KOCAELİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ
ANABİLİM DALI

YÜKSEK LİSANS TEZİ

DENİZ ARAÇLARI İÇİN NESNELERİN İNTERNETİ TEKNOLOJİLERİ TEMELLİ ŞARJ KONTROL SİSTEMİ TASARLANMASI

YAŞAM USTAOĞLU

KOCAELİ 2021

KOCAELİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ
ANABİLİM DALI

YÜKSEK LİSANS TEZİ

DENİZ ARAÇLARI İÇİN NESNELERİN İNTERNETİ TEKNOLOJİLERİ TEMELLİ ŞARJ KONTROL SİSTEMİ TASARLANMASI

YAŞAM USTAOĞLU

Prof. Dr. Kerem KÜÇÜK
Danışman, Kocaeli Üniv. .....

Dr. Öğr. Üyesi Alev MUTLU
Jüri Üyesi, Kocaeli Üniv. .....

Doç. Dr. Cüneyt BAYILMIŞ
Jüri Üyesi, Sakarya Üniv. .....

Tezin Savunulduğu Tarih:24.08.2021

ÖNSÖZ VE TEŞEKKÜR

20.yüzyılda teknolojide meydana gelen hızlı gelişmelere paralel olarak toplumsal hayatta da gelişmeler meydana gelmiştir. Bu gelişmelerden en önemlisi kuşkusuz ki Nesnelerin İnterneti (Internet of Things, IoT) teknolojisidir. İnsanlar yaşantılarının hemen her alanında bu teknolojiden yararlanmaktadır. Bu alanlardan biri de deniz araçlarıdır.

Deniz araçları düzenli olarak kullanılan araçlar değildir. Bu yüzden motor düzenli olarak çalışmamakta ve batarya da şarj olmamaktadır. Bu durum bataryalardan alınan verimi düşürmekte ve bataryanın ömrünü azaltmaktadır. Diğer bir problem de deniz araçları açık denizde durağan hale geçince sürekli olarak bataryanın voltaj durumunun kontrol edilmesi gerekmektedir. Aksi halde aracı tekrar çalıştırabilmek için bataryada yeteri kadar enerji kalmayabilmektedir.

Bu tez çalışmasında IoT teknolojisinden yararlanarak insanların deniz araçlarında bataryadan dolayı yaşadıkları sorunların önüne geçebilmek için yeni bir sistem prototip geliştirilmiştir ve prototipi gerçeklenmiştir. Bu prototip ile deniz aracının motoru kullanıcıdan bağımsız olarak diğer etkenlerde kontrol edilerek belirtilen süre kadar çalıştırılıp bataryanın şarj verimliliği sağlanmıştır.

Tez çalışması boyunca bana olan desteğini hiçbir zaman eksik etmeyen, ne zaman olursa olsun çalışmalarımda takılıp kaldığım noktalarda benimle iletişime geçerek beni yönlendiren ve beni bu çalışmayı başarılı bir şekilde sonuçlandırabilmek için her zaman motive eden değerli danışmanım Prof. Dr. Kerem KÜÇÜK'e saygılarımı ve teşekkürlerimi bir borç bilirim.

Çalışmam boyunca bana olan desteğini, güvenini ve sabrını hiçbir zaman eksik etmeyen değerli eşim Arif USTAOĞLU'na, desteği ve yardımları ile her zaman yanımda olan sevgili arkadaşım Damla GÖR'e teşekkürlerimi içtenlikle sunarım.

Mayıs – 2021 Yaşam USTAOĞLU

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

SİMGELER VE KISALTMALAR DİZİNİ

° : Derece

Kısaltmalar

AH	:	Amper Saat
DC	:	Direct Current (Doğru Akım)
DEG	:	Diesel Engine Generator (Dizel Motor Jeneratörü)
HMI	:	Human Machine Interface (Makine İnsan Arayüzü)
IDE	:	Integrated Development Environment (Entegre Geliştirme Ortamı)
IoT	:	Internet of Things (Nesnelerin İnterneti)
KB	:	Kilobyte
LDR	:	Light Dependent Resistor (Işığa Bağımlı Direnç)
mm	:	Milimetre
MPPT	:	Maximum Power Point Tracking (Maksimum Güç Noktası Takibi)
NB-IoT	:	Narrow-Band IoT (Dar Band IoT)
PLC	:	Programmable Logic Controller (Programlanabilir Mantıksal Denetleyici)
PV	:	PhotoVoltaic (fotovoltaik)
PWM	:	Pulse Width Modulation (Sinyal Genişlik Modülasyonu)
RFID	:	Radio Frequency Identification (Radyo Frekansı ile Tanımlama)
RTC	:	Real Time Clock (Gerçek Zamanlı Saat)
UART	:	Universal Asynchronous Receiver Transmitter (Evrensel Asenkron Alıcı Verici)
Wi-Fi	:	Wireless Fidelity (Kablosuz Alan)
V	:	Volt
WTG	:	Wind Turbine Generator (Rüzgâr Türbini Jeneratörü)

DENİZ ARAÇLARI İÇİN NESNELERİN İNTERNETİ TEKNOLOJİLERİ TEMELLİ ŞARJ KONTROL SİSTEMİ TASARLANMASI

ÖZET

Günümüzde küçük boyutlardaki gömülü sistem donanımı kullanımının yaygınlaşmasıyla birlikte Nesnelerin İnterneti kapsamında (Internet of Things, IoT) özgün ve yenilikçi çözümler ortaya çıkmaktadır. Bu doğrultuda IoT çözümlerinin uygulama alanlarından bir tanesi deniz araçları için uzaktan izleme ve kontrol sistemlerinde kullanımı olarak ortaya çıkmaktadır. Dolayısıyla IoT teknolojileri kullanarak deniz araçlarında sıklıkla ortaya çıkabilen elektrik batarya probleminin çözümüne yönelik IoT temelli bir sistem mimarisi gereksinimi bulunmaktadır.
Deniz aracı bataryasının şarj edilebilmesi ve ömrünün uzun olabilmesi için elektrik motorunun düzenli olarak çalıştırılması gerekmektedir. Deniz araçlarının kullanım aralıkları sık olmaması nedeniyle aküler düzenli olarak şarj dolumu yapamamaktadır. Özellikle aracın durağan halde olması durumunda kullanıcının sürekli olarak akünün durumunu kontrol etmesi gerekmektedir. Bu çalışma kapsamında IoT teknolojilerini kullanan deniz araçlarının durağan halde iken otomatik şarj edilmesini gerçekleştirebilen bir sistem mimarisi ortaya konulmaktadır. Mimari sistem bileşenlerinde Arduino IDE (Integrated Development Environment), Nextion HMI (Human Machine Interface) ekran, Mobil uygulama ve Firebase bulut platformu bulunmaktadır. Geliştirilen prototipte; Arduino Uno, Arduino Mega, NodeMCU LoLin ESP8266 geliştirme kartı, RTC (Real Time Clock) saat modülü, seviye ölçme sensörü, buzzer, DC (doğru akım) ve Servo motorlar kullanılmaktadır. Arduino IDE de geliştirilen yazılımda; yakıt miktarı ve kontak, gaz kolu kilidi, dümen kilidi durumları kontrol edilebilmektedir. Mobil uygulama tarafında gerçekleştirilen işlemlerin tamamı araç içinde Nextion ekran ile de gerçekleştirilebilmektedir. Kullanıcıların sistemin durumunu kontrol edebilmeleri için ESP8266 Wi-Fi modülü kullanılarak veriler anlık olarak Firebase bulut ortamına aktarılmaktadır. Bulut ortamına aktarılan veriler Android Studio'da geliştirilen mobil uygulamada kullanıcıya gösterilmektedir. Önerilen sistem mimarisi ile deniz aracının motoru düzenli olarak çalıştırıldığı için aküden yüksek verim alınarak ve akünün ömrü korunmuştur.

Anahtar Kelimeler: Arduino, Deniz Araçları, ESP8266, Nesnelerin İnterneti, Şarj Kontrol.

DESIGNING A CHARGE CONTROL SYSTEM BASED ON INTERNET OF THINGS TECHNOLOGIES FOR MARINE VEHICLES

ABSTRACT

Today, with the widespread use of small size embedded system hardware, unique and innovative solutions are emerging within the scope of the Internet of Things (IoT). In this direction, one of the application areas of IoT solutions emerges as the use of remote monitoring and control systems for marine vehicles. Therefore, there is a need for an IoT-based system architecture to solve the electric battery problem that can often occur in marine vehicles using IoT technologies.

In order to the marine vehicle battery to be recharged and to have a long life, the electric motor must be operated regularly. Due to the fact that the usage intervals of the marine vehicles are not frequent, the batteries cannot be charged regularly. Especially if the vehicle is stationary, the user must constantly check the state of the battery. Within the scope of this study, a system architecture that can automatically charge marine vehicles using IoT technologies while at rest is presented. Architectural system components include Arduino IDE, Nextion Display, Mobile application, and Firebase cloud platform. In the developed prototype, Arduino Uno, Arduino Mega, NodeMCU LoLin ESP8266 development board, RTC clock module, level measuring sensor, buzzer, DC and Servo motors are used. In the software developed in Arduino IDE; Fuel quantity and ignition, gas handle lock, rudder lock conditions can be controlled. All transactions performed on the mobile application side can also be carried out in the vehicle with the Nextion Display. In order to users to check the status of the system, the data is instantly transferred to the Firebase cloud environment using the ESP8266 Wi-Fi module. The data transferred to the cloud is shown to the user in the mobile application developed in Android Studio. With the proposed system architecture, since the engine of the marine vehicle is operated regularly, high efficiency is obtained from the battery and the life of the battery is protected.

Keywords: Arduino, Marine Vehicles, ESP8266, Internet of Things, Battery Control.

GİRİŞ

Nasa'ya ait Curiosity keşif robotu, 26 Kasım 2011 tarihinde araştırma yapmak için Mars'a fırlatılmıştır. Curiosity keşif robotunun Mars'a gönderilmesinden bu yana teknolojide hızlı bir ivme yaşanmıştır [2]. Teknolojinin hızlı gelişmesine paralel olarak insan hayatında büyük gelişmeler yaşanmıştır. Yaşanan bu gelişmeler arasında nesnelerin interneti önemli bir yer tutmaktadır. Hız ve bant genişliği açısından İnternet'teki her gelişmeyle birlikte, IOT (Nesnelerin İnterneti) pazarı yeni icat fırsatlarıyla kapıyı çalmaktadır [3]. Nesnelerin interneti sürekli olarak gelişen özellikleri ve yapısı sayesinde insan hayatına da birçok yenilik katmaktadır [4]. IoT sayesinde insanlar mesafe tanımaksızın cihazlar ile haberleşebilmektedir.

Bu çalışma kapsamında IoT teknolojisinden yararlanarak deniz araçlarındaki batarya sorunu çözmeye yönelik çalışma yapılmıştır. Deniz araçları düzenli olarak kullanılan araçlar değildir. Bu nedenden dolayı bataryaları düzenli olarak şarj edilmemektedir. Kullanıldıkları zamanlarda ise açık denizde durağan halde iken, deniz aracında enerji sarfiyatı var ise bataryadaki voltaj azalmaktadır. Kullanıcı bataryadaki kalan voltaj miktarını sürekli olarak kontrol etmek zorundadır aksi halde bataryada motoru çalıştırabilecek güç kalmayabilmektedir. Genellikle insanlar yedek batarya bulundurarak ya da sürekli olarak batarya voltaj değerini kontrol ederek bu durumun önüne geçmeye çalışmaktadırlar. Bataryalar maliyetli ekipmanlardır. Bu maliyet göze alınarak yedek batarya alınsa bile, düzenli olarak şarj edilmez ise batarya ömrü ve bataryadan alınan verim azalmaktadır.

Deniz araçlarında bataryayı şarj edebilmek için; rüzgâr türbini, güneş paneli ve su çarkından yararlanılabilmektedir. Rüzgâr türbini ile bataryanın şarj edilebilmesi için rüzgârın olması gerekmektedir. Rüzgâr türbini kanatlarına etki eden rüzgâr hızına bağlı olarak, enerji dönüşüm hedefleri ve kontrol hedefleri her bölge için farklı olabilmektedir [5]. Güneş panelleri ile batarya belli oranlarda şarj olabilmektedir. Bu oran panel büyüklüğü ve güneş ışınlarının yoğunluğu ile doğru orantılı olmaktadır. Enerji sarfiyatını karşılayabilecek büyüklükteki güneş panelleri deniz araçlarında yer ve maliyet sorunu oluşturmaktadır. Kullanıcı yer ve maliyet sorununu çözebilse bile

güneş olmadığında bu sistem ile bataryayı şarj edebilme mümkün olmamaktadır. Su türbininden yararlanarak bataryayı şarj edebilmek için ise deniz aracının hareket halinde olması gerekmektedir. Bataryayı şarj edebilmek için görüldüğü üzere çevresel faktörlere bağlı kalınmaktadır. Bu faktörler ortamda bulunmadığında bataryayı şarj edebilmek mümkün değildir. Yenilenebilir enerji kaynaklarında elde edilen güç depolanabilmektedir fakat elde edilen güç motorun çalışmasını sağlayacak güç için yeterli gelmemektedir. Bunun dışında bataryada meydana gelebilen bazı arızalar, bataryanın enerji depolamasını engelleyebilmektedir.

Bunlardan yola çıkarak deniz aracı bataryasını düzenli olarak şarj ederek bataryadan alınan verimi yükseltmek ve batarya ömrünü uzatmak için IoT tabanlı şarj kontrol uygulaması geliştirilmiş. Bu uygulamada herhangi bir çevresel etkene ihtiyaç bulunmamaktadır. Kesintili ve belirsiz bir enerji kaynağı olan rüzgâr üniteleri ve fotovoltaik panellerden güç sağlama konusunda zorluklarla karşılaşılmaktadır [6]. Yapılan çalışma ile yenilenebilir enerji kaynaklarının yetersiz geldiği durumlarda, bataryayı şarj ederek deniz aracında motoru çalıştırabilecek yeterli gücün depolanması sağlanabilmektedir. Bataryada şarj depolanmadığında kullanıcı bilgilendirilerek kullanıcının bataryayı kontrol etmesi sağlanarak oluşabilecek sıkıntıların önüne geçilebilmektedir. Düzenli olarak batarya durumu kontrol edilip şarj edildiği için bataryadan yüksek verim alınması sağlanmaktadır. Batarya durumu gerçek zamanlı olarak android uygulamasında ve nextion ekranda izlenilebilmektedir [7].

Tezin çalışmasının organizasyonu şu şekildedir: Tezin birinci bölümünde IoT teknolojisinin ortaya çıkışı, uygulama alanları ve deniz araçlarındaki kullanımı hakkında bilgi verilmektedir. İkinci bölümde literatürde ve güncel hayatta deniz aracı bataryasını şarj etme yöntemleri anlatılmaktadır. Üçüncü bölümde geliştirilen şarj kontrol uygulamasının amacı anlatılmaktadır. Bu bölümde sistem mimarisi ve kullanılan cihazlar arasındaki etkileşim detaylı bir şekilde anlatılmıştır.

Geliştirilen uygulamanın akış diyagramı verilmiştir. Uygulamanın prototipi verilmiştir. Prototipte kullanılan cihazlar, bu cihazlar hakkında bilgi ve bunların kullanılma sebebleri anlatılmaktadır. Geliştirme yapılan ortamlar ve bu ortamlarda yapılan geliştirmeler anlatılmaktadır. Tezin dördüncü bölümünde hazırlanan

senaryolardan bahsedilmiştir. Bu senaryoların uygulama üzerindeki sonuçları anlatılmaktadır. Dördüncü ve son bölümde ise sonuçlar yorumlanmakta, gelecek çalışmalarda yapılması gereken performans sorunlarını çözecek konulardan ve eklenebilecek yeni özelliklerden söz edilmektedir.

1. NESNELERİN İNTERNETİ

Nesnelerin İnterneti konusu, 1999 yılında Kevin Ashton'un bir şirketin tedarik zincirinde Radyo Frekansı ile Tanımlama (Radio Frequency Identification, RFID) teknolojisini kullanmanın firmaya sağladığı faydaları anlatan sunumunda tanıtılmıştır [8, 9]. Nesnelerin interneti, bir ağa bağlı tüm nesnelerin (internete bağlanılabilen/haberleşme kanalı olan elektronik cihazların), insan müdahalesine ve veri girişine gerek duymadan belirli bir protokol ile kendi aralarında veri iletişimi yaptığı, hayatın her anından milyarlarca veri toplayarak bu veriler ile karar verdiği, küresel internet tabanlı, akıllı bir iletişim sistemidir [10, 11]. IoT sistemi Şekil 1.1'de gösterilmiştir.

Şekil 1.1. IoT sistemi [4]

Nesnelerin interneti çok geniş bir programlama platformu haline dönüşmüştür [12]. Günümüzde IoT teknolojisinden birçok farklı alanda yararlanılmaktadır.

IoT nin kullanıldığı alanlarda daha kaliteli hizmet vermek, verimliliği ve üretkenliği arttırmak için duyargaların aracılığıyla veriler toplanmaktadır. Bu veriler "Veri Tabanı" oluşturarak bulut bilişim sistemlerinde depolanmaktadır [13].

1.1. IoT Uygulama Alanları

IoT teknolojisinin günümüzde gelmiş olduğu nokta itibari ile hayatımızın hemen her alanında kullanılmaktadır. Kullanılan alanların belli başlıları şunlardır [14];

- Akıllı ev uygulamaları,
- Akıllı şehir uygulamaları,
- Bilimsel çalışma uygulamaları,
- Bilişim sektörü uygulamaları,
- Enerji uygulamaları,
- Günlük kullanım uygulamaları,
- Güvenlik uygulamaları,
- İmalat/üretim uygulamaları,
- İnşaat uygulamaları,
- Kamu sektörü uygulamaları,
- Sağlık uygulamaları,
- Servis sağlayıcı uygulamaları,
- Tarımsal üretim uygulamaları,
- Taşımacılık uygulamaları,
- Ticaret uygulamaları.

1.2. Deniz Araçlarında IoT Kullanımı

Literatürde ve güncel hayatta deniz araçlarında IoT teknolojisini kullanan çalışmalar mevcuttur.

Bilişim sektöründe faaliyet gösteren Data Market, marina sektörü ve yat kullanıcılarının ihtiyaçlarını dikkate alarak bir uygulama geliştirmişlerdir.

Bu uygulama ile [15];

- Yatın marinaya geliş gidişlerini kayıt altına almışlardır.
- Motor çalışmasını takip etmişlerdir.
- Elektrik kesintilerinin takibini yapmış ve kullanıcıyı bilgilendirmişlerdir.
- Yat içinde herhangi bir kapak veya dolabın açılmasını izlemişlerdir.

- Yat içindeki nem ve sıcaklık seviyesini izleyen sensörler ile beklenmedik ani yükselmelerde alarm üretmişlerdir.
- Nem ve sıcaklık değişiminin kaydını tutmuşlardır.
- Akü voltajını ve seviyesini izleyen akü sensörü ile akü problemleri hakkında önceden bilgi verilmesini sağlamışlardır.
- Yat içinde duman ve karbon monoksit seviyesini izleyen CO sensörü ile yangın veya egzoz sızıntısında uyarı vermişlerdir.
- Yat içine su sızmalarını izleyen su algılama sensörü ile olası su sızıntılarını anında tespit edilerek yat sahibinin bilgilenmesini sağlamışlardır.

Siraporn ve Suratsavadee, genişlik sensörü, derinlik sensörü ve su akış sensöründen oluşan NB-IoT dağıtımı, dere, kanal, nehir vb. gibi su kaynakları yönetimi için bir sistem tasarlamış ve uygulamışlardır. Bu sensörlerden ölçülen verileri yakalamak ve küçük teknede gezinmek için bu sistemi geliştirmişlerdir. Mobilden mikro denetleyici kartına veri aktarımı bluetooth aracılığıyla sağlanmıştır. Nehir genişliği ve derinlik ölçümleri, Arduino Uno mikro denetleyicisi tarafından doğru bir şekilde yapılmış ve ölçülen tüm veriler bluetooth aracılığıyla geri gönderilmiştir. Önerilen mimarinin avantajı, bazı su kaynağı alanlarına ulaşmak için kablolu bağlantıların karmaşıklığını ve tüm erişilebilirlik sorunlarını azaltmasıdır. Deneysel sonuçlar, önerilen sistemin su kaynağının genişlik, derinlik ve su akış parametrelerini (su akış yönü) etkili bir şekilde belirleyebildiğini ve ardından kullanıcıya akış deşarjını tahmin edebildiğini ortaya koymuştur. Ek olarak teknenin konumunu, yerelleştirme özellikleri sayesinde mobil uygulama üzerinden konumlandırmasını Google haritası ile temsil etmişlerdir [16]. Önerilen sistem için Şekil 1.2'deki görseli kullanmışlardır.

Navjeet ve arkadaşları, şarj edilen telefonu kullanarak bulut üzerinden davranışını otomatik olarak kontrol eden akıllı bir şarj sistemini uygulamak için bir sistem tasarlamışlardır. Tasarlanan sistemde ESP8266, Arduino Uno (ESP8266 modülünü programlamak için), MB102 (güç kaynağı modülü olarak), IRF540 MOFSET(anahtar olarak) donanımlarını kullanmışlardır.

Sistemi otomatik ve yarı otomatik olmak üzere iki modda tasarlamışlardır. Otomatik modda, kullanıcının hiçbir şey yapmasına gerek yoktur, sunucu pil seviyesini

otomatik olarak alır ve önceden ayarlanmış MAX ve MIN şarj aralıklarına göre çalışır. Şarj seviyesi MAX seviyesine ulaşırsa, şarj cihazı kapanır ve bir süre sonra cihaz boşalıp MIN seviyesine ulaştığında otomatik olarak şarj cihazını açar ve şarj etmeye başlar. Yarı otomatik mod zamanlayıcıya dayalıdır. Kullanıcının yarı otomatik modda zamanlayıcı değerini forma girmesi gerekir. Zamanlayıcı değeri sunucuya gönderilir ve gömülü kod buna göre çalışır. Süre dolduktan sonra şarj cihazı otomatik olarak şarj etmeyi durdurur. ESP8266 ile şarj değerini yüzde olarak bulut ortamına göndermişlerdir. Evothings Viewer ya da ESP8266 modülünün IP adresi web tarayıcısında girilerek bulut ortamındaki şarj yüzdesini izlemişlerdir [17].

Şekil 1.2. Önerilen sisteme genel bakış [16]

Lai ve arkadaşları, kamuya açık alanlara insanların telefonlarını ücretsiz olarak şarj edebilmeleri için şarj noktaları koymuşlardır. Bu şarj noktalarına yerleştirmek için AnyCharge adlı IoT tabanlı kablosuz şarj sistemi geliştirmişlerdir. Bu şarj sisteminde, otomatik ve güvenli Wi-Fi bağlantı algoritmasını kullanarak bir IoT ağ geçidine otomatik olarak bağlanabilen Wi-Fi özellikli bir şarj cihazı kullanmışlardır. Wi-Fi özellikli şarj cihazı, Wi-Fi bağlantısını kurmak için kullanıcı arayüzü gerektirmez. Bu nedenle, şarj cihazları ve ağ geçitleri arasındaki bağlantıları otomatik olarak kuran birkaç güvenli ve otomatik Wi-Fi bağlantı algoritması geliştirmişlerdir. Tüm şarj cihazlarını ve ağ geçitlerini yönetmek için web tabanlı bir platform geliştirmişlerdir. Tüm şarj cihazlarını buluta bağlayarak şarj takibini yapmışlardır. Kullanıcıların en yakın şarj noktalarını bulabilmeleri için Android ve IOS uygulama geliştirmişlerdir. Ayrıca, her kullanıcının şarj süresini hesaplamışlardır [18].

Harish ve arkadaşları, pil performansının sürekli olarak izlenebilmesi için daha sistematik bir pil yönetim sisteminin uygulanması gerektiğini savunarak IoT tabanlı batarya yönetim sistemi geliştirmişlerdir. Bataryanın aşırı şarj edilmesi sonucunda hidrojen, oksijen vb. gazların emisyonu olabilmektedir. Sülfürik asit olan elektrolitin sulu çözeltisinin buharlaştırılmasıyla üretilmektedirler. Ayrıca sülfür gazı emisyonu olasılığı da yüksektir. Bu gazlar yanıcıdır ve endüstriyel alanlarda zararlı çalışma koşulları yaratmaktadır. Bir ışık durumunda bu gazların yakınında kıvılcım pilin patlamasına neden olabilmektedir. Bu yüzden pilin aşırı şarj olmasını önlemek için uyarı oluşturulması ve böylece herhangi bir tehlikeyi önlemek bu gazların tespitinin önemli olduğu sonucuna varmışlardır. Pilin çevresinde çok güvenli bir ortam sağlamayı amaçlamışlardır.

Bu sistemde aşırı şarj olan bataryadan salınan gazları, akünün gerilim, akım, sıcaklık gibi diğer temel parametrelerini izlemişlerdir. Negatif sıcaklık katsayılı termistör ile gerilim ve akımı ölçmüşlerdir. Bataryadan salınan gazların ölçümü için MQ-8 hidrojen sensörünü kullanmışlardır. Sıcaklık ölçümü için termistör kullanmışlardır. GPS modülü de kullanarak cihazın konum takibini yapmışlardır. Konum verisini ve okunan diğer parametreleri bulut ortamına aktarmışlardır. Böylece tüm bunlardan yola çıkılarak bir prototip model geliştirmişler ve başarıyla uygulamış ve test etmişlerdir [19].

2. DENİZ ARAÇLARINDA BATARYA ŞARJ ETME YÖNTEMLERİ

Literatürde ve güncel hayatta deniz araçlarında bataryayı şarj etmek için farklı teknikler kullanılmaktadır. Bu bölüm altında deniz aracı bataryasını şarj edebilmek için kullanılan yöntemler anlatılmaktadır.

2.1. Rüzgar Türbini ile Batarya Şarj Etme

Li Wang ve arkadaşları ticari bir balıkçı teknesinde enerji tasarrufu sağlamak için 400 watt WTG (rüzgâr türbini jeneratörü) kurulumunu yaparak, WTG' nin alan ölçüm sonuçlarını sunmuşlardır. PLC (Programlanabilir Mantıksal Denetleyici) tabanlı bir izleme ve kontrol sistemi de, çeşitli rüzgâr hızları ve yelken koşulları altında çalışılan WTG'nin ve balıkçı teknesindeki bataryanın ölçülen tüm elektrik miktarlarını etkili bir şekilde yakalamak için balıkçı teknesine uygun şekilde tasarlamış ve kurmuşlardır. Pil, yüksek rüzgâr hızlarında WTG'den kullanılabilir şarj gücüne sahip olduğunda, akünün DEG (dizel motor jeneratörü)'den gelen şarj akımı ve gerçek balıkçı teknelerinin dizel yakıt tüketimi, etkili bir şekilde enerji tasarrufu sağlamak ve hava kirliliği amaçlarını azaltmak için aynı anda azaltılabilesceği sonucuna varmışlardır. Üstelik balıkçı tekneleri kıyıya yakın demirlendiğinde WTG'nin, rastgele rüzgâr enerjisi kullanarak pil birimlerini de şarj edebildiğini gözlemlemişlerdir. Bunun birçok balıkçı teknesinde dizel yakıtın bir sonraki denize açılmadan önce DEG kullanarak akü ünitelerini şarj etmesi için zamandan ve paradan tasarruf eden önemli bir avantaj olduğu sonucuna varmışlardır [20].

2.2. Güneş Paneli İle Batarya Şarj Etme

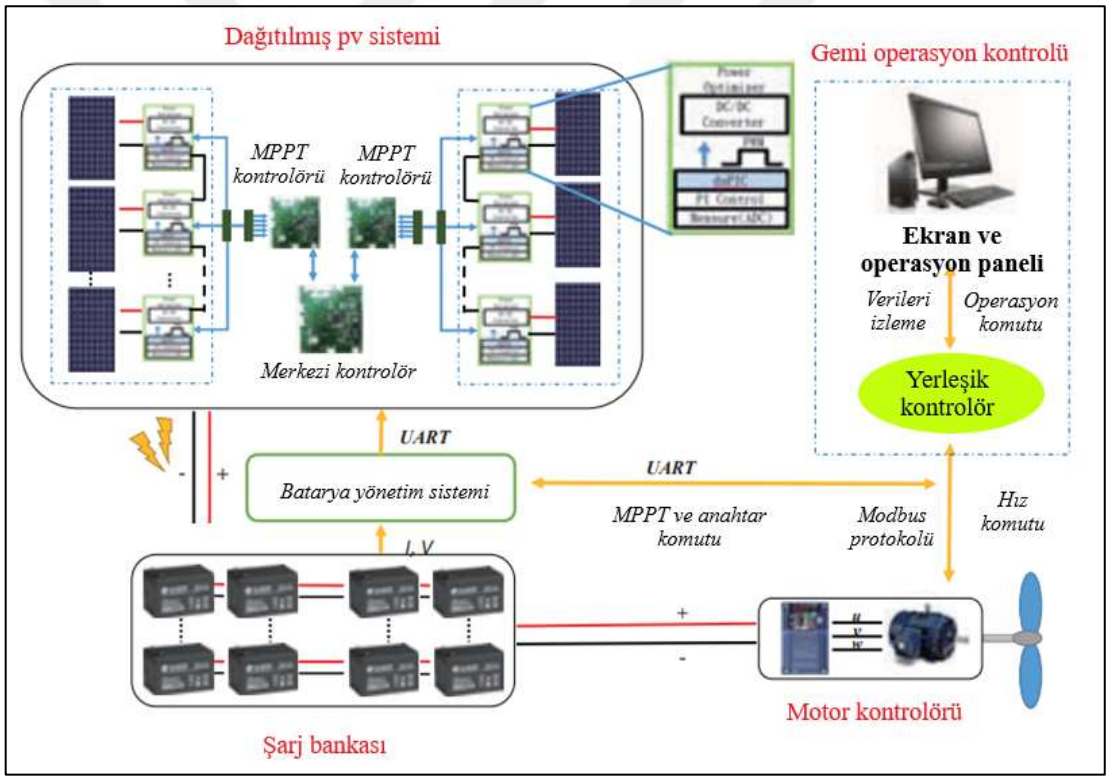
Güneş panellerinden üretilen enerji şarj kontrol cihazları ile bataryaya aktarılmaktadır. Bu şekilde bataryalar belli oranda şarj edilebilmektedir. Güneş panellerinden elde edilen enerjinin, deniz aracının tüm enerji kullanımını karşılayabilmesi mümkün olmamaktadır. Güneş paneli sadece batarya üzerindeki yükü azaltmaktadır. Çok donanımlı ve enerji sarfiyatı fazla olan deniz araçları için güneş panellerinden üretilen enerji yeterli gelmemektedir.

Bu sebepten dolayı enerji sarfiyatı az olan deniz araçlarında güneş paneli kullanımı daha çok tercih edilmektedir. Enerji sarfiyatı fazla olan deniz araçlarında güneş panellerinden ede edilen enerji, harcanan enerjinin bir kısmını karşılayabilmektedir.

Mudong ve Ani, yaptıkları araştırmada güneş panelinin performansının panel yüzeyine gelen ışığa bağlı olduğunu ve panel yüzeyinde başta toz ve yosun birikintileri olmak üzere çevresel parametreler tarafından yönetildiğinin sonucuna varmışlardır. Bu çevresel faktörlerin de panelin güç ve performansını önemli ölçüde azalttığını savunmuşlardır. Bu sorunları ortadan kaldırabilmek için kendi kendini temizleyebilen bir güneş paneli tasarlamışlardır. Önemli ölçüde güç kaybına neden olan güneş panelleri üzerindeki engelleri tespit etmek için LDR (foto dirençler) kullanmışlardır. Tespit edilen engelleri temizlemek için ise güneş panelleri boyunca uzanan temizleme mekanizması tasarlamışlardır. Tasarladıkları sistem herhangi bir engel tespit ettiğinde temizleme mekanizmasını etkinleştirmiştir. Tasarlanan sistemi kurdukları yerde fiziksel olarak bulunmadan IoT sayesinde internet bağlantısına erişerek istenilen yerde sistemi izlemişler ve güncellemişlerdir. Bu sistem sayesinde batarya boş olduğunda güneş paneli yeterli güneş ışığı alabildiği için bataryayı şarj edebilmiştir [21].

Chao, lin ve Wu, en son teknoloji maksimum güç noktası izleme teknolojisi, güç optimize edici ve PV (fotovoltaik) güç denetleyicisini içeren en son patentli dağıtılmış PV güç sistemini kullanan, akü yönetim sistemi, motor kontrolü ve çalışma kontrolünü içeren güneş enerjili bir tekne tasarımı çalışması yapmışlardır. Yerli eğlence yat üreticisinin yenilenebilir güneş enerjisi teknolojisini benimsemesine ve daha yüksek oranda yeşil enerji penetrasyonu ile yeni güneş gemisi tasarımını teşvik etmesine yardımcı olacak bir çerçeve çalışması sunmuşlardır. Panel üzerinde oluşan gölgelenmenin ve panel yaşlanmasının yüksek miktarda enerji kaybına sebeb olmaktadır. Bu etkiyi en aza indirebilmek için PV panellerinin her birine bağlı özel bir DC/DC dönüştürücü kullanmayı içeren "Dağıtılmış" fotovoltaik sistemini kullanmışlardır. Merkezi MPPT (Maximum Power Point Tracking) denetleyici tasarımı yapmışlardır. Bu MPPT denetleyici PIC32 mikro denetleyicisine dayanmaktadır. Kontrolörün kendisi, PV gücü veya harici bir pil ile çalıştırılabilmektedir. Varsayılan olarak 4 PV dönüştürücü, bir kişisel bilgisayar ve bluetooth cihazı ile iletişim kurmak için UART bağlantı noktası

kullanmışlardır. Merkezi bir kontrol mekanizması yapmışlardır. Bu merkezi kontrol mekanizması ile batarya yönetim sistemi ve motor kontrol sistemi ile iletişim kurmuşlardır. Tasarlanan sistemin diyagramı Şekil 2.1' de gösterilmiştir. Deniz aracı seyir halindeyken, merkezi kontrolör MPPT kontrolöründen maksimum güç çıkışı talep etmektedir. Bataryadaki enerjiyi sürekli olarak kontrol ederek, bataryanın aşırı şarj olmaması için PV çıkışını merkezi kontrolör ile dikkatli bir şekilde kontrol etmişlerdir. Bu geliştirdikleri tasarımı gerçek hayatta güneş enerjisi kullanan bir tekneyi yeniden modelleyerek gerçekleştirmişlerdir. Bununla birlikte güneş panellerinden gelen enerji, motorun çalışması için doğrudan bataryaya bağlanmıştır. Bu teknede daha önce güneş panellerinden elde edilen enerji sadece aydınlatma ve yardımcı ekipmanlar için kullanılmaktaymış [22].



Şekil 2.1. Güneş teknesi mekatronik sistemi için şematik diyagram [22]

Leung ve Cheng, güneş enerjisiyle çalışan arabaların araştırılması ve geliştirilmesi için büyük bütçe ayrıldığını fakat güneş enerjisiyle çalışan gemiler üzerinde durulmadığını söylemişlerdir. Motorlu deniz araçlarının da hava ve petrol kirliliği olmak üzere önemli ölçüde çevre kirliliği yarattığını savunmuşlardır. Bu sebepten dolayı güneş enerjisiyle çalışan tekne çalışması yapmışlardır. Yaptıkları çalışmada

dizel motorla çalışan geleneksel bir tekneyi, güneş enerjisiyle çalışan bir tekneye dönüştürmüşlerdir. Geliştirdikleri çalışmada güneş panelleri, pil bankası, motor kontrolörü, elektrik motoru, şanzıman ve pervane kullanmışlardır. Güneş enerjisini absorve etmek ve elektrik enerjisine dönüştürmek için güneş panellerini kullanmışlardır. Güneş panellerinden üretilen enerjinin gemideki tüm parçalara enerji sağlaması için pil bankasında depolanmasını sağlamışlardır. Motor kontrolörünü, motorun giriş gücünü, motorun hızına göre kontrol etmek için kullanmışlardır. Motoru, elektrik gücünü mekanik gücüne dönüştürmek için kullanmışlardır. Şanzıman sistemi, motor ve pervane arasındaki ortamdır. Şanzıman sistemi motorun mekanik gücünü pervaneye iletmek için kullanılmıştır. Yapılan çalışmada dizel motorlu tekne yerine güneş enerjili tekne kullanılması karbondioksit salınımının %74,2 azaltılabildiği sonucuna ulaşmışlardır. Güneş panellerinin verimliliğinin arttırılmasıyla bu oranın daha da yükselebileceğini söylemişlerdir. Bu sayede sıfır sera gazı emisyonu ile küresel ısınmayı yavaşlatmanın yanı sıra durdurmaya da yardımcı olabileceğini söylemişlerdir. Dizel motorlu tekne ile karşılaştırıldığında güneş enerjili tekne için maliyet tasarrufunun %87'den fazla olduğunu analiz etmişlerdir. Güneş panellerinin verimliliğinin arttırılmasıyla bu tasarruf oranının %89'un üzerine çıktığını görmüşlerdir. Aynı zamanda güneş enerjisiyle çalışan elektrikli teknelerin fosil yakıt bağımlılığı bulunmadığını, bu araçların kullanımı sırasında çok fazla gürültü çıkaran dizel ve benzinli gemilerin aksine sessiz çalıştığını, motorlu gemilerle karşılaştırıldığında makine mekanizmalarının daha basit olduğunu ve bu sebepten dolayı bakımlarının kolay olduğunu savunmuşlardır [23].

Mohapatra, Padhee ve Jena, kurşun asit pil için fotovoltaik tabanlı bir pil şarj cihazı tasarlamak için gömülü sistem tasarlamış ve uygulamışlardır. Kurşun-asit pili hem yüzer şarj modunda hem de toplu şarj modunda şarj etmişlerdir. Toplu şarj modunda, pili şarj etmek için maksimum güç noktası izleme (MPPT) algoritması kullanmışlardır. Batarya voltajı maksimum batarya voltajına eşit olduğunda bataryayı, sabit şarj modunda şarj etmişlerdir. Batarya yönetim sisteminde bataryayı şarj etmek için anahtarlamalı güç dönüştürücü kullanmışlardır. Boost dönüştürücü ve diğer yükseltici dönüştürücü topolojisi, pilin voltajı PV panelinin voltajından yüksek olduğunda kullanılmaktadır [24]. Bataryanın voltajı PV panelinden daha düşük

olduğunda Buck dönüştürücü kullanılmaktadır. Batarya voltajının pv panelinden düşük olduğunu varsayarak buck dönüştürücü kullanmışlardır. Şarjı depolamak için kurşun asitli batarya (12 V, 18 AH) kullanmışlardır. PV panelinden bataryaya beslemeyi kesmek için ek anahtar kullanmışlardır. PV voltajını ve batarya voltajını ölçmek için iki adet voltaj sensörü ve PV panelinin akımını ölçmek için bir adet akım sensörü kullanmışlardır. Donanım için ATmega328 mikrodenetleyici ve elde edilen değerleri göstermek için 20x4 LCD ekran kullanmışlardır. PV tabanlı batarya şarj modülünün deneysel prototipini geliştirmişler ve daha sonra dış ortamda test etmişlerdir. Bataryanın toplu modda şarj edildiği ve söz konusu şarj modunda MPPT algoritmasının şarjı kolaylaştırdığını gözlemişlerdir. Deneysel sonuçlar, pilin hem yüzer hem de toplu şarj modunda şarj edildiğini göstermiştir [25].

2.3. Su Gücü İle Batarya Şarj Etme

Akarsulara kurulan hidroelektrik santralleri, su gücünden yararlanarak elektrik enerjisi üretmektedir. Bu yöntemden yola çıkarak deniz araçlarında da su gücünden yararlanarak enerji üretebilme üzerine araştırma yapılmıştır.

Arif, Ahmet ve Yağmur, deniz ve okyanusta bulunan birden fazla enerji kaynağını elektriğe çevirecek bir sistem tasarlamışlardır. Açık denizde rüzgâr ve yüzey akış enerjisinden yararlanılarak, yüksek potansiyelli hibrit güç üretim sistemi modeline sahip bir platform oluşturmuşlardır. Yenilenebilir enerji kaynaklarının doğasında bulunan kesintili ve kararsız enerji türlerinin dayanıklılığını sağlamak için önerdikleri hibrit güç üretim sistemine pil ve ultra kapasitörden oluşan hibrit depolama uygulayarak optimum çözüm sağlamışlardır. Yaptıkları çalışmada hibrit güç üretim sistemi ve hibrit enerji depolama sistemini MATLAB/Simulik programını kullanarak tüm ünitelerin simülasyonunu yapmışlardır. Bu simülasyon çalışmasındaki sistemde; rüzgar jeneratörü, deniz akış jeneratörü ve bu güç üretim sistemlerine bağlı redüktör ve DC/DC boost dönüştürücüler, batarya ve ultrakapasitör grubu, çift yönlü DC/DC dönüştürücüler kullanmışlardır. Voltaj bazlı evirici kontrol algoritması ve akıllı enerji yönetimi algoritmasını da simüle etmişlerdir. Bu çalışmada, rüzgâr ve deniz akımı enerjisinden elde edilen mekanik enerjiyi elektrik enerjisine dönüştürmek için Futureenergy marka 48V1kW sabit mıknatıslı jeneratörü simüle etmişlerdir. Aynı şekilde enerji depolama ünitesinde

kullanılacak pil ve ultrakapasitör grubu simülasyonu yaparak MATLAB/Simulik ile sistem analizi yapmışlardır. Çalışma sonucunda yenilenebilir enerji kaynağı, rüzgâr ve deniz akıntısı enerjisi ve enerji depolama birimi, yüklerin birincil tedarikini sağlayarak kaliteli bir enerji transferi gerçekleştiği sonucuna varmışlardır. Sistemdeki tüm birimleri akıllı enerji yönetim algoritması ile kontrol etmişler. MATLAB/Simulik ile daha önce deneysel çalışma yapılmadan sistemin dinamik tepkisini incelemişlerdir. Böylece oluşabilecek arıza ve hataları önceden tahmin edebilmişlerdir [26]. Kullandıkları enerji yönetimi yöntemi ile enerji depolama ünitesinin doluluk durumunu kontrol etmişlerdir. Bunun enerji depolama ünitesi hizmet ömrünü uzattığını ve pil performansına fayda sağladığını görmüşlerdir. Yaptıkları çalışma ile yenilenebilir enerji kaynaklarındaki kesintili ve kararsız enerjiyi pil depolama sistemi ile azaltmışlardır.

3. İNTERNETİ TEKNOLOJİLERİ TEMELLİ ŞARJ KONTROL SİSTEMİ

Şarj kontrol uygulamasında; bataryanın voltajı belirli bir voltajın altına düştüğünde, deniz aracındaki yakıt miktarı, dümen, kontak, gaz kolu ve motor kontrol edilerek istenilen durumlar sağlanıyorsa motor çalıştırılarak batarya düzenli olarak şarj edilmektedir.

Bu bölüm altında bataryayı şarj edebilmek için geliştirilen uygulama üzerinde durulmuştur. Uygulamanın amacı, sistem mimarisi, nasıl çalıştığı, geliştirme yapılan ortamlar, iş akışı ve geliştirilen prototip hakkında detaylı bilgi verilmektedir. Sonraki bölümlerde, hazırlanan senaryolar prototip üzerinde gerçekleştirilmiş ve sonuçlar gözlemlenmiştir.

3.1. Uygulamanın Amacı

Bu çalışmanın amacı bataryanın sistem tarafından düzenli olarak şarj edilerek, ömrünün korunması ve bataryadan yüksek verim alınmasıdır.

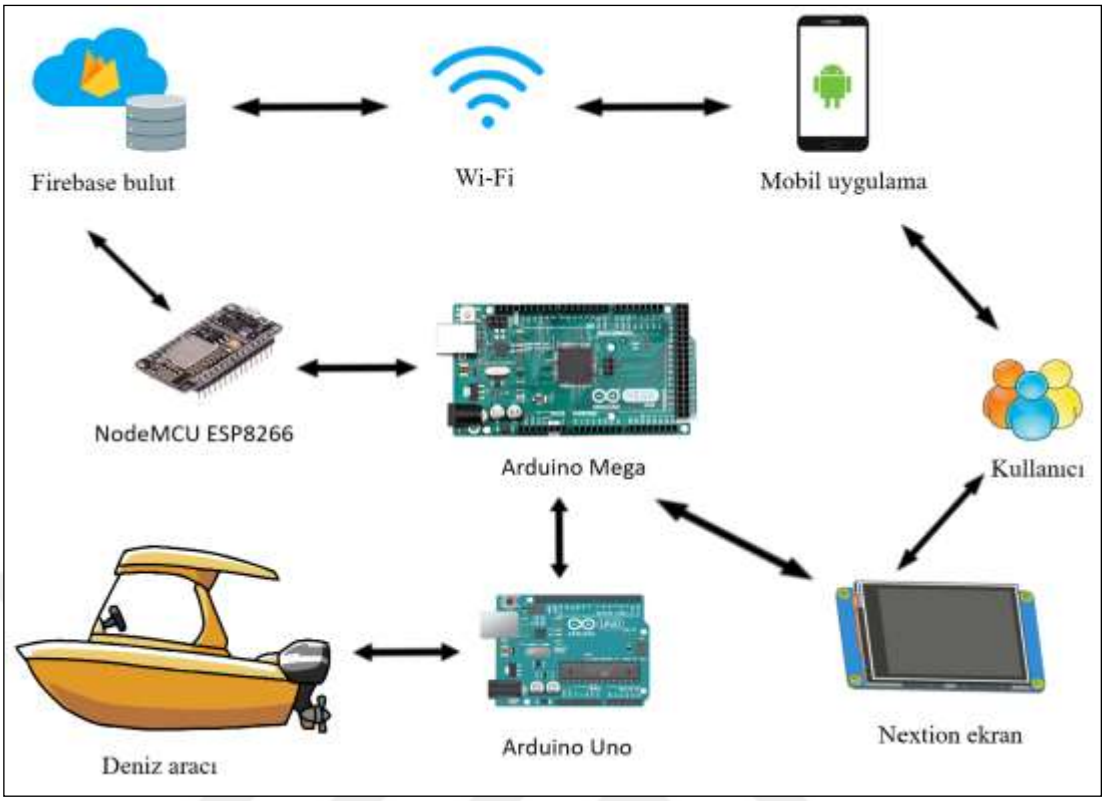
3.2. Sistem Mimarisi

Şarj kontrol uygulamasında kullanılan temel cihaz ve kavramların birbirleri ile etkileşimi Şekil 3.1'de gösterilmiştir. Sistem mimarisinde kullanılan tüm donanımlar detaylı olarak geliştirilen prototip üzerinde gösterilmektedir. Kullanılan tüm donanımlar ve bu donanımların kullanılma sebebleri bu bölüm altında "Uygulama Prototipi" başlığı altında detaylı olarak açıklanmaktadır.

Firebase veri tabanı ile mobil uygulama arasındaki çift taraflı veri aktarımı Wi-Fi ile sağlanmaktadır.

Firebase veri tabanı ile Arduio Mega arasındaki çift taraflı veri aktarımı NodeMCU v3 modülü ile sağlanmaktadır.

Arduino Mega ve NodeMCU v3 modülü ile çift taraflı veri aktarımı seri haberleşme ile sağlanmaktadır.



Şekil 3.1. Şarj kontrol uygulaması sistem mimarisi

Arduino Uno ve Arduino Mega arasındaki çift taraflı veri alışverişi IC2 Seri haberleşme protokolü kullanılarak sağlanmaktadır.

Arduino Mega ile nextion ekran arasındaki çift taraflı veri alışverişi seri haberleşme ile sağlanmaktadır.

Deniz aracındaki durumları kontrol eden sensörler Arduino Uno mikro denetleyicisine bağlıdır. Sensörlerden alınan veriler Arduino Mega mikro denetleyicisine gönderilmektedir. Arduino Mega da bu verileri nextion ekrana ve NodeMCU v3 modülü ile firebase veri tabanına göndermektedir. Firebase veri tabanına gönderilen veriler de Wi-Fi aracılığı ile mobil uygulamaya gönderilmektedir.

Kullanıcının nextion ekran üzerinden yaptığı işlemler Arduino Mega mikro denetleyicisine gönderilmektedir. Arduino Mega da bu verileri Arduino Uno mikro denetleyicisine ve NodeMCU Wi-Fi modülü ile firabase veri tabanına göndermektedir. Firebase veri tabanına gönderilen veriler de Wi-Fi aracılığı ile mobil

uygulamaya gönderilmektedir. Kullanıcı son durumları mobil uygulama ve nextion ekran üzerinden görebilmektedir.

Kullanıcının mobil uygulama ile müdahale ettiği veriler Wi-Fi aracılığı ile firebase veri tabanına aktarılmaktadır. Firebase veri tabanında veriler NodeMCU Wi-Fi modülü ile Arduino Mega mikro denetleyicisine aktarılmaktadır. Arduino Mega bu verileri Arduino Uno ile sensörlere ve nextion ekrana göndermektedir.

3.3. Akış Diyagramı

Şarj kontrol uygulamasının akış diyagramı Şekil 3.3'de detaylı bir şekilde gösterilmektedir.

Kontak ve gaz kolu durumunda meydana gelen değişikliğin anlık olarak sisteme yansıması için kesmelerden yararlanılmıştır. Bu kesmelere ait akış diyagramı Şekil 3.4 ve Şekil 3.5 te belirtilmiştir.

3.4. Uygulama Prototipi

İlgili çalışmanın prototipi Şekil 3.6 daki görselde belirtilmiştir.

3.4.1. Batarya

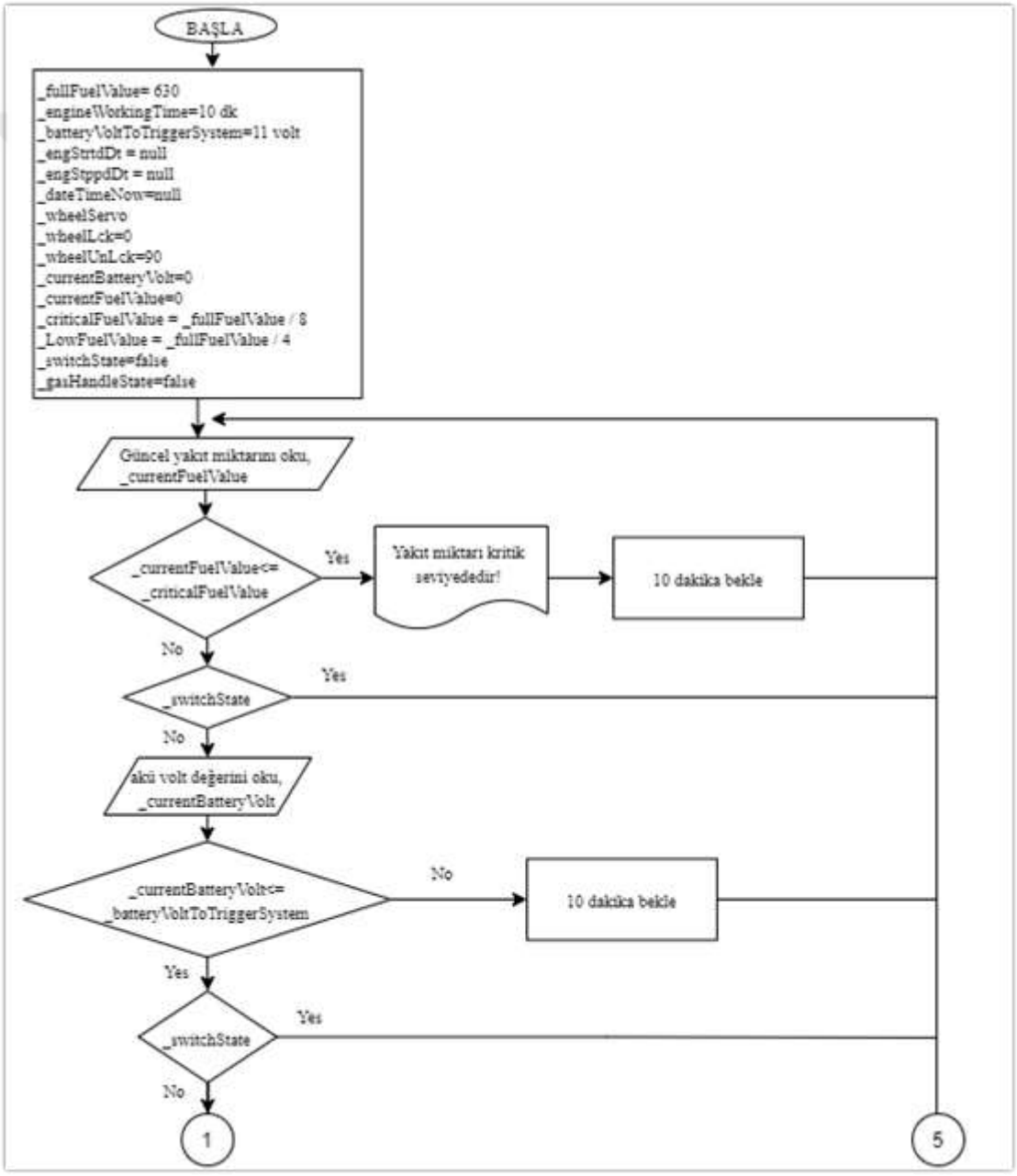
9 V (volt) ve 4,8 V olmak üzere iki adet pil kullanılmıştır. Farklı senaryoları test edebilmek için bu şekilde iki pil kullanılmıştır. Bazı senaryolarda sadece 9 volt olan pil kullanılmıştır. Bazı senaryolarda ise bu iki pil birbirine seri bağlanılarak toplam voltaj değeri kullanılmıştır. Prototipte deniz aracının bataryası olarak görevlendirilmiştir.

3.4.2. Su seviyesi, yağmur sensörü

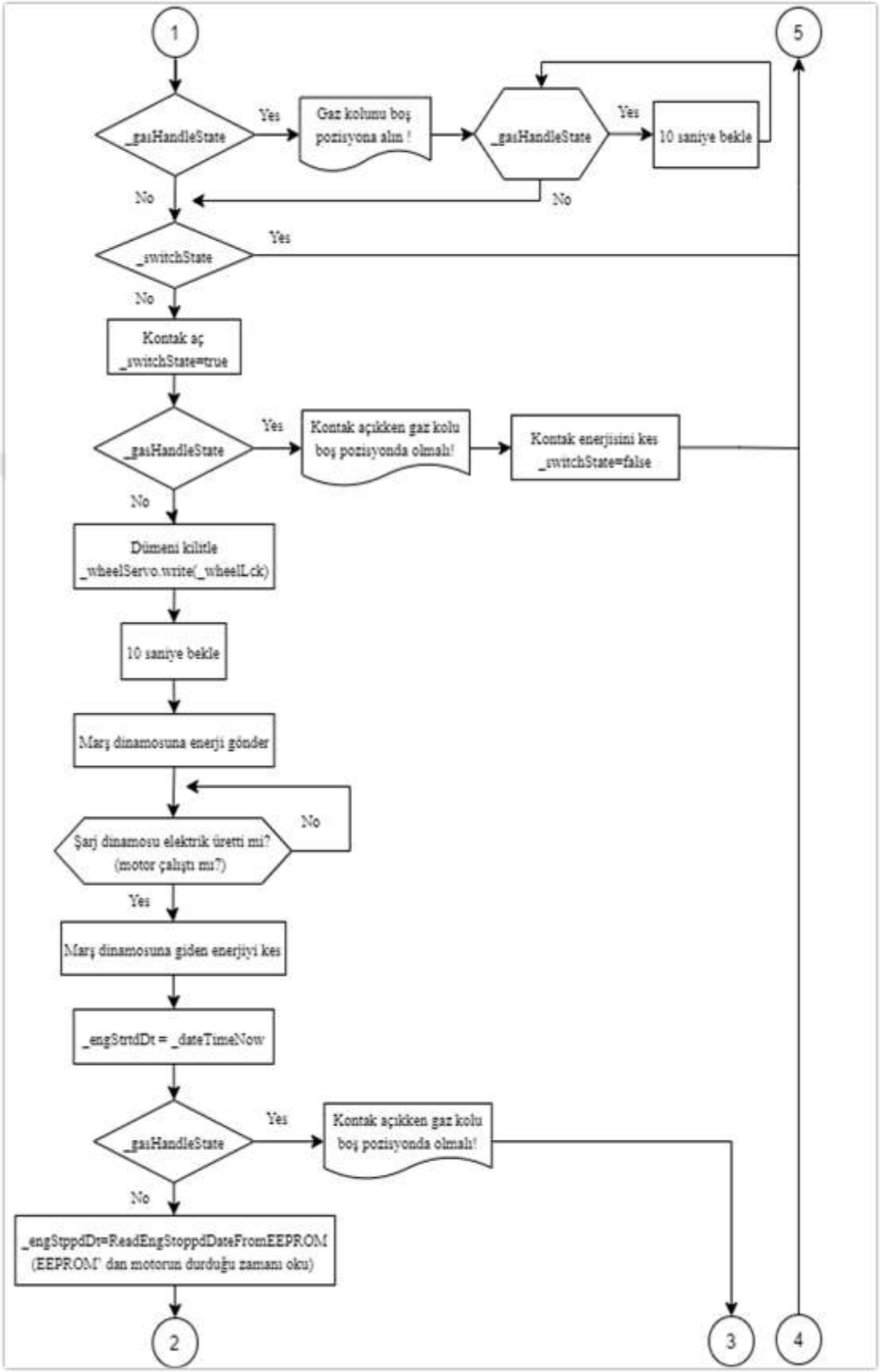
Yağmur sensörü sığ su seviyelerinde ve yağmurlu ortamda kullanılan bir sensör çeşididir. Bu sensör 40 mm su seviyesine kadar ölçüm yapabilmektedir. Yağmur sensörünün yapısında birbirine paralel olarak bağlanmış iletken hatlar bulunmaktadır. Bu hatlar su ile temas ettiğinde Arduino'ya analog bir sinyal gönderir. Arduino haricinde birçok mikro kontrolcü ile de çalışmaktadır [27]. Prototipte deniz aracının deposundaki yakıt miktarını ölçmek için görevlendirilmiştir.

3.4.3. Dc motor

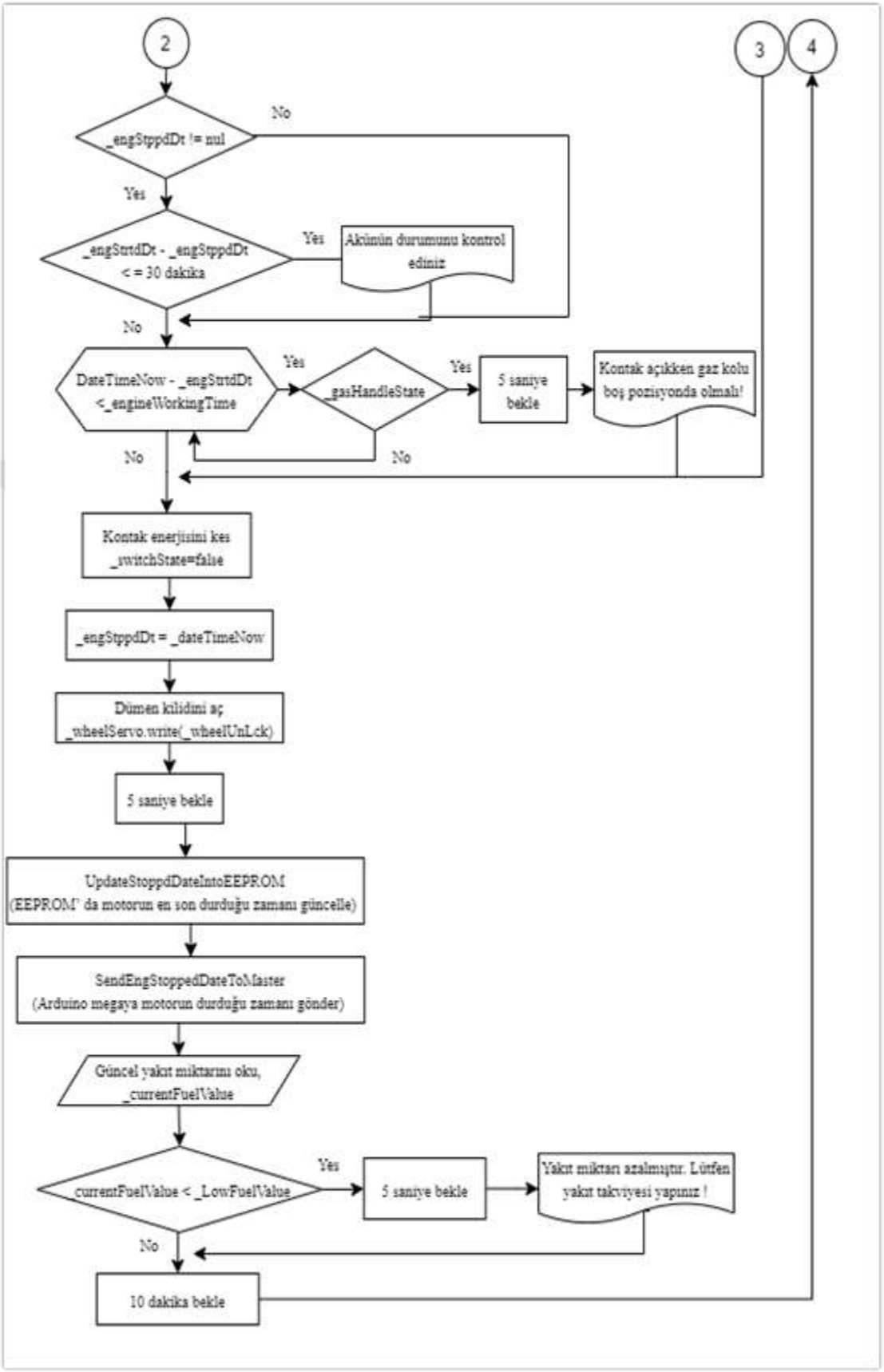
Doğru akım enerjisini mekanik enerjiye çeviren makinelerdir. Motorun içinde yer alan sargılara elektrik akımı uygulandığında, yine motorun içerisinde bulunan sabit mıknatıslara zıt yönde oluşan manyetik kuvvetin etkisi ile hareket etmesi prensibine dayanmaktadır [28]. Prototipte iki adet dc motor kullanılmıştır. İki dc motorun hareket eden uçlarına çarklar yerleştirilerek biri hareket ettiğinde diğerini de hareket ettirecek şekilde uç uca monte edilmiş ve bir tanesinin çıkışına led takılmıştır.



Şekil 3.2. Tüm sistemin akış diyagramı

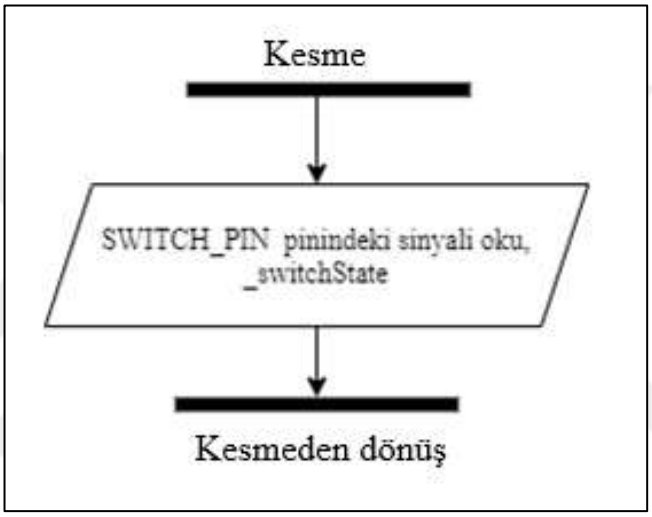


Şekil 3.2. (Devam) Tüm sistemin akış diyagramı

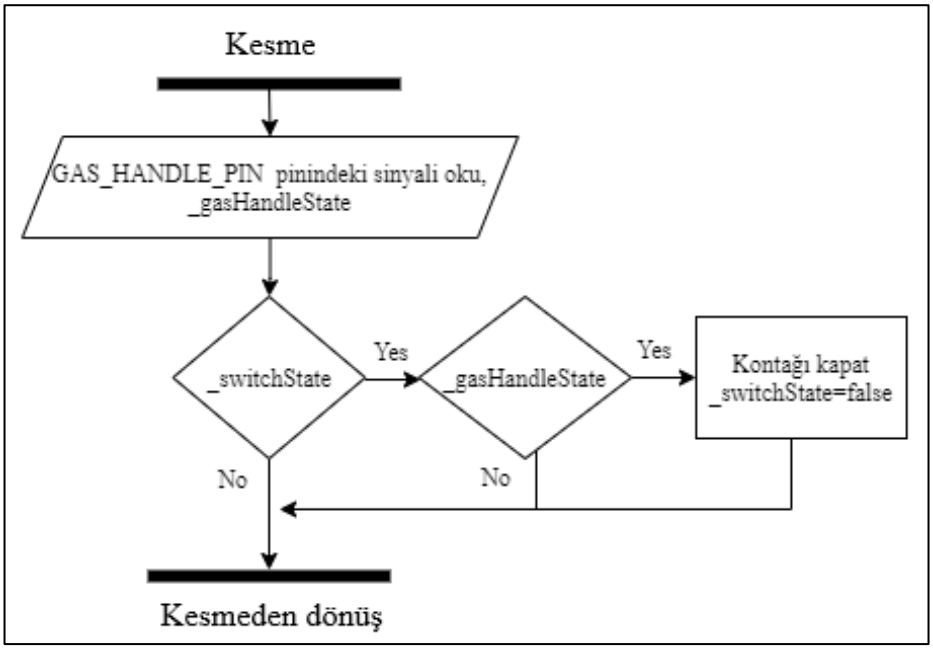


Şekil 3.2. (Devam) Tüm sistemin akış diyagramı

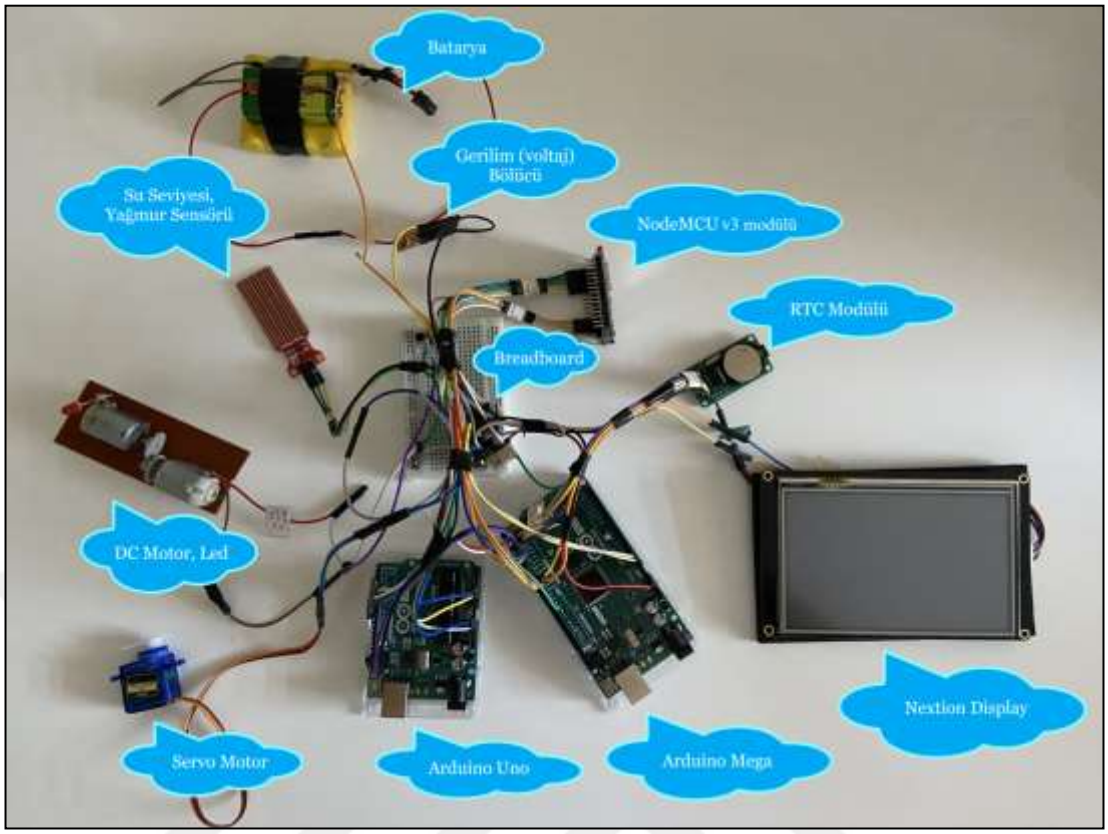
Bu dc motorlardan bir tanesi deniz aracının marş dinamosu olarak görevlendirilmiştir. Diğer dc motor ise prototipte deniz aracının hem şarj dinamosu hem de motoru olarak görevlendirilmişti. Motor çalıştırılarak bataryanın şarj üretmesi için mars dinamosu görevindeki dc motora elektrik verilerek hareket etmesi sağlanmıştır. Bu dc motor hareket ederek diğer şarj dinamosu ve motor görevindeki dc motoru hareket ettirerek bu dc motorun elektrik üreterek bağlı bulunduğu ledin yanmasını sağlamıştır. Prototipteki bu olay, motorun çalışarak bataryayı şarj ettiği anlamına gelmektedir.



Şekil 3.3. SWITC_PIN kesme akış diyagramı



Şekil 3.4. GAS_HANDLE_PIN kesme akış diyagramı



Şekil 3.5. Uygulama prototipi

3.4.4. Servo motor

Servo motor bir mekanizmanın performansını etkileyebilecek hataları geri bildirim sinyalleri yardımı ile kısa zaman aralığında hataları kontrol eden ve bu hataları engelleyen bir DC motor çeşididir. Servo motor dönüş yönünün belirli açılarda dönmesinin istenilen uygulama alanlarında tercih edilmektedir.

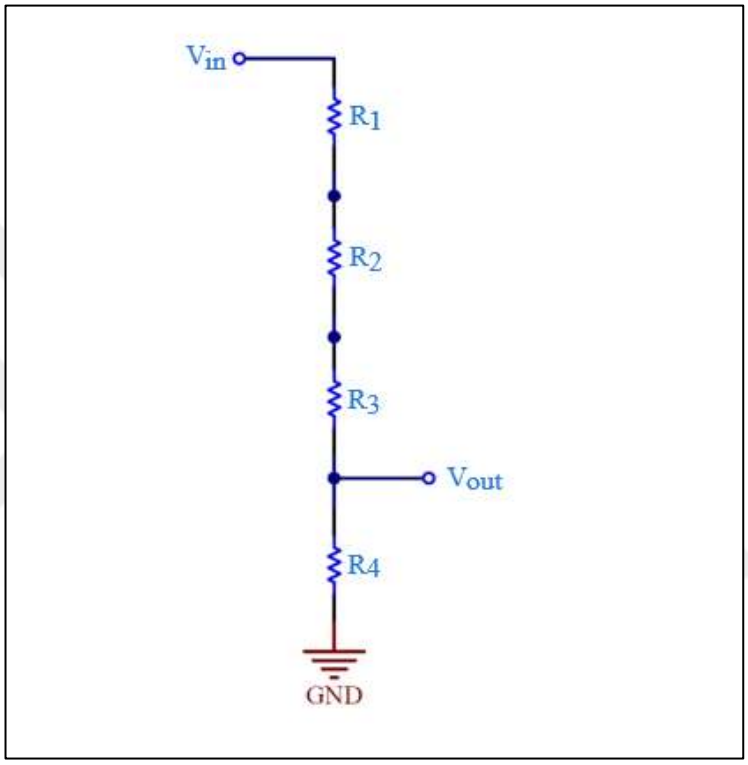
Servo motorlar bazı uygulama alanlarında motorun yüksek hızlarda sürekli dönmesinin istenmediği ve belirli aralıklar ve açılarda pozisyon almalarda kullanılmaktadır [29].

Prototipte deniz aracının dümen kilidi olarak görevlendirilmiştir. Dümen kilitlendiğinde servo motorun açısı 0° olarak ayarlanmıştır. Dümen kilidi açıldığında servo motor açısı 90° olarak ayarlanmıştır.

3.4.5. Gerilim (voltaj) bölücü

Prototipte farklı senaryolar için 9 volt ve 13,8 volt çıkış veren batarya kullanılmıştır.

Bataryadan gelen voltaj Arduino'nun analog pininden okunmaktadır. Arduino'da bulunan analog pinler 0-5 volt arası gerilimi 5/1024 hassasiyet ile alıp verebilmektedir [30]. Bu yüzden dört tane 10K ohm direnç kullanılarak bataryadan gelen voltaj dörde bölünecek şekilde bir gerilim bölücü devresi tasarlanmıştır. Toplam voltajın dörtte biri Arduino'nun analog pinine gönderilmiştir. Gerilim bölücü devresi Şekil 3.7 de verilmiştir.



Şekil 3.6. Gerilim bölücü devre

Bataryanın toplam voltaj değeri Formül (3.1), Formül (3.2), Formül (3.3), Formül (3.4), Formül (3.5) ve Formül (3.6) kullanılarak bulunmuştur.

$$V_{out}=I.R_4 \tag{3.1}$$

$$I=V_{out}/R_4 \tag{3.2}$$

$$V_{in}=I.R_1+I.R_2+I.R_3+I.R_4 \tag{3.3}$$

$$I=\frac{V_{in}}{R_1+R_2+R_3+R_4} \tag{3.4}$$

$$\frac{V_{out}}{R_4}=\frac{V_{in}}{R_1+R_2+R_3+R_4} \tag{3.5}$$

$$V_{in}=V_{out}.\frac{R_1+R_2+R_3+R_4}{R_4} \quad (3.6)$$

3.4.6. RTC modülü

RTC Real Time Clock'un kısalmasıdır. Yani gerçek zamanlı saat anlamına gelmektedir. RTC saat modülünde DS1302 entegresi bulunmaktadır. Modül üzerinde 3 V bir pil de bulunmaktadır. Bu pilin amacı saat tarih bilgisinin modüle yüklendikten sonra sürekli olarak saklanmasıdır. Bu pil çıkartıldığı zaman saat ve tarih bilgisi kaybolmaktadır. Pil çıkartılmadığı sürece saat ve tarih bilgisi saklanmaktadır. Bu modül içerisinde yıl, ay, gün, haftanın günü, saat, dakika ve saniye bilgilerini saklayabilmektedir ve anlık olarak bu bilgilere erişilebilmektedir [31].

Prototipte güncel olarak tarih saat bilgilerini tutabilmek için rtc modülü kullanılmıştır. Nextion ekranda tarih saat bilgilerinin gösterilmesi ve değiştirilebilmesi için ekran geliştirilmiştir. Bu ekran sonraki bölümlerde detaylı olarak açıklanmıştır.

Nextion ekranda geliştirilen ekran üzerinden girilen tarih saat bilgileri seri haberleşme ile Arduino Mega'ya gönderilmektedir. Arduino Mega'ya gönderilen tarih saat bilgileri ile rtc modülündeki tarih-saat bilgileri güncellenmektedir.

3.4.7. NodeMCU v3 modülü

NodeMCU v3 programlanabilir bir açık kaynak IoT platformudur. "Lua" olarak isimlendirilen programlama dili kullanılmaktadır. Arduino IDE programı ile programlanabilmektedir. Üzerinde ESP8266 wi-Fi modülü bulunmaktadır [32]. NodeMCU v3 Wi-Fi modülü şarj kontrol uygulamasındaki verilerin firebase cloud veri tabanına aktarılması için kullanılmıştır [33]. Mobil uygulama ve Arduino mikro denetleyicisi firebase veri tabanı üzerinden haberleşmektedirler.

3.4.8. Arduino Uno geliştirme kartı

Arduino Uno, ATmega328 mikro denetleyici içeren bir Arduino kartıdır. 14 adet Dijital giriş / çıkış pini bulunmaktadır. Bu dijital pinlerden 6 tanesi PWM (Pulse Width Modulation) çıkışını destekler. 6 adet analog giriş pini bulunmaktadır [34].

32 KB (kilobayt) Flash belleğe, 1 KB EEprom belleğe ve 2KB SRAM belleğe sahiptir. Arduino IDE programında C programlama dili kullanılarak programlanılabilmektedir. Arduino Mega üzerinde 1 tane UART(Evrensel Asenkron Alıcı Verici) bulunmaktadır.

Prototipte sensörlerden, bataryadan ve yakıttan okunan değerlere göre deniz aracı motorunun çalışmasına karar veren geliştirme Arduino Uno'da yapılmıştır.

Arduino Uno, üzerindeki digital ve analog pinlerinin yeterli olması ve maliyet açısından da diğer Arduino kartlara göre daha uygun olmasından dolayı tercih edilmiştir.

3.4.9. Arduino Mega geliştirme kartı

Arduino Mega, ATmega2560 mikro denetleyici içeren bir Arduino kartıdır. Arduino Mega'da 54 adet dijital giriş/çıkış pini bulunmaktadır. Bu dijital pinlerden 14′ü PWM çıkışı olarak kullanılabilmektedir. 16 adet analog giriş pini bulunmaktadır. 256 KB flash belleğe, 4 KB EEPROM belleğe ve 8 KB SRAM belleğe sahiptir.

Arduino Mega ve nextion ekran arasında seri haberleşme kullanılarak çift taraflı veri iletişimi sağlanmıştır. Nextion ekran arayüzlerinden yapılan işlemler Arduino Mega'ya iletilmektedir. Arduino Mega ve NodeMCU arasında da seri haberleşme kullanılarak çift taraflı veri iletişimi sağlanmıştır. Sensörlerden gelen verileri okuyarak nextion ekrana ve NodeMCU aracılığı ile firebase veri tabanına aktarmaktadır.

Prototipte Arduino Mega üzerinden 3 cihazla seri haberleşme yapılmaktadır. NodeMCU, Arduino Uno ve birçok sensörlerle bağlantılı olarak kullanılmıştır. Arduino Uno'daki dijital pinler ve uartlar yetersiz geldiği için Arduino Mega kullanılmıştır.

3.4.10. Nextion HMI ekran

Nextion ekranlar, insan ve proses, makina, uygulama ya da cihaz arasında kontrol ve görselleştirme arabirimi olarak görevli bir Seamless Human Machine Interface (HMI) çözümüdür [35].

Kullanıcının sistemdeki çıktıları anlayabileceği şekilde görebilmesi ve sisteme müdahale edebilmesi için bir ekran kullanma ihtiyacı olmuştur. Nextion ekran, Arduino ile uyumlu bir şekilde çalışabilmesi ve dokunmatik olarak kontrol edilmesinden dolayı tercih edilmiştir.

3.5. Geliştirme Ortamları

Birden fazla farklı ortamda geliştirme yapılmıştır. Bu bölüm altında geliştirme yapılan ortamlar ve yapılan geliştirmeler detaylı olarak anlatılmaktadır.

3.5.1. Arduino Mega kod geliştirme

Nextion ekrandaki tüm işlemler Arduino Mega üzerinden gerçekleştirilmiştir. Deniz aracındaki verilerin bulut ortamına aktarılması ve bulut ortamından verilerin alınıp nextion ekrana, Arduino Uno'ya gönderilmesi Arduino Mega ile sağlanmıştır. Arduino Mega'da yazılan kodlar Ek-A' da verilmektedir.

3.5.2. Arduino uno kod geliştirme

Deniz aracında yapılan kontrol ve işlemler için Arduino Uno'da geliştirme yapılmıştır. Yapılan işlemler Şekil 3.7'deki sözde kodlar ile ifade edilmektedir. Arduino Uno'da yazılan kodlar Ek-B' da verilmektedir.

3.5.3. Firebase cloud veri tabanı tasarımı

Verileri bulut ortamında tutmak için firebase cloud veri tabanında geliştirme yapılmıştır. Veri tabanı tasarımı Şekil 3.8'de gösterilmektedir.

3.5.4. NodeMCU v3 modülünde kod geliştirme

NodeMCU modülünde internete bağlanabilmek ve verileri firebase verit tabanına iletmek için geliştirme yapılmıştır. NodeMCU modülünde yazılan kodlar Ek-A'da verilmektedir.

3.5.5. Nextion HMI ekran geliştirmesi

Arduino Mega, Arduino Uno ve NodeMCU kartlarında gerçekleşen işlemlerin kullanıcıya anlaşılır gelmesi ve kullanıcının bu işlemlere müdahale etmesi gerektiği

zamanlarda kolaylıkla müdahale edebilmesi için nextion ekranda geliştirme yapılmıştır. Nextion Editör isimli uygulama üzerinden ekran tasarımları yapılarak nextion ekrana aktarılmıştır. Bu bölüm altında geliştirilen nextion ekran arayüzleri detaylı bir şekilde anlatılmaktadır.

1. BAŞLA
2. Yakıt deposunda kalan yakıt miktarını oku
3. Kalan yakıt miktarı kritik seviyede ise (motoru çalıştıramayacak kadar az);
"Yakıt miktarı kritik seviyededir!" mesajını gönder.
10 dakika bekle.
2. adıma git
4. Kontak açık ise 2. adıma git.
5. Bataryadaki kalan voltaj değerini oku.
6. Bataryada kalan voltaj değeri, ekrandan seçilen sistemi tetikleyecek voltaj değerinden küçük değil ise 2. adıma git
7. Kontak açık ise 2. adıma git.
8. Gaz kolu nötr pozisyonda değil ise;
"Gaz kolunu boş pozisyona alın" mesajını ver.
10 saniye bekle.
8. adıma git.
9. Kontak açık ise 2. adıma git.
10. Kontağa enerji gönder (Kontağı aç).
11. Gaz kolu nötr pozisyonda değil ise;
"Kontak açıkken gaz kolu boş pozisyonda olmalı!" mesajını ver.
Kontağa giden enerjiyi keserek kontağı kapat.
2. adıma git.
12. Dümeni kilitle.
13. 10 saniye bekle.
14. Marş dinamosuna enerji gönder.

Şekil 3.7. Şarj kontrol sisteminin sözde kodu

15. Marş dinamosu elektrik üretmedi ise 15. adıma git.

16. Marş dinamosuna giden enerjiyi kes.

17. Motorun çalışmaya başladığı zamana şimdiki zamanı ata.

18. Gaz kolunu boş pozisyonda değil ise;

"Kontak açıkken gaz kolu boş pozisyonda olmalı!" mesajını ver.

23. adıma git.

19. Motorun uygulama tarafından en son çalıştırıldıktan sonra durduğu tarihi oku.

20. Motorun en son durduğu tarih boş ise 22. adıma git.

21. Motorun en son durdurulduğu tarih ile şimdiki çalışmaya başladığı tarih arasında 30 dakikadan az zaman var ise;

"Akünün durumunu kontrol ediniz" mesajını ver.

22. Motor, kullanıcının ekran üzerinden girdiği motor çalıştırılma süresi kadar çalışmadı ise;

Gaz kolu nötr pozisyonda değil ise;

5 saniye bekle.

"Kontak açıkken gaz kolu boş pozisyonda olmalı!" mesajını ver.

23. adıma git.

Gaz kolu nötr pozisyonda ise 22. adıma git.

23. Kontak enerjisini kes (Kontağı kapat).

24. Dümen kilidini aç.

25. 5 saniye bekle.

26. Motorun en son durduğu zamana şimdiki zamanı ata.

27. Yakıt deposunda kalan yakıt miktarını oku.

28. Kalan yakıt miktarı düşük seviyede ise;

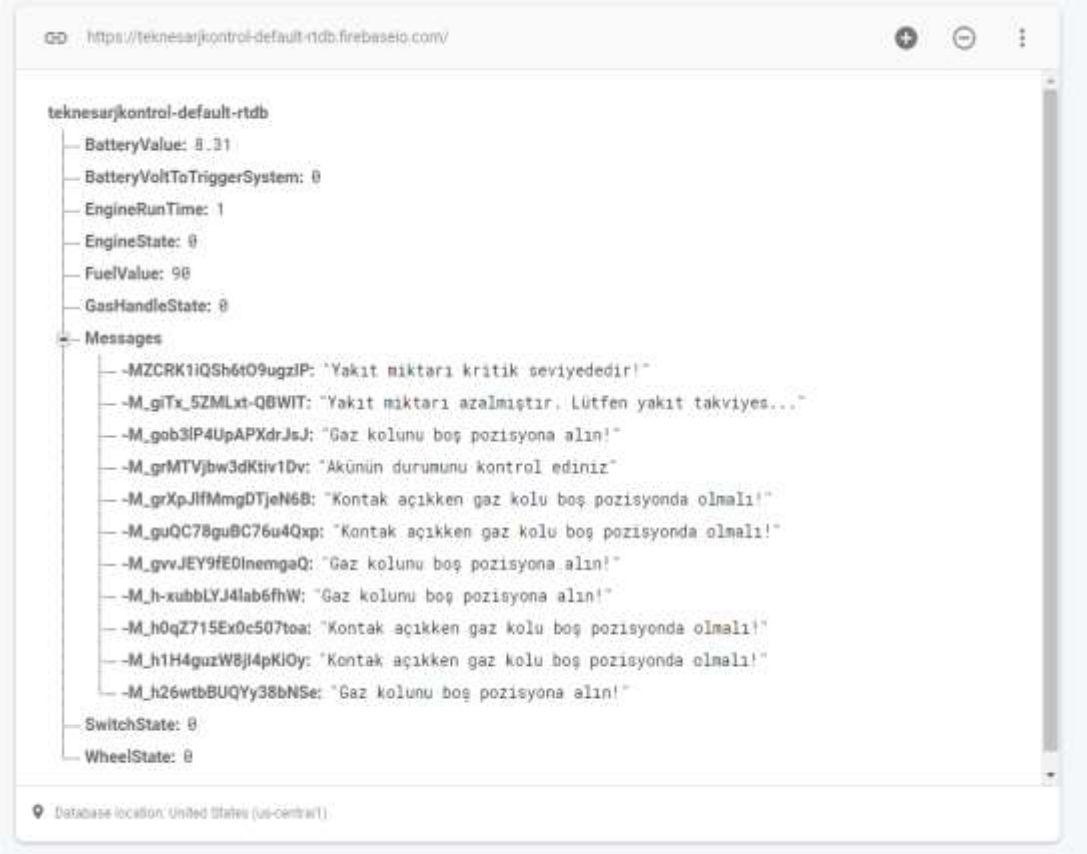
5 saniye bekle.

"Yakıt miktarı azalmıştır. Lütfen yakıt takviyesi yapınız!" mesajını ver.

29. 10 dakika bekle.

30. Adım 2'ye git.

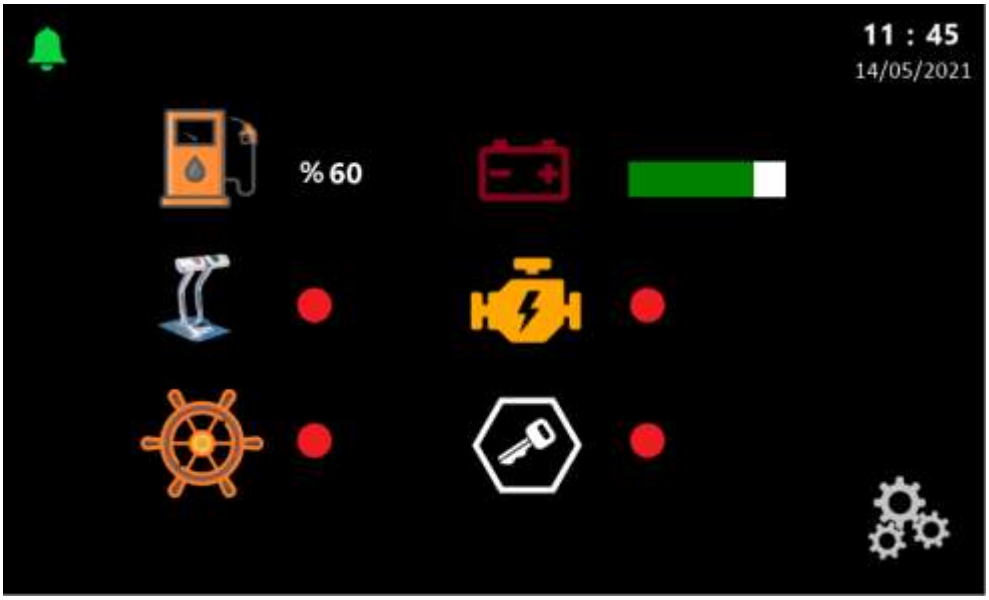
Şekil 3.7. (Devam) Şarj kontrol sisteminin sözde kodu



Şekil 3.8. Firebase cloud veri tabanı tasarımı

3.5.5.1 Anasayfa

Ana sayfa arayüzü Şekil 3.9'da gösterilmektedir.



Şekil 3.9. Nextion ana sayfa arayüzü

Ana sayfa ekranı özellikleri:

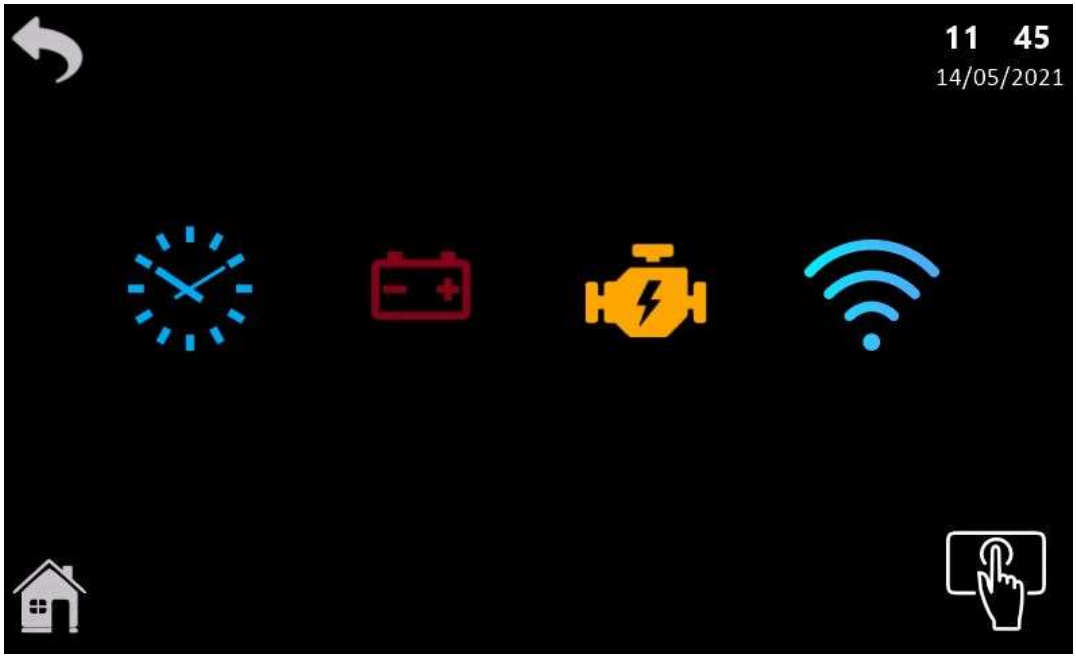
- Yakıt deposunda kalan yakıt miktarı yüzde olarak gösterilmektedir.
- Bataryada kalan voltaj gösterilmektedir.
- Gaz kolunun, dümen kilidinin ve kontağın durumu gösterilmektedir (Açık ya da kapalı).
- Motorun çalışıp çalışmadığı gösterilmektedir.
- Sol üst köşede bulunan zil ikonuna tıklandığında mesaj ekranı açılmaktadır.
- Sağ üst köşede rtc modülünden okunan saat ve tarih bilgisi gösterilir.
- Sağ alt köşede bulunan ayarlar ikonuna tıklandığında ayarlar ekranı açılmaktadır.

3.5.5.2. Ayarlar

Ayarlar arayüzü Şekil 3.10'da gösterilmektedir.

Ayarlar arayüzü özellikleri:

- Batarya ikonuna basıldığında batarya ile ilgili ayarlar ekranı açılmaktadır.
- Saat ikonuna tıklandığında saat ve tarih ayarlarının manuel yapıldığı ekran açılmaktadır.



Şekil 3.10. Nextion ayarlar arayüzü

- Motor ikonuna basıldığında motor ile ilgili ayarlar ekranı açılmaktadır.
- Wi-Fi ikonuna basıldığında Wi-Fi ile ilgili ayarlar ekranı açılmaktadır.

- Sağ alt köşedeki kontrol ikonuna tıklandığında kontrol ekranı açılmaktadır.
- Sol üst köşedeki geri ikonuna tıklandığında bir önceki ekran açılmaktadır.
- Sol alt köşedeki home ikonuna tıklandığında ana sayfa ekranı açılmaktadır.

3.5.5.3. Tarih-saat ayarları

RTC modülünden okunan tarih saat bilgisi bu ekranda gösterilmektedir. Tarih-saat arayüzü Şekil 3.11'de gösterilmektedir.

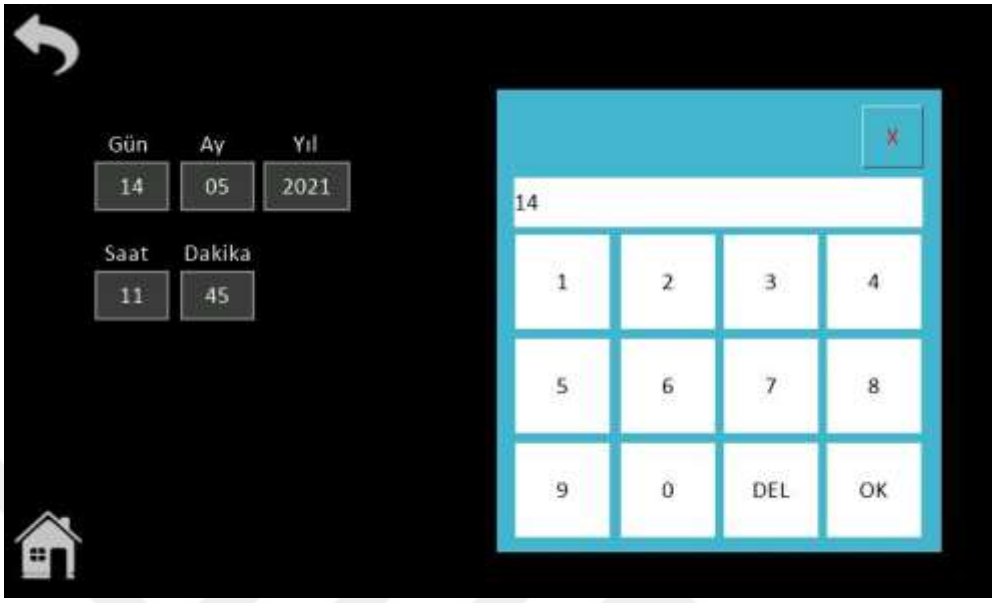
Kullanıcı buradan manuel olarak tarih ve saat bilgisinde değişiklik yapabilmektedir. Hangi değer değiştirilmek isteniyorsa o alana tıklanır. Tıklandıktan sonra Şekil 3.12'deki klavye açılmaktadır. Açılan klavye üzerinden seçilen bilgi değiştirildikten sonra "OK" butonuna basıldığında ilgili ekrana değişiklik yansımaktadır. Gün, ay, saat ve dakika girişinde en fazla 2 karaktere kadar giriş yapılabilmektedir. Yıl girişinde ise 4 karaktere kadar giriş yapılabilmektedir.

Şekil 3.11. Nextion tarih-saat arayüzü

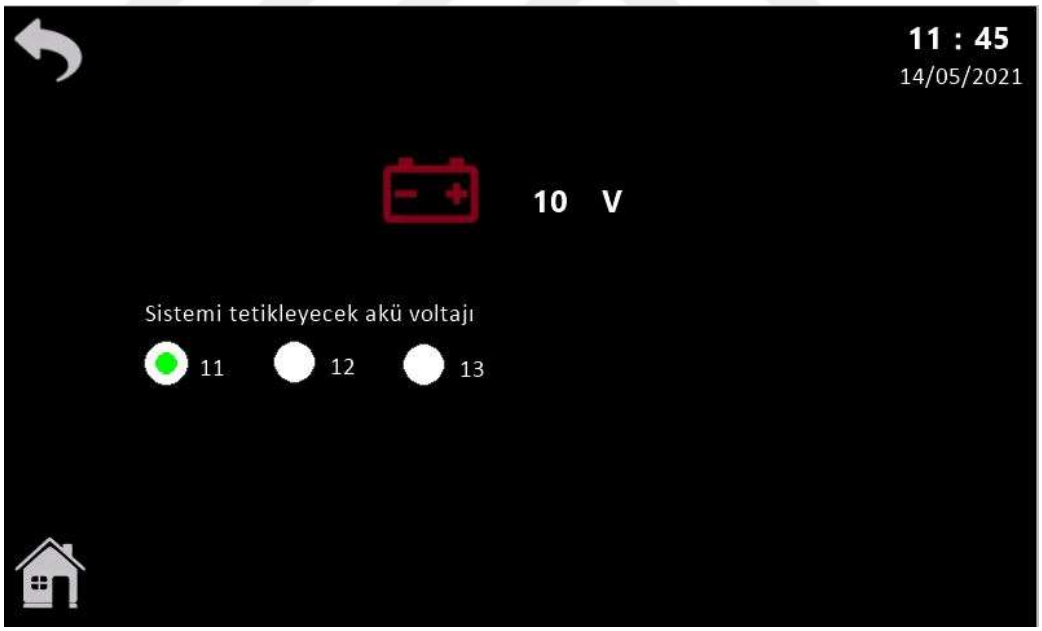
3.5.5.4. Batarya ayarları

Bataryadan anlık olarak okunan voltaj değeri ekranda gösterilmektedir. Bataryadaki voltaj değeri, kullanıcın bu ekrandan seçtiği değerin altına düştüğünde motor sistem

tarafından otomatik olarak çalıştırılmaktadır. Batarya arayüzü Şekil 3.13'de gösterilmektedir.



Şekil 3.12. Nextion tarih ve saat güncelleme



Şekil 3.13. Nextion batarya arayüzü

3.5.5.5. Motor ayarları

Motor ilk çalışmaya başladığı tarih bilgisi ve motorun sistem tarafından çalıştırıldığı en son tarih bilgisi bu ekranda gösterilmektedir. Motor sistem tarafından otomatik olarak çalıştırıldığında kaç dakika çalıştırılsın, bu ekrandan seçilmektedir. Bir dakika

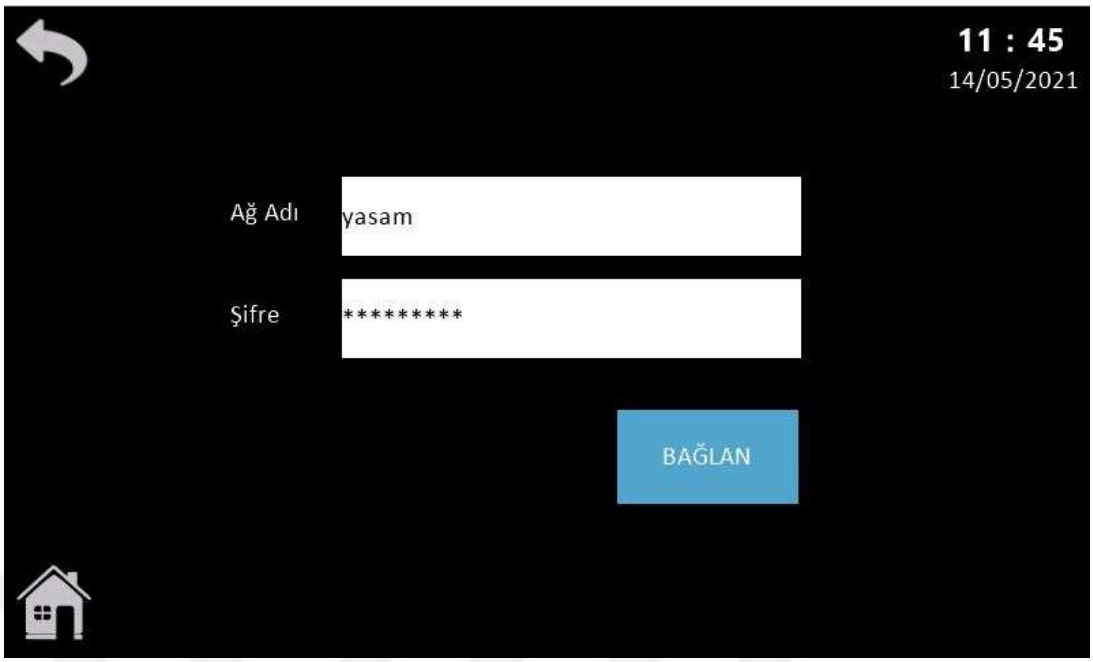
seçeneği prototipte kullanabilmek için eklenmiştir. Motor arayüzü Şekil 3.14'de gösterilmektedir.

Şekil 3.14. Nextion motor arayüzü

3.5.5.6. Ağ ayarları

Ağ bağlantısının olup olmadığı bilgisi Şekil 3.15'deki arayüzde gösterilmektedir. Kullanıcı bu arayüzde bulunana "Yeni Bağlantı" butonuna tıkladığında ağ bağlantısı yapabildiği ekran açılmaktadır (Şekil 3.16).

Şekil 3.15. Nextion ağ ayarları arayüzü (bağlantı yok)



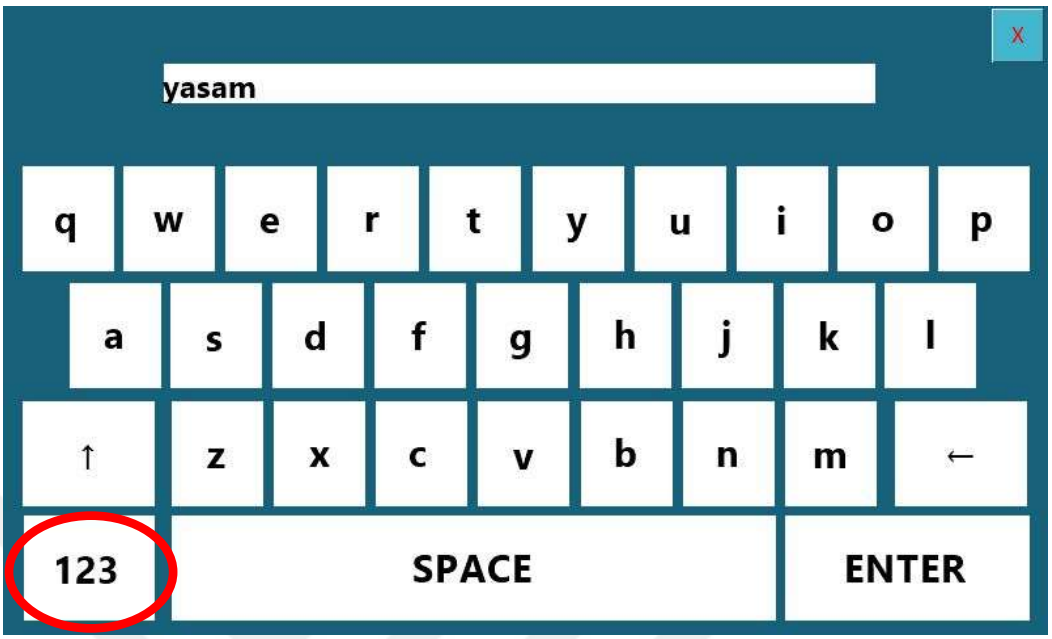
Şekil 3.16. Nextion yeni ağ bağlantı arayüzü

Kullanıcı ağ adı ve şifresini girip bağlan dedikten sonra ağ bağlantısının olup olmadığı bilgisinin gösterildiği ekran açılmaktadır. Bağlantı kurulursa bağlı olunan ağ adı ekranda gösterilmektedir (Şekil 3.17).

Şekil 3.17. Nextion ağ ayarları arayüzü (bağlantı var)

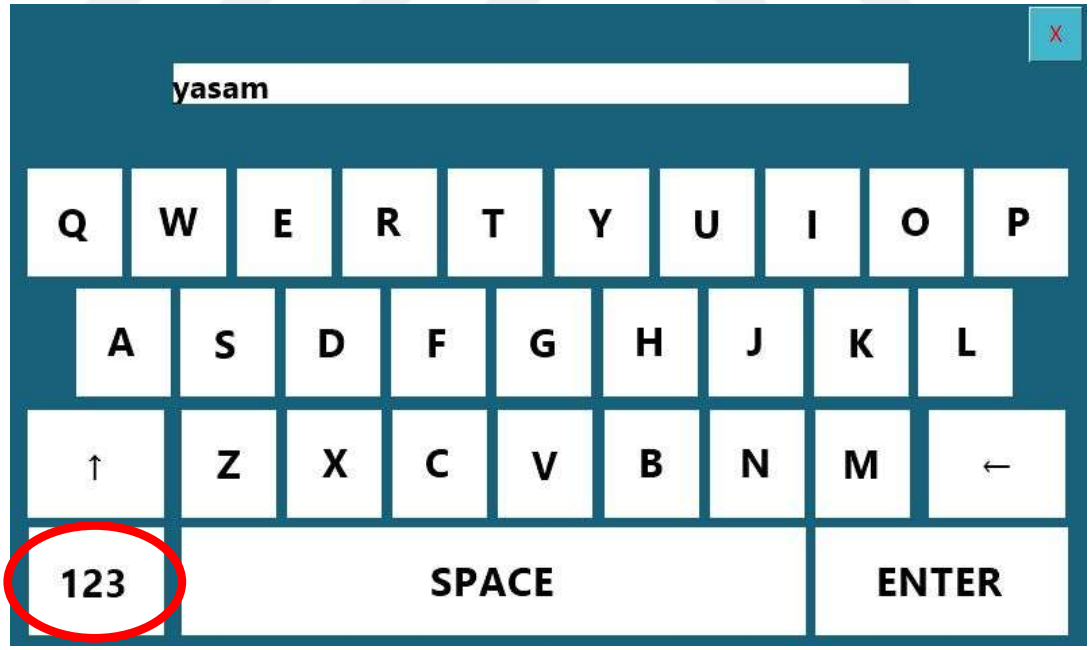
Kullanıcının ağ adı ve şifresi girebilmeleri için dört tane klavye ekranı tasarlanmıştır.

Yeni bağlantı tıklandığında Şekil 3.18 deki klavye ekranı açılmaktadır.



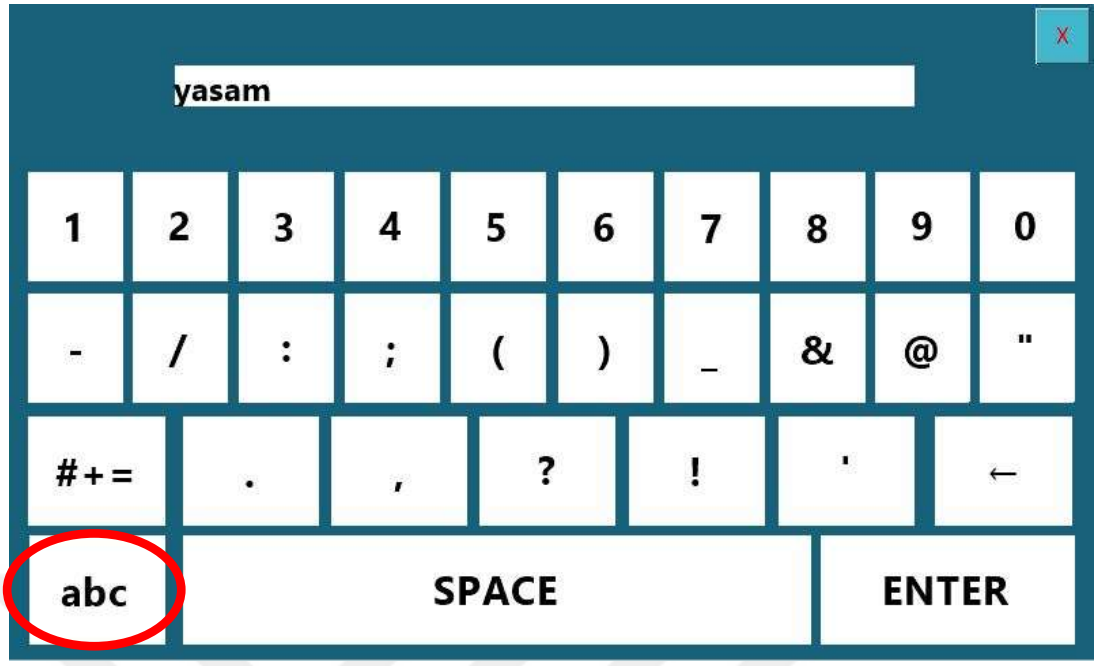
Şekil 3.18. Nextion birinci klavye

Seçili tuşa tıklandığında Şekil 3.19 deki klavye ekranı açılmaktadır.



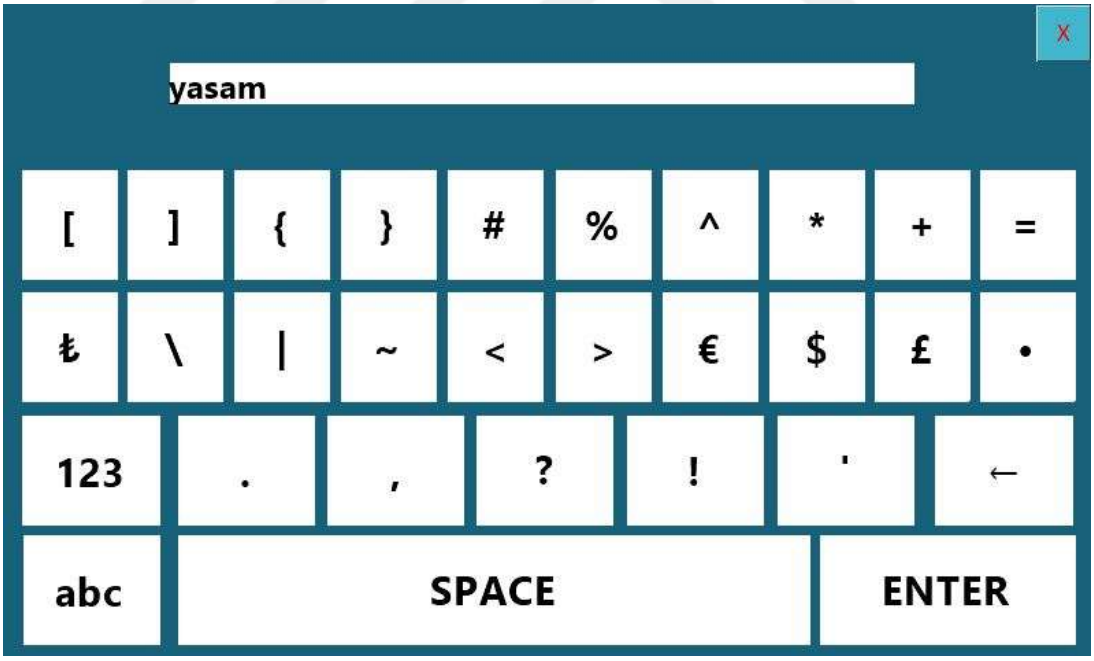
Şekil 3.19. Nextion ikinci klavye

Seçili tuşa basıldığında Şekil 3.20'deki klavye ekranı açılmaktadır.



Şekil 3.20. Nextion üçüncü klavye

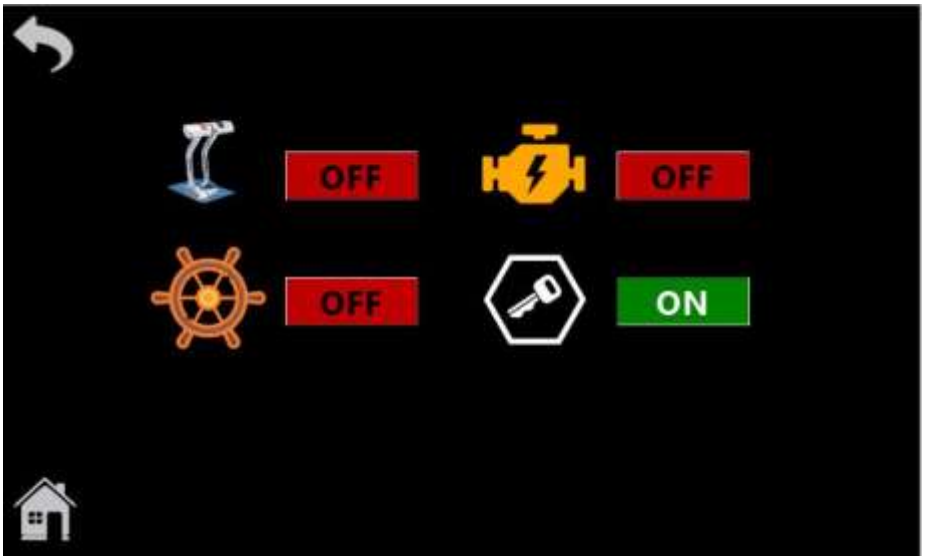
Seçili tuşa basıldığında Şekil 3.21'deki klavye ekranı açılmaktadır. Şekil 3.21.



Nextion dördüncü klavye

3.5.5.7. Kontrol

Kon trol arayüzü Şekil 3.22'de gösterilmektedir.



Şekil 3.22. Nextion kontrol arayüzü

Bu arayüz üzerinden;

- Gaz kolu pozisyonu değiştirme
- Dümen kilitlenip, kilidi açılma
- Kontak açılıp kapatılma
- Motor çalıştırılıp durdurulma işlemleri yapılabilmektedir.

3.5.5.8. Mesaj

Arduino'dan gelen mesajlar bu ekranda gösterilmektedir. Mesaj arayüzü Şekil 3.23'de gösterilmektedir.

Şekil 3.23. Nextion mesaj arayüzü

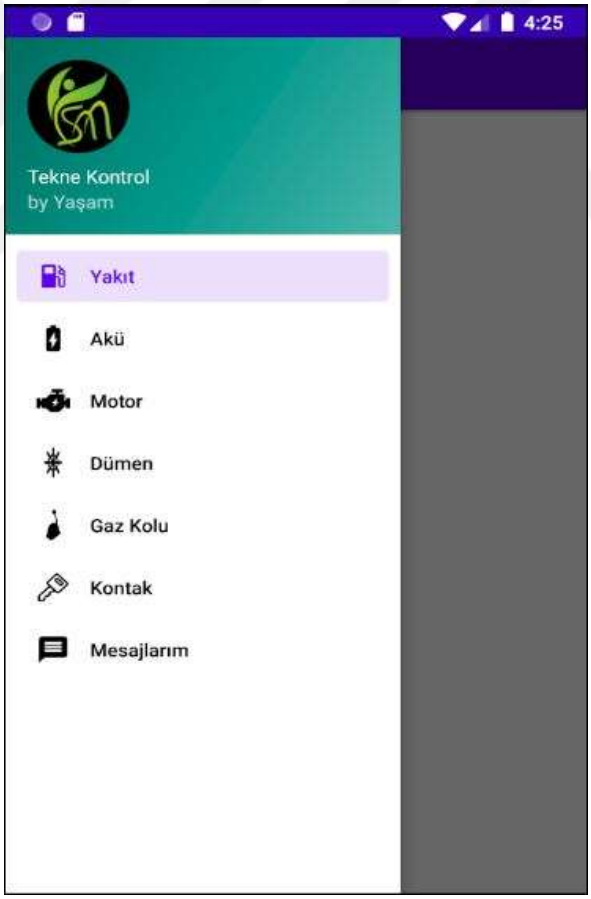
3.5.6. Mobil uygulama geliştirme

Kullanıcının deniz aracına uzaktan müdahale edebilmesi ve deniz aracında gerçekleşen durumları görebilmesi için mobil uygulama geliştirildi. Bu kısımda mobil uygulamada geliştirilen ekranlar detaylı bir şekilde anlatılmıştır.

3.5.6. Mobil uygulama geliştirme

Kullanıcının deniz aracına uzaktan müdahale edebilmesi ve deniz aracında gerçekleşen durumları görebilmesi için mobil uygulama geliştirildi. Bu kısımda mobil uygulamada geliştirilen ekranlar detaylı bir şekilde anlatılmıştır.

Mobil uygulamada geliştirilern tüm arayüzlerin listesi Şekil 3.24'de gösterilmiştir.

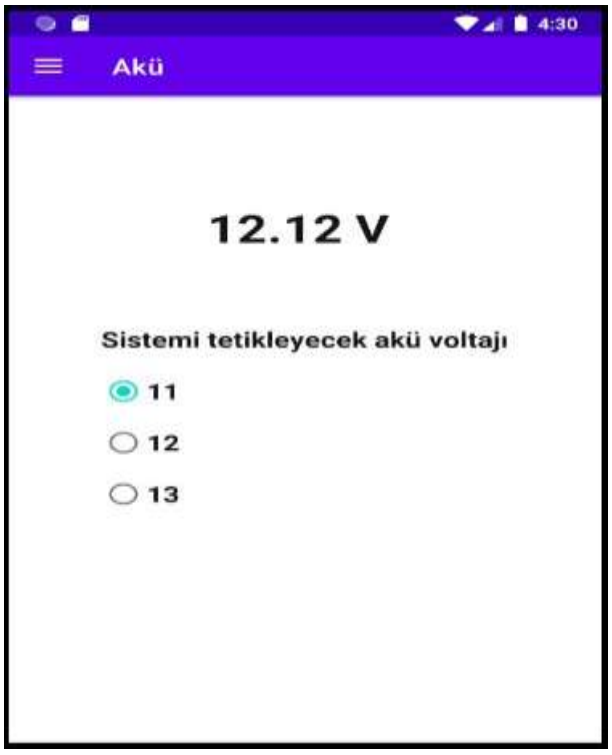


Şekil 3.24. Mobil uygulamada arayüzü listesi

Yakıt arayüzünde deniz aracının güncel yakıt miktarı gösterilmiştir. Yakıt arayüzü Şekil 3.25'de gösterilmiştir.

Şekil 3.25. Mobil uygulamada yakıt arayüzü

Akü arayüzünde deniz aracının bataryasındaki kalan şarj miktarı gösterilmiştir. Kullanıcı bu arayüz üzerinden sistemi tetikleyecek batarya voltajını seçebilmektedir. Batarya sayfası Şekil 3.26'da gösterilmiştir.



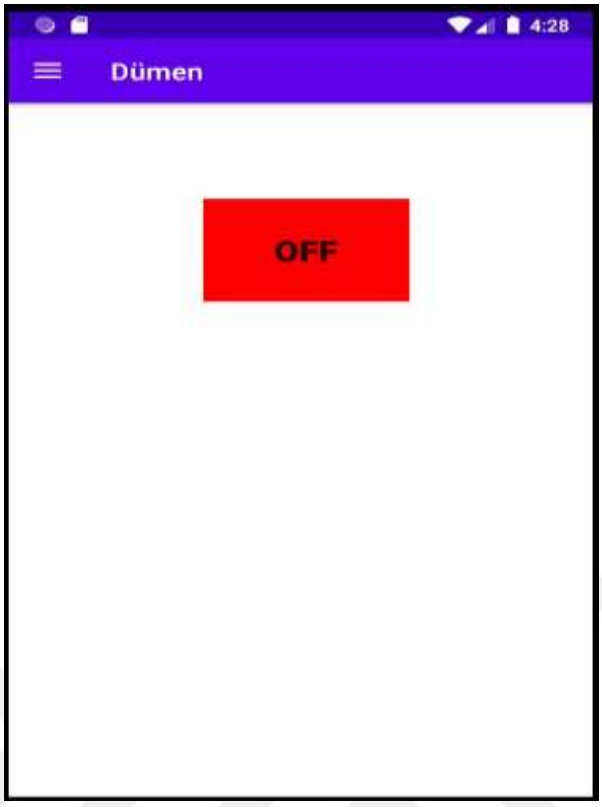
Şekil 3.26. Mobil uygulamada batarya arayüzü

Motor arayüzünde deniz aracı motorunun çalışma durumu gösterilmiştir. Motor durumu toggle buton kullanılarak gösterilmiştir. Toggle butona basılı iken motor çalıştırılmıştır, basılı değil iken motor durdurulmuştur. Kullanıcı bu arayüz üzerinden sistemin motoru çalıştırma süresini de seçebilmektedir. Motor arayüzü Şekil 3.27'de gösterilmiştir.

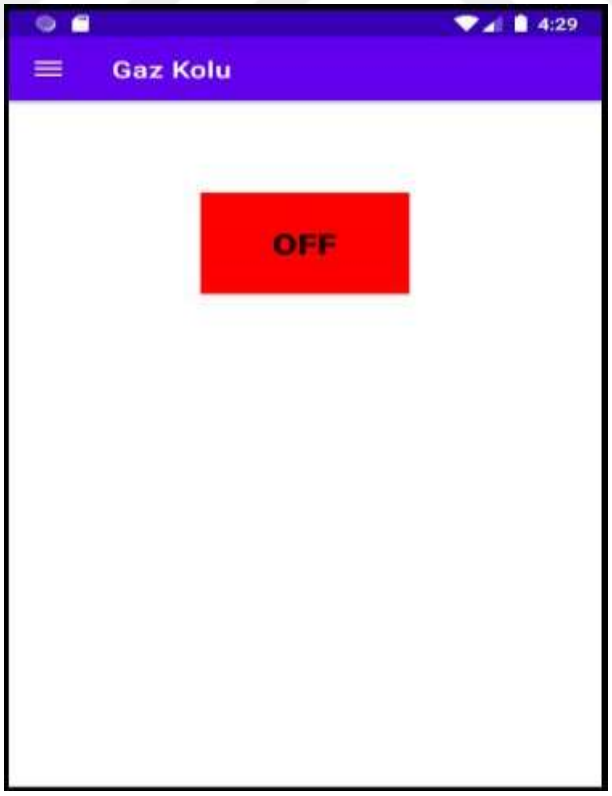
Dümen arayüzünde deniz aracı dümeninin durumu gösterilmiştir. Kullanıcı bu arayüz üzerinden dümeni kilitleyebilmekte ve dümen kilidini açabilmektedir. Dümen arayüzü Şekil 3.28'de gösterilmiştir.

Şekil 3.27. Mobil uygulamada motor arayüzü

Gaz kolu arayüzünde deniz aracındaki gaz kolunun durumu gösterilmiştir. Kullanıcı bu arayüz üzerinden gaz kolunun durumunu değiştirebilmektedir. Gaz kolu arayüzü Şekil 3.29'da gösterilmiştir.



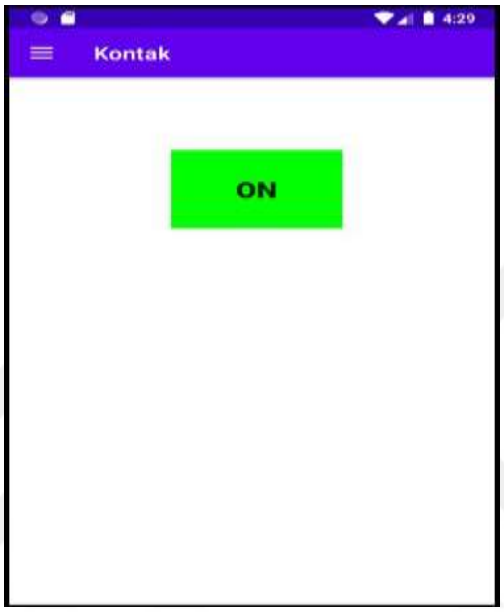
Şekil 3.28. Mobil uygulamada dümen arayüzü



Şekil 3.29. Mobil uygulamada gaz kolu arayüzü

Kontak arayüzünde kullanıcı deniz aracının kontak durumu gösterilmiştir. Kullanıcı bu arayüz üzerinden kontak durumunu değiştirebilmektedir. Kontak arayüzü Şekil 3.30 da gösterilmiştir.

Şarj kontrol sistemi devredeyken, Arduino Mega'dan gönderilen mesajların gösterildiği arayüz Şekil 3.31 de gösterilmiştir.



Şekil 3.30. Mobil uygulamada kontak arayüzü

Şekil 3.31. Mobil uygulamada mesaj arayüzü

4. DENEYSEL ÇALIŞMALAR

Çalışma kapsamında sistemin çalışmasını etkileyecek tüm durumlar göz önünde bulundurularak senaryolar hazırlanmıştır. Bu senaryolar hazırlanan prototip üzerinde uygulanmış ve sonuçlar gözlenmiştir.

4.1. Senaryolar ve Sonuçları

- Yakıt miktarı ekranda yüzdesel olarak gösterilmiştir. Yakıt deposu tam dolu iken sensörden okunan değer 630 dur ve ekranda %100 olarak gösterilmiştir.

- Kritik seviyedeki yakıt miktarı;

630 / 8=78.75 dir. Bunun yüzde olarak değeri %12.50 dir.

- Düşük seviyedeki yakıt miktarı;

630 / 4=157.5 dir. Bunun yüzde olarak değeri %24 dir.

1. Yakıt miktarı kritik seviyededir;
"Yakıt miktarı kritik seviyededir!" mesajı verilmiş ve devam edilmemiştir. Başa dönülmüştür.

2. Yakıt miktarı kritik seviye değildir. Kontak açılmıştır;
Devam edilmemiş ve başa dönülmüştür.

3. Yakıt miktarı kritik seviye değildir (okunan değer > %12.50). Kontak kapatılmıştır. Ekrandan sistemi tetikleyecek batarya voltaj değeri 11 olarak seçilmiştir. Bataryadan okunan anlık voltaj değeri 11 in üzerindedir.
Devam edilmemiş ve başa dönülmüştür;

4. Yakıt miktarı kritik seviye değildir. Kontak açılmıştır. Ekrandan sistemi tetikleyecek batarya voltaj değeri 12 olarak seçilmiştir. Bataryadan okunan anlık voltaj değeri 12'nin üzerindedir;
Devam edilmemiş ve başa dönülmüştür.

5. Yakıt miktarı kritik seviye değildir. Kontak kapatılmıştır. Ekrandan sistemi tetikleyecek batarya voltaj değeri 12 olarak seçilmiştir. Bataryadan okunan anlık voltaj değeri 12 nin altındadır. Gaz kolu boş pozisyondan çıkartılmıştır;

"Gaz kolunu boş pozisyona alın" mesajı verilmiştir. Ve gaz kolu boş pozisyona alınana kadar beklenmiştir;

- Gaz kolu boş pozisyona alınmıştır.

- Sistem tarafından kontak açılmıştır.

- Kontak açıkken gaz kolu boş pozisyondan çıkartılmıştır.

- "Kontak açıkken gaz kolu boş pozisyonda olmalı" mesajı verilmiş ve kontak sistem tarafından kapatılmıştır. Bundan sonra devam edilmemiş ve başa dönülmüştür.

6. Yakıt miktarı kritik seviye değildir. Kontak kapatılmıştır. Ekrandan sistemi tetikleyecek batarya voltaj değeri 12 olarak seçilmiştir. Bataryadan okunan anlık voltaj değeri 12 nin altındadır. Gaz kolu boş pozisyona alınmıştır. Motorun bir önceki çalışma zamanı boş değildir. Motorun bir önceki durduğu zaman ile şimdiki çalışmaya başladığı zaman arasında 30 dakikadan daha az bir süre vardır. Motor çalışma süresi 1 dakika olarak seçilmiştir;

- Kontak açılmıştır.
- Dümen kilitlenmiştir.
- Motor çalıştırılmıştır.
- Motorun çalışmaya başladığı zamana şimdiki zaman atanmıştır.
- "Akünün durumunu kontrol ediniz" mesajı verilmiştir ve devam edilmiştir.
- Motor 1 dakika çalıştıktan sonra;

- Kontak enerjisi kesilmiştir.

- Motorun durduğu zamana şimdi zaman atanmıştır.

- Dümen kilidi açılmıştır.

Bu işlemlerden sonra başa dönülmüştür.

7. Yakıt miktarı kritik seviye değildir. Kontak kapatılmıştır. Ekrandan sistemi tetikleyecek batarya voltaj değeri 12 olarak seçilmiştir. Bataryadan okunan anlık voltaj değeri 12 nin altındadır. Gaz kolu boş pozisyona alınmıştır. Motorun bir önceki çalışma zamanı boş değildir. Motorun bir önceki durduğu zaman ile şimdiki çalışmaya başladığı zaman arasında 30 dakikadan daha az bir süre vardır. Motor

çalışma süresi 1 dakika olarak seçilmiştir. Motor çalıştırılıp durdurulduktan sonra kalan yakıt miktarı düşük seviyededir;

- Kontak açılmıştır.
- Dümen kilitlenmiştir.
- Motor çalıştırılmıştır.
- Motorun çalışmaya başladığı zamana şimdiki zaman atanmıştır.
- "Akünün durumunu kontrol ediniz" mesajı verilmiştir ve devam edilmiştir.
- Motor 1 dakika çalıştıktan sonra;

- Kontak enerjisi kesilmiştir.
- Motorun durduğu zamana şimdiki zaman atanmıştır.
- Dümen kilidi açılmıştır.

* Kontak enerjisi kesilmiştir.
* Motorun durduğu zamana şimdiki zaman atanmıştır.
* Dümen kilidi açılmıştır.
* Kalan yakıt miktarı %24 ün altındadır. "Yakıt miktarı azalmıştır. Lütfen yakıt takviyesi yapınız!" mesajı verilmiştir.

Bu işlemlerden sonar başa dönülmüştür.

8. Yakıt miktarı kritik seviye değildir. Kontak kapatılmıştır. Ekrandan sistemi tetikleyecek batarya voltaj değeri 12 olarak seçilmiştir. Bataryadan okunan anlık voltaj değeri 12 nin altındadır. Gaz kolu boş pozisyona alınmıştır. Motorun bir önceki çalışma zamanı boş değildir. Motorun bir önceki durduğu zaman ile şimdiki çalışmaya başladığı zaman arasında 30 dakikadan daha fazla bir süre vardır. Motor çalışma süresi 1 dakika olarak seçilmiştir;

- Kontak açılmıştır.
- Dümen kilitlenmiştir.
- Motor çalıştırılmıştır.
- Motorun çalışmaya başladığı zamana şimdiki zaman atanmıştır.
- Motor 1 dakika çalıştıktan sonra;

- Kontak enerjisi kesilmiştir.
- Motorun durduğu zamana şimdiki zaman atanmıştır.
- Dümen kilidi açılmıştır.

Bu işlemlerden sonra başa dönülmüştür.

9. Yakıt miktarı kritik seviye değildir. Kontak kapatılmıştır. Ekrandan sistemi tetikleyecek batarya voltaj değeri 12 olarak seçilmiştir. Bataryadan okunan anlık voltaj değeri 12 nin altındadır. Gaz kolu boş pozisyona alınmıştır. Motorun bir önceki çalışma zamanı boş değildir. Motorun bir önceki durduğu zaman ile şimdiki çalışmaya başladığı zaman arasında 30 dakikadan daha fazla bir süre vardır. Motor çalışma süresi 1 dakika olarak seçilmiştir. Motor çalışmaya başladığında gaz kolu boş pozisyondan çıkartılmıştır.

- Kontak açılmıştır.
- Dümen kilitlenmiştir.
- Motor çalıştırılmıştır.
- Motorun çalışmaya başladığı zaman şimdi zaman atanmıştır.

- Gaz kolu boş pozisyondan çıkartıldıktan sonra "Kontak açıkken gaz kolu boş pozisyonda olmalı!" mesajı verilmiştir.

* Kontak enerjisi kesilmiştir.
* Motorun durduğu zamana şimdi zaman atanmıştır.
* Dümen kilidi açılmıştır.

Bu işlemlerden sonra başa dönülmüştür.

5. SONUÇLAR VE ÖNERİLER

Bu tez çalışmasında Deniz araçlarında batarya ömrünü uzatmak ve bataryayı verimli bir şekilde kullanabilmek için bir uygulama geliştirilmiştir. Uygulamada kullanılan tüm cihazların performansı ve uygulamanın batarya üzerindeki etkileri incelenmiştir.

İlk olarak uygulamada sadece bir tane arduino uno geliştirme kartı kullanılmıştır. Fakat bu geliştirme kartındaki pinlerin ve kartın hafızasının yetersiz geldiği görülmüştür. Bu yüzden daha fazla pin girişine ve daha büyük hafızaya sahip arduino mega geliştirme kartı kullanılmıştır.

Arduino mega ile iletişim halinde olan nextion ekran üzerinden yapılan tüm işlemler arduino megaya gönderilmiştir. Nextion ekrandan gelen verilere göre arduino megada işlemler yapılmıştır. Arduino Mega nextion ekrandan gelen işlemleri yaparken diğer yandan da deniz aracının bataryasındaki voltaj değerini ve diğer bağlı sensörlerden gelen verileri okumaktadır. Bilindiği üzere arduino geliştirme kartlarında işlemler sırasıyla yapılmaktadır. Sadece kesme pinlerine gelen sinyal ile o anki işlemden kesme işlemine gitmekte, kesme işlemindeki iş bittiğinde tekrar kaldığı yere dönmektedir. Bu sebebten dolayı 2 tane geliştirme kartı kullanmaya karar verilmiştir. Bu kartlar arduino uno ve arduino mega geliştirme kartlarıdır. Bu sayede ekrandan gelen işlemler ve deniz aracı bataryasını çalıştırmak için yapılan işlemler birbirlerinden bağımsız olarak yapılmıştır. Sonuç olarak birbirlerinin işlemlerinde beklemeye sebep olmadıkları görülmüştür.

Uygulamada verilerini bulut ortamındaki veri tabanına aktarmak için ilk olarak ESP8266 Wi-Fi modülü kullanılmıştır. Fakat bu Wi-Fi modülünden iyi verim alınamamıştır. Sağlıklı bir internet bağlantısı sağlanamamıştır. Bu yüzden bulut ortamındaki veri tabanı ile veri alışverişini sağlamak için NodeMCU v3 Wi-Fi modülü kullanılmaya karar verilmiştir. NodeMCU ile internet bağlantısında ve bulut ortamındaki veri tabanı ile çift taraflı veri alışverişinde herhangi bir sorun yaşanmamıştır.

Geliştirilen uygulamadaki tüm işlemler test edilmiştir. Herhangi bir hataya rastlanmamıştır. Deniz aracı bataryasını şarj etmek için yapılan geliştirme için ise senaryolar hazırlanmıştır. Hazırlanan senaryolar prototip üzerinde uygulanmış ve olması gereken sonuçların geldiği görülmüştür.

Arduino mega geliştirme kartında nextion ekrandaki özelliklere ulaşabilmek ve bunlara müdahele edebilmek için nextion kütüphanesi kullanılmıştır. Bu kütüphanede ekran üzerindeki işlemlerde gecikme yaşandığı görülmüştür. Örnek olarak kontrol arayüzündeki motor durumu butonuna basıldıktan 2 saniye sonra motor çalışmaya başlamış ve ekranda motor durum ikonu çalışıyor olarak değişmiştir. Buradaki performans sorununu ortadan kaldırmak ve sonuçların anlık olarak yansıması için uygulamadaki hazır nextion kütüphanesinin kaldırılması planlanmaktadır. Kütüphane kullanılmadan nextion ekrandaki işlemler için kod geliştirmesi yapılacaktır.

Tüm sonuçlar değerlendirildiğinde deniz aracı bataryası geliştirilen uygulama tarafından düzenli olarak şarj edilmiştir ve bataryadan yüksek verim alınmıştır. Kullanılan arayüzler sayesinde kullanıcıya her şey anlaşılır gelmiştir ve kullanıcının sisteme müdahale etmesi kolaylaştırılmıştır. Bu çalışmanın gerçek bir deniz aracına uygulanarak deniz aracındaki sonuçlarının gözlemlenmesi planlanmaktadır. Prototipte düşük maliyetli sensor, modül ve ek ekipmanlar kullanılmıştır. Gerçek hayatta bir deniz aracında uygulamaya geçildiğinde donanım performansını artırmak için sistemde kullanılan sensor, modül ve ek ekipmanlarla aynı görevde fakat daha yüksek performans ve verim alınabilecekler araştırılarak tercih edilecektir. Ayrıca geliştirmeye gaz, sıcaklık ve nem, hareket gibi sensörleri ve gps modülünün eklenmesi planlanmaktadır. Bu sayede kullanıcıya çok özellikli bir uygulama sunulması hedeflenmektedir.

KAYNAKLAR

[1] Ustaoglu Y., Küçük K., Deniz Araçları için Nesnelerin İnterneti Teknolojileri Temelli Şarj Kontrol Sistemi tasarımı, *6. Uluslararası Mühendislik ve Tasarım Kongresi (MMT)*, İstanbul, Türkiye, 17-18 Aralık 2020.

[2] https:// www.garantibbva.com.tr/tr/blog/teknolojik-gelismeler.page (Ziyaret tarihi : 20 Mayıs 2021)

[3] Gupta A. K., Johari R., IOT based Electrical Device Surveillance and Control System, *2019 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU)*, Ghaziabad, India, 18-19- Nisan 2019.

[4] Karakuş K., Yeşilyurt B., Eren T., Sağlık Sektöründe IoT Uygulamalarının Analitik Ağ Süreci Yöntemi ile Değerlendirilmesi, *Samsun Sağlık Bilimleri Dergisi*, 2019, **4**(2), 86-92.

[5] Liao M., Dong L., Jin lu., Wang S., Study on Rotational Speed Feedback Torque Control for Wind Turbine Generator System, *2009 International Conference on Energy and Environment Technology*, Guilin, China, 16-18 Ekim 2009.

[6] Shahinzadeh H., Moradi J., Gharehpetian G. B., Fathi S. H., Abedi M., Green Power Island, a Blue Battery Concept for Energy Management of High Penetration of Renewable Energy Sources with Techno-Economic and Environmental Considerations, *2018 Smart Grid Conference (SGC),* Sanandaj, İran, 28-29 Kasım 2018.

[7] Muni T. V., Pranav A. S., Srinivas A. A., IoT Based Smart Battery Station Using Wireless Power Transfer Technology, *International Journal of Scientific & Technology Research*, 2020, **9**(1), 31-35.

[8] Ashton K. That 'Internet of Things' thing in the real world, things matter more than ideas. RFID Journal.,http://www.rfidjournal.com/, (Ziyaret tarihi: 21 Nisan 2021)

[9] Kutup N., Nesnelerin interneti; 4H Her yerden, Herkesle, Her zaman, Her nesne ile bağlantı, *16. Türkiye'de İnternet Konferansı inettr'11*, İzmir, Türkiye, 30 Kasım-2 Aralık 2011.

[10] Weber R.H., Internet of things - new security and privacy challenges, *Computer Law and Security Review*, 2010, **26**(1), 23-30.

[11] Ammar M., Russello G. ve Crispo B., Internet of Things: A survey on the security of IoT frameworks, Journal of Information Security and Applications, 2018, **38**, 8-27.

[12] Atzori L., Lera A., Morabito G., The internet of things: A survey, *Computer Networks*, 2010, **54** (15), 2787–2805.

[13] Bozuklu M., Çevresel veriler ile Gerçek Zamanlı Nesnelerin İnterneti Uygulaması, Yüksek Lisans Tezi, Gaziosmanpaşa Üniversitesi, Fen Bilimleri Enstitüsü, Tokat, 2016, 436403.

[14] http://www.beechamresearch.com/article.aspx?id=4 (Ziyaret tarihi: 10 Ocak 2021)

[15] https://www.7deniz.net/haber-nesnelerin-interneti-ile-denizleri-asacak-21494.html (Ziyaret tarihi : 10 Ocak 2021)

[16] Sakphrom S., Korkua S., Simplified Stream Discharge Estimation for Hydrological Application based on NB-IoT Deployment, *2019 10th International Conference of Information and Communication Technology for Embedded Systems (IC-ICTES)*, Bangkok, 25-27 Mart 2019.

[17] Kumar N., Sundaram K., Anusuya, IoT Based Smart Charger: An ESP8266 Based Automatic Charger, *BDIOT2017: Proceedings of the International Conference on Big Data and Internet of Thing*, London United Kingdom, 20-22 Aralık 2017.

[18] K. Lai, F. Cheng, S. T. Chou, Y. Chang, G. Wu and J. Tsai, AnyCharge: An IoT-Based Wireless Charging Service for the Public, *IEEE Internet of Things Journal*, **6**(6), 10888-10901

[19] Harish N., Prashal V., Sivakumar D., IOT Based Battery Management System, *International Journal of Applied Engineering Research ISSN 0973-4562*, 2018, **13**(8), 5711-5714

[20] L. Wang et al., Installation of a 400-W wind turbine generator on a commercial fishing boat to achieve energy saving, *IEEE PES General Meeting*, Minneapolis, MN, USA, 25-29 Temmuz 2010.

[21] Nado M., Taggu A., Intelligent Automated Smart Solar Panel Using Internet Of Things, *Proceedings of the 5th International Conference on Computers & Management Skills (ICCM) | North Eastern Regional Institute of Science & Technology (NERIST)*, Nirjuli, Arunachal Pradesh, India, 9 ocak 2020.

[22] Chao R., Lin H., Wu C., Solar-powered boat design using standalone distributed PV system, *2018 IEEE International Conference on Applied System Invention (ICASI)*, Chiba, Japan,13-17 Nisan 2018.

[23] Leung C. P., Cheng K. W. E., Zero emission solar-powered boat development, *2017 7th International conference on Power Electronics Systems and Applications – Smart Mobility, Power Transfer & Security (PESA)*, Hong Kong, China, 12-14 Aralık 2017.

[24] Fathabadi H., Novel photovoltaic based battery charger including novel high efficiency step-up DC/DC converter and novel high accurate fast maximum power point tracking controller, *Energy Conversion and Management Research ISSN 0196-8904*, 2016, **110**, 200-211.

[25] Debashish M., Subhransu P., Jhansirani J., Design of Solar Powered Battery Charger: An Experimental Verification, *2018 IEEE International Student's Conference on Electrical, Electronics and Computer Science (SCEECS)*, Bhopal, India, 24-25 Şubat 2018.

[26] Şener A. Ş., Aktaş A., Kırçiçek Y., The Assessment of Wind and Sea Flow Energy Production from Seas by Using Energy Storage Unit, *2018 7th International Conference on Renewable Energy Research and Applications (ICRERA)*, Paris, France, 14-17 Ekim 2018

[27] https://maker.robotistan.com/arduino-yagmur-sensoru-alarmi/ (Ziyaret tarihi: 14 Nisan 2021)

[28] https://akademi.robolinkmarket.com/dc-motor-nedir-ne-ise-yarar-arduino-ile-nasil-kullanilir/(Ziyaret tarihi: 15 Nisan 2021)

[29] https://www.elektrikde.com/servo-motor-nedir-calisma-prensibi/ (Ziyaret tarihi: 15 Nisan 2021)

[30] https://diyot.net/arduino-uno-pinleri/(Ziyaret tarihi: 15 Nisan 2021)

[31] https://akademi.robolinkmarket.com/rtc-saat-modulu-nedir-nasil-kullanilir/ (Ziyaret tarihi : 05 Ocak 2021)

[44] Ayyıldız M., Denizli M., Nesnelerin İnterneti ile Akıllı bir Priz Prototipi, *Düzce Üniversitesi Bilim ve Teknoloji Dergisi*, 2019, 7(1), 722-728.

[33] Bayılmış C., Küçük K., Nesnelerin İnterneti: Teori ve Uygulamaları, 1rd ed., Papatya Yayıncılık, İstanbul, 2019

[34] Özerdem M.S., Cengiz R., GSM Tabanlı Çoklu Takip Sistem Uygulaması, *Dicle Üniversitesi Mühendislik Fakültesi Mühendislik Dergisi*, 2018, 9(1), 153-160.

[35] https://www.direnc.net/2-4-inch-nextion-hmi-dokunmatik-tft-lcd-ekran-4mb (Ziyaret tarihi : 15 Nisan 2021)

EKLER

Ek-A

Arduino Mega içerisine aktarılan kodlar verilmiştir (tekneSarjKontrol_nextion.ino).

```
#include <virtuabotixRTC.h>
#include "TimeLib.h"
#include "Nextion.h"
#include <Wire.h>
#include <EEPROM.h>
enum DataType
{
 NOTHING,
 ENGWORKINGTIME,
 BATTRIGVOLT,
 CONNECTWIFI,
 ENGSTOPPEDDATE,
 MESSAGE,
 DATETIMENOW,
 CURRFUELVAL,
 ENGSTRTDDATE,
 WHEEL_ST,
 ENGINE_ST,
 SWITCH_ST,
 GASHANDLE_ST,
 BATTRYVAL
} dataType;
enum MessageType
{
 FUEL_CRITICAL,
 TAKE_NOTR_GASHANDLE,
 SWITCH_OPEN_GASHANDLE_NOTR,
 CONTROL_AKU_STATE,
 FUEL_LOW
};
```

```
enum BatteryTrigOptions
{
 batOpt11 = 11,
 batOpt12 = 12,
 batOpt13 = 13
};
enum EngWorkOptions
{
 engOpt1 = 1,
 engOpt10 = 10,
 engOpt15 = 15
};
static const char *MessageType_String[] = {
 "Yakıt miktarı kritik seviyededir!",
 "Gaz kolunu boş pozisyona alın!",
 "Kontak açıkken gaz kolu boş pozisyonda olmalı!",
 "Akünün durumunu kontrol ediniz",
 "Yakıt miktarı azalmıştır. Lütfen yakıt takviyesi yapınız."
};
class DateTime {

   uint8_t second;
   uint8_t minute;
   uint8_t hour;
   uint8_t day;
   uint8_t month;
   uint16_t year;
 public:
   void setDateTime( uint8_t _second, uint8_t _minute , uint8_t _hour , uint8_t
_day, uint8_t _month, uint16_t _year)
   {
     second = _second;
```

```
    minute = _minute;
    hour = _hour;
    day = _day;
    month = _month;
    year = _year;
  }
public:
  void setSecond(uint8_t _second) {
    second = _second;
  }
public:
  uint8_t getSecond() {
    return second;
  }
public:
  void setMinute(uint8_t _minute) {
    minute = _minute;
  }
public:
  uint8_t getMinute() {
    return minute;
  }
public:
  void setHour(uint8_t _hour) {
    hour = _hour;
  }
public:
  uint8_t getHour() {
    return hour;
  }
public:
  void setDay(uint8_t _day) {
    day = _day; }
```

```
  public:
    uint8_t getDay() {
      return day;
    }
  public:
    void setMonth(uint8_t _month) {
      month = _month;
    }
  public:
    uint8_t getMonth() {
      return month;
    }
  public:
    void setYear(uint16_t _year) {
      year = _year;
    }
  public:
    uint16_t getYear() {
      return year;
    }
};
virtuabotixRTC boatRTC(22, 23, 24);
#define WHEEL_PIN 2
#define CONN_STATE 25
#define SWITCH_PIN 26
#define GAS_HANDLE_PIN 27
#define ENGINE_PIN 28
#define FUEL_PIN A0
#define BATTERY_PIN A1
#define DEVICE_ID_UNO 1
DateTime* _engFrstStrtDt = NULL;
DateTime* _engStrtdDt = NULL;
```

```
DateTime* _engStppdDt = NULL;
uint8_t _hourAddr = 0;
uint8_t _minuteAddr = 1;
uint8_t _secondAddr = 2;
uint8_t _dayOfMonthAddr = 3;
uint8_t _monthAddr = 4;
uint8_t _yearLowAddr = 5;
uint8_t _yearHighAddr = 6;
const uint16_t _fllFuelVal = 630;
char *_ssid = NULL;
char *_wifiPassword = NULL;
char _batteryValue[5] = {0};
bool _connState = false;
uint32_t _pageDateSett = 0;
uint32_t _pageEngSett = 0;
uint32_t _pageBatSett = 0;
uint32_t _pageChnge = 0;
uint8_t  _engWorkTime = 1;
uint8_t _batVoltToTrigSys = 11 ;
float _curBatVolt = 0;
String _message = "";
char _open[] = "ON";
char _close[] = "OFF";
NexNumber nFuelVal = NexNumber(0, 12, "nFuelVal");
NexPicture pGasHandState = NexPicture(0, 7, "pGasHandState");
NexPicture pWheelState = NexPicture(0, 8, "pWheelState");
NexPicture pEngState = NexPicture(0, 9, "pEngState");
NexPicture pSwitchState = NexPicture(0, 10, "pSwitchState");
NexText tDateNav = NexText(0, 21, "tDateNav");
NexText tBatteryVal = NexText(0, 23, "tBatteryVal");
NexText tHourNav = NexText(0, 25, "tHourNav");
NexText tMinuteNav = NexText(0, 26, "tMinuteNav");
```

```
NexNumber nLoadDate = NexNumber(0, 33, "nLoadDate");
NexPicture pMsg = NexPicture(0, 23, "pMsg");
NexButton bChangeDate = NexButton(1, 6, "bChangeDate");
NexText tDay = NexText(1, 23, "tDay");
NexText tMonth = NexText(1, 24, "tMonth");
NexText tYear = NexText(1, 25, "tYear");
NexText tHour = NexText(1, 26, "tHour");
NexText tMinute = NexText(1, 27, "tMinute");
NexNumber vaVisNumMenu = NexNumber(1, 31, "vaVisNumMenu");
NexText tEngDateLast = NexText(3, 9, "tDateL");
NexText tEngTimeLast = NexText(3, 10, "tTimeL");
NexRadio rEng10 = NexRadio(3, 13, "rEng10");
NexRadio rEng15 = NexRadio(3, 15, "rEng15");
NexText tEngDateFirst = NexText(3, 21, "tDateF");
NexText tEngTimeFirst = NexText(3, 22, "tTimeF");
NexRadio rEng1 = NexRadio(3, 24, "rEng1");
NexNumber nLoadEng = NexNumber(3, 25, "nLoadEng");
NexRadio rBat11 = NexRadio(4, 7, "rBat11");
NexRadio rBat12 = NexRadio(4, 9, "rBat12");
NexText tBatteryValInBatPage = NexText(4, 17, "tBatteryVal");
NexNumber nLoadBat = NexNumber(4, 18, "nLoadBat");
NexRadio rBat13 = NexRadio(4, 19, "rBat13");
NexText tNetName = NexText(5, 8, "tNetName");
NexText tNetPass = NexText(5, 9, "tNetPass");
NexButton bConnectWifi = NexButton(5, 10, "bConnectWifi");
NexPicture pWifiState = NexPicture(10, 10, "pWifiState");
NexText tWifiName = NexText(10, 11, "tWifiName");
NexButton bNewConnect = NexButton(10, 12, "bNewConnect");
NexDSButton btGasH = NexDSButton(11, 7, "btGasH");
NexDSButton btWhll = NexDSButton(11, 8, "btWhll");
NexDSButton btEng = NexDSButton(11, 9, "btEng");
NexDSButton btSwtch = NexDSButton(11, 10, "btSwtch");
```

```
NexNumber nLoadChng = NexNumber(11, 11, "nLoadChng");
NexText tMsg = NexText(12, 1, "tMsg");
NexTouch *nex_listen_list[] =
{
  &bChangeDate,
  &bConnectWifi,
  &bNewConnect,
  &rEng1,
  &rEng10,
  &rEng15,
  &rBat11,
  &rBat12,
  &rBat13,
  &btGasH,
  &btWhll,
  &btEng,
  &btSwtch,
  NULL
};
void bConnectWifiPopCallback(void *ptr)
{
  char bufferSSID[20] = {0};
  char bufferPasswrd[20] = {0};
  memset(bufferSSID, 0, sizeof(bufferSSID));
  tNetName.getText(bufferSSID, sizeof(bufferSSID));
  _ssid = bufferSSID;
  memset(bufferPasswrd, 0, sizeof(bufferPasswrd));
  tNetPass.getText(bufferPasswrd, sizeof(bufferPasswrd));
  _wifiPassword = bufferPasswrd;
  ConnectWifi(_ssid, _wifiPassword);
  Serial2.print("page pageWifiSett");
  SetHMISettingsForWrite();
```

```
LoadWifiValuesForPage();
nexLoop(nex_listen_list);
}
void bNewConnectPopCallback(void *ptr)
{
 tNetName.setText(_ssid);
 tNetPass.setText(_wifiPassword);
 nexLoop(nex_listen_list);
}
void bChangeDatePopCallback(void *ptr)
{
 DateTime date;
 char bufferClockSet[10] = {0};
 memset(bufferClockSet, 0, sizeof(bufferClockSet));
 tDay.getText(bufferClockSet, sizeof(bufferClockSet));
 date.setDay(atoi(bufferClockSet));
 memset(bufferClockSet, 0, sizeof(bufferClockSet));
 tMonth.getText(bufferClockSet, sizeof(bufferClockSet));
 date.setMonth(atoi(bufferClockSet));
 memset(bufferClockSet, 0, sizeof(bufferClockSet));
 tYear.getText(bufferClockSet, sizeof(bufferClockSet));
 date.setYear(atoi(bufferClockSet));
 memset(bufferClockSet, 0, sizeof(bufferClockSet));
 tHour.getText(bufferClockSet, sizeof(bufferClockSet));
 date.setHour(atoi(bufferClockSet));
 memset(bufferClockSet, 0, sizeof(bufferClockSet));
 tMinute.getText(bufferClockSet, sizeof(bufferClockSet));
 date.setMinute(atoi(bufferClockSet));
 uint8_t dayOfWeek = GetDayOfWeek(date.getDay(), date.getMonth(),
date.getYear());
 boatRTC.setDS1302Time(00, date.getMinute(), date.getHour(), dayOfWeek,
date.getDay(), date.getMonth(), date.getYear());
```

```
  SendDateTimeNowToUno();
  Serial2.print("page pageHome");
  SetHMISettingsForWrite();
  nexLoop(nex_listen_list);
}
void SendDataWithICToSlave(String str, uint8_t slaveId)
{
  Wire.beginTransmission(slaveId);
  Wire.write(str.c_str());
  Wire.endTransmission();
}
void SendDataToESP(String str)
{
  Serial3.println(str);
}
void SplitStringAndSetDate(String str, char* chr, uint8_t dateType)
{
  int i = 0;
  char *array[6];
  int strLen = str.length() + 1;
  char char_array[strLen];
  str.toCharArray(char_array, strLen);
  char *token = strtok(char_array, chr);
  while (token != NULL)
  {
    array[i++] = token;
    token = strtok(NULL, chr);
  }
  if (dateType == ENGSTOPPEDDATE)
  {
    _engStppdDt = new DateTime();
    _engStppdDt->setHour(atoi(array[0]));
```

```
  _engStppdDt->setMinute(atoi(array[1]));
   _engStppdDt->setSecond(atoi(array[2]));
   _engStppdDt->setDay(atoi(array[3]));
   _engStppdDt->setMonth(atoi(array[4]));
   _engStppdDt->setYear(atoi(array[5]));
   UpdateStoppdDateIntoEEPROM(_engStppdDt);
  }
  else if (dateType == ENGSTRTDDATE)
  {
   _engStrtdDt = new DateTime();
   _engStrtdDt->setHour(atoi(array[0]));
   _engStrtdDt->setMinute(atoi(array[1]));
   _engStrtdDt->setSecond(atoi(array[2]));
   _engStrtdDt->setDay(atoi(array[3]));
   _engStrtdDt->setMonth(atoi(array[4]));
   _engStrtdDt->setYear(atoi(array[5]));
  }
}
void SetComingDate(String value, uint8_t dateType)
{
  SplitStringAndSetDate(value, "/", dateType);
}
void SendDataToSlaveUno(uint8_t type, String value)
{
  String str;
  str += type;
  str += ":";
  str += value;
  str += "@";
  SendDataWithICToSlave(str, DEVICE_ID_UNO);
}
```

```
void SendDateTimeNowToUno()
{
 String str = "";
 String date = "";
 date = boatRTC.hours;
 date += "/";
 date += boatRTC.minutes;
 date += "/";
 date += boatRTC.seconds;
 date += "/";
 date += boatRTC.dayofmonth;
 date += "/";
 date += boatRTC.month;
 date += "/";
 date += boatRTC.year;
 SendDataToSlaveUno(DATETIMENOW, date);
}
void SetMessageForNextion(uint8_t msgNo)
{
 _message = MessageType_String[msgNo];
 Serial2.print("page pageMessage");
 SetHMISettingsForWrite();
 tMsg.setText(_message.c_str());
}
void TakeDataComingFromSlave()
{
 String coming;
 char c;
 Wire.requestFrom(DEVICE_ID_UNO, 24);
 while (Wire.available()) {
   c = Wire.read();
   coming += c;}
```

```
 if (coming.indexOf(':') > -1)
 {
  int firstSeparateIndex = coming.indexOf(':');
  int secndSeparateIndex = coming.indexOf('@');
  uint8_t comingDataType = coming.substring(0, firstSeparateIndex).toInt();
  String comingVal = coming.substring(firstSeparateIndex + 1,
secndSeparateIndex);
  switch (comingDataType)
  {
    case  NOTHING:
      break;
    case ENGSTOPPEDDATE:
      SetComingDate(comingVal, ENGSTOPPEDDATE);
      break;
    case ENGSTRTDDATE:
      SetComingDate(comingVal, ENGSTRTDDATE);
      break;
    case MESSAGE:
      SetMessageForNextion(comingVal.toInt());
      SendDataToESP(coming);
      break;
  };
 }
}
void SendDataToSlaveEsp(uint8_t type, String value)
{
 String str;
 str += type;
 str += ":";
 str += value;
 str += "@";
 SendDataToESP(str);}
```

```
void bSetEngWorkingTime1PushCallback(void *ptr)
{
 _engWorkTime = 1;
 SendDataToSlaveUno(ENGWORKINGTIME, (String)_engWorkTime);
 SendDataToSlaveEsp(ENGWORKINGTIME, (String)_engWorkTime);
 nexLoop(nex_listen_list);
}
void bSetEngWorkingTime10PushCallback(void *ptr)
{
 _engWorkTime = 10;
 SendDataToSlaveUno(ENGWORKINGTIME, (String)_engWorkTime);
 SendDataToSlaveEsp(ENGWORKINGTIME, (String)_engWorkTime);
 nexLoop(nex_listen_list);
}
void bSetEngWorkingTime15PushCallback(void *ptr)
{
 _engWorkTime = 15;
 SendDataToSlaveUno(ENGWORKINGTIME, (String)_engWorkTime);
 SendDataToSlaveEsp(ENGWORKINGTIME, (String)_engWorkTime);
 nexLoop(nex_listen_list);
}
void bSetEngWorkingTime20PushCallback(void *ptr)
{
 _engWorkTime = 20;
 SendDataToSlaveUno(ENGWORKINGTIME, (String)_engWorkTime);
 SendDataToSlaveEsp(ENGWORKINGTIME, (String)_engWorkTime);
 nexLoop(nex_listen_list);
}
void bSetBatTrigVolt11PushCallback(void *ptr)
{
 _batVoltToTrigSys = 11;
 SendDataToSlaveUno(BATTRIGVOLT, (String)_batVoltToTrigSys);
```

```
 SendDataToSlaveEsp(BATTRIGVOLT, (String)_batVoltToTrigSys);
 nexLoop(nex_listen_list);
}
void bSetBatTrigVolt12PushCallback(void *ptr)
{
 _batVoltToTrigSys = 12;
 SendDataToSlaveUno(BATTRIGVOLT, (String)_batVoltToTrigSys);
 SendDataToSlaveEsp(BATTRIGVOLT, (String)_batVoltToTrigSys);
 nexLoop(nex_listen_list);
}
void bSetBatTrigVolt13PushCallback(void *ptr)
{
 _batVoltToTrigSys = 13;
 SendDataToSlaveUno(BATTRIGVOLT, (String)_batVoltToTrigSys);
 SendDataToSlaveEsp(BATTRIGVOLT, (String)_batVoltToTrigSys);
 nexLoop(nex_listen_list);
}
void bChangeGasHndleSt(void *ptr)
{
 uint32_t dual_state;
 btGasH.getValue(&dual_state);
 SendDataToSlaveUno(GASHANDLE_ST, (String)dual_state);
 nexLoop(nex_listen_list);
}
void bChangeWheelSt(void *ptr)
{
 uint32_t dual_state;
 btWhll.getValue(&dual_state);
 SendDataToSlaveUno(WHEEL_ST, (String)dual_state);
 nexLoop(nex_listen_list);
}
```

```
void bChangeEngSt(void *ptr)
{
 uint32_t dual_state;
 btEng.getValue(&dual_state);
 SendDataToSlaveUno(ENGINE_ST, (String)dual_state);
 nexLoop(nex_listen_list);
}
void bChangeSwitchSt(void *ptr)
{
 uint32_t dual_state;
 btSwtch.getValue(&dual_state);
 SendDataToSlaveUno(SWITCH_ST, (String)dual_state);
 nexLoop(nex_listen_list);
}
void ShowFuelValue()
{
 uint16_t fuelValue = ReadCurrentFuelValue();
 nFuelVal.setValue(fuelValue);
 SendDataToSlaveEsp(CURRFUELVAL, (String)fuelValue);
}
void ShowBatteryValue()
{
 float batteryValue = ReadBatteryCurrentValue();
 _curBatVolt = batteryValue;
 dtostrf(batteryValue, 5, 2, _batteryValue);
 tBatteryVal.setText(_batteryValue);
 tBatteryValInBatPage.setText(_batteryValue);
 SendDataToSlaveEsp(BATTRYVAL, (String)_curBatVolt);
}
void ShowBatTrigVoltChoose()
{
 switch (_batVoltToTrigSys)
```

```
  {
    case batOpt11:
      rBat11.setValue(1);
      rBat12.setValue(0);
      rBat13.setValue(0);
      break;
    case batOpt12:
      rBat11.setValue(0);
      rBat12.setValue(1);
      rBat13.setValue(0);
      break;
    case batOpt13:
      rBat11.setValue(0);
      rBat12.setValue(0);
      rBat13.setValue(1);
      break;
  }
}
void ShowEngWorkingTimeChoose()
{
  switch (_engWorkTime)
  {
    case engOpt1:
      rEng1.setValue(1);
      rEng10.setValue(0);
      rEng15.setValue(0);
      break;
    case engOpt10:
      rEng1.setValue(0);
      rEng10.setValue(1);
      rEng15.setValue(0);
      break;
    case engOpt15:
```

```
    rEng1.setValue(0);
    rEng10.setValue(0);
    rEng15.setValue(1);
    break;
  }
}
void ShowChangeableValues()
{
  boolean switchSt = digitalRead(SWITCH_PIN);
  boolean wheelSt = digitalRead(WHEEL_PIN);
  boolean gasHndSt = digitalRead(GAS_HANDLE_PIN);
  boolean engSt = digitalRead(ENGINE_PIN);
  btSwtch.setValue(switchSt);
  if (switchSt)
  {
   btSwtch.setText(_open);
  }
  else
  {
   btSwtch.setText(_close);
  }
  btWhll.setValue(wheelSt);
  if (wheelSt)
  {
   btWhll.setText(_open);
  }
  else
  {
   btWhll.setText(_close);
  }
  btGasH.setValue(gasHndSt);
  if (gasHndSt)
  {
```

```
  btGasH.setText(_open);
  }
  else
  {
   btGasH.setText(_close);
  }
  btEng.setValue(engSt);
  if (engSt)
  {
   btEng.setText(_open);
  }
  else
  {
   btEng.setText(_close);
  }
}
void LoadWifiValuesForPage()
{
  bool state = digitalRead(CONN_STATE);
  if (_connState = !state)
  {
    _connState = state;
    if (state)
    {
      pWifiState.Set_background_image_pic(26);
      tWifiName.setText(_ssid);
    }
    else
    {
      pWifiState.Set_background_image_pic(27);
      tWifiName.setText("bağlantı yok");
    }
```

```
 }
}
uint8_t GetDayOfWeek( uint8_t day, uint8_t month, uint16_t year)
{
  static int t[] = {0, 3, 2, 5, 0, 3, 5, 1, 4, 6, 2, 4};
  year -= month < 3;
  return (year + year / 4 - year / 100 + year / 400 + t[month - 1] + day) % 7;
}
void LoadDateFromRtcToClockNex()
{
  DateTime date;
  char clockNex[10] = {0};
  date.setDay(boatRTC.dayofmonth);
  date.setMonth(boatRTC.month);
  date.setYear(boatRTC.year);
  date.setHour(boatRTC.hours);
  date.setMinute(boatRTC.minutes);
  sprintf(clockNex, "%02d", date.getDay());
  tDay.setText(clockNex);
  sprintf(clockNex, "%02d", date.getMonth());
  tMonth.setText(clockNex);
  sprintf(clockNex, "%0002d", date.getYear());
  tYear.setText(clockNex);
  sprintf(clockNex, "%02d", date.getHour());
  tHour.setText(clockNex);
  sprintf(clockNex, "%02d", date.getMinute());
  tMinute.setText(clockNex);
}
void SetDateTimeToRTC( uint8_t _second, uint8_t _minute , uint8_t _hour ,
uint8_t _dayOfWeek, uint8_t _day, uint8_t _month, uint16_t _year)
{
  boatRTC.setDS1302Time(_second, _minute, _hour, _dayOfWeek, _day,
```

```
_month, _year);
}
void SetHMISettingsForWrite()
{
  Serial2.write(0xff);
  Serial2.write(0xff);
  Serial2.write(0xff);
}
void SetNavDateTimeNex()
{
  char dateNav[11] = {0};
  DateTime dateForNex;
  dateForNex.setHour(boatRTC.hours);
  dateForNex.setMinute(boatRTC.minutes);
  dateForNex.setDay(boatRTC.dayofmonth);
  dateForNex.setMonth(boatRTC.month);
  dateForNex.setYear(boatRTC.year);
  sprintf(dateNav, "%02d", dateForNex.getHour());
  tHourNav.setText(dateNav);
  sprintf(dateNav, "%02d", dateForNex.getMinute());
  tMinuteNav.setText(dateNav);
  sprintf(dateNav,  "%02d/%02d/%0002d  ",  dateForNex.getDay(),
dateForNex.getMonth(), dateForNex.getYear());
  tDateNav.setText(dateNav);
}
uint16_t ReadCurrentFuelValue()
{
  uint16_t valueFromFuelTank;
  uint16_t calculatedRateByFullFuel;
  valueFromFuelTank = analogRead(FUEL_PIN);
  calculatedRateByFullFuel = (valueFromFuelTank * 100) / _fllFuelVal;
```

```
 return calculatedRateByFullFuel;
}
float CalculateBatteryTotalValue(uint16_t value)
{
  float vin = 0.0;
  float vout = 0.0;
  float R1 = 10000.0;
  float R2 = 10000.0;
  float R3 = 10000.0;
  float R4 = 10000.0;
  vout = value * (5.0 / 1023.0);
  vin = (vout * (R1 + R2 + R3 + R4) ) / R4;
  if (vin < 0.09) {
    vin = 0.0;
  }
  return vin;
}
float ReadBatteryCurrentValue()
{
  uint16_t batteryValue = analogRead(BATTERY_PIN);
  float vin = CalculateBatteryTotalValue(batteryValue);
  return vin;
}
void ConnectWifi(String ssid, String wifiPassword)
{
  String coming;
  String strSending;
  String str;
  String value;
  char c;
  ssid.replace(" ", "");
  wifiPassword.replace(" ", "");
```

```
 if (ssid != "" && wifiPassword != "")
 {
  strSending += CONNECTWIFI;
  strSending += ":";
  strSending += ssid;
  strSending += ",";
  strSending += wifiPassword;
  strSending += "@";
  SendDataToESP(strSending);
  delay(5000);
  _connState = digitalRead(CONN_STATE);
 }
}
void SetEngineWorkingTime(uint8_t minuteVal)
{
 _engWorkTime = minuteVal;
 SendDataToSlaveUno(ENGWORKINGTIME, (String)minuteVal);
 SendDataToSlaveEsp(ENGWORKINGTIME, (String)minuteVal);
}
void ShowSwitchState()
{
 boolean state = digitalRead(SWITCH_PIN);
 if (state)
 {
  pSwitchState.Set_background_image_pic(13);
 }
 else
 {
  pSwitchState.Set_background_image_pic(12);
 }
}
```

```
void ShowGasHandleState()
{
 boolean state = digitalRead(GAS_HANDLE_PIN);
 if (state)
 {
  pGasHandState.Set_background_image_pic(13);
 }
 else
 {
  pGasHandState.Set_background_image_pic(12);
 }
}
void ShowWheelState()
{
 boolean state = digitalRead(WHEEL_PIN);
 if (state)
 {
  pWheelState.Set_background_image_pic(13);
 }
 else
 {
  pWheelState.Set_background_image_pic(12);
 }
}
void ShowEngineState()
{
 boolean state = digitalRead(ENGINE_PIN);
 if (state)
 {
  pEngState.Set_background_image_pic(13);
 }
```

```
else
  {
    pEngState.Set_background_image_pic(12);
  }
}
void ShowEngStoppdDate()
{
  if (_engStppdDt->getYear() > 1999)
  {
    char dateStr[11] = {0};
    char timeStr[6] = {0};
    sprintf(dateStr,        "%02d.%02d.%0002d        ",        _engStppdDt->getDay(),
_engStppdDt->getMonth(), _engStppdDt->getYear());
    tEngDateLast.setText(dateStr);
    sprintf(timeStr,    "%02d.%02d    ",    _engStppdDt->getHour(),    _engStppdDt-
>getMinute());
    tEngTimeLast.setText(timeStr);
  }
}
void UpdateStoppdDateIntoEEPROM(DateTime* engStoppdDate)
{
  if (engStoppdDate != NULL)
  {
    EEPROM.update(_hourAddr, engStoppdDate->getHour());
    EEPROM.update(_minuteAddr, engStoppdDate->getMinute());
    EEPROM.update(_secondAddr, engStoppdDate->getSecond());
    EEPROM.update(_dayOfMonthAddr, engStoppdDate->getDay());
    EEPROM.update(_monthAddr, engStoppdDate->getMonth());
    byte byte1 = engStoppdDate->getYear() >> 8;
    byte byte2 = engStoppdDate->getYear() & 0xFF;
    EEPROM.update(_yearHighAddr, byte1);
    EEPROM.update(_yearLowAddr, byte2);
```

```
 }}

void ReadEngStoppdDateFromEEPROM()
{
 _engStppdDt = new DateTime();
 _engStppdDt->setHour(EEPROM.read(_hourAddr));
 _engStppdDt->setMinute(EEPROM.read(_minuteAddr));
 _engStppdDt->setSecond(EEPROM.read(_secondAddr));
 _engStppdDt->setDay(EEPROM.read(_dayOfMonthAddr));
 _engStppdDt->setMonth(EEPROM.read(_monthAddr));
 byte byte1 = EEPROM.read(_yearHighAddr);
 byte byte2 = EEPROM.read(_yearLowAddr);
 _engStppdDt->setYear((byte1 << 8) + byte2);
}
void setup() {
 Wire.begin();
 Serial.begin(9600);
 Serial3.begin(115200);
 pinMode(FUEL_PIN, INPUT);
 pinMode(BATTERY_PIN, INPUT);
 pinMode(CONN_STATE, INPUT);
 pinMode(WHEEL_PIN, INPUT);
 pinMode(ENGINE_PIN, INPUT);
 pinMode(GAS_HANDLE_PIN, INPUT);
 pinMode(SWITCH_PIN, INPUT);
 nexInit();
 ShowGasHandleState();
 ShowSwitchState();
 ShowWheelState();
 ShowEngineState();
 bChangeDate.attachPop(bChangeDatePopCallback, &bChangeDate);
 bConnectWifi.attachPop(bConnectWifiPopCallback, &bConnectWifi);
 bNewConnect.attachPop(bNewConnectPopCallback, &bNewConnect);
```

```
  rEng1.attachPush(bSetEngWorkingTime1PushCallback, &rEng1);
  rEng10.attachPush(bSetEngWorkingTime10PushCallback, &rEng10);
  rEng15.attachPush(bSetEngWorkingTime15PushCallback, &rEng15);
  rBat11.attachPush(bSetBatTrigVolt11PushCallback, &rBat11);
  rBat12.attachPush(bSetBatTrigVolt12PushCallback, &rBat12);
  rBat13.attachPush(bSetBatTrigVolt13PushCallback, &rBat13);
  btGasH.attachPush(bChangeGasHndleSt, &btGasH);
  btWhll.attachPush(bChangeWheelSt, &btWhll);
  btEng.attachPush(bChangeEngSt, &btEng);
  btSwtch.attachPush(bChangeSwitchSt, &btSwtch);
  delay(1000);
}
void loop() {
  nexLoop(nex_listen_list);
  TakeDataComingFromSlave();
  boatRTC.updateTime();
  SendDateTimeNowToUno();
  ShowFuelValue();
  ShowBatteryValue();
  ShowGasHandleState();
  ShowSwitchState();
  ShowWheelState();
  ShowEngineState();
  LoadWifiValuesForPage();
  nLoadDate.getValue(&_pageDateSett);
  SetNavDateTimeNex();
  if (_pageDateSett == 1)
  {
    LoadDateFromRtcToClockNex();
    nLoadDate.setValue(0);
  }
  nLoadEng.getValue(&_pageEngSett);
```

```
  if (_pageEngSett == 1)
  {
    ShowEngWorkingTimeChoose();
    ReadEngStoppdDateFromEEPROM();
    ShowEngStoppdDate();
    nLoadEng.setValue(0);
  }
  nLoadBat.getValue(&_pageBatSett);
  if (_pageBatSett == 1)
  {
    ShowBatTrigVoltChoose();
    nLoadBat.setValue(0);
  }
  nLoadChng.getValue(&_pageChnge);
  if (_pageChnge == 1)
  {
    ShowChangeableValues();
  }
  delay(2000);
}
```

Ek-B

Arduino Uno içerisine aktarılan kodlar verilmiştir (tekneSarjKontrol_Battery.ino).

```
#include <Wire.h>
#include "TimeLib.h"
#include <Servo.h>
#include <EEPROM.h>
#define SWITCH_PIN 2
#define GAS_HANDLE_PIN 3
#define WHEEL_PIN 4
#define WHEEL_SRV_PIN 5
#define ENGINE_PIN 7
#define ENG_WORK_PIN 8
#define FUEL_PIN A1
#define BATTERY_PIN A2
#define DEVICE_ID_UNO 1
enum DataType
{
  NOTHING,
  ENGWORKINGTIME,
  BATTRIGVOLT,
  CONNECTWIFI,
  ENGSTOPPEDDATE,
  MESSAGE,
  DATETIMENOW,
  CURRFUELVAL,
  ENGSTRTDDATE,
  WHEEL_ST,
  ENGINE_ST,
  SWITCH_ST,
  GASHANDLE_ST
};
enum MessageType
```

```
{
  FUEL_CRITICAL,
  TAKE_NOTR_GASHANDLE,
  SWITCH_OPEN_GASHANDLE_NOTR,
  CONTROL_AKU_STATE,
  FUEL_LOW
} messageType;
class DateTime {
    uint8_t second;
    uint8_t minute;
    uint8_t hour;
    uint8_t day;
    uint8_t month;
    uint16_t year;
  public:
    void setDateTime( uint8_t _second, uint8_t _minute , uint8_t _hour , uint8_t
_day, uint8_t _month, uint16_t _year)
    {
      second = _second;
      minute = _minute;
      hour = _hour;
      day = _day;
      month = _month;
      year = _year;
    }
  public:
    void setSecond(uint8_t _second) {
      second = _second;
    }
  public:
    uint8_t getSecond() {
      return second;
    }
```

```
public:
  void setMinute(uint8_t _minute) {
    minute = _minute;
  }
public:
  uint8_t getMinute() {
    return minute;
  }
public:
  void setHour(uint8_t _hour) {
    hour = _hour;
  }
public:
  uint8_t getHour() {
    return hour;
  }
public:
  void setDay(uint8_t _day) {
    day = _day;
  }
public:
  uint8_t getDay() {
    return day;
  }
public:
  void setMonth(uint8_t _month) {
    month = _month;
  }
public:
  uint8_t getMonth() {
    return month;
  }
public:
```

```
    void setYear(uint16_t _year) {
      year = _year;
    }
  public:
    uint16_t getYear() {
      return year;
    }
};
DateTime* _engStrtdDt = NULL;
DateTime* _engStppdDt = NULL;
DateTime* _dateTimeNow = NULL;
Servo _wheelServo;
uint8_t _wheelLck = 0;
uint8_t _wheelUnLck = 90;
const uint16_t _fullFuelValue = 630;
uint8_t  _engineWorkingTime = 1;
uint8_t _batteryVoltToTriggerSystem = 11;
float _currentBatteryVolt = 0;
uint16_t _currentFuelValue = 0;
uint16_t _criticalFuelValue = _fullFuelValue / 8;
uint16_t _LowFuelValue = _fullFuelValue / 4;
bool volatile _switchState = false;
bool volatile _gasHandleState = false;
String _sendingMessage = "";
uint8_t comingDataType = 0;
uint8_t sendngDataType = 0;
uint8_t _hourAddr = 0;
uint8_t _minuteAddr = 1;
uint8_t _secondAddr = 2;
uint8_t _dayOfMonthAddr = 3;
uint8_t _monthAddr = 4;
uint8_t _yearLowAddr = 5;
uint8_t _yearHighAddr = 6;
```

```
uint16_t ReadCurrentFuelValue()
{
  uint16_t valueFromFuelTank = analogRead(FUEL_PIN);
  return valueFromFuelTank;
}
float CalculateBatteryTotalValue(uint16_t value)
{
  float vin = 0.0;
  float vout = 0.0;
  float R1 = 10000.0;
  float R2 = 10000.0;
  float R3 = 10000.0;
  float R4 = 10000.0;
  vout = value * (5.0 / 1023.0);
  vin = (vout * (R1 + R2 + R3 + R4) ) / R4;
  if (vin < 0.09) {
    vin = 0.0;
  }
  return vin;
}
float ReadBatteryCurrentValue()
{
  uint16_t batteryValue = analogRead(BATTERY_PIN);
  float vin = CalculateBatteryTotalValue(batteryValue);
  return vin;
}
void ReadSwitchState()
{
  _switchState = digitalRead(SWITCH_PIN);
}
void CloseSwitch()
{
  digitalWrite(SWITCH_PIN, LOW);
```

```
}
void OpenSwitch()
{
  digitalWrite(SWITCH_PIN, HIGH);
}
void ReadGasHandleState()
{
  _gasHandleState = digitalRead(GAS_HANDLE_PIN);
}
void SendMessageToMaster(uint8_t  msgType)
{
  messageType = msgType;
  sendngDataType = MESSAGE;
}
void SendEngStoppedDateToMaster()
{
  sendngDataType = ENGSTOPPEDDATE;
}
void SetEngineWorkingTime(uint8_t minute)
_engineWorkingTime = minute;
}
void SetBatteryVoltToTriggerSystem(uint8_t batteryVolt)
{
  _batteryVoltToTriggerSystem = batteryVolt;
}
void LockWheel()
{
  _wheelServo.write(_wheelLck);
  digitalWrite(WHEEL_PIN, LOW);
}
void UnLockWheel()
{
  _wheelServo.write(_wheelUnLck);
```

```
  digitalWrite(WHEEL_PIN, HIGH);
}
void StartEngine()
{
  digitalWrite(ENGINE_PIN, HIGH);
  digitalWrite(ENG_WORK_PIN, HIGH);
}
void StopEngine()
{
  digitalWrite(ENGINE_PIN, LOW);
  digitalWrite(ENG_WORK_PIN, LOW);
}
void SetWheelState(uint8_t value) {
  if (value)
  {
    UnLockWheel();
  }
  else {
    LockWheel();
  }
}
void SetEngineState(uint8_t value) {
  if (value)
  {
    StartEngine();
  }
  else
  {
    StopEngine();
  }
}
void SetSwitchState(uint8_t value) {
  digitalWrite(SWITCH_PIN, value);
```

```
}
void SetGasHandleState(uint8_t value) {
  digitalWrite(GAS_HANDLE_PIN, value);
}
uint32_t DifferentDatetimeAsSecond(DateTime* _dateFirst, DateTime* _dateLast)
{
  tmElements_t firsTm;
  tmElements_t lastTm;
  firsTm.Hour = _dateFirst->getHour();
  firsTm.Minute = _dateFirst->getMinute();
  firsTm.Second = _dateFirst->getSecond();
  firsTm.Day = _dateFirst->getDay();
  firsTm.Month = _dateFirst->getMonth();
  firsTm.Year = _dateFirst->getYear() - 1970;
  lastTm.Hour = _dateLast->getHour();
  lastTm.Minute = _dateLast->getMinute();
  lastTm.Second = _dateLast->getSecond();
  lastTm.Day = _dateLast->getDay();
  lastTm.Month = _dateLast->getMonth();
  lastTm.Year = _dateLast->getYear() - 1970;
  time_t first_tm = makeTime(firsTm);
  time_t last_tm = makeTime(lastTm);
  uint32_t result = last_tm - first_tm;
  return result;
}
uint32_t ConvertMinuteToMillis(uint32_t value)
{
  return value * 60 * 1000;
}
void SplitStringAndSetDate(String str, char* chr)
{
  int i = 0;
  char *array[6];
```

```
  int strLen = str.length() + 1;
  char char_array[strLen];
  str.toCharArray(char_array, strLen);
  char *token = strtok(char_array, chr);
  while (token != NULL)
  {
    array[i++] = token;
    token = strtok(NULL, chr);
  }
  _dateTimeNow = new DateTime();
  _dateTimeNow->setHour(atoi(array[0]));
  _dateTimeNow->setMinute(atoi(array[1]));
  _dateTimeNow->setSecond(atoi(array[2]));
  _dateTimeNow->setDay(atoi(array[3]));
  _dateTimeNow->setMonth(atoi(array[4]));
  _dateTimeNow->setYear(atoi(array[5]));
}
void SetDateTimeNow(String value)
{
  SplitStringAndSetDate(value, "/");
}
void UpdateStoppdDateIntoEEPROM(DateTime* engStoppdDate)
{
  if (engStoppdDate != NULL)
  {
    EEPROM.update(_hourAddr, engStoppdDate->getHour());
    EEPROM.update(_minuteAddr, engStoppdDate->getMinute());
    EEPROM.update(_secondAddr, engStoppdDate->getSecond());
    EEPROM.update(_dayOfMonthAddr, engStoppdDate->getDay());
    EEPROM.update(_monthAddr, engStoppdDate->getMonth());
    byte byte1 = engStoppdDate->getYear() >> 8;
    byte byte2 = engStoppdDate->getYear() & 0xFF;
    EEPROM.update(_yearHighAddr, byte1);
```

```
    EEPROM.update(_yearLowAddr, byte2);
  }
}
void SetDefaultStoppdDateIntoEEPROM(DateTime* engStoppdDate)
{
  if (engStoppdDate != NULL)
  {
    EEPROM.update(_hourAddr, 0);
    EEPROM.update(_minuteAddr, 0);
    EEPROM.update(_secondAddr, 0);
    EEPROM.update(_dayOfMonthAddr, 0);
    EEPROM.update(_monthAddr, 0);
    EEPROM.update(_yearHighAddr, 0);
    EEPROM.update(_yearLowAddr, 0);
  }
}
void ReadEngStoppdDateFromEEPROM()
{
  _engStppdDt = new DateTime();
  _engStppdDt->setHour(EEPROM.read(_hourAddr));
  _engStppdDt->setMinute(EEPROM.read(_minuteAddr));
  _engStppdDt->setSecond(EEPROM.read(_secondAddr));
  _engStppdDt->setDay(EEPROM.read(_dayOfMonthAddr));
  _engStppdDt->setMonth(EEPROM.read(_monthAddr));
  byte byte1 = EEPROM.read(_yearHighAddr);
  byte byte2 = EEPROM.read(_yearLowAddr);
  _engStppdDt->setYear((byte1 << 8) + byte2);
}
void SetStartedValues ()
{
  SetWheelState(0);
  SetEngineState(0);
  SetSwitchState(0);
```

```
  SetGasHandleState(0);
}
void setup() {
  Serial.begin(9600);
  Wire.begin(DEVICE_ID_UNO);
  Wire.onRequest(requestEvent);
  Wire.onReceive(receiveEvent);
  pinMode(FUEL_PIN, INPUT);
  pinMode(BATTERY_PIN, INPUT);
  pinMode(GAS_HANDLE_PIN, OUTPUT);
  pinMode(SWITCH_PIN, OUTPUT);
  pinMode(ENGINE_PIN, OUTPUT);
  pinMode(WHEEL_PIN, OUTPUT);
  pinMode(ENG_WORK_PIN, OUTPUT);
  _wheelServo.attach(WHEEL_SRV_PIN);
  SetStartedValues();
  ReadSwitchState();
  ReadGasHandleState();
  attachInterrupt(digitalPinToInterrupt(SWITCH_PIN),ControlSwitchState,
CHANGE);
  attachInterrupt(digitalPinToInterrupt(GAS_HANDLE_PIN),
ControlGasHandleState, CHANGE);
}
void loop() {
programStart:
readFuelStart:
  _currentFuelValue = ReadCurrentFuelValue();
  if (_currentFuelValue <= _criticalFuelValue)
  {
    SendMessageToMaster(FUEL_CRITICAL);
    delay(600000);
    goto programStart;
  }
```

```
  if (_switchState)
  {
    goto programStart;
  }
readBatteryVoltStart:
  _currentBatteryVolt = ReadBatteryCurrentValue();
  if (_currentBatteryVolt > _batteryVoltToTriggerSystem)
  {
    delay(600000);
    goto programStart;
  }
  if (_switchState)
  {
    goto programStart;
  }
readGasHandleStateStart:
  if (_gasHandleState)
  {
    SendMessageToMaster(TAKE_NOTR_GASHANDLE);
    while (_gasHandleState)
      delay(10000);
  }
  if (_switchState)
  {
    goto programStart;
  }
  OpenSwitch();
  if (_gasHandleState)
  {
    SendMessageToMaster(SWITCH_OPEN_GASHANDLE_NOTR);
    goto StopEngine;
  }
  LockWheel();
```

```
  delay(10000);
  StartEngine();
  _engStrtdDt = new DateTime();
  _engStrtdDt = _dateTimeNow;
  unsigned long engStartedMillis = millis();
  if (_gasHandleState)
  {
    SendMessageToMaster(SWITCH_OPEN_GASHANDLE_NOTR);
    goto StopEngine;
  }
  ReadEngStoppdDateFromEEPROM();
  if (_engStppdDt->getYear() == 0)
  {
    goto EngineStoppedDateIsNull;
  }
  int32_t dateDiffAsSecond = DifferentDatetimeAsSecond(_engStrtdDt,
_engStppdDt);
  if (dateDiffAsSecond < 18000)
  {
    SendMessageToMaster(CONTROL_AKU_STATE);
  }
EngineStoppedDateIsNull:
  uint32_t engineWorkingTimeAsMillis =
ConvertMinuteToMillis(_engineWorkingTime);
  while (millis() - engStartedMillis < engineWorkingTimeAsMillis)
  {
    if (_gasHandleState)
    {
      delay(5000);
      SendMessageToMaster(SWITCH_OPEN_GASHANDLE_NOTR);
      goto StopEngine;
    }
  }
```

```
StopEngine:
  StopEngine();
  _engStppdDt == new DateTime();
  _engStppdDt = _dateTimeNow;
  CloseSwitch();
  UnLockWheel();
  delay(5000);
  UpdateStoppdDateIntoEEPROM(_engStppdDt);
  SendEngStoppedDateToMaster();
  _currentFuelValue = ReadCurrentFuelValue();
  if (_currentFuelValue <= _LowFuelValue)
  {
    delay(5000);
    SendMessageToMaster(FUEL_LOW);
  }
  delay(60000);
}
void ControlSwitchState()
{
  ReadSwitchState();
}
void ControlGasHandleState()
{
  ReadGasHandleState();
  if (_switchState)
  {
    if (_gasHandleState)
    {
      CloseSwitch();
    }
  }
}
void receiveEvent(int howMany)
```

```
{
 String coming;
 String str;
 String value;
 char c;
 while (Wire.available()) {
  c = Wire.read();
  coming += c;
 }
 coming.replace(" ", "");
 if (coming != "" && coming.indexOf(":") > -1)
 {
  int separateIndex = coming.indexOf(':');
  int secndSeparateIndex = coming.indexOf('@');
  str = coming.substring(0, separateIndex);
  value = coming.substring(separateIndex + 1, secndSeparateIndex);
  if (str != ""  && value != "")
  {
   comingDataType = str.toInt();
   switch (comingDataType)
   {
    case ENGWORKINGTIME:
     SetEngineWorkingTime(value.toInt());
     break;
    case BATTRIGVOLT:
     SetBatteryVoltToTriggerSystem(value.toInt());
     break;
    case DATETIMENOW:
     SetDateTimeNow(value);
     break;
    case WHEEL_ST:
     SetWheelState(value.toInt());
     break;
```

```
        case ENGINE_ST:
          SetEngineState(value.toInt());
          break;
        case SWITCH_ST:
          SetSwitchState(value.toInt());
          break;
        case GASHANDLE_ST:
          SetGasHandleState(value.toInt());
          break;
      }}}}
void requestEvent()
{
 String sending = "";
 switch (sendngDataType)
 {
  case  NOTHING:
    break;
  case ENGSTRTDDATE:
    sending += ENGSTRTDDATE;
    sending += ":";
    sending += _engStrtdDt->getHour();
    sending += "/";
    sending += _engStrtdDt->getMinute();
    sending += "/";
    sending += _engStrtdDt->getSecond();
    sending += "/";
    sending += _engStrtdDt->getDay();
    sending += "/";
    sending += _engStrtdDt->getMonth();
    sending += "/";
    sending += _engStrtdDt->getYear();
    sending += "@";
    break;
```

```
    case ENGSTOPPEDDATE:
      sending += ENGSTOPPEDDATE;
      sending += ":";
      sending += _engStppdDt->getHour();
      sending += "/";
      sending += _engStppdDt->getMinute();
      sending += "/";
      sending += _engStppdDt->getSecond();
      sending += "/";
      sending += _engStppdDt->getDay();
      sending += "/";
      sending += _engStppdDt->getMonth();
      sending += "/";
      sending += _engStppdDt->getYear();
      sending += "@";
      break;
    case MESSAGE:
      sending += MESSAGE;
      sending += ":";
      sending += messageType;
      sending += "@";
      break;
  };
  if (sendngDataType != NOTHING)
  {
    Wire.write(sending.c_str());
  }
  sendngDataType = NOTHING;
}
```

Ek-C

NodeMCU içerisine aktarılan kodlar verilmiştir (tekneSarjKontrol_esp.ino).

```
#include <ESP8266WiFi.h>
#include <FirebaseArduino.h>
#define FIREBASE_HOST "…….firebaseio.com"
#define FIREBASE_AUTH "***"
#define WIFI_SSID "TEST"
#define WIFI_PASSWORD "TEST"
#define CONN_STATE D0
#define ENGINE_PIN D1
#define WHEEL_PIN D2
#define SWITCH_PIN D3
#define GAS_HANDLE_PIN D4
bool _connectionState = false;
uint8_t _engineWorkingTime = 1;
uint8_t _batteryVoltToTriggerSystem = 11;
float _currentBatteryVolt = 0;
uint16_t _currentFuelValue = 0;
bool _switchState = false;
bool _gasHandleState = false;
bool _wheelState = false;
bool _engineState = false;
String _message = "";
enum DataType
{
  NOTHING,
  ENGWORKINGTIME,
  BATTRIGVOLT,
  CONNECTWIFI,
  ENGSTOPPEDDATE,
  MESSAGE,
  DATETIMENOW,
```

```
  CURRFUELVAL,
  ENGSTRTDDATE,
  WHEEL_ST,
  ENGINE_ST,
  SWITCH_ST,
  GASHANDLE_ST,
  BATTRYVAL
} dataType;
enum MessageType
{
  FUEL_CRITICAL,
  TAKE_NOTR_GASHANDLE,
  SWITCH_OPEN_GASHANDLE_NOTR,
  CONTROL_AKU_STATE,
  FUEL_LOW
};
static const char *MessageType_String[] = {
  "Yakıt miktarı kritik seviyededir!",
  "Gaz kolunu boş pozisyona alın!",
  "Kontak açıkken gaz kolu boş pozisyonda olmalı!",
  "Akünün durumunu kontrol ediniz",
  "Yakıt miktarı azalmıştır. Lütfen yakıt takviyesi yapınız !"
};
void ConnectWifi(String connString)
{
  connString.replace(" ", "");
  if (connString != "" && connString.indexOf(',') > -1)
  {
    int separateIndex = connString.indexOf(',');
    String ssid = connString.substring(0, separateIndex);
    String pass = connString.substring(separateIndex + 1);
    Connect(ssid, pass);
  }
```

```
}
void Connect(String ssid, String pass)
{
  WiFi.disconnect();
  digitalWrite(CONN_STATE, LOW);
  WiFi.hostname("ysmysm");
  WiFi.begin(ssid.c_str(), pass.c_str());
  unsigned long start = millis();
  while (millis() - start < 10000)
  {
    if (WiFi.status() == WL_CONNECTED)
    {
      digitalWrite(CONN_STATE, HIGH);
      break;
    }
    delay(100);
  }
  _connectionState = (WiFi.status() == WL_CONNECTED);
}
void ConnectFirebase()
{
  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
}
void SendDataToFirebase()
{
  ConnectFirebase();

  _switchState = digitalRead(SWITCH_PIN);
  _gasHandleState = digitalRead(GAS_HANDLE_PIN);
  _wheelState = digitalRead(WHEEL_PIN);
  _engineState = digitalRead(ENGINE_PIN);
  Firebase.setFloat("BatteryValue", _currentBatteryVolt);
  if (Firebase.failed())
```

```
    return;
  Firebase.setFloat("BatteryVoltToTriggerSystem", _batteryVoltToTriggerSystem);
  if (Firebase.failed())
    return;
  Firebase.setFloat("EngineRunTime", _engineWorkingTime);
  if (Firebase.failed())
    return;
  Firebase.setFloat("EngineState", _engineState);
  if (Firebase.failed())
    return;
  Firebase.setFloat("FuelValue", _currentFuelValue);
  if (Firebase.failed())
    return;
  Firebase.setFloat("GasHandleState", _gasHandleState);
  if (Firebase.failed())
    return;
  Firebase.setFloat("SwitchState", _switchState);
  if (Firebase.failed())
    return;
  Firebase.setFloat("WheelState", _wheelState);
  if (Firebase.failed())
    return;
  if (_message != "")
  {
    Firebase.pushString("Messages", _message);
    _message = "";
  }
}
void SetEngineWorkingTime(uint8_t minute)
{
  _engineWorkingTime = minute;
}
void SetBatteryVoltToTriggerSystem(uint8_t batteryVolt)
```

```
{
  _batteryVoltToTriggerSystem = batteryVolt;
}
void SetCurrentFuelVal(uint16_t value)
{
  _currentFuelValue = value;
}
void SetCurrentBattryVal(float value)
{
  _currentBatteryVolt = value;
}
void SetMessage(uint8_t msgInd)
{
  _message = MessageType_String[msgInd];
}
void setup() {
  Serial.begin(115200);
  pinMode(CONN_STATE, OUTPUT);
  pinMode(ENGINE_PIN, INPUT);
  pinMode(GAS_HANDLE_PIN, INPUT);
  pinMode(SWITCH_PIN, INPUT);
  pinMode(WHEEL_PIN, INPUT);
  Connect(WIFI_SSID, WIFI_PASSWORD);
}
void loop() {
  TakeDataComingFromMaster();

  if (!(WiFi.status() == WL_CONNECTED))
  {
    Connect(WIFI_SSID, WIFI_PASSWORD);
  }
  if (_connectionState)
  {
```

```
    SendDataToFirebase();
  }
  delay(100);
}
void TakeDataComingFromMaster()
{
  String coming;
  String type;
  String value;
  char c;
  while (Serial.available())
  {
    char c = Serial.read();
    coming += c;
  }
  coming.replace(" ", "");
  if (coming != "" && coming.indexOf(":") > -1)
  {
    uint8_t separateIndex = coming.indexOf(':');
    int secndSeparateIndex = coming.indexOf('@');
    type = coming.substring(0, separateIndex);
    value = coming.substring(separateIndex + 1, secndSeparateIndex);
    if (type != ""  && value != "")
    {
      switch (type.toInt())
      {
        case ENGWORKINGTIME:
          SetEngineWorkingTime(value.toInt());
          break;
        case BATTRIGVOLT:
          SetBatteryVoltToTriggerSystem(value.toInt());
          break;
        case CURRFUELVAL:
```

```
        SetCurrentFuelVal(value.toInt());
        break;
      case CONNECTWIFI:
        ConnectWifi(value);
        break;
      case MESSAGE:
        SetMessage(value.toInt());
        break;
      case BATTRYVAL:
        SetCurrentBattryVal(value.toFloat());
        break;
    }
   }
  }
}
```

KİŞİSEL YAYIN VE ESERLER

[1] **Ustaoglu Y.,** Küçük K., Deniz Araçları için Nesnelerin İnterneti Teknolojileri Temelli Şarj Kontrol Sistemi tasarımı, *6. Uluslararası Mühendislik ve Tasarım Kongresi (MMT)*, İstanbul, Türkiye, 17-18 Aralık 2020.

ÖZGEÇMİŞ

Yaşam Ustaoğlu, ilköğretim ve lise eğitimini İstanbul'da tamamlamıştır. 2012 yılında girdiği Sakarya Üniversitesi Bilgisayar Mühendisliği bölümünden 2016 yılında mezun olmuştur. Mezun olduktan sonra yazılım geliştirici olarak otomotiv ve e-ticaret projelerinde görev almıştır. 2021 yılında Kocaeli Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda yüksek lisans eğitimini tamamlamıştır. Yüksek lisans eğitimi sırasında, "Deniz Araçları İçin Nesnelerin İnterneti Teknolojileri Temelli Şarj Kontrol Sistemi Tasarımı" başlıklı bir çalışma yayınlamıştır. 2020 yılından itibaren özel bir bankanın bilgi işleminde yazılım geliştirici olarak görev almaktadır.