

Pamukkale Üniversitesi, Mühendislik Fakültesi  
Bilgisayar Mühendisliği Bölümü

*Lisans Bitirme Tezi*

**Elektrik Sistemlerinin   
Uzaktan Denetimi ve Yönetimi: intelliPWR**

*Tarafından*

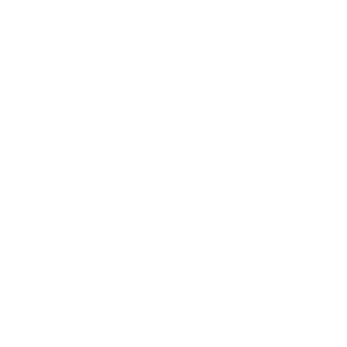
**Veysel Berk ALTUN   
13253004**

*Denetleyen*

**Yrd. Doç. Dr. Elif HAYTAOĞLU**

Mayıs 21, 2018

Denizli



Pamukkale Üniversitesi, Mühendislik Fakültesi  
Bilgisayar Mühendisliği Bölümü

*Lisans Bitirme Tezi*

**Elektrik Sistemlerinin   
Uzaktan Denetimi ve Yönetimi: intelliPWR**

*Tarafından*

**Veysel Berk ALTUN  
13253004**

*Denetleyen*

**Yrd. Doç. Dr. Elif HAYTAOĞLU**

Mayıs 21, 2018

Denizli

**LİSANS TEZİ ONAY FORMU**

Veysel Berk ALTUN tarafından Yrd. Doç. Dr. Elif HAYTAOĞLU yönetiminde hazırlanan “**Elektrik Sistemlerinin Uzaktan Denetimi ve Yönetimi: intelliPWR**” başlıklı tez tarafımızdan okunmuş, kapsamı ve niteliği açısından bir lisans tezi olarak kabul edilmiştir.

Yrd. Doç. Dr. Elif HAYTAOĞLU  
*Danışman*

|  |  |
| --- | --- |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Jüri Üyesi | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Jüri Üyesi |

Pamukkale Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölüm Kurulu’nun   
\_\_\_\_\_ / \_\_\_\_\_ / \_\_\_\_\_ tarih ve \_\_\_\_\_\_\_\_\_\_ sayılı kararıyla onaylanmıştır.

**Prof. Dr. Sezai Tokat***Bölüm Başkanı*

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

Veysel Berk ALTUN  
 Mayıs 2018

ÖNSÖZ

Tez çalışmam sırasında kıymetli bilgi, birikim ve tecrübeleri ile bana yol gösterici ve destek olan değerli danışman hocam sayın *Yrd. Doç. Dr. Elif HAYTAOĞLU*’na, ilgisini ve önerilerini göstermekten kaçınmayan Bilgisayar Mühendisliği Bölümü Ana Bilim Dalı Başkanı sayın *Prof. Dr. Sezai Tokat*’a sonsuz teşekkür ve saygılarımı sunarım.

Her türlü maddi ve manevi destekleriyle beni hiçbir zaman yalnız bırakmayan *Gülyesa ALTUN*’a ve aileme sonsuz teşekkür ederim.

Son olarak yardımlarını hiç esirgemeyen değerli arkadaşlarım *Gülşah ARAS*’a, *Osman YILMAZTÜRK*’e, *Egemen AYHAN*’a ve *Ahmet YILMAZ*’a da teşekkürlerimi bir borç bilirim.

Veysel Berk ALTUN  
 Mayıs 2018

**ÖZET**

Elektriğin ulaştığı her alanda hayata geçirilmesi planlanan bu projede günümüzün priz temelleri yeniden ele alınmış ve tasarımı konusunda inovatif bir fikir ortaya atılmıştır. Yeniden tasarlanan bu prizler ile gelecekte üretilecek olan yeni nesil akıllı ev aygıtlarının bu prizler ile haberleşmesi sağlanmış ve bu akıllı ev aygıtlarının prizler yardımıyla uzaktan denetimi ve yönetimi gerçekleştirilmiştir.

Ortaya atılan bu yeni priz tasarımı fikrinde, priz soketlerinin faz ve nötr pimlerini altına 5 adet küçük bağlantı pimi eklenmiş ve bu pimler yardımı ile prize bağlanan aygıtların geçerli priz ile haberleşmesi amaçlanmıştır.

Tezin Ar–Ge sürecinde kablolu haberleşme süreçlerinde kullanılması amacıyla *RaspBerry Pi 3*, *Arduino Mega*, *Arduino Pro Mini* ve *NodeMCU* platformları ile uyumlu 3 farklı kütüphane yazılmıştır. Bütün bu kütüphanelerin yazım aşamasında *C*, *C++* ve *Phyton* dilleri tercih edilmiş; daha kolay bir Ar–Ge süreci için ise *GIT* versiyon kontrol sistemi kullanılmıştır. Tezin kablosuz haberleşme süreçlerinde ise açık kaynak kodlu olan ve halen geliştirilmeye devam edilen *openHAB* sistemi tercih edilmiştir.

Sonuç olarak bütün proje kapsamında temel bilgisayar mühendisliği etiğine ve kurallarına uygun olarak fonksiyonel ve modüler toplam 4400 satır kod yazılmış ve çağımıza uygun yeni nesil bir priz tasarımı ortaya konmuştur.

**GİRİŞ**

Günümüzde insanlığın elektriğe olan ihtiyacını bir bitkinin suya olan ihtiyacı ile benzer şekilde tanımlayabiliriz. Bir bitki su olmadan nasıl en temel faaliyetlerini gerçekleştiremiyorsa, insanlar da artık elektrik olmadan yaşayamıyor ve gündelik faaliyetlerinin çoğunu gerçekleştiremiyorlar. Elektriğin insan hayatında bu derece önemli olduğu 21. Yüzyılda, kullanım alanları ve popülaritesi gün geçtikçe artan *“Nesnelerin Interneti – Iot (Internet of Things)”* ile yapılar içerisindeki elektrik kaynaklarına artık uzaktan erişebilmekte ve yönetebilmekteyiz.

İngilizce karşılığı *Internet of Things* olan *Nesnelerin İnterneti*, ilk kez *1999* yılında Britanyalı teknoloji öncüsü *Kevin Ashton* tarafından ortaya atılmış bir terimdir. En genel tanımıyla fiziksel aygıt, sensör, araç ve elektronik aletlerin, sahip oldukları yazılımsal ve donanımsal kaynaklar ile bir ağ üzerinden birbirleri ile haberleşmeleri olarak tanımlanabilir. Benzer biçimde çeşitli haberleşme protokolleri sayesinde birbirleri ile haberleşen ve birbirine bağlanarak akıllı bir ağ oluşturmuş aygıtlar sistemi olarak da tanımlamak mümkündür.

Nesnelerin İnternetine alternatif olarak günümüzde birçok benzer kavramlar vardır; ancak Nesnelerin İnterneti, bu fenomeni açıklamak için en popüler terimdir.

Günümüzde basit bir *IoT* aygıtının internet ile haberleşmesi aşamasında kullanılan en yaygın yöntem *Wi–Fi* teknolojisi olarak tanımlanabilir. Herhangi bir üreticinin *IoT* kavramına uygun bir aygıt üretmeyi planlaması durumunda, *Wi–Fi* teknolojisi üzerine bir Ar–Ge yapması kaçınılmazdır. Benzer biçimde üretmeyi planladığı yeni nesil bir akıllı ev aygıtına *Wi–Fi* teknoloji destekleyen bir donanım eklemesi gerekmektedir. Bütün bu süreçler ise bir üreticiye ek bir kaynak tüketimi oluşturur.

Tam da bu noktada, üreticilerin kaynak tüketimini daha aza indirmek amacıyla inovatif bir çözüm arayışı doğmuştur. Literatürde yapılan uzun araştırmalar sonucu, donanımlar arası *Wi–Fi* haberleşme altyapısını gruplar halinde gerçekleştirecek bir öneri ortaya atılmıştır.

Bu aşama bir *IoT* aygıtının elektrik olmadan çalışamayacağı varsayılarak, gruplar halinde *Wi–Fi* haberleşme teknolojisini sağlayacak olan yeni nesil priz tasarımı ortaya atılmıştır. Bu prizler ile üreticilerin *Wi–Fi* teknoloji ya da benzer herhangi bir haberleşme altyapısına ihtiyaç duymadan, aynı haberleşme sürecinin daha düşük bir bütçe ile gerçekleştirebilmesi amaçlanmıştır.

**I²C**

*I²C (Inter–Integrated Circuit),* *1982* yılında *Philips* tarafından geliştirilmiş, hızlı veri aktarımına olanak tanıyan, *Half–duplex (yarı eş zamanlı çift yönlü çalışabilen)* olarak çalışabilen toprağa referanslı bir seri veri yoludur. *1990*’ların ortasından beri *Siemens*, *NEC*, *STMicroelectronics*, *Motorola*, *Intersil* gibi pek çok firma *NXP* geliştirdiği *I²C* sistem standartlıyla tam uyumlu ürünler piyasaya sürmüşlerdir.

*1 Ekim 2006* itibariyle *I²C* protokolünün lisans zorunluluğu ortadan kaldırılmış, ancak *NXP* tarafından tahsis edilen *I²C* birim adresine sahip olabilmek için üreticilerin ücrete tabii tutulmuştur.

*I²C* protokolünde *SCL* (*SerialClock)* ve *SDA* (*SerialData)* olmak üzere temel iki haberleşme hattı vardır. *SCL,* aygıtlar arası veri senkronizasyonunu gerçekleştiren darbe hattı iken *SDA*, aygıtların birbirleri ile haberleşmesini sağlayan veri yolu hattıdır.

*I²C* protokolünde aygıtlar arası haberleşme *start (başla)* komutuyla başlatılır ve *stop (bitir)* komutuyla bitirilir. Bu protokolde yapılacak olan her haberleşme süreci, saat darbesinin *lojik 0* durumunda gerçekleştirilir. Aygıtlar arasında haberleşmenin başladığını veya tamamlandığını belirten *start* ve *stop* durum komutları ise sadece *SCL* hattının, yani saat sinyalinin *lojik 1* olduğu durumlarda gönderilir. *SCL* hattı *lojik 1* iken *SDA* hattının *lojik 1*’den *lojik 0*’a geçişi *start* komutu anlamına gelir. Bu durumda aygıtlar arası haberleşme başlatılmış olunur. Benzer şekilde *SCL* hattı *lojik 1* iken *SDA* hattının *lojik 0*’dan *lojik 1*’e geçişi *stop* komutu anlamına gelir ve bu durum haberleşen iki aygıt arasındaki haberleşmenin tamamlandığını anlamına gelir.

*I²C* protokolünde gönderilen ile okunan her veri *SDA* hattı üzerinden gerçekleştirilir. *SDA* pimi, sürekli olarak bir *Pull Up* direnci ile güç kaynağına bağlı olduğundan hattaki *start* ve *stop* bitlerinin anlaşılması hattın *lojik 0* olup olmamasıyla anlaşılır. Ayrıca hattın sürekli *lojik 1* olarak kurulması bazı elemanların yüksek empedans konumlarında bile hattan *lojik 1* okunmasına neden olur.

**MQTT**

*MQTT (Message Queuing Telemetry Transport)*, *1999* yılında *IBM*’de çalışan *Andy Clark* *ile* Cirrus*’da çalışan Arlen Nipper* tarafından *OASIS* şirketinde geliştirilmeye başlanmış açık kaynak kodlu bir haberleşme protokolüdür. Kısıtlı kaynağa sahip ağ istemcilerinin telemetri bilgilerini dağıtmasını kolaylaştıran hafif bir iletişim protokolüdür. Yayın – abone iletişim modelini kullanan *MQTT*, *M2M* ve *IoT* platformlarında önemli bir rol oynar.

Bu protokolde istemciler, ağdaki bir yayıncıya bağlanabilir ve aracı istemciler arasındaki iletişimi düzenleyebilir. Ağdaki bir aygıt abone olduğu konuyla ilgili bir mesaj yayınladığında, geçerli aygıt ağdaki diğer tüm abone aygıtlara aynı mesajı iletebilir. Her aygıt belirli konulara abone olabilir veya aboneliklerden çıkabilir.

*MQTT* protokolünün hafif yüklü tasarımı (2 baytlık üstbilgi), yayın – abone olma modeli ve çift yönlü iletişim yetenekleri, elektronik tabanlı kontrol sistemlerin taleplerini karşılamak amacıyla en uygun altyapı modeli olarak tanımlanabilir.

**openHAB**

*openHAB (open Home Automation Bus)*, *Java* programlama dilinde yazılmış, farklı ev otomasyon sistemlerini, cihazlarını ve teknolojilerini tek bir çözümde birleştiren yazılım çözümüdür. Bununla birlikte herhangi bir işletim sistemine bağlı olmaksızın ek bağlantılarla yeni teknolojilerle ve protokollerle uyumlu hale getiriliebilir. *OpenHAB* yazılımı, günümüzde halen *Eclipse Kamu Lisansı (EPL)* altında açık kaynak kodlu olarak geliştirilmektedir. Günümüzde *iOS* ve *Android* temelli bütün aygıtlarda yazılımsal olarak desteklenmektedir.

**ÖNHAZIRLIK VE KURULUM**

Tezin uygulama süreçlerinde yapılan testlerin ve uygulamaların tamamında *Ubuntu* işletim sisteminin *14.04* sürümü tercih edilmiştir. Bu nedenle örnek bir uygulama sürecinde yapılacak olan bütün uygulamaların, bu işletim sisteminin daha eski sürümlerinde yapılacak olan uygulamalar ile aynı sonucu vereceği kesinlikle teyit edilemez.

Bir diğer önemli konu, *MQTT* haberleşme sürecinde zorunlu olarak gereken aktif internet bağlantısı üzerinedir. Bu aşamada *port blocking* özelliği olmayan *WPA2 Personal* kimlik doğrulamalı bir bireysel ağ yapısı tercih edilmiştir. Tezin uygulama süreçlerinin güvenlik üzerine yeterli bir Ar–Ge çalışması içermemesinden dolayı kamuya açık olan ağların bu tez sürecinde kullanılması önerilemez. Tercihen kablolu internet bağlantısı da alternatif olarak kullanılabilir.

*Ubuntu* işletim sisteminin kurulumu hakkında detaylı bilgi için;  
<https://www.ubuntu.com/download>

**I²C**

*I²C* protokolü, *UART* veya *SPI* protokollerinden farklı olarak *open–drain* yapıya sahiptir. Sıradan bir *open–drain* hattında geçerli pimin çıkış bacağı *P–MOSFET* ile *N–MOSFET* devre elemanları arasında bulunmaktadır. *open–drain* durumuna getirilmiş bir bağlantı hattında *MOSFET* elemanının *source* hattı toprağa bağlıdır, *gate* içeriden sürülmüş durumdadır ve *drain* açıktadır. Bu aşamada faz ile geçerli bu pim arasında sonsuz empedans oluşur. Bu durumda pimi *high* konumuna getirebilmek için dışarıdan uygun değerlikte bir *pull–up* direncinin eklenmesi gerekmektedir. Bu özellikle, çıkışa bağlanacak olan devre elemanının daha fazla akım çekmesi sağlanır ve böylece akım mikroişlemciden değil, *pull–up* direnci üzerinden çekilir.

Uygulama sürecinde *I²C* protokolünün bütün *SCL* ve *SDA* hatlarına birer adet *pull–up* direnci bağlanmıştır. Dirençlerin değerleri, yarı iletken üreticisi *Microchip Technology*’nin *ATmega8*, *ATmega168* ve *ATmega328* modelleri için 2016 yılında güncel olarak yayınlamış olduğu kullanım kılavuzundaki formüller ile hesaplanmıştır. Verilen formüller;

Formüllerde verilen değeri *SCL* hattının çalışma frekansını, değeri *pull–up* direncinin sahip olabileceği en küçük direnç değerini, değeri hattın sahip olduğu gerilim değerini, değeri *pull–up* direncinin sahip olabileceği en büyük direnç değerini ve son olarak değeri *I²C* hattında sahip olunan toplam aygıt sayısını temsil eder.

Formüldeki bütün değişkenlerin uygun değerler ile değiştirilmesi sonucu en uygun *pull–up* direnç değeri hesaplanabilir. Bu tezin uygulama süreçlerinde kullanılacak olan bütün *pull–up* direnç değerleri olarak baz alınmıştır.

*I²C* protokolü ve kullanımı hakkında detaylı bilgi için;   
<https://www.i2c-bus.org/>

**ARDUINO**

*Arduino* platformundaki bütün uygulama geliştirme süreçlerinde *Arduino IDE* yazılımının *1.8.5* sürümü tercih edilmiştir.

*Arduino IDE* yazılımının kurulumu hakkında detaylı bilgi için;  
<https://www.arduino.cc/en/Main/Software>

**NodeMCU**

*NodeMCU* platformu her ne kadar *Lua* *Scripting* programlama dilini benimsemiş olsa da uygun yapılandırmalar ile *NodeMCU* platformunun bütün uygulama geliştirme süreçleri *Arduino IDE* yazılımıyla da yapılabilir.

*NodeMCU* platformunun *Arduino IDE* kurulumu hakkında detaylı bilgi için; <https://github.com/esp8266/Arduino>

**openHAB**

*openHAB* yazılımının *Ubuntu* işletim sisteminde kararlı çalışabilmesi için işletim sistemine uygun bir *Java SE Runtime Environment* dağıtımı yazılım gerekmektedir. Buna ek olarak *Java SE Runtime Environment* yazılımın *1.9.x* sürümleri *openHAB* ile kararlı bir şekilde çalışmadığından, bu yazılımın *1.8.x* sürümleri kullanıcılar için resmi olarak önerilmektedir.

Tezin uygulama süreçlerindeki uzak bağlantılar için *openHAB Foundation* tarafından barındırılan *openHAB* *Cloud* altyapısı kullanılmıştır. Bu altyapının, *openHAB* yazılımı ile kararlı çalışabilmesi için bir takım yapılandırma ayarlarının yapılması gerekmektedir.

|  |  |  |
| --- | --- | --- |
| **Dosya** | **Normal Kurulum** | **APT Kurulumu** |
| UUID | userdata/uuid | /var/lib/openhab2/uuid |
| Secret | userdata/openhabcloud/secret | /var/lib/openhab2/openhabcloud/secret |

*openHAB Cloud* ile kimlik doğrulaması için yerel *openHAB* yazılımı, *openHAB Cloud* servisinin hesap ayarlarınızda girilmesi gereken iki değer üretir. İlki, çalışma zamanınızı tanımlamaya izin veren benzersiz bir kimlik tanımlayıcıdır. İkincisi, şifre olarak hizmet sunan rastgele bir gizli anahtardır. Her iki değer de yerel dosya sistemine yazılır.

Bu dosyaları bir sebepten dolayı hasar görmesi durumunda, *openHAB* yazılımı otomatik olarak yenilerini oluşturur. Daha sonra *UUID* ve *SECRET* değerlerini *openHAB Cloud* hizmetinde yeniden yapılandırmanız gerekecektir.

*openHAB* yazılımının kurulumu hakkında detaylı bilgi için;   
<https://docs.openhab.org/installation>

*openHAB Cloud Connector* hakkında detaylı bilgi için; <https://docs.openhab.org/addons/ios/openhabcloud/readme>

*MQTT Binding* hakkında detaylı bilgi için; [https://docs.openhab.org/addons/bindings/mqtt/readme](https://docs.openhab.org/addons/bindings/mqtt1/readme)

**MQTT**

Günümüzde IoT platformlarında *Mosquitto*, *Kafka*, *RabbitMQ*, *emqttd* ve *VerneMQ* olmak üzere tercih edilebilecek en az *5* adet *MQTT* yazılımı bulunmaktadır. Bu tezin uygulama süreçlerinde kurulumu ve kullanımı en kolay olan *Mosquitto* *MQTT* yazılımı tercih edilmiştir.

*openHAB* yazılımının bir *MQTT* istemcisi olarak çalışmasını sağlamak için bir takım yapılandırma ayarlarının yapılması gerekmektedir. Bu yapılandırmalar *openHAB* yazılımına *MQTT* işlevselliği kazandırmaz, bunun için *Mosquitto* veya bir başka altyapı yazılımının kullanılması gerekmektedir.

|  |  |  |
| --- | --- | --- |
| **Özellik** | **Varsayılan** | **Zorunlu** |
| *<broker>*.url | - | Evet |
| *<broker>*.clientId | Rassal | Hayır |
| *<broker>*.user | - | Evet |
| *<broker>*.pwd | - | Evet |
| *<broker>*.qos | 0 | Hayır |
| *<broker>*.retain | Yanlış | Hayır |
| *<broker>*.async | Doğru | Hayır |
| *<broker>*.keepAlive | 60 | Hayır |
| *<broker>*.allowLongerClientIds | Yanlış | Hayır |

Bir *MQTT* yayıncısına mesaj göndermek veya yayınlamak için, *services/mqtt.cfg* dosyasına kullanılması planlanan tüm aracıların bildirimlerinin yapılması gerekmektedir.

*Mosquitto* yazılımının kurulumu hakkında detaylı bilgi için;   
<https://mosquitto.org/download>

**DONANIMLAR**

**ARDUINO PRO MINI**

*Arduino Pro Mini*, *ATmega328* tabanlı bir *Arduino* geliştirme kartıdır. Bu geliştirme kartı üzerinde *14* adet dijital giriş – çıkış pimi (bu pimlerden *6* tanesi *PWM* çıkışı), *6* adet analog giriş – çıkış, bir adet kristal osilatör (*8 MHz* ya da *16 MHz*) ve bir adet *reset* butonu bulundurur. Sahip olduğu entegre voltaj regülatörü ile *12V* gerilime kadar voltaj beslemelerine izin vermektedir. Regüle edilmemiş voltaj beslemesi *RAW* piminden yapılmaktadır.

*Arduino Pro Mini* geliştirme kartının *3.3V 8 MHz* ve *5V 16 MHz* olmak üzere iki farklı versiyonu bulunmaktadır.



**NodeMCU**

*NodeMCU*, *ESP8266-12* tabanlı bir geliştirme kartıdır. Bu geliştirme kartı üzerinde *17* adet dijital giriş – çıkış pimi (bu pimlerden *9* tanesi *PWM* çıkışı), *1* adet analog giriş – çıkış, bir adet *80 MHz* kristal osilatör, bir adet 802.11 b/g/n *Wi–Fi* modülü ve bir adet *reset* butonu bulundurur. Sahip olduğu entegre voltaj regülatörü ile *12V* gerilime kadar voltaj beslemelerine izin vermektedir.

