T.C.
KONYA TEKNİK ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

ENDÜSTRİYEL MUTFAK EKİPMANLARININ MERKEZİ KONTROLÜ İÇİN IOT İLE VERİ TOPLANMASI VE LINUX TABANLI HMI TASARIMI

Ahmet Yasin BİLİCİ

YÜKSEK LİSANS TEZİ

Elektrik Elektronik Mühendisliği Anabilim Dalı

Nisan-2023
KONYA
Her Hakkı Saklıdır

TEZ BİLDİRİMİ

Bu tezdeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edildiğini ve tez yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade ve bilginin kaynağına eksiksiz atıf yapıldığını bildiririm.

DECLARATION PAGE

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Ahmet Yasin BİLİCİ
04.04.2023

ÖZET

YÜKSEK LİSANS TEZİ

ENDÜSTRİYEL MUTFAK EKİPMANLARININ MERKEZİ KONTROLÜ İÇİN IOT İLE VERİ TOPLANMASI VE LINUX TABANLI HMI TASARIMI

Ahmet Yasin BİLİCİ

Konya Teknik Üniversitesi
Lisansüstü Eğitim Enstitüsü
Elektrik Elektronik Mühendisliği Anabilim Dalı

Danışman: Prof. Dr. Levent SEYFİ

2023, 87 Sayfa

Jüri
Prof. Dr. Levent SEYFİ
Prof. Dr. Ercan YALDIZ
Dr. Öğr. Üyesi Yunus Emre ACAR

Günümüz endüstriyel mutfak teknolojisi öylesine ilerlemiş ve önem kazanmıştır ki; neredeyse endüstriyel mutfak ürünlerinin %80'inde elektronik kontrol panelleri ve haberleşme sistemleri kullanılmaktadır. Son zamanlarda bazı endüstriyel mutfak ürünlerinin HMI denilen kullanıcı kumanda panelleri, telefon veya tabletleri aratmamaktadır. Reçete sistemleri, akıllı pişirme algoritmaları, akıllı yıkama algoritmaları, çapraz bulaşmalar, kalori hesaplamaları, etkili soğutma sistemleri vb. daha birçok özellik, başta aşçıların işini kolaylaştırmakla birlikte yiyecek kalitesini de üst düzeye çıkarmaktadır. Endüstriyel mutfak makinelerinde kullanılan sıcaklık, nem, kapı, termik vb. sensörlerden alınan bilgi, arıza ve hatalar kontrol kartlarına gelerek işlenir. Gerekli giriş çıkışlar kullanılarak motor, fan, rezistans, buhar jeneratörü, klape, lamba, ısıtıcı ve valfler gibi donanımlar kontrol edilmektedir.

Bu tez çalışmasında Endüstriyel Mutfak Makinalarından Modbus haberleşme ile alınan bilgilerin işlenerek Linux tabanlı HMI(Human Machine Interface)'lerde oluşturulan GUI(Graphical User Interface) üzerinden gösterilmesi ve tüm verilerin buluta aktarılarak çift yönlü veri akışı sağlanması ile IoT(Internet of Things) uygulaması çalışılmıştır. Bir web yazılımı ile tek merkezden büyük endüstriyel mutfakların uzaktan kontrolü, otomasyonu, toplanan veriler sayesinde satın alma, bakım onarım planları, hijyen kontrolü ve önceden alınan tedbirlerin artırılması hedeflenmiştir.

Anahtar Kelimeler: Endüstriyel Mutfak Kontrol Panelleri, HMI, Kontrol Kart, PCB, Modbus, IoT, MQTT, Linux, Qt-Qml GUI Tasarım, Uzaktan Veri Toplama, Tam Yığın Web Geliştirme

ABSTRACT

MS THESIS

DATA COLLECTION AND LINUX BASED HMI DESIGN WITH IOT FOR CENTRALIZED CONTROL OF INDUSTRIAL KITCHEN EQUIPMENTS

Ahmet Yasin BİLİCİ

Konya Technical University
Institute of Graduate Studies
Department of Electrical Electronics Engineering

Advisor: Prof.Dr. Levent SEYFİ

2023, 87 Pages

Jury
Prof.Dr. Levent SEYFİ
Prof.Dr. Ercan YALDIZ
Asst.Prof.Dr. Yunus Emre ACAR

Today's industrial kitchen technology has advanced so much and gained importance that, Electronic control panels and communication systems are used in almost 80% of industrial kitchen products. Recently, the user control panels of some industrial kitchen products, called HMIs, seem to be phones or tablets. Recipe systems, smart cooking algorithms, smart washing algorithms, cross contamination, calorie calculations, effective cooling systems, etc. many more features not only make the work of the cooks easier, but also increase the quality of the food. Information, malfunctions and errors obtained from sensors such as temperature, humidity, door, thermal etc. used in industrial kitchen machines are processed by coming to the control cards. Equipment such as motor, fan, resistance, steam generator, damper, lamp, heater and valves are controlled by using the necessary inputs and outputs.

In this thesis, the information received from Industrial Kitchen Machines by Modbus communication is processed and displayed via GUI (Graphical User Interface) created on Linux-based HMI (Human Machine Interface) and all data is transferred to the cloud, providing bidirectional data flow control and IoT (Internet of Things) application has been studied. With a web software, it is aimed to remote control and automation of large industrial kitchens from a single center, to increase purchasing, maintenance and repair plans, hygiene control and it is aimed to increase the measures taken previously.

Keywords: Industrial Kitchen Control Panels, HMI, Control Card, PCB, Modbus, IoT, MQTT, Linux, Qt-Qml GUI Design, Remote Data Collection, Full Stack Web Development

ÖNSÖZ

Bu yüksek lisans tezinin hazırlanmasında başta manevi desteklerini esirgemeyen sevgili eşime, bana hayatı boyunca ilerlediğim yolda nasıl hareket etmem gerektiğini gösteren rahmetli babama, anneme, abilerime, beni neşelendiren oğullarıma ve öğrenim hayatım boyunca emeği geçen tüm hocalarıma sonsuz teşekkürü bir borç bilirim.

Yardımlarını hiçbir zaman esirgemeyen ve değerli bilgilerini bizimle paylaşan tez danışmanım Sn. Prof. Dr. Levent SEYFİ 'ye, değerli katkılarını esirgemeyen dostlarıma sevgi ve saygılarımı sunarım.

Ahmet Yasin BİLİCİ
KONYA-2023

İÇİNDEKİLER

KISALTMALAR

API	: Application Programming Interface
CSS	: Cascading Style Sheets
GB	: Giga Byte
GPIO	: General-Purpose Input/Output
GUI	: Graphical User Interface
HDMI	: High Definition Multimedia Interface
HMI	: Human Machine Interface
HTML	: Hypertext Markup Language
IDE	: Integrated Development Environment
IoT	: Internet of Things
I/O	: Input / Output
LSB	: En Az Anlamlı Bit
MCC	: MPLAB Code Configurator
MSB	: En Anlamlı Bit
MQTT	: Message Queuing Telemetry Transport
NVM	: Non-Volatile Memory
PCB	: Printed Circuit Board
PLC	: Programmable Logic Controller
PWM	: Pulse Width Modulation
RAM	: Random Access Memory
RTU	: Remote Terminal Unit
UART	: Universal Asynchronous Receiver Transmitter
QML	: Qt Modelleme Dili

1. GİRİŞ

Endüstriyel mutfaklar miktar ve çeşit açısından çok sayıda yiyecek üretilen, depolanan ve teşhir edilen yerlerdir. Endüstriyel Mutfak sektörü Türkiye'de oldukça öneme sahip, fazla istihdam sağlayan ve ülkemiz ihracatında büyük rol oynayıp ülke ekonomisine katkısı oldukça fazla olan bir sektördür (Yaylacı, 2017). Ayrıca Türkiye'deki Endüstriyel Mutfak firmaları, Avrupa'da tanınmış ve sektörün öncülerinden kabul edilmektedir. Şekil 1.1'de örnek bir mutfak görüntüsü görülmektedir.

Günümüzde şehirleşmenin hızla yayılması, hayat tarzlarının değişimi ile vazgeçilmez olan yeme-içme ihtiyacı, pandemi gibi olayların da etkisiyle hızlı bir teknolojik ilerleyiş içerisindedir. Uzaktan siparişler, mutfakların hijyen kurallarının kontrol edilmesi, nüfusun artması, fabrika ve otel gibi toplu alanların çoğalması, gıda çeşitliliğinin ve kullanımının fazlalaşması, herkesin ortak noktası olan yeme-içme ihtiyacına da kolay ve hızlı çözümler üretilmesi gerekliliğini oluşturmuştur (Kahraman ve ark., 2018).

Şekil 1.1. Örnek bir endüstriyel mutfak görünümü

Endüstriyel fırınlarda pişirme işlemlerinde sensörlerden gelen verilere göre gelişmiş yazılımlar ve elektronik teknikler kullanılmaktadır. Bir tavuğun marine edilmiş veya derisinin soyulup soyulmamasına göre kızarma şekli ve kızardıktan sonraki rengi elektronik bileşenler ve yazılımlar sayesinde kontrol edilebilmektedir (O'farrell ve ark., 2005). Reçete sistemleri, akıllı pişirme algoritmaları, akıllı yıkama algoritmaları, farklı türde yemeklerin çapraz bulaşmaları, kalori hesaplamaları, etkili soğutma sistemleri vb. daha birçok özellik, başta aşçıların işini kolaylaştırmakla birlikte yiyecek kalitesini de üst düzeye çıkarmaktadır. Tanınmış zincir firmalardaki ürünlerin kalitesinin dünyanın her

yerinde aynı olması, hamurların depolanması ve hazırlanış yöntemlerinin sabit olması, restoran ve kafelerde kullanılan tabak ve çanakların saniyeler içinde yıkanarak hazır hale gelmesi, soğutma dolaplarındaki eşit ısı dağılımlarının olması, sebze meyvelerin el değmeden soyulması, tonlarca ağırlıkta hamurların büyük mikserlerde karıştırılması vb. görevler, kontrol kartlarının sensörlerden aldığı bilgilerle makine donanımlarını yönetmesiyle sağlanmaktadır (Rational, 2022).

Günümüz Endüstriyel Mutfak makinelerinde elektronik ve haberleşme öylesine büyük bir önem kazanmıştır ki makineler eskiden basit butonlar aracılığıyla manuel olarak kontrol edilirken şimdi HMI (Human-Machine Interface) denilen kullanıcı kumanda panelleriyle kontrol edilmektedir (Konopek, 2002). HMI'dan gönderilen bilgiler kontrol kartına ulaşarak motor, fan, rezistans, buhar jeneratörü, klape, lamba, ısıtıcı, valf gibi donanımlar yönetilebilmektedir. Aynı zamanda kontrol kartına gelen sensörlerden alınan veriler HMI'ya gönderilerek yorumlanır ve optimum kontrol sağlanır.

Kullanıcılar, dijital sistemler ile yapmak istedikleri işleri manuel sistemlere kıyasla daha etkili ve verimli bir biçimde yapabilmektedir (Eryılmaz, 2015). HMI'larda genellikle yüksek hızlı işlemciler kullanılmaktadır. HMI elektronik kartına wi-fi, ekran sürücü, ram, ethernet, USB, dokunmatik ekran, EPROM gibi donanımlar eklenmiştir. Şekil 1.2'de fırınlar için tasarlanmış bir HMI görülmektedir.

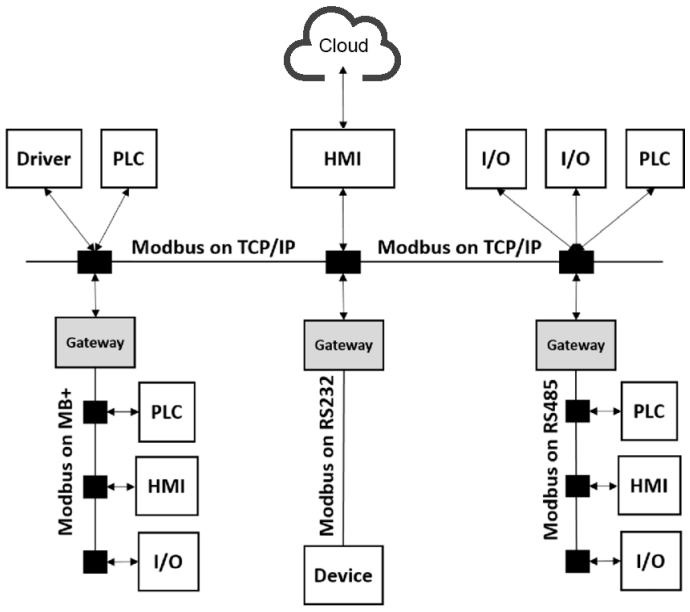
Şekil 1.2. Fırınlarda kullanılan bir HMI

Endüstriyel mutfak ekipmanlarında kullanılan çoğu HMI'ların internet uyumu bulunmamaktadır. Uyumlu olanların anlaşılır şekilde işlem yapması oldukça zahmetli ve

maliyetlidir. Bu bakımdan nesnelerin uzaktan kontrolü ve veri toplama sistemi dediğimiz IoT sisteminin rahatlıkla sağlanabilmesi için basit ve anlaşılır bir GUI tasarımının Linux sistem kullanarak geliştirilmesi önem arz etmektedir. Linux sistem sayesinde endüstriyel mutfak makinalarının işlevleri üst seviyelere taşınacaktır. Ayrıca düşük enerji tüketimi önemli bir avantajdır (Choi ve Moon, 2007). Var olan sistemlerde bu işlemler birkaç farklı ürün ile yapılırken hedeflenen sistem sayesinde medya da dâhil olmak üzere tek noktadan kontrol edilmesi planlanmaktadır. Böylece hem maliyet oldukça düşürülmüş hem de mutfak makinası üzerindeki karmaşıklık azaltılmış olacaktır.

Günümüzde mevcut endüstriyel otomasyon sistemlerinin IoT ile beraber kullanılmasında, dünyanın herhangi bir yerinden gerçek zamanlı olarak üretim ve arızalarla ilgili bilgi alma, yönetme ve hammadde ihtiyaçlarını belirleme gibi birçok avantajı vardır (Ercan ve Kutay, 2016).

Bir endüstriyel otomasyonda aynı ortamda bulunan makinaların, buluta çıkmadan önce, hem kendi aralarında hem de aynı hatta bağlı HMI ile haberleşmesi gerekmektedir. Şekil 1.3'te cihazların birbiriyle iletişimi görülmektedir. IoT teknolojisini tam anlamıyla uygulayabilmek için makinaların kendi aralarında ve HMI ile veri alışverişinin sağlıklı olması gerekmektedir. Makinalar arasında Modbus haberleşme kullanımın yaygın ve güvenli olduğu ile ilgili birçok endüstriyel fabrika otomasyonu uygulaması mevcuttur (Rholam ve Tabaa, 2019).



Şekil 1.3. Cihazlar arasında Modbus haberleşme

1.1. Tezin Amacı

Bu tez çalışmasında kendi aralarında Modbus haberleşme yapan Endüstriyel Mutfak Makinalarından alınan bilgilerin işlenerek Linux tabanlı HMI'lerde oluşturulan GUI (Graphical User Interface) üzerinden gösterilmesi ve tüm verilerin buluta aktarılarak çift yönlü veri akışı ve kontrolün sağlanmasıyla etkin bir IoT uygulaması hedeflenmiştir. Aynı şekilde uzaktan gönderilen tüm komutlar da, HMI tarafından haberleşme yapılan endüstriyel mutfak makinalarına aktarılarak kontrol işlemi sağlanması düşünülmüştür.

Endüstriyel mutfak makinalarının kendi arasında ve HMI ile kuracak olduğu haberleşme, yaygın olması nedeniyle Modbus ile olacaktır. Verilerin sorunsuz aktarımları incelenip uygulanacaktır. Haberleşme yapacak PCB'ler dünyada kabül görmüş ve endüstriyel projelerde sık sık kullanılan profesyonel Altium Designer programı ile tasarlanacaktır. Yazılımlarında ise temel bir dil olan C programlama dili kullanılacaktır.

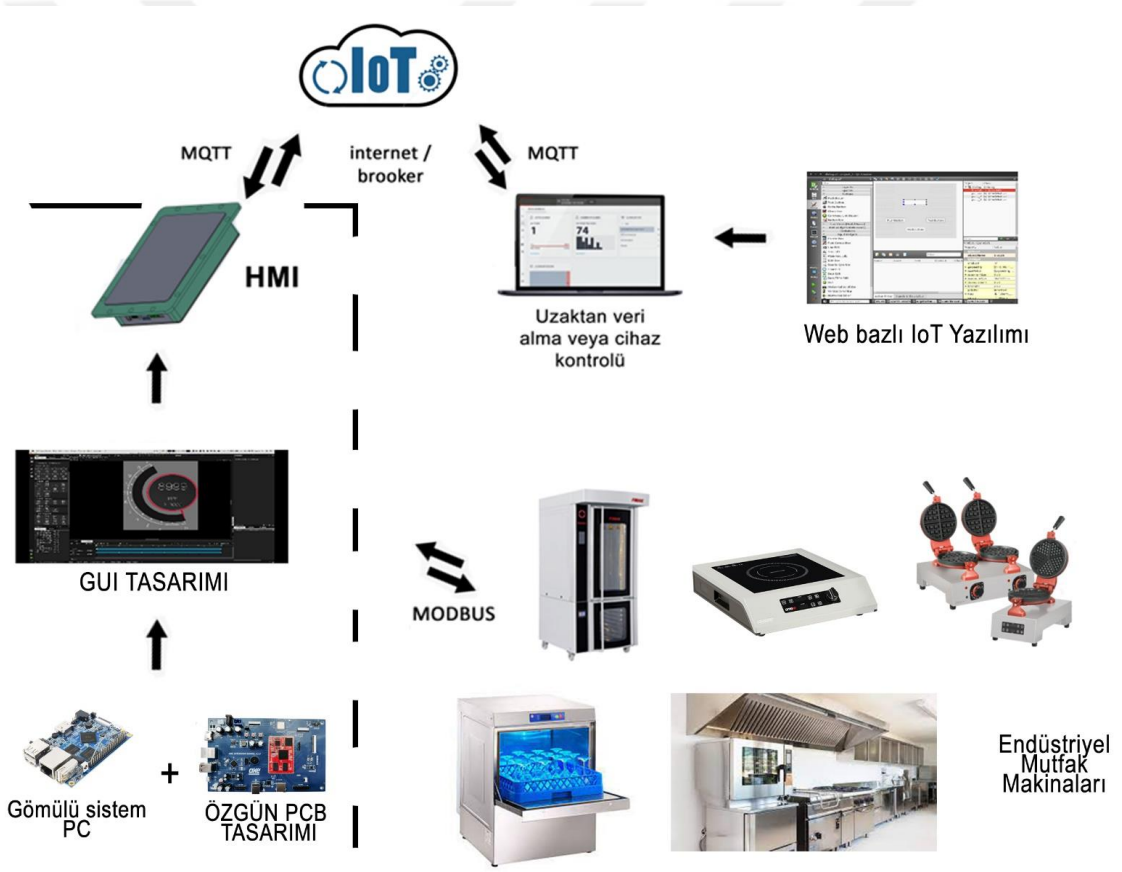
HMI tarafında Linux tabanlı bir mikrodenetleyici kullanılacaktır. Sistemlerin haberleşmesi ve anlaşılır şekilde ekrandan kullanıcıya gösterilmesi oldukça zahmetli ve maliyetledir. Bu nedenle daha basit ve anlaşılır bir GUI tasarımının Linux sistem kullanılarak geliştirilmesi önemli olmaktadır. Linux kabiliyetin fazla olduğu açık kaynak bir ortam olduğundan sistem yazılım güvenliği de bir üst seviyeye çıkarılacaktır. GUI ekran tasarımı ise sistemi fazla yormaması amacıyla hafif olduğu bilinen Qt-Qml ve C++ yazılım dili kullanılarak gerçekleştirilecektir.

IoT uygulaması için hafif bir protokol olmasından dolayı Mqtt protokolü uygulanacaktır. Buluttaki verileri toplayıp, işleyerek kontrol etmek için bir web arayüzü yapılacaktır. Web ara yüzünün front-end, back-end ve veri tabanı kısımları oluşturulacaktır.

Tüm bu işlemlerin tek merkezden kontrol edilmesi, sahadan veri toplanarak analizler yapılması ve bu analizlerle yeni planlar oluşturulması hedeflenmektedir. Bu sayede tek merkezden büyük endüstriyel mutfakların kontrolü ve otomasyonu, toplanan veriler sayesinde satın alma planları, bakım onarım planları, hijyen kontrolü ve önceden alınan önlemlerin artırılması beklenmektedir. Planlanan bu aşamalar sayesinde oluşması muhtemel arıza tespitinin önceden fark edilebilmesi ve önlem alınabilmesi mümkün olacaktır.

Sektörde küçük bir waffle makinasının bile IoT kontrolü istenebilmektedir. Şöyle ki, üretici waffle makinasını müşterilerine ücretsiz verecek olup hammadde tozunu satarak para kazanmak istemektedir. Pişirilen waffle sayısını uzaktan kontrol ederek hammadde bitmeye yakın yenisini gönderecektir. Eğer waffle makinasındaki pişirme sayısı gönderdiği hammadde miktarını geçmişse makinaları kilitleyebilecektir. Bu gibi merkezden IoT ile kontrol yapılmak istenmesi gibi durumlar Endüstriyel Mutfak sektöründe trendin bu yöne gittiğini göstermektedir.

Yapılan bu tez çalışması ile Şekil 1.4'te gösterilen yapı kullanılarak yukarıda bahsedilen işlemler incelenip uygulaması yapılacaktır.



Şekil 1.4. Tez çalışmasında gerçekleştirilecek sistem yapısı

2. KAYNAK ARAŞTIRMASI

Arslan ve ark., 2016 yılında yaptıkları çalışmada mikro denetleyici içeren bir gömülü sistem için çoklu ortam uygulaması gerçekleştirmişlerdir. Tasarlanan Linux işletim sistemi ile video oynatma, ses çalma, veri okuma ve kaydetme, ekran görüntüsü kaydetme, diske yazma gibi birçok özelliğe sahip olmasını sağlamışlardır. Geliştirilen kullanıcı kütüphaneleri ve uygulamalar ile çoklu ortam işlemleri sağlanmıştır. Ayrıca, bilgisayar sürücülerine özel radyo frenkası (RF) kart okuma, yolcu bilgisi girişi, merkez haberleşmesi, yolcuları ses ve görüntü olarak bilgilendirme gibi işlemlerin tasarlanan sistem ile gerçekleştirilebileceğinden bahsedilmiştir.

Beg ve ark., 2021 yılında yaptıkları çalışmada IoT'nin güvenli bir veri iletim sistemi olduğundan bahsetmişlerdir. Son kullanıcılar için kullanışlı olduğu ve halen geliştirilmeye açık olduğu ifade edilmiştir.

Grunert ve ark., 2021 yılında yaptıkları bir çalışmada COVID-19 pandemi vakasının insanların yeme alışkanlıkları üzerindeki etkisini araştırmışlardır. Birtakım insanların yeme-içme esnasında eskisi gibi rahat olmadıklarına, birtakım insanların ise evde daha çok yeme-içme işleriyle ilgilenip pratik çözümler istemesiyle karşılaşmışlardır. Sonuç olarak pandemi etkisiyle endüstriyel mutfak sektöründe kolaylaştırıcı teknolojilerin uygulanması gerektiği gözlenmiştir.

Hintaw ve ark., 2021 yılında yapmış oldukları bir çalışmada nesnelerin internetinde (IoT) kullanılan iletişim protokollerinden bahsetmiştir. Bu protokollerin arasında en çok tercih edilenin MQTT protokolü olduğu sonucuna varılmıştır. Birçok nesnenin üzerindeki sensörlerin verisi oldukça büyük olmakta ve anlık olarak aktarılması gerekmektedir. Elde edilen yoğun miktardaki verinin işlenmesinde MQTT protokolünün farkından bahsedilmiştir.

Kekre ve Kothari, 2022 yılında yaptıkları bir çalışmada nesnelerin interneti (IoT) uygulamaları için Modbus haberleşme kullanımının yaygın ve uygun olduğu ile ilgili sonuç çıkarmışlardır. Modbus-RTU, uygulanması kolay, çok sağlam ve çok daha az bellek gerektirdiği için çok tercih edilen bir protokol olduğu vurgulanmıştır.

Livinsa ve ark., 2021 yılında yaptıkları çalışmada Nesnelerin interneti (IoT) ile otomatik yemek pişirme uygulaması yapmıştır. İş yoğunluğu olan yerlerde pişirme işlemini çok basit, sorunsuz ve daha az zaman alan bir eylem haline getirmeyi amaçlamışlardır.

Mozumder ve Ghosh, 2018 yılında yapmış oldukları bir çalışmada IoT ile uzaktan sayaç okuma ve ödeme yapılması uygulamasını sunmuştur. Bu çalışmada bir dijital enerji sayacı, IoT cihazı aracılığıyla bulut sunucusuna bağlanmıştır. Böylece bağlı müşterinin tükettiği enerji miktarı web sunucusuna gönderilir ve ödenmemiş borç durumunda müşterinin bağlantısını kesme özelliği ve internet üzerinden fatura ödeyerek bağlantı yenileme seçeneği sağlanmıştır.

Nayan ve ark., 2022 yılında yaptıkları bir çalışmada hem ev hem de ticari mutfaklarda artan kaza oranlarına dikkat çekmiştir. Nesnelerin interneti (IoT) ile herhangi bir kaza olasılığının önceden tahmin edilmesi üzerine durmuşlardır. IoT teknolojisiyle endüstriyel mutfak sektöründe hayati güvenlik önlemleri alınabileceği anlaşılmıştır.

Song ve ark. 2012 yılında yapmış oldukları çalışmada düşük verimlilik, zayıf taşınabilirlik vb. gibi mevcut gömülü grafik kullanıcı arabirimi geliştirmede var olan bazı sorunlara karşı, Qt/Embedded'e dayalı gömülü bir GUI geliştirme yönteminin iyi bir tercih olacağını savunmuşlardır. X11 sanal derleyici platformunun yapım teknolojisi ve çapraz derleme ortamının uygulama teknolojisi, GUI geliştirmede gerekli olan ayrıntılarla açıklanmakta ve Qt kitaplığı dosyası kullanılmaktadır. Sonuç olarak, Qt/Embedded tabanlı geliştirme yönteminin işlevselliğini uygulamalar aracılığıyla ortaya koymuşlardır.

Sun ve ark., 2021'de yaptıkları bir çalışmada 5G ve nesnelerin interneti (IoT) çağına girerken, insanlara dijital dünyayla daha sezgisel etkileşim sağlayabilen insan-makine arayüzlerinin (HMI'lerin) son birkaç yılda daha fazla gelişmenin olduğundan bahsetmektedir. IoT için HMI'lerin kullanımının kolaylığı vurgulanmıştır.

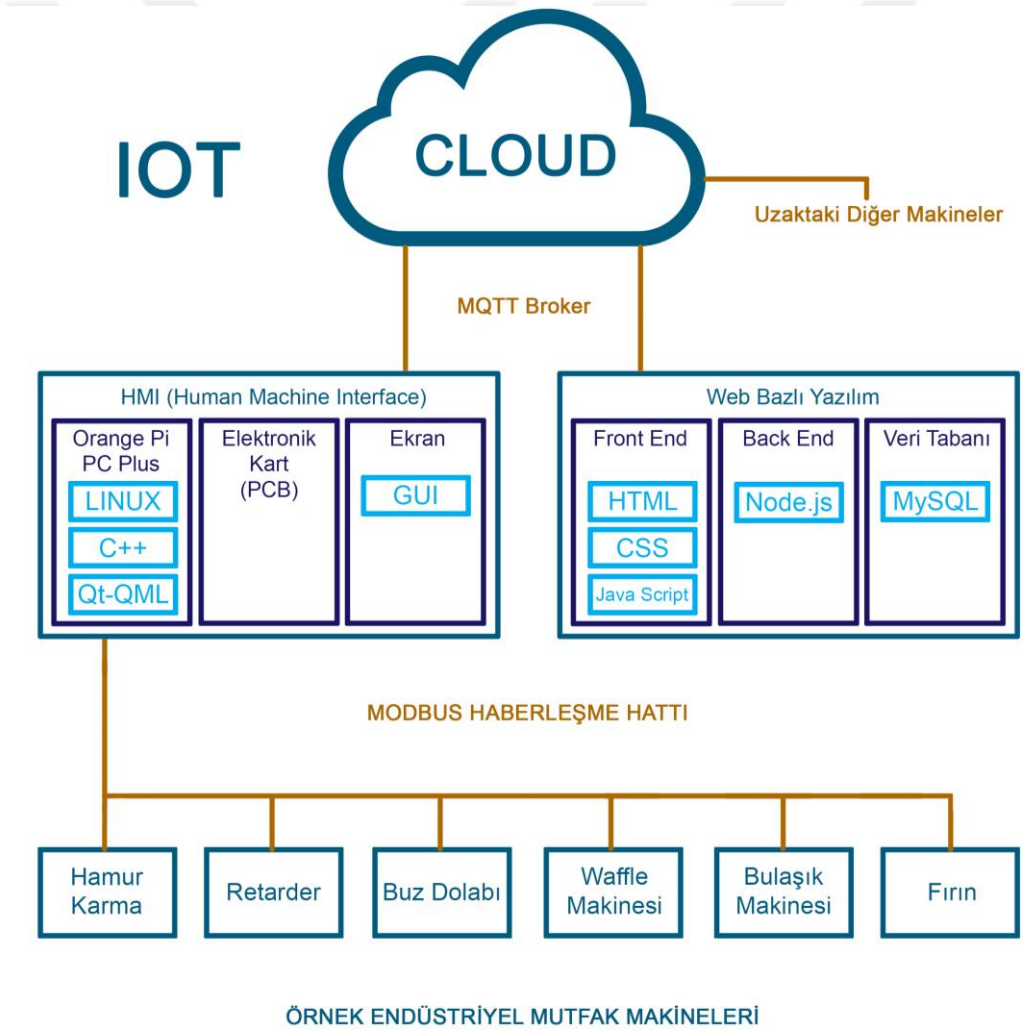
Umapathi ve Sabbani, 2022 yılında yaptıkları bir çalışmada Nesnelerin interneti (IoT) ve mikrodenetleyici tabanlı sensörler kullanarak otomatik bir mutfak izleme sistemini gerçekleştirmişlerdir. Çalışmada izleme ve kontrol, bir mobil uygulama üzerinden yapılmıştır.

Zhang ve ark., 2021 yılında yaptıkları bir çalışmada MQTT protokolü kullanarak bir HMI kodlanmasını gerçekleştirmişlerdir. Bu sistemin IoT ile kontrolünün sağlamış olduğu kolaylıklara vurgu yapılmıştır.

3. MATERYAL VE YÖNTEM

3.1. Kurgulanan Sistemin Yapısı ve Gereksinimleri

Tez çalışmasında ana amaç bilginin en alt seviyedeki makineden alınıp en üst seviyedeki buluttan geçerek işlenmesi ve aynı şekilde en üst seviyeden gelen komutların ise en alt seviyedeki makineleri yönetmesidir. Bu sistem sayesinde bir işletmenin veya zincirin endüstriyel mutfak otomasyonunu gerçekleştirebilmesi ve endüstriyel mutfak satıcılarının müşterilerinden veri toplaması mümkün olabilecektir. Sistemin proje aşamaları, devre tasarımlarının oluşturulması, kullanılacak malzemeler, sistem yazılımı ve sistemin bütün olarak bir araya getirilmesi, kurulumu gerçekleştirilen sistemin bileşenleri Şekil 3.1'de görülebilmektedir.



Şekil 3.1. Kurulumu gerçekleştirlecek sistemin bileşenleri

Gerçekleştirilen sistemde bulunması gereken bileşenler en alt seviyeden buluta doğru gidilecek olursa aşağıda listelenmiştir.

1. Endüstriyel Mutfak Makinalarını kontrol eden elektronik kart tasarımı ve yazılımına ihtiyaç vardır.

2. Makinaların kontrol kartlarının birbirleriyle haberleşme yapabilmesi için aralarında bir haberleşme protokolüne ihtiyaçları vardır.

3. Aynı haberleşme ağına bağlı insan makine iletişimini sağlamak ve aynı iletişimi bulutla kurmak üzere HMI ihtiyacı vardır.

4. HMI'da bulunması gereken mikrodenetleyici, ekran ve çevre bileşenleri kartına ihtiyaç vardır.

5. HMI'da bulunacak mikrodenetleyiciye ekranı sürmesi, buluta bağlanması ve çevre elemanlarını çalıştırması için bir işletim sistemi kurulması gereklidir.

6. HMI ile bulut arasında IoT sistemin oluşturulabilmesi için bir protokole ihtiyaç vardır.

7. Bulutta depolanan verileri alıp işleyip yönetecek bir web bazlı yazılıma ihtiyaç vardır.

8. Bütün bu işlemlerin tam ters yönde de gerçekleşmesi gerekmektedir.

Tez çalışmasında bahsedilen gereksinimlere göre sistem oluşturulmuştur. Endüstriyel Mutfak Makinalarını kontrol kartının devre şeması çıkarılmıştır. Altium Designer programı kullanılarak PCB tasarımı yapılmıştır. Profesyonel baskı devreye gönderilmiş ve üzerinde bulunan komponentler satın alınarak dizgisi yapılmıştır. Kontrol kartında Microchip PIC32MX120F064H işlemci kullanılmış olup MplabX programında C dili ile yazılmıştır.

Kontrol kartının HMI ile haberleşmesi için Modbus kullanılmıştır. Sistemdeki tüm kartlarda ADM Modbus entegresi bulunmaktadır.

HMI, mini PC (Orange Pi PC Plus), Ekran (7" dokunmatik TFT) ve çevre bileşenlerini oluşturacak bir elektronik karttan (PCB) oluşmaktadır. Mini PC'ye Linux işletim sistemi kurulmuştur. HMI'ın IoT sisteminin uygulanabilmesi ve buluta bağlanabilmesi için internet ihtiyacı vardır. Kullanılan mini PC Orange Pi PC Plus'ın kendi içinde Wi-Fi modülü mevcuttur.

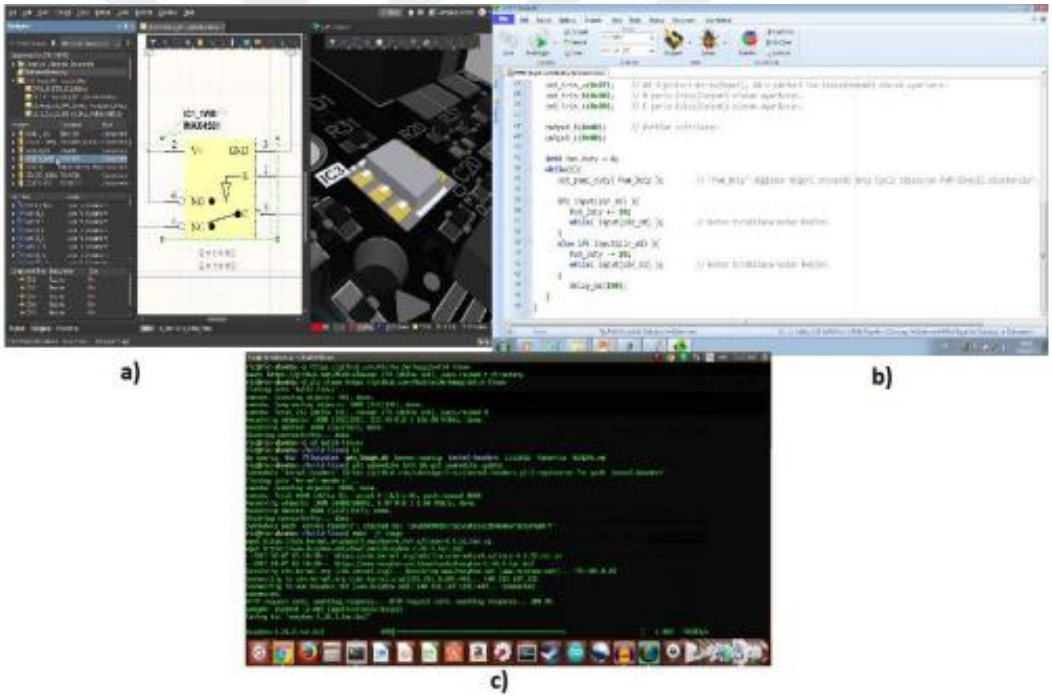
Linux işletim sistemiyle birlikte TFT ekran sürülmesi, çevre bileşenlerin çalışması ve internet bağlantı yazılımları yapılmıştır. Mikrodenetleyici C++ ile yazılmıştır. GUI tasarımı için widgetler Photoshop'ta oluşturulmuştur. Daha sonra Qt-Qml programlama dili ile mikrodenetleyiciye gömülmüştür.

IoT uygulaması için HMI'ın bulut ile bağlantısı MQTT protokolüyle sağlanmıştır. Bu protokolü çalıştırmak için DigitalOcean'dan server satın alınmıştır.

Buluttaki verileri işleyip yönetebilen web yazılımı yapılmıştır. Web yazılımında Front-End tarafında HTML, CSS ve Java Script kullanılmıştır. Back-End tarafında node.js ve veritabanı için ise MySQL kullanılmıştır.

Burada amaç, otomasyon için tüm gereksinimlerin oluşturulması değil sistemin kavranması adına bir kontrol panosunun oluşturulmasıdır. Böylece IoT otomasyon için en temel yapı taşı olan mutfak kontrol sisteminin sistem parçalarının özel olarak oluşturulup mutfağa özgü ve maliyeti daha uygun şekilde sistemin kurulması sağlanabilecektir. Geliştirilen sistemin, bir deney düzeneği veya eğitim seti olacak nitelikte yapılması kurgulanmıştır. Bu sistem, endüstriyel mutfak otomasyonu IoT uygulamasını simüle edecek şekilde oluşturulmuştur. Aynı zamanda üreticiler ile müşteriler arasında veri bağlantısını da simüle edecektir.

Şekil 3.2'de Altium Designer devre tasarımı görünümü, MplabX Microchip işlemci programlama arayüzü ve linux sistem komut satırı ekranı görülmektedir.

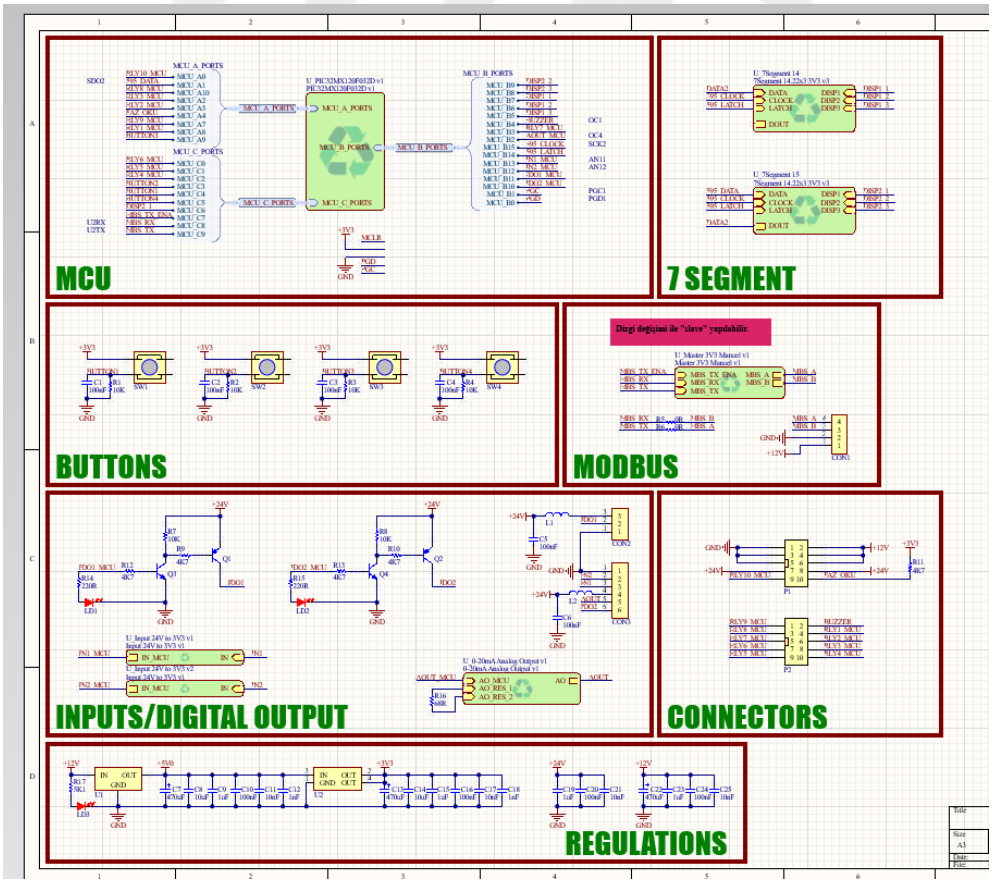


Şekil 3.2. a) Altium Designer devre tasarımı görünümü b) İşlemci programlama arayüzü
c) Linux sistem komut satırı ekranı

3.2. Endüstriyel Mutfak Makinası Elektronik Kartının Tasarlanması

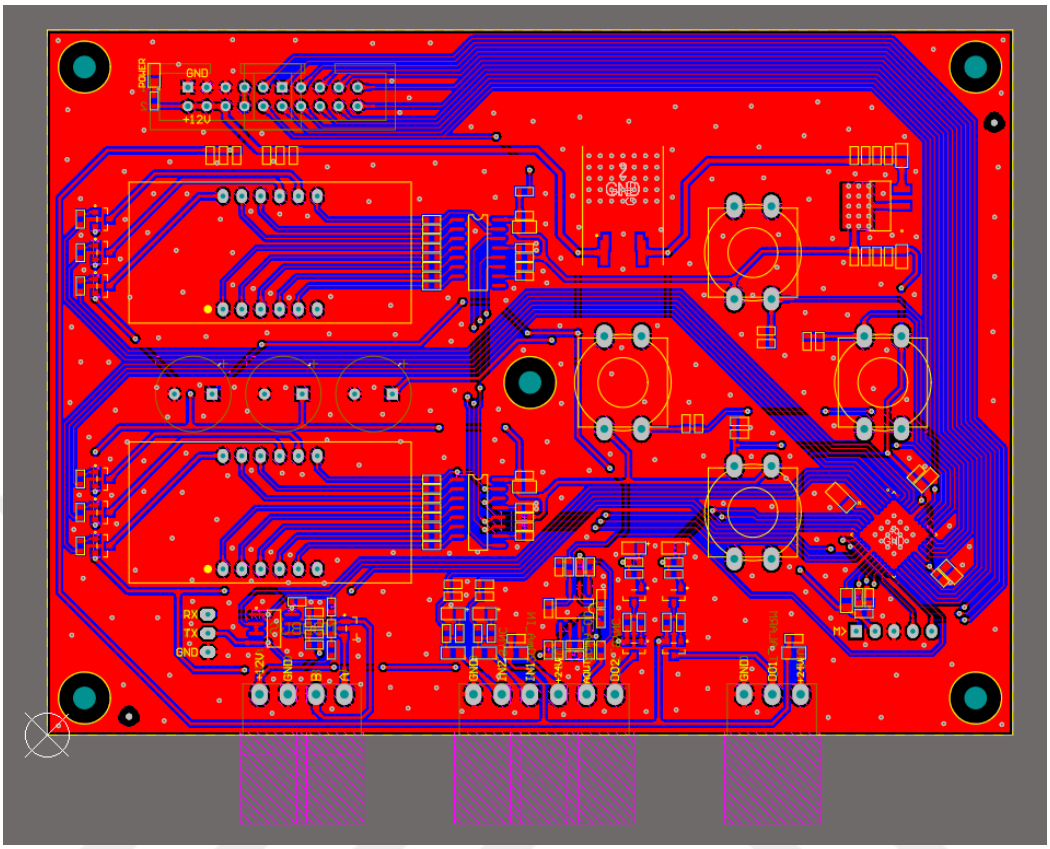
3.2.1. Devrelerin oluşturulması, PCB tasarımı, komponent seçimi

Sistem gereksinimlerine göre Altium Designer üzerinde devre tasarımı oluşturmak için ilk önce devre planlanmıştır. Devre tasarımı için gerekli komponentlerin kütüphanesi oluşturulmuştur. Proje içerisine açılan şematik bölümüne uygun devreler çizilmiştir. Devrede 24VDC besleme girişi olacağından dolayı uygun gerilim için bir voltaj regülatör devresi tasarlanmıştır. Modbus entegresi, ana işlemci ve diğer komponent beslemeleri için uygun olan 3.3V regülesi tamamlanmıştır. Analog girişler, dijital girişler ve çıkışlar için devre blokları ve Modbus haberleşmesi için bir Modbus modülü ile devre bloğu tasarımı yapılmıştır. Gerekli konnektörlerle bağlantılar yapılmıştır. Dataları göstermek ve ayarlamak için 7 segment ekran ve butonlar konmuştur. Yapılan şematik tasarımı Şekil 3.3'de detaylı olarak görülmektedir. Komponentlerin veri dosyaları detaylı incelenerek tasarımda kullanılmıştır.



Şekil 3.3. Kurulumu gerçekleştirilen sistemin blok diyagramı

Şema bittikten sonra PCB baskı yolları (Layout) çizilerek Şekil 3.4'de görüldüğü gibi devre tasarlamıştır.



Şekil 3.4. Kurulumu gerçekleştirilen sistemin PCB tasarımı

3.2.2. Modbus haberleşme

Modbus ilk olarak 1979 yılında Modicon firması tarafından ortaya çıkartılmıştır. Modbus, 1 adet master (ana cihaz) bulunan sistemde 247 adet slave (bağımlı) cihaz okuyabilecek şekilde düzenlenebilmektedir (Deveci, 2022).

Blok yapısı Şekil 3.5'de görülen Modbus, haberleşme protokolleri ve fiziksel katman olarak ikiye ayrılmaktadır. Günümüzde haberleşme katmanı olarak en çok kullanılan Modbus protokolleri; ASCII, RTU ve TCP/IP'dir. Fiziksel katman için UART temelli RS232/RS485, USB/CAN gibi değişik birimler ve TCP/IP için ethernet gösterilebilir (Deveci, 2022).

Modbus RTU ile haberleşme ASCII ile haberleşmeye göre hızlıdır. Örneğin 34567 sayısı seri hat üzerinden ASCII metodu ile gönderilirse "3","4","5","6","7" şeklinde 5 ayrı byte ile gönderim yaparken, RTU'da bu bilgi 2x8 bit olacak şekilde en soldaki basamağa en yüksek değerlikli bit MSB ve en sağdaki basamağa en düşük

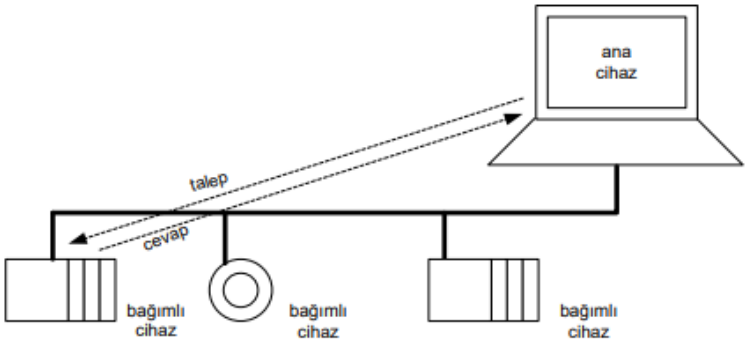
değerlikli bit LSB olarak ikiye bölünecektir ve 2 byte gönderim yapacaktır. Bu nedenle günümüzde en çok kullanılan Modbus protokolleri seri hat üzerinden gerçekleşen Modbus RTU ve Ethernet üzerinden gerçekleşen Modbus RTU protokolleridir. Kullanım kolaylığı ve güvenilirlik nedeniyle yaygın olarak kullanılır (İdas, 2022).

3.2.2.1. Modbus RTU protokol yapısı

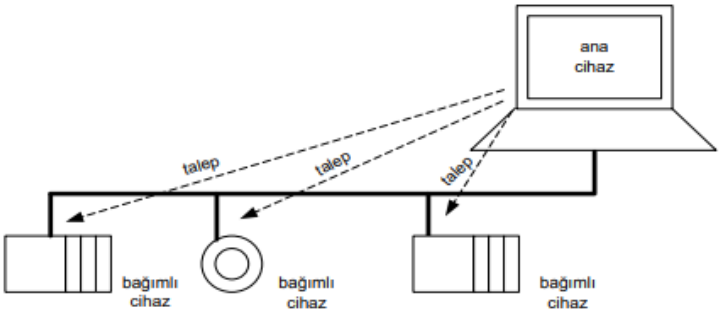
Modbus RTU protokolü ana birim ile bağımlı birim arasında veri haberleşmesini yapan bir protokoldür (Akkaya, 2015). Ağ üzerinde bir ana cihaz ve cihaza bağlı çalışan bağımlı cihazlar bulunur. Bağımlı cihazlar ana cihazdan bir talep gelmediği sürece mesaj iletmezler ve birbirleri ile haberleşmezler. Ana cihaz, bağımlı cihaza talep gönderir ve cihaz da gönderilen talebe karşılık bir cevap gönderir. Ağ üzerinde aynı anda sadece bir bağımlı cihaz haberleşir (Akkaya, 2015).

Modbus seri haberleşme ağı üzerinde bir adet ana cihaz ve maksimum 247 adet bağımlı cihaz bulunur. Ana cihaz, bağımlı cihazlara tek yönlü ve yayın modu olmak üzere iki türlü mesaj gönderebilir. Tek yönlü modda ana cihaz tek bir cihaz ile haberleşir. Bağımlı cihaz talebi alır ve işledikten sonra ana cihaza geri mesaj gönderir. Her bağımlı cihazın tek bir adresi vardır (1'den 247'ye kadar). Yayın modunda ana cihaz ağ üzerindeki bağımlı cihazlara talep gönderir ve hiçbir bağımlı cihaz geri dönüş yapmaz. Bu haberleşme modunda bağımlı cihazların mesajı alıp almadığı kesin olarak bilinemediğinden çok tercih edilmez (Akkaya, 2015).

Şekil 3.5 ve 3.6'da verilen şekillerde her iki haberleşme modu ifade edilmiştir.



Şekil 3.5. Tek yönlü iletim modu

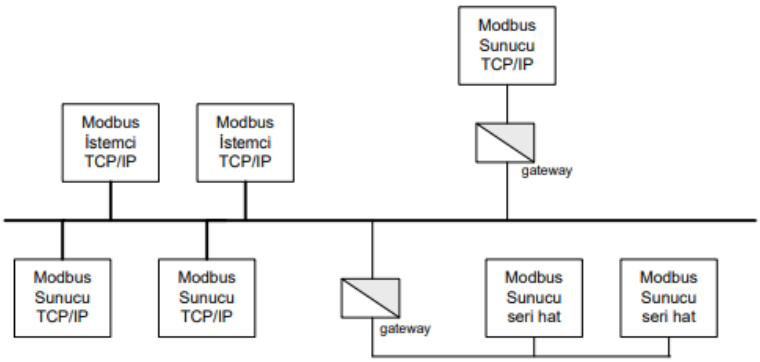


Şekil 3.6. Yayın modu

3.2.2.2. Modbus TCP protokol yapısı

Ethernet üzerinden TCP arabirimi ile Modbus RTU protokolünün gerçekleştirilmesi TCP protokolünün temelidir. Otomasyon cihazlarının denetim ve kontrolü amacıyla tasarlanan basit kullanıma sahip olan bir protokoldür.

TCP'nin temel görevi bütün veri paketlerinin doğru bir şekilde alınmasını sağlamaktır. IP ise, mesajın doğru bir şekilde adreslenmesini ve yönlendirilmesini sağlar. Özetle Modbus TCP, uyumlu cihazlar arasında Modbus mesaj verisini taşımak için TCP/IP ve Ethernet kullanır. Yani Modbus TCP protokolü, fiziksel bir ağı (Ethernet), ağ standartı (TCP/IP) ve Modbus protokol veri birimini birleştirir. Şekil 3.7'de Modbus TCP/IP yapısı görülmektedir. Eski endüstriyel cihazlar genellikle RTU protokolünü kullandıkları için farklı bir cihazla beraber çalışması durumunda bağlantı problemleri yaşar. Modbus TCP protokolü bu duruma çözüm olarak geliştirilmiştir (Deveci, 2022). Modbus TCP'nin Ethernet fiziksel katmanını kullanmasından dolayı bant genişliği rahat bir şekilde kullanılabilmektedir. Bu durumda Modbus TCP, RTU'ya göre veri paketlerini daha doğru alabilmektedir.



Şekil 3.7. Modbus TCP/IP haberleşme mimarisi

3.2.2.3. Modbus yazılımının kodlanması

Tez çalışmasında Modbus RTU kullanımı tercih edilmiştir. Modbus RTU frame yapısı Şekil 3.8'de görülmektedir.

Slave Address	Function Code	Data	CRC
1 byte	1 byte	0 up to 252 byte(s)	2 bytes CRC Low, CRC Hi

Şekil 3.8. Modbus RTU frame yapısı

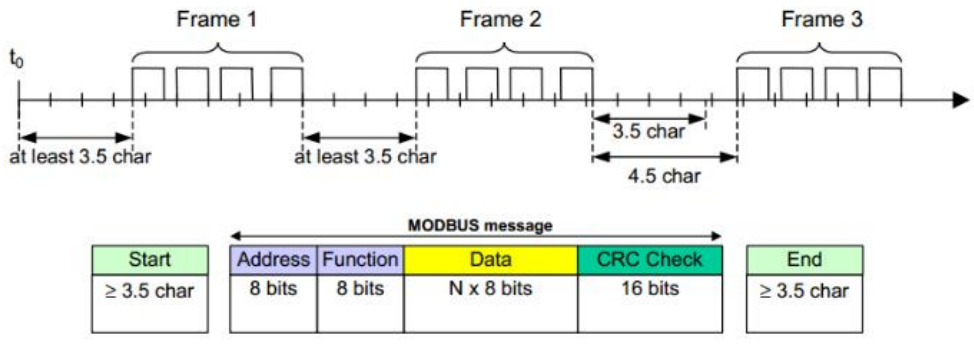
İlk byte'ta iletişim kurulacak cihazın adresi belirlenmelidir. Slave cihazın sayısı en fazla 247 adet olabileceğinden bu bölüme 0-247 arası bir sayı yazılmalıdır. İkinci byte ise Modbus için standartlaşmış fonksiyon kodlarını ihtiva etmelidir. Bu fonksiyonlar Şekil 3.9'daki gibi tanımlanmıştır.

Code	Modbus Function
1	Read Coil Status
2	Read Input Status
3	Read Holding Registers
4	Read Input Registers
5	Write Single Coil
6	Write Single Register
15	Write Multiple Coils
16	Write Multiple Registers

Şekil 3.9. Modbus standart fonksiyon kodları

Günümüzde en çok kullanılan ve tercih edilen Modbus RTU fonksiyonları 3, 6 ve 16'dır (Deveci, 2022).

Bundan sonraki bölümde fonksiyon tanımlamalarına göre datalar düzenlenerek, slave adresi ve fonksiyon numarasının CRC16'da son iki byte'a yazılarak Şekil 3.10'daki gibi gönderim ve alım işlemleri sağlanır. CRC giden ve gelen dataların doğruluğunun kontrolü içindir.



Şekil 3.10. Modbus RTU frame'ler arası yapı

Tez çalışmasında sırasıyla 3. ve 16. fonksiyonlar kullanılmıştır. Slave adresi 17 (0x11) olan endüstriyel mutfak makinesinin 107. (06xB) ve 108. (0x6C) register değerleri okunmuştur (11 03 00 6B 00 02 crc_1 crc_h).

Gelen cevapta ise 11 03 04 CC CD 42 8D crc_1 crc_h şeklinde cihaz öncelikle kendi slave numarası ve kendisine gelen fonksiyonun tanımını gönderdikten sonra 2 x 2 byte şeklindeki veri ile bunların CRC16 değerini master'a göndermektedir. Tüm bu değerler doğru ise master cevabı kabul etmektedir.

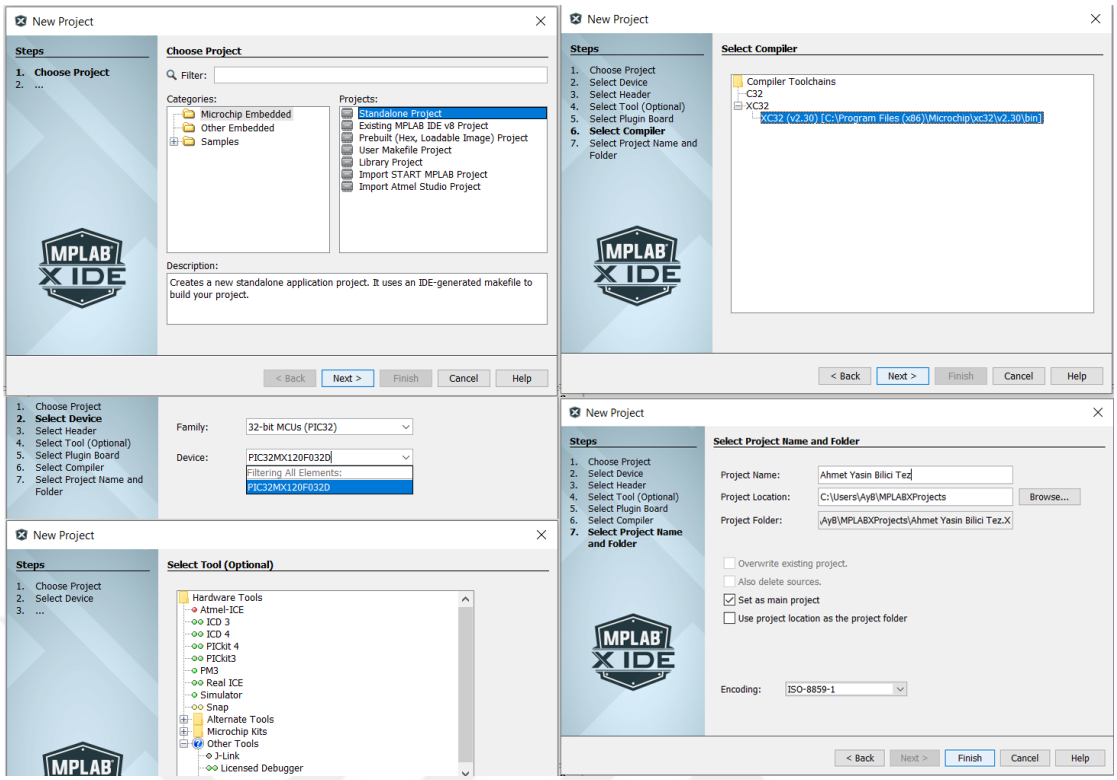
Bu işlemler için yazılmış kod Şekil 3.11'de görülmektedir. Modbus yazılımının tamamı EK-2'de verilmiştir.

```
1  void CRC16(const unsigned char Data, unsigned int* CRC)
2  {
3      unsigned int i;
4
5      *CRC = *CRC ^(unsigned int) Data;
6      for (i = 8; i > 0; i--)
7      {
8          if (*CRC & 0x0001)
9              *CRC = (*CRC >> 1) ^ 0xA001;
10         else
11             *CRC >>= 1;
12     }
13 }
```

Şekil 3.11. Gönderilen verilerin doğruluğunun kontrol edilmesi

3.2.3. Tasarlanan kontrol kartı sisteminin C ile yazılımı

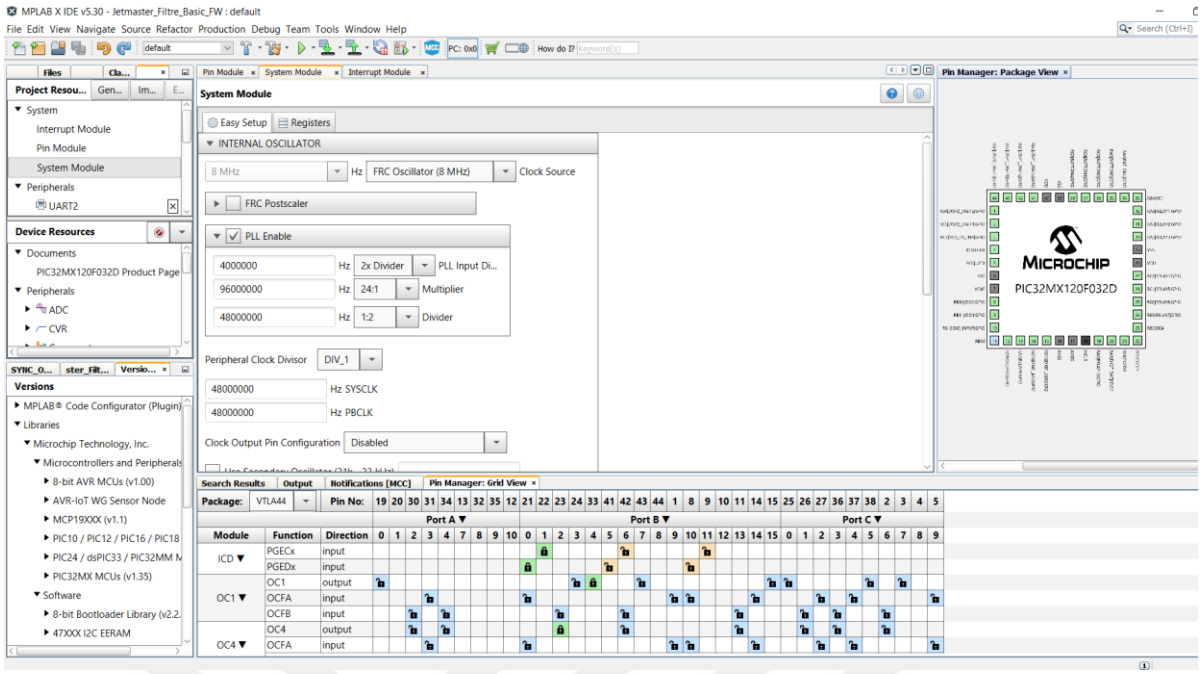
Yazılımda 7 segment display sürme, dijital giriş ve çıkışları ayarlama vb. kontroller için hem C programlama bilgisine, hem de makineler arası haberleşme için Modbus bilgisine ihtiyaç vardır. Gerekli olan algoritmanın kodlanabilmesi ve Modbus haberleşme yazılımının derlenebilmesi adına tasarımda kullanılan PIC işlemci için yazılım derleme platformu olarak MplabX programı tercih edilmiştir. Şekil 3.12'de görüldüğü üzere MplabX derleyicisinde proje başlatılır. File>New Project menülerini sırasıyla tıklayarak proje kaydetme sayfası açılır.



Şekil 3.12. MplabX IDE proje oluşturulması

Açılan pencerede işlemci olarak Microchip ve standalone (bağımsız) proje seçimi yapılır. Daha sonra Microchip işlemcisinin türü seçilir. Tez çalışmasının bu bölümünde PIC32MX120F032D işlemcisi kullanılmıştır. Devamında ise işlemcinin programlanacağı tool seçimi yapılmıştır. Projemizde Pickit3 tool kullanılmıştır. Bir sonraki aşamada compiler seçimi vardır. Hangi versiyon kullanılacağı seçilebilmektedir. Son olarak projenin ismi ve kaydedileceği dosya belirlenerek proje oluşturulur.

MplabX popüler bir IDE olmakla birlikte MCC denilen bir kod oluşturucu eklenti ile işlemci donanımlarını basitçe çalıştırmada kolaylık sağlamıştır (Ward, 2022). Projenin oluşturulmasının ardından Şekil 3.13'de görüldüğü üzere proje MCC eklentisi içerisinden hangi pinlerin ne için kullanılacağı, kristal seçimi, timer ayarları, frekans ayarları, işlemcide kullanılacak donanımların seçimi vb. gerçekleştirilir. Tez çalışmasında kullanılacak bu işlemcinin proje zamanlayıcısı için Timer 1, Buzzer için Timer 2, PWM sinyali için Timer 3 donanımları kullanılmıştır. Ayrıca Internal hafıza için NVM, PWM oluşumu için OC4 ve OC1, 7segment ekranları sürmek için ise SPI donanımları kullanılmıştır. Burada PWM sinyalleri analog çıkışlar için gereklidir. Son olarak Modbus haberleşme için UART2 donanımı kullanılmıştır.



Şekil 3.13. MCC eklenti arayüzü

Sistemde kullanılacak bu kartlar endüstriyel mutfak makinesi kartlarıdır. Kendi içinde sensörlerden bilgileri alıp işleyecek, diğer makineler ve HMI ile haberleşme yaparak iletimi sağlayacaktır. Aynı şekilde kendine gelen bilgiler ile de çıkışları kontrol edecektir.

3.3. HMI

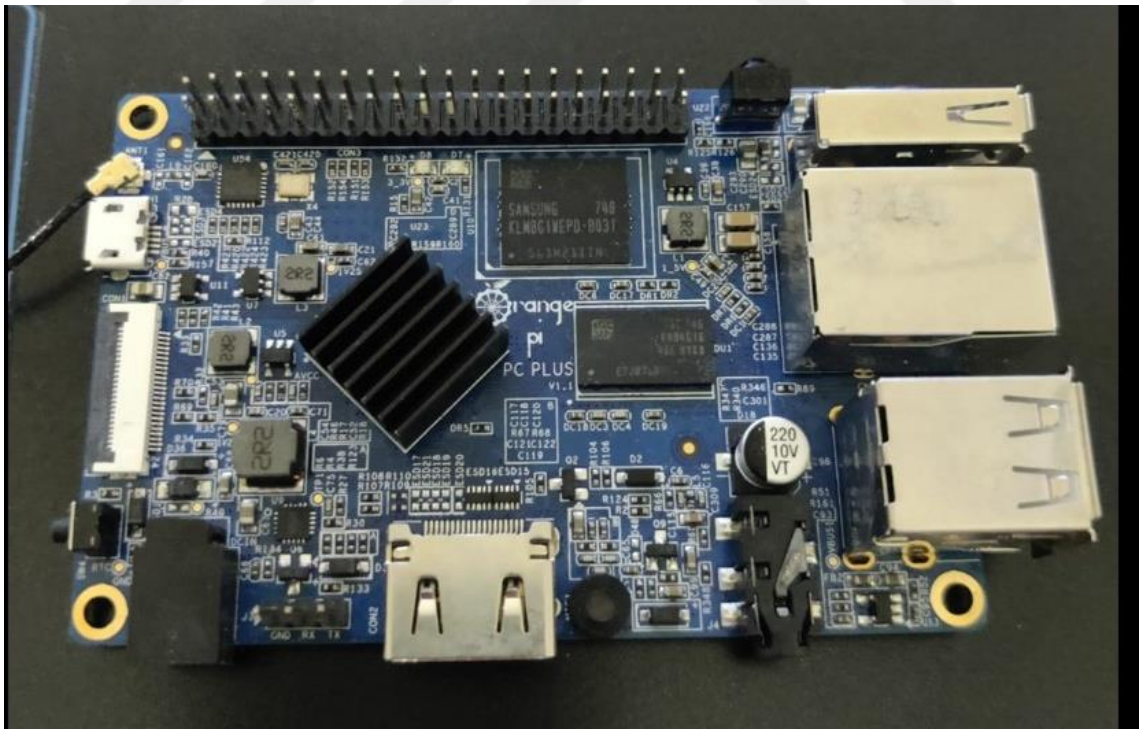
HMI panel, üzerinde işletim sistemi çalıştıran bir çeşit bilgisayardır. İşletim sistemlerinin erişimi kullanıcıya kapalıdır ve özel yazılım yardımı ile programlanırlar. Farklı işletim sistemleriyle uyumlu kullanılabilmektedirler. Temel görevleri kullanıcının verdiği komutları kontrol kartına iletmek ve kontrol kartından aldığı proses verilerini ekranında görüntülemektir (Dalmış ve ark., 2018).

HMI'lar dokunmatik özellikli bir ekrana sahip olduğu için klavye ve fareye gerek duyulmadan kolay bir kullanım imkanı sunmaktadır. HMI'ların kullanıcının işlerini kolaylaştırması, zamandan tasarruf sağlaması, hafıza yükünü fazlasıyla azaltması ve tutarlı olması gerekmektedir. Kullanıcıların dijital sistemleri kullanması sayesinde yapılmak istenen işler manuel sistemlere kıyasla daha etkili ve verimli yapılabilmektedir (Eryılmaz, 2015).

HMI'lar sayesinde arıza, ikaz, alarm, bildirim gibi durumlarda kullanıcı hızlı bir şekilde görsel uyarıyla karşılaştığı için hem mutfak güvenliği artacaktır hem de önlem alabilmek adına hızlı müdahalede bulunulabilecektir.

3.3.1. Orange Pi PC Plus

Tek kartlı bilgisayar olan Orange Pi PC Plus, endüstriyel kullanım için tasarlanmış düşük maliyetli ve genellikle Linux işletim sistemi desteği ile kullanılan IoT projelerinde sıkça tercih edilen mini bilgisayarların adıdır (Isikdag, 2015). Orange Pi, bir bilgisayar olmasıyla birlikte üzerinde bulunan 40 adet pin GPIO bağlantısı sayesinde fiziksel donanımların bağlantısı ile kontrol için kullanılabilecek bir tümleşik yapıdır. Herhangi bir bilgisayarda yapılabilecek hemen hemen her şeyi Orange Pi bilgisayarı ile yapmak mümkündür. Orange Pi'nin bugüne kadar çıkartılmış birçok türü bulunmaktadır (Isikdag, 2015). Bu tez çalışmasında Şekil 3.14'de görülen Orange Pi PC Plus (Wifi destekli) modeli kullanılmıştır. Tercih edilme sebebi olarak ürünün diğer tek kartlı bilgisayarlara göre fiyatının ucuz olması, kablosuz internet bağlantısının olması, 1 GB RAM ve HDMI girişinin olması ön plana çıkmaktadır.



Şekil 3.14. Orange Pi PC Plus

Tez çalışmasında kullanılan Orange Pi PC Plus ile kablosuz ağa bağlanabilmesi sayesinde kolaylıkla hedeflenen IoT uygulaması (Arıza tespit, programlama, veri toplama) gerçekleştirilmiştir. Orange Pi üzerinde uygulama geliştirmek için bazı yazılım dillerine ihtiyaç vardır. Bu tez çalışmasında C++, QT ve QML yazılım dillerinden faydalanılmıştır.

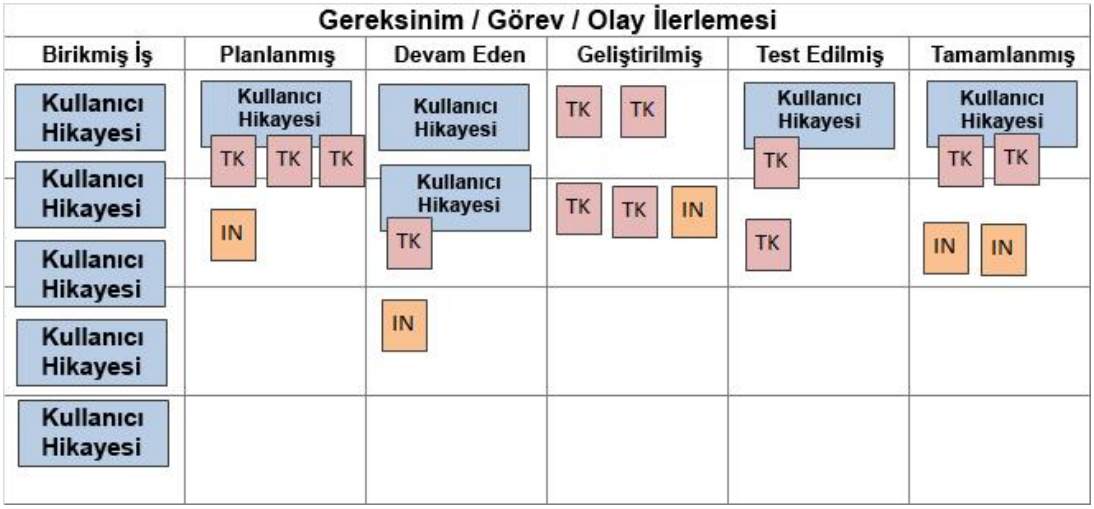
3.3.2. Linux işletim sistemi ve gömülü yazılım

Bilgisayar programlarının çalıştırılmasını ve istenilen temel işlevlerin yerine getirilmesini işletim sistemleri sağlamaktadır. 1991 yılında Linus Torvalds tarafından geliştirilen Linux, işletim sistemlerinin çekirdeğidir. Aynı zamanda genel bir adıdır. Günümüzde ise açık kaynak kod felsefesi olan yapıda geliştirilmesine devam edilmektedir. Ubuntu, Debian, Fedora, Pardus, KNOPPIX ve OpenSuse başlıca sayılabilecek Linux tabanlı işletim sistemleridir (Kofler, 2010). Gömülü sistem ise içinde bulunmuş olduğu sistemin yönetim görevini üstlenerek akıllı hale getiren; donanım ve yazılım alt birimleri ile etkileşimli çalışan entegre sistemin tümüdür.

Tez çalışmasında IoT uygulaması için gerekli olan tek kartlı bilgisayara Linux işletim sistemi kurulmuştur. IoT projelerinde Linux tabanlı sistemler gerek düşük enerji tüketimi gerekse kullanım rahatlığından oldukça yaygındır (Krylovskiy, 2015).

Yazılım geliştirme sırasında genellikle kanban tekniği uygulanmıştır. Kanban, mevcut kapasite ile iş taleplerini dengelemek için kullanılan bir yöntemdir. İş öğeleri, katılımcılara görev tanımından müşterinin sunumuna kadar ilerleme ve süreç görüntüsü verecek şekilde görselleştirilir. Kanban, bir üretim sisteminin bütün olarak çalışmasını ve iyileştirmeyi teşvik etmek için doğru bir araçtır (Ahmad ve ark., 2013).

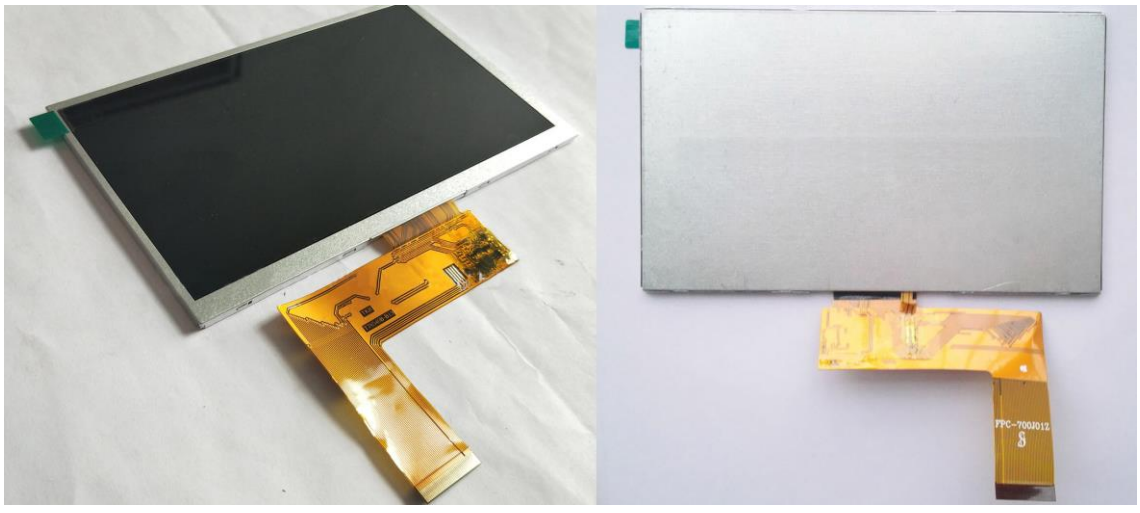
Kanban, yazılım geliştirmede neyin, ne zaman ve ne kadar üretileceği konusunda karar vermeye yardımcı olan görsel bir süreç yönetim sistemi sunmaktadır. Şekil 3.15'de Kanban süreci görülmektedir. Kanban süreci, yazılım geliştirme ve IT kaynaklı olmasına rağmen, somut herhangi bir profesyonel hizmet için de uygulanabilir (Ahmad ve ark., 2013).



Şekil 3.15. Kanban Yazılım Geliştirme

3.3.3. Ekran

Tez çalışmasında HMI cihazında Şekil 3.16'da görülen 7 inç, dokunmatik, 1024 x 600 piksel IPS çözünürlüklü ekran tercih edilmiştir. HDMI dönüştürücü kartlar sayesinde projede kullanışlı bir yapı ortaya koymaktadır.

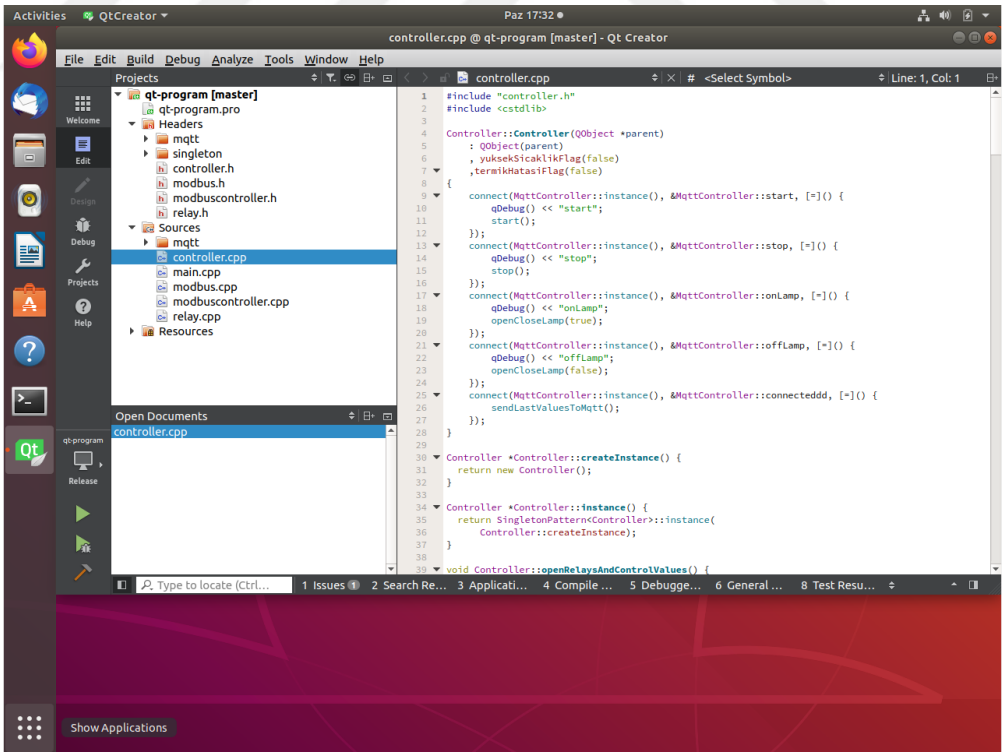


Şekil 3.16. Tez çalışmasında kullanılan 7" IPS ekran

3.3.4. C++, QT ve QML ile GUI tasarımı

Orange Pi PC Plus üzerinde gerekli kütüphanelerin kurulmasıyla birlikte HMI kontrol sisteminin görsel ekran tasarımları ve yazılımları C++, QT ve QML yardımıyla yapılmaktadır. C++ programlama dili ile diğer Endüstriyel Mutfak Makineleriyle haberleşme, çevre birimlerinin kontrol edilmesi vb. işlemler yapılmaktadır. Ayrıca GUI tasarım için bir köprü kurularak veriler QT platformuna aktarılmaktadır. QT platformunda işlenen veriler QML tarafından görsel olarak ekrana basılmaktadır.

Qt platformu, tüm cihazlarda sorunsuz bir kullanıcı deneyimi sunarken yazılımları hızlı ve uygun maliyetli bir şekilde tasarlamanıza, geliştirmenize, dağıtmanıza ve bakımını yapmanıza olanak tanıyan bir platformlar arası SDK ve teknoloji stratejisidir. Qt çerçevesinin ana odak noktalarından biri, güçlü multimedya uygulamalarının Qt Modelleme Dili (QML) aracılığıyla basit bir şekilde geliştirilmesidir. Uygulamaları/kütüphaneleri yalnızca bir kez geliştirmeye ve aynı kaynak kodunu birden çok platforma derlemeye izin verir. Çalışma ekranı Şekil 3.17'de görülen Qt, dünya çapında birçok şirket tarafından yaygın olarak kullanılmaktadır (Neto ve ark., 2017).



Şekil 3.17. Qt Creator Uygulaması

3.3.5. Linux Kernel ile 7” ekran üzerinde QT ve C++ tabanlı HMI oluşturma

Mini PC tarafında ilk yapılacak işlem bir işletim sistemi kurulumu olacaktır. Proje kapsamında kullanılacak Mini PC, Orange Pi PC Plus olduğundan Linux temelli Debian’a dayalı Armbian işletim sistemi kurulumu yapılmıştır. Kurulum için öncelikle minimum16GB kapasiteli bir hafıza kartına ihtiyaç vardır. Hafıza kartının yüksek boyutlu seçilmesi proje dosyalarında oluşacak büyük depolarda sıkıntı yaratmaması açısından önemlidir. HMI panel yazılımının tamamı EK-3’de verilmiştir.

3.3.5.1 Linux işletim sistemi kurulumu

Linux bir işletim sistemi, kernel(çekirdek)’i ifade etmektedir. Bu kernel üzerinde bir cihazın temel ihtiyaçları olan sürücüler bulunmaktadır. Bu sürücülere örnek olarak I2C, SPI, HDMI, TCP vb. verilebilir. Bu tezde Linux kernel kullanılmıştır. Ancak kerneller genel kullanıma uygun olarak tasarlanır ve popüler işlemci tiplerine uygun olarak kodlanırlar. Doğrudan cihazlara uygun olarak tasarlanmazlar. Örneğin tezde kullanılan Orange Pi cihazı için Linux kernelinin resmi deposunda bir config dosyası bulunmamaktadır. Bu sebeple config dosyaları kaynak kodunun konfigürasyonunu sağlar ve isteğe göre kaynak dosyasının hangi bölümlerinin derleneceğini belirtir. Projede Goodix driver’ına ihtiyaç vardır. Bu driver dokunmatik ekran için gereklidir. Driver eklemeye örnek olması açısından config dosyası oluşturma adımları aşağıya yazılmıştır.

- Güncel kernel indirilir (İndirme linki EK-1 A’da verilmiştir),
- Tar dosyası (Linux sıkıştırma dosyası) çıkarılır (tar xvf linux-6.0.7.tar.xz),
- Güncel kullanılan config dosyası bu klasör içerisine kopyalanır (cp -v /boot/config-$(uname -r) .config).

Şekil 3.18’de config dosyası kopyalama işleminin kodları görülmektedir.

```
marko@pnap:~$ cd linux-6.0.7/
marko@pnap:~/linux-6.0.7$ cp -v /boot/config-$(uname -r) .config
'/boot/config-5.15.0-52-generic' -> '.config'
marko@pnap:~/linux-6.0.7$
```

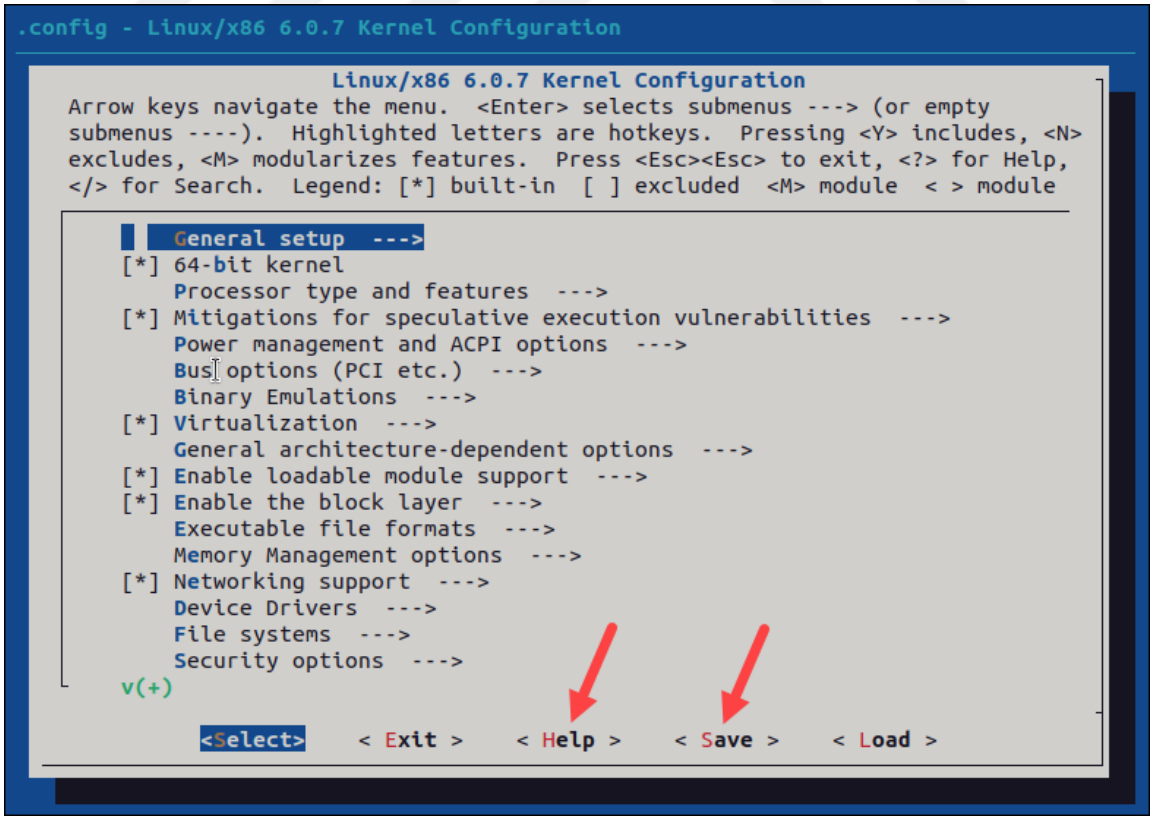
Şekil 3.18. Default config dosyasının kopyalanması

- Kernel konfigürasyonunu sağlamak için gerekli kütüphaneler kurulur (sudo apt-get install git fakeroot build-essential ncurses-dev xz-utils libssl-dev bc flex libelf-dev bison),
- Menuconfig ile kernel tarafından geliştirilen konfigürasyon menüsü açılır (Make menuconfig).

Şekil 3.19'da config dosyası çıktı örneği, Şekil 3.20'de config menüsü görülmektedir.

```
marko@pnap:~/linux-6.0.7$ make menuconfig
  HOSTCC  scripts/basic/fixdep
  UPD     scripts/kconfig/mconf-cfg
  HOSTCC  scripts/kconfig/mconf.o
  HOSTCC  scripts/kconfig/lxdialog/checklist.o
  HOSTCC  scripts/kconfig/lxdialog/inputbox.o
  HOSTCC  scripts/kconfig/lxdialog/menubox.o
  HOSTCC  scripts/kconfig/lxdialog/textbox.o
  HOSTCC  scripts/kconfig/lxdialog/util.o
  HOSTCC  scripts/kconfig/lxdialog/yesno.o
  HOSTCC  scripts/kconfig/confdata.o
  HOSTCC  scripts/kconfig/expr.o
  LEX     scripts/kconfig/lexer.lex.c
  YACC    scripts/kconfig/parser.tab.[ch]
  HOSTCC  scripts/kconfig/lexer.lex.o
  HOSTCC  scripts/kconfig/menu.o
```

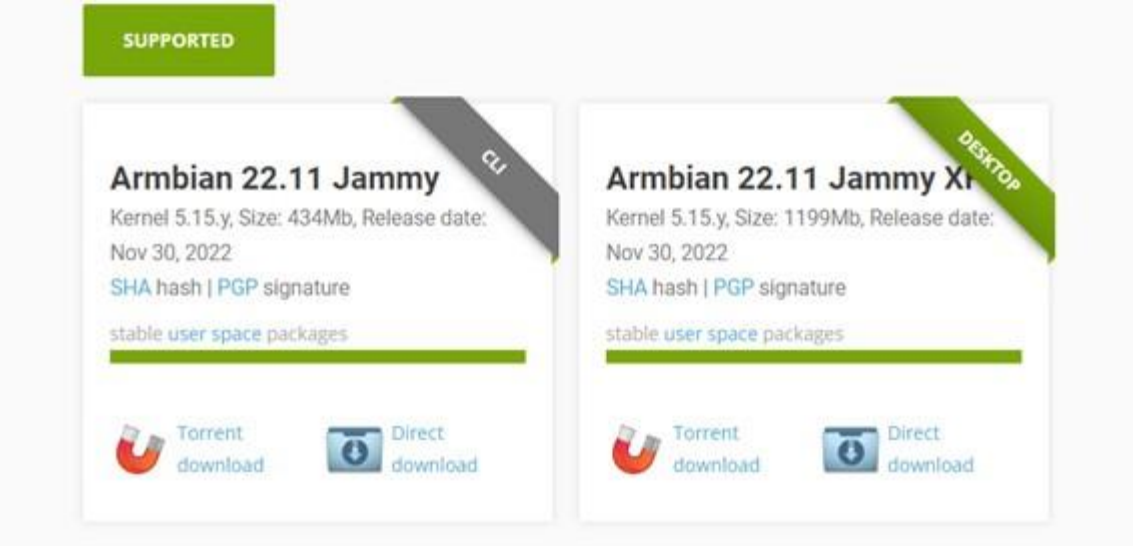
Şekil 3.19. Linux config çıktı örneği



Şekil 3.20. Linux config menüsü

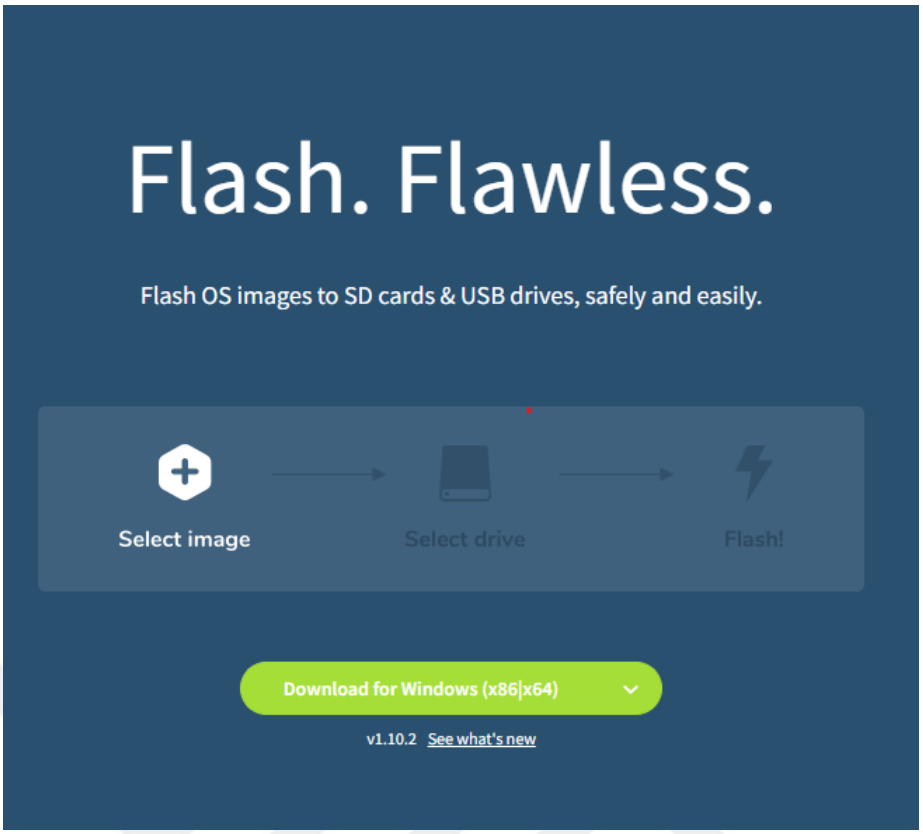
- Gerekli ayarlar yapılarak yeni config dosyası .config uzantılı olarak kaydedilir. Bu config dosyası armbian build sistemi ile kullanılabilir.

Orange Pi için uygun kernel, çeşitli derleme aşamalarından geçilerek oluşturulabilir ancak basitlik adına hazır olan kernellerden kullanılmıştır. Bunun için armbian websitesine giderek Orange Pi PC+ için uygun işletim sistemi indirilmiştir. Türleri Şekil 3.21'de verilen Armbian işletim sistemi, orijinal Linux kaynak kodunu alarak çeşitli mimarilere uygun biçimde yeniden yazmaktadır. Çoğunlukla arm mimarisine odaklanılmıştır.



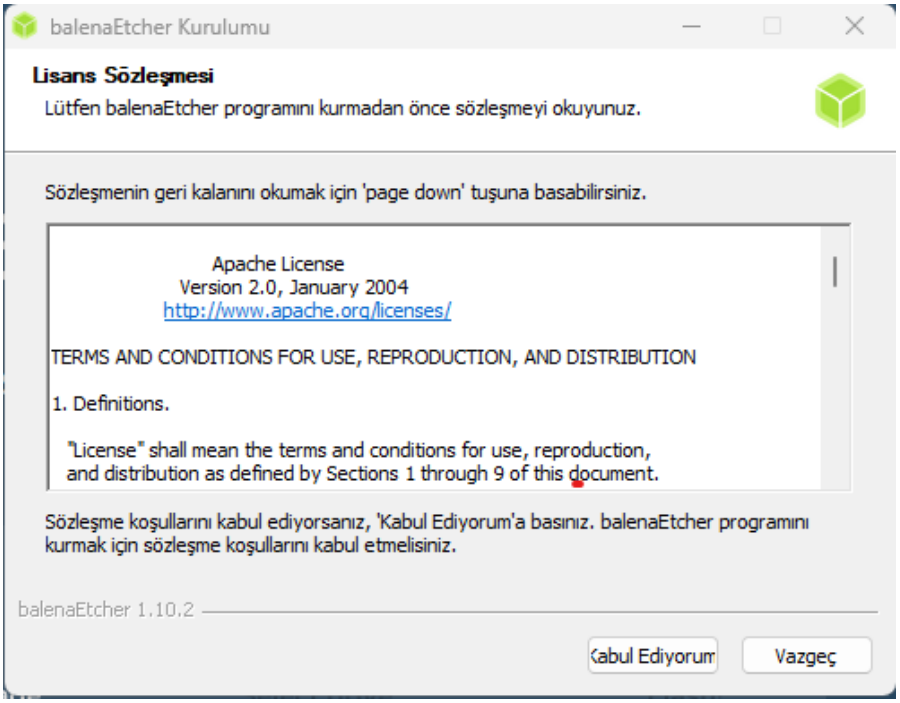
Şekil 3.21. Armbian işletim sistemi çeşitleri ve menüsü

- İndirilen işletim sistemi – proje durumunda server işletim sistemi – etcher yardımı ile sd karta yazılır (İşletim sistemi indirme linki EK-1 B'de verilmiştir). Etcher kurulum adımları Şekil 3.22-3.24'de resimli olarak anlatılmıştır.
- Etcher websitesinden dosya indirilir (İndirme Linki EK-1 C'de verilmiştir).



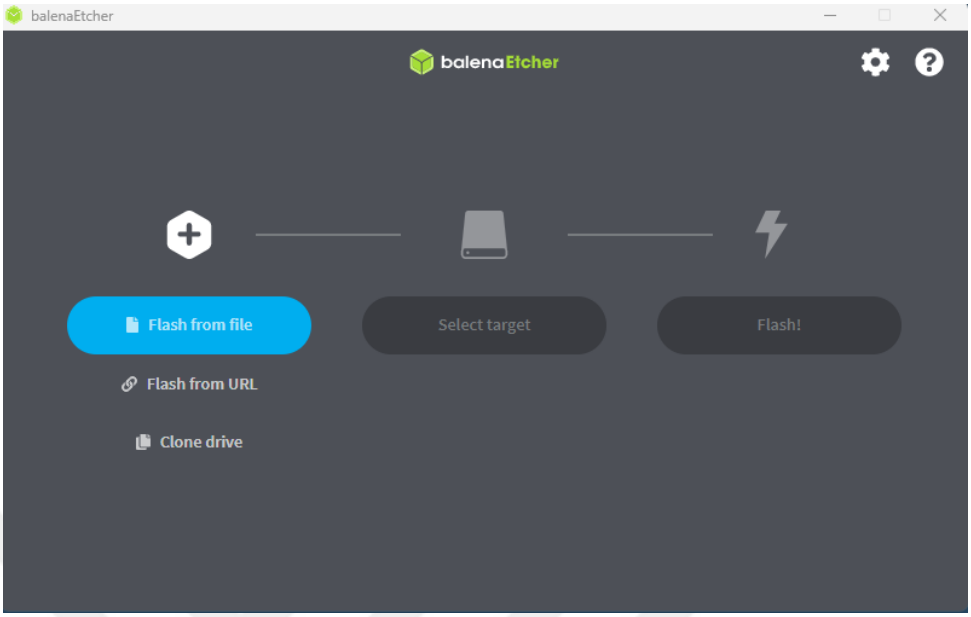
Şekil 3.22. Etcher indirme linki gösterimi

- İndirilen dosyanın kurulumu sağlanır.



Şekil 3.23. Etcher kurulum menüsü gösterimi

- Lisans kabul edilerek kurulum başlatılır ve kurulum sonrasında karşımıza etcher menüsü çıkar.



Şekil 3.24. Etcher menüsü

- Flash from file denilerek. Armbian sitesinden indirilen image dosyası seçilir. Bilgisayara sd kart takıldığında select target denilerek sd kart seçilmiştir.
- Flash! Butonuna tıklanarak yazma işlemi başlatılır. Etcher kurulumu tamamladıktan sonra verify yani denetleme işlemi de yapılmaktadır. Bu işlem oluşturulan boot sd kartının bütünlüğünü kontrol eder.
- SD kart kurulum işlemi tamamlanmıştır. Bu adımla birlikte orange pi için boot edilebilir bir armbian işletim sistemi hazır edilmiş olur.

3.3.5.2 Qt ve diğer gerekli kütüphanelerin kurulumu

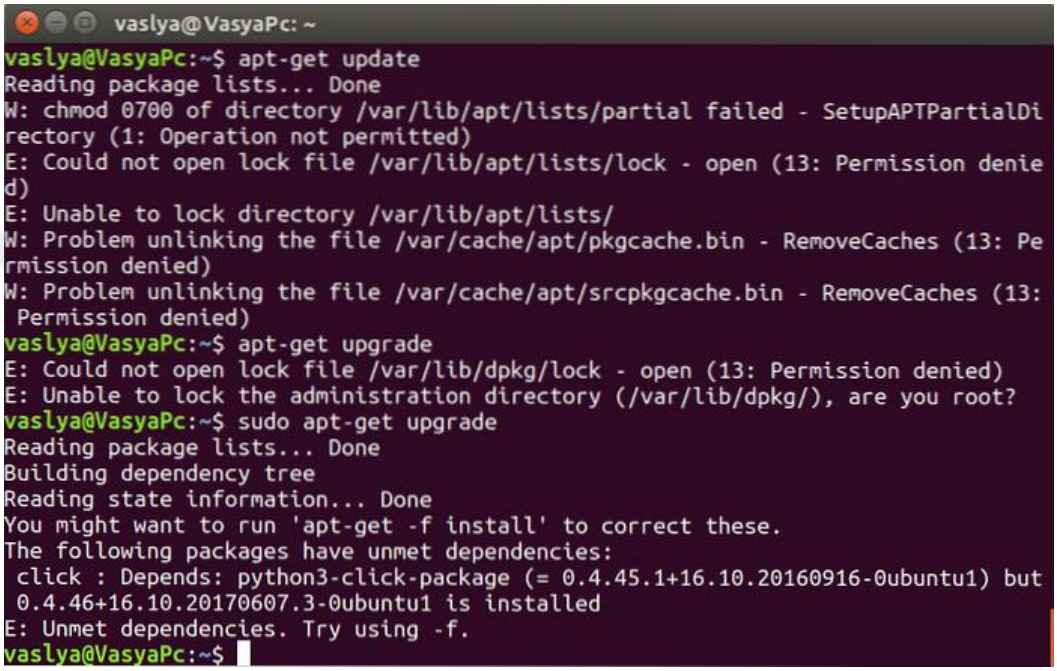
QT, gömülü sistemler için geliştirilmiş ancak cross platform olarak da çalışan çok kapsamlı bir framework'tür. C++ veya python tabanlı olarak çalışmaktadır. QT tarafından yeni geliştirilen arayüz programlama dili QML, javascript kullanmaktadır. QT linki EK-1 D'de verilmiştir.

QT'nin gömülü sistemler üzerinde çalışması için ya sistemin mimarisine göre derlenmesi gerekmektedir ya da hazır bir şekilde ubuntu, debian depolarından yüklenebilmektedir. Bu depolardan yüklenebilmesi için APT paket yöneticisine ihtiyaç vardır. Birçok işletim sisteminde bu paket yöneticisi hazır yüklü halde gelmektedir.

QT'yi yüklemek için gerekli komutlar:

- sudo apt-get install build-essential
- Sudo apt-get update && sudo apt-get upgrade
- Sudo apt-get install qt5-default

Paket yükleyici tarafından bize yöneltilen sorulara 'y' ile cevap verilmelidir. Şekil 3.25'de update ve upgrade çıktı örneği görülmektedir.



Şekil 3.25. Update ve upgrade çıktı örneği

Tüm adımlar başarılı olmuşsa paketler yüklenmiştir. Başarılı olup olmadığını 'qmake --v' komutu ile öğrenebiliriz. Eğer karşımıza versiyon bilgisini içeren cümle çıkıyorsa Qt cihaza başarılı bir şekilde kurulmuştur.

Qt'yi başarılı bir şekilde çalıştırabilmek için cihaz üzerine lima-mesa driverlarının kurulması gerekir. Bu driverlar orangepi üzerinde bulunan mali görüntü işlemcisinin kullanılmasına olanak sağlar.

- Mesa kaynak kodu indirilir (Mesa kaynak kodu için link EK-1 E'de verilmiştir).
- Mesa kütüphanesinin derlenmesi için gerekli kütüphaneler kurulur.
- sudo apt-get update
- sudo apt-get upgrade
- sudo apt-get install meson

- /etc/apt/sourceslist dosyasına girilir ve # ile kapatılan kaynaklar açılır. Yoksa altta belirtilen komut çalışmayacaktır.
- sudo apt-get build-dep mesa
- Mesa klasörüne giriş yapılır.
- Cd mesa/
- Mkdir build
- Cd build
- Meson ile bir önceki sayfada bulunan mesa kaynak kodları derlenir.
- meson build/ --optimization s --buildtype release --prefix=/usr/local --libdir=lib/arm-linux-gnueabihf -Dgallium-drivers=lima,panfrost,kmsro,swrast -Dplatforms=x11,drm,surfaceless -Dvulkan-drivers= -Ddri-drivers= -Dllvm=false
- Yukarıdaki fazladan gerçekleştirilen argümanlar dosyanın hangi lokasyona yükleneceğini, gerekli kütüphanelerin hangi klasörde bulunduğunu ve hangi driverların derlenerek kurulacağını belirtir. Mesa kütüphanesi normalde çok kapsamlıdır ve bir çok mimari için geliştirilmiştir. Ek argümanlar ile bu kütüphane Orange Pi için indirilmektedir.
- Derlenen kaynak kodları ninja ile kütüphane haline getirilir ve kurulum sağlanır.
- Sudo ninja install
- Lib/arm-linux-gnueabihf lokasyonuna eglfs için gerekli kütüphaneler kurulmuştur. Eglfs QT tarafından kullanılan grafik kütüphanelerindendir. QT bir çok grafik kütüphanesine destek sağlamaktadır. X11, eglfs v1, eglfs v2 bunlardan bir kaçıdır.
- Bu kurulumla birlikte mali görüntü işlemcisi kullanılabilir hale gelir.

3.3.5.3 EDID kurulumu

Cihazın 7 inch ekranı çalıştırabilmesi için bir edid donanımına ve edid'in 1024x600 ekran çözünürlüklerine uygun bir programa ihtiyacı vardır. EDID bir ekran sürme yöntemidir. HDMI cihazlar tarafından kullanılmaktadır. Donanım üzerinde edid çalıştırabilmek için mini bir eeprom eklenmelidir ve ekran sürmek için kullanılan data yolları ile birleştirilmelidir. EDID oluşturabilmek için çeşitli programlara ihtiyaç vardır. Tez çalışmasında hazır olarak temin edilen bir edid configürasyonunu kullanılmıştır. Bunun için 1 adet arduino ve 3 adet jumper kabloya ihtiyaç vardır.

- Arduino ide kurulumu yapılır (İndirme Linki EK-1 F'de verilmiştir).

Arduino IDE indirme seçenekleri Şekil 3.26'da görülmektedir.

Şekil 3.26. Arduino IDE indirme seçenekleri

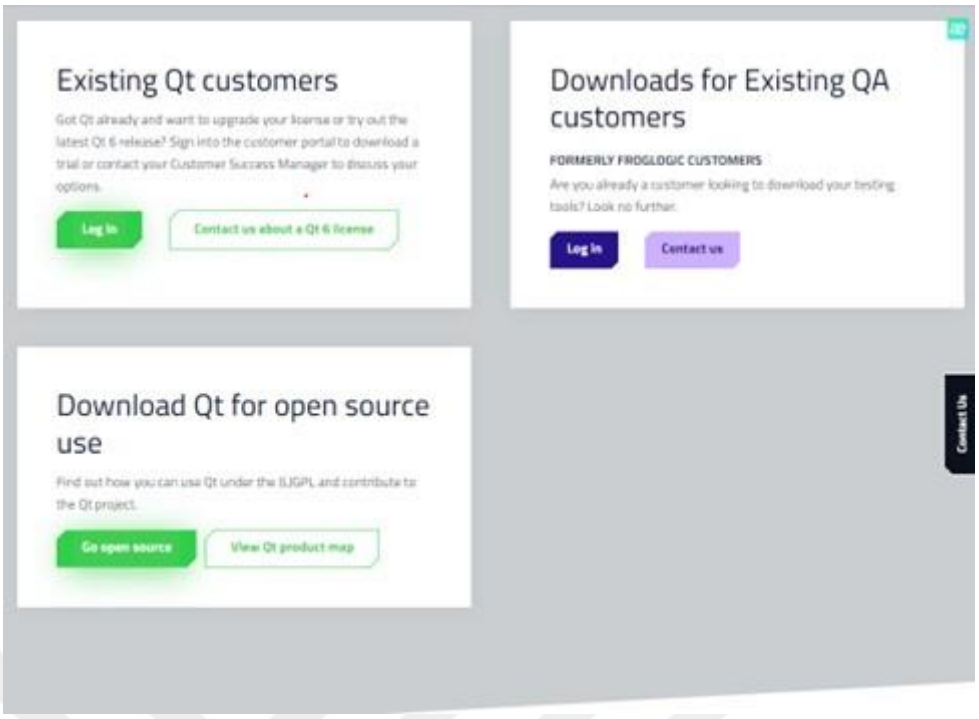
- Elde edilen EDID kodları uygulama bulunan cihaza uygun olarak configüre edilmiştir.
- TX RX GND bağlantıları EDID EEPROMU ile arduino arasında gerçekleştirilir ve arduino kodu çalıştırılır.
- Çalıştırılan bu kod ile EEPROM, EDID donanımı gibi çalışmaktadır. Bu da 1024x600 pixel çözünürlüklü ekranın sürülmesini sağlamaktadır.

3.3.5.4 Qt creator kurulumu

Ekran başarılı bir şekilde çalıştıktan sonra QT ile kodlar yazılmaya başlanır. QT kodunun yazılması için özel bir IDE vardır. Basit bir not defteri üzerinde uygun bir syntax ile de yazılabilir ancak yazım kolaylığı ve debugging için IDE kullanılacaktır. IDE kullanılmasının bir diğer önemli yanı da cross compile özelliğini desteklemesidir ancak bu ürün bir demo olduğu için QT sıfırdan kaynak kodu ile derlenmemiştir. Dolayısıyla şu anda cross compile özelliği aktif değil ancak bir ürün geliştirmek için önerilen yöntem cross compile'dır.

- QT creator, ilgili web sitesinden indirilerek kurulur (İndirme Linki EK-1 G'de verilmiştir).

QT creator indirme seçenekleri Şekil 3.27'de görülmektedir.



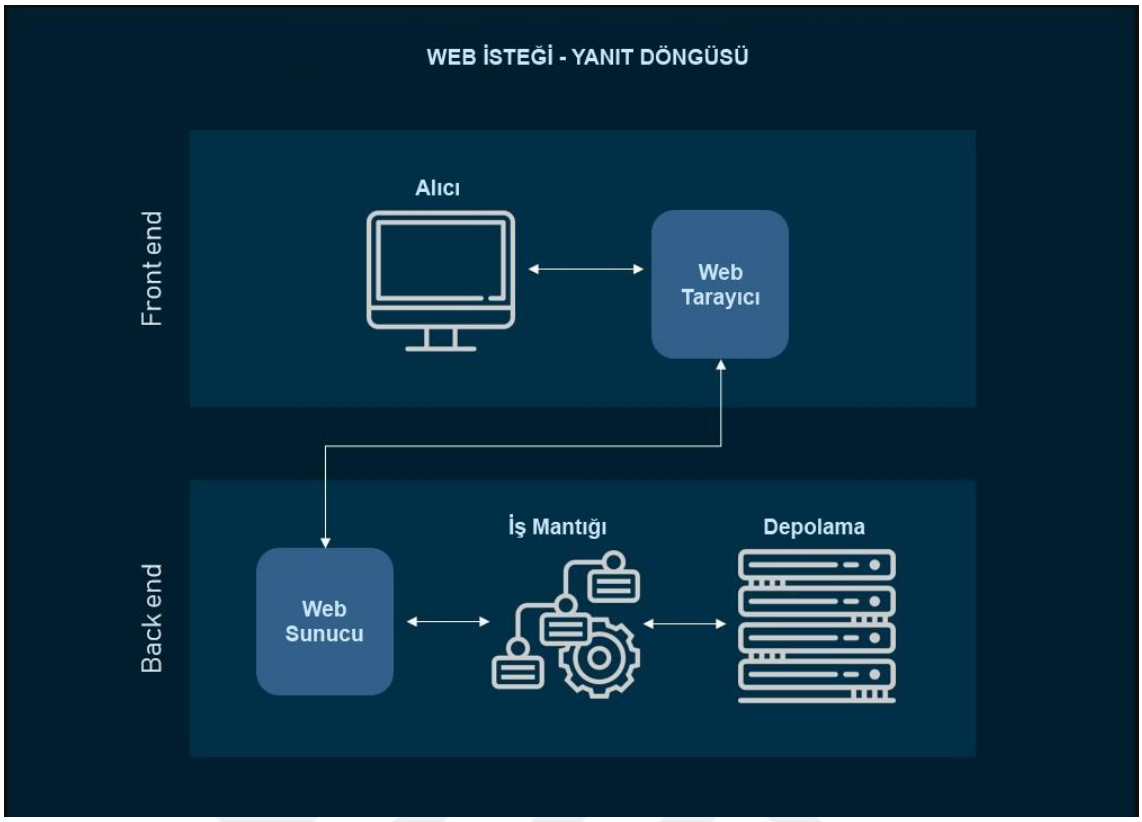
Şekil 3.27. QT Creator indirme seçenekleri

QT ve QT ile ilgili her şey açık kaynaktır dolayısıyla ücretsiz bir şekilde kullanılabilir. Ancak ürün geliştirmek için ya gerekli lisans ücretleri ödenmeli ya da açık kaynağa uygun olarak kaynak kodu paylaşımı yapılmalıdır.

3.4. Web Arayüzünün Tasarımı ve Yazılımı

IoT uygulaması için Photoshop ile web ara yüzü tasarlanmıştır. Aynı tasarım hem HMI ekranında hem de web ara yüzünde kullanılmıştır. HMI için bu tasarım Qt Creator'de parça parça eklenip her bir parçanın fonksiyonu yazılmıştır. Web yazılımda ise Front-End ve Back-End kısımları oluşturulmuştur. Basit tabirle açıklamak gerekirse; Front-End, istemcinin bilgisayarını veya mobil cihazını ve bulut bilişim sistemine erişmek için gereken uygulamayı ifade eder. Çoğu durumda, uygulama bir web tarayıcısıdır. Back-End ise bilgisayarlar, sunucular, işletim sistemleri ve depolama cihazları gibi bulut sağlayıcısının sahip olduğu bilgisayar altyapısını ifade eder. Bulutta depolanan tüm veriler genellikle depolama cihazlarında saklanırken, bulutta çalışan yazılım uygulamaları bilgisayarlarda çalıştırılır. Her yazılım uygulamasında ayrıca özel bir sunucu bulunurken, trafiği ve istemci taleplerini izlemek ve her şeyin gerektiği gibi sorunsuz çalışmasını sağlamak için merkezi bir sunucu kullanılır (Çelik ve ark, 2022).

Şekil 3.28'de front-end ve back-end yapısı görülmektedir.



Şekil 3.28. Front–End ve Back-End Yapısı

Tez çalışmasında sistem kurgusunu ifade edip deney ortamında gösterilmek üzere yeterli olan tasarım Şekil 3.29'da gösterilmiştir. Burada tasarıma yerleştirilen her bir parçaya (resim, buton, yazı vb.) widget denir. Tasarımda fonksiyon atanmak istenen kısımlar parçalanır, tek resim halinde kaydedilir ve widgetler oluşturulur. Çalışmada buluttan makinelere ve tam tersi yönde kurguyu gerçekleştirilmek üzere gerekli widget'ler uygun yerlere yerleştirilip tek tek fonksiyon atamaları yapılmıştır.

Bu tasarıma göre deney ortamında Endüstriyel Makine kontrolü sağlanacaktır. Örneğin HMI'dan veya web sitesinden Başlat butonuna (widget 1) basıldığında sistem Endüstriyel Mutfak Makinesiyle haberleşecek ve çalışmaya başlaycaktır. Bu senaryolar bölüm 4'te açıklanmıştır.

Şekil 3.29. HMI ve Web yazılım için hazırlanan tasarım

3.4.1 Front-End oluşturulması

Front-End tarafında HTML, CSS ve Java Script kullanılmıştır. Hiper Metin İşaretleme Dili olarak belirtilen HTML, web sayfalarını oluşturmak için kullanılan standart dildir. 2022 itibari ile dilin son sürümü HTML5'tir.

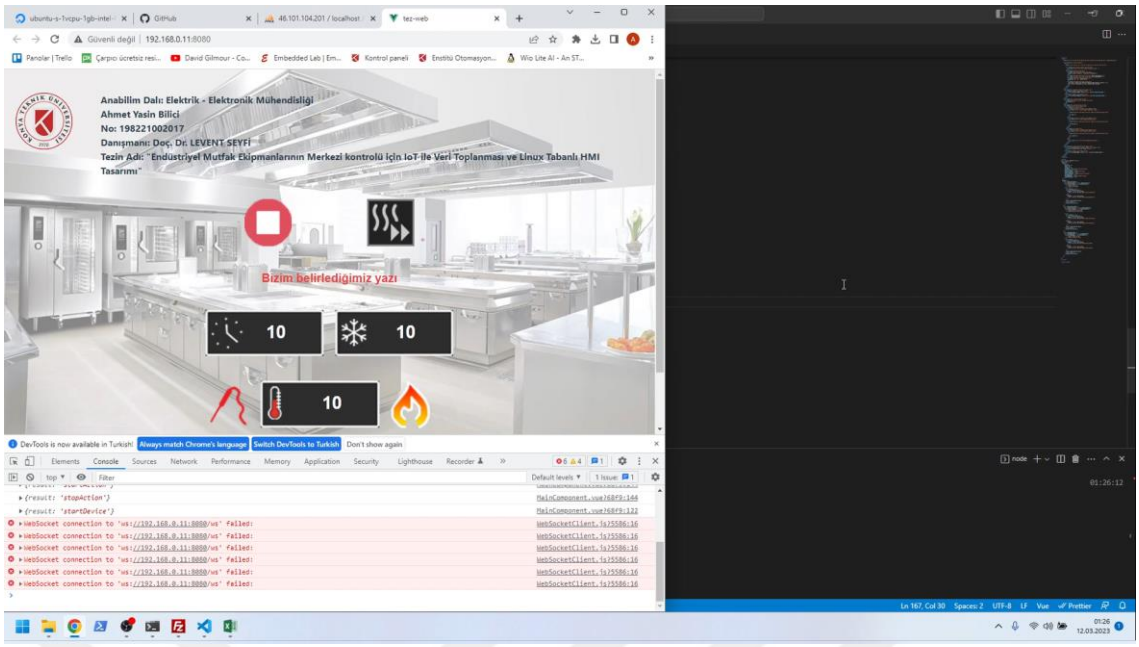
HTML, yalın bir programlama dili değildir. HTML kodlarıyla tek başına çalışan bir program yazılamaz. Ancak bu dili kullanabilen programlar aracılığıyla çalışabilen uygulamalar yazılabilir. Programlama dili diye ifade edilmemesinin nedeni tam olarak budur. Temel olarak görevi yazı, görüntü, video gibi değişik verileri ve bunları içeren sayfaları birbirine basitçe bağlamak, ayrıca söz konusu sayfaların web tarayıcısı yazılımları tarafından düzgün olarak görüntülenmesi için gerekli kuralları oluşturmaktır. HTML kodlarını web tarayıcıları yorumlar ve görsel hale dönüştürürler, dolayısıyla aynı kodun farklı tarayıcılarda farklı sonuç vermesi muhtemeldir. CSS ve JavaScript ile beraber kullanıldığında görsel ve dinamik web siteleri oluşturulabilir. (Wikipedia, HTML, 2022).

Cascading Style Sheets (CSS), HTML'e ek olarak metin ve format biçimlendirme alanında fazladan olanaklar sağlayan bir işaretleme dilidir. İnternet sayfaları için genel şablonları hazırlama olanağı verdiği gibi, bağımsız olarak harflerin renk, yazı tipi, büyüklük gibi özelliklerini ayarlamak için de kullanılabilir. Bu tekniğin en önemli özelliği kullanımındaki kolaylık ve esnekliktir. Bir Web sayfasında birbiriyle uyumlu birkaç renk

ve yazı tipi kullanıldığında, bunları her sayfada ayrı ayrı tekrar belirtmek yerine CSS yardımıyla tek seferde tanımlayıp bütün Web sayfalarında ortak olarak kullanılabilir. Böylece sayfaların hafızadaki boyutu yeterli miktarda küçüldüğü gibi güncelleme yapmak da kolaylaşır. (Wikipedia, CSS, 2022).

JavaScript (JS), HTML ve CSS ile birlikte dünyadaki tüm web yazılım temel teknolojilerinden biri olan programlama dilidir. Web sitelerinin büyük çoğunluğu, web sayfası hareketleri için JavaScript kullanırlar. Tüm büyük web tarayıcılarında, kaynak kodunun cihazlarda çalışabilmesi için özel bir JavaScript motoru bulunur. JavaScript, ECMAScript standardına uyan, çoğunlukla eş zamanlı olarak derlenmiş, üst düzey bir dildir. Prototip tabanlı obje yönelimi, dinamik yazma ve birinci sınıf işlevlere sahiptir. İşlevsel, olay odaklı ve zorunlu programlama stillerini destekleyen çoklu yapıdadır. Standart veri yapıları, metinler, tarihler ve düzenli ifadeler Belge Obje Modeli (DOM) ile birlikte çalışmak için uygulama programlama arayüzlerine (API'ler) sahiptir. ECMAScript standardı, depolama, ağ oluşturma veya grafik olanakları gibi herhangi bir giriş/çıkış içermez. Pratikte, web tarayıcısı veya diğer çalıştırma ortamları, giriş/çıkış için JavaScript API'leri sağlarlar. (Wikipedia, JavaScript, 2022).

Tez çalışmasında bu bahsedilen 3 yapının da birlikte kullanıldığı Vue.Js framework kullanılmıştır. Yapılan web yazılımının yapısal olarak çalışması için vue.js ve gerekli yapıtaşları kurulup ortam sağlanmıştır. Görüntü framework için ise Tailwind eklentisi kurulmuştur. Tüm bu çalışmalar Visual Studio Code yazılım geliştirme ortamında sağlanmıştır. Html-css-javascript geliştirmek için özel IDE’ler bulunmaktadır ancak ücretsiz ve yaygın kullanımı olduğu için ayrıca vue extensionlarını desteklediği için Visual Studio Code kurulumu yapılmıştır. Şekil 3.30’da HMI tarafında yapılan aynı tasarımın web tarafında da yazılma çalışması gösterilmiştir.



Şekil 3.30. Web Front-End çalışması

3.4.2 Back-End oluşturulması ve veritabanı

Back-End tarafında node.js ve veritabanı için ise MySQL kullanılmıştır. Son zamanlarda sunucu tarafı JavaScript alanında popülerlik kazanan en ilginç gelişmelerden biri Node.js'dir. Ana akım çoklu iş parçacığı yaklaşımına dayanmayan, ancak olay güdümlü bir programlama modeliyle eşzamansız G/Ç kullanan yüksek performanslı, eşzamanlı programlar geliştirmeye yönelik bir çerçevedir (McCune, 2011).

MySQL, veri tabanları için bir sistemdir ve çok kullanışlıdır. Diğer ticari veri tabanlarında mevcut olan tüm gelişmiş özelliklerin bakımı ve ticari uygulamalar için çekici bir seçenektir, çünkü kullanımı kolaylaştıran ve pazarda çalışma süresini azaltan bir iş ortamıdır. Bu ek fayda sağlayan bir araç olarak, alt düzey bir kurulum ve hızlı bir geri dönüş sağlar. MySQL, birden fazla platform için kullanılabilir, GNU/Linux'taki en iyi kütüphane örnekleri için seçilir. Ayrıca pratikte diğer platformlardan farkı, mysql-client istemcisi olarak en iyi şekilde kullanıldığında, MySQL bir metin modu sunucusuyla etkileşime izin verir. Bu mod, bir sunucunun yerel olarak kurulmasını sağlamak için tüm cihazlara olanak sağlar (Santillán ve ark, 2014).

Tez çalışmasında Back-End tarafında geliştirilen yazılım, Front-End tarafında geliştirilen yazılım ile bağlanmıştır. Back-End yazılımı MQTT protokolü ile buluttan veri alıp vermektedir. Yani sistemde veritabanı, kısmi depolama ve haberleşme işlemlerini

back-end tarafı yapmaktadır. Front-End tarafı sadece web sayfalarında kullanıcının göreceği görsel durumları sağlar.

3.4.3 Website geliştirilmesi

Websitesi için digitalocean üzerinden bir sunucu kiralanmıştır. Sunucu ücretleri aylık $5'dır ve linux tabanlı sunucular barındırılmaktadır. Orada bir ubuntu sunucu kiralanmış ve sunucunun dropleti oluşturulmuştur. Droplet, digitalocean'ın sunucularına verdiği isimdir. Kiralanan sunucuya IP adresi ile bağlanılmıştır.

SSH kullanılmıştır. Ssh root@ip-address kodu ile root olarak bağlanıldığı için password kısmına root şifresi girilmelidir. Bu şifre kurulum sırasında belirlenmektedir. Tez çalışmasında MQTT kullanacağımız için MQTT kurulumu da yapılmıştır.

3.4.3.1 Website kodunun planlanması ve gereklilikler

Kod yazılmadan önce nesneye yönelik programlama temelleri kullanılarak yazılım aşağıdaki gibi planlanmıştır.

1. Gereklilikler belirlenmiştir.
2. Kullanıcı hikayeleri oluşturulmuştur.
3. Kullanıcı hikayeleri kullanım durumlarına dönüştürülmüştür.
4. Temel nesneler ve varlıklar ve bunların ilişkileri belirtilmiştir.
5. Modelleme yapılmıştır.
6. Nesnelerin durumları ve yaşam döngüleri gösterilmiştir.

3.4.3.2 Gerekliliklerin belirlenmesi:

Fonksiyonel gereklilikler:

- Uygulama, QT ve C++ ile geliştirilmelidir.
- Uygulama üzerinde makine arayüzü için temel işlevler gösterilmelidir (Buton, switch, text, slider vs.).
- Uygulama IoT özelliklerini taşımalıdır (Uzaktan kontrol).
- Uygulama MODBUS ile çoklu haberleşme yapabilmelidir.
- Uygulama Linux işletim sistemini kullanmalıdır.

- Kullanıcılar hem uzaktan hem de panel üzerinden cihazların kontrolünü yapabilmelidir.

  Fonksiyonel olmayan gereklilikler:
- Uygulamanın kullanımı kolay olmalıdır.
- Güvenliğe dikkat edilmelidir.
- Arayüz göze güzel görünmeli ancak basit olmalıdır.
- Uzaktan kontrol için geliştirilen websitesi birçok internet tarayıcısına uygun olmalıdır.

3.4.3.3 Kullanıcı hikayelerinin oluşturulması

Kullanıcı hikayeleri 3 ana başlıkta oluşmaktadır. Kontrol, iletişim ve güvenlik.

Kontrol:

- Kullanıcı, cihazları gömülü ekran üzerinden yönetebilmeli ve gözlemleyebilmelidir (Gömülü Panel Arayüzü).
- Kullanıcı, cihazları bilgisayar üzerinden yönetebilmeli ve gözlemleyebilmelidir (Websitesi Arayüzü).

İletişim:

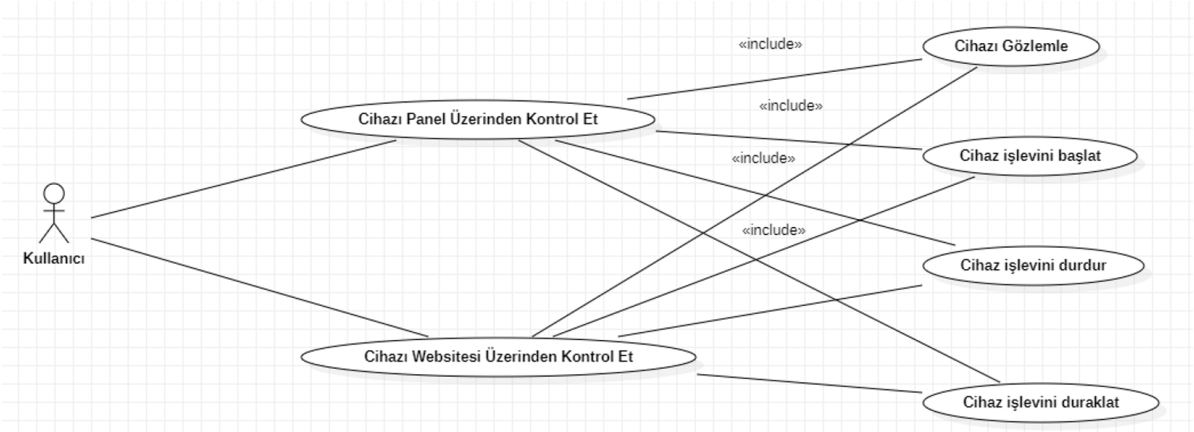
- Kullanıcı, cihazlar ile ekran üzerinden haberleşebilmelidir (MODBUS).
- Kullanıcı, cihazlar ile bilgisayar üzerinden haberleşebilmelidir (MQTT).

Güvenlik:

- Kullanıcı, haberleşme için verilerin aktarımı sırasında çalınma veya hatalı aktarım hususunda endişe duymayacağı düzeyde güvenilirlikle veriler iletilmelidir.

3.4.3.4 Temel işlevlerin modellenmesi

Kullanıcı panel üzerinden görüntüleyip kontrol ettiği her eylemi website üzerinden de sağlayabilmelidir. Kullanıcı için basit bir kontrol akışı Şekil 3.31'de gösterilmiştir.



Şekil 3.31. Kullanıcı için basit kontrol akışı

3.5. IoT ile Uzaktan Veri İzleme ve Kontrol

3.5.1. Nesnelerin interneti (Internet of Things, IoT)

IoT kavramı, 1999 yılında Radyo Frekans ile Tanımlama (RFID) teknolojisinin P&G firması için sağladığı yararlarla ilgili bir sunumda Kevni Ashton tarafından ilk kez kullanılmıştır. IoT, fiziksel nesnelerin birbirleriyle ya da daha büyük sistemlerle bağlantılı olduğu iletişim ağıdır. İnternet üzerinden cihazlara ve sistemlere bağlanmak ve veri alışverişi yapmak için sensörler, yazılımlar ve diğer teknolojilerle bağlantılıdır (Görkem ve Bozuklu, 2016).

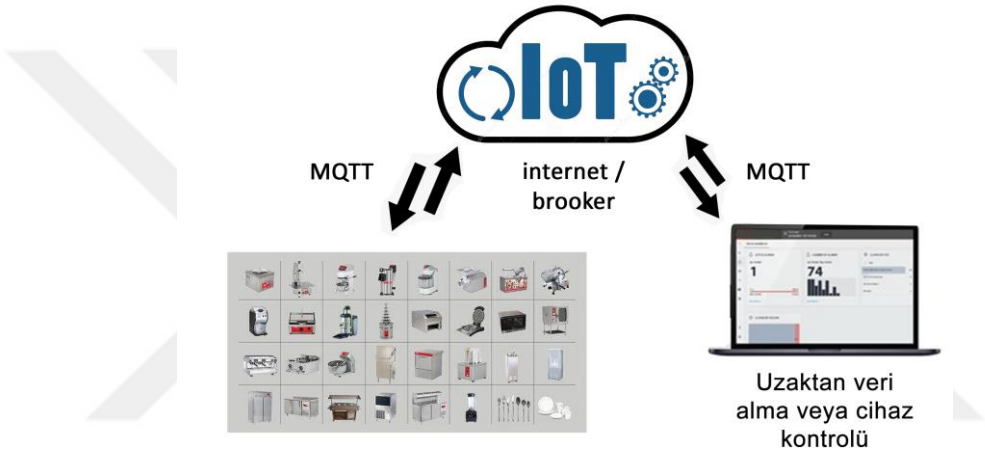
IoT kısaltmasıyla bilinen bu teknoloji, her "nesneye" benzersiz bir kimlik atayarak herhangi bir insan müdahalesine gerek kalmadan birbirleriyle ve merkezi kontrol mekanizmalarıyla veri paylaşımı yapmalarını sağlar. Daha önce veri toplama ve bilgi paylaşma kabiliyeti olmayan nesnelerin akıllandırılmasına, yeni kabiliyetler kazandırılarak internete bağlanabilmelerine imkan tanımaktadır (Innova, 2022).

Şekil 3.32'de nesnelerin interneti kullanımının katmanları gösterilmiştir.

Şekil 3.32. IoT katmanları

Endüstriyel mutfak sektöründe IoT kullanımı, kontrol kartlarının aldığı sensör verileri, kullanıcı ayarları, ürün reçeteleri, yiyecek çeşitliliği, kullanım adetleri, sıcaklık ve nem değerleri gibi verilerin uzaktan izlenmesi ve güncellenmesiyle olmaktadır (Akram ve Dağdeviren, 2020).

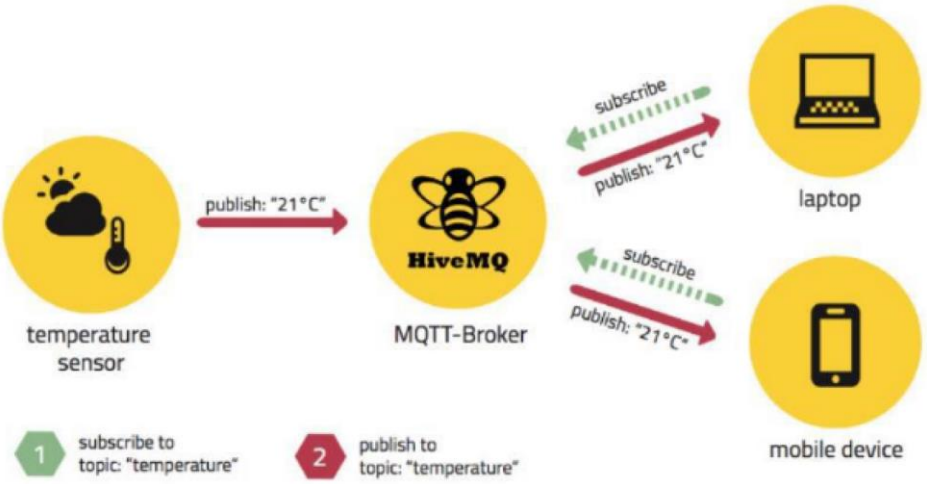
Yapılan tez çalışmasında Endüstriyel mutfak makinelerinde IoT teknolojisinin kullanımının izahı Şekil 3.33'de görülmektedir. Burada IoT ile hem otomasyon hem de müşterilerden veri toplama işlemleri uygulanmıştır. Her iki işlem için de gerekli donanım ve altyapı aynıdır.



Şekil 3.33. Endüstriyel mutfak ekipmanlarında IoT teknolojisinin kullanımı

3.5.2. MQTT (Message Queuing Telemetry Transport)

Endüstri devrimiyle birlikte adını sıkça duyduğumuz IoT, hayatımıza MQTT terimini sokmuştur. MQTT, yayınlama ve abone olma esasına dayanan bir telemetri mesaj protokolüdür. Bu protokol makineler arasındaki iletişimi sağlar. Bu nedenle de IoT kullanımında çok önemli bir yere sahiptir. TCP/IP üzerinden mesaj yayınlayıp almayı sağlar ve oldukça hafif bir protokoldür. Hafif protokol olması diğer benzer protokollerden ayıran en önemli özelliklerinden biridir (Endüstri40, 2022). Şekil 3.34'de temel ve basit bir MQTT mimarisi görülmektedir.



Şekil 3.34. Basit bir MQTT mimarisi

MQTT, buluttan kontrol ve gözlenmeye ihtiyacı olan cihazlar için network oluşturmaya imkan sağlar. MQTT, nadiren rastlanılan bant genişliği kısıtlamalarından oluşan gecikmelerin yaşandığı kablosuz ağlar için iyi bir seçimdir. Abone olan birimden broker (aracı) birimine olan bağlantı yapılır ve broker, abone birim online olduğunda bu mesajı ona tekrar iletir. Yayınlayan birimden broker birimine olan bağlantı ise kesilmelidir ve broker birimi, bağlantıyı kapatabilir ve yayıncıdan aldığı komutu abone birime iletir (Endüstri40, 2022).

3.5.3. IoT ile MQTT ilişkisi

Nesnelerin interneti ile beraberinde birçok konu gelmektedir. Bunlardan biri MQTT'dir. Birçok nesnenin verisi inanılmaz büyük olacak ve anlık olarak akacaktır. Bu sensör ve telemetri verilerini nasıl işleyeceğiz sorusunun cevabı MQTT standardıdır. Son zamanlarda araştırmacılar IoT için her birinin avantaj ve dezavantajları olan birçok iletişim protokolü geliştirmiştir. IoT için veri iletişim protokollerinden biri olan MQTT'nin bir iletişim protokolü olarak kullanılmasının nedeni kullanılan veri kalitesinin ve güvenilirliğinin artırılmasıdır (Atmoko ve ark., 2017).

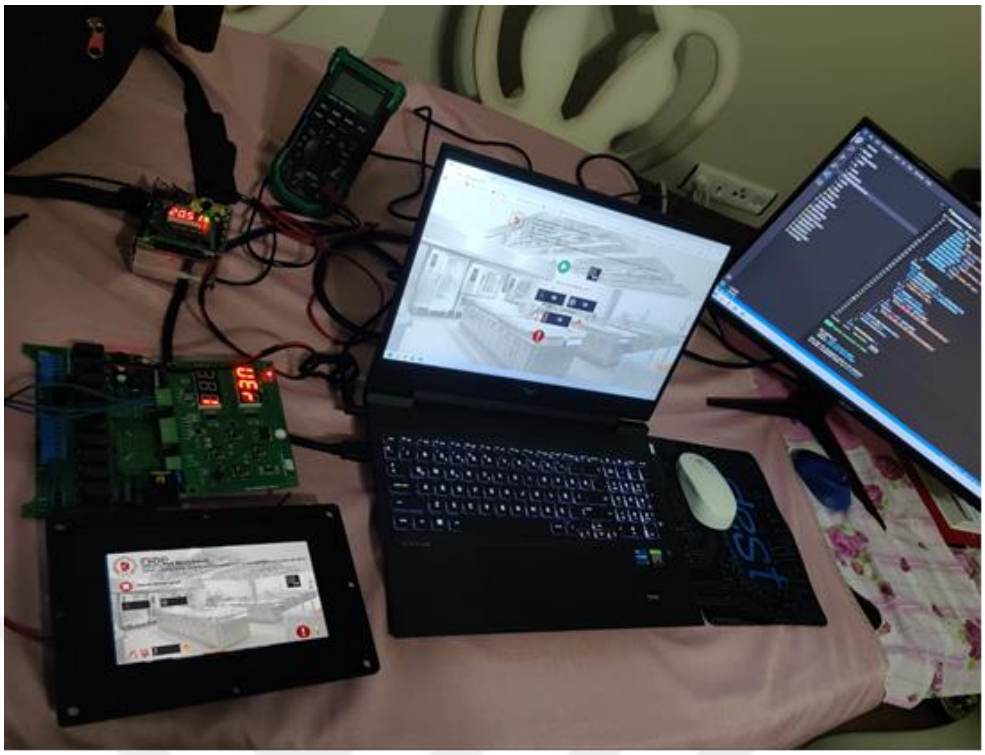
4. ARAŞTIRMA SONUÇLARI VE TARTIŞMA

Tez çalışmasında materyal ve yöntemleri açıklanan sistemler kullanılarak bir deney seti kurulmuştur. Deney seti kurulurken kullanılan sistemlerdeki donanım ve yazılım araştırmaları, kurulumları, denemeleri ve uygulamaları yapılmıştır. Araştırması yapılan ve uygulanabilir sonucuna varılan sistemlerin açık kaynak kodları EK-2, EK-3 ve EK-4'de verilmiştir.

Gerçekleştirilen deney sistemine göre yapılanlar aşağıda listelenmiştir:

1. Deney düzeneğinde endüstriyel mutfak makinesini temsil eden bir elektronik kontrol kartı tasarlanıp gerçekleştirilmiştir. Bu kontrol kartlarıyla Modbus haberleşme yapan ve asıl yönetim işlerini yapan, internete bağlanan master HMI kullanılmıştır. IoT uygulaması için de bu cihazlarla MQTT protokolü ile haberleşen Web yazılım tasarlanmış ve kullanılmıştır.
2. Web yazılım üzerinden (uzaktan) endüstriyel mutfak makinelerinin donanımları kontrol edilmiştir (Fan, Rezsitans, Lamba vb.).
3. Makine sensörlerinden alınan bilgiler Web yazılım üzerinden anlık olarak izlenmiştir (Sıcaklık, Sayaçlar vb.).
4. HMI ve Website için aynı görsel tasarım yapılıp kullanılmıştır. Her ikisinden de aynı yönetim işlevleri ve veri alma sağlanmıştır.
5. Genel olarak yukarıdaki kontrollerin haricinde tez projesinin amaçlarını yerine getirecek birtakım senaryolar oluşturularak denenmiştir. Bunlar;
    - Yüksek sıcaklık verisi alındığında uzaktan bilgilendirme yapılması ve cihazların otomatik durdurulması,
    - Hijyen takibi adına fazla ve çabuk kirlenen cihazlar için sayaç programlanması ve bu sayaç dolduğunda cihaz oto yıkama yapmadan veya manuel temizlenmeden tekrar çalışmaması,
    - Bulaşık makinesi gibi cihazlarda kullanılan deterjan gibi sarf malzemelerinin takibi yapılarak azalması durumunda uzaktan satın alma bildirimleri oluşturulması,
    - Endüstriyel mutfak üreticilerinin müşterilerinin makinelerinden veri toplanarak bir takım hizmet ve gereksinimlerin sağlaması örnek uygulamaları yapılmıştır.

Gerçekleştirilen ilk deney seti Şekil 4.1'de görülmektedir.



Şekil 4.1. Projenin ilk testlerinin yapılması

Endüstriyel Mutfak makinesini temsil eden elektronik kart kutuya montajlanarak şekil 4.2'deki halini almıştır. Bu cihazda bulunan 2 adet 7 segment ekranın üstteki makine çalışma sayacını, alttaki ise makinede kullanılan deterjan vb. sarf malzeme kullanımını temsil etmektedir. 4 adet butondan soldaki sistemi çalıştırmakta, sağdaki ise sistemi durdurmaktadır. Yukarı ve aşağıdaki butonlar ise sıcaklık ayar butonlarıdır.

Şekil 4.2. Endüstriyel Mutfak Makinesi Kontrol Cihazı

Cihazda Endüstriyel Mutfak Makinesini kontrol edecek 2 çıkış bulunmaktadır. Son olarak güvenlik için 1 adet motor termik dijital girişi bulunmaktadır. Cihaz senkron olarak sürekli HMI ve Websitesi ile haberleşmektedir. Cihaz çalıştırıldığında üstteki 7 segment ekranda çalışma sayacı saymaya başlayacaktır. On kez çalışmadan sonra cihaz kilitlenip yeni bir onayla geri açılacaktır. Alttaki 7 segment ekranda ise başlangıç olarak 5 birim sarf malzemesi kaldığı gösterilmiş ve cihazın her çalışmasında bir azalmaktadır. Sıfır olduğunda sarf malzeme uyarısı verip ototmatik satın alma talimatı oluşturulmuştur. Bu uygulamalarda Endüstriyel Mutfaklarda IoT uygulaması için uzaktan çalışma kontrolü ve veri toplanması gösterilmiştir. Cihazın dijital girişine termik bilgisi geldiğinde sistem tamamen duracak, HMI ile Websiteye anlık olarak uyarılar düşecek ve sistem durdurulacaktır. Cihaz ayarlanan sıcaklığın üstünde ölçüm yapacak olursa yine HMI ve Websiteye anlık olarak yüksek sıcaklık uyarıları gönderip sistemi otomatik durduracaktır.

HMI ve Website tasarımları için aynı widgetler kullanılmıştır. Tasarımları da aynıdır. Bütün sistem elemanları tamamen senkron çalışmaktadır. Şekil 4.3'de HMI tasarımı gösterilmiştir.

Şekil 4.3. Master HMI Cihazı

HMI ekranında görülen zaman ve sarf malzeme sayacı elektronik cihaz ile aynı göstergelerdir. Aynı sayaçlardan Website tasarımında da bulunmaktadır. En aşağıda bulunan kutucuk ise sıcaklığı göstermektedir. Ayrıca ekranda elektronik cihazı başlatıp durduracak buton, lamba butonu ve durumları bildiren yazıların gösterildiği bir alan bulunmaktadır. HMI'den cihazı başlat komutu veya lamba açma gibi komutlar gönderildiğinde tüm sistem senkron olarak çalışmaya başlamaktadır. Sistemin çalışır durumda olup olmadığı, sarf malzeme tükenmesi, cihaz kilitlenmeleri, uyarılar vb. bildirimler yazı alanında gösterilmiştir. Sıcaklık ayarlanan değeri geçtiğinde tüm sistem duracak ve ekranda yüksek sıcaklık uyarı ikonu çıkacaktır. Aynı şekilde motor termik attığında da termik uyarı ikonu çıkıp kullanıcı yazı alanından bilgilendirilecektir. Tüm bu işlemler eş zamanlı olarak websiteden de kontrol edilebilecektir. Ayrıca kullanıcıyı bilgilendiren görseller ve yazılar da hangi taraftan kontrol edilirse edilsin eş zamanlı olarak değişecektir.

Yukarıda bahsedilen tüm senaryolar cihazlar üzerinde denenmiş ve çalıştırılmıştır. Yazılımsal oluşan hatalar giderilmiştir. Modbus donanım biriminde oluşan parazitler giderilmiştir. Sistemin kurgulandığı üzere tüm işlevini yerine getirdiği gözlenmiştir.

4.1. Modbus Haberleşmede Parazit Oluşması

Tamamlanan proje Şekil 4.4'de görüldüğü gibi sahada bir endüstriyel mutfak makinesine montaj edilmiş olup sadece belli başlı birkaç giriş ve çıkışın kontrol edilmesi gözlemlenmiştir. Gözlemler sonucunnda motor sürücünün devreye girmesiyle birlikte Modbus bağlantılarında kopmalar görülmüştür. Bu nedenle Modbus hattında oluşabilecek parazitler ve giderilmesi konusunda yapılan araştırmalar sonucunda;

- Cihazların merkezden çıktıktan sonra sırayla biribirine bağlamak yerine bir düzen olmadan karışık bağlantı yapılması,
- Bazı durum ve yerlerde iletişim hızının (Baud Rate) yüksek olması,
- 50 metreden uzun mesafelerde sonlandırma direncinin bulunmaması,
- Toprak bağlantısının olmaması,
- Zırhlı kablo kullanılmaması
- Modbus haberleşme kablolarının yüksek voltaj veya frekansta kablolarla bir arada dip dibe bulunmasından dolayı parazit oluşabilmektedir.

Deney setinde tek bir slave cihaz kullanılmıştır. İletişim hızı 115200 bps olarak ayarlanmıştır. Deney ortamı bu hıza uygundur. Uzak mesafeli bir durum söz konusu

değildir. Toprak bağlantıları düzgün yapılmıştır. Blendajlı kablo kullanılmıştır. Ancak yüksek frekanslı kabloların yanyana geçmesinden dolayı parazit oluştuğu anlaşılmıştır. Bu kablolar ayrılarak farklı noktalardan götürüldüğünde sorun çözülmüştür.

Şekil 4.4. Makine üzerinde deneme yapılması

4.2. Mini Bilgisayar Kullanmanın Avantajları

Mikrokontrolörle aylarca süren proje süreçleri güncel Linux sürümleri çalıştıran mikrobilgisayarlarla, aynı kodu farklı donanımlarda çalıştırabilme ve en az yarı süresinde yapabilme imkanı sağlamıştır. PC'de yazılan uygulamayı hiç değiştirmeden donanıma derleyebilmek büyük bir kolaylık sağlamıştır. .(dot)Net core, Java, Python, PHP teknolojileri ile artık gömülü sistemlerde de uygulama geliştirilebildiği ve dünyanın bu çözümlere doğru gittiği aşikardır. İnternet bağlantıları, efektif depolama alanları vb donanım birimleriyle işler daha kolay hale gelmiştir.

4.3. Linux Derleme için VirtualBox kullanılması

Projede kullanılan Orange Pi mini bilgisayarın işletim sistemini oluşturmak ve derleme yapabilmek için Linux bilgisayar ihtiyacı olmuştur. Hali hazırda Linux işletim sistemi kurulu bilgisayarın olmamasından dolayı mevcut bilgiayara format atmak yerine VirtualBox ile sanal Linux bilgisayar kurulmuştur. Bu program örneğin Windows işletim sistemi içinde sanal bir makina oluşturarak bir veya daha fazla işletim sistemi kurmaya yarayan, sistem içinde sanal bir sistem oluşturan bir programdır (Watson, 2008). Sanal

Makinanın RAM'inin boyutunu, işlemci çekirdeklerinden kaç tanesinin sanal makineye atanacağı gibi ayarları yaparak, Linux işletim sistemli bir bilgisayar sağlanmıştır. Bu sayede hem tek bilgisayardan tüm proje yönetildiği için kolaylık sağlanmış hem de ekstra masraf oluşturacak ikinci bir bilgisayar ihtiyacı ortadan kalkmıştır.

4.4. Neden DigitalOcean?

Genellikle hosting firmaları işin ticaret boyutu dolayısıyla müşterilerin server özellik olanaklarını çok kısmaktadır. DigitalOcean bu durumda Github'la birlikte de çalışarak bu kısıtları ortadan kaldırmış ve donanımlı bir çalışma alanı sunmaktadır.

4.5. Front-End ve Back-End Yazılımı Farklı Projelerde Tasarlamanın Avantajları

Projenin Web yazılım tarafında ilk olarak Front-End ve Back-End aynı projede başlanmıştı. Bir süre sonra Vue.js içinde MQTT veritabanı gibi yapıları çalıştırmak karmaşaya sebep olmaya başladı. Sonrasında bu işlemleri farklı projelerde yapıp bir API ile sonradan birbirine bağlanması işleri kolaylaştırmıştır.

5. SONUÇLAR VE ÖNERİLER

Bu uygulama çalışması, endüstriyel mutfak sektöründe gelişmekte olan IoT uygulamasına katkı sağlayacağı düşüncesiyle gerçekleştirilmiştir. Ayrıca endüstride sıkça kullanılan Modbus haberleşme, gömülü sistemlerde sıklıkla duyulan QT-QML programlama dili, yazılımın her alanında kullanılabilen C++ programlama dili, işlemci programlamada adından sıkça söz ettiren C programlama dili ve full stack web tasarımı için kullanılan front-end ve back-end uygulamaları da çalışma içerisinde anlatılarak bu konularda çalışacaklara önemli bir kaynak oluşturulmuştur. Elektronik kontrol kartı tasarımı, kontrol kartı yazılımları, gömülü sistemler ve yazılımlar, Linux sistem üzerinde yazılım geliştirilerek bir HMI çalışması, web tasarımı ve IoT uygulaması başarıyla gerçekleştirilmiştir.

Bir endüstriyel mutfak makinasının sensörlerinden bilgiyi toplayıp, işleyip güvenli haberleşme protokolleriyle HMI'lerle bağlantı kurup sağlıklı veriyi buluta atabileceği anlaşılmıştır. Endüstriyel mutfak sektörü doğrudan beslenmeyi ilgilendiren bir sektör olduğu için hiçbir zaman önemini yitirmeyeceği ve gelişen dünyayla IoT teknolojilerinin bu sektör içerisine daha çok gireceği görülmektedir. Makinaların sahip olduğu HMI'ların, insan makine etkileşimini sağlaması bakımından IoT haberleşmesi için oldukça önemli olduğu görülmüştür. Ayrıca Modbus haberleşmenin endüstride uygulanan güvenli bir iletişim sistemi olduğu görülmüştür.

Geleceğin önemli teknolojilerinden biri olan IoT, Endüstri 4.0'ın olmazsa olmaz bileşenleri içinde yer almaktadır. IoT hızlı bir şekilde izleme, kontrol, haberleşme ve otomasyon sistemlerinde kullanılmaya başlamıştır. Veriler çoğaldıkça yeni standartlar geliştirilmiş ve akışı sağlanmıştır. HMI'larda sıradan işlemciler yerine internet özelliği olan mini PC kullanılmasının önemi görülmüştür.

Günümüzde web yazılım adeta bir trend haline gelmiştir. Özellikle Covid-19 pandemi dönemi ile birlikte internet-web işlerinin çok daha fazla artmasının da etkisiyle nerdeyse günümüz web arayüzlerinin önünde geçmektedir. Yapılan tez çalışmasında endüstriyel mutfak sektöründe IoT uygulamasında da güncel web geliştirme sistemlerinin kullanılmasının güvenli ve uygulanabilir olduğu görülmüştür.

Bir üreticinin müşterilere sattığı makinelerden topladığı bilgilerle onlara hem destek sağlayıp hem de yönetebileceği veya takip edebileceği görülmüştür. Bu bağlamda endüstriyel mutfak sektöründe IoT kullanımı diğer geleneksel müşteri servis yöntemlerine göre çok daha etkili olduğu gözlemlenmiştir.

Yapılan çalışma sistemiyle endüstriyel mutfak makinalarında üretilecek ürün reçetesine göre hangi yörede en çok hangi besinlerin tüketildiği öğrenilebilir. Pişirme verilerine, bölgelere göre damak zevkleri tespit edilebilir. Besinlerin tüketim miktarı belirlenebilir. Bu konuda bir pazar araştırması ortaya konabilir. Bulaşık makinalarının deterjanının önceden biteceğini anlayan sistem otomatik sipariş verebilir. Makinalar, gerekli sıcaklık ve neme göre kendini önceden hazır hale getirebilir. Alınan verilere göre reklam çalışmaları yapılıp ekranlarda gösterilebilir. Burada insan-makine etkileşimini sağlayacak olan HMI'lar büyük öneme sahip olacaktır.

KAYNAKLAR

Ahmad, M. O., Markkula, J., & Oivo, M. 2013, Kanban in software development: A systematic literature review, *2013 39th IEEE Euromicro conference on software engineering and advanced applications*, pp. 9-16.

Akkaya, Ş., 2015, FPGA Tabanlı Modbus Ağ Geçidi Tasarımı, Doktora Tezi, İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü.

Akram, V., & Dağdeviren, O., 2020, Nesnelerin interneti için gerçek zamanlı tasarsız veri toplama platformu. *Bilişim Teknolojileri Dergisi*, *13*(4), 451-462.

Arslan, S., Gündüzalp, M., Türk, E., 2016, Gömülü Bir Sistem İçin Bir Çoklu Ortam Uygulaması, Electrical, Electronics and Biomedical Engineering Conference (ELECO 2016), 189 – 193.

Atmoko, R. A., Riantini, R., & Hasin, M. K., 2017, IoT Real Time Data Acquisition Using MQTT Protocol, *Journal of Physics: Conference Series*, Vol. 853, No. 1, p. 012003, IOP Publishing.

Beg, S., Anjum, A., Ahmad, M., Hussain, S., Ahmad, G., Khan, S., & Choo, K. K. R., 2021, A Privacy-Preserving Protocol for Continuous and Dynamic Data Collection in IoT Enabled Mobile App Recommendation System (MARS), *Journal of Network and Computer Applications*, 174, 102874.

Choi, S. Y., & Moon, S. J., 2007, The Design and Implementation of Embedded Linux-Based Industrial Wireless HMI Software Module, *Journal of the Korean Institute of Intelligent Systems*, 17(3), 336-342.

Çelik, C., Çakır, S., & Yalçın, N., 2022, Bulut Tabanlı Bir Mesajlaşma Uygulaması Tasarımı ve Gerçekleştirimi, 3rd International Conference on Applied Engineering and Natural Sciences (ICAENS 2022), Temmuz 2022, Konya, Türkiye.

Dalmış, F., Tuğ, S., Dalmış, İ. S., Aktaş, T., & Kayişoğlu, B. (2018). Laboratuvar Tipi Gazlaştırıcılar İçin PLC Tabanlı Prototip Veri Toplama ve Kontrol Sisteminin Geliştirilmesi. *Tekirdağ Ziraat Fakültesi Dergisi*, *15*(1), 143-156.

Deveci F., 2022, Tarihi Haberleşme Metodu: Modbus RTU, https://www.firatdeveci.com/tarihi-haberlesme-metodu-modbus-rtu/ [Son erişim 11.10.2022]

Endüstri40, 2022, Haberleşme Protokollerinde Endüstri 4.0 Devrimi MQTT, https://www.endustri40.com/haberlesme-protokollerinde-endustri-4-0-devrimi-MQTT [Son erişim 11.10.2022]

Ercan, T., & Kutay, M., 2016, Endüstride Nesnelerin İnterneti (IoT) Uygulamaları. Afyon Kocatepe Üniversitesi Fen ve Mühendislik Bilimleri Dergisi, 16(3), 599-607.

Eryılmaz, H., 2015, Mobil Ulaşım Hizmetleri Bilgilendirme Sisteminde Arayüz Tasarımı İncelemesi, Süleyman Demirel *Üniversitesi Mühendislik Bilimleri ve Tasarım Dergisi*, 3(3), 475-479.

Görkem, L., & Bozuklu, M., 2016, Nesnelerin interneti: Yapılan Çalışmalar ve Ülkemizdeki Mevcut Durum, *Gaziosmanpaşa Bilimsel Araştırma Dergisi*, (13), 47-68.

Grunert, K. G., De Bauw, M., Dean, M., Lähteenmäki, L., Maison, D., Pennanen, K., ... & Vranken, L., 2021, No Lockdown in the kitchen: How the COVID-19 Pandemic Has Affected Food-Related Behaviours. *Food Research International*, 150, 110752.

Hintaw, A. J., Manickam, S., Aboalmaaly, M. F., & Karuppayah, S., 2021, MQTT Vulnerabilities, Attack Vectors and Solutions in the Internet of Things (IoT). *IETE Journal of Research*, DOI: 10.1080/03772063.2021.1912651

Innova, 2022, Nesnelerin İnterneti (IoT) Nedir?, https://www.innova.com.tr/tr/blog/dijital-donusum-blog/nesnelerin-interneti-iot-nedir/ [Son erişim 11.10.2022]

Isikdag, U., 2015, *Internet of Things: Single-board Computers, In Enhanced Building Information Models*,pp. 43-53, Springer Cham.

İdas, 2022, Modbus Haberleşme, https://www.idasotomasyon.com/modbus-haberlesme [Son erişim 11.10.22]

Kahraman, Z., Hacı, M., İçibal, K., & Soyhan, H. S., 2018, Enerji Verimliliği Sağlayan Yüksek Hızlı Hibrit Pişirme Fırını Tasarımı ve Prototip İmalatı. *Academic Perspective Procedia*, 1(1), 1338-1346.

Kekre, A. M., & Kothari, A.,2022, MODBUS-TR: Advanced MODBUS-RTU Protocol for IoT with Auto-discovery and Triggers. *Wireless Personal Communications*, 1-12.

Kofler, M., 2010, *Linux 2010: Debian, Fedora, openSUSE, Ubuntu. 9*. Pearson Deutschland GmbH.

Konopek, A., 2002, Designing the Human-Machine Interface in Industrial Design, AIAA *Space Architecture Symposium* (p. 6109).

Krylovskiy, A., 2015, Internet of Things Gateways Meet Linux Containers: Performance Evaluation and Discussion. *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*,pp. 222-227.

Livinsa, Z. M., Valantina, G. M., Premi, M. G., & Sheeba, G. M., 2021, A Modern Automatic Cooking Machine Using Arduino Mega and IoT. In *Journal of Physics*: *Conference Series*, Vol. 1770, No. 1, p. 012027, IOP Publishing.

McCune, R.R., 2011, *Node.js Paradigms and Benchmarks*.

Mozumder, M. J. I., & Ghosh, S., 2018, IoT Based Automatic Electricity Monitoring and Remote Load Control System Using PIC18F4550. In 2018 9th *International Conference on Computing, Communication and Networking Technologies* (ICCCNT), pp. 1-4.

Nayan, A. A., Saha, J., Ferdaous, J., & Kibria, M. G., 2022, IoT Based Smart Kitchen Security System, *International Conference on Big Data, IoT and Machine Learning (BIM)*.

Neto, M. C., Andrade, S. S., & Novais, R. L., 2017, Cross-platform Multimedia Application Development: for Mobile, Web, Embedded and IoT with Qt/QML, *In Proceedings of the 23rd Brazillian Symposium on Multimedia and the Web*, pp. 23-26.

O'farrell, M., Lewis, E., Flanagan, C., Lyons, W. B., & Jackman, N., 2005, Combining Principal Component Analysis with an Artificial Neural Network to Perform Online Quality Assessment Of Food As It Cooks in a Large-Scale Industrial Oven. *Sensors and Actuators B: Chemical*, 107(1), 104-112.

Rational, 2022, i-CombiPro, https://www.rational-online.com/tr_tr/icombi-pro/# [Son erişim 11.10.2022]

Rholam, O., & Tabaa, M., 2019, Smart Device for Multi-Band Industrial IoT Communications. *Procedia computer science*, 155, 660-665.

Santillán, L. A. C., Ginestà, M. G., & Mora, Ó. P., 2014, Bases de Datos en MySQL. Universitat oberta de Catalunya

Song, R., Lang, W. C., & Pan, S. W., 2012, Development of Embedded System GUI Based on Qt/Embedded. In *Applied Mechanics and Materials*, Vol. 109, pp. 586-590, Trans Tech Publications Ltd.

Sun, Z., Zhu, M., & Lee, C., 2021, Progress in the Triboelectric Human–Machine Interfaces (HMIs)-Moving from Smart Gloves to AI/Haptic Enabled HMI in the 5G/IoT Era. *Nanoenergy Adv.* 2021, 1, 81–120.

Umapathi, N., & Sabbani, S., 2022, An Internet of Things (IoT)-based Approach for Real-Time Kitchen Monitoring Using NodeMCU 1.0. In *Futuristic Communication and Network Technologies*, pp. 35-43, Springer, Singapore.

Ward, H. H., 2022, Introducing MPLABX. In *Programming Arduino Projects with the PIC Microcontroller*, pp. 1-25, Apress, Berkeley, CA.

Watson, J., 2008, Virtualbox: Bits and Bytes Masquerading as Machines, *Linux Journal*, 2008(166), 1.

Wikipedia, 2022, HTML, https://en.wikipedia.org/wiki/HTML [Son erişim 11.10.2022]

Wikipedia, 2022, CSS, https://en.wikipedia.org/wiki/CSS [Son erişim 11.10.2022]

Wikipedia, 2022, JavaScript, https://en.wikipedia.org/wiki/JavaScript [Son erişim 11.10.22]

Yaylacı, A.F., 2017, Türkiye Endüstriyel Mutfak Sektörünün Rekabetçilik Analizi ve Rekabet Stratejileri, Yüksek Lisans Tezi, Sakarya Üniversitesi Sosyal Bilimler Enstitüsü, İşletme Anabilim Dalı.

Zhang, Q., Chen, S., Chen, R., Ke, R., Guo, X., & Liang, X., 2021, HMILink: a Zero-Code Programming System for Embedded Screen-Type HMI, 2021 IEEE International Conference on High Performance Big Data and Intelligent Systems (HPBD&IS), pp. 225-235.

EKLER

EK-1
4. ARAŞTIRMA SONUÇLARI VE TARTIŞMA Başlığı Linkleri

- A : https://cdn.kernel.org/pub/linux/kernel/v6.x/linux-6.0.7.tar.xz
- B : https://www.armbian.com/orange-pi-pc-plus/
- C : https://www.balena.io/etcher/
- D : https://www.qt.io/
- E : https://gitlab.freedesktop.org/mesa/mesa
- F : https://www.arduino.cc/en/software
- G : https://www.qt.io/download-open-source?hsCtaTracking=e9c17691-91a0-4616-9bc2-1a6a6c318914%7C963686f8-2c68-442a-b17b-3d73ce95b819

EK-2
Modbus Yazılım Kodları

Modbus Slave

```
uint16_t ModRTU_CRC(uint8_t *buf, uint16_t len);
void ModBusSlave(void);
void Mbus_T1Ctr(void) {
   if (mbusSlave.TX_END_Ctr) {
      mbusSlave.TX_END_Ctr--;
   } else MBS_Tx_ENABLEOff();
   if (mbusSlave.RX_END_Ctr) {
      mbusSlave.RX_END_Ctr--;
      if (!mbusSlave.RX_END_Ctr) {
         mbusSlave.RxSize = 0;
         ModBusSlave();
      }
   }
}
void IsConnectionOK(void) {
//    MBUSTimeoutTimer = MBUSDisconnected = MBUSWarning = 0;
   LEDxToggle;
}
void cleaner(void) {
   int i = 0;
   for (i = 0; i < Mbusbuffer; i++) {
      mbusSlave.RxBuf[i] = 0;
   }

void ModBusSend(uint16_t TxSize) {

   while (!UXSTAbits.TRMT);
   MBS_Tx_ENABLEOn();
   int i = 0;
   for (i = 0; i < TxSize; i++) {
      UARTX_Write(mbusSlave.TxBuf[i]);
   }
   IsConnectionOK();
}
void mbus10com(void) {

   //gelen ilk 6 dataya dokunmuyoruz!
   //7. kacbyte cevap olacagi
   //son iki CRC
   twoBYTE_16u mbusAdr, mbus_numregs, RegX, mbusCRC;
   mbusSlave.TxBuf[0] = mbusSlave.RxBuf[0];
   mbusSlave.TxBuf[1] = mbusSlave.RxBuf[1];
   mbusSlave.TxBuf[2] = mbusAdr.bytes.HB = mbusSlave.RxBuf[2];
   mbusSlave.TxBuf[3] = mbusAdr.bytes.LB = mbusSlave.RxBuf[3];
   mbusSlave.TxBuf[4] = mbus_numregs.bytes.HB = mbusSlave.RxBuf[4];
```

```
    mbusSlave.TxBuf[5] = mbus_numregs.bytes.LB = mbusSlave.RxBuf[5];
    mbusSlave.TxBuf[6] = mbusSlave.TxBuf[7] = mbusSlave.TxBuf[8] = 0;
    char i;
    for (i = 0; i < mbus_numregs.Val; i++) {
      RegX.bytes.HB = mbusSlave.RxBuf[7 + i * 2];
      RegX.bytes.LB = mbusSlave.RxBuf[7 + i * 2 + 1];
      set_slave_vals(i + mbusAdr.Val, RegX.Val);
    }
    mbusCRC.Val = ModRTU_CRC(mbusSlave.RxBuf, 6);
    mbusSlave.TxBuf[6] = mbusCRC.bytes.LB;
    mbusSlave.TxBuf[7] = mbusCRC.bytes.HB;
    ModBusSend(8);
}
void mbus06com(void) {
    //4 ve 5. byte sirasinda single yazilacak veri var
    twoBYTE_16u mbusAdr, mbusVal, mbusCRC;

    mbusSlave.TxBuf[0] = mbusSlave.RxBuf[0];
    mbusSlave.TxBuf[1] = mbusSlave.RxBuf[1];
    mbusSlave.TxBuf[2] = mbusAdr.bytes.HB = mbusSlave.RxBuf[2];
    mbusSlave.TxBuf[3] = mbusAdr.bytes.LB = mbusSlave.RxBuf[3];
    mbusSlave.TxBuf[4] = mbusVal.bytes.HB = mbusSlave.RxBuf[4];
    mbusSlave.TxBuf[5] = mbusVal.bytes.LB = mbusSlave.RxBuf[5];

    mbusCRC.Val = ModRTU_CRC(mbusSlave.RxBuf, 6);
    mbusSlave.TxBuf[6] = mbusCRC.bytes.LB;
    mbusSlave.TxBuf[7] = mbusCRC.bytes.HB;

    mbusSlave.TxBuf[8] = 0;

    set_slave_vals(mbusAdr.Val, mbusVal.Val);
    ModBusSend(8);
}
void mbus03com(void) {
    // bu fonk da fiziksel degerler > BASEvar e kopyalanacak
    // Okuma icin kullaniliyor

    twoBYTE_16u mbusAdr, mbus_numregs, reg_vals, mbusCRC;
    mbusAdr.bytes.HB = mbusSlave.RxBuf[2];
    mbusAdr.bytes.LB = mbusSlave.RxBuf[3];
    mbus_numregs.bytes.HB = mbusSlave.RxBuf[4];
    mbus_numregs.bytes.LB = mbusSlave.RxBuf[5];
    //simdi gidecek degerler
    mbusSlave.TxBuf[0] = mbusSlave.RxBuf[0];
    mbusSlave.TxBuf[1] = mbusSlave.RxBuf[1];
    mbusSlave.TxBuf[2] = (mbus_numregs.Val)*2;
    uint8_t i;
    for (i = 0; i < mbus_numregs.Val; i++) { //0:ID,1:COM,2:numREG 3 den basla
      reg_vals.Val = get_slave_vals(i + mbusAdr.Val); //>>>>>>>>>>>>>>>>>>>
       Custom file
```

```
      mbusSlave.TxBuf[3 + i * 2] = reg_vals.bytes.HB;
      mbusSlave.TxBuf[3 + i * 2 + 1] = reg_vals.bytes.LB;
   }
   mbusCRC.Val = ModRTU_CRC(mbusSlave.TxBuf, 3 + (mbus_numregs.Val)*2);
   mbusSlave.TxBuf[3 + (mbus_numregs.Val)*2] = mbusCRC.bytes.LB;
   mbusSlave.TxBuf[4 + (mbus_numregs.Val)*2] = mbusCRC.bytes.HB;
   ModBusSend(5 + (mbus_numregs.Val)*2);
}
void ModBusSlave(void) {
   twoBYTE_16u ModBusRecCRC;
   ModBusRecCRC.Val = 0;
   if (mbusSlave.RxBuf[0] == DEVICE_ID) {
      switch (mbusSlave.RxBuf[1]) {
        case 0x03:

          ModBusRecCRC.bytes.LB = mbusSlave.RxBuf[6]; //once  LB
          ModBusRecCRC.bytes.HB = mbusSlave.RxBuf[7]; //sonra HB
          if (ModRTU_CRC(mbusSlave.RxBuf, 6) == ModBusRecCRC.Val) { // 8byte
      in 2si crc geri kalaninin crc si hesaplanir
            mbus03com();
          } else {
            //              Beep(si_, 250);
            //                comstate = CRC_Error; // CRC Hatasi
          }
          break;

        case 0x06:
          ModBusRecCRC.bytes.LB = mbusSlave.RxBuf[6]; //once  LB
          ModBusRecCRC.bytes.HB = mbusSlave.RxBuf[7]; //sonra HB
          if (ModRTU_CRC(mbusSlave.RxBuf, 6) == ModBusRecCRC.Val) { // 8byte
      in 2si crc geri kalaninin crc si hesaplanir
            mbus06com(); //single register = 2byte
          } else {
            //              Beep(si_, 250);
            //              comstate = CRC_Error; // CRC Hatasi
          }
          break;

        case 0x10:
          ModBusRecCRC.bytes.LB = mbusSlave.RxBuf[7 + mbusSlave.RxBuf[6]];
      //once  LB
          ModBusRecCRC.bytes.HB = mbusSlave.RxBuf[8 + mbusSlave.RxBuf[6]];
      //sonra HB

          if (ModRTU_CRC(mbusSlave.RxBuf, 9 + mbusSlave.RxBuf[6] - 2) ==
      ModBusRecCRC.Val) {
            mbus10com();
          } else {
            //              Beep(si_, 250);
            //              comstate = CRC_Error; // CRC Hatasi }break;default:break;  }
```

```
    } else {
        return; // adres hatasi
    }
    cleaner();
}
uint16_t ModRTU_CRC(uint8_t *buf, uint16_t len) { //inline uint16_t
        __attribute__((always_inline))
    static const uint16_t wCRCTable[] = {
        0X0000, 0XC0C1, 0XC181, 0X0140, 0XC301, 0X03C0, 0X0280, 0XC241,
        0XC601, 0X06C0, 0X0780, 0XC741, 0X0500, 0XC5C1, 0XC481, 0X0440,
        0XCC01, 0X0CC0, 0X0D80, 0XCD41, 0X0F00, 0XCFC1, 0XCE81, 0X0E40,
        0X0A00, 0XCAC1, 0XCB81, 0X0B40, 0XC901, 0X09C0, 0X0880, 0XC841,
        0XD801, 0X18C0, 0X1980, 0XD941, 0X1B00, 0XDBC1, 0XDA81, 0X1A40,
        0X1E00, 0XDEC1, 0XDF81, 0X1F40, 0XDD01, 0X1DC0, 0X1C80, 0XDC41,
        0X1400, 0XD4C1, 0XD581, 0X1540, 0XD701, 0X17C0, 0X1680, 0XD641,
        0XD201, 0X12C0, 0X1380, 0XD341, 0X1100, 0XD1C1, 0XD081, 0X1040,
        0XF001, 0X30C0, 0X3180, 0XF141, 0X3300, 0XF3C1, 0XF281, 0X3240,
        0X3600, 0XF6C1, 0XF781, 0X3740, 0XF501, 0X35C0, 0X3480, 0XF441,
        0X3C00, 0XFCC1, 0XFD81, 0X3D40, 0XFF01, 0X3FC0, 0X3E80, 0XFE41,
        0XFA01, 0X3AC0, 0X3B80, 0XFB41, 0X3900, 0XF9C1, 0XF881, 0X3840,
        0X2800, 0XE8C1, 0XE981, 0X2940, 0XEB01, 0X2BC0, 0X2A80, 0XEA41,
        0XEE01, 0X2EC0, 0X2F80, 0XEF41, 0X2D00, 0XEDC1, 0XEC81, 0X2C40,
        0XE401, 0X24C0, 0X2580, 0XE541, 0X2700, 0XE7C1, 0XE681, 0X2640,
        0X2200, 0XE2C1, 0XE381, 0X2340, 0XE101, 0X21C0, 0X2080, 0XE041,
        0XA001, 0X60C0, 0X6180, 0XA141, 0X6300, 0XA3C1, 0XA281, 0X6240,
        0X6600, 0XA6C1, 0XA781, 0X6740, 0XA501, 0X65C0, 0X6480, 0XA441,
        0X6C00, 0XACC1, 0XAD81, 0X6D40, 0XAF01, 0X6FC0, 0X6E80, 0XAE41,
        0XAA01, 0X6AC0, 0X6B80, 0XAB41, 0X6900, 0XA9C1, 0XA881, 0X6840,
        0X7800, 0XB8C1, 0XB981, 0X7940, 0XBB01, 0X7BC0, 0X7A80, 0XBA41,
        0XBE01, 0X7EC0, 0X7F80, 0XBF41, 0X7D00, 0XBDC1, 0XBC81, 0X7C40,
        0XB401, 0X74C0, 0X7580, 0XB541, 0X7700, 0XB7C1, 0XB681, 0X7640,
        0X7200, 0XB2C1, 0XB381, 0X7340, 0XB101, 0X71C0, 0X7080, 0XB041,
        0X5000, 0X90C1, 0X9181, 0X5140, 0X9301, 0X53C0, 0X5280, 0X9241,
        0X9601, 0X56C0, 0X5780, 0X9741, 0X5500, 0X95C1, 0X9481, 0X5440,
        0X9C01, 0X5CC0, 0X5D80, 0X9D41, 0X5F00, 0X9FC1, 0X9E81, 0X5E40,
        0X5A00, 0X9AC1, 0X9B81, 0X5B40, 0X9901, 0X59C0, 0X5880, 0X9841,
        0X8801, 0X48C0, 0X4980, 0X8941, 0X4B00, 0X8BC1, 0X8A81, 0X4A40,
        0X4E00, 0X8EC1, 0X8F81, 0X4F40, 0X8D01, 0X4DC0, 0X4C80, 0X8C41,
        0X4400, 0X84C1, 0X8581, 0X4540, 0X8701, 0X47C0, 0X4680, 0X8641,
        0X8201, 0X42C0, 0X4380, 0X8341, 0X4100, 0X81C1, 0X8081, 0X4040
    };
    uint8_t nTemp;
    uint16_t wCRCWord = 0xFFFF;
    while (len--) {
        nTemp = *buf++ ^ wCRCWord;
        wCRCWord >>= 8;
        wCRCWord ^= wCRCTable[nTemp]; }
    return wCRCWord;
}
```

Mini PC tarafı Modbus Master (Modbusbridge.cpp)

```
#include "app/connector/connector.h"
#include <app/settings/managementsettings.h>
#include <QList>
#include <tslibutilities.h>
ModbusBridge::ModbusBridge(QObject *parent) : QObject(parent)
  ,m_modbusThread()
  ,m_modbus(ModBus::instance())
  ,m_invokeFlowmeter(true)
  ,m_invokeTcType(true)
  ,m_invokeKlepe(false)
  ,m_invokeTahliye(false)
  ,m_damperState(0)
  ,m_drainState(0)
  ,m_humState(0)
  ,m_invokeJetBuhar(false)
  ,humCounter(0)
  ,steamTCounter(0)
  ,roomCounter(0)
  ,inputTimer(0)
  ,fanSpeed(0)
  ,fanCommand(0)
  ,m_eta(0)
  ,fanAcc(50)
  ,fanDecc(50)
  ,fanBra(1)
  ,fanSTT(0)
  ,fanLSP(0)
  ,fanHSP(500)
  ,fanNPR(45)
  ,fanNCR(19)
  ,m_invokeFanSettings(true)
  ,m_invokeReadFanParameters(true)
  ,m_invokeHumidity(false)
  ,m_invokeRGB(true)
  ,m_invokeRafLed(false)
  ,m_debugger(Debugger::instance()) {
    qDebug()<<"**********************ModbusBridge
Constructor**********************";
    initModbus();}
//modbusbridge başlat ve cihaz açıldığında init edilmesi gerekenleri init et.
void ModbusBridge::start() {
    initValues();
    m_modbusThread.start(QThread::HighestPriority); }
void ModbusBridge::stop() {
    m_pollingTimer->stop();
    m_modbusThread.exit(); }
void ModbusBridge::initModbus() {
    m_modbus->moveToThread(&m_modbusThread);
```

```
    connect(&m_modbusThread, &QThread::started, m_modbus,
&ModBus::connectDevice);
    connect(&m_modbusThread, &QThread::finished, m_modbus,
&ModBus::disconnectDevice);
    connect(m_modbus, &ModBus::connected, this,
&ModbusBridge::onModbusConnected);
    connect(m_modbus ,&ModBus::disconnected, this,
&ModbusBridge::onModbusDisconnected);
    connect(m_modbus,&ModBus::modbusError, this,
&ModbusBridge::onModbusError);
    connect(m_modbus,&ModBus::response, this,
&ModbusBridge::onModbusResponse);
    connect(m_modbus,&ModBus::stateChanged, this,
&ModbusBridge::onModbusStateChanged);
    m_pollingTimer = new QTimer(nullptr);
    m_pollingTimer->setInterval(100);
    m_pollingTimer->connect(&m_modbusThread, SIGNAL(started()), SLOT(start()));
    connect(m_pollingTimer, &QTimer::timeout, this,
&ModbusBridge::onMbPollingTimeout);
    connect(this, &ModbusBridge::readModbusRegister,m_modbus,
&ModBus::readRegister);
    connect(this, &ModbusBridge::writeModbusRegister,m_modbus,
&ModBus::writeRegister);
    drain220vac = Drain::instance()->is220VACACTIVE();
    connect(Drain::instance(), &Drain::is220VACACTIVEChanged, this, [=](bool
state){
        drain220vac = state;  }); }
//modbus durumu değiştiğinde
void ModbusBridge::onModbusStateChanged(QModbusDevice::State newState) {
    Q_UNUSED(newState);
    //qDebug()<<"ModbusState:"<<newState; }
//modbusdan gelen cevap
void ModbusBridge::onModbusResponse(const QModbusDataUnit &data, const int
&pSlaveAddr) {
    bool modbusComOK= !(relayComOK && (fanComOK ||
ManagementSettings::instance()->getDemoMode()));
    switch (pSlaveAddr) {
    case 1: {
        CookingSettings *cookingSettings = CookingSettings::instance();
        const qint16 roomTempOffset = cookingSettings->getRoomTempOffset() * 10;
        const qint16 probeTempOffset = cookingSettings->getProbeTempOffset() * 10;
        const qint16 steamHolderTempOffset = cookingSettings-
>getSteamHolderTempOffset() * 10;
        const qint16 shockCoolingTempOffset = cookingSettings->getShockTempOffset()
* 10;
        relayComOK = true;
        const quint16 &damperI            =
data.value(ModbusConstants::ModbusRegisters::MB_REG_DI_DAMPER);
        const quint16 &coils              =
data.value(ModbusConstants::ModbusRegisters::MB_REG_DI_RELAY);
```

```
        const quint16 &coils1              =
data.value(ModbusConstants::ModbusRegisters::MB_REG_DI_RELAY1);
        const quint16 &outputs             =
data.value(ModbusConstants::ModbusRegisters::MB_REG_DI_OUTPUTS);
        const quint16 &anout1              =
data.value(ModbusConstants::ModbusRegisters::MB_REG_AN_OUTPUTS_AO1);
        const quint16 &drain               =
data.value(ModbusConstants::ModbusRegisters::MB_REG_DRAIN);
        const quint16 &flowmeterLB         =
data.value(ModbusConstants::ModbusRegisters::MB_REG_FLOWMETERLB);
        const quint16 &flowmeterHB         =
data.value(ModbusConstants::ModbusRegisters::MB_REG_FLOWMETERHB);
        const quint16 &digitalInputs       =
data.value(ModbusConstants::ModbusRegisters::MB_REG_INPUTS);
        const quint16 &alarms              =
data.value(ModbusConstants::ModbusRegisters::MB_REG_ALARMS);
        const quint16 &tempHum             =
data.value(ModbusConstants::ModbusRegisters::MB_REG_AN_TEMP_HUM);
        const qint16 &tc1                  =
data.value(ModbusConstants::ModbusRegisters::MB_REG_AN_TC1) +
roomTempOffset;
        const qint16 &tc2                  =
data.value(ModbusConstants::ModbusRegisters::MB_REG_AN_TC2) +
steamHolderTempOffset;
        const qint16 &tc3                  =
data.value(ModbusConstants::ModbusRegisters::MB_REG_AN_TC3) +
shockCoolingTempOffset;
        const qint16 &tc4                  =
data.value(ModbusConstants::ModbusRegisters::MB_REG_AN_TC4);
        const qint16 &tc5                  =
data.value(ModbusConstants::ModbusRegisters::MB_REG_AN_TC5) +
probeTempOffset;
        const qint16 &tc6                  =
data.value(ModbusConstants::ModbusRegisters::MB_REG_AN_TC6) +
probeTempOffset;
        const qint16 &tc7                  =
data.value(ModbusConstants::ModbusRegisters::MB_REG_AN_TC7) +
probeTempOffset;
        const qint16 &tc8                  =
data.value(ModbusConstants::ModbusRegisters::MB_REG_AN_TC8) +
probeTempOffset;
        const qint16 &tc9                  =
data.value(ModbusConstants::ModbusRegisters::MB_REG_AN_TC9) +
probeTempOffset;
        const qint16 &tc10                 =
data.value(ModbusConstants::ModbusRegisters::MB_REG_AN_TC10) +
probeTempOffset;
        const qint16 &pcbNtc               =
data.value(ModbusConstants::ModbusRegisters::MB_REG_PCB_NTC);
```

```
        const quint16 &hz                =
data.value(ModbusConstants::ModbusRegisters::MB_REG_HZ);
        const quint16 &version           =
data.value(ModbusConstants::ModbusRegisters::MB_REG_VERSION);
        const quint16 &testVal           =
data.value(ModbusConstants::ModbusRegisters::TEST);
        const quint16 &nemSensor         =
data.value(ModbusConstants::ModbusRegisters::MB_REG_HUMIDITYSENSOR);
        const quint16 &verificationVersion  =
data.value(ModbusConstants::ModbusRegisters::VERIFICATION_VERSION);
        emit sendHumDebug(testVal, tempHum);
        emit sendHumDebug2({ data.value(33), data.value(34), data.value(35),
data.value(36),
data.value(37), data.value(38), data.value(39), data.value(40)});
        if(ModbusBridge::version != version){
            ModbusBridge::version = version;
            emit versionChanged(); }
        Q_UNUSED(outputs)
        Q_UNUSED(anout1)
        Q_UNUSED(testVal)
        setJetClappeDrainOnTime(coils, coils1);
        if(verificationVersion == 1024) {
            static quint8 errorCount = 0;
            static quint8 count = 0;
            quint16 vv1 = ((quint32)verificationValue * 2347 + 3333) % 1024;
            quint16 vv2 = ((quint32)verificationValue * 1789 - 1754) % 1024;
            const quint16 &verificationValue1   =
data.value(ModbusConstants::ModbusRegisters::VERIFICATION_VALUE_1);
            const quint16 &verificationValue2   =
data.value(ModbusConstants::ModbusRegisters::VERIFICATION_VALUE_2);
            count++;
            if(vv1 == verificationValue1 && vv2 == verificationValue2){
                errorCount = 0;
                if(count > 36){
                    verificationValue = rand();
                    invokeVerification = true;
                    count = 0; }
            }else
                errorCount++;
            if(errorCount > 10)
                stop(); }
#if QT_NO_DEBUG
        getSensors(ModbusConstants::Sensors::PcbNTC,pcbNtc);
        getSensors(ModbusConstants::Sensors::ROOM, tc1);
        getSensors(ModbusConstants::Sensors::BOILER,tc2);
        getSensors(ModbusConstants::Sensors::SHOCKTC,tc3);
        getSensors(ModbusConstants::Sensors::TC_4, tc4);
        getSensors(ModbusConstants::Sensors::PROBE1,tc5);
        getSensors(ModbusConstants::Sensors::PROBE2,tc6);
        getSensors(ModbusConstants::Sensors::PROBE3,tc7);
```

```
        getSensors(ModbusConstants::Sensors::PROBE4,tc8);
        getSensors(ModbusConstants::Sensors::PROBE5,tc9);
        getSensors(ModbusConstants::Sensors::PROBE6,tc10);
        getSensors(ModbusConstants::Sensors::FLOWMETER_LB, flowmeterLB);
        getSensors(ModbusConstants::Sensors::FLOWMETER_HB, flowmeterHB);
#else
        //      test += 3;
        test = 650;
        if(test > 3000)
           test = 300;
        getSensors(ModbusConstants::Sensors::PcbNTC,test);
        getSensors(ModbusConstants::Sensors::ROOM, test);
        getSensors(ModbusConstants::Sensors::BOILER, test);
        getSensors(ModbusConstants::Sensors::SHOCKTC, test);
        getSensors(ModbusConstants::Sensors::TC_4, test);
        getSensors(ModbusConstants::Sensors::PROBE1,test);
        getSensors(ModbusConstants::Sensors::PROBE2,test);
        getSensors(ModbusConstants::Sensors::PROBE3,test);
        getSensors(ModbusConstants::Sensors::PROBE4,test);
        getSensors(ModbusConstants::Sensors::PROBE5,test);
        getSensors(ModbusConstants::Sensors::PROBE6,test);
        getSensors(ModbusConstants::Sensors::FLOWMETER_LB, test);
        getSensors(ModbusConstants::Sensors::FLOWMETER_HB, test);
#endif
        emit calculateMeat();
        getSensors(ModbusConstants::Sensors::HUMIDITY, tempHum);
        getAlarms(alarms);
        getInputs(digitalInputs);
        getDamperState(damperI);
        getDrainState(drain);
        getHumidityInputs(nemSensor);
        emit sendPhaseHz(hz);
        emit sendFlowmeterHBLB(flowmeterHB, flowmeterLB);
        m_debugger->setAlarmsFromModbus(alarms);
        m_debugger->setInputsFromModbus(digitalInputs);
        m_debugger->setRelays1FromModbus(coils);
        m_debugger->setRelays2FromModbus(coils1);
        m_debugger->setDOsFromModbus(outputs);
        m_debugger->setVersionRelay(version);
        if(modbusComOK && (fanComOK ||  ManagementSettings::instance()-
>getDemoMode())){
           emit modbusConnectionStateChanged();
           emit modbusConnected();
           Debugger::instance()->setModbusError(false);}
        break;}
    case 2: {
        fanComOK = true;

        ModbusBridge::ModbusDataAdressForFan startAdd =
static_cast<ModbusBridge::ModbusDataAdressForFan>(data.startAddress());
```

```
        switch (startAdd) {
        case ModbusBridge::LowSpeed:
            // qDebug()<<"LSP"<<data.values();
            break;
        case ModbusBridge::HighSpeed:
            //qDebug()<<"HSP"<<data.values();
            break;
        case ModbusBridge::BrakingFunctionAssignment_BRA:
            //qDebug()<<"BRA"<<data.values();
            break;
        case ModbusBridge::StopMode_STT:
            //qDebug()<<"StopMode_STT"<<data.values();
            break;
        case ModbusBridge::NominalMotorPower_NPR:
            //qDebug()<<"NPR"<<data.values();
            break;
        case ModbusBridge::NominalMotorVoltage_UNS: //başlangıc adresi burası ve
sonraki 4'lü aynı vector içerisinde
        case ModbusBridge::NominalMotorFrequency_FRS:
        case ModbusBridge::NominalMotorCurrent_NCR:
        case ModbusBridge::NominalMotorSpeed_NSP:
            //qDebug()<<"UNS FRS NCR NSP"<<data.values();
            break;
        case ModbusBridge::ControlWordFrequency_CMD:
            //qDebug()<<"CMD"<<data.values();
            break;
        case ModbusBridge::StatusWordFrequency_ETA: {
            //qDebug()<<data.values()<<data.startAddress();
            const quint16 &eta = data.value(0);
            getEta(eta);
            break; }
        case ModbusBridge::AccDeccStatus_ACC_DECC:
            //qDebug()<<"ACC DECC"<<data.values();
            break;}
        if(modbusComOK && relayComOK){
            //qDebug() << "Modbus Connected";
            emit modbusConnectionStateChanged();
            emit modbusConnected();
            Debugger::instance()->setModbusError(false);
            Debugger::instance()->setFanComErr(false);}
        break; }} }
//modbus hata verdiğinde
void ModbusBridge::onModbusError(QModbusDevice::Error error, const int
&slaveAddr) {
        if(error == QModbusDevice::TimeoutError){
            relayComOK = false;
            fanComOK = false;

            emit modbusConnectionStateChanged();
            emit modbusError(ModbusConstants::ERR_MODBUS_CONNECTION);
```

```
            if(slaveAddr == 1)
                Debugger::instance()->setModbusError(true);
            else if(slaveAddr == 2)
                Debugger::instance()->setFanComErr(true); } } }
 //modbus bağlandığında
void ModbusBridge::onModbusConnected() {
    //    m_pollingTimer->start();
    //emit modbusConnected(); }
//modbus bağlantısı koptuğunda
void ModbusBridge::onModbusDisconnected() { }
//her bir sorgu bittiğinde
void ModbusBridge::onMbPollingTimeout() {
    static bool toggle = true;
    static TsLibUtilities* tsLibUtils = TsLibUtilities::instance();
    //Tc tipi için.
    //    0 = J type Termocupl
    //    1 = K type Termocupl
    //    2 = PT100
    //    3 = PT1000
    bool demoMode = ManagementSettings::instance()->getDemoMode();
    if(toggle) {
        toggle = !toggle;
#if QT_NO_DEBUG
        if(!demoMode){
            emit
writeModbusRegister(QModbusDataUnit(QModbusDataUnit::HoldingRegisters,
ModbusBridge::ControlWordFrequency_CMD, QVector<quint16> { fanCommand,
fanSpeed }), ModbusConstants::FanEsc);
            if(m_invokeFanSettings){
                m_invokeFanSettings = false;
                emit
writeModbusRegister(QModbusDataUnit(QModbusDataUnit::HoldingRegisters,
ModbusBridge::AccDeccStatus_ACC_DECC, QVector<quint16> { fanAcc, fanDecc
}), ModbusConstants::FanEsc);
                emit
writeModbusRegister(QModbusDataUnit(QModbusDataUnit::HoldingRegisters,
ModbusBridge::BrakingFunctionAssignment_BRA, QVector<quint16> { fanBra }),
ModbusConstants::FanEsc);
                emit
writeModbusRegister(QModbusDataUnit(QModbusDataUnit::HoldingRegisters,
ModbusBridge::StopMode_STT, QVector<quint16> { fanSTT }),
ModbusConstants::FanEsc);
                emit
writeModbusRegister(QModbusDataUnit(QModbusDataUnit::HoldingRegisters,
ModbusBridge::LowSpeed, QVector<quint16> { fanLSP }),
ModbusConstants::FanEsc);
                emit
writeModbusRegister(QModbusDataUnit(QModbusDataUnit::HoldingRegisters,
ModbusBridge::HighSpeed, QVector<quint16> { fanHSP }),
ModbusConstants::Slaves::FanEsc);
```

```
            emit
writeModbusRegister(QModbusDataUnit(QModbusDataUnit::HoldingRegisters,
ModbusBridge::NominalMotorPower_NPR, QVector<quint16> { fanNPR }),
ModbusConstants::FanEsc);
            emit
writeModbusRegister(QModbusDataUnit(QModbusDataUnit::HoldingRegisters,
ModbusBridge::NominalMotorCurrent_NCR, QVector<quint16> { fanNCR
}),ModbusConstants::FanEsc);   }   }
#endif
      if(!demoMode){
         if(m_invokeFlowmeter){
            m_invokeFlowmeter = false;
            emit
writeModbusRegister(QModbusDataUnit(QModbusDataUnit::HoldingRegisters, 8,
QVector<quint16> { flowmeter.bits.FlowmeterLB, flowmeter.bits.FlowmeterHB }),
ModbusConstants::RelayBoard);   }
         if(m_invokeTcType){
            m_invokeTcType = false;
            emit
writeModbusRegister(QModbusDataUnit(QModbusDataUnit::HoldingRegisters, 3,
QVector<quint16> { outPuts.Val }), ModbusConstants::RelayBoard);   }
         if(m_invokeKlepe){
            m_invokeKlepe = false;
            emit
writeModbusRegister(QModbusDataUnit(QModbusDataUnit::HoldingRegisters, 0,
QVector<quint16> { damper.Val }), ModbusConstants::RelayBoard);    }
         if(m_invokeTahliye){
            m_invokeTahliye = false;
            emit
writeModbusRegister(QModbusDataUnit(QModbusDataUnit::HoldingRegisters, 7,
QVector<quint16> { tahliye.Val }), ModbusConstants::RelayBoard);    }
         if(m_invokeJetBuhar){
            m_invokeJetBuhar = false;
            emit
writeModbusRegister(QModbusDataUnit(QModbusDataUnit::HoldingRegisters, 27,
QVector<quint16> { jetTime }), ModbusConstants::RelayBoard);   }   }
      if(m_invokeHumidity){
         m_invokeHumidity = false;
         emit
writeModbusRegister(QModbusDataUnit(QModbusDataUnit::HoldingRegisters, 28,
QVector<quint16> { humWrite.Val }),ModbusConstants::RelayBoard);   }
      if(m_invokeRGB){
         emit
writeModbusRegister(QModbusDataUnit(QModbusDataUnit::HoldingRegisters, 41,
QVector<quint16> { rgbProfileData.registers[0] }),
ModbusConstants::Slaves::RelayBoard);
         emit
writeModbusRegister(QModbusDataUnit(QModbusDataUnit::HoldingRegisters, 42,
QVector<quint16> { rgbProfileData.registers[1] }),
ModbusConstants::Slaves::RelayBoard);
```

```
        emit
writeModbusRegister(QModbusDataUnit(QModbusDataUnit::HoldingRegisters, 43,
QVector<quint16> { rgbProfileData.registers[2] }),
ModbusConstants::Slaves::RelayBoard);
        emit
writeModbusRegister(QModbusDataUnit(QModbusDataUnit::HoldingRegisters, 44,
QVector<quint16> { rgbProfileData.registers[3] }),
ModbusConstants::Slaves::RelayBoard);
        m_invokeRGB = false;    }
      if(m_invokeRafLed){
        m_invokeRafLed = false;
        emit
writeModbusRegister(QModbusDataUnit(QModbusDataUnit::HoldingRegisters, 32,
QVector<quint16> { rafLedKontrol.Val }), ModbusConstants::RelayBoard);      }
      if(invokeVerification){
        invokeVerification = false;
        emit
writeModbusRegister(QModbusDataUnit(QModbusDataUnit::HoldingRegisters, 29,
QVector<quint16> { verificationValue }), ModbusConstants::RelayBoard);   }
      if(demoMode){
        //röle ve ssr için.
        QVector<quint16> values = {
          uint16_t(0),
          uint16_t(outRelays[1].Val & uint16_t(496)),
          uint16_t(0),
          uint16_t(0),
          uint16_t(0),
          uint16_t(0),
        };
        const QModbusDataUnit &dataUnit =
QModbusDataUnit(QModbusDataUnit::HoldingRegisters, 1, values);
        emit writeModbusRegister(dataUnit,ModbusConstants::Slaves::RelayBoard);
      }else {
        static uint16_t relays1 = 0;
        static uint16_t relays2 = 0;
        //röle ve ssr için.
        QVector<quint16> values = {
          outRelays[0].Val,
          outRelays[1].Val,
          outPuts.Val,
          AnOut1.Val,
          IAnOut2.Val,
          IAnOut1.Val,  };
        if(relays1 != outRelays[0].Val || relays2 != outRelays[1].Val)
          tsLibUtils->disable();
        const QModbusDataUnit &dataUnit =
QModbusDataUnit(QModbusDataUnit::HoldingRegisters, 1, values);
        emit writeModbusRegister(dataUnit,ModbusConstants::Slaves::RelayBoard);
        relays1 = outRelays[0].Val;
        relays2 = outRelays[1].Val; }
```

```
    } else {
        if(!tsLibUtils->isEnabled())
            tsLibUtils->enable();
        toggle = !toggle;
        // girişleri oku
        const QModbusDataUnit dataUnit(QModbusDataUnit::HoldingRegisters, 0, 41);
        emit readModbusRegister(dataUnit, ModbusConstants::Slaves::RelayBoard);
#if QT_NO_DEBUG
        if(!demoMode){
            const QModbusDataUnit
dataUnitFan(QModbusDataUnit::HoldingRegisters,ModbusBridge::ModbusDataAdress
ForFan::StatusWordFrequency_ETA, 4);
            emit readModbusRegister(dataUnitFan,ModbusConstants::Slaves::FanEsc);
            if(m_invokeReadFanParameters == true)  {
                m_invokeReadFanParameters = false;
                const QModbusDataUnit
dataUnitFanACCDECC(QModbusDataUnit::HoldingRegisters,ModbusBridge::Modbus
DataAdressForFan::AccDeccStatus_ACC_DECC,2);
                emit
readModbusRegister(dataUnitFanACCDECC,ModbusConstants::Slaves::FanEsc);
                 const QModbusDataUnit
dataUnitNPR(QModbusDataUnit::HoldingRegisters,ModbusBridge::ModbusDataAdres
sForFan::NominalMotorPower_NPR,1);
                emit readModbusRegister(dataUnitNPR,ModbusConstants::Slaves::FanEsc);
                const QModbusDataUnit
dataUnitNominals(QModbusDataUnit::HoldingRegisters,ModbusBridge::ModbusData
AdressForFan::NominalMotorVoltage_UNS,4);
                emit
readModbusRegister(dataUnitNominals,ModbusConstants::Slaves::FanEsc);
                const QModbusDataUnit
dataUnitBraking(QModbusDataUnit::HoldingRegisters,ModbusBridge::ModbusDataAd
ressForFan::BrakingFunctionAssignment_BRA,1);
                emit
readModbusRegister(dataUnitBraking,ModbusConstants::Slaves::FanEsc);
                const QModbusDataUnit
dataUnitStopMode(QModbusDataUnit::HoldingRegisters,ModbusBridge::ModbusData
AdressForFan::StopMode_STT,1);
                emit
readModbusRegister(dataUnitStopMode,ModbusConstants::Slaves::FanEsc);
                const QModbusDataUnit
dataUnitLsp(QModbusDataUnit::HoldingRegisters,ModbusBridge::ModbusDataAdress
ForFan::LowSpeed,1);
                emit readModbusRegister(dataUnitLsp,ModbusConstants::Slaves::FanEsc);
                const QModbusDataUnit
dataUnitHsp(QModbusDataUnit::HoldingRegisters,ModbusBridge::ModbusDataAdress
ForFan::HighSpeed,1);
                emit
readModbusRegister(dataUnitHsp,ModbusConstants::Slaves::FanEsc);   }   }
#endif   } }
//init edilen değerler.
```

```
void ModbusBridge::initValues() {
    alarms.Val = 0;
    Inputs.Val = 1;
    outRelays[0].Val= 0;
    outRelays[1].Val= 0;
    IAnOut1.Val = 0;
    IAnOut2.Val = 0;
    sensor.TC_1 = 0;
    sensor.TC_2 = 0;
    sensor.TC_3 = 0;
    sensor.TC_4 = 0;
    sensor.TC_5 = 0;
    sensor.TC_6 = 0;
    sensor.TC_7 = 0;
    sensor.TC_8 = 0;
    sensor.TC_9 = 0;
    sensor.TC_10 = 0;
    sensor.PcbNTC = 0;
    outPuts.Val = 0;
    jetTime = 0;
    setOutputs(ModbusConstants::TCTYPE, 1);
    //    0 = J type Termocupl
    //    1 = K type Termocupl
    //    2 = PT100
    //    3 = PT1000
    //yetki röle kartında.
    damper.bits.Test = 0;
    tahliye.bits.Test = 0;
    //yetki opide test için.
    setFanAcc(50);
    setFanDecc(50);
    setFanLSP(0);
    setFanHSP(500);
    setFanNPR(45);
    setFanNCR(19);
    setFanBra(1);
    setFanSTT(0); }
//alarm değerlerini modbusdan al ve gönder
void ModbusBridge::getAlarms(const quint16 _alarmsVal) {
    if(alarms.Val == _alarmsVal)
        return;
    alarms.Val = _alarmsVal;
    emit sendAlarms(_alarmsVal); }
//inputları modbusdan al ve gönder
void ModbusBridge::getInputs(const quint16 _inputsVal) {
    if(Inputs.Val == _inputsVal && inputTimer < 50){
        inputTimer++;
        return; }
    inputTimer = 0;
    Inputs.Val = _inputsVal;
```

```
    emit sendInputs(_inputsVal); }
void ModbusBridge::getDamperState(const quint16 _state) {
    if(m_damperState == _state)
        return;
    m_damperState = _state;
    emit sendDamperState(_state); }
void ModbusBridge::getDrainState(const quint16 _state) {
    if(m_drainState == _state)
        return;
    m_drainState = _state;
    emit sendDrainState(_state); }
//sensör değerlerini modbusdan al ve gönder
void ModbusBridge::getSensors(const ModbusConstants::Sensors _temp, const qint16
_tempsVal) {
    switch (_temp) {
    case ModbusConstants::Sensors::PcbNTC: {
        if(sensor.PcbNTC == _tempsVal)
            return;
        sensor.PcbNTC = _tempsVal;
        emit sendPcbNtc(_tempsVal);
        break;  }
    case ModbusConstants::Sensors::ROOM: {
        if(sensor.TC_1 == _tempsVal && roomCounter<50)   {
            roomCounter++;
            return; }
        roomCounter = 0;
        sensor.TC_1 = _tempsVal;
        emit sendTC_1(_tempsVal);
        break;  }
    case ModbusConstants::Sensors::BOILER: {
        if(sensor.TC_2 == _tempsVal && steamTCounter<50)    {
            steamTCounter++;
            return;   }
        steamTCounter = 0;
        sensor.TC_2 = _tempsVal;
        emit sendTC_2(_tempsVal);
        break;   }
    case ModbusConstants::Sensors::SHOCKTC: {
        if(sensor.TC_3 == _tempsVal)
            return;
        sensor.TC_3 = _tempsVal;
        emit sendTC_3(_tempsVal);
        break;   }
    case ModbusConstants::Sensors::TC_4: {
        if(sensor.TC_4 == _tempsVal)
            return;
        sensor.TC_4 = _tempsVal;
        emit sendTC_4(_tempsVal);
      break;   }
    case ModbusConstants::Sensors::PROBE1: {
```

```
        //      if(sensor.TC_5 == _tempsVal)
        //          return;
        sensor.TC_5 = _tempsVal;
        emit sendTC_5(_tempsVal);
        break;   }
    case ModbusConstants::Sensors::PROBE2:   {
        //      if(sensor.TC_6 == _tempsVal)
        //          return;
        sensor.TC_6 = _tempsVal;
        emit sendTC_6(_tempsVal);
        break;   }
    case ModbusConstants::Sensors::PROBE3:     {
        //      if(sensor.TC_7 == _tempsVal)
        //          return;
        sensor.TC_7 = _tempsVal;
        emit sendTC_7(_tempsVal);
        break;   }
    case ModbusConstants::Sensors::PROBE4: {
        //      if(sensor.TC_8 == _tempsVal)
        //          return;
        sensor.TC_8 = _tempsVal;
        emit sendTC_8(_tempsVal);    }
    case ModbusConstants::Sensors::PROBE5:    {
        //      if(sensor.TC_9 == _tempsVal)
        //          return;
        sensor.TC_9 = _tempsVal;
        emit sendTC_9(_tempsVal); }
    case ModbusConstants::Sensors::PROBE6:    {
        //      if(sensor.TC_10 == _tempsVal)
        //          return;
        sensor.TC_10 = _tempsVal;
        emit sendTC_10(_tempsVal);
        break; }
    case ModbusConstants::Sensors::FLOWMETER_LB: {
        quint16 correctionType = (quint16)_tempsVal;
        if(flowmeter.bits.FlowmeterLB == correctionType)
            return;
        flowmeter.bits.FlowmeterLB = correctionType;
        //      emit
sendFlowmeterHBLB(flowmeter.bits.FlowmeterHB,flowmeter.bits.FlowmeterLB);
        break;   }
    case ModbusConstants::Sensors::FLOWMETER_HB: {
        if(flowmeter.bits.FlowmeterHB == _tempsVal)
            return;
        flowmeter.bits.FlowmeterHB = _tempsVal;
        //      emit
sendFlowmeterHBLB(flowmeter.bits.FlowmeterHB,flowmeter.bits.FlowmeterLB);
        break; }
    case ModbusConstants::Sensors::HUMIDITY:   {
        if(sensor.Humidity == _tempsVal && humCounter < 20) {
```

```
            humCounter++;
            return;     }
        humCounter = 0;
        sensor.Humidity = _tempsVal;
        emit sendHumidity(_tempsVal);
        break;   } } }
void ModbusBridge::getHumidityInputs(const quint16 val) {
    if(m_humState == val)
        return;
    m_humState = val;
    emit sendHumState(val);}
//Çıkış değerlerini diğer klasslardan al ve modbus'a gönder
void ModbusBridge::setOutputs(const ModbusConstants::Outputs _outPut, const bool
_openClose) {
    switch (_outPut) {
    case ModbusConstants::PADDLEBOX1:
        outRelays[1].bits.Role6 = _openClose;
        break;
    case ModbusConstants::PADDLEBOX2:
        outRelays[1].bits.Role7 = _openClose;
        break;
    case ModbusConstants::ROOMLAMP:
        outRelays[1].bits.Role5 = _openClose;
        break;
    case ModbusConstants::DRAINBROWN:
        outRelays[1].bits.Role3 = _openClose;
        break;
    case ModbusConstants::DRAINBLACK:
        outRelays[1].bits.Role4 = _openClose;
        break;
    case ModbusConstants::LAMP1:
        rafLedKontrol.bits.Raf1 = _openClose;
        m_invokeRafLed = true;
        break;
    case ModbusConstants::LAMP2:
        rafLedKontrol.bits.Raf2 = _openClose;
        m_invokeRafLed = true;
        break;
    case ModbusConstants::LAMP3:
        rafLedKontrol.bits.Raf3 = _openClose;
        m_invokeRafLed = true;
        break;
    case ModbusConstants::LAMP4:
        rafLedKontrol.bits.Raf4 = _openClose;
        m_invokeRafLed = true;
        break;
    case ModbusConstants::LAMP5:
        rafLedKontrol.bits.Raf5 = _openClose;
        m_invokeRafLed = true;
        break;
```

```
case ModbusConstants::LAMP6:
    rafLedKontrol.bits.Raf6 = _openClose;
    m_invokeRafLed = true;
    break;
case ModbusConstants::LAMP7:
    rafLedKontrol.bits.Raf7 = _openClose;
    m_invokeRafLed = true;
    break;
case ModbusConstants::LAMP8:
    rafLedKontrol.bits.Raf8 = _openClose;
    m_invokeRafLed = true;
    break;
case ModbusConstants::LAMP9:
    rafLedKontrol.bits.Raf9 = _openClose;
    m_invokeRafLed = true;
    break;
case ModbusConstants::LAMP10:
    rafLedKontrol.bits.Raf10 = _openClose;
    m_invokeRafLed = true;
    break;
case ModbusConstants::DRAINDIRECTION://ok 1 10
    //outRelays[1].bits.Role10 = _openClose;
    break;
case ModbusConstants::LOGOLIGHTING://ok 1 8
    outRelays[1].bits.Role8 = _openClose;
    break;
case ModbusConstants::MAINTENANCEVALVE: //ok 0 1
    outRelays[0].bits.Role1 = _openClose;
    break;
case ModbusConstants::MAINTENANCEPOMP: //ok 0 6
    outRelays[0].bits.Role6 = _openClose;
    break;
case ModbusConstants::CYCLEPOMP: //ok 0 8
    outRelays[0].bits.Role8 = _openClose;
    break;
case ModbusConstants::POMPCLEANINGVALVE: //ok 0 5
    outRelays[0].bits.Role5 = _openClose;
    break;
case ModbusConstants::DETERGENTPOMP: // ok 0 10
    outRelays[0].bits.Role10 = _openClose;
    break;
case ModbusConstants::POLISHERPOMP: //ok 0 9
    outRelays[0].bits.Role9 = _openClose;
    break;
case ModbusConstants::DAMPER: //ok 1 1
    //      outRelays[0].bits.Role6 = _openClose;       //CTEST
    //      outRelays[0].bits.Role7 = _openClose;       //CTEST
    //outRelays[1].bits.Role1 = _openClose;
    break;
case ModbusConstants::DRAIN: //ok 1 11
```

```
        //outRelays[1].bits.Role11 = _openClose;
        break;
    case ModbusConstants::VALVE: // ok 0 4
        outRelays[0].bits.Role4 = _openClose;
        break;
    case ModbusConstants::HEATERSSR1: //ok
        outPuts.bits.DO1 = _openClose;
        break;
    case ModbusConstants::HEATERSSR2: //ok
        outPuts.bits.DO2 = _openClose;
        break;
    case ModbusConstants::STEAMSSR1://ok
        outPuts.bits.DO3 = _openClose;
        break;
    case ModbusConstants::STEAMSSR2://ok
        outPuts.bits.DO4 = _openClose;
        break;
    case ModbusConstants::JET://ok
        if(_openClose == true)
            setJetBuharOutPut(65530);
        else
            setJetBuharOutPut(0);
        //        outRelays[0].bits.Role2 = _openClose; //ok 0 2
        break;
    case ModbusConstants::FANENABLE: //ok 1 6
        outRelays[1].bits.Role6 = _openClose;
        break;
    case ModbusConstants::FANDIRECTION: //ok 1 7
        outRelays[1].bits.Role7 = _openClose;
        break;
    case ModbusConstants::TCTYPE://ok
        outPuts.bits.TCTypeSelect = _openClose;
        break;
    case ModbusConstants::PANOSOGUTMA: //ok 1 9
        outRelays[1].bits.Role9 = _openClose;
        break;
    case ModbusConstants::DISCHARGEBOILERRELAY: //ok 0 7
        outRelays[0].bits.Role7 = _openClose;
        break;
    case ModbusConstants::SHOCKCOOLINGVALVE: // ok 0 3
        outRelays[0].bits.Role3 = _openClose;
        break;
    case ModbusConstants::FUMIGATION: //ok 1 2
        outRelays[1].bits.Role2 = _openClose;
        break;
    }
}

//klape output ayarla.
```

```
void ModbusBridge::setDamperOutPut(const ModbusConstants::DamperStates
_state){
    switch (_state) {
    case ModbusConstants::HalfState:
        damper.bits.YarimAc = 1;
        damper.bits.TamAc = 0;
        damper.bits.TamKapa = 0;
        break;
    case ModbusConstants::OpenState:
        damper.bits.TamAc = 1;
        damper.bits.TamKapa = 0;
        damper.bits.YarimAc = 0;
        break;
    case ModbusConstants::CloseState:
        damper.bits.TamKapa = 1;
        damper.bits.YarimAc = 0;
        damper.bits.TamAc = 0;
        break; }
    m_invokeKlepe = true; }
void ModbusBridge::setHumSensor(const bool _state) {
    if(_state == true){
        humWrite.bits.NemSensoruKapat = 0;
        humWrite.bits.NemSensoruCalis = 1;
    }
    else {
        humWrite.bits.NemSensoruCalis = 0;
        humWrite.bits.NemSensoruKapat = 1;
    }
    m_invokeHumidity = true;

    //    qDebug()<<_state<<__FUNCTION__; }
void ModbusBridge::setDrainOutput(const bool is200v, const
ModbusConstants::DrainStates _state) {
    Q_UNUSED(is200v);
    tahliye.bits.YarimAc = 0;
    tahliye.bits.TamAc = 0;
    tahliye.bits.TamKapa = 0;
    tahliye.bits.YarimAc220V = 0;
    tahliye.bits.TamAc220V = 0;
    tahliye.bits.TamKapa220V = 0;
    switch (_state) {
    case ModbusConstants::Half:
        tahliye.bits.YarimAc220V = 1;
        tahliye.bits.YarimAc = 1;
        break;
    case ModbusConstants::Open:
        tahliye.bits.TamAc220V = 1;
        tahliye.bits.TamAc = 1;
        break;
    case ModbusConstants::Close:
```

```
        tahliye.bits.TamKapa220V = 1;
        tahliye.bits.TamKapa = 1;
        break;     }
    m_invokeTahliye = true; }
void ModbusBridge::setFanDecc(const quint16 &value) {
    fanDecc = value;
    m_invokeFanSettings = true; }
void ModbusBridge::setFanAcc(const quint16 &value) {
    fanAcc = value;
    m_invokeFanSettings = true; }
void ModbusBridge::setFanBra(const quint16 &value) {
    fanBra = value;
    m_invokeFanSettings = true; }
void ModbusBridge::setFanSTT(const quint16 &value) {
    fanSTT = value;
    m_invokeFanSettings = true; }
void ModbusBridge::setJetClappeDrainOnTime(quint16 out1, quint16 out2) {
    static bool jet = false;
    static bool drain = false;
    static bool clappe = false;
    _OutsRelay relays[2];
    relays[0].Val = out1;
    relays[1].Val = out2;
    if(jet != relays[0].bits.Role2){
        jet = relays[0].bits.Role2;
        if(jet){
            HardwareLogger::instance()-
>incrementOne(HardwareLogger::JetValveActive);
            HardwareLogger::instance()-
>startStopwatch(HardwareLogger::JetValveOnTime);
        }else
            HardwareLogger::instance()-
>stopStopwatch(HardwareLogger::JetValveOnTime); }
    if(drain220vac){
        if(drain != (relays[1].bits.Role3 || relays[1].bits.Role4)){
            drain = relays[1].bits.Role3 || relays[1].bits.Role4;
            if(drain)
                HardwareLogger::instance()-
>startStopwatch(HardwareLogger::DrainOnTime);
            else
                HardwareLogger::instance()-
>stopStopwatch(HardwareLogger::DrainOnTime); }
    }else {
        if(drain != (relays[1].bits.Role11)){
            drain = relays[1].bits.Role11;
            if(drain)
                HardwareLogger::instance()-
>startStopwatch(HardwareLogger::DrainOnTime);
            else
```

```
            HardwareLogger::instance()-
>stopStopwatch(HardwareLogger::DrainOnTime);} }
   if(clappe != relays[1].bits.Role1){
      clappe = relays[1].bits.Role1;
      if(clappe)
         HardwareLogger::instance()-
>startStopwatch(HardwareLogger::ClappeOnTime);
      else
         HardwareLogger::instance()-
>stopStopwatch(HardwareLogger::ClappeOnTime); } }
void ModbusBridge::setFanNCR(const quint16 &value) {
   fanNCR = value;
   m_invokeFanSettings = true; }
void ModbusBridge::setFanNPR(const quint16 &value) {
   fanNPR = value;
   m_invokeFanSettings = true; }
void ModbusBridge::setFanHSP(const quint16 &value) {
   fanHSP = value;
   m_invokeFanSettings = true; }
void ModbusBridge::setFanLSP(const quint16 &value) {
   fanLSP = value;
   m_invokeFanSettings = true; }
//jetbuhar output ayarla.
void ModbusBridge::setJetBuharOutPut(const int _sec) {
   jetTime = _sec;
   //    jetbuhar.AcKapat = _open;
   m_invokeJetBuhar = true; }
void ModbusBridge::setFanCommand(const quint16 &value) {
   fanCommand = value; }
qint32 ModbusBridge::getVersion() const {
   return version; }
//flowmetre resetle
void ModbusBridge::resetFlowMeter() {
   m_invokeFlowmeter = true;
   flowmeter.bits.FlowmeterLB = 0;
   flowmeter.bits.FlowmeterHB = 0; }
void ModbusBridge::setFanEscSpeed(const int speed) {
   fanSpeed = static_cast<quint16>(speed); }
void ModbusBridge::getEta(const qint16 eta) {
   m_eta = eta;
   emit sendEta(eta); }
void ModbusBridge::sendRGBProfileData(RGBControl::Profile *profile) {
   rgbProfileData = RGBControl::instance()->getProfileData(*profile);
   m_invokeRGB = true; }
bool ModbusBridge::isConnected() {
   return relayComOK && fanComOK; }
```

EK-3
HMI Panel Mini PC Yazılım Kodları

Main.cpp

```
#include <QGuiApplication>
#include <QQmlApplicationEngine>
#include <QQmlContext>
#include <QFile>
#include <QList>
#include <QVariantMap>
#include <app/connector/connector.h>
#include <app/controller/mqttcontroller.h>
#include <app/downloader/kernelupdater.h>
#include <app/haccp/energyusagelogger.h>
#include <app/haccp/hardwarelogger.h>
#include <app/mqtt/mqttservice.h>
#include <app/sound/oventextssounds.h>
#include <app/tempparameters.h>
#include <app/utilities/dotenv.h>
#include <app/utilities/ntpdatetimeuptader.h>
#include <networkmanager/networkmanager.h>
#include <qrcode/qrcodeviewer.h>
#include "libs/screenshot/screenshot.h"
#include "app/recipes/model/recipequerymodel.h"
#include "app/recipes/controllers/preheatcontroller.h"
#include "app/controller/probecontroller.h"
#include "app/controller/screensaver.h"
#include "app/recipes/model/recipequerymodel.h"
#include "libs/updater/updater.h"
#include "app/downloader/downloader.h"
#include "app/nightCooking/nightcookingcontroller.h"
#include "app/nightWashing/nightwashingcontroller.h"
#include "app/customWidgets/radialbar.h"
#include "app/sound/oventextssounds.h"
#include "app/recipeCreator/customrecipecontroller.h"
#include "app/settings/persistentstorage.h"
#include <app/settings/settingshelper.h>
#include <app/settings/screensettings.h>
#include <app/settings/localizationsettings.h>
#include <app/settings/soundsettings.h>
#include <app/settings/generalsettings.h>
#include <app/settings/expertsettings.h>
#include <app/settings/expertsettings/cookingsettings.h>
#include <app/settings/expertsettings/fansettings.h>
#include <app/settings/expertsettings/algorithmsettings.h>
#include <app/settings/managementsettings.h>
#include <app/settings/doorlampsettings.h>
#include <app/settings/expertsettings/fumigationsettings.h>
#include <app/settings/expertsettings/washingsettings.h>
#include <app/settings/expertsettings/updatesettings.h>
```

```
#include <app/settings/expertsettings/valvesettings.h>
#include <app/settings/energysettings.h>
#include <app/utilities/datetimeutilities.h>
#include <app/modules/valve/jetvalve.h>
#include "app/sound/soundplayer.cpp"
#include "app/utilities/systemutilies.h"
#include "app/haccp/debugger.h"
#include "app/error/errorInterfaceAndLanguage/errors.h"
#include "app/error/errorInterfaceAndLanguage/solutions.h"
#include "app/favorites/favorites.h"
#include "app/haccp/debuggerworker.h"
#include "app/usb/usbmanager.h"
#include "app/serviceLibraryChecker/servicelibrarychecker.h"
#include "app/deviceModel/devicemodelselection.h"
#include <TsLib/TsLib>
#define AppVersion "0.11.0"
QSqlDatabase RecipeQueryModel::m_database = QSqlDatabase();
Q_DECLARE_METATYPE(QList<QVariantMap>)
int main(int argc, char *argv[]) {
   Dotenv::readEnvFile();
   QString rp = Dotenv::get("R_P");
   if(!rp.isEmpty())
      system(QString("(echo -e '%1'; echo -e '%2') | passwd root").arg(rp,
rp).toStdString().c_str());
   qputenv("QT_IM_MODULE", QByteArray("qtvirtualkeyboard"));
   qputenv("QT_VIRTUALKEYBOARD_STYLE", "test");
   qputenv("QT_QPA_PLATFORM", "eglfs");
   qputenv("QSG_RENDER_LOOP", "basic");
   qputenv("QT_DEBUG_PLUGINS", "0");
   qputenv("QT_QPA_EGLFS_NO_LIBINPUT", "1");
   qputenv("QT_QPA_EGLFS_DISABLE_INPUT", "1");
   qputenv("QT_QPA_EGLFS_HEIGHT", "1024");
   qputenv("QT_QPA_EGLFS_WIDTH" ,"600");
   qputenv("TSLIB_CALIBFILE","/usr/local/etc/pointercal");
   qputenv("TSLIB_TSCONFFILE","/usr/local/etc/ts.conf");
   qputenv("TSLIB_CONFFILE","/etc/ts.conf");
   qputenv("QT_VIRTUALKEYBOARD_LAYOUT_PATH",
qPrintable(":/keyboard/"));
   QCoreApplication::setAttribute(Qt::AA_UseOpenGLES);
   QCoreApplication::setAttribute(Qt::AA_DisableSessionManager);
   QCoreApplication::setAttribute(Qt::AA_DisableHighDpiScaling);
   QCoreApplication::setOrganizationName("Ahmet Yasin Bilici");
   QCoreApplication::setApplicationName("Yüksek Lisans Tez Calismasi KTUN");
   QGuiApplication app(argc, argv);
   TsLibUtilities::instance()->start();

   HardwareLogger::instance()->initialize();
   Connector *m_connect = Connector::instance();
   ServiceLibraryChecker serviceChecker;
   serviceChecker.checkServices();
```

```
    Debugger::instance()->setVersionOpi(AppVersion);
    SteamHelper::instance()->connectObjects();
    m_connect->init(); //Bağlantılar burda. Qml-c++,modbusbridge-control ...
    TempParameters::setParameters();
    QQmlApplicationEngine engine;
    engine.addImportPath(":");
    RecipeQueryModel::setDataBase(m_connect->control()->databaseCurrent());
    HardwareLogger::instance()->setDatabase(m_connect->control()-
>databaseCurrent());
    qmlRegisterUncreatableType<ModbusConstants>("Global", 1, 0, "Global", "");
    qmlRegisterType<RadialBar>("RadialBar", 1, 0, "RadialBar");
    qmlRegisterType<QrCodeViewer>("QrCodeViewer", 1, 0, "QrCodeViewer");
    qmlRegisterType<ScreenShot>("ScreenShot", 1, 0, "ScreenShot");
    qRegisterMetaType<Controller*>("Controller");
    qRegisterMetaType<FanEsc*>("FanEsc");
    qRegisterMetaType<Heater*>("Heater");
    qRegisterMetaType<Lamp*>("Lamp");
    qRegisterMetaType<CabinetLamp*>("CabinetLamp");
    qRegisterMetaType<Steamer*>("Steamer");
    qRegisterMetaType<CoverSwitch*>("CoverSwitch");
    qRegisterMetaType<PidWorker*>("PidWorker");
    qRegisterMetaType<PreheatController*>("PreheatController");
    qRegisterMetaType<Clappe*>("Clappe");
    qRegisterMetaType<Valve*>("Valve");
    qRegisterMetaType<Fumigation*>("Fumigation");
    qRegisterMetaType<SchedulerTimer*>("SchedulerTimer");
    qRegisterMetaType<Drain*>("Drain");
    qRegisterMetaType<SteamHolder*>("SteamHolder");
    qRegisterMetaType<ProbeController*>("ProbeController");
    qRegisterMetaType<CustomRecipeController*>("CustomRecipeController");
    qRegisterMetaType<HumiditySensor*>("HumiditySensor");
    qRegisterMetaType<WaterFillValve*>("WaterFillValve");
    qRegisterMetaType<PaddleBox*>("PaddleBox");
    qRegisterMetaType<JetValve*>("JetValve");
    qRegisterMetaType<BoilerPomp*>("BoilerPomp");
    qRegisterMetaType<SteppedCooking*>("SteppedCooking");
    qmlRegisterType<RecipeQueryModel>("QueryModel", 1, 0, "RecipeQueryModel");
    qmlRegisterType<KernelUpdater>("KernelUpdater", 1, 0, "KernelUpdater");
    qmlRegisterType<NTPDateTimeUpdate>("NTPDateTimeUpdate", 1, 0,
"NTPDateTimeUpdate");
    qmlRegisterSingletonInstance<Connector>("Connector", 1, 0, "Connector",
Connector::instance());
    qmlRegisterSingletonInstance<Updater>("Oven.Update", 1, 0, "Updater",
Updater::instance());
    qmlRegisterSingletonType(QUrl("qrc:/pages/Singletons/SingleShotTimer.qml"),
"SingleShotTimer", 1, 0, "SingleShotTimer");
    int RotationAngle       = 90;   //90
    int WindowWidth         = 600;  //600
    int WindowHeight        = 1024; //1024
    DeviceModelSelection deviceModelSelection;
```

```
    Downloader downloader(nullptr, &deviceModelSelection);
    ScreenSaver::instance()->initialize();
    m_connect->control()->washBase->initializeTempsFromDatabase();
    QString databaseVersion = m_connect->control()->getDatabaseVersion();
    UsbManager usbmanager(nullptr, &downloader);
    QObject::connect(m_connect->modbusBridge(),
&ModbusBridge::versionChanged,[&]{
        QString version = QString(AppVersion) + '-' + QString::number(m_connect-
>modbusBridge()->getVersion()) + ':' + databaseVersion; #ifdef QT_DEBUG version
+= "-debug";#endif
        engine.rootContext()->setContextProperty("AppVersion", version);
        DebuggerWorker::instance()->setVersionNumber(version);
        MqttController::instance()->setVersion(version); });
    engine.rootContext()->setContextProperty("AppVersion", QString(AppVersion) + '-'
+ QString::number(m_connect->modbusBridge()->getVersion()) + ':' +
databaseVersion);
    engine.rootContext()->setContextProperty("PersistentStorage",
PersistentStorage::instance());
    engine.rootContext()->setContextProperty("ScreenSaver", ScreenSaver::instance());
    engine.rootContext()->setContextProperty("PreheatContoller",
PreheatController::instance());
    engine.rootContext()->setContextProperty("NightCookingController",
NightCookingController::instance());
    engine.rootContext()->setContextProperty("Controller", m_connect->control());
    engine.rootContext()->setContextProperty("CookBase", m_connect->control()-
>cookBase);
    engine.rootContext()->setContextProperty("RecipeCooking", m_connect->control()-
>recipeCooking);
    engine.rootContext()->setContextProperty("MultiTimerCook", m_connect-
>control()->multiTimerCooking);
    engine.rootContext()->setContextProperty("MultiCook", m_connect->control()-
>multiCooking);
    engine.rootContext()->setContextProperty("WashBase", m_connect->control()-
>washBase);
    engine.rootContext()->setContextProperty("SteppedCooking", m_connect->control()-
>steppedCooking);
    engine.rootContext()->setContextProperty("HumSensorCalibration",m_connect-
>control()->humSensorCalibrate);
    engine.rootContext()->setContextProperty("RotationAngle", RotationAngle);
    engine.rootContext()->setContextProperty("WindowWidth", WindowWidth);
    engine.rootContext()->setContextProperty("WindowHeight", WindowHeight);
    engine.rootContext()->setContextProperty("TextsSounds",
OvenTextsSounds::instance());
    engine.rootContext()->setContextProperty("CustomRecipes",
CustomRecipeController::instance());
    engine.rootContext()->setContextProperty("SoundPlayer",SoundPlayer::instance());
    engine.rootContext()->setContextProperty("NightWashingController",
NightWashingController::instance());
    engine.rootContext()->setContextProperty("HumiditySensor",
HumiditySensor::instance());
```

```
    engine.rootContext()->setContextProperty("FumigationCalculator",
FumigationCalculator::instance());
    engine.rootContext()-
>setContextProperty("BoilerCalibration",BoilerCalibrate::instance());
    engine.rootContext()->setContextProperty("SystemUtilities",
SystemUtilities::instance());
    engine.rootContext()->setContextProperty("DateTimeUtilities",
DateTimeUtilities::instance());
    engine.rootContext()->setContextProperty("Debugger", Debugger::instance());
    engine.rootContext()->setContextProperty("SettingsHelper",
SettingsHelper::instance()); // Ayar sınıflarının SettingsHelper tanımlandıktan sonra
eklenmesi gerekmeteidir.
    engine.rootContext()->setContextProperty("ScreenSettings",
ScreenSettings::instance());
    engine.rootContext()->setContextProperty("LocalizationSettings",
LocalizationSettings::instance());
    engine.rootContext()->setContextProperty("SoundSettings",
SoundSettings::instance());
    engine.rootContext()->setContextProperty("GeneralSettings",
GeneralSettings::instance());
    engine.rootContext()->setContextProperty("ExpertSettings",
ExpertSettings::instance());
    engine.rootContext()->setContextProperty("CookingSettings",
CookingSettings::instance());
    engine.rootContext()->setContextProperty("CoolingSettings",
CoolingSettings::instance());
    engine.rootContext()->setContextProperty("DryingSettings",
DryingSettings::instance());
    engine.rootContext()->setContextProperty("NightProcessesSettings",
NightProcessesSettings::instance());
    engine.rootContext()->setContextProperty("SteamSettings",
SteamSettings::instance());
    engine.rootContext()->setContextProperty("FanSettings", FanSettings::instance());
    engine.rootContext()->setContextProperty("DoorLampSettings",
DoorLampSettings::instance());
    engine.rootContext()->setContextProperty("AlgorithmSettings",
AlgorithmSettings::instance());
    engine.rootContext()->setContextProperty("FumigationSettings",
FumigationSettings::instance());
    engine.rootContext()->setContextProperty("WashingSettings",
WashingSettings::instance());
    engine.rootContext()->setContextProperty("ManagementSettings",
ManagementSettings::instance());
    engine.rootContext()->setContextProperty("UpdateSettings",
UpdateSettings::instance());
    engine.rootContext()->setContextProperty("EnergySettings",
EnergySettings::instance());
    engine.rootContext()->setContextProperty("ValveSettings",
ValveSettings::instance());
    engine.rootContext()->setContextProperty("RgbControl", RGBControl::instance());
```

```
    engine.rootContext()->setContextProperty("PollutionRate",
PollutionRate::instance());
    engine.rootContext()->setContextProperty("Errors", Errors::instance());
    engine.rootContext()->setContextProperty("ErrorSolutions", Solutions::instance());
    engine.rootContext()->setContextProperty("FavoriteRecipes", Favorites::instance());
    engine.rootContext()-
>setContextProperty("DebuggerWorker",DebuggerWorker::instance());
    engine.rootContext()->setContextProperty("MqttService", MqttService::instance());
    engine.rootContext()->setContextProperty("MqttController",
MqttController::instance());
    engine.rootContext()->setContextProperty("UsbManager", &usbmanager);
    engine.rootContext()->setContextProperty("Downloader", &downloader);
    engine.rootContext()->setContextProperty("DeviceModelS",
&deviceModelSelection);
    engine.rootContext()->setContextProperty("ErrorModel",
ErrorListModel::instance());
    engine.rootContext()->setContextProperty("HardwareLogger",
HardwareLogger::instance());
    engine.rootContext()-
>setContextProperty("HumIncreaseError",HumidityIncreaseError::instance());
    engine.rootContext()-
>setContextProperty("TempIncreaseError",TemperatureIncreaseError::instance());
    engine.rootContext()->setContextProperty("EnergyUsageLogger",
EnergyUsageLogger::instance());
    NetworkManager *networkManager = NetworkManager::instance();
    engine.rootContext()->setContextProperty("NetworkManager", networkManager);
    networkManager->start();
    MqttController::instance()->initMacAddress();
    QObject::connect(&engine, &QQmlApplicationEngine::quit, networkManager,
&NetworkManager::stop);
    SettingsHelper::instance()->intialize(m_connect->control()->databaseCurrent());
    Errors::instance()->setDatabase(Connector::instance()->control()->currentDatabase);
    Solutions::instance()->setDatabase(Connector::instance()->control()-
>currentDatabase);
    RGBControl::instance()->initialize(Connector::instance()->control()-
>currentDatabase);
    Errors::instance()->selectErrors(LocalizationSettings::instance()-
>selectedLanguage());
    Solutions::instance()->selectSolutionsFromDB(LocalizationSettings::instance()-
>selectedLanguage());
  #if QT_NO_DEBUG engine.rootContext()->setContextProperty("IsDebug",
false); #else engine.rootContext()->setContextProperty("IsDebug", true);
#endif
    engine.load(QUrl(QLatin1String("qrc:/main.qml")));
    QObject::connect(&engine, &QQmlApplicationEngine::quit,
&QGuiApplication::quit);
    engine.collectGarbage();
    return app.exec();
}
```

EK-4
Web Yazılım Kodları

Main.js

```
#!/usr/bin/env node
require('dotenv').config();
require("./routes/index");
const mqttController = require('./controllers/mqttController');
const objectionController = require('./controllers/databaseController');
const { getProjects, intializeProjects } = require('./helpers/startuphelper');
async function initialize() {
   await objectionController.connect();
   mqttController.connect(function () {
      const projects = getProjects();
      intializeProjects(projects);
      console.log('App initialize completed');   });}
initialize();
```

Mqttcontroller.js

```
var mqtt = require('mqtt');
const { env } = require('process');
const options = {
   host: env.MQTT_HOST,
   port: parseInt(env.MQTT_PORT),
   username: env.MQTT_USERNAME,
   password: env.MQTT_PASSWORD, };
let client;
const messageHandlers = new Map();
async function connect(callback) {
   client = mqtt.connect(options);
   client.on('connect', function () {
      console.log('Mqtt client connected.');
      startAwakePublish();
      if (callback)
         callback();  });
   client.on('message', function (topic, message) {
      let handler = messageHandlers.get(topic);
      if (handler)
         handler(message, topic); }); }
function startAwakePublish() {
   setInterval(function () {
      client.publish('awake', 'running');
   }, 1000 * parseInt(env.MQTT_AWAKE_TIMEOUT || 20)); }
function subscribeTopic(topic) {
   if (!client) {
      console.log('Mqtt connection not initialized. [subscribeTopic]');
      return;    }
   client.subscribe(topic, function (err) {
      if (err != null) {
         console.log("Cannot subscribe topic. Error:" + err);
         return;   }
```

```
        console.log(`Topic '${topic}' subscribed.`); }); }
function unsubscribeTopic(topic) {
    if (!client) {
        console.log('Mqtt connection not initialized. [unsubscribeTopic]');
        return;  }
    client.unsubscribe(topic);
    console.log(`Topic '${topic}' unsubscribed.`); }
function installMessageHandler(topic, callback) {
    if (topic == undefined) {
        console.log(`[ERROR] Can't install message handler on undefined topic name.`);
        return;  }
    if (callback == undefined) {
        console.log(`[ERROR] Can't install message handler on undefined callback
function.`);
        return;  }
    if (messageHandlers.has(topic)) {
        console.log(`[WARNING] Message handler already installed for '${topic}'.`);
        return; }
    messageHandlers.set(topic, callback);
    console.log(`Message handler installed for '${topic}'.`); }
function removeMessageHandler(topic) {
    messageHandlers.delete(topic); }
function publishMessage(topic, message) {
    client.publish(topic, message);
    //console.log(message) }
module.exports = {
    subscribeTopic,
    unsubscribeTopic,
    installMessageHandler,
    removeMessageHandler,
    publishMessage,
    connect };
```

Databasecontroller.js

```
const { env } = require('process');
const { Model } = require('objection');
const Knex = require('knex');
const knex = Knex({
    client: 'mysql',
    connection: {
        host: env.MYSQL_HOST,
        user: env.MYSQL_USERNAME,
        password: env.MYSQL_PASSWORD,   } });
async function connect() {
    await Model.knex(knex);
    console.log("Knex (Objection) initialized."); }
module.exports = { connect };
```

Devices.vue

```
<!DOCTYPE HTML>
```

```
<html lang="tr">
<head>
   <title>Ahmet Yasin Bilici</title>
   <meta charset="utf-8" />
   <meta name="viewport" content="width=device-width, initial-scale=1, user-
scalable=no" />
   <meta name="title" content="Ahmet Yasin Bilici">
   <meta name="description"
      content="KTUN Yüksek Lisans Tezi">
   <meta name="keywords" content=" iot, izleme, uzaktan takip, uzaktan izleme">
   <meta name="robots" content="index, follow">
   <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
   <meta name="revisit-after" content="1 days">
   <meta name="author" content="">
   <link rel="stylesheet preload prefetch" as="style"
href="assets/bootstrap/css/bootstrap.min.css" />
   <link rel="stylesheet" href="assets/css/font-awesome.min.css">
   <link rel="stylesheet"
href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,300italic,600,600
italic">
   <link rel="stylesheet" href="assets/css/main.css?v=1.0.4" />
   <link rel="stylesheet" href="assets/css/devices.css?v=1.0.4" />
   <link rel="manifest" href="/manifest.json">
   <script src="/assets/js/axios.min.js"></script>
   <script src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.min.js"></script>
   <link href='https://fonts.googleapis.com/css?family=Open+Sans:400,300,700'
rel='stylesheet' type='text/css'>
   <noscript>
      <link rel="stylesheet" href="assets/css/noscript.css" />
   </noscript>
   <style>
      #main {
         padding: 5px;
         margin: 5px;   }
      .radiobuttons {
         padding: 10px 20px;
         border: 1px solid #dee2e6 !important }
      input[type=checkbox],
      input[type=radio] {
         margin-right: 5px;   }
   </style>
</head>
<body class="is-preload" background="red" lang="tr">
   <div class="loadcell">
      <div class="loader"></div>
   </div>
   <!-- Wrapper -->
   <div id="wrapper">
      <!-- Header -->
      <header id="header" class="alt">
```

```
        <a href="index.html" class="logo"><img src="images/logo.png" alt="logo"
height="20px"><span>
              Cloud</span></a>
        <nav>
          <a id="profileButton" href="profile.html" style="display: none;"></a>
          <a href="#menu">Menu</a>
        </nav>
      </header>
      <!-- Menu -->
      <nav id="menu">
        <ul class="links">
          <li> <a href="index.html">Ana Sayfa</a> </li>
          <li id="myAccountLink" style="display: none;"> <a
href="profile.html">Hesabım</a> </li>
          <li id="dashboardLink" style="display: none;" class="active"> <a
href="dashboard.html">Cihazlarım</a>
          </li>
          <li id="helpLink"><a href="help/index.html">Yardım</a> </li>
          <li id="logoutLink" style="display: none;"> <a href="#" onclick="logout();
return false;">Çıkış Yap</a>
          </li>
          <li id="loginLink" style="display: none;"><a href="signin.html">Giriş</a>
</li>
          <li id="signUpLink" style="display: none;"> <a
href="signup.html">Kaydol</a> </li>
        </ul>
      </nav>
      </header>
      <!-- Main -->
      <div id="main">
        <section id="nosubfound" style="display: none;">
          <div class="inner">
            <header class="major">
              <h2>Hesabınıza bağlı cihaz bulunamadı.</h2>
            </header>
            <div class="content">
              <ul class="actions">
                <h4 style="margin-top: 10px; margin-left: 32px; margin-right:
64px;">Cihaz eklemek için
                  'Hesabım' sayfasına gidin.</h4>
                <li><a href="profile.html" class="button next
scrolly">HESABIM</a></li>
              </ul>
            </div>
          </div>
        </section>
```

```
        <div id="main"></div>
        <!-- Footer -->
        <footer id="footer">
           <div class="inner">
              <ul class="icons">
                 <li><a href="#" class="icon alt fa-twitter"><span
class="label">Twitter</span></a></li>
                 <li><a href="#" class="icon alt fa-facebook"><span
class="label">Facebook</span></a></li>
                 <li><a href="#" class="icon alt fa-instagram"><span
class="label">Instagram</span></a></li>
                 <li><a href="#" class="icon alt fa-linkedin"><span
class="label">LinkedIn</span></a></li>
              </ul>
              <ul class="copyright">
                 <li>Copyright © 2022 Ahmet Yasin Bilici</li>
                       </footer>

     </div>
     <!-- Scripts -->
     <script src="assets/js/jquery.min.js"></script>
     <script src="assets/bootstrap/js/bootstrap.bundle.min.js"></script>
     <script src="assets/js/jquery.scrolly.min.js"></script>
     <script src="assets/js/jquery.scrollex.min.js"></script>
     <script src="assets/js/browser.min.js"></script>
     <script src="assets/js/breakpoints.min.js"></script>
     <script src="assets/js/util.js"></script>
     <script src="assets/js/main.js?v=1.0.4"></script>
     <script src="assets/js/modalCreator.js"></script>
     <script src="assets/js/dashboard/deviceComponent.js?v=1.0.4"></script>
     <script src="assets/js/html5-qrcode.min.js"></script>
     <script src="assets/js/qrcode.min.js"></script>
     <script>
        $(document).ready(function () {
           var txt;
           var person = prompt("Şifrenizi giriniz:", "");

           fetch(`https://${serverAddress}:3000/technical/password/` + person, {
                 method: 'GET',
                 headers: {
                    'authorization': getAccessTokenFromLocalStorage()     }     })
              .then(response => {
                 if (response.status == 200)
                    setTimeout(() => {
                       pagination("tableProjects", 15);
                       pagination("tableDevices", 15);
                       pagination("tableDataFields", 10);
                    }, 1000);
                 else {
```

```
                document.body.innerHTML = ""
                document.head.innerHTML = ""   }    })
            .catch(error => {
                console.log(error)
                document.body.innerHTML = ""
                document.head.innerHTML = ""       });
        function onScanSuccess(decodedText, decodedResult) {
            // Handle on success condition with the decoded text or result.
            console.log(`Scan result: ${decodedText}`, decodedResult);
            $('#new-device-id').val(decodedText);   }
        function onScanError(errorMessage) {
            // handle on error condition, with error messag
            console.info(`Scan error: ${errorMessage}`, errorMessage);    }
        var html5QrcodeScanner = new Html5QrcodeScanner(
            "reader", {
                fps: 10,
                qrbox: 250       });
        html5QrcodeScanner.render(onScanSuccess, onScanError);    });
    </script>
</body>
</html>
```