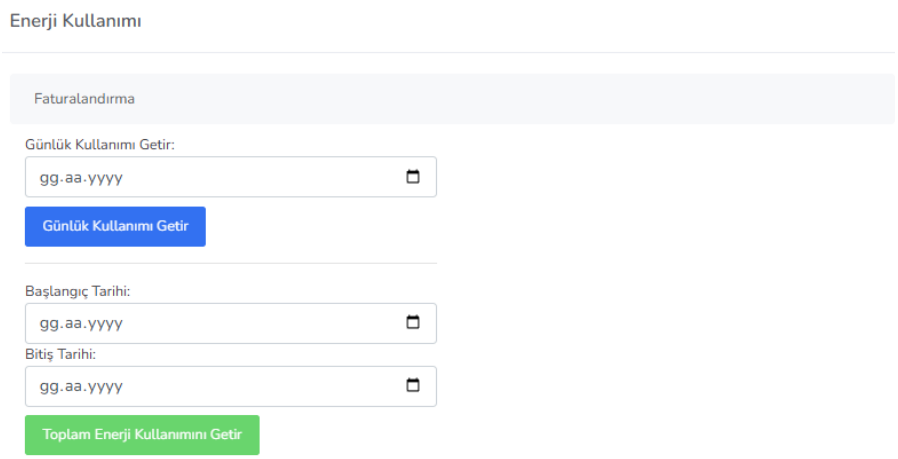


Şekil 4.10: Klima kontrol ve izleme sistemine ait web arayüzü

Şekil 4.10,klimanın enerji tüketimini anlık olarak izleyebileceğimiz bir takip panelini göstermektedir. *Shelly Plug / PlugS* akıllı priz kullanılarak klimanın çalışma durumu, çektiği akım (A), voltaj (V), güç tüketimi (W), toplam tüketim (Wh) ve cihaz sıcaklığı (°C) gibi veriler kaydedilmekte ve kullanıcıya sunulmaktadır.

Sağ üstteki "Kapalı" butonu, şu anda klimanın kapalı olduğunu gösteriyor. Kullanıcı, klimayı açıp kapamak için bu butonu kullanabilir. Açılan buton "Açık" yazısını alır ve klimanın açık çalıştığı anlaşılır. Sol alttaki "Yeni Durum Bilgisi Al" butonu ile kullanıcı, anlık enerji tüketimi verilerini güncellemek için tıklayarak gerekli bilgileri getirebilir. Klimanın açılması ya da kapanması durumuna göre, bu buton sayesinde güncel veriler ekrana yansıtılır. Bazı satırlarda anlık akım, voltaj ve güç tüketimi sıfır olarak görünüyor. Bu, klimanın o an kapalı olduğu veya güç çekmediği anlamına gelir. Örneğin, 2. ve 4. satırlarda voltaj 0.0V ve güç tüketimi 0.00W olarak

görünmektedir. Bazı satırlarda voltaj değeri 220V civarında ve güç tüketimi yüksek. Bu, klimanın aktif olarak çalıştığını ve enerji tükettiğini gösteriyor. Örneğin, 1. satırda 151.3V gerilim ve 45.10W güç tüketimi kaydedilmiş.



Şekil 4.11: Enerji kullanımı ve faturalandırma arayüzü

Şekil 4.11, enerji tüketiminin belirli tarihler arasında sorgulanmasını ve faturalandırılmasını sağlayan bir kullanıcı arayüzünü göstermektedir. Kullanıcılar günlük veya belirli bir zaman aralığında enerji tüketimini sorgulayarak toplam enerji harcamalarını analiz edebilir.

4.2.1.4 Multi-Stage Filtering Algorithm (MSFA)

Multi-Stage Filtering Algorithm (*MSFA*), gürültülü güç ölçümlerini daha güvenilir hale getirmek için birden fazla filtreleme aşaması kullanan bir algoritmadır. Özellikle enerji tüketimi ölçümlerinde ani sıçramaları ve gürültüyü azaltarak daha stabil ve güvenilir veriler elde etmeyi amaçlamaktadır.

Bu algoritma, Uyarlanabilir Hibrit Filtre yapısını kullanarak iki farklı filtreleme tekniğini birleştirir: *Medyan* Filtresi, Hareketli Ortalama (*Moving Average*) Filtresi. MSFA, güç ölçümlerini (Watt) işleyerek anlık ölçümlerdeki dalgalanmaları kontrol altına alır. Kod aşağıdaki temel adımlarla çalışır:

- Ham veri (güç ölçümü) algılandıktan sonra geçmiş veriler bir liste içinde saklanır.

- Eğer ani bir sıçrama algılanırsa, medyan filtresi uygulanır.
- Eğer ölçüm dalgalanıyorsa ancak büyük bir ani değişiklik yoksa, hareketli ortalama filtresi uygulanır.
- Yeterli veri yoksa, ölçüm ham haliyle bırakılır.

Medyan filtresi, son N adet ölçümün *medyanını* hesaplar ve bu değeri çıktı olarak döndürür.

$$y_t = median(x_{t-N}, \ldots, x_t) \tag{4.1}$$

Hareketli ortalama filtresi, ölçümlerde doğal olarak oluşan küçük dalgalanmaları azaltmak için kullanılır.

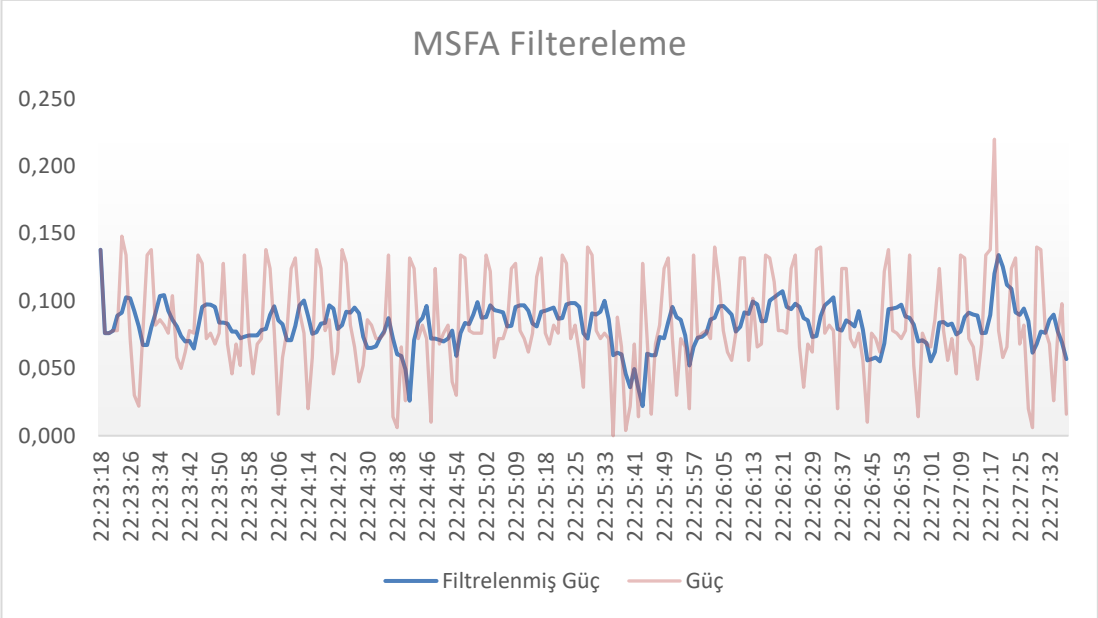
- Ani sıçrama yoksa, bu filtre devreye girer.
- Önceki N ölçümün ortalamasını alarak veriyi pürüzsüz hale getirir.
- Bu yöntem, özellikle güç tüketimi gibi dalgalı verilerde düzgün bir trend oluşturur.

$$y_t = \frac{1}{N}\sum_{i=t-N}^{t} x_i \tag{4.2}$$

Şekil 4.12'de çoklu hibrit sistemiyle geliştirilmiş pseudo kodu gösterilmiştir.

```
Başlat MultiStageFilteringAlgorithm (pencere_boyutu, gürültü_eşiği=0.1)
    pencere_boyutu = pencere_boyutu
    gürültü_eşiği = gürültü_eşiği
    geçmiş_veri = []

Fonksiyon Hareketli_Ortalama (veri)
    Eğer veri uzunluğu < pencere_boyutu ise:
        Dönüş yap (veri toplamı / veri uzunluğu)
    Aksi halde:
        Dönüş yap (son pencere_boyutu kadar verinin ortalaması)

Fonksiyon Medyan_Filtre (veri)
    Dönüş yap (son pencere_boyutu kadar verinin medyan değeri)

Fonksiyon Filtrele (yeni_değer)
    yeni_değeri geçmiş_veriye ekle
    Eğer geçmiş_veri uzunluğu > pencere_boyutu ise:
        En eski veriyi çıkar  // Pencere boyutunu koru

    Eğer geçmiş_veri uzunluğu > 2 ve |son_veri - önceki_veri| > gürültü_eşiği ise:
        Dönüş yap (Medyan_Filtre(geçmiş_veri))
    Eğer geçmiş_veri uzunluğu >= pencere_boyutu ise:
        Dönüş yap (Hareketli_Ortalama(geçmiş_veri))
    Aksi halde:
        Dönüş yap (yeni_değer)
```

Şekil 4.12: Multi-Stage Filtering Algorithm Pseudo kodu



Şekil 4.13: MSFA ile Güç (Power) filtreleme grafiği

Şekil 4.13, MSFA kullanılarak güç verilerinin filtrelenmesini göstermektedir. MSFA hem hareketli ortalama filtresi hem de medyan filtresi içeren hibrit bir yöntemdir ve ölçüm verilerindeki gürültüyü azaltmak amacıyla kullanılır.

- X ekseni (SANIYE): Zaman serisi içerisinde alınan ölçümlerin saniye bazlı zaman dalgalanmalarını gösterir.
- Y ekseni (WATT): Güç tüketim değerlerini watt cinsinden ifade eder.
- Kırmızı çizgi (*Power*): Ham güç verisi, doğrudan ölçüm cihazından gelen dalgalı ve gürültülü verileri temsil eder.
- Mavi çizgi (*SmoothedPower*): MSFA algoritması uygulanarak filtrelenmiş ve düzeltilmiş güç verisini göstermektedir.

Şekil 4.13'teki grafik sonuçlarına göre:

- Gürültü Azaltımı: Ham güç verisindeki (kırmızı çizgi) yüksek frekanslı dalgalanmalar ve ani sıçramalar MSFA algoritması ile büyük ölçüde ortadan kaldırılmıştır.
- Daha Kararlı Güç Eğrisi: Filtrelenmiş güç verisi (mavi çizgi), daha düzgün ve analiz edilebilir bir hale gelmiştir.

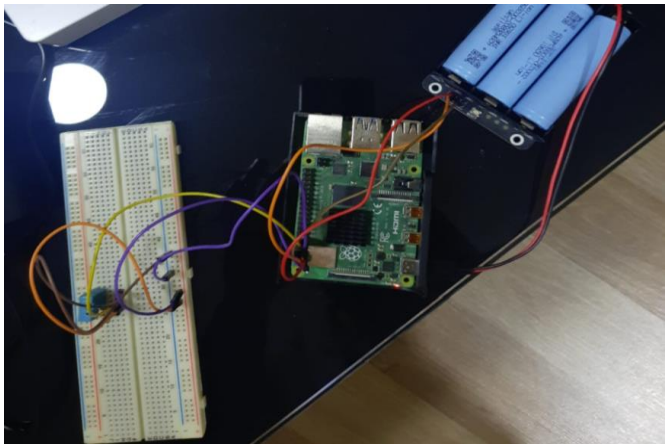
- Ani Piklerin Engellenmesi: Ham veride bulunan ani sıçramalar ve keskin değişimler, medyan filtresi sayesinde yumuşatılmıştır.
- Genel Stabilite Artışı: Hareketli ortalama filtresi, güç eğrisini daha akıcı ve dengeli hale getirerek, enerji tüketimi analizlerinde daha güvenilir sonuçlar elde edilmesini sağlamıştır.

4.3 Kullanılan Donanımlar ve Haberleşme Protokolleri

Bu çalışmada, güç yönetimi ve sensör verilerinin işlenmesi amacıyla *Raspberry Pi* 4, UPS *Module 3S*, DHT11 sıcaklık ve nem sensörü, akıllı priz ve LM2596 *step-down* gerilim düşürücü donanım bileşenleri kullanılmıştır. Bu bileşenler arasındaki veri iletişimi için I²C *(Inter-Integrated Circuit)*, GPIO *(General-Purpose Input/Output)* tercih edilmiştir.

4.3.1 Raspberry Pi 4 GPIO Pinleri ile Güç Modülünün ve Sensörünün Donanımsal Bağlantıları

Sistem *Raspberry Pi 4* üzerine inşa edilmiştir. *Raspberry Pi* hem bir veri toplama birimi hem de bir kontrol merkezi olarak işlev görmektedir. UPS modülünden güç durumu ve akım bilgilerini almak, DHT11 sensörü üzerinden ortam sıcaklığı ve nem değerlerini işlemek ve akıllı priz üzerinden klima gibi cihazları uzaktan kontrol etmek için kullanılmaktadır.



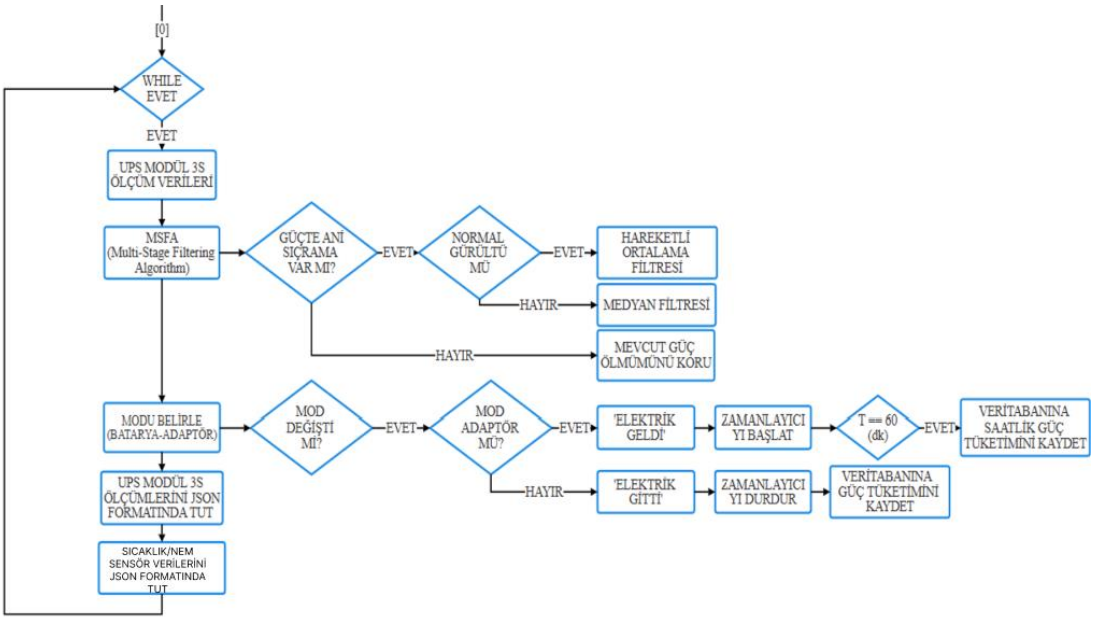
Şekil 4.14: Raspberry Pi 4, UPS Module 3S ve DHT11 sıcaklık nem sensörünün fiziksel bağlantısı

Şekil 4.14‘de görülmek üzere, fiziksel bağlantısı DHT11 sensörü, *Raspberry Pi'ye* doğrudan GPIO pinleri aracılığıyla bağlanmıştır. Sensörün veri çıkışı, *Raspberry Pi'nin* GPIO18 Şekil 2.3'te gösterilen pin 12 girişine bağlanmış olup, kararlı çalışmayı sağlamak amacıyla 10kΩ’luk bir *pull-up* direnci kullanılmıştır. DHT11'in GND pini, *Raspberry Pi'nin* Şekil 2.3'teki pin 9 olan GND bağlantısına bağlanmış, VCC pini ise 3.3V güç kaynağına bağlanarak enerji verilmiştir ancak veri iletiminde kararsızlıklar gözlemlendiği için 5V pinine geçilmiştir. Bu bağlantılar sayesinde sıcaklık ve nem verileri *Raspberry Pi* tarafından okunmakta ve işlenmektedir.

UPS *Module 3S* ise, *Raspberry Pi* ile I²C *(Inter-Integrated Circuit)* haberleşme protokolü üzerinden iletişim kurmaktadır. UPS modülü, *Raspberry Pi’ye* bağlı olan cihazlara kesintisiz güç sağlamakla birlikte, voltaj, akım ve batarya durumlarını ölçerek veri aktarımı gerçekleştirmektedir. UPS modülünün SCL pini, *Raspberry Pi'nin* GPIO3 Şekil 2.3'te gösterilen pin 5 olan I²C SCL hattına bağlanmış olup, SDA pini ise GPIO2 Şekil 2.3'te gösterilen pin 3 olan I²C SDA bağlantısına entegre edilmiştir. Ayrıca, UPS modülünün GND bağlantısı, *Raspberry Pi'nin* GND hattına Şekil 2.3'de gösterilen Pin 6'ya bağlanarak devrenin tamamlanması sağlanmıştır.

Bu bağlantılar sayesinde *Raspberry Pi,* UPS modülü üzerinden güç tüketimini analiz edebilmekte, batarya-adaptör geçişlerini takip edebilmekte ve bu verileri gerçek zamanlı olarak sistemin diğer bileşenlerine iletebilmektedir.

Şekil 4.15’te Raspberry Pi 4 ile UPS Module 3S güç yönetimi, sensör veri işleme ve SMTP bildirim akış diyagramı gösterilmiştir.



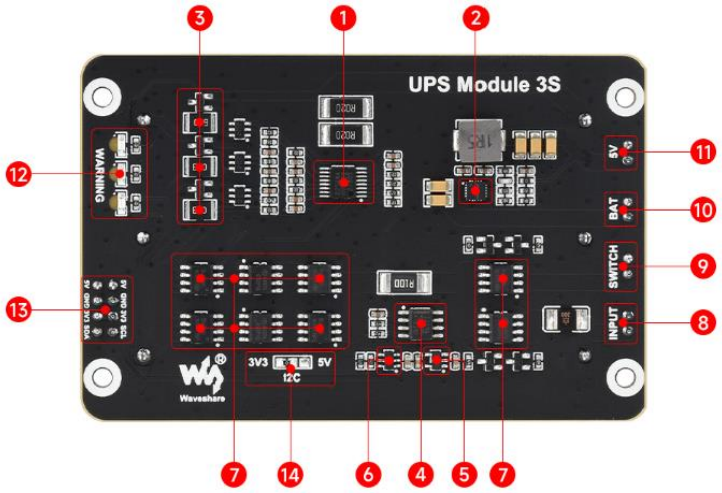
Şekil 4.15: Raspberry Pi 4 ile UPS Module 3S güç yönetimi, sensör veri işleme ve SMTP bildirim akış diyagramı

4.3.2 UPS Module 3S

UPS *Module 3S*, sistemin kesintisiz güç kaynağı olarak işlev görmektedir. Elektrik kesintisi durumunda bağlı cihazlara güç sağlayarak çalışma sürekliliğini korur. UPS modülü, dahili bir *INA219* akım ve voltaj sensörü içerir ve bu sensör I²C veri hattı üzerinden *Raspberry Pi'ye* veri göndermektedir. Böylece batarya seviyesinin izlenmesi ve adaptör/batarya modları arasında geçiş yapılmasını sağlar. *Lityum* iyon veya *lityum polimer* bataryalar ile uyumlu olan UPS modülü, 5V çıkış gerilimi sunarak sistemin stabil bir şekilde çalışmasını desteklemektedir.

Raspberry Pi, UPS modülünden gelen verileri analiz ederek güç yönetimini optimize etmekte ve batarya seviyesinin düşmesi durumunda kullanıcıya bildirim gönderebilmektedir. Adaptör ve batarya modları arasında geçiş yapılmasını sağlayan bu yapı, Raspberry Pi'nin güç kesintileri sırasında çalışmaya devam etmesine olanak tanımaktadır. UPS modülünün *Raspberry Pi* ile bağlantısı, SCL ve SDA üzerinden sağlanmaktadır. Ancak doğrudan gelen verinin doğruluğu, sistem yüküne bağlı olarak değişkenlik gösterebilmekte ve bazı durumlarda yanlış yorumlanabilmektedir. Bu

sebeple, verilerin analiz edilmesi ve işlenmesi sırasında ani sıçramaları ve hatalı okumaları engellemek adına ek optimizasyon teknikleri uygulanmıştır.



Chips

1. S-8254AA
   Li battery protection
2. SY8286
   5V voltage regulator
3. HY2213
   Equalizing charge
4. INA219
   Voltage/current monitoring
5. RT9193
   3.3V voltage regulator
6. NDC7002N
   Voltage translator

Others

7. AO4407A
   Battery protection MOS tube
8. Power port
   12.6 2A power input
9. Power output switch
10. Batteries series power output
11. 5V power output
12. Reverse warning indicators
13. Multi-functional expansion header
14. I2C level selection
    3.3V (default) / 5V

Şekil 4.16: UPS Module 3S bileşenleri ve bağlantı noktaları (Waveshare, 2025)

Şekil 4.16'da gösterilen 1 numaralı bölüm S-8254AA (*Li battery protection*), *Lityum iyon* bataryaların güvenli çalışmasını sağlamak amacıyla aşırı şarj, aşırı deşarj ve kısa devre koruması sağlayan bir batarya yönetim entegresidir.

2 numaralı bölüm SY8286 (5V *voltage regulator*), 5V çıkış gerilimi sağlayan bir voltaj regülatörüdür. Bu sayede *Raspberry Pi* gibi bağlı cihazlar stabil bir gerilimle beslenir. UPS *Module 3S*'in çıkış kapasitesi 5A'ya kadar ulaşabilir, ancak *Raspberry Pi* 4'ün anlık akım çekişleri 5A'yı aşabilir. *Raspberry Pi 4*, özellikle yoğun işlem yükü altında ve USB, kamera, ekran gibi bileşenler bağlıyken anlık olarak yüksek akım çekebilir. Eğer UPS modülü bu ani akım taleplerine yanıt veremezse, voltaj düşüşleri yaşanır ve *Raspberry Pi 4* beklenmedik şekilde kapanabilir veya yeniden başlayabilir.

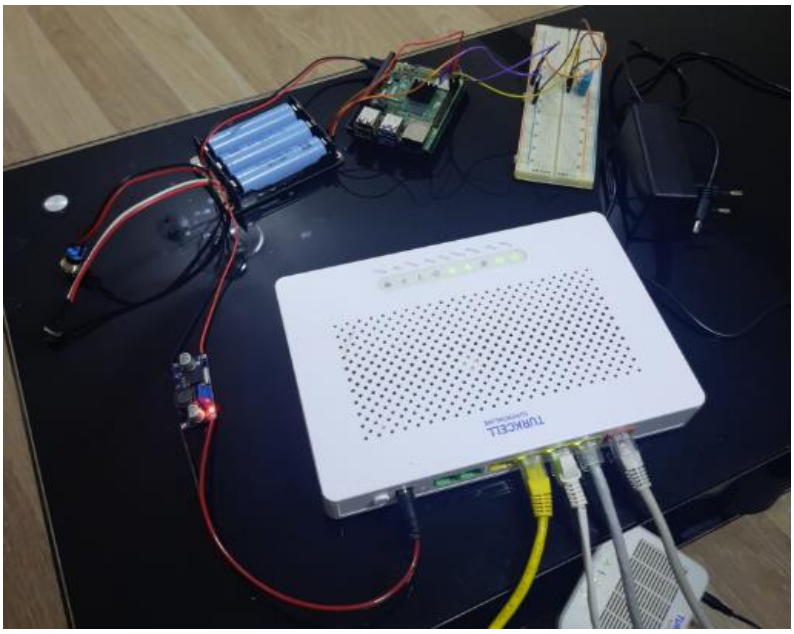
3 numaralı bölüm HY2213 *(Equalizing charge*), batarya hücreleri arasındaki şarj dengesini sağlayarak, bataryaların ömrünü uzatan bir dengeleme (*balancing*) sistemidir.

5 numaralı bölüm RT9193 (3.3V *voltage regulator*), *Raspberry Pi'nin* ve UPS modülünün düşük güç tüketimli bileşenlerini beslemek için 3.3V regülatör görevi görür.

7 numaralı bölüm AO4407A (*Battery protection MOS tube*), batarya devresini aşırı akım ve kısa devre durumlarına karşı koruyan bir MOSFET'dir. Bataryanın güvenli çalışmasını sağlayarak aşırı akım yüklenmesini engeller.

8 numaralı bölüm *Power port* (12.6V 2A *power input*), UPS modülünün 12.6V ve 2A seviyesinde enerji almasını sağlayan giriş portudur. Adaptör veya harici güç kaynağından gelen enerjiyi bataryalara iletir.

10 numaralı bölüm *Batteries Series Power Output*, batarya hücrelerinin seri bağlanarak güç çıkışı sağlamasını kontrol eden bağlantı noktasıdır. Bu çıkış, UPS modülünün batarya voltajını doğrudan sağlar ve yaklaşık 12.6V çıkış gerilimi üretir. Bu çıkışın sistem açısından önemi oldukça büyüktür çünkü bu noktadan alınan güç, *Raspberry Pi* haricinde farklı cihazların da beslenmesini mümkün kılmaktadır. Bu çalışmada, söz konusu çıkış bir modem beslemek için kullanılmıştır. Ancak, modemlerin genellikle 12V giriş gerilimiyle çalıştığı göz önünde bulundurulduğunda, UPS modülünden gelen 12.6V çıkış doğrudan modeme verilmemiş, önce LM2596 *step-down* voltaj regülatörü ile 12V seviyesine düşürülerek besleme sağlanmıştır. LM2596 voltaj düşürücü regülatörünün kullanılması, modemin gereğinden yüksek voltajla beslenerek zarar görmesini engellemekte ve sistemin uzun ömürlü olmasını sağlamaktadır.



Şekil 4.17: UPS Modülü ile Step-Down voltaj düşürücü kullanılarak modemin batarya beslemesi

UPS modülü bataryalardan beslendiğinde çıkış voltajı zamanla değişiklik gösterebileceğinden, *step-down* regülatörü kullanmak voltajı sabitleyerek modem ve diğer bileşenlerin kararlı çalışmasını sağlar. Bu yapı sayesinde, Şekil 4.17'de şebekeden gelen bir elektrik beslemesi olmadığı görülmekte, modemin bataryadan beslenerek çalıştığı görülmektedir, elektrik kesintisi yaşansa dahi modemin kesintisiz çalışması sağlanmakta, internet bağlantısı korunarak uzaktan erişim ve bildirim sistemleri sürdürülebilir hale getirilmektedir. Özellikle, *Raspberry Pi* ve UPS modülü internete bağımlı bir sistem olarak çalışıyorsa, modemin UPS üzerinden güç alarak çalışması, sistemin uzaktan yönetimini ve kesinti bildirimlerinin gönderilmesini mümkün kılmaktadır.

4.3.2.1 UPS Module 3S ile Float Charge Modunda Karşılaşılan Sorunlar

Bu bölümde, UPS *Module 3S'nin float charge* (damlama şarj) modunda karşılaşılan sensör okuma hataları ve yanlış mod tespiti gibi sorunlar ele alınacaktır. Adaptör bağlantısı aktifken bataryanın %95 ve üzerindeki doluluk seviyelerinde, UPS modülünün şarj yönetim sisteminde meydana gelen dalgalanmalar, yanlış elektrik kesintisi algılanmasına ve sensör verilerinde gürültü oluşmasına neden olmaktadır.

Float charge modu, batarya doluluk seviyesi %95'in üzerine çıktığında UPS modülünün bataryayı düşük akımla şarj etmeye devam ettiği durumdur. Bu modda adaptör voltajı, batarya voltajının hemen üstünde tutulmakta ve batarya şarj işlemi tamamen durmadan, mikro akımlar ile devam etmektedir. Bu sırada UPS, şarj devresini belirli aralıklarla açıp kapamakta ve bu geçişler sırasında *INA219* akım sensörü yanlış değerler üretmektedir.

Adaptör bağlı olmasına rağmen UPS modülü zaman zaman batarya modunda olduğunu rapor edebilmektedir. *Float charge* modunda değilken adaptör akımı genellikle 0.1*A* ve üzeri olur, batarya akımı da genellikle -0.1A ve altı olur. Akım verilerinde dalgalanma meydana gelerek -0.001 A ile +0.001 A arasında değişen mikro akım değerleri sensör tarafından okunmaktadır.

Bu sorunun giderilmesi için üç farklı yazılımsal filtreleme tekniği geliştirilmiştir:

Sensör Reset ve Sahte Okuma Filtresi: Sistem ilk başlatıldığında, *INA219* sensöründen gelen ilk 5 okuma sahte okuma olarak değerlendirilmiş ve işleme dahil edilmemiştir. Sensör her başlangıçta sıfırlanarak (*reset*) en doğru verinin alınması sağlanmıştır.

%95 Üzeri Batarya Doluluğunda Akım Gürültüsü Filtreleme: Eğer batarya doluluk oranı %95'in üzerindeyse ve akım -0.001 A ile +0.001 A arasında değişiyorsa, bu değer gürültü olarak kabul edilmiştir.

Mod Geçişleri İçin *Debounce* Filtresi: UPS modunun yanlış algılanmasını önlemek amacıyla, sistemin bir moda geçiş yapmadan önce en az 3 ardışık okuma ile aynı modda olduğunu doğrulaması sağlanmıştır. Eğer son 5 okumanın en az 3'ü adaptör modundaysa, adaptör moduna geçiş onaylanmaktadır. Aynı şekilde, son 5 okumanın en az 3'ü batarya modundaysa, batarya moduna geçiş gerçekleştirilmektedir.

4.4 Sistem Pseudocode ve Veri Akış Süreci

Bu bölümde, Raspberry Pi tabanlı güç yönetimi, sensör verilerinin işlenmesi, UPS modülü entegrasyonu ve sistemin genel işleyişini temsil eden *pseudo code* (kaba kod) sunulmaktadır. Sistem, donanım bileşenlerinin başlatılması, güç yönetiminin gerçekleştirilmesi, veri akışının kontrol edilmesi ve kullanıcı arayüzü ile entegrasyon süreçlerini içermektedir.

4.4.1 Sensör Başlatma (INA219 ve DHT11)

```
Fonksiyon SensörleriBaşlat()
  INA219 başlat
  DHT11 başlat
Fonksiyon Sonu
```

4.4.2 Multi-Stage Filtering Algorithm (MSFA) ile Veri Filtreleme

```
Sınıf MultiStageFilterAlgorithm(pencere_boyutu)
  Özellikler:
    pencere_boyutu (int) → Hareketli ortalama için pencere büyüklüğü
    geçmiş_veri (liste) → Son ölçüm değerlerini saklar

  Fonksiyon HareketliOrtalamaFiltrele(veri)
    Eğer veri uzunluğu pencere_boyutundan küçükse:
      Dönüş veri toplamı / veri uzunluğu
    Aksi Halde:
      Dönüş Son pencere_boyutu kadar verinin ortalaması

  Fonksiyon MedyanFiltrele(veri)
    Dönüş Son pencere_boyutu kadar verinin medyanı

  Fonksiyon VeriİşleYeni_değer)
    geçmiş_veri listesine yeni_değer ekle
    Eğer geçmiş_veri uzunluğu pencere_boyutundan büyükse:
      İlk elemanı sil

    Eğer Son iki değer arasındaki fark > 0.1 ise:
      Dönüş MedyanFiltrele(geçmiş_veri)
    Eğer geçmiş_veri uzunluğu pencere_boyutuna eşit veya büyükse:
      Dönüş HareketliOrtalamaFiltrele(geçmiş_veri)
    Aksi Halde:
      Dönüş yeni_değer
```

4.4.3 Güç Kaynağı Modu Tespit Etme

```
Fonksiyon GüçModuTespit(Voltaj, Akım_mA)
   Eğer Akım_mA < -250 mA ve Voltaj < 12.4 ise:
      Eğer BataryaModuGeçişZamanı >= 3 saniye ise:
         Dönüş "batarya"
      Aksi Halde:
         BataryaModuGeçişZamanı = ŞuAnkiZaman
   Aksi Halde:
      BataryaModuGeçişZamanı = Sıfırla

   Eğer Voltaj >= 12.4 ve Akım_mA > -50 mA ise:
      Eğer AdaptörModuGeçişZamanı >= 1 saniye ise:
         Dönüş "adaptör"
      Aksi Halde:
         AdaptörModuGeçişZamanı = ŞuAnkiZaman
   Aksi Halde:
      AdaptörModuGeçişZamanı = Sıfırla

   Dönüş ÖncekiMod
Fonksiyon Sonu
```

4.4.4 Mod Değişiminde Bildirim Gönderme

```
Fonksiyon ModDeğişti(ŞuAnkiMod)
   Eğer ŞuAnkiMod ≠ ÖncekiMod ise:
      Eğer ŞuAnkiMod = "batarya" ise:
         MailGönder("Elektrik Kesildi", "Sistem batarya moduna geçti.")
      Eğer ŞuAnkiMod = "adaptör" ise:
         MailGönder("Elektrik Geldi", "Sistem tekrar adaptör ile çalışıyor.")
      ÖncekiMod = ŞuAnkiMod
Fonksiyon Sonu
```

4.4.5 UPS ve Sıcaklık Verilerini Güncelleme

```
Fonksiyon UPSVerisiniGüncelle(Voltaj, Akım, Güç, BataryaYüzdesi, Mod)
   Güncel UPS verilerini kaydet:
      - Voltaj
      - Akım
      - Güç (MSFA ile filtrelenmiş)
      - Batarya Yüzdesi
      - Çalışma Modu (adaptör/batarya)
Fonksiyon Sonu

Fonksiyon SicaklikVerisiniGüncelle()
```

```
    Eğer DHT11 verileri başarılıysa:
        Güncel sıcaklık ve nem verilerini kaydet
    Aksi Halde:
        Hata mesajı kaydet
Fonksiyon Sonu
```

4.4.6 Sensör Takip Döngüsü

```
Fonksiyon SensörTakipDöngüsü()
    SensörleriBaşlat()
    Sonsuz Döngü Başlat:
        Voltaj = INA219OkuBusVoltage()
        Akım_mA = INA219OkuCurrent()
        Güç = INA219OkuPower()

        FiltrelenmişGüç = MSFA.VerİşleGüç)

        Eğer MevcutMod = "adaptör":
            GünlükEnerji += FiltrelenmişGüç * (1/3600000)

        BataryaYüzdesi = (Voltaj - 9) / 3.6 * 100
        YeniMod = GüçModuTespit(Voltaj, Akım_mA)

        Eğer YeniMod ≠ MevcutMod:
            ModDeğişti(YeniMod)
            MevcutMod = YeniMod

        UPSVerisiniGüncelle(Voltaj, Akım_mA, FiltrelenmişGüç,
BataryaYüzdesi, MevcutMod)
        SicaklikVerisiniGüncelle()

        1 saniye bekle
    Sonsuz Döngü Sonu
Fonksiyon Sonu
```

4.4.7 Mail Gönderme

```
Fonksiyon MailGönder(Konu, İçerik)
    SMTP Bağlan
    Maili Gönder
    SMTP Bağlantısını Kapat
Fonksiyon Sonu
```

4.4.8 Flask Web Servis Endpoint'leri

Fonksiyon GET /get-UPS-data()
  Dönüş Güncel UPS Verileri

Fonksiyon GET /get-temperature()
  Dönüş Güncel Sıcaklık ve Nem Verileri

Fonksiyon POST /turn-on()
  Akıllı Prizi Aç ve Sonucu Dön

Fonksiyon POST /turn-off()
  Akıllı Prizi Kapat ve Sonucu Dön

Fonksiyon GET /get-status()
  Akıllı Priz Durumunu Döndür

Fonksiyon GET /get-energy()
  Akıllı Priz Enerji Tüketimini Döndür

4.4.9 SQL Server ile Saatlik Enerji Kaydı

Fonksiyon SaatlikTüketimKaydet(ToplamEnerji)
  Güncel Elektrik Fiyatını Al
  Tüketim Maliyeti = ToplamEnerji * Fiyat
  SQL'e Saatlik Kayıt Ekle
Fonksiyon Sonu

4.4.10 Flask API ve Sensör Takip Başlatma

Fonksiyon Başlangıç()
  SensörTakipDöngüsü Başlat (Ayrı Thread)
  Flask Web Server Başlat (Port 5000)
Fonksiyon Sonu

5. SONUÇ VE ÖNERİLER

Bu çalışmada, *Raspberry Pi* tabanlı bir güç yönetim sistemi geliştirilmiş ve UPS *Module 3S* ile kesintisiz enerji yönetimi sağlanmıştır. Çalışmanın temel amacı, elektrik kesintileri sırasında sistemin çalışma sürekliliğini sağlamak, batarya-adaptör geçişlerini optimize etmek ve uzaktan izlenebilir bir güç takip mekanizması oluşturmaktır. Sistem, *INA219* sensöründen alınan verileri işleyerek batarya şarj durumu, enerji tüketimi ve adaptör bağlantısının durumunu analiz edebilmekte, *RESTful* API sayesinde bu verileri anlık olarak kullanıcıya sunmaktadır.

Geliştirilen *Multi-Stage Filtering Algorithm* (MSFA), UPS'in akım ve voltaj ölçümlerinde oluşan dalgalanmaları başarılı bir şekilde filtreleyerek ölçümlerin güvenilirliğini artırmıştır. Özellikle batarya doluluk oranı %95'in üzerine çıktığında UPS'in şarj yönetim modunda oluşan dalgalanmalar filtrelenmiş, yanlış elektrik kesintisi algılamaları engellenmiştir. Bu sayede sistemin kararlılığı artırılmış ve batarya-adaptör geçişlerinde daha doğru ölçümler elde edilmiştir.

Bu çalışmada geliştirilen sistem, temel güç yönetimi ve sensör verisi takibini sağlamaktadır. Ancak, sistemin daha akıllı ve verimli çalışabilmesi için gelecekte yapılabilecek geliştirmeler şu şekilde önerilebilir:

- Mobil Uygulama Entegrasyonu: *RESTful* API sayesinde sistem mobil bir uygulama ile entegre edilerek kullanıcı dostu bir arayüz oluşturulabilir. Kullanıcılar, sistem durumunu mobil uygulama üzerinden anlık olarak takip edebilir ve bildirimler alabilir.
- Röle Tabanlı Akıllı Geçiş Sistemi: Batarya ve adaptör arasında geçiş yaparken tüketilen güç hesaplanarak, belirlenen bir eşik değerine göre sistem otomatik olarak en verimli güç kaynağına geçebilir. Böylece gereksiz batarya kullanımı azaltılarak enerji tasarrufu sağlanabilir.
- Makine Öğrenmesi ile Güç Tüketimi Optimizasyonu: Toplanan güç tüketim verileri kullanılarak, makine öğrenmesi algoritmaları ile farklı senaryolara göre batarya-adaptör geçiş stratejileri belirlenebilir. Örneğin, sistemin bağlı olduğu cihazların kullanım saatlerine göre batarya tüketimi optimize edilebilir.
- GSM Modülü Kullanarak Alternatif Bağlantı: Elektrik kesintisi durumunda modem çalışmaya devam etse bile, internet servis sağlayıcısının altyapısında

kesinti yaşanabilir. Bu durumda *SIM800C* gibi *GSM* modülleri kullanılarak kesintisiz haberleşme sağlanabilir ve kullanıcıya mobil ağ üzerinden bildirim gönderilebilir.

- Akıllı Priz Yönetimi ile Dinamik Enerji Tasarrufu: Sisteme bağlı akıllı prizler, elektrik kesintisi anında gereksiz yükleri otomatik olarak kapatacak şekilde programlanabilir. Örneğin, kritik olmayan cihazların güç tüketimi azaltılarak batarya ömrü uzatılabilir.
- JWT ile Kimlik Doğrulama ve Yetkilendirme: Mevcut sistem, yetkilendirme mekanizması olmadan çalışmaktadır. Kullanıcıların belirli yetkilere sahip olması ve erişim kontrolünün sağlanabilmesi için JSON Web Token (JWT) ile güvenli kimlik doğrulama mekanizması entegre edilebilir. Bu sayede yalnızca yetkilendirilmiş kullanıcılar enerji tüketim verilerini izleyebilir veya sistem üzerinde değişiklik yapabilir.

Bu çalışma, düşük maliyetli ve modüler bir yapıya sahip olması nedeniyle farklı senaryolara uyarlanabilir bir sistem olarak tasarlanmıştır. Önerilen geliştirmeler ile sistemin daha akıllı ve verimli hale getirilmesi mümkündür. Özellikle, endüstriyel ortamlarda, IoT tabanlı güç yönetimi uygulamalarında ve enerji tasarrufu odaklı sistemlerde bu yapıdan faydalanılabilir.

6. KAYNAKLAR

Aftab, M., Hameed, A. N., Samiq, A., Shah, M. B., Pasha, M. A., Zaffar, N. A. ve Sikora, A., "IoT-enabled Smart Energy Metering Solution with Soft-UPS for Developing Countries", *Proceedings of the 11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS 2021)*, c. 2, Institute of Electrical and Electronics Engineers Inc., 899-904, doi:10.1109/IDAACS53288.2021.9660881, (2021).

Alzamzami, M. N. ve Azizah, F. N., "GraphQL-based Backend Service Development Tool for CRUD Operations, Authentication, and Authorization", *Proceedings of 2022 International Conference on Data and Software Engineering (ICoDSE 2022)*, Institute of Electrical and Electronics Engineers Inc., 77-82, doi:10.1109/ICoDSE56892.2022.9971818, (2022).

Anilkumar, A., George, A., Aswin, M. R., Devika, G. ve Jyothy, A., "IoT Based Smart Plug", *Proceedings - 2023 International Conference on Innovations in Engineering and Technology (ICIET 2023)*, Institute of Electrical and Electronics Engineers Inc., doi:10.1109/ICIET57285.2023.10220898, (2023).

Ronacher, A., "Flask Documentation [online]", (10 Mayıs 2025), Web adresi: https://flask.palletsprojects.com/en/stable/, (2017).

Bolanowski, M., Żak, K., Paszkiewicz, A., Ganzha, M., Paprzycki, M., Sowiński, P. ve Palau, C. E., "Eficiency of REST and gRPC realizing communication tasks in microservice-based ecosystems", doi:10.3233/FAIA220242, (2022).

Brito, G. ve Valente, M. T., "REST vs GraphQL: A controlled experiment", *Proceedings - IEEE 17th International Conference on Software Architecture (ICSA 2020)*, Institute of Electrical and Electronics Engineers Inc., 81-91, doi:10.1109/ICSA47634.2020.00016, (2020).

Broadcom Europe Ltd. ve Raspberry Pi Ltd., "Colophon Legal Disclaimer Notice TECHNICAL AND RELIABILITY DATA FOR RASPBERRY PI PRODUCTS (INCLUDING DATASHEETS) AS MODIFIED FROM TIME TO TIME ('RESOURCES') ARE PROVIDED BY RASPBERRY PI LTD ('RPL') 'AS IS' AND ANY EXPRESS OR IMPLIED [online]", (10 Mayıs 2025), Web adresi: https://datasheets.raspberrypi.com/bcm2711/bcm2711-peripherals.pdf, (2022).

Chen, G. ve Wanner, R., "Secure Email Transmission Protocols -- A New Architecture Design [online]", (10 Mayıs 2025), Web adresi: http://arxiv.org/abs/2208.00388, (2022).

Deering, S. ve Hinden, R., "Internet Protocol, Version 6 (IPv6) Specification [online]", (10 Mayıs 2025), Web adresi: https://doi.org/10.17487/rfc2460, (1998).

Duman, B. ve Şen, E. B., "Yapay Zeka Tekniklerinin Tek Kart Bilgisayar Sistemlerinde Uygulanabilirliği (Applicability of Artificial Intelligent on Single Board Computer Systems) [online]", (10 Mayıs 2025), Web adresi: https://www.researchgate.net/publication/346970622, (2019).

Eddy, W. M., “RFC 9293: Transmission Control Protocol (TCP) [online]”, (10 Mayıs 2025), Web adresi: https://www.rfc-editor.org/info/rfc9293, (2022).

Ehsan, A., Abuhaliqa, M. A. M. E., Catal, C. ve Mishra, D., “RESTful API Testing Methodologies: Rationale, Challenges, and Solution Directions”, *Applied Sciences (Switzerland)*, MDPI, doi:10.3390/app12094369, (2022).

Kıral, G. E., “Akıllı Şebekelerde Enerji Yönetimi İçin Akıllı Priz Geliştirilmesi”, Yüksek Lisans Tezi, *Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü*, İstanbul, (2014).

Hossain, M., Binti, J. ve Uddin, M., “A Review Paper on IPv4 and IPv6: A Comprehensive Survey”, *American Journal of Computer Science and Technology*, 7(4), 170-175, doi:10.11648/j.ajcst.20240704.14, (2024).

John G. Myers ve Marshall T. Rose. (1996). *Post Office Protocol - Version 3 (POP3)*. https://datatracker.ietf.org/doc/html/rfc1939

Myers, J. G. ve Rose, M. T., “Post Office Protocol - Version 3 (POP3) [online]”, (10 Mayıs 2025), Web adresi: https://datatracker.ietf.org/doc/html/rfc1939, (1996).

Kamiński, Ł., Kozłowski, M., Sporysz, D., Wolska, K., Zaniewski, P. ve Roszczyk, R., “Comparative review of selected Internet communication protocols [online]”, (10 Mayıs 2025), Web adresi: http://arxiv.org/abs/2212.07475, (2022).

Khan, R. ve Mian, A. N., “Sustainable IoT sensing applications development through GraphQL-based abstraction layer”, *Electronics (Switzerland)*, 9(4), doi:10.3390/electronics9040564, (2020).

Leens, F., “An introduction to I²C and SPI protocols”, *IEEE Instrumentation & Measurement Magazine*, 12(1), 8-13, doi:10.1109/MIM.2009.4762946, (2009).

Malakhov, K. S., Kurgaev, A. P. ve Velychko, V. Y., “Modern RESTful API DSLs and frameworks for RESTful web services API schema modeling, documenting, visualizing”, *Problems in Programming*, (4), 59-68, doi:10.15407/pp2018.04.059, (2018).

Muller, G. L., “HTML5 WebSocket protocol and its application to distributed computing [online]”, (10 Mayıs 2025), Web adresi: http://arxiv.org/abs/1409.3367, (2014).

Nieman, K. ve Sajal, S., “A Comparative Analysis on Load Balancing and gRPC Microservices in Kubernetes”, *Proceedings - 2023 Intermountain Engineering, Technology and Computing (IETC 2023)*, Institute of Electrical and Electronics Engineers Inc., 322-327, doi:10.1109/IETC57902.2023.10152023, (2023).

NXP Semiconductors, “UM10204 I²C-bus specification and user manual Rev. 7.0-1 October 2021 [online]”, (10 Mayıs 2025), Web adresi: https://www.nxp.com/docs/en/user-guide/UM10204.pdf, (2021).

Prathyusha, M. R. ve Bhowmik, B., “Development of IoT-Based Smart Home Application with Energy Management”, *Proceedings - International Conference on Sustainable Communication Networks and Application (ICSCNA 2023)*, Institute of Electrical and Electronics Engineers Inc., 367-373, doi:10.1109/ICSCNA58489.2023.10370276, (2023).

Sanayi ve Verimlilik Genel Müdürlüğü, “Elektrik-Elektronik Sektörü Raporu”, *Sektörel Raporlar ve Analizler Serisi*, Sanayi ve Teknoloji Bakanlığı, (2020).

Rodríguez, C., Baez, M., Daniel, F., Casati, F., Trabucco, J. C., Canali, L. ve Percannella, G., “REST APIs: A large-scale analysis of compliance with principles and best practices”, *Lecture Notes in Computer Science*, c. 9671, Springer Verlag, 21-39, doi:10.1007/978-3-319-38791-8_2, (2016).

Şahin, S. K. ve Delebe, E., *Raspberry Pi & Python ile Nesnelerin İnterneti*, (Ed: B. Elitoğ), İstanbul: Unikod Yayın, (2021).

Ylonen, T. ve Lonvick, C., “SSH Connection Protocol [online]”, (10 Mayıs 2025), Web adresi: https://www.rfc-editor.org/rfc/pdfrfc/rfc4254.txt.pdf, (2006).

Tekli, J. M., Damiani, E., Chbeir, R. ve Gianini, G., “SOAP processing performance and enhancement”, *IEEE Transactions on Services Computing*, 5(3), 387-403, doi:10.1109/TSC.2011.11, (2012).

Walugembe, F. N. ve Kawuma, R., “A Primer on Network Communication [online]”, (10 Mayıs 2025), Web adresi: https://www.researchgate.net/publication/369146994, (2023).

Wang, Y. ve Yi, Z., “Reconfigurable optical demultiplexer calling framework based on gRPC Design”, *Proceedings - 2022 7th International Conference on Communication, Image and Signal Processing (CCISP 2022)*, Institute of Electrical and Electronics Engineers Inc., 13-17, doi:10.1109/CCISP55629.2022.9974287, (2022).

Yüce, E., Bektaş, O., Orçan, S., Gökırmak, Y., Soysal, M. ve Ünver, M., “Ulusal IPv6 Protokol Altyapısı Tasarımı ve Geçişi Projesi [online]”, (10 Mayıs 2025), Web adresi: https://ulakbim.tubitak.gov.tr/sites/images/Ulakbim/didim_calistayi_ipv6.pdf, (2008).

Zeadally, S., Siddiqui, F. ve Baig, Z., “25 years of Bluetooth technology”, *Future Internet*, 11(9), doi:10.3390/fi11090194, (2019).

Raspberry Pi Foundation, “Raspberry Pi 4 Model B Specifications [online]”, (15 Mart 2025), Web adresi: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/, (2025).

Raspberry Pi Foundation, “15W USB-C Power Supply Product Brief [online]”, (15 Mart 2025), Web adresi: https://datasheets.raspberrypi.com/power-supply/15w-usb-c-power-supply-product-brief.pdf, (2024).

Raspberry Pi Foundation, “Products [online]”, (15 Mart 2025), Web adresi: https://www.raspberrypi.com/products/, (2024).

Raspberry Pi Foundation, “Raspbian Release Notes [online]”, (15 Mart 2025), Web adresi: https://downloads.raspberrypi.com/raspbian/release_notes.txt, (2025).

Raspberry Pi Foundation, “Operating Systems [online]”, (15 Mart 2025), Web adresi: https://www.raspberrypi.com/software/operating-systems/, (2025).

SSH Communications Security, “History [online]”, (15 Mart 2025), Web adresi: https://www.ssh.com/about/history/, (2025).

Waveshare, “UPS Module 3S [online]”, (15 Mart 2025), Web adresi: https://www.waveshare.com/ups-module-3s.htm, (2025).