T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ENDÜSTRİYEL KULLANIMA UYGUN LABVIEW VE GÖMÜLÜ SİSTEM TABANLI YENİ BİR IOT ÇÖZÜMÜ

YÜKSEK LİSANS TEZİ

Görkem SUNGUR

Enstitü Anabilim Dalı : MEKATRONİK MÜHENDİSLİĞİ

Tez Danışmanı : Dr. Öğr. Üyesi Barış BORU

Ağustos 2018

T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ENDÜSTRİYEL KULLANIMA UYGUN LABVIEW VE GÖMÜLÜ SİSTEM TABANLI YENİ BİR IOT ÇÖZÜMÜ

YÜKSEK LİSANS TEZİ

Görkem SUNGUR

Enstitü Anabilim Dalı : MEKATRONİK MÜHENDİSLİĞİ

Bu tez 31/08/2018 tarihinde aşağıdaki jüri tarafından oybirliği / ~~oyçokluğu~~ ile kabul edilmiştir.

Dr. Öğr. Üyesi Barış BORU Jüri Başkanı	Dr. Öğr. Üyesi Sezgin KAÇAR Üye	Dr. Öğr. Üyesi Gökay BAYRAK Üye

BEYAN

Tez içindeki tüm verilerin akademik kurallar çerçevesinde tarafımdan elde edildiğini, görsel ve yazılı tüm bilgi ve sonuçların akademik ve etik kurallara uygun şekilde sunulduğunu, kullanılan verilerde herhangi bir tahrifat yapılmadığını, başkalarının eserlerinden yararlanılması durumunda bilimşel normlara uygun olarak atıfta bulunulduğunu, tezde yer alan verilerin bu üniversite veya başka bir üniversitede herhangi bir tez çalışmasında kullanılmadığını beyan ederim.

Görkem SUNGUR

31.08.2018

TEŞEKKÜR

Yüksek lisans eğitimim boyunca değerli bilgi ve deneyimlerinden yararlandığım, her konuda bilgi ve desteğini almaktan çekinmediğim, araştırmanın planlanmasından yazılmasına kadar tüm aşamalarında yardımlarını esirgemeyen, teşvik eden, aynı titizlikte beni yönlendiren değerli danışman hocam Dr. Öğr. Üyesi Barış BORU'ya teşekkür ederim.

Bu çalışma boyunca yardımlarını ve desteklerini esirgemeyen sevgili aileme ve çalışma arkadaşlarıma teşekkürlerimi sunarım.

İÇİNDEKİLER

SİMGELER VE KISALTMALAR LİSTESİ

AMC : Asynchronous Message Communication
CPU : Central Processing Unit
ECG : Electrocardiogram
FGV : Functional Global Variable
GPIO : General-Purpose Input/Output
GPU : Graphics Processing Unit
GUI : Graphical User Interface
HTML : Hypertext Markup Language
I2C : Inter-Integrated Circuit
IDE : Integrated Development Environment
IP : Internet Protocol
ISO : International Organization for Standardization
IoT : Internet of Things
LP : LaunchPad
M2M : Machine to Machine
MQTT : Message Queuing Telemetry Transport
ODBC : Open Database Connectivity
OSI : Open Systems Interconnection
PC : Personal Computer
PIC : Peripheral Interface Controller
PLC : Programmable Logic Controller
RPI : Raspberry Pi
SMTP : Simple Mail Transfer Protocol
SPI : Serial Peripheral Interface
SQL : Structured Query Language
TCP : Transmission Control Protocol

TLS	: Transport Layer Security
UDL	: Universal Data Link
UDP	: User Datagram Protocol
USB	: Universal Serial Bus
VI	: Virtual Instruments
VIPM	: Virtual Instruments Package Manager

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

ÖZET

Anahtar kelimeler: Gömülü Sistemler, Nesnelerin İnterneti (IoT), Endüstri 4.0, Raspberry Pi

Günümüzün popüler konularından biri olan nesnelerin interneti (IoT) teknolojisi dünya çapında giderek yayılmaya başlamıştır. Nesnelerin interneti teknolojisi, tüm akıllı cihazların ve makinelerin kablosuz haberleşme metotlarını kullanarak birbirleriyle veya daha büyük sistemlerle haberleşmesini sağlayan iletişim ağıdır. Bu teknoloji hemen hemen tüm sektörlerde kullanılarak gerek zamandan, gerek verimlilik ve maliyet açısından oldukça fayda sağlamaktadır.

Bu çalışmada, nesnelerin interneti teknolojisine farklı ve yenilikçi bir çözüm getirmek amacıyla, özelleştirilebilir, endüstriyel otomasyon uygulamalarında kullanılabilecek genel bir IoT sisteminin tasarımı gerçekleştirilmiştir. Hem uygulamadan, hem donanımdan, hem de yazılımdan bağımsız olarak, kolay eklemeli mimari yapısıyla 1-N sunucu-istemcili bir IoT sistemi oluşturulmuştur.

Geliştirilen sistemde, sunucu için LabVIEW programı kullanılarak kolay eklemeli mimaride bir arayüz yazılımı geliştirilmiştir. Raspberry Pi 2 ve MSP430F5529 LP donanım kartları istemci olarak kullanılmıştır. Bu kartlara bir takım analog ve dijital sensörler bağlanarak verilerin sunucuya internet üzerinden aktarılması sağlanmıştır. Kartların sunucuya kolayca eklenebilmesi için sistemin alt yapısında TCP/IP protokolü kullanılmasıyla birlikte yeni ekle-çıkar protokolü oluşturulmuştur. Bu yeni bir ekle-çıkar protokolü sayesinde sunucu yazılımında değişiklik yapmadan, farklı donanımların farklı yazılım dilleri ile birlikte sunucuya kolayca bağlanabilmesi sağlanmıştır. Tüm istemcilerin verileri asenkron olarak toplanmış, sunucu tarafından yerel MySQL veri tabanına kaydedilmiştir.

Bu çalışma sonucunda, sunucu yazılımı değiştirilmeden, donanımdan ve yazılımdan bağımsız, yeni bir endüstriyel IoT yaklaşımı geliştirilmiştir. Bu yeni yaklaşım, hız, esneklik ve maliyet avantajı ile ön plana çıkmaktadır.

A NEW LABVIEW AND EMBEDDED SYSTEM BASED IOT SOLUTION FOR INDUSTRIAL APPLICATIONS

SUMMARY

Keywords: Embedded Systems, Internet of Things (IoT), Industry 4.0, Raspberry Pi

The Internet of things (IoT) technology, one of today's popular topics, has become increasingly spreading worldwide. The IoT technology is a communication network that enables all smart devices and machines to communicate with each other or with larger systems using wireless communication methods. This technology is used in almost all sectors, and it is very beneficial both in terms of efficiency and cost.

In this study, a generalized IoT system design for use in industrial automation applications that can be customized to find a different and innovative solution to the Internet of things technology has been carried out. 1-N server-client IoT system has been created with easy plugin architecture independent of the application and both hardware and software.

In this system, interface software is created in the easy plugin architecture using LabVIEW program for the server. The Raspberry Pi 2 and MSP430F5529 LP embedded system hardware were used as clients. A number of analog and digital sensors can be connected to this hardware and the data can be transferred to the server over the internet. In order to easily add hardware to the server, a new application protocol has been created with the use of TCP/IP protocol in the system's substructure. With this new application protocol, different type hardware has been easily connected to the server with using different software languages, without changing the software of the server. Each client's data has been aggregated asynchronously and saved to the local MySQL database by the server.

As a result of this study, a new industrial IoT approach has been developed which is independent of hardware and software without changing user interface software. This innovative approach is at the forefront with speed, flexibility and cost advantages.

BÖLÜM 1. GİRİŞ

Günümüzde teknolojinin gelişmesiyle, endüstride devrim niteliğinde değişimler meydana gelmiştir. Endüstrideki verimliliğin arttırılması ve hataların minimize edilmesi adına yapılan çalışmalar sonucunda üretim sahalarında çalışan iş gücünün makineleştirilmesi ve daha sonrasında üretimin otomatik hale getirilmesi planlanmıştır. Bu çalışmalar ile birlikte Sanayi Devrimi olarak isimlendirilen, endüstriyel faaliyetleri ve özellikle de üretime yönelik uygulamaları büyük oranda arttıran aşamalardan geçilmiştir [1].

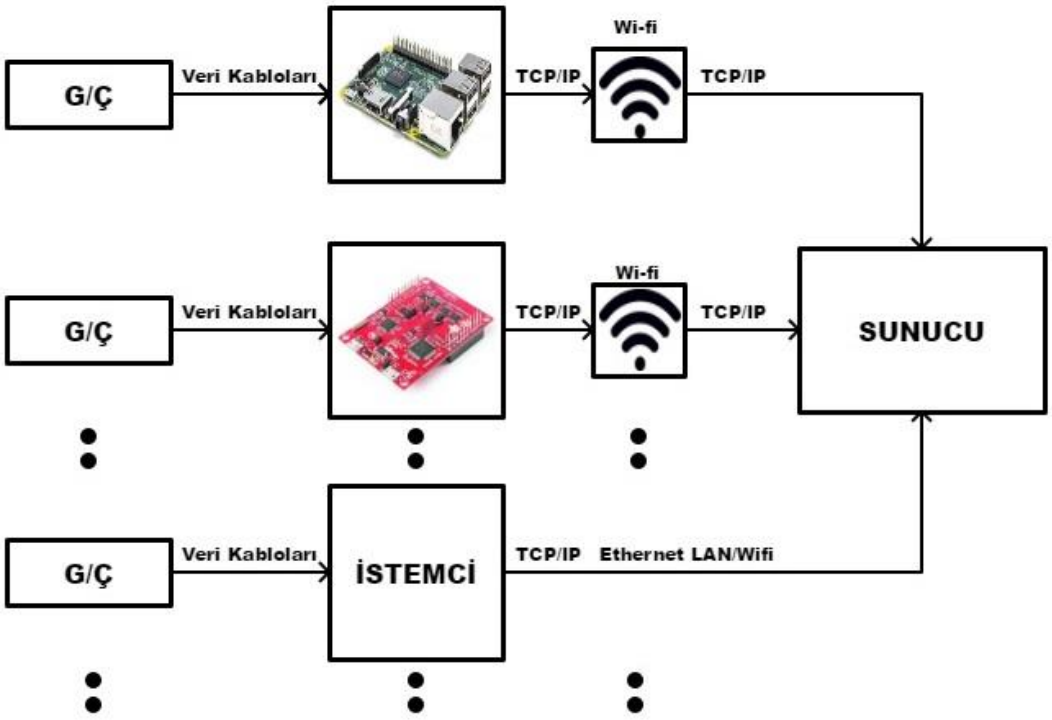
İlk sanayi devrimi su ve buhar gücünü kullanarak mekanik üretim sistemleri ile ortaya çıkmıştır. İkinci sanayi devrimi ile elektrik gücünün yardımıyla seri üretim tanıtılmıştır. Üçüncü sanayi devrimi olarak adlandırılan dijital devrim, elektronik sistemlerin kullanımı ve bilgi teknolojilerinin gelişmesiyle, üretimin daha da otomatik hale geldiği bir yapıya kavuşmuştur. Günümüzde hala devam etmekte olan dördüncü sanayi devrimi ile nesnelerin interneti (IoT) kavramı ortaya çıkmıştır. Nesnelerin interneti ile siber-fiziksel sistemler birbirleriyle ve insanlarla gerçek zamanlı olarak iletişime geçip iş birliği içerisinde çalışabileceklerdir. Otonom çalışan robotlar, büyük veri, nesnelerin interneti, siber güvenliği, bulut işlemleri, eklemeli imalat yöntemleri, sanal gerçeklik, sistem bütünleşmesi ve benzetimler ile dijital devrim olarak adlandırılan Endüstri 4.0 devrimi başlamıştır [2].

Endüstri 4.0 ile birlikte sistemlerin izlenmesinin ve arıza teşhisinin kolaylaştırılması, sistemlerin ve bileşenlerin öz farkındalık kazanması, sistemin çevre dostu ve kaynak tasarrufu hassasiyeti ile sürdürülebilir olması, yüksek verimliliğin sağlanması, üretimde esnekliğin arttırılması, maliyetin azalması ve yeni hizmet iş modellerinin geliştirilmesi gibi birçok avantaj sanayilere kazandırılmaktadır [2].

Bu bağlamda Endüstri 4.0 sanayi devrimi kapsamında kurulan sanayilerde birbirleriyle haberleşen akıllı cihazlar ve makineler kullanılmaktadır. Bu makinelerin durumlarını incelemek ve kontrollerinin sağlanması için genel izleme ve kontrol merkezleri kurulmaktadır. Makinelerin başında beklemeksizin gömülü sistem elektronik kartları ile makinelerin hataya geçme durumları fark edilmektedir. Hata durumları uzaktan kontrol edilerek giderilmekte veya makine başına personel gönderilmektedir. Makinelerin ürettikleri hatalı veya hatasız ürün miktarları, makinelerin çalışma verileri gibi birçok veri elektronik kartlar tarafından toplanıp, internet aracılığı ile ilgili kontrol merkezine gönderilmektedir. Gönderilen bu verilerin anlık olarak incelenmesi ve kayıt altına alınması üretimin verimliliği ve maliyeti açısından hayati önem taşımaktadır.

Çok geniş bir alanda kullanılabilen IoT teknolojisi çevre ve alt yapı izleme, endüstriyel uygulamalar, enerji yönetimi, medikal endüstri, ev ve bina otomasyonu, taşımacılık ve gıda sektörü gibi birçok sektörde kullanılmaktadır [3].

Bu tezde, otomasyon sistemleri ile beraber çalışmaya uygun ve birçok endüstriyel sektöre uyarlanabilen, genel amaçlı bir IoT sistemi tasarlanmıştır. Geliştirilen sistem, kolay eklemeli mimari ile oluşturulmuş ana sunucu yazılımı sayesinde, donanımdan ve yazılım dilinden bağımsız olarak, her çeşit gömülü sistem kartını yeni bir uygulama protokolü ile sunucuya aktarma imkânı vermiştir. Ana sunucu yazılımı değiştirilmeksizin, sunucuya herhangi bir zamanda yeni bir istemci bağlanabilmektedir. Sunucu üzerinden tüm makinelerin durumları izlenmiş ve kontrol edilmiş olup, sunucuya internet üzerinden gelen tüm sensör verileri, yerel veri tabanına kaydedilmiştir. Sınırsız sayıda istemci, asenkron ve dinamik olarak sunucu ile haberleşebilmektedir. Tezde geliştirilen mevcut istemcilerin yerine, yeni bir istemci eklenmek istendiğinde, sistemin haberleşme kurallarına uyulması yeterli olacaktır. Böylece hem maliyetten, hem de zamandan kazandıran yeni bir IoT çözümü sunulmuştur. Oluşturulan IoT sisteminin şeması Şekil 1.1.'de gösterilmiştir.



Şekil 1.1. Oluşturulan IoT sisteminin şeması

Tezin ikinci bölümünde sorunun tanımı ve sorunun çözümüne yönelik literatürde yapılmış çalışmalara değinilmiştir. Bu çalışmanın daha önce yapılmış çalışmalara göre üstünlüklerine ve yeniliklerine yer verilmiştir.

Tezin üçüncü bölümünde kullanılan gömülü sistem kartlarından, yazılım dillerinden ve istemci sunucu haberleşme sistemlerinden bahsedilmiştir.

Dördüncü kısımda LabVIEW ve gömülü sistem tabanlı yeni bir IoT çözümü için yapılan geliştirme aşamaları ve sistemin işleyişinden bahsedilmiştir.

Beşinci kısmında ise yapılan deneysel çalışmaların sonuçlarından, sistemin üstünlüklerinden ve gelecekte nasıl geliştirilebileceğinden bahsedilmiştir.

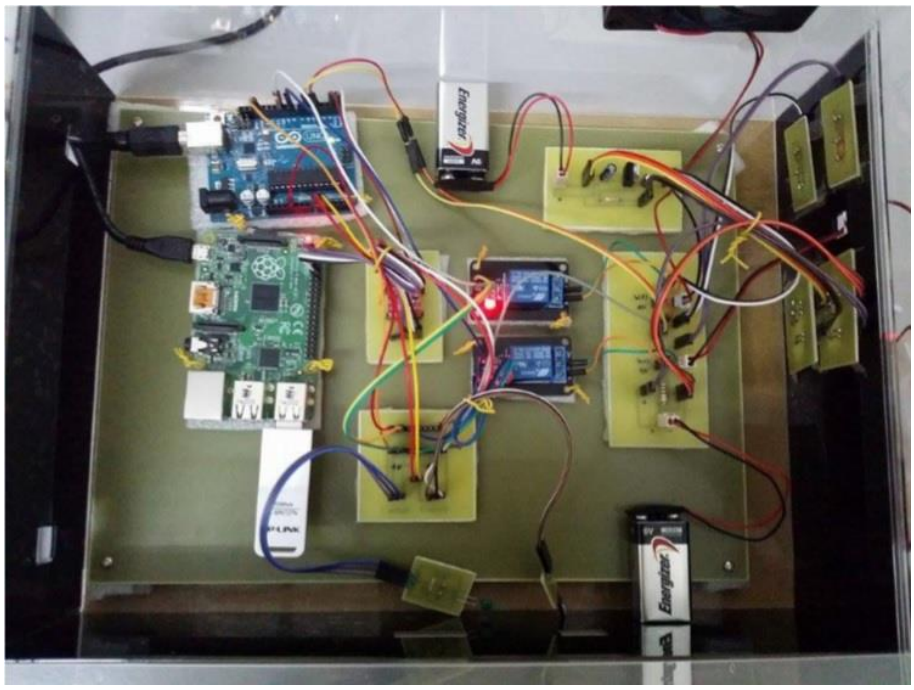
BÖLÜM 2. KAYNAK ARAŞTIRMASI

Nesnelerin interneti son yıllarda popülerliği artan bir konudur. Sanayinin gelişmesiyle birlikte, makinelerin birbiri ile haberleşmesi ve verilerinin yönetim merkezine anlık olarak aktırılması bir ihtiyaç haline gelmiştir. Üretimi insan elinden çıkarıp daha hızlı, hatasız, esnek ve verimli hale getirmek, evrensel rekabet için önem arz etmektedir. Endüstri 4.0 ile birlikte nesnelerin interneti üzerine akademik çalışmalar da artmaya başlamıştır.

IoT teknolojisinde her nesne için çeşitli algılayıcılar kullanarak, Zigbee, Kızıl ötesi, Wi-fi ve Wimax gibi kablosuz iletişim yolları ile nesneler hakkında veriler elde edilmektedir. Nesnelerin interneti kavramının makineler arası iletişim (M2M) kavramından türediği düşünülmektedir. M2M teknolojisinde insan müdahalesine gerek kalmadan makineler birbirleriyle iletişimde bulunabilirler. IoT, M2M teknolojisinden daha kapsamlı bir teknolojidir. IoT teknolojisinde M2M etkileşimine insan-makine etkileşimi de dâhil olabilir [3].

Nesnelerin interneti henüz gelişmekte olan bir teknolojidir. IoT teknolojisi kapsamında istenen amaçlara uygun olarak cihazların birbirleriyle haberleşmeleri için pek çok firma tarafından geliştirilen, bazı katmanlı IoT mimari önerileri bulunmaktadır. Ancak, henüz kabul edilmiş standart bir katmanlı mimari modeli bulunmamaktadır. 2020'ye kadar 25 milyar nesnenin internete bağlanacağı beklendiğinden, mevcut durumda IPv4 sisteminin IP adres yetersizliği bulunmaktadır. 32 bitlik adres alanı olan IPv4 sisteminin yerine, 128 bitlik IPv6 sistemine geçişe ihtiyaç duyulmuştur [3]. Bu durumda ortaya birçok IoT uygulama, ağ ve iletim katmanı çıkmaktadır. Her teknolojinin birbirine göre avantajı veya dezavantajı bulunmaktadır. İhtiyaca yönelik en uygun mimari seçilerek akademik çalışmalar ve uygulamalar yapılmaktadır.

Chuan ve Ruslan [4] yaptıkları çalışmada medikal bir deponun içindeki tıbbi malzemeleri uygun sıcaklık ve nem sınırlarında tutmak için uzaktan kontrol ve izleme sistemi geliştirmişlerdir. Çalışmalarındaki kontrol ve izleme sisteminde LabVIEW yazılımını kullanarak bir arayüz oluşturmuşlardır. Sıcaklık ve nem sensör verilerini Arduino Uno gömülü sistem kartı ile toplayıp, Raspberry Pi kartına aktarmışlardır. Raspberry Pi kartını kullanma amaçları Wi-fi aracılığıyla verileri LabVIEW arayüzüne göndermektir. Ayrıca verilerini Excel formatında kayıt altına almışlar, hata durumlarında kullanıcıları e-mail ve SMS yoluyla uyarmışlardır. Raspberry Pi kartının sistem yazılımını LabVIEW LINX kütüphanesini kullanarak gerçekleştirmişlerdir. Statik yapı kullanarak sadece 1-1 yapıda TCP/IP protokolünü kullanmışlardır. Yaptıkları çalışmada hedeflerine ulaşarak tıbbi malzemeleri beklenilen sıcaklık ve nemde tutmayı başarmışlardır. Şekil 2.1.'de Chuan ve Ruslan'ın geliştirdikleri sistem gösterilmiştir.



Şekil 2.1. Chuan ve Ruslan'ın geliştirdiği sistem [4]

Kaya ve arkadaşları [5] gerçekleştirdikleri çalışmada, bir kablo fabrikasının 8 adet üretim hattına kurmuş oldukları PLC, PC ve LED tabelalar ile yönetim birimine esnek otomasyon sistemli veri takip sisteminin tasarımını ve uygulamasını

gerçekleştirmişlerdir. Bu sistemde yönetim biriminde LabVIEW yazılımı ile sunucu oluşturmuşlar, TCP/IP protokolünü kullanarak her üretim hattındaki istemcilerden ürün takip verilerinin izlenmesini başarmışlardır. Her üretim hattından PLC ile sensörlerden verileri toplayıp, hat başındaki LabVIEW istemci yazılım arayüzünün çalıştığı PC'ye verileri seri haberleşme ile aktarmışlardır. İstemci PC'ler aldıkları verileri internet üzerinden sunucu PC'ye aktarmışlardır. Hem hat başında, hem de yönetim merkezinde monitörler üzerinden tüm verileri izlemişler ve kontrol etmişlerdir. Şekil 2.2.'de Kaya ve arkadaşlarının üretim hatlarına kurdukları sistem gösterilmiştir.

Şekil 2.2. Kaya ve arkadaşlarının üretim hatlarına kurduğu sistem [5]

Bolivar ve Silva [6] ise güneş radyasyonunu izleme sistemi kurmuşlardır. Bu çalışmada PIC 16F877 mikrodenetleyici yardımıyla optik sensörden güneş ışınlarını

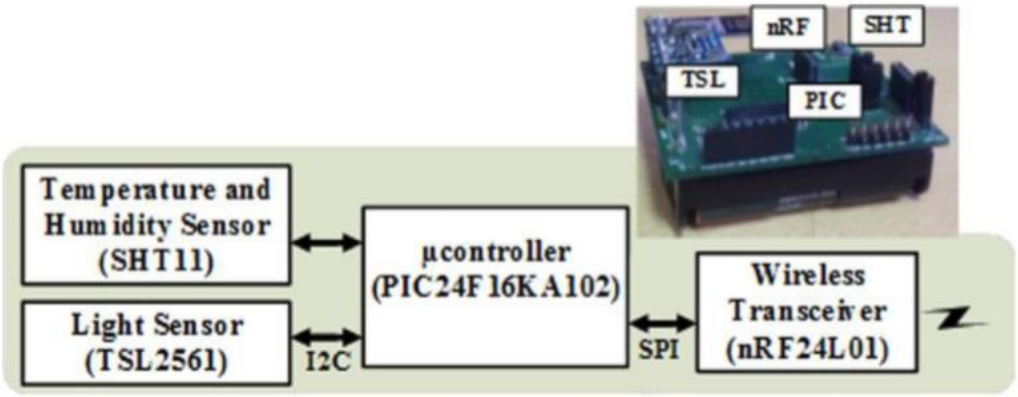
ölçmüşlerdir. Ölçülmüş değeri Raspberry Pi gömülü sistem kartına aktararak (User Datagram Protocol) UDP haberleşme protokolü üzerinden sunucuya iletmişlerdir. Sunucu verilerini hem yerel hem de bulut tabanlı veri depolama sistemlerine kaydetmişlerdir. PIC ile Raspberry Pi haberleşmesini (Inter-Integrated Circuit) I2C seri haberleşme protokolünü kullanarak gerçekleştirmişlerdir. Raspberry Pi'den aldıkları verileri UDP protokolü ile sunucu arayüzüne yazmış LabVIEW programına göndermişlerdir. Toplanan verileri uzaktan izlemek için hem sunucu arayüzünü, hem de Ubidots bulut arayüzünü kullanmışlardır.

Swain ve arkadaşları [7] yaptıkları çalışmada, güvenlik gerektiren bölgelere düşmanı uyarmadan ulaşabilen otonom bir taşıt tasarlamışladır. Bu taşıt, güvenlik görevlileri için sınır güvenlik bölgelerinde veya savaş alanlarında bomba tespiti ve imhası, tehdit teşhisi gibi işlemlerde kullanılmaktadır. Ana kontrol merkezinde Raspberry Pi kartı kullanılarak birçok sensörü sisteme bağlanmışlardır. Kamera da dâhil olmak üzere birçok sensör verisini uzaktan kontrol ve izleme için LabVIEW ile oluşturulmuş arayüzü kullanmışlardır. Raspberry Pi yazılımını python script dili ile yazmışlardır. Arayüz ile kartın haberleşmesini TCP/IP protokolü üzerinden gerçekleştirmişlerdir.

Vujovic ve Maksimovic [8] ise çalışmalarında Raspberry Pi kartını kullanarak ev otomasyon sistemi gerçekleştirmişlerdir. Nesnelerin interneti kapsamında her RPI kartını sensör düğümü olarak kullanmışlardır. Farklı sensörlerden verileri toplayarak web ortamına aktarmışlardır. HTML, CCS, Javascript gibi dillerle oluşturmuş oldukları web sitesine verileri aktarmışlardır. RPI donanım kartı yazılımını Python dilinde yazmışlardır.

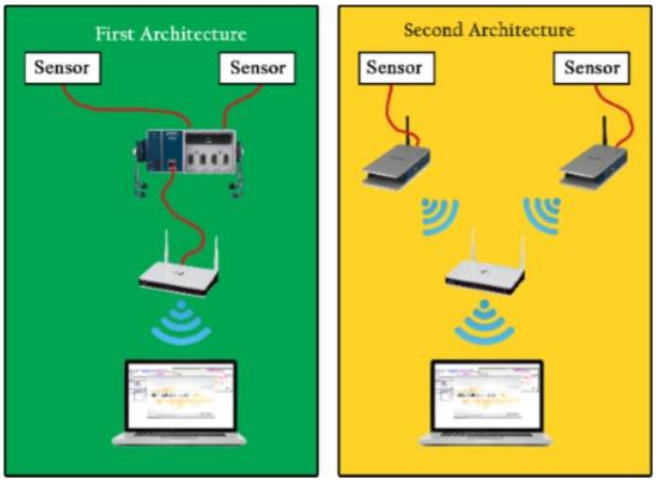
Jegan ve Nimi [9] gerçekleştirdikleri çalışmada ECG sinyalinin işlenmesi için, filtreleme tekniklerini kullanarak elde ettikleri ölçümleri, TCP/IP üzerinden aktararak LabVIEW yazılımı ile izlemişlerdir. Hastadan sensörler aracılığı ile alınan ECG sinyallerini ileri seviye filtreleme yöntemleri ile gürültüleri kaldırarak filtrelemişlerdir. Filtrelenmiş ECG sinyalleri internet üzerinden LabVIEW ile oluşturulmuş arayüze anlık olarak göndermişlerdir.

Shah ve Mishra [10] yaptıkları çalışmada akıllı binalar için kablosuz algılama ve izlemeyi mümkün kılan özelleştirilmiş bir IoT sistemi tasarlamışlardır. Bu çalışmada bina otomasyonunun ışık, nem ve sıcaklık gibi verilerini toplamak için PIC 24F16KA102 mikrodenetleyicisini kullanmışlardır. nRF24L01 RF haberleşme modülü ile PIC, SPI üzerinden haberleştirerek verileri, kablosuz olarak LabVIEW izleme ekranına iletmişlerdir. Şekil 2.3.'te IoT sisteminin kablosuz sensör düğümü gösterilmiştir.



Şekil 2.3. Shah ve Mishra IoT sisteminin kablosuz sensör düğümü [10]

Hnidka ve Rozehnal [11] yaptıkları çalışmada, LabVIEW programında oluşturdukları yazılım ile TCP/IP haberleşme protokolünü kullanarak Wi-fi üzerinden gerilim sensörü verilerini uzaktan izlemişlerdir. NI DAQ kartları ile toplandıkları sensör verilerini Wi-fi ile GUI'ye göndermişlerdir. Gerçek zamanlı toplanan verileri TDMS formatında ileride kullanılmak üzere kayıt altına almışlardır. Bu sistemi hem cDAQ kasasıyla toplayıp modem aracılığı ile hem de ayrı ayrı sensörlerdeki Wi-fi bulunduran DAQ kasaları ile sunucuya yollamışladır. Bu iki sistemin avantaj ve dezavantajlarını kıyaslamışlardır. Şekil 2.4.'te iki ayrı sistemin karşılaştırılması gösterilmiştir. Yapılan çalışmalar 1-1 yapıda kurulmuş olup, statik yapıda IoT mimarileri oluşturmuşlardır.



Şekil 2.4. Hnidka ve Rozehnal çalışmasındaki iki sistemin karşılaştırılması [11]

Gómez ve arkadaşları [12] çevresel değişkenlerle ilgili bilgi toplama kapasitesine sahip sensörlerin kullanımına olanak tanıyan ve aynı zamanda ilgili diğer sensörlerle kolay entegrasyona imkân sağlayan nesnelerin interneti (IoT) temelinde bir mimari geliştirmeyi amaçlamışlardır.

Prada ve arkadaşları [13] kaynak kısıtlamalı cihazlarla iletişim için (Message Queuing Telemetry Transport) MQTT protokolünü önermişlerdir. Özelikle Javascript gibi web standartlarını kullananlar için yeni cihazların sisteme kolayca entegre edilebilmesi adına yeni bir araç önermişlerdir.

Oksanen ve arkadaşları [14] mobil tarım makinelerinin uzaktan izlenmesi için endüstriyel otomasyon protokolü olan OPC UA (Open Platform Communications Unified Architecture) teknolojisini IoT sistemlerine adapte etme üzerine bir çalışma gerçekleştirmişlerdir. İnternet bağlantısı üzerinden tespit edilen uçtan uca gecikme süresini 250 milisaniye altında tutarak, telemetri uygulamasını başarı ile tamamlamışlardır.

Shah ve Bharadi [15] yaptıkları çalışmada IoT cihazı olarak kullandıkları Raspberry Pi kartı ile biyometrik verileri toplayıp, RSA ve AES-256 kriptolama algoritmalarını kullanarak topladıkları verileri şifrelemişlerdir. Şifrelenen biyometrik verileri Azure

bulut sistemine göndermişlerdir. Böylece bilgi güvenliğini sağlayarak bulut sisteminde biyometrik verileri güvenli bir şekilde tutmayı başarmışlardır.

2.1. Tezin Literatüre Katkısı

Mevcut literatürde esnek, çoğaltılabilir ve genişletilebilir gömülü sistem tabanlı bir IoT çözümü önerilmemiştir. Bu çalışmanın en önemli özelliklerinden birisi TCP/IP alt katman protokolünü ve yeni ekle-çıkar protokolünü kullanan herhangi bir gömülü sistem donanımının, ana sunucunun yazılımını değiştirmeden sisteme kolayca bağlanabilmiş olmasıdır. 1-N eklemeli, esnek bir mimaride kurulmuş bu IoT sistemi, sunucu tarafından tüm istemcileri kontrol edebilme, izleyebilme ve verilerini yerel veri tabanlarında tutabilme imkânı sunmuştur.

Çalışmanın ikinci önemli özelliği ise donanımdan ve yazılımdan bağımsız olmasıdır. Yani ister özel gömülü sistem kartı tasarlansın, ister bilinen bir gömülü sistem kartı kullanılsın, belirlenen ekle-çıkar protokolü ile sisteme bağlanılabilme imkânı sunulmaktadır. Kullanılan kartın programlama dilinde TCP/IP fonksiyonlarını ve yeni ekle-çıkar protokolünü kullanmak şartıyla her yazılım dili, sistem tarafından kabul görmektedir.

Çalışmanın bir başka önemli özelliği ise LabVIEW programlama dilinin gücünü kullanarak hızlı, performansı yüksek, asenkron mesajlaşma teknolojisi ve dinamik bir yapıya sahip sunucunun geliştirilmesidir.

Ayrıca LabVIEW Raspberry Pi kütüphanesi kullanılarak, Raspberry Pi kartlarını doğrudan LabVIEW ile programlayarak veya görsel kullanıcı arayüzünü oluşturarak, LabVIEW kodunu Python diline dönüştürme imkânı oluşturulmuştur. Böylece ilk prototipin hızlı ve kolay bir şekilde gerçekleştirilmesi mümkün olmuştur.

BÖLÜM 3. MATERYAL VE YÖNTEM

3.1. Tezde Kullanılan Yazılımlar

Bu bölümde, gerçekleştirilen IoT sisteminde kullanılan yazılım geliştirme araçlarından bahsedilecektir. IoT sisteminde sunucu yazılımı LabVIEW programında yazılmıştır. İstemci yazılımlarında Raspberry Pi için LabVIEW RPI kütüphanesi kullanılmıştır. Bu kütüphane LabVIEW kodunu Python koduna dönüştürmektedir. MSP430 istemcisi için Energia IDE platformu kullanılarak, C/C++ tabanlı Energia dili kullanılmıştır.

3.1.1. LabVIEW ile programlama

Nesnelerin interneti konusu üzerine yapılacak çalışmalarda C/C++, Python, Java gibi birçok programlama dili kullanılabilir. Ancak metin tabanlı programlama dillerinde yazılım geliştirmek uzun süreler alabilmektedir. Bu programlama dillerinde görsel arayüz inşa etmek için bir takım kütüphaneler ya da GUI geliştirme ortamlarına ihtiyaç duyulmaktadır. Python için PyQT veya Tkinter, C++ için QT gibi GUI geliştirme ortamları buna örnek olarak verilebilir.

LabVIEW programı, GUI tasarlamak için kendi bünyesinde barındırdığı ön panel ile hem kolay hem de hızlı bir çözüm üretmektedir. LabVIEW, donanım yapılandırması, veri toplama ve hata ayıklama da dâhil olmak üzere uygulamaların her alanını görselleştirmeye yardımcı olan grafiksel bir programlama yaklaşımı sunmaktadır. Bu görselleştirme, herhangi bir tedarikçiden ölçüm ve veri toplama donanımını sisteme dâhil etmeyi, diyagramda karmaşık mantığı temsil etmeyi, veri analiz algoritmaları geliştirmeyi ve özel mühendislik gerektiren kullanıcı arayüzleri tasarlamayı kolaylaştırmaktadır [16].

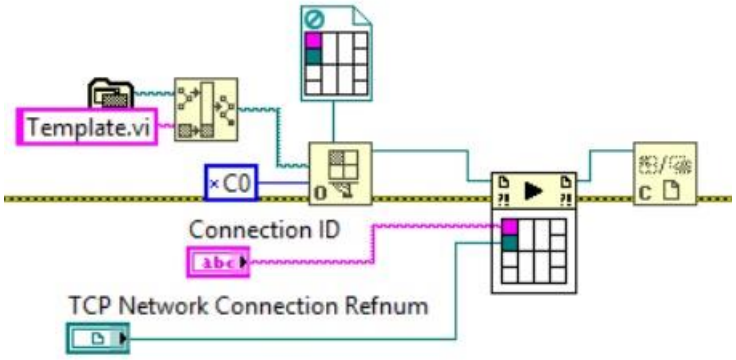
Grafiksel programlama yaklaşımı ile güçlü fonksiyonları ve işlevselliği kullanılarak her ihtiyaca yönelik yazılım LabVIEW ortamında geliştirilebilmektedir. Bu çalışmada dinamik ve hazır eklemeli bir mimari ihtiyacı duyulduğundan sunucu yazılımının LabVIEW ortamında geliştirilmesi uygun görülmüştür. Asenkron mesajlaşma ve haberleşme (AMC) yapısı kullanılarak süreçler arası veri iletişimi gerçekleştirilmiştir. Durumlar mesaj komutlarına göre işlenmiştir.

Sunucu yazılımının çalışma anında, kodu değiştirilmeksizin birçok yeni gömülü sistemin, donanım istemcisinin sisteme eklenebilmesi düşünülerek sunucunun dinamik bir mimariye sahip olması gerekmektedir. LabVIEW programı dinamik yapıya elverişli bir yazılım sistemi sunmaktadır. Dinamik yapıda, asenkron çalışma imkânı sunması gerekmektedir. Yani birçok noktadan çağrı yapılsa bile farklı zamanlarda ve yerlerde süreçlerin kopyasını çalıştırabilmesi gerekmektedir. LabVIEW programlama dilinde her bir pencere ve yazılımı birer VI (Virtual Instruments) olarak adlandırılır.

Bu çalışmada, farklı donanımların sisteme bağlanabilmesi için dinamik ve asenkron çağrılacak şablon bir VI oluşturulmuştur. Şablon VI, kendi bünyesinde görünmez kontrol ve göstergeler bulundurmaktadır. İstemci donanımdan gelen bilgilere göre bu kontrol ve göstergelerin bazılarının isimleri, ölçekleri, birimleri ile görünür hale gelmektedir. Bunların yanında veri toplama hızı ve bağlanan donanımın ismi de şablon VI tarafından elde edilip gösterilmekte ve işlenmektedir. Her şablon VI kendisine ait donanımın verilerini kullanıcıya sunmakta ve arka planda veri tabanına kaydetmektedir. Bu şablon VI ile donanımlar arasında çift yönlü haberleşme sistemi kurulmuştur. Yani şablon VI, verileri donanımdan topladığı gibi kendisi de bazı verileri donanıma göndermektedir. Şablon VI'lar, sunucuya (Asynchronous Message Communication) AMC mimarisi ile bağlanarak haberleşme sağlamaktadır. Sunucu üzerinden şablon VI'lar kontrol edilebilmektedir. Sunucuya bağlanmış donanım kadar şablon VI aktif olarak çalıştırılmakta ve bunlar üzerinde bir takım işlemler, sunucu tarafından gerçekleştirilmektedir.

LabVIEW programında bu şablon VI'ın çoklu kopyasını paralel ve asenkron olarak çağırmak için "Start Asynchronous Call" fonksiyonu kullanılmıştır. Bu fonksiyon konumu belirtilen şablon VI'ı sunucu çalışırken çoklu kopyasını oluşturarak, çağırıp, çalıştırmaktadır [17].

Açılmak istenen şablon VI'ın konumu "Open VI Reference" fonksiyonu ile referansa dönüştürülmüştür. Akabinde bu referans kullanılarak "Start Asynchronous Call" fonksiyonu ile şablon VI'ların çoklu kopyası çalıştırılmıştır. Donanımdan sunucuya gelen başlatma verileri de şablon VI çalıştırılırken girdi olarak aktarılmıştır. Bu veriler; bağlantı bilgileri ve TCP referans numarasıdır. Verileri alan şablon VI, sistemi hazır hale getirip arayüzünü aktif hale getirmektedir. Şekil 3.1.'de şablon VI'ın dinamik olarak çağırılması gösterilmiştir.

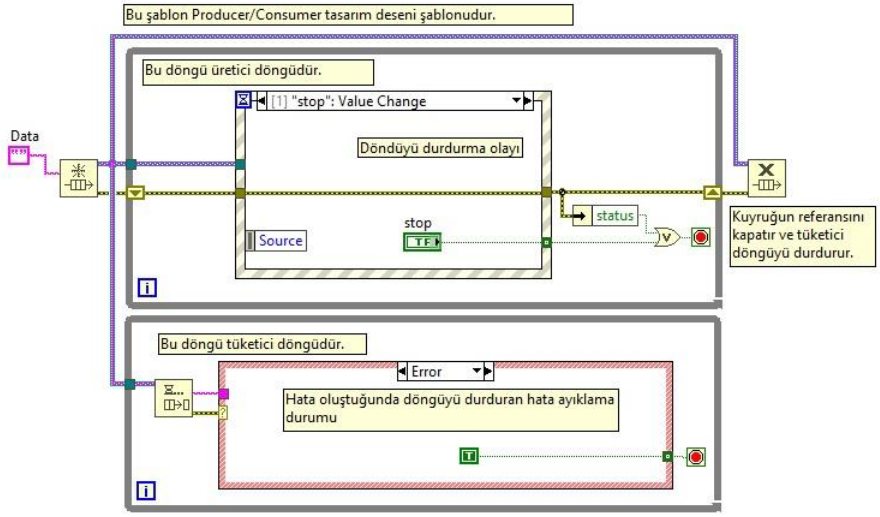


Şekil 3.2. Şablon VI'ın dinamik olarak çağrılması

Sunucu ve şablon VI için Producer/Consumer tasarım şablonu kullanılmaktadır. İhtiyaca yönelik seçilen bu şablonda kullanıcı etkileşimli olayları barındıran Producer döngüsü ve bu olaylara uygun verilerin işlendiği Consumer döngüsü bulunmaktadır. Producer/Consumer tasarım şablonu, bireysel oranlarda yinelenirken aynı anda birden çok işlemi kolaylıkla idare edebilmeyi sağlamaktadır. Bu şablonu benzersiz kılan şey, uygulama süreçleri arasında arabelleğe alınmış iletişimin yararıdır. Farklı hızlarda çalışan birden fazla süreç olduğunda arabelleğe alınmış iletişim de son derece etkili olmaktadır [18].

Ayrı döngülerde üretilen ve işlenen süreçlerin birbirleriyle haberleşmesi için farklı metotlar bulunmaktadır. FGV, evrensel ve yerel değişkenler, bildiriciler, kuyruklar ve kullanıcı olay tutucular bunlardan bazılarıdır [19]. Genellikle kuyruk yapısı Producer/Consumer tasarım şablonunda en çok kullanılan haberleşme metotlarından biridir. Şekil 3.2.'de Producer/Consumer tasarımının örnek şablonu gösterilmiştir.

Bu çalışmada alt yapısını kuyrukların oluşturduğu daha üst haberleşme teknolojisi içeren ve asenkron çalışabilen AMC alt yapısı kullanılmıştır.



Şekil 3.3. Producer/Consumer tasarım deseni şablonu [18]

Çalışmada bu şablonun kullanılma amacı, farklı süreçlerin birbirinden bağımsız olarak farklı zamanlarda ve aynı anda çalışması gerekliliğidir. Sunucu yazılımında kullanıcı etkileşimli üretici bir döngü, bu üretilen verileri işleyen tüketici bir döngü ve TCP bağlantı işlemlerinin ele alındığı başka bir tüketici döngü kullanılmıştır. Bu döngüler birbirleriyle AMC alt yapısı ile haberleşmektedirler. Şablon VI tüketici döngüsünde ise veri toplama ve kayıt etme gibi işlemler yer almaktadır.

AMC, National Instruments firmasının oluşturduğu ancak LabVIEW içerisinde doğrudan bulunmayan bir haberleşme kütüphanesidir. LabVIEW içerisinde doğrudan bulunmayan ancak VIPM (VI Package Manager) programı aracılığı ile yüklenen resmi ya da gayri resmi birçok kütüphane bulunmaktadır. İhtiyaca yönelik

kütüphaneler VIPM programından indirilerek kurulmaktadır. VIPM programı binlerce kütüphaneyi bir araya getirerek standartlaşmayı sağlamıştır. Kullanıcılar veya firmalar tarafından oluşturulan kütüphaneler VIPM programından kolayca indirilip LabVIEW yazılımına dâhil edilebilmektedir. VIPM, kullanıcıların hem kütüphane oluşturmasına hem de kullanmasına ve yeni kütüphaneler keşfetmesine olanak sunmaktadır [20]. LabVIEW ve VIPM programları TCP/IP protokolü ile haberleşmektedirler. Kurulacak kütüphanenin dosyaları LabVIEW klasöründe gerekli konumlara yüklenerek işlemler gerçekleştirilmektedir.

AMC haberleşme kütüphanesi VIPM aracılığı ile indirilip kurulmuştur. AMC kütüphanesi bir süreç içerisinde, süreçler arasında ve ağ üzerinden farklı donanımlar ve uygulamalar arasında mesaj göndermek için kullanılan haberleşme kütüphanesidir [21]. Eşsiz isimler verilerek üretilen kuyruklar, yazılımın her yerinden erişebilecek şekilde kullanıma açılır. İsme göre mesaj alma veya gönderme fonksiyonları kullanılarak haberleşme sağlanır. Mesajlar kendi bünyesinde metin, anahtarlı metin, öncelik sıralaması, özellikler ve parametreler içerir. Bu mesajlar yapılandırılarak haberleşme gerçekleştirilir. Süreçler arası mesajlar aynı veya ayrı isimli kuyruklarla yollanabilir ancak işlenecek durum mesajı, gönderilen süreçte bulunması gerekmektedir. Şekil 3.3.'te süreçler arası haberleşme mantığı gösterilmiştir.

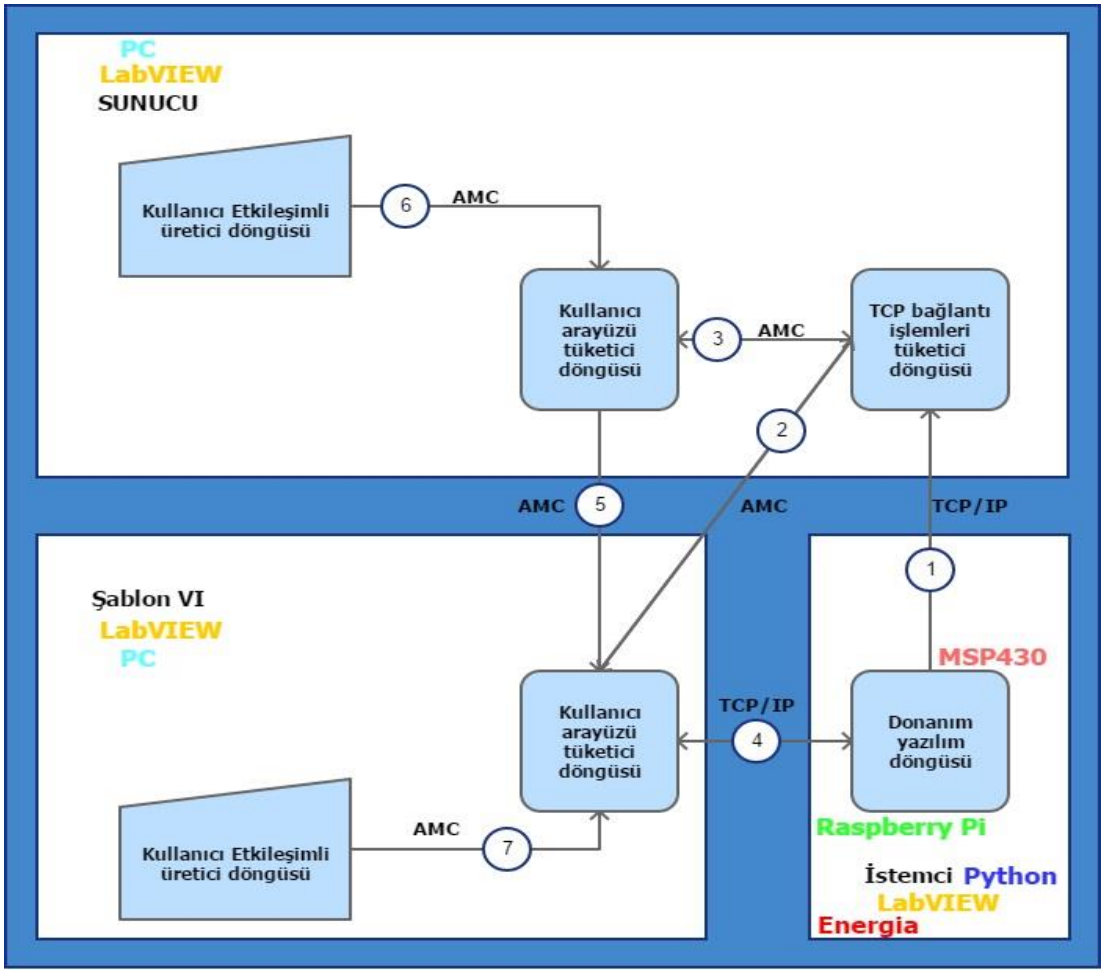
Şekil 4.3. Süreçler arası haberleşme [21]

Bu çalışmada kullanıcı etkileşimli komutlar, tüketici döngülere işlenmek üzere yollanmıştır. Kullanıcı – Şablon VI, Sunucu – Şablon VI ve Kullanıcı – Sunucu haberleşmesi AMC yapısıyla gerçekleştirilmiştir. 1-N bağlantı yapısıyla oluşturulmuş sunucu – istemci IoT sisteminde istemciler, birbirinden bağımsız ancak sunucuya ve kullanıcıya bağımlı olacak şekilde yapılandırılmışlardır.

Sistemin haberleşme yapısında TCP/IP ve AMC protokolleri kullanılmıştır. İlk olarak gömülü sistem kartı TCP/IP protokolü aracılığı ile sunucunun TCP bağlantı döngüsüne bağlanmaktadır. Oluşturulan TCP referans numarası ve kartın bağlantı ismi, dinamik olarak açılan şablon VI'ya "Start Asynchronous Call" fonksiyonu aracılığı ile gönderilmektedir. Gönderilen bağlantı isminden şablon VI kendine yeni bir AMC mesaj kuyruğu yaratmaktadır. Sunucudan şablon VI'ya, kartın TCP/IP yoluyla yollamış olduğu ilk bağlantı verileri (Kart ismi, IO ve isimleri, ölçekleri, veri toplama süresi) kuyruk ismi üzerinden gönderilmektedir.

Bununla birlikte sunucu da kullanıcının istemci üzerinde işlem yapabilmesi için yeni bir istemci bağlantı ismi bilgisi kullanıcı tüketim döngüsüne de gönderilmektedir. Şablon VI kullanıcı arayüzü tüketici döngüsü, sunucudan aldığı bilgileri işleyerek arayüze gerekli olan dijital analog girdi ve çıktıları, isim ve ölçekleri ile birlikte görünür hale getirmektedir. Veri toplama hızı da döngüye aktarılmakta ve arayüzün başlığına bağlanan kartın ismi yazılmaktadır. Daha sonra istemci kart ve şablon VI arasında istenen aralıklarla veri aktarımı TCP/IP protokolü aracılığı ile gerçekleştirilebilmektedir. Şablon VI veri toplarken aynı zamanda veri tabanına kayıt işlemlerini de gerçekleştirmektedir.

Kullanıcı, Şablon VI arayüzünde bir takım işlemler yapabilmektedir. Kullanıcı etkileşimli döngü yardımıyla şablon VI kullanıcı tüketici döngüsü arasında AMC üzerinden iletişim kurulmuştur. Aynı şekilde sunucu kullanıcı etkileşimli döngü ile tüm kullanıcı tüketici döngüleri arasında yine AMC üzerinden iletişim kurulmuştur. Şekil 3.4.'te sistemin mesajlaşma diyagramı gösterilmiştir.



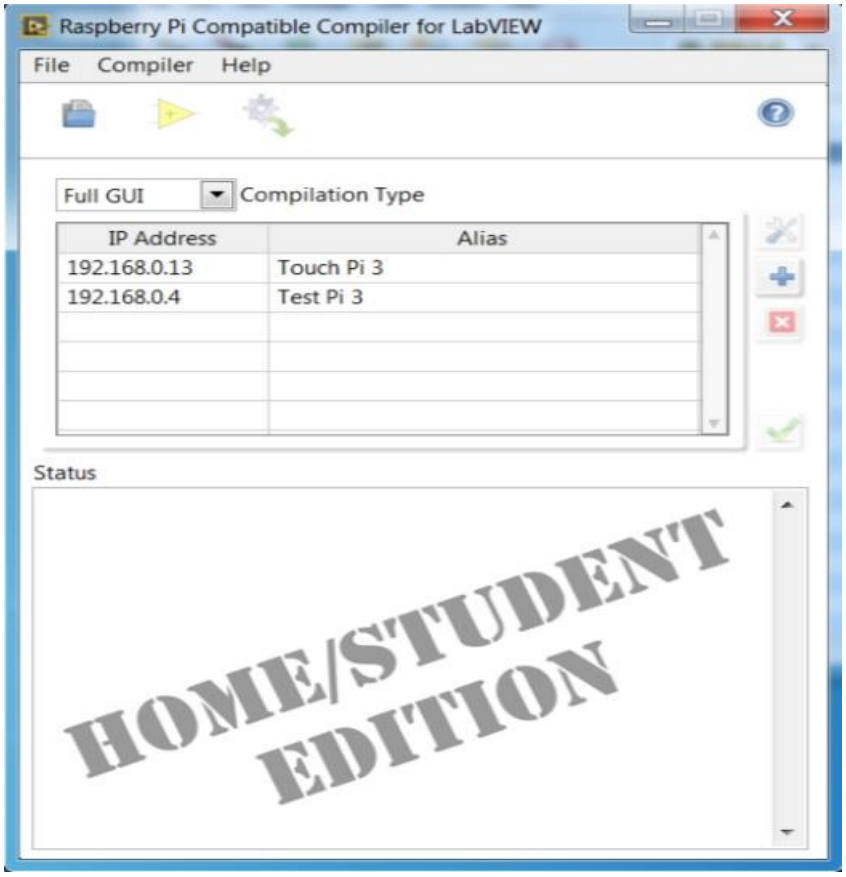
Şekil 3.4. Geliştirilen IoT sisteminin mesajlaşma diyagramı

3.1.1.1. LabVIEW Raspberry Pi Kütüphanesi

LabVIEW Raspberry Pi, LabVIEW programında yazılmış kodu Python diline dönüştüren ve yürütülebilir uygulama halinde Raspberry Pi kartı için hazırlayan bir derleyicidir. Labview Raspberry Pi kütüphanesi, VIPM aracılığı ile yüklendikten sonra LabVIEW menülerinde kullanılabilir hale gelmektedir. LabVIEW RPI, LabVIEW'in tüm yerel fonksiyonlarını desteklemese de birçok fonksiyonu desteklemektedir.

Derleyici RPI'ye TCP/IP aracılığı ile bağlanmaktadır. İstemci VI yazılımı yazıldıktan sonra derleyicide açılmıştır. Raspberry Pi, derleyiciye bağlanmıştır.

Derleme işlemi için iki işlem sunulmaktadır. Bunlardan birincisi Console, diğeri Full GUI işlemleridir [22]. Labview for RPI derleyicisi Şekil 3.5.'te gösterilmiştir.

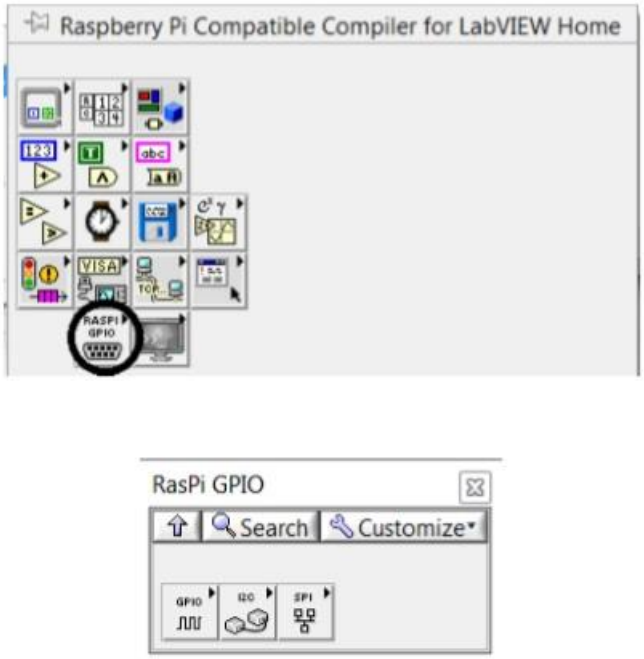


Şekil 3.5. LabVIEW for Raspberry Pi derleyicisi [22]

Console işlemi sadece LabVIEW blok diyagram kodunu dönüştürerek Raspberry Pi'ye yükler. Bu uygulama konsoldan çağırılarak çalıştırılır. Diğer Full GUI işlemi ise LabVIEW programının hem ön panelini hem de blok diyagramını derler ve yükler. Bu uygulama açıldığında Raspberry Pi için bir arayüz açılır. LabVIEW arayüzünün Raspberry Pi de çalıştırılması için bu işlem seçilmelidir.

Kütüphane TSXperts firması tarafından yazılmış olup ücretli bir kütüphanedir. İki adet lisansı mevcuttur. Birincisi ev sürümü ikincisi ticari sürümüdür. Ev sürümü akademik ve ticari olmayan hobi amaçlı çalışmalar için kullanılmaktadır. Ticari sürümü ise ticari uygulamalar da kullanılmak üzere lisanslanmaktadır [22].

Bu çalışmada FULL GUI işlemi kullanılmıştır. Uygulama için kütüphanenin 7 günlük ücretsiz deneme sürümü kullanılmıştır. İstemcinin arayüzünden yapılan işlemler incelenmiştir. İstemci yazılımı Labview ortamında yazılmıştır. Full GUI işlemiyle Raspberry Pi kartına gömülerek çalıştırılmıştır. Şekil 3.6.'da kütüphanenin blok diyagram menüsü gösterilmiştir.



Şekil 3.6. LabVIEW for Raspberry Pi Kütüphanenin menüleri [22]

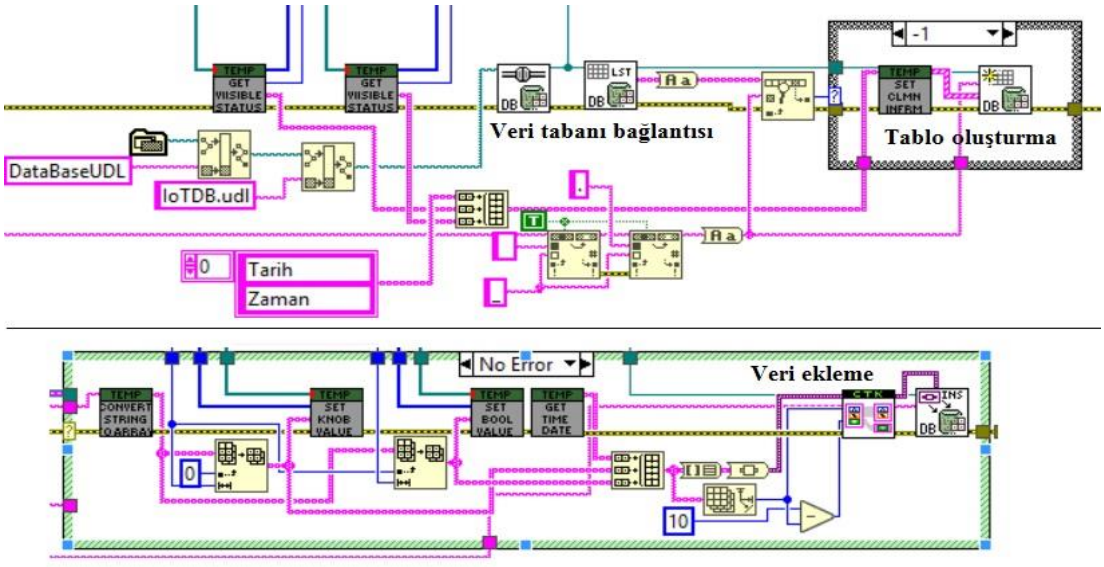
3.1.1.2. MariaDB veri tabanı

MySQL/MariaDB bir ilişkisel veri tabanı yönetim sistemidir. Veri tabanı yönetim sistemi, veri tabanlarını yaratmak, kullanmak, değiştirmek ve veri tabanı sistemleri ile ilgili her türlü işletimsel gereksinimleri karşılamak için tasarlanmış sistem ve yazılımdır. İlişkisel yönetim sistemi ise ilişkisel veri tabanını çeşitli tablolar arasında organize edilmiş verilerden oluşan veri tabanı olarak açıklanmaktadır. Bu farklı tablolar arasındaki veriler, çeşitli anahtarlar vasıtası ile birbirlerine bağlanırlar. İlgili tablolarda, sütunlar arasında bir anahtar sütün yer alır. Bu anahtar sütün aracılığı ile birden çok tablo verileri birbirleriyle bağlantı sağlamaktadır. Herhangi bir sorgulamada birlikte görüntülenmektedir [23].

MariaDB Server dünyanın en popüler veri tabanı sunucularından biridir. MySQL'in orijinal geliştiricileri tarafından oluşturulmuştur ve açık kaynak kodlu kalması garanti edilmiştir [24].

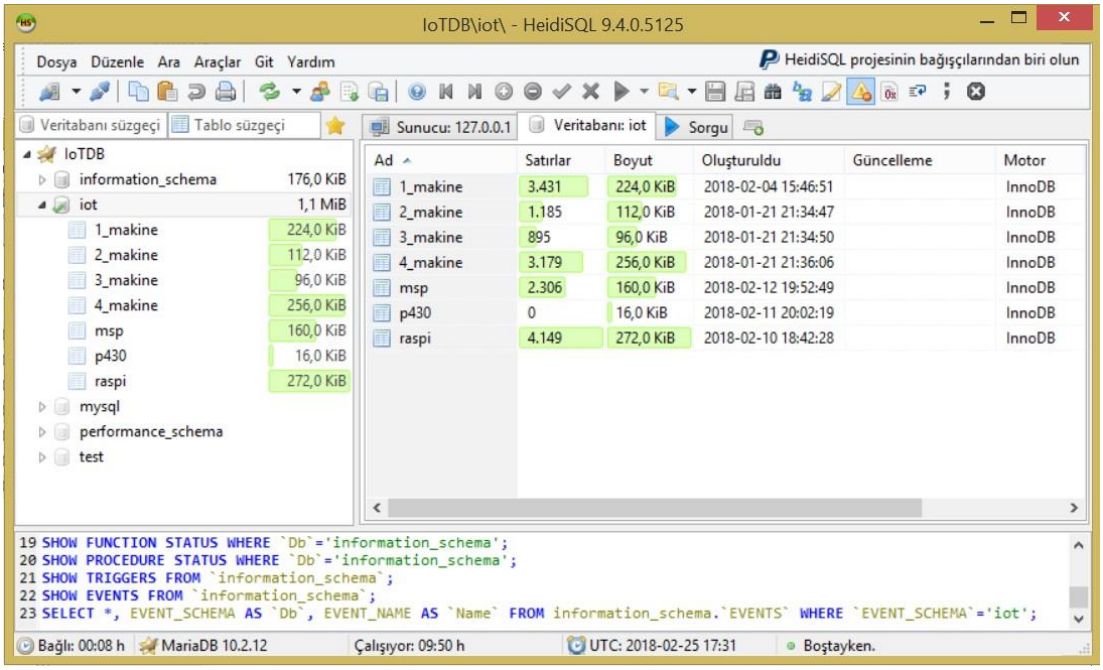
Bu çalışmada MariaDB veri tabanı yönetim sistemi kullanılmıştır. MariaDB sunucusunun yüklenmesi ile birlikte MariaDB veri tabanı yönetim sisteminin sunmuş olduğu ODBC sürücüsü yüklenmiştir. Buradaki önemli nokta LabVIEW yazılımının desteklemiş olduğu bit sayısındaki sürücüyü yüklemektir. Bu çalışmada, LabVIEW 2016 32 bit destekli geliştirme ortamı kullanılmıştır. ODBC sürücüsü de 32 bit seçilerek 3.0.3 sürümü yüklenmiştir.

LabVIEW programında Database Connectivity Kütüphanesi yüklenmiştir. Bu kütüphane sayesinde veri tabanına erişim sağlanarak veriler kaydedilmiş ve okunmuştur. Sisteme bağlanan her istemci için otomatik olarak ayrı bir tablo oluşturulmuştur. Bu tablonun sütunları veri isimlerinden oluşmaktadır. Veri isimlerine ek olarak tarih ve saatte kayıt altına alınmaktadır. Bağlanan bir istemci sisteme tekrar aynı isimle bağlandığında yeni bir tablo oluşturulmamakta ve var olan tabloya veriler aktarılmaktadır. Ancak yeni bir isimle bağlandığı takdirde ayrı bir tablo oluşturulacak yeni gelen veriler bu tabloya kaydedilecektir. Şekil 3.7.'de Database Connectivity kütüphanesi ile yazılmış veri tabanı işlemleri gösterilmiştir.



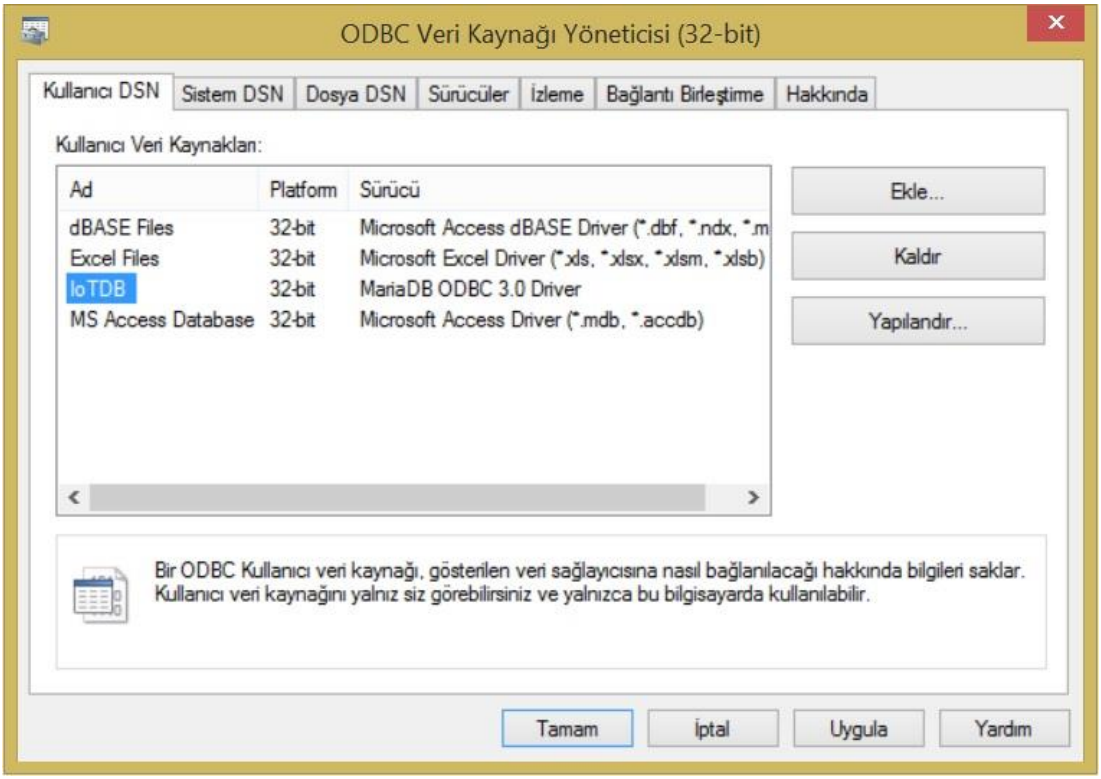
Şekil 3.7. Veri tabanı bağlantısı, tablo oluşturma ve tabloya veri ekleme kodu

Tablo isimlerinde özel karakterlere ve boşluğa yer verilmediği için sunucu otomatik olarak bu karakterleri "_" karakterine çevirmiştir. Örneğin 1.Makine isimli bir istemcinin verileri, veri tabanında 1_makine isminde bir tabloya kaydedilmiştir. Büyük karakterler küçük karaktere dönüştürülmüştür. İstemci verilerine sunucu yazılımından ulaşılabildiği gibi HeidiSQL veri tabanı istemcisinden de ulaşılabilme imkânı vardır. HeidiSQL veri tabanı istemcisi ücretsiz bir istemcidir. MariaDB kurulduğunda yanında ek olarak kurulmaktadır. HeidiSQL veri tabanı istemcisi MySQL, MariaDB, Microsoft SQL, PostgreSQL gibi veri tabanı yönetim sistemlerini kullanan geliştiricilerin kullanmış olduğu güvenilir ve yararlı bir veri tabanı istemcisidir [25]. Şekil 3.8.'de HeidiSQL istemcisinin sistem verilerine erişim ekranı gösterilmiştir.



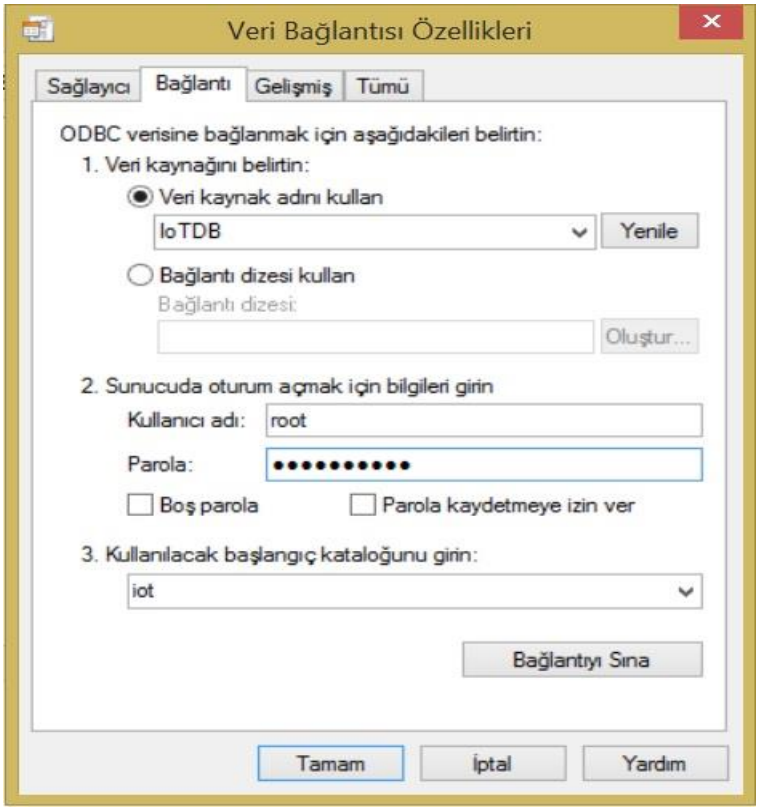
Şekil 3.8. HeidiSQL tarafından istemci verilerine erişim

ODBC sürücüsünde veri tabanı için bir takım ayarlamalar yapılmıştır. ODBC veri tabanı seçilerek kaydı oluşturulmuştur. MariaDB sunucusunda IoTDB adında oluşturduğumuz veri tabanı kaynak olarak ODBC veri kaynağı yöneticisinden belirlenmiştir. ODBC sürücüsü LabVIEW ve MariaDB arasındaki bağlantının kurulması için açık veri tabanı bağlantısıdır. ODBC sürücüsü veri kaynağı yöneticisi arayüzü Şekil 3.9.'da gösterilmiştir.



Şekil 3.9. ODBC sürücü ayarları

LabVIEW programında Tools menüsünün Create Data Link sekmesinden Veri bağlantısı oluşturma sayfası açılmıştır. Bu sayfada UDL dosyası oluşturulmuştur. UDL, evrensel veri bağlantı dosyasıdır. LabVIEW veri tabanı bağlantısı, veri tabanı kütüphanesindeki veri tabanı oluşturma fonksiyonu tarafından UDL dosyası okunarak oluşturulmuştur. Bu sayfada veri kaynağı seçilmektedir. Veri tabanı için kullanıcı adı ve şifre girilir. Son olarak başlangıç kataloğu seçilerek bağlantı test edilmektedir. Bağlantı başarıyla gerçekleştirildikten sonra UDL dosyası için konum seçilmektedir. Bu konumdan UDL dosyası, sunucu yazılımdan okunarak veri tabanı işlemleri gerçekleştirilmiştir. Şekil 3.10.'da veri bağlantısı oluşturma sayfası gösterilmiştir.

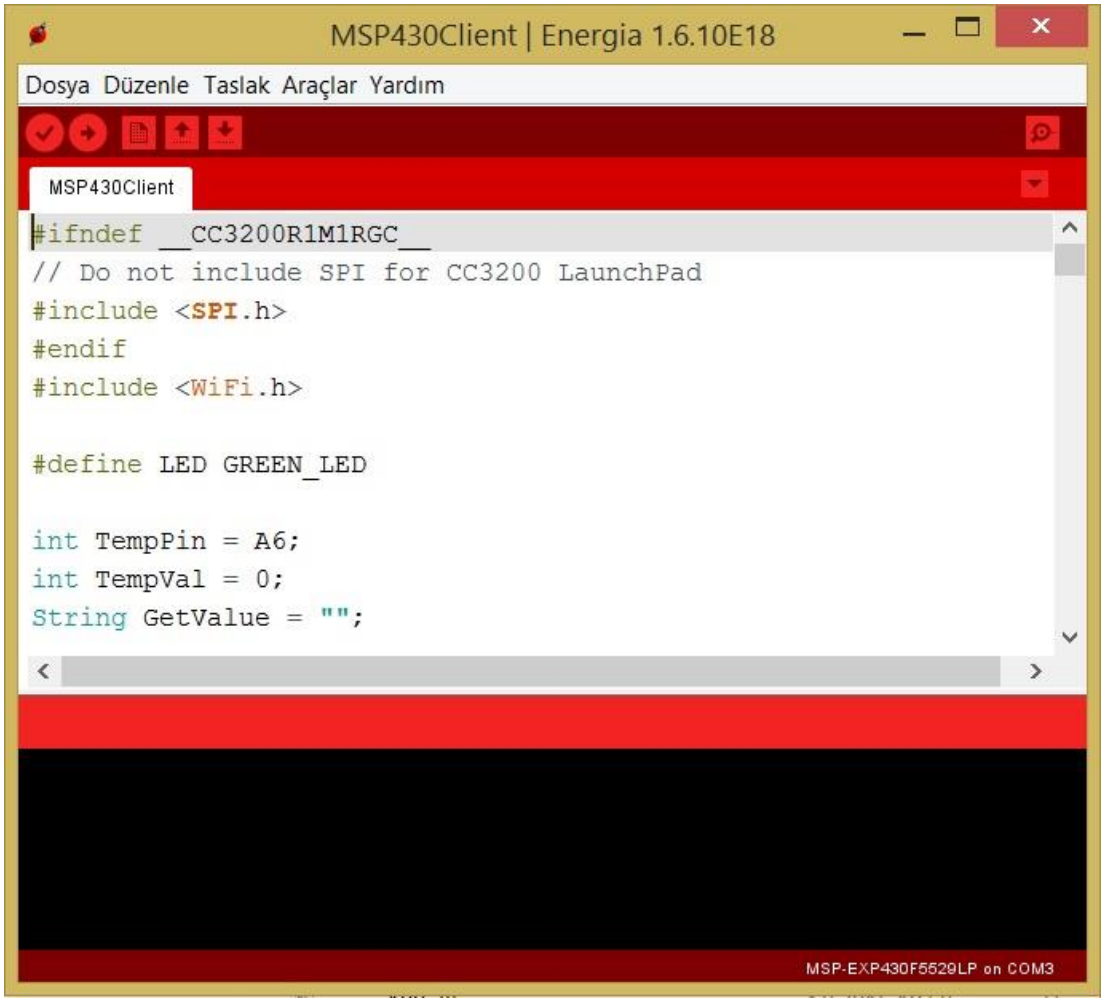


Şekil 3.10. Veri bağlantısı dosyası oluşturma sayfası

3.1.2. Energia ile programlama

Energia, Arduino IDE (Integrated Development Environment) tabanlı bir geliştirme ortamıdır. Energia ile MSP430 ve 432, CC3200 gibi Launchplad serileri ile CC3100 Wi-fi ve eğitim amaçlı Boosterpack serileri programlanabilmektedir. Energia da Arduino yazılımı gibi taslak içerir [26]. Ayarları, işlem ve yazılım mantığı aynıdır. Energia ile yazılım geliştirme ortamı Şekil 3.11.'de gösterilmiştir.

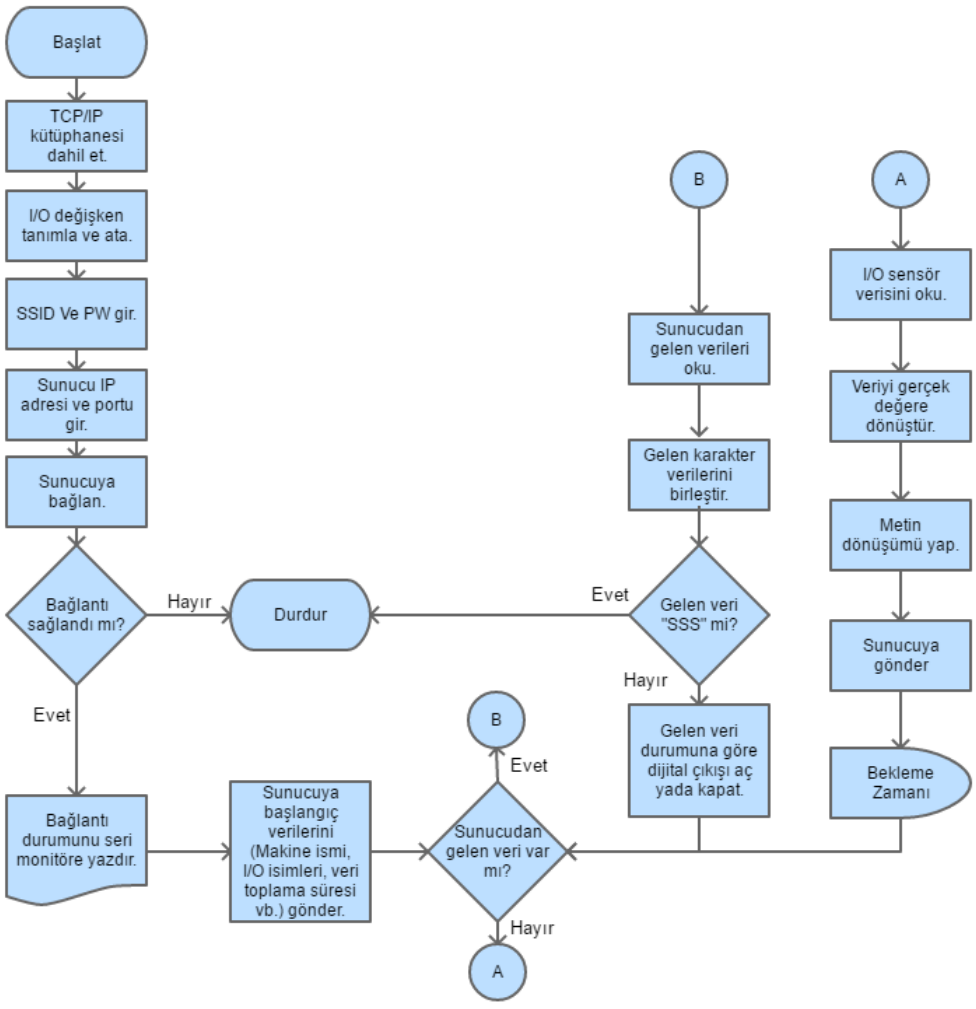
Bu çalışmada MSP430F5529 Launchpad gömülü sistem donanımını programlamak için Energia geliştirme ortamı kullanılmıştır. Yazılıma özel ekle-çıkar protokolü kullanılarak C/C++ tabanlı ortamda istemci yazılımı gerçekleştirilmiştir. Verileri internet üzerinden göndermek için CC3100 Boosterpack kullanılmıştır. Energia yazılımında SPI ve Wi-fi kütüphaneleri kullanılmıştır. Yazılım, Wi-fi kütüphanesindeki TCP/IP protokolü ile istemci sınıfına göre yazılmıştır.



Şekil 3.11. Energia geliştirme platformu

MSP430F5529 gömülü sistem kartı ile CC3100 Wifi Boosterpack kartı SPI haberleşme arayüzü üzerinden haberleşmektedir. Yazılımda genel olarak Wi-fi üzerinden internete bağlantı gerçekleştirilmiştir. Ardından sunucuya gidecek ilk veriler girilmiştir. LM35 sıcaklık sensöründen analog veriler sıcaklık verisine dönüştürülmüştür. Sıcaklık verisi, sunucuya 400 milisaniye aralıklarla yollanacak şekilde hazırlanmıştır. Sunucu ise dijital çıkış olarak gömülü sistem kartı üzerindeki yeşil lambayı kontrol edebilmektedir. Sunucu, kartın haberleşmesini sürekli kontrol etmektedir. İstenirse sunucu, istemciyi sistemden çıkarabilmektedir. MSP430, USB ile bilgisayara bağlandığında seri haberleşme monitöründen giden ve gelen veriler izlenebilmektedir. Ayrıca istemcinin sunucuya bağlantı kontrolü de yine bu monitörden izlenebilir.

TCP/IP haberleşme fonksiyonlarında veri yazılırken gönderilecek verinin uzunluğu hesaplanır. Hesaplanan boyut işaretsiz 8 bit sayıya dönüştürülür. Bu sayı metne dönüştürülüp gerçek metin verisi ile birleştirilerek sunucuya yollanmaktadır. Metni alan sunucu fonksiyonu ilk baytı okuyarak daha sonra okunacak bayt sayısını tespit etmektedir. İkinci okuma fonksiyonu geriye kalan metin verisinin tamamını okumaktadır. İstemciler için özel ekle-çıkar protokolüne uygun, yazılım dilinden bağımsız olarak Şekil 3.12.'de genel yazılım akış şeması gösterilmiştir.



Şekil 3.12. İstemcilerin genel yazılım akış şeması

3.2. Tezde Kullanılan Donanımlar

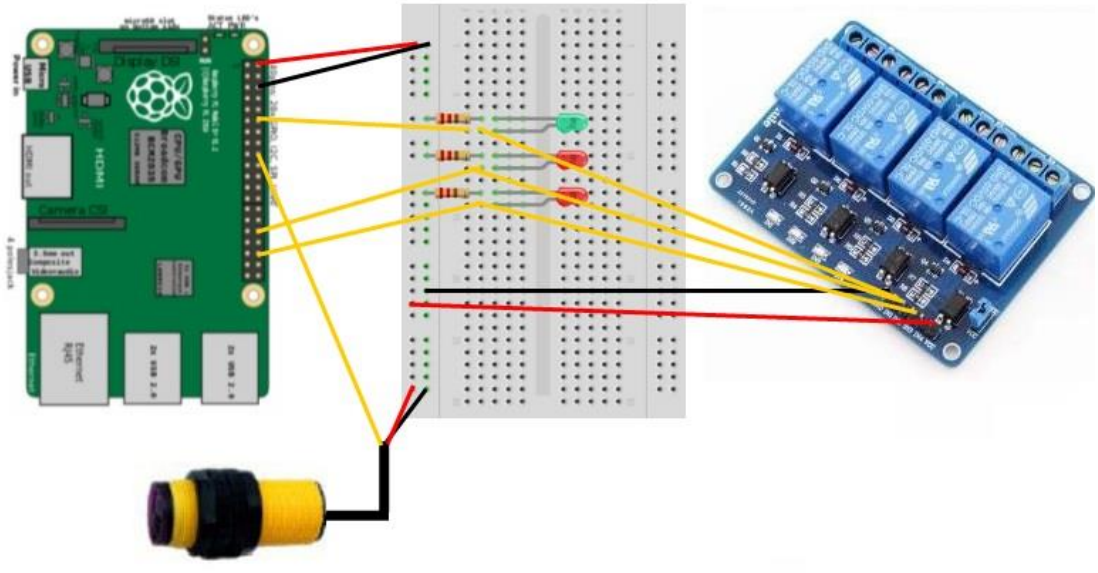
Bu deneysel çalışmada test amaçlı iki adet farklı istemci gömülü sistem donanım kartı kullanılmıştır. Bunlar Raspberry Pi 2 ve MSP430F5529 kartlarıdır.

3.2.1. Raspberry Pi

Raspberry Pi, kredi kartı büyüklüğünde tek kartlı bir mini bilgisayardır. Üzerinde 900 MHz dört çekirdekli ARM Cortex-A7 CPU bulundurmaktadır. İşlemcisi dışında 4 adet USB portu, 1GB RAM, 40 adet GPIO pini, HDMI portu, ethernet portu, ses çıkışı, kamera bağlantı arayüzü, görüntüleme arayüzü, mikro SD kart yuvası, 3D grafik video çekirdeği bulundurmaktadır [27].

Bu çalışmada Raspberry Pi 2 B model kart kullanılmıştır. HDMI portu sayesinde monitör ile bağlantısı gerçekleştirilmiştir. Konfigürasyon ayarları yapılmıştır. VNC sunucusu aktif edilmiştir. Tak çalıştır USB Wifi adaptörü sayesinde Wi-fi üzerinden Raspberry Pi yerel ağa bağlanmıştır. Windows tabanlı bilgisayar üzerinde VNC istemcisi kullanılarak Raspberry Pi ana ekran monitörüne de erişim sağlanmıştır.

Raspberry Pi kartının IP adresi LabVIEW for RPI derleyicisine girilerek bağlantı sağlanmaktadır. Bağlantı sağlandıktan sonra LabVIEW kodu Python koduna dönüştürülerek RPI kartına yürütülebilir pyzw uzantısıyla aktarılmıştır. Bu dosya çalıştırılarak istemci RPI kartı, sunucuya başarılı bir şekilde bağlanmıştır. RPI kartının bağlantı şeması Şekil 3.13.'te gösterilmiştir.



Şekil 3.13. Sistemde kullanılan Raspberry Pi genel bağlantı şeması

Raspberry Pi kartına üç adet dijital çıkış ve bir adet dijital giriş bağlanmıştır. Dijital çıkışlar 3 adet LED ve 3 adet röledir. Dijital giriş ise 1 adet 80 cm menzilli kızıl ötesi sensördür. LED ve röle çıkışları aynı pinlerden seçilmiştir. LED ve röleler RPI kartının 12, 16 ve 18. GPIO pinine bağlanmıştır. Sensör ise GPIO 24'e bağlanmıştır. Ayrıca RPI kartının yazılımında 0 ile 10 arasında analog değeri temsil etmesi amaçlı rastgele sanal bir değer bulundurmaktadır. İstemci kart, bu veriyi ve dijital giriş sensör verisini sunucuya 300 milisaniyede bir yollayacak şekilde hazırlamıştır.

Nesnelerin interneti sistemlerinde birçok gömülü sistem kartı kullanılabilir. Bunlardan bazıları Tablo 3.1.'de listelenmiş olup kıyaslaması yapılmıştır.

Tablo 3.1. Nesnelerin interneti için kullanılan donanımlardan bazılarının kıyaslanması [28]

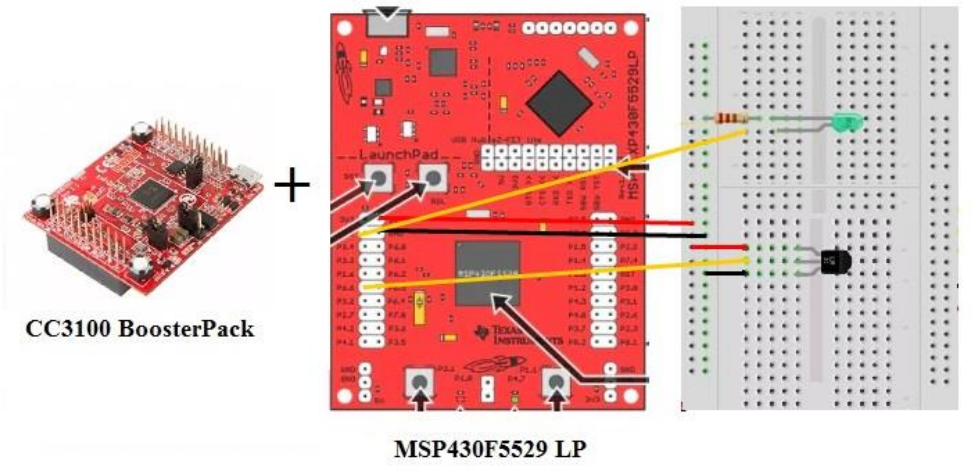
Parametreler	Arduino Uno	Beagle Bone Black	Raspberry B+
İşlemci	ATMega328P	Sitara AM3358BZCZ100	Broadcom BCM2835 SoC based ARM11 76JZF
GPU	-	PowerVR SGX@520 MHz	VideoCore IV 3D 250 MHz
Çalışma voltajı	5V	3,3V	5V
Çevrim Hızı (MHz)	16 MHz	1 GHz	700 MHz
Bus genişliği (bit)	8	32	32
Sistem hafızası	2kB	512MB	512MB
Desteklenen haberleşme	IEEE 802.11 b/g/n, IEEE 802.15.4, 433RF, BLE 4.0, Ethernet, Serial	IEEE 802.11 b/g/n, IEEE 802.15.4, 433RF, BLE 4.0, Ethernet, Serial	IEEE 802.11 b/g/n, IEEE 802.15.4, 433RF, BLE 4.0, Ethernet, Serial
Geliştirme çevresi	Arduino IDE	Debian, Android, Ubuntu, Cloud9 IDE	NOOBS
Programlama dili	Wiring	C, C++, Python, Perl, Ruby, Java, Node.js	Python, C, C++, Java, Seratch, Ruby
I/O Bağlantısı	SPI, I2C, UART, GPIO	SPI, UART, I2C, McASP, GPIO	SPI, DSI, UART, SDIO, CSI, GPIO, I2C

3.2.2. MSP430F5529

MSP430F5529 LP, MSP430F5529 USB mikro denetleyici için düşük maliyetli ve sade bir geliştirme kitidir. Üzerinde bulunan butonlar ve LED'ler sayesinde sade ve anlaşılabilir bir kullanıcı arayüze sahiptir. Birçok çeşit ek paketleri destekleyen başlık eklentisiyle seri üretim daha da basit hale getirilmiştir. Hızlı bir şekilde kablosuz bağlantı, grafiksel ekran, çevreyi algılama gibi birçok özelliği kazandırılabilir.

Texas Instruments'ta ve diğer üreticilerde bulunan hazır ek paketler alınabildiği gibi, özel ek paketler de tasarlanabilmektedir. MSP430F5529, 128KB flaş bellek, 8KB RAM, 25 MHz CPU hızı, 12 bit A/D dönüştürücü, 5 zamanlayıcı, bütünleşik USB özelliklerine ve birçok çevre birimlerine sahiptir. Kartın çalışma gerilimi 1,8V-3,6V arasındadır. Kart üzerinde 40 adet pin bulunmaktadır [29].

Bu çalışmada MSP430F5529 istemci gömülü sistem kartının yazılımı Energia IDE'si kullanılarak gerçekleştirilmiştir. Kablosuz olarak yerel ağa internet üzerinden bağlanmak için CC3100 ek paketi kullanılmıştır. SPI üzerinden veriler CC3100 ek paketine iletilmektedir. Uygulamada analog giriş olarak istemciye bir adet LM35 sıcaklık sensörü ve dijital çıkış olarak bir adet LED bağlanmıştır. Sistemin bağlantı şeması Şekil 3.14.'te gösterilmiştir.



Şekil 3.14. Sistemde kullanılan MSP430F5529 LP şematik bağlantısı

3.3. Sensörler ve Giriş-Çıkış Birimleri

3.3.1. MZ80 kızılötesi sensörü

MZ80 kızılötesi sensörü birçok projeye uygun dijital çıkışlı, yüksek kaliteli, endüstriyel kızılötesi sensörüdür. Ürün özellikleri aşağıdaki gibidir:

- Dijital çıkışlıdır.
- Arkasındaki trimpot ile menzili 3-80 cm arasında ayarlanabilir.
- 5V ile çalışır, tepki süresi 2 milisaniyedir.

Bu çalışmada sensör Raspberry Pi kartına bağlanarak kullanılmıştır. Sunucu arayüzünde "Cisim" gösterge ismiyle aktif ya da pasifliği izlenmiştir. Kırmızı renkli çıkışı +5 Volt, yeşil çıkışı toprak ve sarı çıkışı veri çıkışıdır [31].

3.3.2. LM35 sıcaklık sensörü

LM35 sıcaklık sensörü, santigrat sıcaklığına göre doğrusal orantılı çıkış gerilimi ile hassas entegre devre sıcaklık cihazıdır. -55 °C ila 150 °C sıcaklık aralığında çalışabilen düşük maliyetli sensördür. 4-30 V arasında çalışma voltajı vardır [32].

Bu çalışmada MSP430F5529 istemcisine LM35 sıcaklık sensörü bağlanarak sensörden gelen analog veri okunmuştur. MSP430F5529 istemcisi 3300 milivolt referans voltaja ve 12 bitlik ADC çözünürlüğe sahiptir. Adım başına düşen milivolt çevrim oranı aşağıdaki denklem 3.1'de hesaplanmıştır.

$$ADC\ çevrim\ oranı = \frac{Vref}{2^n} \tag{3.1}$$

Burada n ADC bit sayısını ifade etmektedir. İşlem sonucu ADC çevrim oranı 0,8056 bulunmuştur. LM35 sensöründe 1 santigrat 10 milivolta karşılık geldiğinden dolayı santigrat derece başına düşen ADC değişim değeri ise aşağıdaki denklem 3.2'de hesaplanmıştır.

$$1℃\ başına\ düşen\ ADC\ değişim\ değeri = \frac{10\ mV}{ADC\ çevrim\ oranı} \quad (3.2)$$

Çıkan sonuç 12,42 sayısı her dereceye karşılık gelen ADC verisidir. Sensörden okunan ADC verisi bu sayıya bölündüğünde çıkan sayı sensörün santigrat türünden sıcaklık değerini vermektedir.

3.3.3. Dört Kanallı röle ve ledler

Raspberry Pi istemci kartına dijital çıkış oluşturmak amacıyla kullanılmış devre elemanlarıdır. Röle ve LED'ler 5 Volt referans voltajı ile beslenmişlerdir. Röle 20 mA'lik bir akım çekmektedir. 30V DC veya 220V AC gerilimde 10 A'e kadar akımı anahtarlama yapabilmektedir. Her bir röle için kontrol ledleri bulunmaktadır. Röleler lojik 0 ile tetiklenmektedir [33]. Bağlantı şemaları Şekil 3.13.'te gösterilmiştir.

3.4. Haberleşme Protokolleri

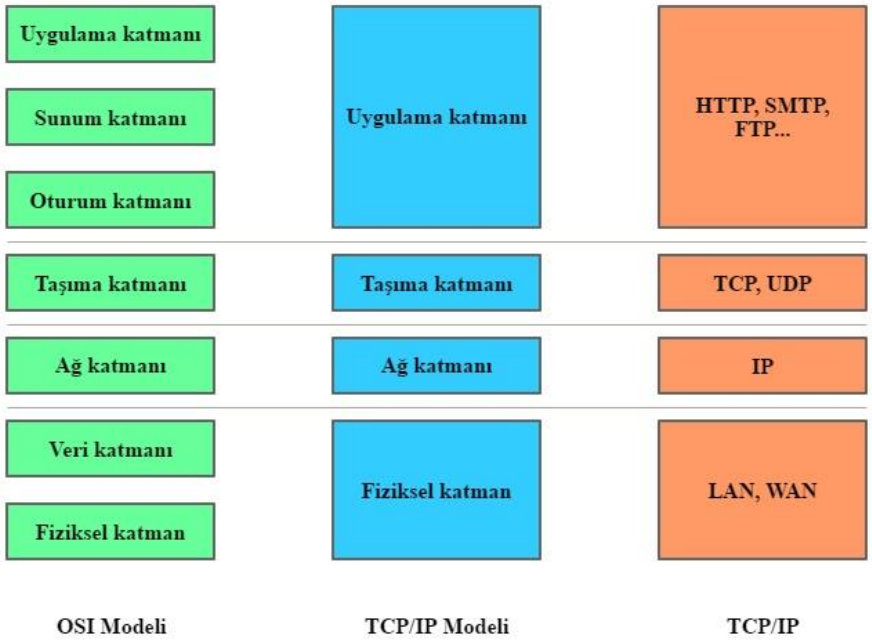
Bu çalışmada sunucu ve istemcinin haberleşmesi için alt yapıda TCP/IP protokolü kullanılmıştır. Sistemde meydana gelen hataların kullanıcıya mail yoluyla ulaştırılması için SMTP mail protokolü kullanılmıştır. Hazır eklemeli mimaride bir sunucu-istemci yapısı oluşturmak için istemci ekle-çıkar protokolü geliştirilmiştir.

3.4.1. TCP/IP haberleşme protokolü

Protokol, bir iletişim sürecinde, internet bağlantısını sağlayan noktalar arasındaki çift yönlü mesajlaşmayı oluşturan ve bu mesajlaşmaları kontrol eden kurallar dizisi olarak adlandırılır. Yazılım veya donanımlar arasında oluşan protokol, öğelerin kendi arasında oluşturmayı hedefledikleri haberleşmeyi kabul ettiği ve uyguladığı anlamına gelir. TCP/IP protokolü de bu şekilde çok fazla bilgi iletişim protokolün toplandığı bir protokoller ailesidir. İnternet en geniş ağlardan biridir. Geniş ağların yönetilmesi ve geliştirilmesi için TCP/IP protokolü gibi protokoller gereklidir.

TCP/IP protokolü, internet ağını bağımsız olarak kontrol edebilen ve yöneten bir protokoldür. TCP/IP protokolü TCP, IP gibi protokollerden oluşmaktadır. Bünyesinde 4 katman barındırır. Diğer bir iletişim modeli olan OSI modeli de TCP/IP modelinden farklı olarak yedi katmandan oluşur [34].

OSI, bilgisayar ağları oluşturulurken kullanılan donanımsal ve yazılımsal çözümleri düzenleyen standarttır. Open System Interconnection (OSI) modeli ISO (International Organization for Standardization) tarafından geliştirilmiştir. OSI modeli 7 katmandan oluşmaktadır. Her katmanın farklı görevleri vardır. Şekil 3.15.'te OSI ve TCP/IP katman yapıları kıyaslanmıştır.



Şekil 3.15. OSI ve TCP/IP katman yapılarının kıyaslanması

Bu çalışmada kullanılan TCP/IP protokolü ise 4 katmanlı modeldir. Uygulama katmanı, gönderilecek veri tipi ve veriyi işleyecek uygulamalar bulundurur. OSI modelindeki sunum ve oturum katmanları TCP/IP modelinde uygulama katmanında bulunmaktadır. E-posta gönderimi için kullanılan SMTP ve dosya gönderimi için kullanılan FTP protokolleri bu katmanda bulunmaktadır.

Taşıma katmanı, verinin nasıl gönderileceği belirlenmektedir. Veri güvenliği kontrolü bu katmanda gerçekleştirilir. TCP ve UDP bu katmandadır. Ağ katmanı, IP katmanı olarak da bilinen verilerin gönderildiği katmandır. Adresler veriye eklenir ve gönderme işlemi gerçekleştirilir. Fiziksel katmanda ise verinin hangi yolla yollanacağı belirlenir. İletişim ortamının özelliklerini, haberleşme hızını ve kodlama şemasını belirler. Ethernet, Wi-fi gibi protokoller bu katmanda bulunur [35].

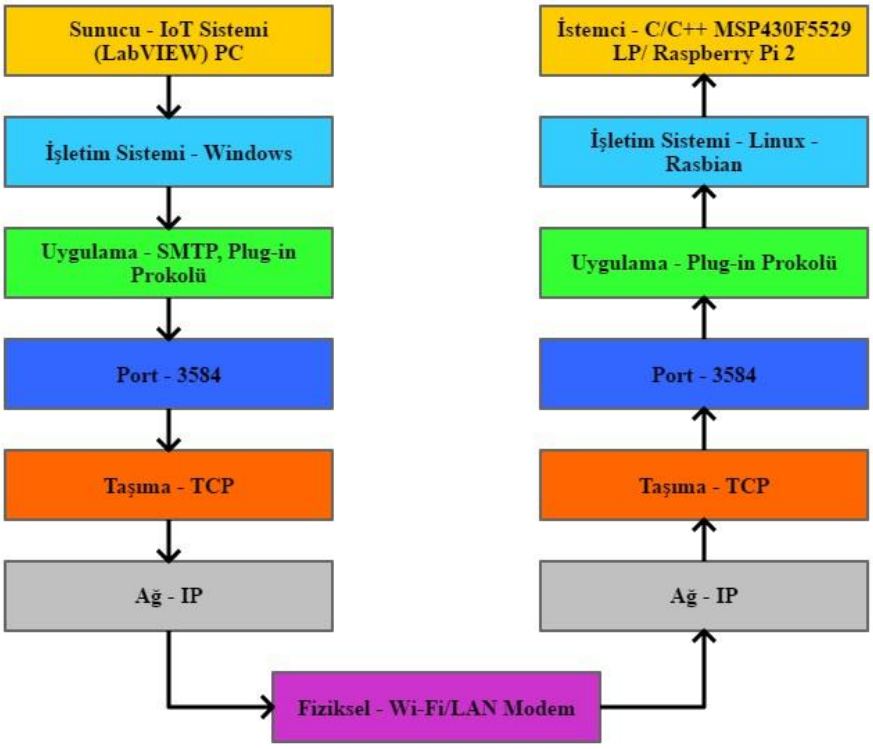
TCP protokolü, üst katmandan gelmiş verileri uygun uzunlukta böler. Parçalanmış verilere alıcı kısmında aynı şekilde alınabilmesi için sıra numarası verir. Kaybolan veya bozuk gelen parçaların tekrarlanmasını sağlar. TCP kendisine atanmış olan görevleri gerçekleştirebilmesi amacıyla ulaşım katmanında veri parçaların önüne başlık bilgisini ekler. Taşıma katmanın bir alt katmanı olan IP katmanı gelen veri parçalarına IP başlığı ekleyerek alıcıya gönderir [36].

IP protokolü, hedef bilgisayarın ağ üzerindeki yerini bulur. Paketlere adres vererek ağ üzerindeki bilgisayarlar arasında yönlendirmeyi sağlayan bağlantısız bir protokoldür. IP adresi, belli bir ağa bağlı cihazların, ağ üzerinde birbirlerine veri yollamak için kullandıkları adrestir. Bir cihaz internete bağlı ise o cihaza bir IP adresi atanır. Tüm internete bağlı cihazlar bu IP adresinden cihaza bağlanırlar. Böylece; farklı cihazlar aynı yerel ağda bulunmasalar bile, birbirleri ile iletişim kurarlar. IP adresleri IPv4 için 32 bit boyutundadırlar. 4 adet 8 bitlik sayıdan oluşurlar. “192.168.1.36” şeklinde gösterilir. Nokta ile ayrılmış her gruba “oktet” adı verilir. IP adresleri ikilik biçimde de gösterilebilir [34].

IP adresleri iki çeşittir. Statik ya da dinamik IP olarak oluşturulabilir. Statik IP adresi atanmış cihazın IP adresi zaman içinde değişmez. Dinamik IP atanmış bir cihazın internet bağlantısı kopup, ağa tekrar bağlandığında o cihaza yeni bir IP adresi atanabilir. Bu yüzden Statik IP adresleri genelde sunuculara verilmektedir. Böylece istemci cihazlar, sunucu bilgisayarı her zaman tanımlar ve bağlantı sağlayabilirler [30]. Bu çalışmada sunucu bilgisayara statik IP atanmıştır. İstemciler statik IP’ye ve sunucu portuna bağlanarak sunucu ile haberleşme sağlamışlardır. Sunucu ve istemciler arasındaki ağ, yerel ağdır. İstemcilerin geniş alan ağı (WAN) üzerinden

sunucuya bağlantı yapabilmesi için modemde port açma ve yönlendirme işlemlerinin yapılması gerekmektedir.

Tezde sunucu olarak PC üzerinde çalışan LabVIEW dilinde yazılmış IoT sunucu sistemi kullanılmıştır. Bilgisayardaki işletim sistemi olarak Windows kullanılmıştır. Uygulama katmanında mail haberleşmesi için SMTP protokolü, eklemeli mimari için özel Plug-in uygulama protokolü kullanılmıştır. Taşıma katmanı TCP protokolüdür. Ağ katmanı IP protokolüdür. Fiziksel katmanda ise Wi-Fi kullanılmıştır. TCP/IP katmanlarının bu çalışmada kullanılma yapısı Şekil 3.16.'da gösterilmiştir.

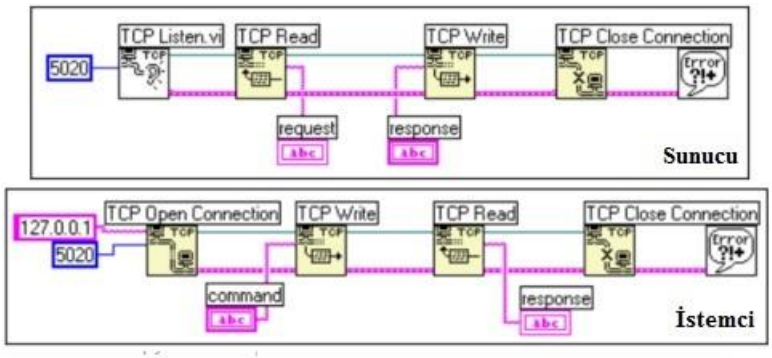


Şekil 3.16. TCP/IP protokolü katmanlarının çalışmada kullanılması

Sunucunun birçok istemci ile eşzamanlı olarak haberleşirken bir yandan yeni istemcilerle bağlantısını sağlamak için çok kanallı (Multi-thread) programlama kullanılmıştır. TCP/IP protokolü, sunucunun IP adresindeki belirtilen porttan birçok istemci ile bağlantı kurarken diğer yandan iletişime imkân vermektedir.

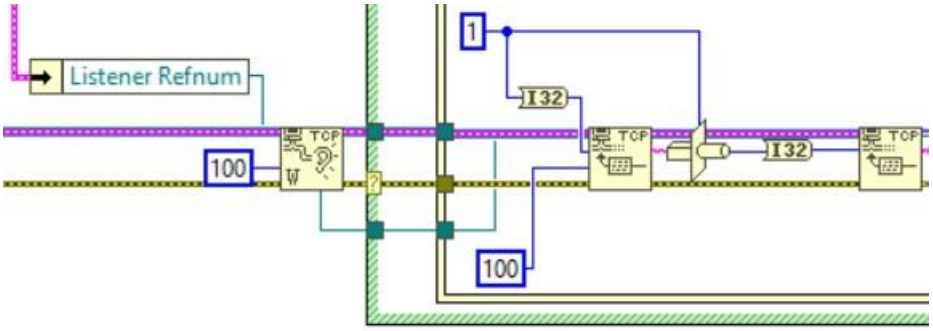
Çok kanallı programlama kullanarak özellikle Giriş-Çıkış (I/O) ve hesaplama gerektiren CPU işlemlerinin karışık kullanıldığı zamanlarda bilgisayar performansının ciddi şekilde arttırılması sağlanmıştır. Böylece birçok istemci TCP/IP protokolü aracılığı ile sunucuya bağlanmış ve işlemlerini gerçekleştirmişlerdir.

LabVIEW programında TCP Kütüphanesi bulunmaktadır. Diğer dillerdeki gibi bu kütüphanenin fonksiyonları kullanarak sunucu yazılımı gerçekleştirilmiştir [37]. Burada sunucu yeni istemci için açtığı portu 100 milisaniyelik aralıklarla dinlemiştir. İstemci ise TCP bağlanma fonksiyonu ile sunucunun IP adresini ve portunu kullanarak sunucuya bağlanmıştır. TCP okuma ve yazma fonksiyonları ile veri alışverişi gerçekleştirmekte ve TCP kapatma fonksiyonu bağlantıyı kapatmaktadır. Örnek olarak basit bir sunucu-istemci kodu Şekil 3.17.'degösterilmiştir.



Şekil 3.17. Örnek basit bir TCP/IP sunucu-istemci kodu [37]

Tez çalışmasında veri alış verişi yapılırken verinin boyutu hesaplanarak bir baytlık boyut verisi metin bilgisine dönüştürülerek gerçek veri ile birleştirilmiş ve gönderilmiştir. Sunucu gelen verinin boyutunu bilmesi için ilk olarak bir baytlık veriyi okumuştur. Okunan bir baytlık veri, sayıya dönüştürülerek ikinci okumada okunacak toplam veri sayısı hesaplanmıştır. İkinci okumada gerçek veri okunarak işlemler gerçekleştirilmiştir. Aynı şekilde sunucudan istemciye de veriler bu şekilde gönderilmiştir. Bağlantı döngüsündeki bağlantı kontrolü ve ilk haberleşme kodu Şekil 3.18.'de gösterilmiştir.



Şekil 3.18. IoT sistemi bağlantı kontrolü ve haberleşme kodu

TCP protokolünün UDP protokolüne göre tercih edilme sebebi daha güvenli olmasıdır. TCP'de gönderilen her veri paketinin ardından verinin yerine doğru bir şekilde ulaşıp ulaşmadığı kontrol edilmektedir. IoT sistemlerinde verilerin istemciden sunucuya veya sunucudan istemciye zamanında ve doğru bir şekilde iletilmesi önem arz etmektedir. IoT sistemi için standartlaşmış bir protokol yoktur. Amaca ve ihtiyaca yönelik protokoller seçilerek kullanılmaktadır. Çalışmada istenen amaca yönelik veri iletimi TCP/IP protokolü kullanılarak haberleşme başarıyla gerçekleştirilmiştir.

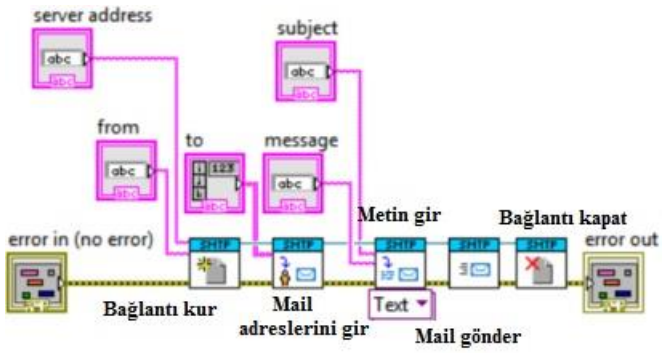
3.4.2. SMTP mail protokolü

Basit Posta Aktarım protokolü (Simple Mail Transport Protocol), POP3 hizmetiyle birlikte e-posta hizmetlerinin parçası olarak yüklenir. SMTP, e-postanın internet boyunca aktarılıp hedef sunucuya teslim edilme yönetimini denetler. POP3 hizmeti e-postayı posta sunucundan kullanıcın bilgisayarına alırken SMTP hizmeti sunucular arasında e-posta alır ve gönderir [36].

SMTP protokolü TCP/IP protokolünün uygulama katmanında bulunur. 25. Port üzerinden çalışmaktadır. Üç aşamadan oluşmaktadır. Birinci aşamada sunucular arasında iletişim başlatılır. İkinci aşamada mesaj transferleri gerçekleştirilir. Son olarak arada kurulmuş bağlantı kapatılır.

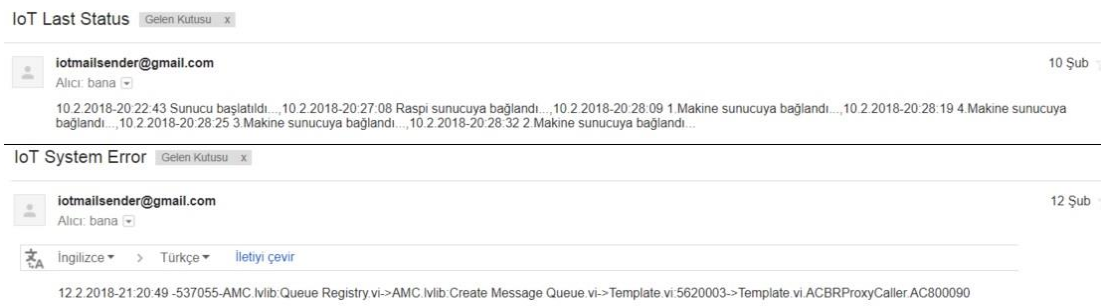
SMTP protokolünde Gmail SMTP yapılandırması kullanılmıştır. Bu yüzden SMTP sunucusu oluştururken Gmail sunucu bilgileri girilmiştir. Sunucu adı

smtp.gmail.com, bağlantı noktası 587, kullanıcı adı mail adresimiz, şifremiz ise mail adresimizin şifresi olarak sisteme tanımlanmıştır. SMTP protokolü, TLS güvenlik protokolü kullanılarak gerçekleştirilmiştir. TLS güvenlik protokolü gönderilecek bilgiyi şifreleyerek yollar. Bu protokol gönderilmiş şifreli bilginin kesinlikle ve sadece doğru adreste deşifre edilmesini sağlar. Gönderici ve alıcı doğrulama yaparak bilginin gizliliğini korumuş olur. Gönderilecek mail adresleri alıcı girme fonksiyonu ile tanımlanmıştır. Mesaj kurma fonksiyonu ile yazılacak mesaj yazılmıştır. Gönderme fonksiyonu, mesajı hedef mail adresine göndermektedir. SMTP sunucu kapama fonksiyonu yazılım kapatılırken kullanılarak sunucu bağlantısını kapatmıştır. Şekil 3.19.'da örnek SMTP yazılım kodu gösterilmiştir.



Şekil 3.19. SMTP yazılım kodu uygulaması

Bu çalışmada sunucu yazılımında herhangi bir hata söz konusu olduğunda istenen mail adreslerine hata kodu ile birlikte hata mesajı otomatik olarak gönderilmiştir. İstenirse sistem kayıtları da manuel olarak belirtilen mail adreslerine mesaj olarak gönderilebilmiştir. Şekil 3.20.'de IoT sisteminin gönderdiği e-postalar gösterilmiştir.



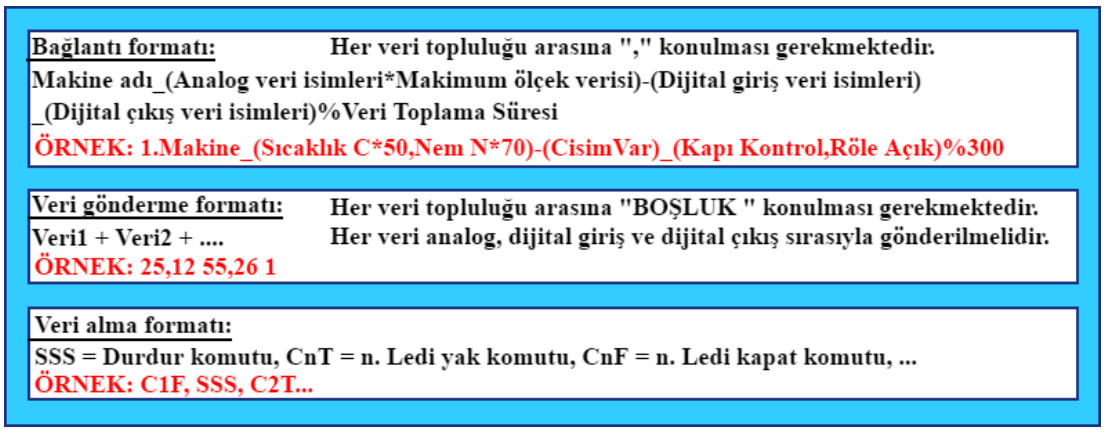
Şekil 3.20. IoT sisteminden gelen uyarı ve durum e-mail mesaj örneği

3.4.3. İstemci ekle-çıkar protokolü

Sunucunun yazılımını değiştirmeden ve çalışma anında iken yeni istemcileri sisteme eklemek için ekle-çıkar protokolü oluşturulmuştur. Bu protokol sayesinde Şekil 3.21.'de belirtilen formatlara uygun istemci yazılımı kullanılarak ve TCP/IP protokolü ile yazılarak donanımdan ve yazılımdan bağımsız istemcileri sunucuya çalışma anında eklemek mümkün olmuştur.

Bağlantı formatında istemci sunucuya bağlandıktan hemen sonra ilk olarak bu formatta donanım bilgilerini sunucuya göndermesi gereklidir. İstemcinin sunucuya bağlanabilmesi için sunucunun istemciyi kabul etme butonu açık olmalıdır. Bağlantı formatı; makine ismi, donanımın üzerindeki analog ve dijital giriş-çıkışların isimleri, ölçekleri, birimleri ve veri toplama süresi girilerek oluşturulmuştur. Veri gönderme formatında analog ve dijital verilerin isim sırasına göre aralarına boşluk koyarak veriler gönderilmelidir. Veri alma formatında ise "SSS" istemciyi durdurma komutudur. "CnT" ve "CnF" komutları sunucudan gelen ve dijital çıkışları kontrol etmek için n. çıkışı aç ve kapat komutlarıdır.

İstemci ekle-çıkar protokolüne uygun olarak yazılımı yazılmış gömülü sistem donanım istemcisi sunucuya bağlandığında sunucu ilk gelen bilgileri bölmektedir. Sunucu, makine ismini ve TCP referans numarasını hafızasında tutmaktadır. Bu veriler ile birlikte gelen diğer tüm verileri şablon VI'ya göndermektedir. Şablon VI istemci ile beraber çalışacak pencereyi açmaktadır. Açılan pencere yazılımı gelen verileri ayıklayarak hangi giriş-çıkışların olduğunu, isimlerini, ölçeklerini ve birimlerini gerekli yerlere atayarak istenilen nesneleri arayüzünde göstermektedir. Veri toplama süresinde veriler, veri formatına uygun olarak alınarak veri tabanına kaydedilmektedir. İstemci penceresi, dijital çıkış kontrol verilerini istemci veri alma formatına göre istemci donanımına göndermektedir. Böylece dinamik eklemeli ekle-çıkar protokolü oluşturulmuştur.



Şekil 3.21. İstemci için uygulama protokolü formatları ve örnekleri

Her şablon VI arayüzü test amaçlı dört analog giriş, dört adet dijital giriş ve dört adet dijital çıkış olacak şekilde tasarlanmıştır. Ancak bu tamamen kullanıcıya bağlıdır. Giriş-çıkış sayısı ihtiyaca yönelik olarak sunucu yazılımında otomatik arttırılabilir. Giriş ve çıkış sayısının artması her istemci için pencerenin büyümesine sebep olacaktır. Büyük boyutlarda birden fazla monitör kullanılarak fazla giriş-çıkışa ve istemciye sahip olan yüksek sistemlerin izlenmesi yapılabilir.

BÖLÜM 4. SİSTEMİN GERÇEKLEŞTİRİLMESİ

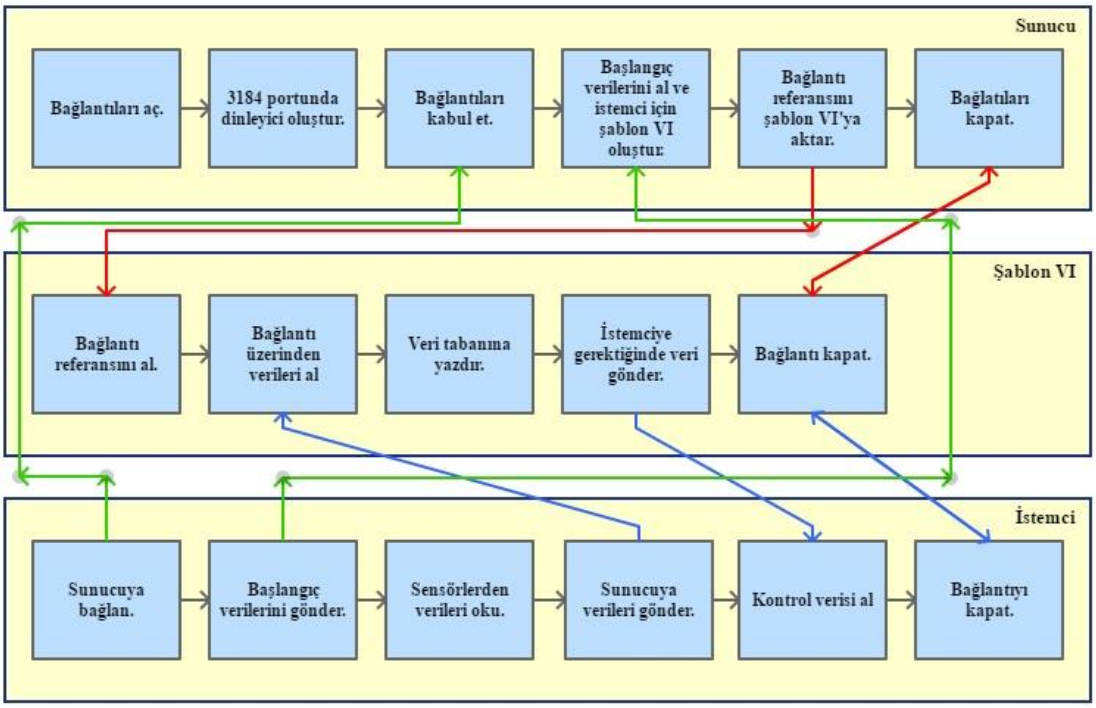
4.1. Sistemin Çalışması

Sunucu programı başlatıldığında sunucu arayüzünde bir takım ayarlamalar ve sıfırlama işlemleri yapılmaktadır. Sunucu 3184. portta dinleyici oluşturmaktadır. Her 100 milisaniyede bir bağlantı kontrolü gerçekleştirilmiştir. İstemci kabul etme butonu aktif ise istemci ekle-çıkar protokolü ile yazılımı yazılmış herhangi bir istemci, sunucu ile bağlantı kurabilmektedir. Sunucuya bağlanan istemci, başlangıç bilgilerini göndermektedir. Başlangıç bilgilerini alan sunucu bu bilgileri ve TCP referansını şablon VI'ya göndermektedir. Şablon VI, referansı ve bilgileri aldıktan sonra bilgileri ayıklamakta ve kendi arayüzünü şekillendirmektedir.

Şablon VI, TCP referansını kullanarak istemciden veri toplama süresi boyunca veri talep etmektedir. Şablon VI tarafından makine ismi kullanılarak veri tabanında tablo oluşturulur. İstemci, sensörlerden okuduğu verileri şablon VI'ya gönderir. Veriler toplandıkça veri tabanına yazma işlemi de gerçekleştirilmektedir. Dijital çıkış kontrolü şablon VI üzerinden gerçekleştirilir. Kontrol verisini alan istemci donanım kartı, sinyal durumunu değiştirerek dijital çıkışını kontrol etmektedir.

Sunucu üzerinden istemci kontrol edildiği gibi, şablon VI üzerinden de istemci kontrol edilebilmektedir. İstenirse sunucu üzerinden istemci ile sunucu bağlantısı koparılabilir. İstemcinin sunucu ile bağlantısı koptuğunda bunu sunucu fark eder ve kullanıcıyı mail yoluyla bilgilendirir. İstemci penceresi, sunucu üzerinden gizlenebilir ya da istenildiğinde gösterilebilir. Sunucu seçtiği istemcinin tüm verilerini kendi tablosuna da aktarabilmektedir.

Şekil 4.1.'de IoT sisteminin genel yapısı gösterilmiştir. Sunucu manuel veya otomatik olarak hata durumlarını mail yoluyla kullanıcıya aktarmaktadır. Kullanıcı tabanlı güvenlik için şifreleme işlemi kullanılarak her kullanıcının, sunucu arayüzü üzerindeki her işlemi gerçekleştirmesinin önüne geçilmiştir.

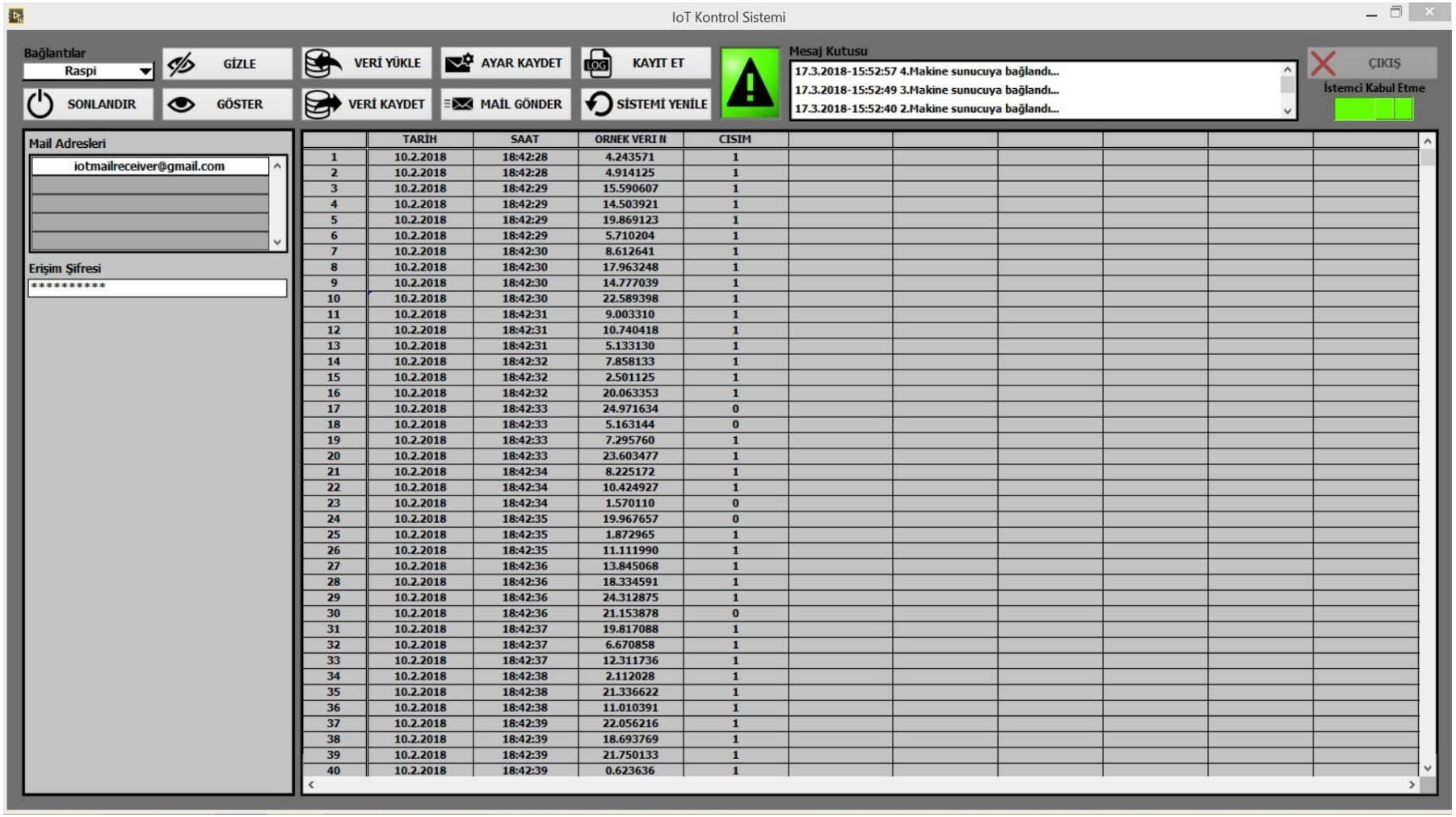


Şekil 4.1. IoT sistemi genel yapısı

4.2. Sunucu Arayüzü

Sunucu arayüzü iki kısımdan oluşmaktadır. Birinci kısım ana kontrollerin yapıldığı ve tüm istemcilerin kontrol edildiği arayüzdür. İkinci kısım her istemciye özel açılan arayüzdür. Bu arayüzler kendisine bağlı olan istemciyle bire bir haberleşerek verileri kullanıcıya göstermektedirler. Veri kayıt işlemleri de bu arayüz aracılığı ile gerçekleştirilerek uyarılar da bu arayüzde gösterilmektedir. Diğer bütün işlemler ana sunucu arayüzünde gerçekleştirilmiştir. Ana sunucu Şekil 4.2.'de gösterilmiştir.

Ana sunucuya bağlanan tüm istemcilerin isimleri "Bağlantılar" nümerik seçim çubuğuna gelmektedir. Buradan seçilecek istemci üzerine bir takım işlemler yapılmaktadır. "GİZLE" butonu seçilmiş istemcinin penceresini gizlemektedir. "GÖSTER" butonu seçilmiş istemcinin penceresini göstermektedir. "SONLANDIR" butonu seçilmiş istemcinin penceresini sonlandırarak bağlantısını kapatır. "VERİ YÜKLE" butonu seçilmiş istemcinin verilerini veri tabanından alarak tabloya yükler. "VERİ KAYDET" butonu tablodaki verileri Excel formatında başka konuma kaydedilmesini sağlar. "Mail Adresleri" bölümüne girilen e-mail adresleri "AYAR KAYDET" butonu ile kaydedilir. Kaydedilmiş e-maillere sistem uyarıları gönderilmektedir. "MAİL GÖNDER" butonu ise "Mesaj kutusundaki bildirimleri manuel olarak e-mail adreslerine gönderir. "Mesaj Kutusu" sunucuda meydana gelen mesajları gösterir. "KAYIT ET" butonu "Mesaj Kutusu" bildirimlerini TXT dosyası olarak kaydeder. "Uyarı LED" kullanıcıyı hatalara karşı uyarır. Yeşil yandığında sistemin sorunsuz çalıştığını ifade eder. Kırmızı yanması durumunda sistem hataya düşmüş demektir. "SİSTEMİ YENİLE" butonu ise sistemdeki hataları yeniler ve sunucunun hatasız bir şekilde devam etmesini sağlar. "Erişim Şifresi" alanı bazı kullanıcıların sunucudaki bazı butonlara erişimini kısıtlar. "Erişim Şifresi" girilmediği takdirde istemci üzerine yapılan işlemlere izin verilmez. "İstemci Kabul Etme" butonu sunucuyu istemci davetine açar veya kapatır. "ÇIKIŞ" butonu sunucuyu durdurur. "ÇIKIŞ" butonu sadece sunucuya bağlı herhangi bir istemci kalmadıysa aktif olmaktadır. Aynı şekilde panel kapama butonu da "ÇIKIŞ" butonu ile benzer düzende çalışmaktadır.



	TARİH	SAAT	ORNEK VERI N	CISIM
1	10.2.2018	18:42:28	4.243571	1
2	10.2.2018	18:42:28	4.914125	1
3	10.2.2018	18:42:29	15.590607	1
4	10.2.2018	18:42:29	14.503921	1
5	10.2.2018	18:42:29	19.869123	1
6	10.2.2018	18:42:29	5.710204	1
7	10.2.2018	18:42:30	8.612641	1
8	10.2.2018	18:42:30	17.963248	1
9	10.2.2018	18:42:30	14.777039	1
10	10.2.2018	18:42:30	22.589398	1
11	10.2.2018	18:42:31	9.003310	1
12	10.2.2018	18:42:31	10.740418	1
13	10.2.2018	18:42:31	5.133130	1
14	10.2.2018	18:42:32	7.858133	1
15	10.2.2018	18:42:32	2.501125	1
16	10.2.2018	18:42:32	20.063353	1
17	10.2.2018	18:42:33	24.971634	0
18	10.2.2018	18:42:33	5.163144	0
19	10.2.2018	18:42:33	7.295760	1
20	10.2.2018	18:42:33	23.603477	1
21	10.2.2018	18:42:34	8.225172	1
22	10.2.2018	18:42:34	10.424927	1
23	10.2.2018	18:42:34	1.570110	0
24	10.2.2018	18:42:35	19.967657	0
25	10.2.2018	18:42:35	1.872965	1
26	10.2.2018	18:42:35	11.111990	1
27	10.2.2018	18:42:36	13.845068	1
28	10.2.2018	18:42:36	18.334591	1
29	10.2.2018	18:42:36	24.312875	1
30	10.2.2018	18:42:36	21.153878	0
31	10.2.2018	18:42:37	19.817088	1
32	10.2.2018	18:42:37	6.670858	1
33	10.2.2018	18:42:37	12.311736	1
34	10.2.2018	18:42:38	2.112028	1
35	10.2.2018	18:42:38	21.336622	1
36	10.2.2018	18:42:38	11.010391	1
37	10.2.2018	18:42:39	22.056216	1
38	10.2.2018	18:42:39	18.693769	1
39	10.2.2018	18:42:39	21.750133	1
40	10.2.2018	18:42:39	0.623636	1

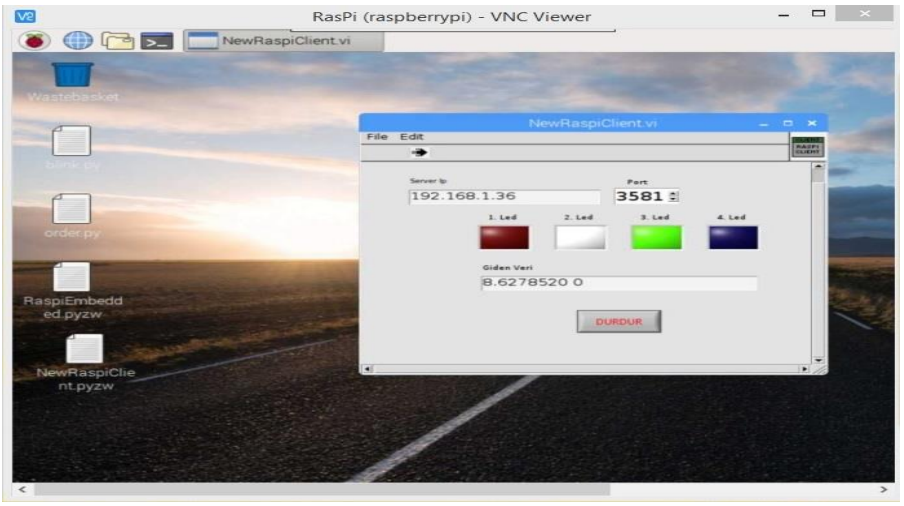
Şekil 4.2. Ana sunucu arayüzü

Şekil 4.3. İstemci pencereleri

İstemci pencereleri, istemciler sunucuya bağlandığında otomatik olarak açılmaktadır. Bu pencereler yeni istemciler eklendikçe çoğalmaktadır. Bu pencereler gizlenebilir ya da ayrı monitörde konumlandırılarak veriler anlık olarak izlenebilmektedir. Her istemci üzerinde bulundurduğu giriş-çıkış sayısı kadar göstergeyi penceresinde isimleri, birimleri ve ölçekleriyle göstermektedir. Analog ve dijital veriler alınırken dijital çıkışlar kontrol edilebilmektedir. Başlıkta istemci ismi yazarken sol üst köşede veri toplama süresi yazmaktadır. Sağ üst köşede ise istemcinin uyarı durum ledi bulunmaktadır. Bağlantı koptuğunda hata ledi uyarıya geçmektedir. Ana sunucu üzerindeki "SONLANDIR" butonu ile aynı işlevi gören "X" butonu pencereyi kapatırken istemciyi sunucudan koparır ve istemciyi durdurur. İstemci pencereleri Şekil 4.3.'de gösterilmiştir.

4.3. İstemci Arayüzü

Sanal istemciler ve Raspberry Pi gibi işletim sistemine sahip istemciler arayüze sahiptir. Bu arayüzler aracılığı ile kullanıcılar makine başında istemci kartının verilerini ve durumlarını izleyebilmektedir. Bu arayüz üzerinden sunucu IP adresi ve portu, veri toplama süresi girilebilmektedir. Böylece mevcut istemci yazılımını da basit işlemler için değiştirmeye gerek kalmamaktadır. Ayrıca giden veriler istemci arayüzünden izlenebilmektedir. Şekil 4.4.'te Raspberry Pi istemci arayüzü gösterilmiştir.



Şekil 4.4. Raspberry Pi istemci arayüzü

BÖLÜM 5. SONUÇLAR

Bu çalışmada, farklı sektörlere uyarlanabilen, yazılımdan ve donanımdan bağımsız bir şekilde bütün istemcileri sunucunun yazılımını değiştirmeden çalışma anında sunucuya kolayca eklenmesini sağlayabilen, yeni ve özgün bir nesnelerin interneti (IoT) sistemi tasarlanmış ve oluşturulmuştur. Bu bağlamda IoT sistemi esnek, çoğaltılabilir, genişletilebilir ve özelleştirilebilir nitelikte mimari üzerine kurulmuştur.

Sistemde asenkron haberleşme, çok kanallı programlama kullanılarak ve özel ekle-çıkar protokolü oluşturularak dördü sanal, ikisi gerçek istemcilerin sunucuya entegrasyonu başarılı bir şekilde gerçekleştirilmiştir. Her bir istemci birbirinden farklı giriş ve çıkışlar içerirken farklı zamanlarda veri gönderimi gerçekleştirmiştir. Sunucu tüm istemcilere aynı anda karşılık vererek sıfır tolerans ile paralel bir şekilde kontrol ve veri toplama işlemleri gerçekleştirilmiştir.

5.1. Deneysel Çalışma

Sistemde test amaçlı 6 adet istemci kullanılmıştır. 2 tanesi gerçek donanım üzerinden 4 tanesi sanal donanım üzerinden veri toplama işlemleri gerçekleştirilmiştir. Sunucu verileri asenkron ve çok kanalı bir yazılım ile topladığı için istemci sayısından bağımsız olarak veri toplama tam zamanlı gerçekleştirilmiştir. 1.Makine istemcisi 300 milisaniye, 2.Makine istemcisi 200 milisaniye, 3.Makine istemcisi 500 milisaniye, 4.Makine 100 milisaniye, MSP430 istemcisi 400 milisaniye, RPI istemcisi 300 milisaniye aralıklarla verileri başarılı bir şekilde toplamıştır. Sunucu verileri en az 30 milisaniye de bir başarılı bir şekilde toplamıştır. Sunucu ve istemci eş zamanlı bir şekilde çalışmıştır. Ancak kullanıcının verileri takip edebilmesi için en ideal veri toplama zamanın en az 100 milisaniye olduğu saptanmıştır.

Veri kontrol ve veri toplama işlemleri birbirlerini etkilememektedir. Veri toplama devam ederken istemciler kontrol edilebilmektedir. İstemcide hata meydana gelmesi durumunda verilerin bekleme zamanı içerisinde gelmemesi ile birlikte sunucu hataya geçerek kullanıcıyı uyarmakta ve istemciyi sistemden otomatik olarak düşürmektedir. 6 istemci aynı anda sisteme bağlanmış ve veriler toplanıp veri tabanına kaydedilmiştir. Şekil 5.1.'de gerçek donanımların verileri, Şekil 5.2.'de 1. ve 2. Makine sanal istemcilerin verileri, Şekil 5.3.'te 3. ve 4. Makine sanal istemcilerinin verileri gösterilmiştir.

iot.msp: 5.305 satır mevcut (yaklaşık)

Tarih	Zaman	Temperature_C
17.3.2018	20:57:19	24
17.3.2018	20:57:19	27
17.3.2018	20:57:20	24
17.3.2018	20:57:20	19
17.3.2018	20:57:20	25
17.3.2018	20:57:21	25
17.3.2018	20:57:21	25
17.3.2018	20:57:22	25
17.3.2018	20:57:22	23
17.3.2018	20:57:22	24
17.3.2018	20:57:23	26
17.3.2018	20:57:23	24
17.3.2018	20:57:24	24
17.3.2018	20:57:24	24
17.3.2018	20:57:24	23
17.3.2018	20:57:25	24
17.3.2018	20:57:25	24
17.3.2018	20:57:26	25
17.3.2018	20:57:26	24
17.3.2018	20:57:26	25
17.3.2018	20:57:27	24
17.3.2018	20:57:27	24
17.3.2018	20:57:28	24
17.3.2018	20:57:28	25
17.3.2018	20:57:29	25
17.3.2018	20:57:29	25
17.3.2018	20:57:29	25
17.3.2018	20:57:30	25
17.3.2018	20:57:30	24
17.3.2018	20:57:31	25
17.3.2018	20:57:31	24
17.3.2018	20:57:31	24

iot.raspi: 6.754 satır mevcut (yaklaşık)

Tarih	Zaman	Ornek_Veri_N	Cisim
17.3.2018	20:57:24	9.9008980	0
17.3.2018	20:57:25	4.2285540	0
17.3.2018	20:57:25	3.6154110	0
17.3.2018	20:57:25	1.3764140	0
17.3.2018	20:57:25	6.1769020	0
17.3.2018	20:57:26	1.5733500	0
17.3.2018	20:57:26	5.1308510	0
17.3.2018	20:57:26	4.3510950	0
17.3.2018	20:57:27	7.5536160	0
17.3.2018	20:57:27	7.4521780	0
17.3.2018	20:57:27	3.4990650	0
17.3.2018	20:57:28	0.7831700	0
17.3.2018	20:57:28	2.6438100	1
17.3.2018	20:57:28	4.6318020	1
17.3.2018	20:57:28	0.2326060	1
17.3.2018	20:57:29	0.6789650	0
17.3.2018	20:57:29	4.6230160	0
17.3.2018	20:57:29	6.4707740	0
17.3.2018	20:57:30	2.0476340	0
17.3.2018	20:57:30	2.3956740	0
17.3.2018	20:57:30	5.5227710	0
17.3.2018	20:57:31	0.8943610	0
17.3.2018	20:57:31	2.6288200	0
17.3.2018	20:57:31	1.2073010	0
17.3.2018	20:57:31	7.6232030	0
17.3.2018	20:57:32	6.8199600	0
17.3.2018	20:57:32	0.8855150	0
17.3.2018	20:57:32	9.1817040	0
17.3.2018	20:57:33	3.8218920	0
17.3.2018	20:57:33	9.6567970	0
17.3.2018	20:57:33	7.0589510	0

Şekil 5.1. MSP430 ve Raspberry Pi gerçek istemcilerinin verileri

iot.1_makine: 5.756 satır mevcut (yaklaşık)

Tarih	Zaman	Örnek_Veri_N
17.3.2018	20:57:24	21,933281
17.3.2018	20:57:24	3,094601
17.3.2018	20:57:25	12,630562
17.3.2018	20:57:25	24,384158
17.3.2018	20:57:25	9,718126
17.3.2018	20:57:25	21,336921
17.3.2018	20:57:26	20,117385
17.3.2018	20:57:26	22,116625
17.3.2018	20:57:26	18,450127
17.3.2018	20:57:27	6,282306
17.3.2018	20:57:27	24,344647
17.3.2018	20:57:27	13,362910
17.3.2018	20:57:28	24,929331
17.3.2018	20:57:28	1,360655
17.3.2018	20:57:28	5,070206
17.3.2018	20:57:28	19,243125
17.3.2018	20:57:29	6,916969
17.3.2018	20:57:29	8,587655
17.3.2018	20:57:29	5,841534
17.3.2018	20:57:30	19,533962
17.3.2018	20:57:30	8,473907
17.3.2018	20:57:30	19,542929
17.3.2018	20:57:31	20,840104
17.3.2018	20:57:31	17,904787
17.3.2018	20:57:31	0,465168
17.3.2018	20:57:31	21,529234
17.3.2018	20:57:32	12,766005
17.3.2018	20:57:32	21,649160
17.3.2018	20:57:32	0,661710
17.3.2018	20:57:33	16,179666
17.3.2018	20:57:33	17,953849
17.3.2018	20:57:33	1,645620

iot.2_makine: 2.600 satır mevcut (yaklaşık)

Tarih	Zaman	Analog_Veri_C	Test_Sm
17.3.2018	20:57:22	11,724521	28,515377
17.3.2018	20:57:22	5,982570	31,695718
17.3.2018	20:57:23	3,662663	17,731211
17.3.2018	20:57:23	2,900985	13,471429
17.3.2018	20:57:23	6,937169	33,572698
17.3.2018	20:57:23	5,079316	16,164783
17.3.2018	20:57:23	12,996603	15,503789
17.3.2018	20:57:24	3,712825	29,271903
17.3.2018	20:57:24	13,309845	13,188843
17.3.2018	20:57:24	12,614443	21,749053
17.3.2018	20:57:24	13,957459	30,793714
17.3.2018	20:57:24	11,786412	9,604171
17.3.2018	20:57:25	12,694360	19,897882
17.3.2018	20:57:25	0,528828	17,379079
17.3.2018	20:57:25	4,584676	8,036716
17.3.2018	20:57:25	6,591042	6,969436
17.3.2018	20:57:25	0,207168	26,991259
17.3.2018	20:57:26	3,726871	12,169321
17.3.2018	20:57:26	8,315300	23,653603
17.3.2018	20:57:26	12,254606	6,711841
17.3.2018	20:57:26	6,267785	23,051739
17.3.2018	20:57:26	7,329811	29,523424
17.3.2018	20:57:27	2,124783	10,847610
17.3.2018	20:57:27	4,013034	4,650951
17.3.2018	20:57:27	2,016026	4,938176
17.3.2018	20:57:27	12,073639	15,118270
17.3.2018	20:57:27	11,844198	25,885127
17.3.2018	20:57:28	12,522482	8,071019
17.3.2018	20:57:28	5,731253	17,228107
17.3.2018	20:57:28	8,269218	24,833499
17.3.2018	20:57:28	13,040523	15,500005
17.3.2018	20:57:28	5,994293	26,873485

Şekil 5.2. 1. ve 2. Makine sanal istemcilerinin verileri

Her istemci giriş ve çıkış birimlerinden toplamış olduğu dijital ve analog verilerini sunucuya göndermektedir. Sunucuya gelen veriler, isimlerine özel açılan sütuna tarih ve zaman ile birlikte belirlenen süre aralıklarında kaydedilmektedir. Tablolarda görüldüğü gibi MSP430 istemcisinden gelen sıcaklık sensör verisi anlık olarak veri tabanına kaydedilmiştir. Aynı şekilde RPI istemcisinden gelen sanal analog ve dijital sensör verileri de sunucu tarafından veri tabanına kaydedilmiştir.

Şekil 5.4.'te MSP430 ve Raspberry Pi gerçek donanım istemcileri ile birlikte cihazlara bağlı sensör, röle, LED gibi giriş-çıkış birimleri gösterilmiştir.

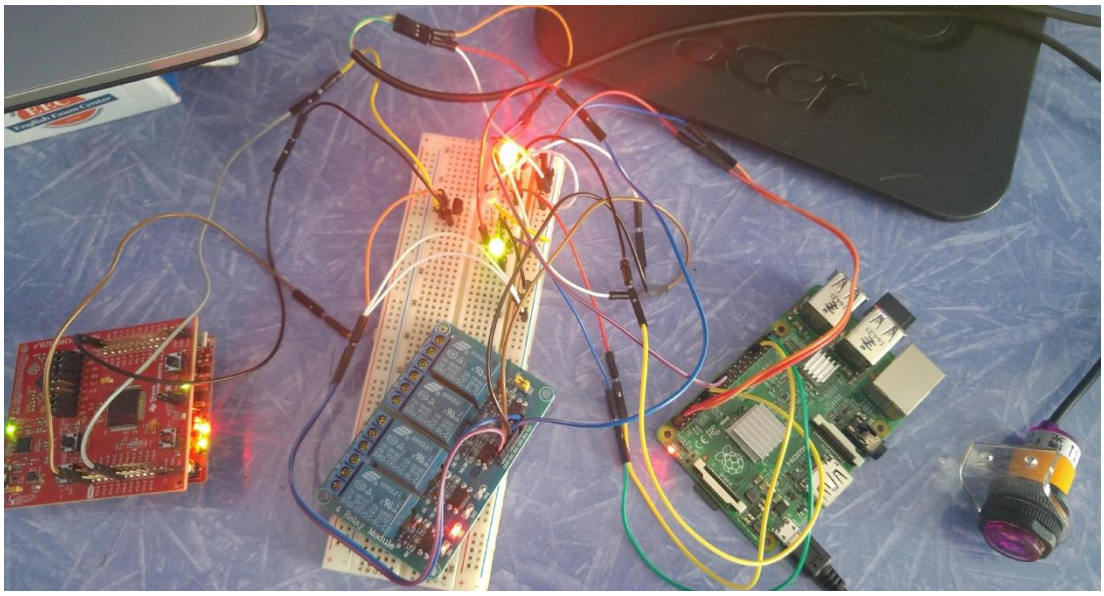
iot.3_makine: 2.026 satır mevcut (yaklaşık)

Tarih	Zaman	İlk_Sensor_C	İkinci_Sensor_D	Üçüncü_Sensor_M
17.3.2018	20:57:17	0,880938	12,498557	12,043932
17.3.2018	20:57:18	7,756876	6,430486	8,437971
17.3.2018	20:57:18	7,952939	6,849284	6,414512
17.3.2018	20:57:19	5,651058	13,155643	18,163881
17.3.2018	20:57:19	0,503120	11,691735	17,255274
17.3.2018	20:57:20	0,819390	13,201755	14,911431
17.3.2018	20:57:20	1,304699	13,917062	11,581214
17.3.2018	20:57:21	2,019358	7,597898	3,005236
17.3.2018	20:57:21	7,952365	4,370325	6,074153
17.3.2018	20:57:22	3,775587	12,795266	1,524677
17.3.2018	20:57:22	0,092564	2,085529	3,753659
17.3.2018	20:57:23	7,371765	6,935621	4,437232
17.3.2018	20:57:23	9,288681	8,715531	5,942606
17.3.2018	20:57:24	0,330934	8,016715	10,821071
17.3.2018	20:57:24	8,669157	10,237540	9,586048
17.3.2018	20:57:25	1,249377	8,882620	0,582236
17.3.2018	20:57:25	0,593138	12,617608	0,266188
17.3.2018	20:57:26	8,476877	7,030992	2,867892
17.3.2018	20:57:26	2,368766	2,094663	4,651874
17.3.2018	20:57:27	1,568896	7,755352	17,992054
17.3.2018	20:57:27	8,462503	1,149978	12,880576
17.3.2018	20:57:28	1,181346	10,969352	6,654214
17.3.2018	20:57:28	2,001885	12,644754	12,014131
17.3.2018	20:57:29	9,645732	3,458139	15,493339
17.3.2018	20:57:29	5,274013	0,281685	1,609021
17.3.2018	20:57:30	8,099033	8,569289	16,722473
17.3.2018	20:57:30	1,162893	14,522619	9,035722
17.3.2018	20:57:31	0,200301	2,482787	15,419267
17.3.2018	20:57:31	9,364594	1,629562	11,597194
17.3.2018	20:57:32	6,301254	13,691781	9,995342
17.3.2018	20:57:32	4,137889	8,012280	12,228778

iot.4_makine: 8.148 satır mevcut (yaklaşık)

Tarih	Zaman	Sıcaklık_C	Nem	Kapı_Açık	Cisim_Var
17.3.2018	20:57:13	5,455680	69,720923	0	1
17.3.2018	20:57:14	47,109547	44,818496	0	1
17.3.2018	20:57:14	21,612502	35,749214	0	1
17.3.2018	20:57:14	11,618296	36,477500	0	1
17.3.2018	20:57:14	35,734839	14,581503	0	1
17.3.2018	20:57:14	24,799619	0,848722	0	1
17.3.2018	20:57:14	36,325965	47,994681	0	1
17.3.2018	20:57:14	38,363782	30,014744	0	1
17.3.2018	20:57:14	31,620489	63,937039	0	1
17.3.2018	20:57:14	1,207999	60,222690	0	1
17.3.2018	20:57:14	1,284416	51,160382	0	1
17.3.2018	20:57:15	1,898434	31,559031	0	1
17.3.2018	20:57:15	43,409734	48,580651	0	1
17.3.2018	20:57:15	14,165776	26,423695	0	1
17.3.2018	20:57:15	23,694854	21,300613	0	1
17.3.2018	20:57:15	12,118930	64,404528	0	1
17.3.2018	20:57:15	3,279827	40,521312	0	1
17.3.2018	20:57:15	5,748807	38,482204	0	1
17.3.2018	20:57:15	22,225745	7,987685	0	1
17.3.2018	20:57:15	23,220713	23,510280	0	1
17.3.2018	20:57:15	41,287648	12,012798	0	1
17.3.2018	20:57:16	28,863243	6,694947	0	1
17.3.2018	20:57:16	16,378547	9,209794	0	1
17.3.2018	20:57:16	0,169661	29,219349	0	1
17.3.2018	20:57:16	47,796972	2,773307	0	1
17.3.2018	20:57:16	24,960257	10,465970	0	1
17.3.2018	20:57:16	20,306473	24,987750	0	1
17.3.2018	20:57:16	24,217945	26,654405	0	1
17.3.2018	20:57:16	32,659589	45,017618	0	1
17.3.2018	20:57:16	33,926228	63,585848	0	1
17.3.2018	20:57:16	31,322665	35,024944	0	1

Şekil 5.3. 3. ve 4. Makine sanal istemcilerinin verileri



Şekil 5.4. IoT sistemi MSP430 ve Raspberry Pi istemcileri

5.2. Sonuç ve Öneriler

Gerçekleştirilen sistemde sunucuya eklenecek istemci sayısında teorik olarak bir sınır bulunmamaktadır. Fakat kullanılan her bir istemci bir şablon VI çağırdığından sunucunun üzerinde çalıştığı bilgisayara bir işlem yükü oluşturmaktadır. Sunucu bilgisayarının donanım kapasitesi istemci sayısını sınırlayacak bir faktördür. Deneysel çalışma sırasında kullanılan Intel Core-i7-4510U, çift çekirdekli, dört kanallı, 2.00 GHz işlemci hızı, 4 MB ön belleğe sahip işlemcili bir sunucu bilgisayarında altı adet istemci sorunsuz bir şekilde çalıştırılmıştır. Bu çalışmada minimum veri toplama hızının 30 milisaniye olduğu saptanmıştır. Bu süre, kullanılan internet alt yapısının bant genişliği arttırılarak veya kullanılan bilgisayarın donanım kapasitesi arttırılarak daha da azaltılabilir.

Çalışmada herhangi bir istemciden belirlenmiş veri toplama süresince veri gelmemesi durumunda şablon VI hataya geçer ve istemciyi sunucudan düşürür. Ayrıca TCP/IP protokolünün sağlamış olduğu hata kontrol algoritmaları da sunucuya adapte edilmiş ve kullanıcı uyarılmıştır. Sistemi casus istemcilerden korumak için sunucu üzerinde istemci kabul butonu kullanılmıştır. Sunucu ve istemcilerin haberleşmesi yerel ağ üzerinden gerçekleştirildiği için sistemin güvenlik sorunu bulunmamaktadır.

Geliştirilen sistem ile günümüzün popüler konularından biri olan nesnelerin interneti konusuna yeni bir bakış açısı getirilmiştir. Belli bir standardı olmamasından dolayı birçok sisteme alternatif olarak donanımdan bağımsız, özelleştirilebilir, kolay eklemeli bu IoT sistemi kullanılabilir. Böylece TCP/IP protokolünün gücünü LabVIEW yazılımı ile birleştirerek güçlü bir nesnelerin interneti sistemi oluşturmak mümkün olmuştur.

Tablo 5.1. Daha önce uygulanmış çalışma ile mevcut tez çalışmasının kıyaslanması

Parametreler	Esnek otomasyon sistemli veri takip sisteminin tasarımı ve uygulanması [5]	Endüstriyel kullanıma uygun LabVIEW ve gömülü sistem tabanlı yeni bir IoT çözümü
Sistem Hızı	Belirsiz	Sistem donanımına ve internet hızına bağlı olarak en az 30 milisaniye bir veri toplama
Maliyet	Maliyetli	Daha az maliyetli
Haberleşme protokolü	TCP/IP	TCP/IP ve İstemci kolay ekle-çıkar uygulama protokolü
Çoğaltılabilir yapı	Sahip değil	Kolay eklemeli
Yazılım Mimarisi	Statik	AMC – Dinamik
İstemci kullanılabilirliği	PLC	Donanımdan bağımsız
Maksimum istemci sayısı	8	Teorik olarak sınırsız
Sunucu yazılımı	LabVIEW	LabVIEW
İstemci yazılımı	PLC, LabVIEW	Yazılımdan bağımsız
Veri kaydetme seçenekleri	Text, Excel	Text, Excel, MySQL (MariaDB)
Kullanıcı uyarı sistemi	E-mail, LED	SMTP protokolü, E-mail, LED

Kaya ve arkadaşlarının gerçekleştirmiş olduğu çalışma [5] ile mevcut tez çalışmasının kıyaslanması gerçekleştirilmiştir. Bu kıyaslama sonucunda mevcut tez çalışmasının diğer çalışmaya göre üstünlükleri şöyle sıralanabilir:

- Her makine başına maliyetli bir bilgisayar koymak yerine mini bilgisayar olarak adlandırılan ve monitöre bağlanabilen ucuz endüstriyel tabanlı gömülü sistem kartları kullanarak çalışmanın maliyeti düşürülmektedir.
- Çalışmada kullanılan dinamik yapı üzerine kurulmuş AMC yazılım mimarisi ve oluşturulmuş ekle-çıkar uygulama protokolü sayesinde sunucu yazılımını değiştirmeden sisteme eklenecek yeni makinelere istemci kartları bağlanarak sunucuya entegre etmek kolay hale geldiğinden maliyet azalmaktadır.
- Donanımdan ve yazılımdan bağımsız istemci yapısı oluşturulduğundan tek bir tip veri toplama cihazına bağımlılık ortadan kaldırılarak farklı sistemlerin sunucuya entegre olması ihtiyaca yönelik çeşitliliği arttırmaktadır.

Geliştirilen sistemin sağladığı genel özellikler şu şekilde özetlenebilir:

- Sunucu LabVIEW programı ile yazılmış olup Windows tabanlı PC üzerinde çalıştırılmıştır.
- Veriler yerel veri tabanına kaydedilmiştir. Veri tabanı yönetim sistemi olarak MariaDB kullanılmıştır.
- Donanımdan ve yazılımdan bağımsız istemciler sunucuya çalışma anında kolayca eklenmiştir.
- Veri toplama süreleri her bir istemci için hem asenkron hem de çok kanallı bir şekilde gerçekleştirilmiştir.
- Uyarılar e-mail yoluyla kullanıcılara iletilmiştir.
- TCP/IP protokolü üzerinden özel uygulama protokolü gerçekleştirilmiştir.
- MSP430F5529 LP ve Raspberry Pi gömülü sistem donanımlarından sensör verileri başarılı bir şekilde toplanmıştır.

Tez çalışması sonrasında yapılabilecek diğer uygulamalar şu şekilde sıralanabilir:

- Veriler bulut sistemine gönderilerek dünyanın her yerinden erişebilir olabilir.
- Mobil uygulama yazılarak veriler mobil üzerinden takip edilebilir.
- Sunucu web üzerinden veya web sunucu üzerinden kontrol edilebilir.
- Kullanıcılara SMS ile uyarı sistemi de kurulabilir.
- Sunucu port yönlendirme ile dış dünyaya açılarak internete bağlı her istemciden veri toplanabilir. Ancak bu durumda gönderilen ve alınan verilerin karşılıklı şifrelenmesi gerekebilir.

KAYNAKLAR

[1] Ercan, T., Kutay, M., Endüstride Nesnelerin İnterneti (IoT) Uygulamaları. Afyon Kocatepe Üniversitesi Fen ve Mühendislik Bilimleri Dergisi, 599-607, 2016.

[2] http://www.endustri40.com/endustri-tarihine-kisa-bir-yolculuk/ Erişim Tarihi: 17.02.2018.

[3] Bozdoğan, Z., Nesnelerin İnterneti için Mimari Tasarımı. Düzce Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Bölümü, Yüksek Lisans Tezi, 2015.

[4] Chuan, O. W., Ruslan, S. H., Medical warehouse monitoring and control system using LabVIEW. International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), 2396-2401, 2016.

[5] Kaya, B., Altıntaş, A., Kök, Ü., Esnek Otomasyon Sistemli Veri Takip Sisteminin Tasarımı ve Uygulaması. Süleyman Demirel Üniversitesi Uluslararası Teknolojik Bilimler Dergisi (UTBD), 30-38, 2015.

[6] Bolivar, L. E. P., Silva, G. A., Solar radiation monitoring using electronic embedded system Raspberry Pi database connection MySQL, Ubidots and TCS-230 Sensor. CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON), 473-479, 2015.

[7] Swain, K. B., Dash, S., Gouda, S. S., Raspberry PI based Integrated Autonomous Vehicle using LabVIEW. 3rd International Conference on Sensing, Signal Processing and Security (ICSSS), 69-73, 2017.

[8] Vujovic, V., Maksimovic, M., Raspberry Pi as a Sensor Web node for home automation. Computers and Electrical Engineering, vol.44, 153-171, 2015.

[9] Jegan, R., Nimi, W. S., Low cost and improved performance measures on filtering techniques for ECG signal processing and TCP/IP based monitoring using LabVIEW. 4th International Conference on Advanced Computing and Communication Systems (ICACCS), 1-7, 2017.

[10] Shah, J., Mishra, B., Customized IoT enabled Wireless Sensing and Monitoring Platform for Smart Buildings. Procedia Technology, vol.23, 256-263, 2016.

[11] Hnidka, J., Rozehnal, D., Strain gauge measurement system with Wi-fi data transfer in LabVIEW. International Conference on Military Technologies (ICMT), 520-525, 2017.

[12] Gómez J. E., Marcillo F. R., Triana F. L., Gallo V. T., Oviedo B. W., Hernández V. L., IoT for environmental variables in urban areas. Procedia Computer Science, Vol. 109, 67-74, 2017.

[13] Prada M. A., Reguera P., Alonso S., Morán A., Fuertes J. J., Domínguez M., Communication with resource-constrained devices through MQTT for control education. IFAC-PapersOnLine, Vol.49, 150-155, 2016.

[14] Oksanen T., Linkolehto R., Seilonen I., Adapting an industrial automation protocol to remote monitoring of mobile agricultural machinery: a combine harvester with IoT. IFAC-PapersOnLine, Vol.49, 127-131, 2016.

[15] Shah D., Haradi V., IoT Based Biometrics Implementation on Raspberry Pi. Procedia Computer Science, Vol.79, 328-336, 2016.

[16] http://www.ni.com/en-us/shop/labview.html Erişim Tarihi: 24.02.2018.

[17] http://zone.ni.com/reference/en-XX/help/371361H-01/glang/start_asynchronous_call/ Erişim Tarihi: 24.02.2018.

[18] http://www.ni.com/white-paper/3023/en/ Erişim Tarihi: 24.02.2018.

[19] zone.ni.com/reference/en-XX/help/371361L-01/lvconcepts/data_comm/ Erişim Tarihi: 24.02.2018.

[20] http://vipm.jki.net/ Erişim Tarihi: 24.02.2018.

[21] http://www.ni.com/example/31091/en/ Erişim Tarihi: 24.02.2018.

[22] https://www.tsxperts.com/labviewforraspberrypi/ Erişim Tarihi: 25.02.2018.

[23] https://www.ahmetiscan.web.tr/mysql-nedir-nerelerde-kullanilir-ozellikleri-nelerdir/ Erişim Tarihi: 25.02.2018.

[24] https://mariadb.org/about/ Erişim Tarihi: 25.02.2018.

[25] https://www.heidisql.com/ Erişim Tarihi: 25.02.2018.

[26] www.mcu-turkey.com/wp-content/uploads/2011/03/EnergiaProgramlama.pdf Erişim Tarihi: 02.03.2018.

[27] https://www.raspberrypi.org/products/raspberry-pi-2-model-b/ Erişim Tarihi: 04.03.2018.

[28] Ray, P. P., A survey on Internet of Things architectures. Journal of King Saud University – Computer and Information Sciences, 1319-1578, 2016.

[29] www.ti.com/tool/MSP-EXP430F5529LP Erişim Tarihi: 04.03.2018.

[30] https://www.element14.com/community/roadTestReviews/1893/l/ti-wi-fi-cc3200-launchpad-cc3100-boosterpack-review Erişim Tarihi: 04.03.2018.

[31] http://tinkbox.ph/sites/tinkbox.ph/files/downloads/Proximity%20Sensor.pdf Erişim Tarihi: 10.03.2018.

[32] www.ti.com/lit/ds/symlink/lm35.pdf Erişim Tarihi: 10.03.2018.

[33] http://www.energiazero.org/arduino_sensori/2_channel_5v_10a_relay_module.pdf Erişim Tarihi: 11.03.2018.

[34] Özgü, A. M., Bir Elektronik Panonun Genel Amaçlı Uzaktan Kontrolü. Gazi Üniversitesi, Bilişim Enstitüsü, Elektronik ve Bilgisayar Eğitimi, Yüksek Lisans Tezi, 2011.

[35] http://www.elektrikport.com/teknik-kutuphane/tcpip-nasil-calisir/9004#ad-image-0 Erişim Tarihi: 17.03.2018.

[36] Ağ Güvenliği Ve Ağ Protokolleri. Elektrik-Elektronik Teknolojileri Bölümü, Milli Eğitim Bakanlığı Yayınları, 2011.

[37] http://www.ni.com/white-paper/2710/en/ Erişim Tarihi: 18.03.2018.

EKLER

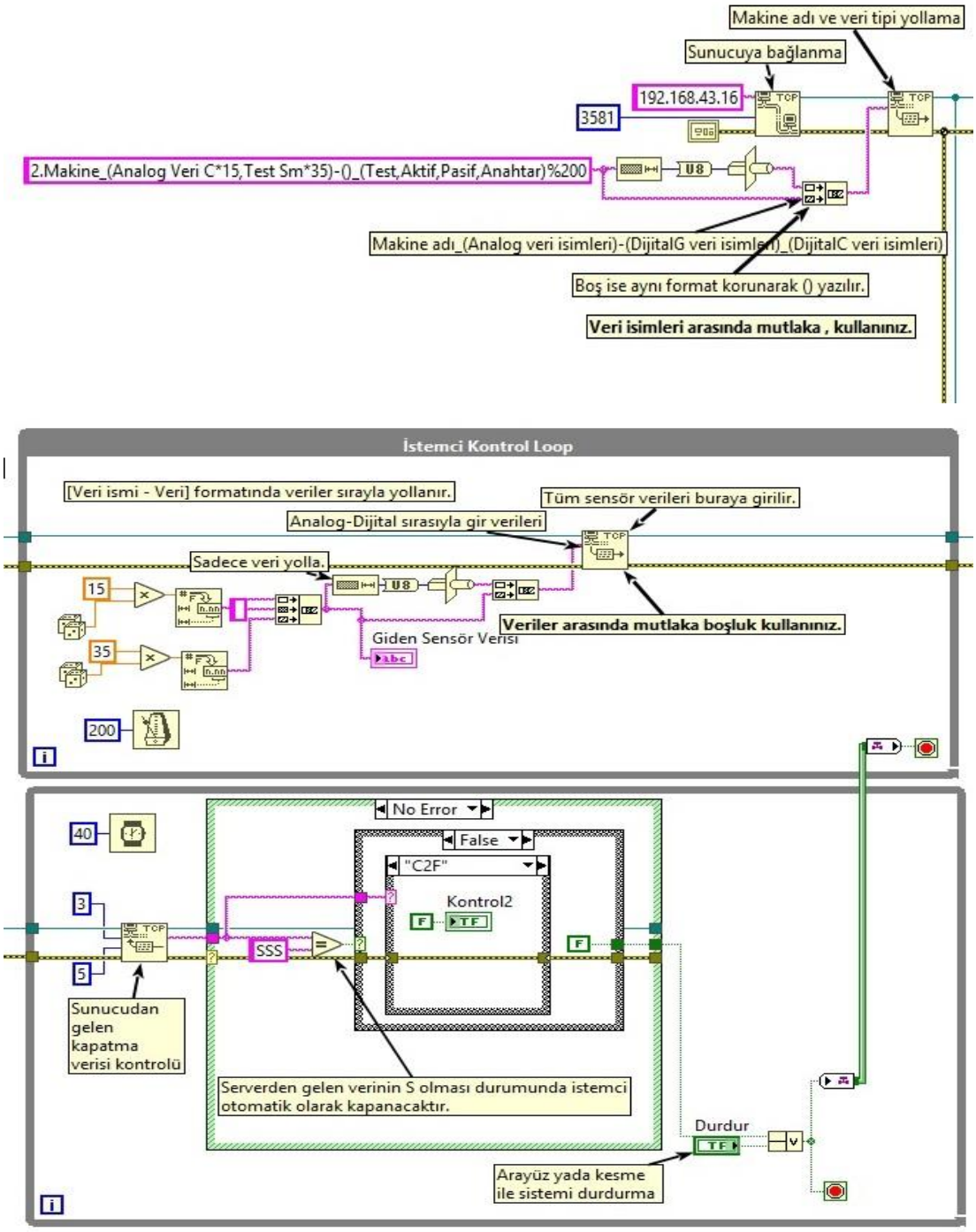
EK 1: MSP430F5529 istemcisi program parçaları

```
#include <SPI.h>
#endif
#include <WiFi.h>
#define LED GREEN_LED
int TempPin = A6;
int TempVal = 0;
String GetValue = "";
char ssid[] = "AndroidAP";
char password[] = "123456789";
IPAddress server(192,168,1,16);
WiFiClient client;
void setup()
{
  pinMode(LED, OUTPUT);
  digitalWrite(LED,LOW);
  String InitValue = "MSP_(Temperature C*80)-()_(Green led)%500";
  char SCharLength = char (InitValue.length());
  String TotalValue =  SCharLength + InitValue;
  Serial.begin(115200);
  Serial.print("Adlandirilan aga baglaniliyor: ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while ( WiFi.status() != WL_CONNECTED)
  {Serial.print(".");
```

```
    delay(300);}
  Serial.println("\n Aga baglanildi.");
  Serial.println("IP adresi bekleniyor.");
  while (WiFi.localIP() == INADDR_NONE)
  {Serial.print(".");
    delay(300); }
  Serial.println("\n IP adresi elde edildi.");
  printWifiStatus();
  Serial.println("\n Sunucuya baglaniliyor....");
  if (client.connect(server, 20))
  {Serial.println("Sunucuya baglandi.");
    client.print(TotalValue);
    Serial.println(TotalValue); }
   }
void loop()
{
  if (!client.connected())
   { Serial.println();
     Serial.println("Sunucu ile baglanti koptu.");
     client.stop();
     delay(3000);
   }
  else
  {
   if(client.available())
   {
      GetValue = "";
      delay(10);
      for(int i=0;i<3;i++)
      {
      char c = client.read();
      GetValue += c;
```

```
      }
    if(GetValue == "SSS")
      {digitalWrite(LED,LOW);}
    else if(GetValue == "C1T")
     { digitalWrite(LED,HIGH);}
    else
    { digitalWrite(LED,LOW); } }
   else
   {
     TempVal = analogRead(TempPin);
     TempVal /= 12.42;
     Serial.println(TempVal);
     String TempNew = String (TempVal);
     Serial.print("Temp=");
     Serial.print(TempVal);
     char TempLen = char (TempNew.length());
     String TotalTemp = TempLen + TempNew;
     client.print(TotalTemp);
     delay(500);
    } } }
void printWifiStatus()
{
  Serial.print("SSID: ");
  Serial.println(WiFi.SSID());
  IPAddress ip = WiFi.localIP();
  Serial.print("IP Address: ");
  Serial.println(ip);
  long rssi = WiFi.RSSI();
  Serial.print("Sinyal gucu (RSSI):");
  Serial.print(rssi);
  Serial.println(" dBm");
}
```

EK 2: Sanal istemci program parçaları



ÖZGEÇMİŞ

Görkem Sungur, 13.08.1992'de Bursa'da doğdu. İlk, orta ve lise eğitimini Bursa'da tamamladı. 2010 yılında Gemlik Lisesi'nden mezun oldu. 2010 yılında başladığı Erciyes Üniversitesi Mekatronik Mühendisliği Bölümü'nü 2015 yılında bitirdi. 2015 Eylül ayında Sakarya Üniversitesi Fen Bilimleri Enstitüsü, Mekatronik Mühendisliği Anabilim Dalı'nda Yüksek Lisans eğitimine başladı. 2016 Eylül ayında TOFAŞ Türk Otomobil Fabrikaları A.Ş' de dönem tabanlı proje çalışmasında bulundu. 2017 Ocak ayında Ermaksan Optoelektronik Ar-Ge Merkezi'nde Yazılım Mühendisi olarak çalışmaya başladı. Halen Ermaksan Optoelektronik Ar-Ge Merkezi'nde Yazılım Mühendisi olarak çalışmaya devam etmektedir.