T.C.
KONYA TEKNİK ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

UHF ARAÇ TANIMA EKİPMANLARI TASARIMI

Hasan SELEK

YÜKSEK LİSANS

Elektrik-Elektronik Mühendisliği Anabilim Dalı

Ağustos-2019
KONYA

TEZ KABUL VE ONAYI

Hasan SELEK tarafından hazırlanan "**UHF Araç Tanıma Ekipmanları Tasarımı**" adlı tez çalışması 19/08/2019 tarihinde aşağıdaki jüri tarafından oy birliği / ~~oy çokluğu~~ ile Konya Teknik Üniversitesi Lisansüstü Eğitim Enstitüsü Elektrik-Elektronik Mühendisliği Anabilim Dalı'nda YÜKSEK LİSANS olarak kabul edilmiştir.

Jüri Üyeleri **İmza**

Başkan
Dr. Öğr. Üyesi Muhammed Fahri ÜNLERŞEN

Danışman
Doç. Dr. Seyfettin Sinan GÜLTEKİN

Üye
Dr. Öğr. Üyesi Dilek UZER

Yukarıdaki sonucu onaylarım.

Prof. Dr. Hakan KARABÖRK
Enstitü Müdürü

TEZ BİLDİRİMİ

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

DECLARATION PAGE

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

Hasan SELEK

Tarih: 27.08.2019

ÖZET

YÜKSEK LİSANS

UHF Araç Tanıma Ekipmanları Tasarımı

Hasan SELEK

Konya Teknik Üniversitesi
Lisansüstü Eğitim Enstitüsü
Elektrik-Elektronik Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Seyfettin Sinan GÜLTEKİN

2019, 118 Sayfa

Jüri
Doç. Dr. Seyfettin Sinan GÜLTEKİN
Dr. Öğr. Üyesi Muhammed Fahri ÜNLERŞEN
Dr. Öğr. Üyesi Dilek UZER

Araç tanıma ekipmanı veya taşıt tanıma sistemi genelde ticari araç filolarının akaryakıt alımı ve filolardaki araçların yakıt takibinin yapılması için çözüm sunan akaryakıt yönetim sistemidir. Bu sistem daha önce kontrol ünitesi kısmının akaryakıt pompasının içine yerleştirildiği ve bu ünitenin pompanın kontrolünü sağladığı bir sistemdi. İnternetin yaygınlaşması ve teknolojinin gelişmesi ile pompanın kontrolü artık veri tabanları üzerindeki kayıtlar kullanılarak yapılmaya başlanmıştır. Böylelikle veri tabanı ile haberleşen otomasyon sistemlerine pompanın kontrolü dahil olmuş oldu. Otomasyon sistemine ise veri tabanına sorgulaması için araç bilgilerini taşıt tanıma ekipmanlarınca bilgi aktarımı yapılmaktadır. Fakat bu aşamada araçtan bilgi okunması, değişen depo girişlerinden dolayı sağlıklı olmamaktadır. Böylelikle araç tanıması yapılamamaktadır.

Bütün bunları önleyebilmek için bu tez çalışmasında bir taşıt tanıma sistemi geliştirilmiş ve elektronik prototipi gerçekleştirilmiştir. Geliştirilen elektronik düzenekle birlikte değişken depo girişlerinden dolayı yaşanan sıkıntıların önüne geçilmiştir. Böylece tabanca ucundaki okuyucu ünite ile kimlik birimi arasındaki haberleşme zorlukları aşılmıştır. Ayrıca pasif kimlik birimleri ile elde edilemeyen mesafelerde kimlik birimi ile haberleşme sağlanabilmiştir. Bu sayede hiç yakıt alamayan büyük araçlar ve yakıt alırken kesilmeler yaşayan diğer araçların sorunları çözülebilmiştir. Bu işlemler için kimlik birimi enerji ile çalışır hale getirilmiş ve radyo frekansı ile haberleşme sağlanmıştır. Okuyucu ünitenin kimlik birimini değişik açılardan düzgün olarak okuyabilmesi içinde yeni bir monopol anten tasarlanmış, üretimi yapılarak sisteme dahil edilmiştir. Sistem birçok araçta denenmiş ve okuyamama probleminin ortadan kalktığı görülmüştür.

Anahtar Kelimeler: PCB antenler, RSSI, Taşıt tanıma sistemi, UHF bandı

ABSTRACT

MS THESIS

UHF Vehicle Recognition Equipment Design

Hasan SELEK

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

Advisor: Assoc. Prof. Dr. Seyfettin Sinan GÜLTEKİN

2019, 118 Pages

Jury
Assoc. Prof. Dr. Seyfettin Sinan GÜLTEKİN
Asst. Prof. Dr.Muhammed Fahri ÜNLERŞEN
Asst. Prof. Dr.Dilek UZER

Vehicle recognition equipment or vehicle identification system is generally a fuel management system that provides solutions for fuel purchase of commercial vehicle fleets and fuel tracking of vehicles in fleets. This system was previously a system in which the control unit part was placed in the fuel pump and this unit provided control of the pump. With the expansion of the internet and the development of technology, the control of the pump has now started to be made using records on the databases. Thus, the control of the pump was included in the automation systems communicating with the database. On the other hand, vehicle information is transferred to the automation system by vehicle recognition equipment for querying the database. However, reading information from the vehicle at this stage is not healthy due to changing warehouse entries. Thus, vehicle recognition is not possible.

In order to prevent all these, a vehicle identification system has been developed and electronic prototype has been realized in this thesis. With the electronic device developed, the problems experienced due to variable storage entrances are prevented. Thus, communication difficulties between the reader unit and the identification unit at the gun tip are overcome. In addition, communication with the identification unit could be achieved at distances that cannot be obtained with passive identification units. In this way, the problems of big vehicles that could not buy any fuel and other vehicles that had interruptions while buying fuel could be solved. For these operations, the identification unit is made operational with energy and communication with radio frequency is provided. For the reader unit to read the identification unit properly from different angles, a new monopoly antenna was designed, manufactured and included in the system. The system has been tried in many vehicles and it has been found that the problem of not being able to read is eliminated.

Keywords: PCB antennas, RSSI, Vehicle identification system, UHF band

ÖNSÖZ

Yüksek lisans süresince bana danışmanlık yapan ve her zaman her konuda desteğini esirgemeyen Sayın Doç. Dr. Seyfettin Sinan GÜLTEKİN hocama sonsuz teşekkürlerimi sunarım.

Ayrıca, her zaman bana yardımcı olan ve yol gösteren Dr. Öğr. Üyesi Dilek UZER, hocama çok teşekkür ederim.

Canımdan çok sevdiğim, eşime, çocuğuma, anneme, kardeşime ve Bilgili ailesine bu süreçteki sabırlarından dolayı teşekkürü bir borç bilirim. Hayat boyu hep yanımda oldunuz ve hiç yalnız bırakmadınız.

Ve yüksek lisansımın bittiğini görmeyi çok isteyen, hayatı boyunca iki çocuğunu yetiştirmek için çalışan, rahmetli babamı burada rahmetle anıyorum.

Hasan SELEK

KONYA-2019

İÇİNDEKİLER

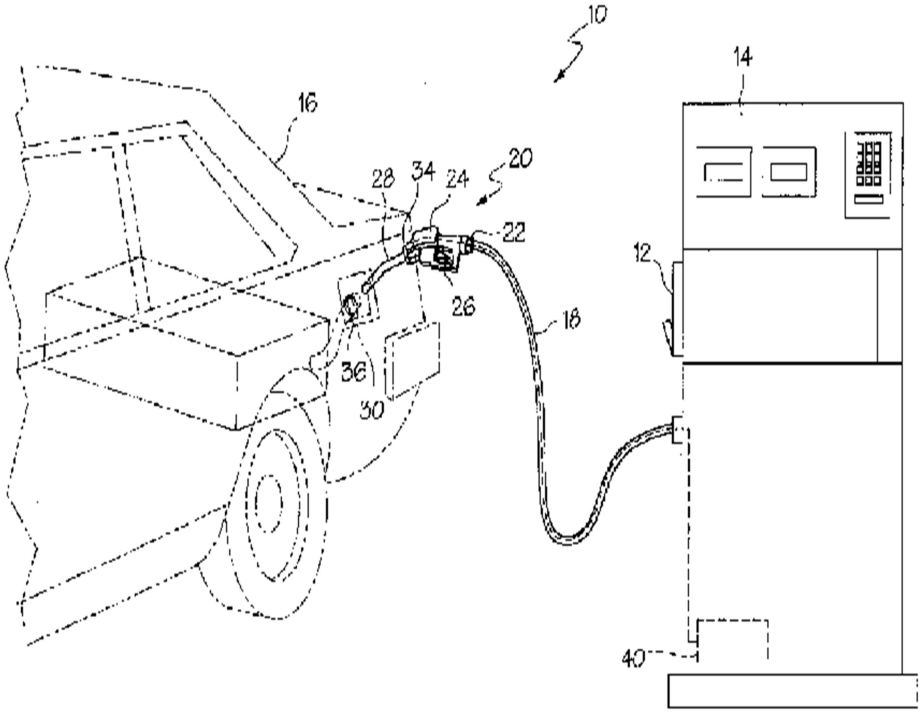
SİMGELER VE KISALTMALAR

Kısaltmalar

LF	Alçak Frekans
kHz	Kilo Hertz
TTS	Taşıt Tanıma Sistemi
PWM	Darbe Genişlik Modülasyonu
ISM	Endüstriyel Bilimsel Tıbbi
RF	Radyo Frekansı
RFID	Radyo Frekanslı Tanımlama
SoC	Çip Üzerinde Sistem
RSSi	Alınan Sinyal Gücü Seviyesi
TOA	Geliş Zamanı
TDOA	Varış Zaman Farkı
EKG	Elektrokardiyografi
UHF	Ultra Yüksek Frekans
ITU	Uluslararası Telekomünikasyon Birliği
ADC	Analog Dijital Çevirici
DAC	Dijital Analog Çevirici
RAM	Okunup Yazılabilen Hafıza
EPROM	Silinebilir Programlanabilir Okunabilir Hafıza
EEPROM	Elektriksel Silinebilir Programlanabilir Okunabilir Hafıza
RTC	Gerçek Zamanlı Saat
UART	Evrensel Asenkron Alıcı Verici
LCD	Sıvı Kristal Ekran
LEL	Düşük Patlama Sınırı
IEEE	Elektrik ve Elektronik Mühendisleri Enstitüsü
WLAN	Kablosuz Yerel Alan Ağı
MAC	Ortam erişim kontrolü
PCB	Baskı Devre Kartı
ATEX	Patlayıcı Atmosfer Kullanım Sertifikası
R	Rezistans
L	Endüktans
C	Kapasitans

1. GİRİŞ

Taşıt tanıma sistemi (TTS), pek çok farklı şekillerde karşımıza çıksa da genel olarak ticari araç filolarının akaryakıt alımı ve filolardaki araçların yakıt takibinin yapılması problemlerini teknoloji ile çözen, otomatik bir filo araçlarının akaryakıt yönetim sistemidir. Bu sistem bugünkü sisteme en yakın ve ilk olarak: 25 Ocak 1997 yılında patentleşmiştir. Bu patentte kontrol ünitesi akaryakıt pompasının içine yerleştirilir ve bu ünite pompanın kontrolünü sağlar. Bu kontrol içinde pompanın aktive edilmesi hangi üründen verileceği ve geri dönüş sisteminin kontrolü vardır. Bu sistemde sürücü devresi tabanca borusunda yer alır ve kontrol devresi ile haberleşir. Bunun için yakıt hortumu içinden kablo geçirilir ve güvenli bir bağlantı yapılır. Bu bağlantı üzerinden kontrol ünitesi ve sürücü devresi programlanmış periyotlar ile düşük güçteki sinyaller ile haberleşir. Bu sürücü devresi tabanca dolum ağzına takıldığında araçtaki tagı enerjilendirir ve sinyale karşılık olarak taşıt kimlik numarası içeren dönüş sinyali alarak haberleşmeye başlar. Bu kimlik numarası taşıt ile ilgili gerekli bilgileri yakıt tipi gibi belirler. Kontrol ünitesi araç kimlik numarasını yorumlar ve akaryakıt pompasına aracın ihtiyacına uygun sinyal üretir. (United States Patent,1997) Aşağıdaki resimde patente ait genel görünüm bulunmaktadır.



Şekil 1.1. Patenti yer taşıt tanıma sistemi

1.1. Günümüzde TTS ve Bileşenleri

Günümüzde taşıt tanıma sistemleri yaygın olarak daha önce bahsedilen patent şeklinde işlemektedir. Yalnız gelişen ve yaygınlaşan internet kullanma imkanları sayesinde artık kontrol panelinin verdiği yakıt tipi gibi bazı kararlar taşıt tanıma sisteminin bağlı olduğu otomasyon panelinin bağlandığı server makineler üzerinde yapılmaktadır. Bu server makinelerde yakıt alacak araca ait yakıt tipi, bu araca ait yakıt limiti gibi pompanın aktive olup olmaması gerektiğini belirleyen bazı bilgiler ve bu aracın hangi zamanlarda ne kadar yakıt aldığını, eğer uygun cihaz var ise bu aracın yakıt tüketimin kilometre başı ölçümü bilgileri bulunmaktadır. Bu özellikleri sayesinde ise filo yönetimi yapmak zorunda kalan şirketler araçlarına ait tüm bilgilere internet üzerinden kolayca erişebilmektedir. Ayrıca taşıt tanıma sistemleri merkezi olarak yönetilmesi sayesinde filoya ait araçlar anlaşmalı olduğu akaryakıt dağıtım şirketinin bu sisteme sahip tüm istasyonlarından ücret ödemeden yakıt alabilmektedir ve bu sayede filo yöneticilerinin çalışanlarına yol boyunca yetecek yakıt miktarını tahmin etmek ve bu miktarının ücretini araç şoförüne vermek gibi bir zorunluluğu kalmamaktadır. Bunlara ek olarak taşıt tanıma sisteminin en büyük özelliği olan, akaryakıt pompasının sadece tabanca aracın yakıt dolum ağzında olduğunda çalışması ve uzaklaştığında ise yakıt vermeyi kesmesi filo yöneticileri için aracın şoförüne güvenme zorunluluğunu ortadan kaldırmakta ve herhangi bir yakıt kaçakçılığına izin vermemektedir. Ve bu talepleri karşılamak amacı ile hemen hemen her akaryakıt dağıtım şirketi bu uygulamayı kendi bünyesinde barındırmak istemektedir.

Yaygın olarak görülen taşıt tanıma sistemleri elektroniksel ve genel olarak alçak frekans araç kimlik ünitesi, bu ünite ile ve otomasyon sistemi ile haberleşen okuyucu ünite olmak üzere iki kısımdan oluşur.

Bu ünite içinde değiştirilemez kimlik numarası ve tasarımına göre aracın plakası, hangi dağıtım şirketine ait olduğu gibi bilgileri barındırmaktadır. Ayrıca bu cihazlar sadece bu cihazları üreten firmaların okuyucuları tarafından okunabilmesi için şifrelerde barındırabilir. Bu birim yaygın iki şekilde kullanılmaktadır. Bunlar:

1.1.1. Aktif kimlik ünitesi

Bu kimlik ünitesi araçtan enerji alma ihtiyacı duyan, bazı modellerinde araç kilometre bilgisini araçtan alabilme veya aracın tekerleğin dönmesinden hesaplayabilme özelliği olan cihazlardır. Bu tip birimler genelde araçta güvenli bir bölgeye yerleştirilir ve haberleşme için kullanılan kısmı ise aracın depo ağzına takılır. Pasif kimlik birimlerine göre daha uzun

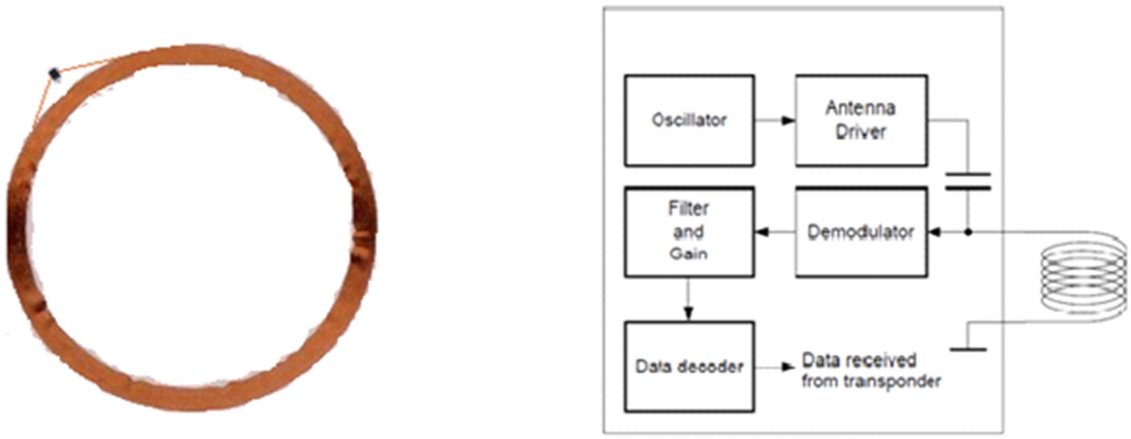
mesafeden haberleşen bu cihazların haberleşme kısmı LF (düşük frekans)'ta çalışırlar. Yerinden sökülmeye karşı genelde korumalı olup yerinden sökülürse pasif hale geçerler ve bu sayede bir araçtan sökülen cihaz diğer araca takılamaz. Aktif kimlik biriminde yerinden sökülmeye karşı koruma her firma tarafından farklı yöntemlerle yapılmaktadır. Örneğin cihazın bir yere vidalanması ve sökülürken sinyal üretmesi bunlardan biridir. Araçta depo ağzına takılan kısma anten adı verilir ve bu anten değişik büyüklükteki depo ağızlarına uygun olması için değişik ebatlarda tasarlanırlar. Ve okuyucu taraftaki bobinin genel olarak 125 kHz sinyal ile enerjilenmesine bağlı olarak oluşan uyartımı aracın içindeki ana işlemciye iletir. Burada ana işlemci kimlik numarasını yine bu anten üzerinden okuyucunun sinyalini bozarak okuyucunun bu kimlik numarasını almasını sağlar. Günlük hayatta kullandığımız transformatörler ile aynı mantıkta olup ardaki nüve yerine hava kullanılmaktadır. Primer ucu okuyucu taraftaki bobin, sekonder ucu ise araçtaki antendir. Primer uca uygulanan sinyalin sekonder tarafta kısa devre edilip serbest bırakılmasına dayanan bir yöntemdir. Bu işlem için bilinen en kolay yöntem bir transistor yardımı ile anten olarak adlandırılan bobinin uçlarının kısa devre edilmesidir. Geliştirilen yöntemler ile bu işlem farklı olarak da yapılabilir. Burada okuma mesafesinin uzun olması için en önemli faktör antenin bobin değerine karşılık uygun eşleşme devresinin yapılmasıdır. Bu değer cihazın araç içinde bulunduğu yerden uzaklığı ile değişebileceğinden dolayı anten ile cihaz arasındaki kablo boyu belirli sınırlar arasında tutulmaya çalışır.

Bu cihazlar araçtan enerji almak zorunda oldukları için değişik araç modellerinde değişik voltaj değerleri ile çalışmak zorundadırlar. Bunun için bu cihazların besleme katmanları her araca uygun olarak çalışmalıdır. Bu noktada lineer regülatörlerin giriş voltaj değerleri yeterli olamayabilir. Bunun için yaygın olarak anahtarlamalı güç kaynakları (switching mode power supply) kullanılmaktadır. Bunlar ile ilgili detaylı bilgi daha sonraki bölümlerde verilecektir.

1.1.2. Pasif kimlik ünitesi

Bu tip kimlik üniteleri kurulumu çok hızlı yapılabilen, aktif kimlik birimine göre biraz daha kısa mesafelerden haberleşebilen, daha düşük maliyetli kimlik birimleridir. Kurulum kolaylığından dolayı aktif kimlik birimin kullanılamayacağı yerlerde bile rahatlıkla kullanılabilir. Bu kimlik birimi tek parça olup aracın depo ağzına monte edilir. Burada aktif kimlik birimindeki antene enerjisini antenin bobininden alan bir adet çip, gerekli ise bobinle kompanzasyonu sağlayacak kondansatör kullanılır. Burada kullanılan çipler, içerisinde bobinden gelen alternatif akımı doğrusal akıma çeviren regülatör, gelen sinyaldeki gürültüleri

arındırmak için filtre devresi, üretimde kendisine verilmiş değiştirilemez kimlik numarası, haberleşmesini sağlamak için anten sürücü devresi, antenden gelen sinyali demodüle eden demodülatör, data dekoder, çipin çalışmasını sağlamak için osilatör, bobine uygulanan sinyali daha iyi algılamak için güçlendirme devresi ve bazı tiplerinde eeprom' a sahiptir.



Şekil 1.2. Pasif kimlik birimi

Yukarıda piyasada çok yaygın olarak kullanılan EM Microelectronic-Marin SA firmasına ait EM4200 serisine ait blok diyagramı bulunmaktadır. Resmin sol tarafında ise pasif kimlik biriminin resmi bulunmaktadır ki burada kullanılan çipin boyutunun küçüklüğü görülmektedir. Bu resim günümüzde kullanımı yaygınlaşan çoğu giriş kapısında kullanılan kartların içyapısı ile aynıdır. Yalnız bu uygulamalarda enerjinin daha çok alınabilmesi için kartlardaki bobin değerlerinden büyük değerler kullanılır. Sistemin diğer parçası okuyucu ünite ise kullanılan teknoloji ile beraber değişik özelliklere sahip olabilirler. Bu cihazlarda genel olarak değişmeyen kısım ise araç kimlik birimi ile haberleşen sürücü devresidir. Bu devre için özel üretilmiş özel işlemciler kullanılacağı gibi, herhangi bir işlemci ile PWM sinyal üretilerek bobin enerjilendirilebilir ve bobinin diğer ucundan ise dönüş sinyali demodule edilip haberleşme sağlanabilir.

1.1.3. Okuyucu ünite

Okuyucu ünitenin en çok değişkenlik gösteren kısmı ise okunan kimlik numarasının otomasyona yollanması ile görevli haberleşme kısmıdır. Bu kısım yaygınlaşan kablosuz haberleşme teknolojileri ile beraber gelişim gösterirken, pil ömrü gibi bazı sorunlardan dolayı kablolu uygulamalar devam etmektedir.

Kablolu uygulamalarda daha önce bahsedilen patentteki gibi otomasyon ile haberleşen kısım pompanın içinde ve kimlik birimi ile haberleşmesi için anten diye adlandırılan bobin pompanın tabancasında yer alır. Burada hortumun içinden akaryakıta dayanıklı kablolar ile kontrol devresi ve anten birbirine bağlanır. Kablolu uygulamalarda kontrol devresi birden çok tabancaya bağlı bobini kumanda edebilir. Bu ise sistem için maliyeti düşürmektedir. Kontrol devresinin otomasyon ile bağlantısı yine kullanılan teknolojiye göre RS485, Ethernet gibi yöntemlerle yapılabilir. Otomasyonun kolay kurulumu için kullanılan kablosuz cihazların kontrol ünitesine bağlanması yine mümkündür.

Kablosuz uygulamalarda ise kontrol ünitesi ve bobin tek parça olup tabanca üzerine monte edilir. Kablo bağlantısı yapılması gerekmediği için kurulumu daha kolaydır. Ayrıca sisteme ek olarak değişken kablo boyunun girmesi mümkün değildir ve bu okuyucu kısımda bobine karşılık yapılan kompanzasyon devresinin daha başarılı olmasını sağlar. Buradaki en büyük dezavantaj ise cihaz için kullanılabilecek alanın, kısıtlı ve her an dış etkenlere maruz kalmasıdır ki bu cihazın hem çabuk yıpranması, kırılması gibi sorunlara sebep olur. Kısıtlı alandan dolayı cihazın tasarımında kullanılacak enerji kaynağının küçük boyutlarda olması gerekmektedir ve bu boyutlarda mümkün olan en uzun çalışma ömrünü sağlamak için çeşitli yöntemler geliştirilmelidir. Bunun için öncelikli olarak tabancanın pompanın üzerinde durduğu süre içinde bu cihazın uyku modunda beklemesi, kimlik birimi olmayan araçlarda çalışmaması doğru olacaktır. Bunun yapılması için değişik yöntemler geliştirilmiştir.

Kablosuz bu birimin otomasyona bağlantısını sağlamak için mutlaka kablosuz veriyi otomasyona uygun şekle dönüştürecek bir cihaz kullanılmalıdır. Bu cihaz tüm istasyon için bir tane olabileceği gibi farklı sayılarda da olabilir.

Bu çalışmada daha önce bahsedilen taşıt tanıma sistemlerinde de bulunan kimlik birimi, tabanca ekipmanı ve tabanca ekipmanından kablosuz haberleşme ile gelen verileri otomasyon sisteminin kullanabileceği şekle dönüştüren merkezi toplayıcı birim adı verilen cihazların tasarımı ve uygulaması gerçekleştirilmiştir. Burada tasarlanan kimlik birimi ve tabanca kiti arasında ISM bandında haberleşme kurularak aracın tanınması sağlanmış, taşıt tanıma sisteminin en kritik noktası olan tabancanın araç deposundan çıkarıldığında yakıt akışının durdurulması için ise cihazlar arasındaki RF yayın gücünün ölçümü yapılmasından faydalanılmıştır. Araç kiti ve tabanca ekipmanında kullanılan bu sistem sayesinde tabanca ekipmanı üzerinde tanımlama yapabilmek için ekstra RFID devrelerinin kullanılmasına gerek kalmamıştır. Ayrıca şu an için yaygın olarak kullanılan 125 kHz frekansta çalışan RFID sistemlerin yaşadığı en büyük sorun olan okuma mesafesinin kısalığı bu sistemde aşılmıştır.

Sistemin temelindeki RF yayın ve diğer özellikler için SoC (System-on-Chip) bir mikrodenetleyici kullanılmış ve bu sayede kullanılan devrenin boyutları küçültülebilmiştir. Bu mikro denetleyicinin aşırı düşük güç harcama özelliği sayesinde tabanca ekipmanı kullanılan pil ile uzun süreler çalışabilmektedir. Tüm devrenin aktif yapısı bu mikrodenetleyici içinde olması dolayısı ile mikrodenetleyicinin uyku moduna geçmesi ile tüm devrenin kullandığı enerji miktarı çok düşük seviyelere gelebilmektedir.

Bu tez çalışmasında taşıt filolarında sıklıkla kullanılan radyo frekansı ile haberleşen taşıt tanıma sistemi tasarlanmıştır. Tezin amacı ise TTS'de değişen depo girişi ile okuyucu ünite arasındaki haberleşme zorlukları ortadan kaldırmaktır. Tezin ilk bölümünde genel olarak taşıt tanıma sistemleri ve bileşenlerinden bahsedilmiş, daha sonra bu sistemde kullanılan UHF bandından ve bunun uygulamaları, RSSI ve literatür araştırmalarına yer verilmiştir. Daha sonraki bölümlerde ise sistemin çalışacağı ortam olan patlayıcı ortam ve bu ortamda patlamadan korunma yöntemleri anlatılmış, bu frekansta çalışan antenlerin genel özellikleri ve ışıma örüntülerinden bahsedilmiş ve sistemin güvenliği için gerekli şifreleme algoritmaları ele alınmıştır tasarımın anlatıldığı bölümde ise sistemin genel işleyişi, bileşenlerinin özellikleri ve çalışma sistemleri anlatılmıştır.

2. KAYNAK ARAŞTIRMASI

A. Murat ÖZBAYOĞLU ve İstemihan Şakir KÖK (2011), yaptıkları çalışmada Göreceli Sinyal Güç İndeksi (RSSI) verileri kullanılarak yapay zeka tabanlı bir bina içinde sınırlı bir alanda kablosuz bağlantısı olan bir dizüstü bilgisayarın konumunu tespit etmeye çalışmışlardır. Yaptıkları yazılım için daha önceden bina içinde değişik yerlerden değerler almışlar ve bu değerleri yer tespiti yapmak istedikleri hareketli dizüstü bilgisayarın aldığı değerlerle karşılaştırmış anlık yerini tespit etmeye çalışmışlardır. Dizüstü bilgisayarın yerinin tespiti için Öklit mesafesi temellerine dayanan yöntemler denenmiştir. Bunların dışında yapay sinir ağlarının sistemdeki etkileri gözlenmiş ve dizüstü bilgisayarın yerinin tespiti hangi odada olduğuna kadar netleştirilmiştir.

Uğur BEKÇİBAŞI ve Mahmut TENRUH (2011), bildirilerinde daha önceden kullanılan kablosuz cihaz ağlarının konum belirleme yöntemlerini incelemişlerdir. Bu yöntemlerin incelenmesinin ardından kendilerinin önerdikleri Alınan Sinyal Gücü Göstergesi (RSSi) tabanlı dördüncü çapa düğümü kullanılarak tasarlanan bir yöntem ortaya konulmakta, artı ve eksileri incelenmektedir.

Faruk Baturalp GÜNAY ve Tuğrul ÇAVDAR (2014), çalışmalarında bir araç filosunda içlerinden ilerde olan araca sürücü yerleştirerek kalan araçların kontrolünü kablosuz ağlardaki veriler kullanılarak otonom hale getirmek için çalışmışlardır. Bazı araçlarda uydulardan konum belirlemek için alınan Küresel konumlama sistemi kullanılırken kalan araçların bulundukları yerler ise RSSİ, Geliş zamanı (TOA), Varış zaman farkı (TDOA) yöntemleri kullanılmıştır. MATLAB programı yardımı ile gerçekleştirilme çalışmalarında Küresel konumlama sistemine sahip araçların(çapa) sayısı, araçların mesafeleri gibi etkiler incelenmiştir. Küresel konumlama sistemi olan araçların doğru yerde ve yeterli sayıda bulunmaları yer tahminde yanlışlıkları azatlığını ortaya koymuştur.

Matthew LOY ve Iboun SYLLA (2005), hazırlanan Endüstriyel Bilimsel Tıbbi (ISM) bandı ve kısa mesafe cihaz antenleri adlı uygulama raporunda kısa mesafe cihazlar için antenlerin temelleri ve anten türleri tartışmış. Bu anten temellerini pratik tasarımlarla beraber sunmuştur.

Richard WALLACE (2010), tarafından Texas Instruments için hazırlanan Antenna Selection Guide adlı uygulama notu AN058'de kısa mesafe cihazlarda kullanılacak anten tercihlerinde göz önünde bulundurulacak önemli parametreleri sunmaktadır. Antenler için önemli parametreler, değişik anten tipleri, görünüm ve anten karakteri için teknikleri açıklamıştır. Radyasyon paterni, kazanç, empedans, bant genişliği, boyut, maliyet gibi bazı

parametleri bu dokümanda tartışmıştır. Ek olarak avantaj ve dezavantajları ile değişik anten tipleri sunulmuştur.

Richard WALLACE (2013), tarafından Texas Instruments için hazırlanan Monopole PCB Antenna with Single or Dual Band Option adlı tasarım notu DN024'de iki mod için configure edilebilen bir antenden bahseder. Bu anten tek frekansta çalışması modu olarak 868 MHz (Avrupa), 915MHz (Amerika Birleşik Devletleri) ve 920MHz (Japonya) ISM bandı ile çalışabilirken eğer çift frekans olarak konfigüre edilecek olur ise 868 MHz ile 2440MHz olarak kullanılabilir.

Nordic Semiconductor (2005), tarafından yayınlanan λ/4 printed monopole antenna for 868/915MHz adlı white paper'da FR4 malzeme üzerine tasarlanacak antenin geometrik özelliklerinin nasıl olması hakkında bilgi verilmiş ve ölçüler çıkarılmıştır. Ve sonuç olarak PCB üzerine yapılacak prototip antenin olması gerekenden daha uzun yapılarak denemelerde boyunun kısaltılması ve ölçüm yapılması tavsiye edilmiştir.

Bülent Bora SUYABATMAZ (2006), tezinde kablosuz iletişim için veri alışverişi yapabilen bir kart tasarlamış ve bir örnek uygulama eklemiştir. Tezinde kablosuz olarak bilgi alışverişi yapan sistemleri incelemiş ve nerelerde kullanıldığına dair bilgiler vermiştir. Kablosuz haberleşme uygulamalarına altyapı olması ve üzerinde geliştirilmeler yapılabilmesi için bir platform olması amacı ile bir kablosuz haberleşen modül tasarlamış ve uygulamaya dökmüştür. Platform kullanıcılara tasarlanan ara yüz sayesinde değişen şartlara göre modüle ait özellikleri değiştirme imkanı vermiştir. Sonuç olarak ise diğer kablosuz haberleşme sistemleri ile kıyaslamalar yapılmıştır.

Nusret BAŞÇİFCİ (2011), Zigbee altyapısını kullanarak, EKG verilerini bilgisayarda incelenebilmesi için kablosuz olarak iletilmesini tezine konu olarak almıştır. Tez kapsamında sistem EKG kemeri, EKG kemerinden elde edilen sinyalleri yükseltmek için enstrümantasyon yükselteç, bu gelen sinyallerin içinde kullanılmayacak sinyalleri filtreleyen filtreleme devresi ve sinyal bindirme devresi EKG algılayıcı devreyi oluşturur. Daha sonra elde edilen bu sinyaller devrenin çıkışında maksimum 1,5 volt olacak şekilde sınırlandırılır. Bu aşamada kullanılan mikrodenetleyici MSP430F2274 analog olan bu sinyali sayısallaştır. Bu sayısallaştırılmış sinyaller ZigBee protokolünü barındıran Texas Instruments firmasının üretmiş olduğu MSP430 RF2480 geliştirme kiti ile bilgisayar ortamına iletilir. Bilgisayar ortamına iletilen bu sinyaller Visual Studio 2010 C# programlama dili ile yazılan program ile grafikleştirilmiş ve nabız sesi ve nabız tespit özelliği konulmuştur.

Texas Instruments (2009), Ocak 2013 de revize edilen literatur numarası SLAU259E olan ve bu çalışmada da kullanılan CC430F6137 mikrodenetleyicisinin ait olduğu CC430

ailesine ait kullanma kılavuzudur. Bu kılavuzda işlemciye ait çevresel birimler ve modüllerin kullanımı hakkında bilgi verilmektedir.

Texas Instruments (2009), Ocak 2013 de revize edilen literatur numarası SLAS554H olan ve bu çalışmada da kullanılan CC430F6137 mikrodenetleyicisinin de elektriksel ve diğer parametrelerinin bulunduğu datasheet'tir

3. UHF BANDI VE ANTEN IŞIMA ÖRÜNTÜLERİ

3.1. UHF Bandı

Ultra High Frequency (UHF) (Ultra Yüksek Frekans) ITU (International Telecomunication Union) (Uluslararası Telekomünikasyon Birliği) tarafından belirlenmiş, frekans aralığı 300MHz ile 3GHz arasındaki radyo frekanslarıdır. Desimetre bandı veya desimetre dalga olarak da bilinen bu bandın dalga boyu 1 ile 10 desimetre arasında değişir. Bu bandın üst kesiminde süper yüksek frekans veya mikrodalga dalgaları alt kısmında ise çok yüksek frekans veya düşük frekanslar bulunur. UHF dalgaları büyük orandan görüş açısı ile yayılır. Bunlar tepeler, büyük binaların yayını engelleyecek kadar büyük duvarları tarafından engellenebilir. Bu bant televizyon yayınları, kablosuz telefonlar, walkie-talkies, uydu haberleşmesi ve sayısız radyo servisi için kullanılır. (Wikipedia, 2011)

Noktadan noktaya televizyon ve radyo sinyallerinin iletilmesi ve alınması birçok değişkenden etkilenebilir. Atmosferik nem, rüzgarlar, dağlar ve binalar gibi fiziksel engeller ve günün saati sinyalin iletim ve alımını etkiler. Tüm radyo frekansları atmosferdeki nem tarafından emilir. Uzun mesafelerde bu emilim ve radyo sinyalinin gücünü azaltır. Zayıflama ve bozulmanın etkisi frekansla artmaktadır. UHF TV sinyalleri genellikle nemden daha çok daha düşük bant VHF TV sinyallerinden bozulurlar. İyonosfer dünya katmalarından biri olup yüklü partiküllerle doludur ve bazı radyo sinyallerini yansıtır. Amatör radyo kullanıcıları dünya etrafında HF sinyalleri dünya etrafında yaymak için iyonosferin bu özelliğini kullanırlar ve hapsedilen dalgalar tekrar dünya yüzeyine yansıyıncaya kadar iyonosferin üst katmanlarında dolanır. UHF TV sinyalleri iyonosfer boyunca taşınamaz, fakat yüklü parçacıklardan yansıması ile görüş mesafesinden daha uzaklara taşınabilir. Bu atlama mesafesidir. Gün boyu ısınma ve soğumalar gibi troposferik olaylar UHF iletimini veya alımını güçlendirir veya azaltır.

UHF iletimin ana avantajı yüksek frekanstan dolayı kısa dalgada olmasıdır. İletim ve alım antenlerinin boyu radyo dalga boyu ile alakalıdır. UHF antenleri kısa ve küttürler. Daha kısa ve göze daha az batan antenler yüksek frekans bantları ile kullanılabilirler. (Ultra high frequency - Wikipedia, the free encyclopedia.htm) Bu özelliğinden ve ülkelerin bu bantta kısa mesafeli çalışan cihazların kullanımına lisans zorunluluğu getirmemesinden dolayı bu bantta pek çok cihaz tasarımı ve çalışmalar yapılmaktadır. Bunları örneklersek:

3.1.1. Endüstri alanındaki çalışmalar

2005 yılında Denizli'de III. Otomasyon sempozyumunda Ayhan ÖZDEMİR, İrfan YAZICI, Türker KUNDUZ mikrodenetleyici üzerine inşaa ettikleri sistem sayesinde kartların kablosuz olarak kontrol ve kumanda edilmesinin mümkün olduğunu sunmuşlardır. Tasarımlarının otomasyon sistemlerinin gereksimlerini karşılamak için gerekli geliştirmeler ve testler yapıldıktan sonra kullanılabileceğini, sistemin temelinde olan Analog Devices firmasının ADUC841 mikrodenetliyicisi ADC, DAC, RAM, EPROM, EEPROM, Timer, Counter, RTC gibi özellikleri barındırmaktadır. Kablosuz haberleşme için kullanılan Udea firması tarafından üretilen UTR C10M RF modülü sayesinde çift yönlü haberleşme sağlanmıştır. Bu yapılar üzerine inşa edilen iki ayrı cihaz sayesinde birbirine iletilen verileri yorumlayıp kontrol ve kumanda edilebildiği bildirmişlerdir. (A. ÖZDEMİR, İ. YAZICI, T. KUNDUZ, "Mikrodenetleyici tabanlı kablosuz kontrol ve kumanda sistemi tasarımı", III. Otomasyon Sempozyumu, Denizli, 2005.)

SYDMA firması ise fabrikalarda kullanılmak üzere ARM-SE+X8800 Serisi- Kablosuz Forklift Çağrı sistemini fabrikalarda kullanmak üzere sunmuş ve bu sistemi:

SYDMA Kontrol, fabrika içerisindeki iş istasyonları ile forkliftler arasında haberleşmeyi sağlamak amacı ile kendi yapılarına uygun sistem geliştirmişlerdir. Buradaki amaçları forkliftlere kablosuz olarak görülebilir ve işitilebilen sinyaller göndererek iş istasyonları ile iletişim sağlamaktır. Bu sistem sayesinde forklift operatörü çağrının hangi iş istasyonundan geldiğini bilerek o iş istasyonundaki görevini yerine getirmek için iş istasyonuna gitmektedir. İş istasyonlarında kablosuz çağrı noktası denilen iletişim birimi, çağrı için gerekli buton ve gösterge ışığından oluşan bir ünite bulunur. Bu üniteler pilli olarak da çalışabilmekte bu sayede yerleri kolaylıkla değiştirilebilir. Forklift üzerinde ise çağrıların hangi üniteden geldiğinin gösteren bir panel vardır. Ve bu panel sayesinde operatör ihtiyaç olan istasyona yönlendirilir.

3.1.2. Sağlık alanındaki çalışmalar

Batuhan YARIKKAŞ 2009 yılında çeşitli haberlere konu olan ameliyatlar sırasında hasta vücudunda unutulan cerrahi malzemeleri tespit edebilmesini konu alan yüksek lisans tezi sunmuştur.

Bu tezin amacı cerrahi operasyonlarda hasta bedeninde unutulan ameliyat malzemelerini tespit etmeye yarayacak radyo frekans haberleşmesine dayanan tasarımını yapmak, gerçekleştirmek ve performans testlerini yapmaktır. Bu kapsamda tedariği kolay olan

ISM bandlarından 433 MHz ve 868 MHz de çalışan veri iletişimi yapabilen modüller ve bunları kontrol eden mikrodenetleyici bulunan sistem oluşturulmuştur. Bu sistem gerçek ortamı simüle etmek için değişik radyo frekans geçirgenliğe sahip ortamlarda ve tavuk içinde denemeler yapılmıştır. Bu denemelerde elde edilen sonuçlar değerlendirilmiş ve 433 MHz modüllerin daha uygun olduğu belirlenmiş ve gerçek hayatta da kullanılabileceği ortaya konulmuştur.

Başka bir çalışma da 2006 yılında Ersan KABALCI insan vücudunda üretilen sinyallerden EKG sinyallerini 434 MHz ISM bandında bilgisayara göndermiş ve görüntülemiştir. Bu çalışmasında insan vücudundan elektrotlarla aldığı sinyalleri öncelikle işlenebilmesi için belirli bir yükseltme uygulamış daha sonra elde edilen sinyalleri yabancı sinyallerden arındırmıştır. Bu elde edilen sinyalleri dijital olarak göndermek için analog olan bu sinyali mikrodenetleyici yardımı ile dijitale çevirmiş ve seri olarak UART üzerinden 434Mhz frekansta çalışan haberleşme modülüne iletmiştir. Bu bilgiler alıcı modül tarafından alındıktan sonra mikrodenetleyici tarafından analog sinyale çevrilmiş ve bu sinyallerin kayıt birimde görüntülenmesi sağlanmıştır.

3.1.3.Uzaktan sayaç okuma sistemi

Uzaktan sayaç okumada da bazı bu bantta çalışan cihazlar hayatımıza girmiştir:

2007 yılında yürürlüğe giren Enerji Verimliliği Kanunu ve 2010 yılındaki yönetmelik ile 2000 m2 den büyük kullanım alanı olan binalarda;

“Merkezi ısıtma sistemine sahip binalarda, merkezi ve lokal ısı veya sıcaklık kontrol cihazları ile ısınma maliyetlerinin ısı kullanım miktarına bağlı olarak paylaşımını sağlayan sistemler kullanılır. Buna aykırı olarak hazırlanan projeler ilgili mercilerce onaylanmaz.”

“Tüketilen enerjiyi sınırlandırmak için merkezi ısıtma sistemi kullanılan binalarda TS EN 215’e uygun termostatik radyatör vanası kullanılır.”

“Merkezi ısıtma sistemlerinde toplam ısıtma giderlerinin %70’i bağımsız bölümlerin ölçülen ısı tüketimlerine göre paylaştırılır. Toplam ısıtma giderinin, %30’u ortak kullanım mahalleri, sistem kayıpları, asgari ısınma ve işletme giderlerinden kaynaklı ısı giderleri olarak bağımsız bölümlerin kullanım alanlarına göre paylaştırılır.”

Bu kanun ve yönetmeliklere göre merkezi sistem kullanılacak yeni binalarda ısı paylaşım cihazı (Kalorimetre) kullanılması zorunlu hale getirilmiştir. Bu kapsamda üretim alanı bu olan firmalar Isı Sayaçlarını üretmişlerdir. Ve bu sayaçlar üretilir iken ısınma maliyetlerin hesaplanması için sayaçların okunması ihtiyacını kablosuz iletişimin avantajlarını kullanmayı amaçlayan kablosuz sayaçlarda üretmişlerdir. Uzun ar-ge çalışmaları ve

belgelendirme süreçlerinden sonra Envotek Enerji Verimlilik Limited Şirketi geliştirdikleri Lifos RF Sistemini:

Bu sistem tüketim miktarını ve uzaktan erişim için radyo frekanslarını kullanır.

Firmanın kendisine özgün el terminali 868 MHz frekansında çalışmakta ve anlık olarak kalorimetredeki bilgileri okuyabilmektedir.

Ayrıca bu sorgulama bina dışından da yapılabilmektedir.

Kalorimetreler pil ile beslenmekte aşırı düşük pi tüketimi sayesinde uzun yıllar çalışabilmektedir.

Ayrıca cihaz IP54 koruma sınıfına uygun ve üzerindeki LCD sayesinde kullanıcı anlık olarak takip sağlayabilmektedir.

Ankara Büyükşehir Belediyesi Su ve Kanalizasyon İdaresi Genel Müdürlüğü'nde İç Denetim Başkanlığında görev alan Atilla Türkyılmaz'ın Türkiye Cumhuriyeti Bilim, Sanayi ve Teknoloji Bakanlığı Metroloji ve Standardizasyon Genel Müdürlüğü için hazırladığı Su Sayacı Tercihinde Dikkat Edilecek Hususlar başlıklı sunumunun Uzaktan Sayaç Okuma Sistemi bölümünde, bu amaç için üretilen ve kullanıma sunulan ekipmanlar ile geliştirilebilecek bir sistemin genel çalışma prensibi, avantajlarını, dezavantajlarını ortaya koymuştur (Türkyılmaz, 2010). Bu sunumda:

Sayaçların bulundukları yere gitmeden, kablosuz iletişim imkanları kullanılarak uzaktan okunmasıdır. Mobil sistem ve merkezi sistem olmak üzere iki farklı şekilde okuma işlemi yapılabilmektedir.

Şekil 3.1. Uzaktan okuma sistemi

"Uzaktan Sayaç Okuma Sistemi" ile aşağıda sayılan avantajlar sağlanır:

- Kullanımı daha kolay ve daha hızlıdır.

- Düşük maliyetli olup olmadığı, idarelerin kullandığı sisteme göre farklılık gösterebilir.
- Adam-saat tasarrufu sağlar.
- Okunan bilgiler otomatik olarak kayıt altına alındığından geçmişe dönük bilgilere daha hızlı ve kolay kontrol edilir.
- Kayıt altında bulunan bilgiler sayesinde anlık tüketim haricinde günlük, haftalık gibi tüketim miktarları takip edilebilir.
- Ölçüm sonuçlarının girilmesinde insan faktörü olmadığı için okuma hataları ortadan kalkar.
- Su tüketimin izlenmesi ve kontrol altına alınması sağlanır.
- Anlık olarak erişim olmasından dolayı şebeke problemleri anında tespit edilebilir ve müdahale gibi özelliklerinin yanında, yasadışı işlemleri önleyerek gelir artışı sağlar.
- İnternet tabanlı olduğu için her yerden sisteme girilip takip edilebilir.
- Sayaç otomatik olarak cep telefonuna fatura miktarını gönderebilir.
- Abonenin de internet üzerinden erişimine izin verilerek abonelik bilgileri, su tüketimi, geçmiş borçlar gibi bilgileri görmesine imkan sağlar.
- Toplanan verilerden hangi sayacın hangi tüketim profiline göre çalıştığı saptanır.
- İdarenin kullanımda olan sayacı ne zaman değiştireceği karar verme zamanının daha doğruya yakın olarak belirlenmesine fayda sağlar.
- Sistem ön ödemeli veya klasik ödemesiz sayaçlarda kullanılabilir.
- “Uzaktan Sayaç Okuma Sistemi’nin Dezavantajları
- Su faturalarının daha sonra dağıtımı yapılıyor.
- Frekans yayılımını etkileyen ortamlarda okuma mesafesi kısalıyor.
- Bilgisayar korsanlarının hedefi olabilir. Öyle ki bilgisayar korsanları su faturalarına müdahale ederek düşürebilir. Öte yandan su kullanım bilgisi verebilen bu sayaçlar, evleri “Biri Bizi Gözetliyor” haline de getirebilir

3.2. Anten Işıma Örüntüsü

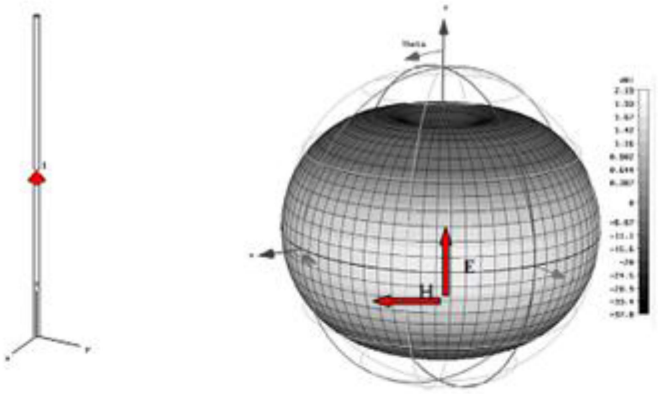
Işıma örüntüsü antenin en temel tanımlayıcılarından birisidir ve bir uygulama için seçilecek antenin ışıma örüntüsü belirleyicidir. Örnek olarak hücresel haberleşmede, kullanılan antenlerin ışıma örüntüsünün yönsüz (omnidirectional) olmasının istenmesinin sebebi kullanıcıların yeri bilinmemesidir. Mobil kullanıcıyla, antenden yayılan gücün kullanıcının etrafında her yöne eşit yayılması istenir ki kaliteli bir haberleşme gerçekleşsin.

Uydu haberleşmesinde ise yüksek derecede yönelimli antenler kullanılır çünkü bu uygulamada yayılan gücün, bilinen belli bir yöne yayılması istenir. IEEE anten terimleri standardına göre, antenin ışıma örüntüsü (veya anten örüntüsü) aşağıdaki gibi tanımlanır: "Uzay koordinatlarının bir fonksiyonu olarak antenin ışıma özelliğinin matematiksel veya grafiksel bir gösterimidir. Işıma örüntüsü, birçok durumda uzak alan bölgesinde tanımlanır ve yönlü koordinatların bir fonksiyonu olarak gösterilir. Işıma özellikleri, güç akış yoğunluğu, ışıma yoğunluğu, alan gücü ve polarizasyonu içerir."

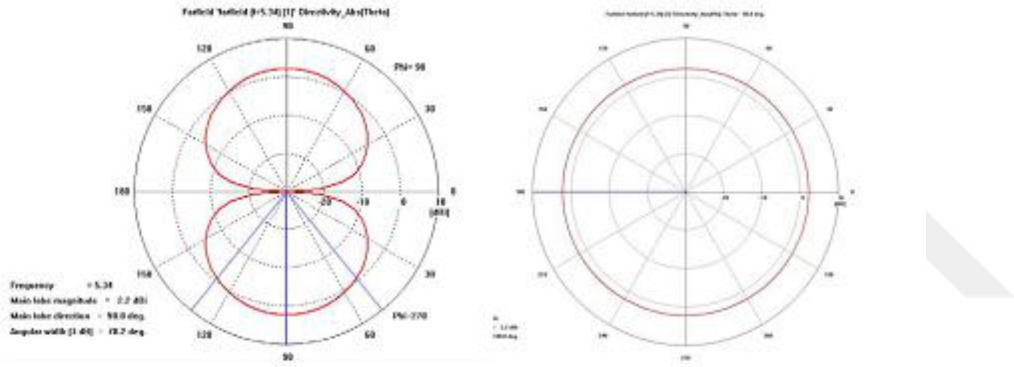
Anteni saran bir küre yüzeyi boyunca antenden yayılan bağıl ışıma gücünün, küresel koordinatlarda üç boyutlu gösterimine üç boyutlu ışıma örüntüsü diyoruz. Küresel koordinat sisteminde x-z düzlemi (θ değişken ve $\phi = 0°$) genellikle düşey düzlem (elevasyon düzlemi) olarak, x-y düzlemi (ϕ değişken ve $\theta = 90°$) ise yatay düzlem (azimut düzlemi) olarak adlandırılır.

Genellikle düşey düzlem, elektrik alan vektörünü (E-düzlemi) ve maksimum ışıma doğrultusunu, yatay düzlem ise manyetik alan vektörünü (H- düzlemi) ve maksimum ışıma doğrultusunu gösterir. Şekil 3.3'de, θ veya ϕ'den biri değişkenken, diğeri sabit bir değer aldığı durumdaki iki boyutlu ışıma örüntüsü çizilmiştir. Şekil 3.2'de yarım dalga dipolü ve bunun üç boyutlu ışıma örüntüsü gösterilmiştir. Kazanç dBi cinsinden ifade edilmiştir. Bu izotropik bir antene göre kazanç demektir. Şekil 3.3'te $\phi = 0°$ iken θ değişken ve $\theta = 90°$ iken ϕ değişken durumları için iki boyutlu ışıma örüntüleri verilmiştir. Şekil 3.2'de çok açık bir şekilde görüldüğü gibi, maksimum ışıma gücü $\theta = 90°$ olan düzlem boyunca olmaktadır. Işımanın kör noktaları ise ışıma örüntüsünden görüldüğü gibi, z ekseni boyunca uzanan dipolün sonlarında (veya $\theta = 0°$ ve $180°$ iken) yer almaktadır. Bunların doğruluğunu iki boyutlu çizimlerden de görmek mümkündür.

Şekil 3.3'de antenin ışıma örüntüsü, yatay düzlem sabit tutulup, düşey düzlem (θ) hareketli (sol), ve düşey düzlem sabit tutulup (ışımanın maksimum olduğu $\theta = 90°$ doğrultusunda) yatay düzlem hareketli olduğu (sağ) durumdaki polar koordinattaki gösterimdir. Görüldüğü gibi üç boyutlu ışıma örüntüsü ile arasında bir fark gözükmez.



Şekil 3.2. Dipol anten benzetimi ve CST Microwave Studio"da çıkarılan 3 boyutlu ışıma örüntüsü



Şekil 3.3. Yarım dalga dipolünün iki boyutlu ışıma örüntüsü çizimi: θ değişken Φ = 0° (sol) ve ϕ değişken θ = 90°(sağ)

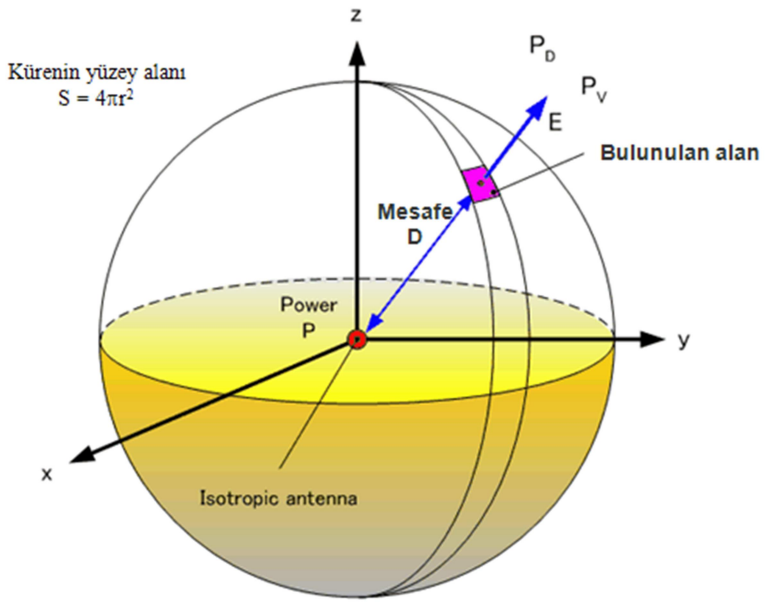
Üç boyutlu ışıma örüntüsünü, birçok iki boyutlu ışıma örüntülerinin toplamıyla gösterilebilirken, en önemli iki tanesi E-düzlemi ve H-düzlemi örüntüleridir. E-düzlemi elektrik alan vektörünü ve maksimum ışımanın yönünü, H-düzlemi ise manyetik alan vektörünü ve maksimum ışıma yönünü gösterir. Şekil 3.2'de basit olarak üç boyutlu ışıma örüntülerinden iki adet "kesimi" gösterir ve üç boyutlu örüntü bu iki boyutlu gösterimlerden anlaşılabilir.

Şekil 3.2 ve Şekil 3.3'deki örüntüler ve gösterimler yarım dalga dipolünün ışıma karakteristiğini gösterir ve sanal olarak oluşturulan yönsüz bir antendir. Yönsüz bir anten, ancak teoride olan izotropik antendir. IEEE anten terimleri standardı, izotropik bir anteni şu şekilde tanımlar: "Kayıpsız ve her yöne eşit ışıma yapan teorik bir anten". Gerçek bir yönsüz kaynak ışıma örüntüsünde, kör nokta bulundurmaz ve yönelimliliği 0dB olur. Fakat doğada

izotropik kaynak bulunmadığından dolayı, yönelimli bir anten yukarıda görülen dipolden daha fazla yönelimli bir anten olacaktır (Elektroport, 2019). Çeşitli antenler ve ışıma örüntüleri,

3.2.1. İzotropik Antenler (Isotropic Antennas)

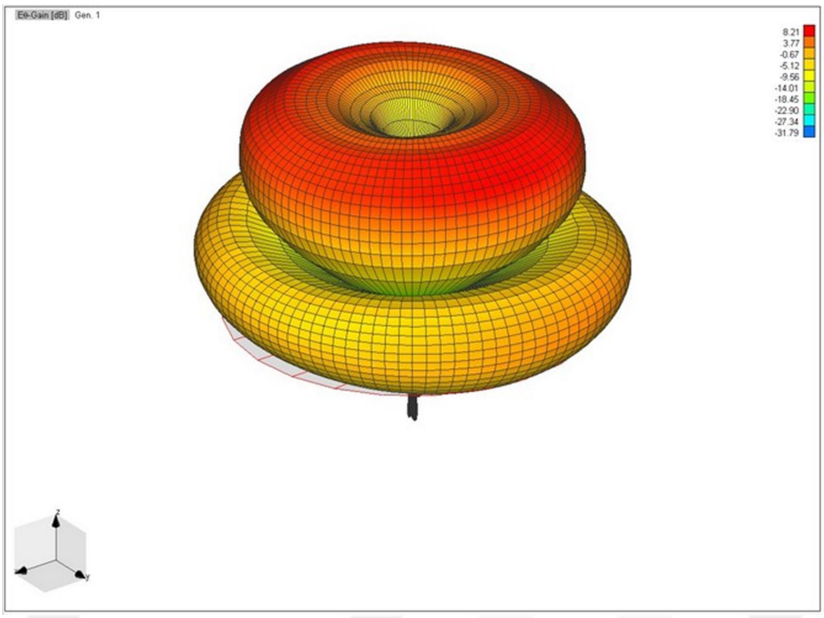
Boşlukta veya uzayda her yöne, eşit güçte elektromanyetik dalga yayan ve teorik bir nokta kaynağı olan izotropik antenler, anten kazançlarının anlaşılmasında referans olarak kullanılır. Aşağıda izotropik antenin küre biçiminde olan ışıma görüntüsü mevcuttur (Elektroport, 2019).



Şekil 3.4.İzotropik antenin küre biçiminde olan ışıma görüntüsü

3.2.2. Monopol Antenler (Monopole Antennas)

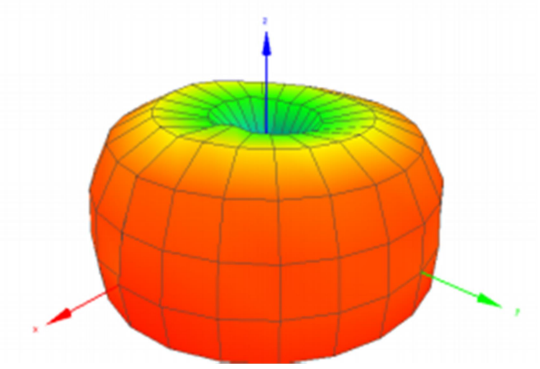
Toprak levhası adı verilen bir iletken plaka üzerine genellikle dik olarak ve iletken plaka ile elektriksel temas ettirilmeden yerleştirilen çeyrek dalga boyunda (λ/4) düz bir metal çubuktan oluşur. Çeyrek-dalga anteni (quarter wave antenna) ya da morkoni anteni (marconi antenna; 1895'de Guglielmo Marconi tarafından üretilmiştir.) olarak da bilinmektedir.



Şekil 3.5. Monopol antenin üç boyutlu ışıma görüntüsü

3.2.3. λ/2 Dipol anten

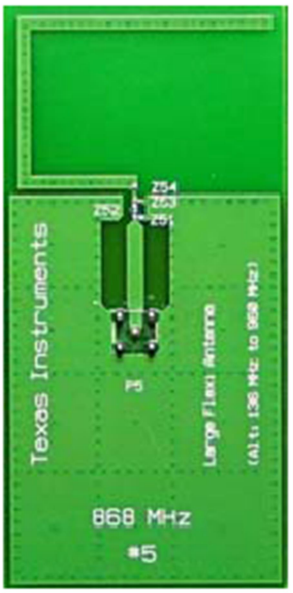
Dipol antenler genellikle λ/2 yarım dalga boyu olarak kullanılırlar. Işıma örüntüsü aşağıdaki şekildeki gibi çubuklar boyunca yayılır ve çubukların uç noktalarında azalır

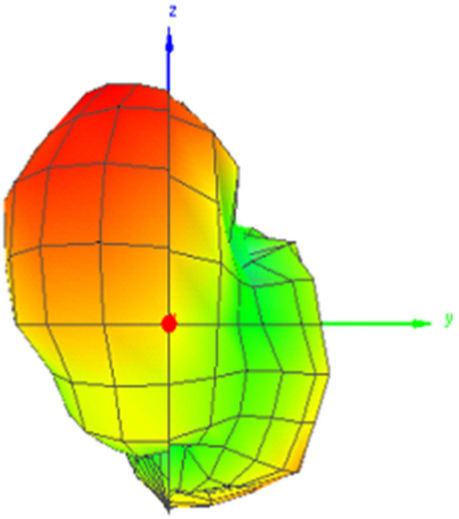


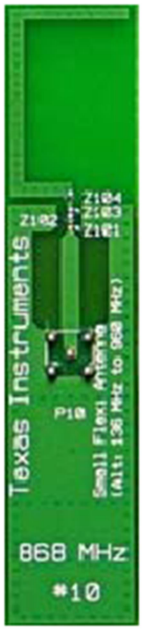
Şekil 3.6. Monopol antenin üç boyutlu ışıma görüntüsü

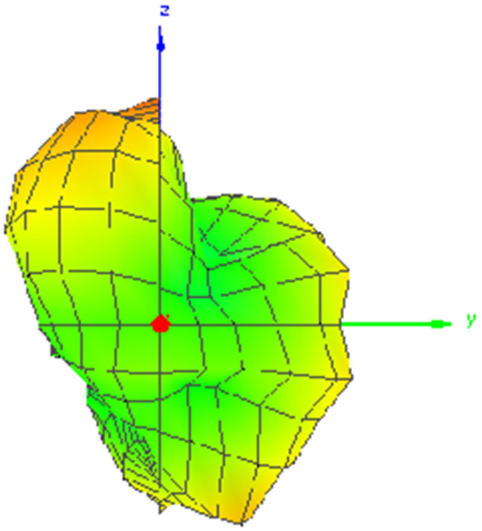
3.2.4. PCB Antenler

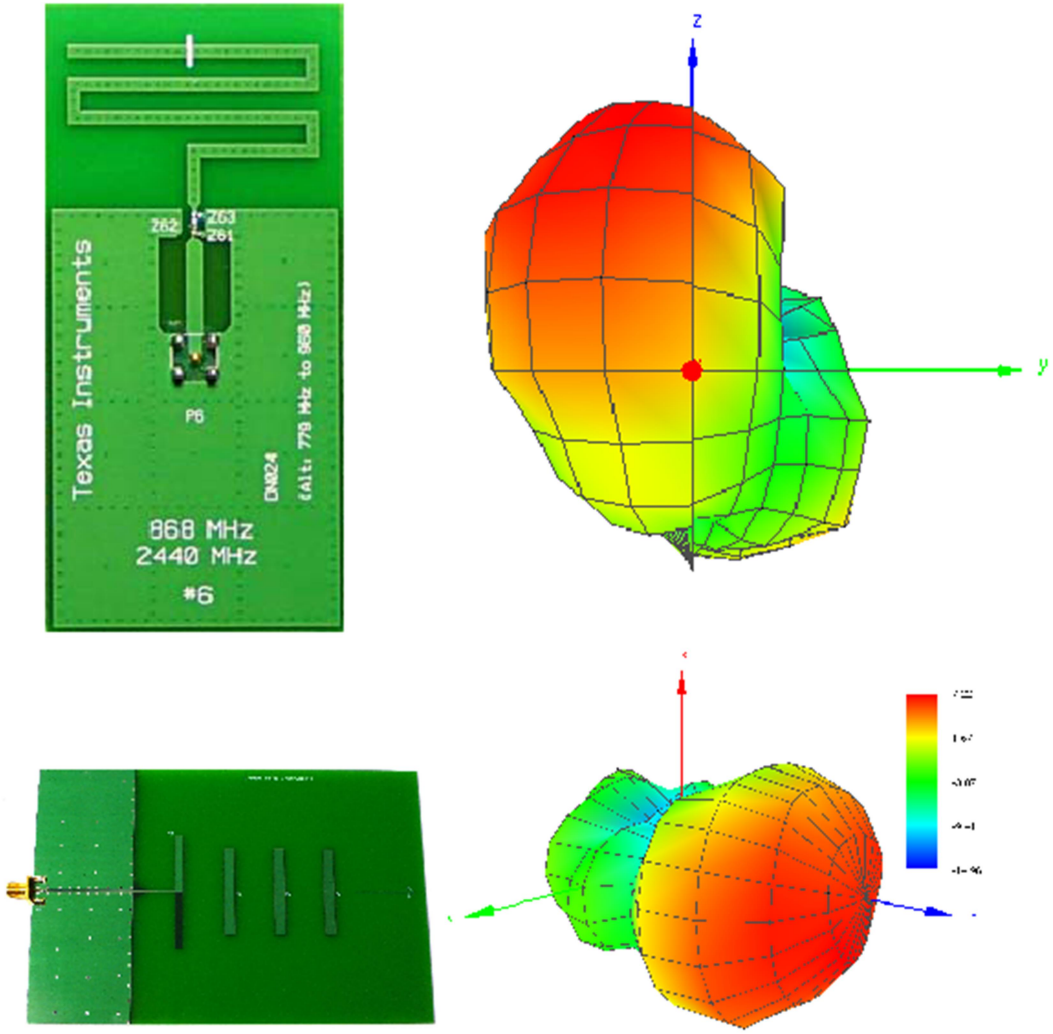
Daha önceden TEXAS INSTRUMENTS firması tarafından tasarlanmış ve ışıma örüntüleri oluşturulmuş PCB antenlere örnekler aşağıdaki gibidir (Wallace, R., 2010).


Texas Instruments
868 MHz
#5


z
y


Texas Instruments
868 MHz
#10


z
y



Şekil 3.7. Bazı Pcb antenlerin üç boyutlu ışıma görüntüsü

4. PATLAYICI ORTAMLAR VE KULLANILACAK ELEKTRİKLİ ALETLERİN ÖZELLİKLERİ

Birçok sanayi kolları ve günlük yaşantımızda karşımıza çıkan patlayıcı ortamlar genellikle petrol ve ürünleri, kimyasallar, günümüzde evlerimizde ve sanayinin hemen hemen her yerinde karşımıza çıkan doğalgaz bulunduğu ortamlardır. Bu ortamlarda çalışan elektrikli aletlerin anlık oluşturdukları arklar, statik birikmeler, statik ısınmalar bu ortamları tehlikeli hale getirmekte ve patlama koşullarının oluşmasını sağlamaktadır. Bu sebeple bu aletlerin kullanımı ve imalatı bazı kurallara göre düzenlenmiştir.

Ülkemizde bu ortamlar patlayıcı ortam olarak tanımlanmış ve bu ortamlarda kullanılan aletler ise İngilizceden direk geçerek exproof olarak adlandırılmıştır (Sarı K., 2019).

4.1. Patlayıcı Ortamların Tarihçesi

Madencilik bilinen en eski patlayıcı ortamdaki sanayi olarak karşımıza çıkmakta ve kömür ocakları ilk patlayıcı ortam ile karşılaşma yaşandığı yerdir. Bundan dolayı madencilik tedbirlerin alındığı ilk sanayi dalıdır ve bu disiplinin öncülüğü madencilik yapmıştır.

Maden sanayi ne kadar eski olsa da madencilikte elektrik kullanımı 1900 yıllarında başlamıştır. 19. yüzyılın sonları ve 20. yüzyılın başlarında kömür ve petrol ihtiyacının artması madencilik sektöründe arayışa sebep olmuş ve diğer sanayi kollarında kullanılan elektrik enerjisin buralarda kullanılmasına itmiştir. Elektrik madenlerle tanışması 20. yüzyıla denk gelmiştir. Madencilikteki tecrübeler daha önceki yaşanmış grizu olayları elektriğin direk olarak madenlere girmesini engellemiştir. Bu sayede çalışmalar yapılmış tehlike oluşturup oluşturmayacağı ve bu tehlikelere karşı alınacak tedbirler maden ocağına elektriğin girmesinden önce yapılmış madencilik sektörü sıkıntılar yaşandıktan sonra önlem alma zorunda kalmamıştır. Özellikle grizulu madenlere özel denemeler yapılmış ve bu ortamlara benzer denemeler yapılmıştır. Ve bu deneme ortamları kolaylaştırılmış ve standart hale getirilmiştir. Ve ülkemizde standartları düzenleme ile ilgili kuruluş olan TSE'nin konu ile ilgili standartları mevcuttur (Sarı K., 2019).

Şekil 4.1. Davi emniyet lambası

19. yüzyılın başlarından itibaren buharlı gemilerin artması ile artan kömür ihtiyacı ve daha yüksek kalorili kömür ihtiyacı madenciliği yer altı taş kömürü çıkarmaya itmiştir. Ve o yıllarda aydınlatma için kullanılan açık alevli petrol lambaları metan gazı ile patlamaya sebep vermekte idi. Madenin daha derinlere gitmesi ile grizu artışı olmakta ve patlama kaçınılamaz hale gelmekte idi. Bu duruma tedbir olarak kimyager Sir Davy 1815 yılında "Davi Emniyet Lambası" nı geliştirmiştir. Yine petrol ile çalışan bu lamba geliştirilen ızgara sistemi ile ateşi dışarı vermemekte ve alev uzayınca ortamdaki grizu miktarının arttığını anlayan madencilere kaçma imkanı vermiştir. Ve halen günümüzde de kullanımı devam etmektedir.

Kurşun oksit ile çalışan 1925'lerde kullanılmaya başlayan baş lambaları artık petrol lambası tehlikelerinden madencileri kurtarmış ve %95 oranında kazaları azaltmıştır. Günümüzde ise kadmiyum nikel hafif aküler kullanılmaya başlamış ve aydınlatma kaynaklı patlamalar nerede ise sona ermiştir (Sarı K., 2019).

Şekil 4.2. Kurşun oksit baş lambası

1912'den sonra madenlerde elektrik aletlerin kullanımı başlamış (Transformatör, elektrik motoru, şalt cihazları v.b.) ve günümüzdeki tarzda önlemler alınmaya çalışılmış ve günümüzde kullanılan d-tipi koruma (alev sızmaz) geliştirilmiştir. Dış camı kırılması zor olacak kadar kalın ve ızgaraları bulunan akkor Flamanlı aydınlatma armatür lamba, asekron motorların ise tam kapalı halde kullanılmasına izin verilmiştir. İlk kez 1926 yılında İngiltere'de "BS229: Flameproff enclosures" adı ile yayınlanmıştır. Daha sonra 1935 ve 1943 yıllarında ise Almanya standartları yayınlamıştır. Uluslararası birliklerin olmayışı her ülkeyi kendi standartlarını oluşturmaya itmiş ve birbirinden bağımsız laboratuvarlarda bu denemeler ve geliştirmeler yapılmıştır. Madenlerde edinilen bu tecrübeler sayesinde diğer sanayi kolları başlangıç yaşamamış ve bu birikimleri kullanmışlardır (Sarı K., 2019).

4.2. Patlayıcı Ortamın Tanımı

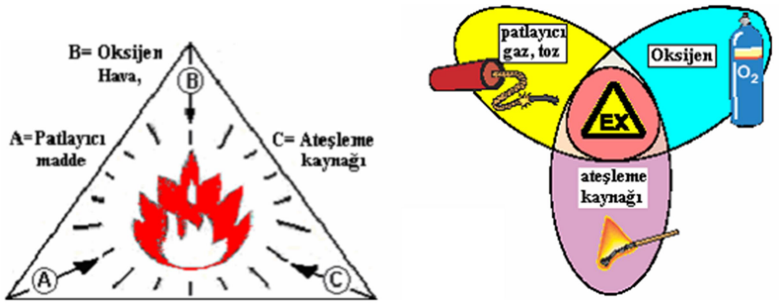
"Patlayıcı, parlayıcı ve yanıcı nitelikteki gaz, toz veya buharın hava ile karışarak patlayıcı kıvama geldikleri yerlere patlayıcı ortam denir." Patlayıcı ortamın oluşması için üç ana bileşen gerekir. Bunlar:

A. Patlayıcı maddeler; Patlayıcı, parlayıcı ve yanıcı gaz, buhar veya toz

B: Oksijen

C: Patlamayı tetikleyecek enerji.

Bu bileşenlerden herhangi biri ortamda bulunmaz ise artık patlama riski kalmayacaktır. Aşağıdaki resimler patlama üçgeni olarak adlandırılır.



Şekil 4.3. Patlama üçgeni

4.3. Patlama Üçgeni Elemanları

4.3.1. Patlayıcı, parlayıcı ve yanıcı gaz, toz ve buharlar

4.3.1.1. Gazlar

Bu gazların başında doğalgaz, sanayide kullanılan hidrojen ve asetilen, günlük hayatımızda araçlar ve evlerde kullanılan sıkıştırılmış petrol gazları yer alır. Bu gazlara ait özellikleri:

Çizelge 4.1. Bazı gazlar ve özellikleri

Bazı Gazlar ve Özellikleri						
Gaz	Hava ile Karışım		Saf Oksijen ile Karışım		Patlama Isısı	Min. Patlama Enerjisi
	LEL	UEL	LEL	UEL		
METAN	4,4-5	15-16,5	4,8	60,0	595,0	280,0
PROPAN	1,7-2,1	10-10,9	2,0	60,0	470,0	250,0
BÜTAN	1,4-1,8	9,3-10,6	1,8	57,5	365,0	
HEXAN	1,0	8,1			265,0	
NONAN	0,7	5,6			205,0	
ETİLEN	2,3-2,9	32,4-33,5	3,0	81,5	425,0	
BENZOL	1,2	8,0			555,0	200,0
HİDROJEN	4,0	77,0	4,0	95,0	560,0	11,0
ASETİLEN	1,5	78,0			305,0	19,0

Bu gazlar yeterli enerji ve hava ile birleşip patlama üçgenini oluşturduklarında patlayabilirler. Patlama hava ile karışım oranına bağlıdır. Bu karışımın alt ve üst sınırları tabloda verilmiştir. Bunlarda alt patlama sınırları LEL (lower explosive limit) üst sınır UEL(upper explosive limit) olarak belirtilmiştir. Patlayıcıların büyük kısmı karbon-hidratlar

oluşturur. Bu bileşimlerdeki zincirler maddenin hallerinin oluşumunda etkilidir. Aşağıdaki tabloda bazı patlayıcı maddelerin halleri ve formülleri verilmiştir (Sarı K., 2019).

Çizelge 4.2. Bazı gazlar ve formülleri

Formül	**İsim**	**Hal**	**Formül**	**İsim**	**Hal**
CH_4	METAN	GAZ	C_6H_{14}	HEKSAN	SIVI
C_2H_6	ETAN	GAZ	C_7H_{16}	HEPTAN	SIVI
C_3H_8	PROPAN	GAZ	C_8H_{18}	OCTAN	SIVI
C_4H_{10}	BUTAN	GAZ	C_9H_{20}	NONAN	SIVI
C_5H_{12}	PETAN	GAZ	$C_{10}H_{22}$	DECAN	SIVI

4.3.1.2. Sıvılar

Başta petrol ürünleri (benzin, fuel oil, tiner v.b.) olan sıvılar "yanıcı parlayıcı ve patlayıcı" olarak adlandırılır. Değişen sıcaklıklarla buharlaşan bu sıvılar hava, buhar karışımı patlayıcı ortam oluştururlar ve bu buharı oluşturan en düşük sıcaklığa "parlama noktası (Flash Point)" olarak adlandırılır. Daha önceden gazlarda bahsedilen LEL değeri gibi tedbirler için önemli olup tehlikenin boyutunu belirler. TS 12820 sınıflandırması tehlike sınıflandırmasını göstermektedir.

Çizelge 4.3. Yanıcı sıvıların tehlike sınıfları

Yanıcı Sıvıların Tehlike Sınıfları		
Sınıf	**Parlama Noktası**	**Kaynama Noktası**
IA(PARLAYICI)	$T_f<22,8^OC$	$T_b<37,8^OC$
IB(PARLAYICI)	$T_f<22,8^OC$	$T_b<37,8^OC$
IC(PARLAYICI)	$T_f>22,8^OC$	$T_b<37,8^OC$
II(YANICI)	$37.8^OC<T_f<60^OC$	II
IIIA(YANICI)	$60^OC<T_f<93^OC$	IIIA
IIIB(YANICI)	$T_f>93^OC$	IIIB

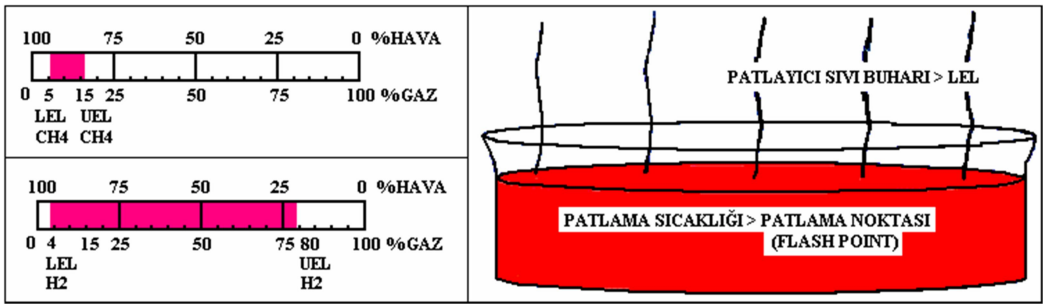
Çizelge 4.4. Parlayıcılık sınıflarındaki sıvılar

IA (PARLAYICI)	Dietil eter,etilen oksit, bazı hafif ham petroller
IB (PARLAYICI)	Araba ve uçak benzinleri, toulen, lakuer, lakuer tiner
IC (PARLAYICI)	Kısilen, bazlı boyalar, solvent tabanlı bazı çimentolar
II (YANICI)	Mazot (disel yakıtı), boya, tiner
IIIA (YANICI)	Evlerde kullanılan yakıtlar, fuel oil, kalorifer yakıtı
IIIB (YANICI)	Yemeklik yağlar, yağlama yağları, motor yağları

Çizelge 4.4. Bazı parlayıcı ve yanıcı sıvılar ve özellikleri

TS12820'ye göre bazı parlayıcı ve yanıcı sıvılar ve özellikleri				
	Patlama noktası	**Sınıf**	**Kaynama noktası**	**Havada asgari tutuşma sıcaklığı**
	^{O}C		^{O}C	^{O}C
Benzin	-40 ile -46	IB	38 ile 204	441
Dizel yakıt	>55	II		
Gaz yağı	>38	II	151 ile 301	227
Antifiriz	110	IIIB	149	
Fren sıvısı	149	IIIB	282	
Şase gresi	204	IIIB	>427	>427
Dişli yağı	202	IIIB	>427	>427
Lityum-moli gres	193	IIIB	>427	>482
Yağlama yağları	149-232	IIIB		
Hidrolik direksiyon sıvısı	177	IIIB	>288	
Beyaz gres	241	IIIB	>427	>427
Cam yıkama sıvısı menatol/su karışımları				
%100 menatol	12	IB	64	385
%50 menatol	27	IB		
%20 menatol	48	IB		
%5 menatol	97	IB		

Gazlar için LEL ve sıvılar için parlama noktası (flash point) aşağıda sembolize edilmiştir.



Şekil 4.4. Sıvılar için parlama noktası

4.3.1.3. Katı maddeler ve tozlar

Büyüklükleri 500 mikrometreden küçük olan ve havada belirli bir süre durabilen maddelere toz adı verilir. Tozlar patlayıcı ortam oluşturmak için bileşenlerden oksijen ile karışımı ya toz bulutu veya ince bir tabaka olacak şekilde olmaktadır. Herhangi bir ısı kaynağı ile küçük bir bölüm toz akkorlaşır ve patlama oluşturur. Bu patlama ile diğer toz zerrecikleri havaya karışarak toz bulutu haline gelir. Bulutun patlaması daha büyük toz bulutları oluşturur ve tekrar diğer toz zerrelerini tetikler ve bu şekilde zincirleme bir patlama oluşur. Tozların patlayıcılığını belirleyen özellikleri:

A. Çekirdek büyüklüğü (M)

B. Patlama için gerekli minimum enerji (MEE)

C. Azami patlama basıncı (EP) ve

D. Patlama şiddeti Kst, hava toz karışımının kapalı bir kapta ürettiği azami basınç değişimidir.

Tozlar için en önemli “tehlike ve tahribat belirleyici değer Kst’ dir. Bazı tozlar ve statik patlama sıcaklıkları aşağıda tablo halinde verilmiştir.

Çizelge 4.5. Patlayıcı Tozlar ve Özellikleri

Patlayıcı Tozlar ve Özellikleri					
Toz Cinsi	**Patlama Isısı**		**Toz Cinsi**	**Patlama Isısı**	
	Bulut	**5mm Film**		**Bulut**	**5mm film**
Alüminyum	560°C	>450°C	Polietilen tozu	440°C	melts
Odun kömürü	520°C	320°C	PVC tozu	700°C	>450°C
Linyit kömürü	380°C	225°C	Şeker tozu	490°C	460°C
Kakao	590°C	250°C	Kurum, is	810°C	570°C
Kahve	580°C	290°C	Nişasta	460°C	435°C
Hububat, mısır	530°C	460°C	Toner	520°C	melts
Methyl cellulose	420°C	320°C	Buğday	510°C	300°C
Kağıt lifi, kırpıntısı	570°C	335°C	Reçine	530°C	>450°C

Çizelge 4.6. Patlayıcı Tozlar ve bazı özellikleri

Patlayıcı tozlar ve özellikleri				
Toz cinsi	**M(µm)**	**Pmax(bar)**	**Kst(bar ms-1)**	**MEE (mJ)**
Aktif kömür	bis 10	7,3	72	500000
Yeşil mercimek unu	27	9,1	109	100
Çamur çöktürücü (76% organik bileşenli)	89	7,5	71	50
Mısır nişastası	10	9	200	10
Paraformaldehyd Polyoxymethylen	19	9,6	405	5
Bal mumu, parafin	<20	8,4	185	5
Polyester, poliester	35	7,8	140	5
Selüloz astat	31	7,5	116	5
İstifleme, örtü tozu (60 % anorganik bileşenli)	40	5,6	90	5
Epoxidharz, epoksi reçine	27	7,5	161	1
Polyurethan, poliüretan	29	7,8	150	1
Alüminyum	<10	11,4	625	0,1

4.3.2. Ateşleme Kaynakları

Patlama üçgenin bileşenlerinden biri ise enerji kaynağıdır. Oksijen ile buluşan patlayıcı madde artık patlamak için gerekli enerjiyi bekler. Genellikle elektrik arkları, sıcaklık, statik birikimler ve bunların boşalmalarıdır. Başlıca enerji kaynakları:

4.3.2.1. Elektrik ark ve kıvılcımı:

Şalter, sigorta v.b. gibi açılıp kapanırken ark çıkaran cihazların açılıp kapanmasında, statik yüke sahip olan materyallerin ani yük boşalmalarında, patlayıcı ortamdaki kabloların kopması veya kısa devre hale geçmesi durumunda çıkan ark veya kıvılcımlar patlayıcı ortama enerji vererek patlamanın başlamasına sebep olabilirler. Bu sebeple elektrikli aletlerin veya statik birikim yapabilecek materyallerin çok iyi bir şekilde bu ortamlara enerji vermeleri engellenmelidirler (Sarı K., 2019).

4.3.2.2. Sıcak yüzeyler (statik ısı ile patlama)

Herhangi bir sebepten oluşan ısı patlayıcı ortama gereken enerjiyi sağlayabilir ve patlamaya sebep olabilir.

4.3.2.3. Mekanik sürtünme ile çıkan kıvılcım

Bu durum öngörülebilir ve öngörülemez olarak iki şekilde incelenebilir. Kıvılcım çıkacağı kesin olan aletler burada gerekli önlemler alınarak kıvılcım çıkarılması engellenebilir ve öngörülebilir olarak niteleyebiliriz. Fakat beklenmedik durumlarda oluşabilecek kazalar mesela bir metal levhanın başka bir metale çarpıp kıvılcım çıkarması gibi öngörülemeyen durumlarda oluşabilir. Bu durumda patlamanın engellenmesi için diğer bileşenler ortamdan çıkarılmalıdır.

4.3.2.4. Her nevi statik elektriklenme:

Hemen hemen her yerde karşımıza çıkabilecek ve ciddi sorun yaratabilecek tehlikelerden biridir. Bu sebeple patlayıcı ortam olan yerlerde statik biriktirmeyen ekipman ve cihazlar kullanılmaktadır.

Çok sıklıkla görülmese de aşağıdaki bazı patlama kaynaklarına da yer verilmiştir.

4.3.2.5. Açık alev sıcak gaz ve akkor haldeki parçacıklar (hot particles)

Kaynak sıranda etrafa yayılan kor halindeki parçacıklar patlayıcı ortamın enerjisini sağlayabilir. Yanıcı tozlarda patlama sebeplerinden biri olarak karşımıza çıkmaktadır.

4.3.2.6. Adiyabatik basınç, şok dalgası:

Tüp şeklinde olup, düşük basınçta çalışan aletler patlama kaynağı teşkil edebilirler. Örneğin floresan tüpleri, kırıldıklarında tehlikeli olabilmektedirler. Yalnız bu olay tüpün kırılış şekline bağlıdır. Adiyabatik basınç sıkışması olabilmesi için tüpün ortadan değil ucundan kırılmış olması gerekir (Sarı K., 2019).

4.3.2.7. Yıldırım düşmesi ve elektrikli hava şartları:

Çok yüksek enerjiye sahip yıldırımlar patlayıcı ortama enerji kaynağı olmaktan öte her türlü yangın ve patlamalara da sebep olabilmektedir. Elektrikli hava şartlarından korunmak için ise metal silo gibi yapıların yeryüzü ile çok iyi şekilde topraklamasının yapılması gerekir. Bu yapılmaz ise bulutlardaki statik elektrik silolarda elektrostatik yüklemeye sebep olur.

4.4. Patlamaya Karşı Alınan Önlemler

"Patlayıcı, parlayıcı ve yanıcı gaz, toz ve buhar" bulunduran ortamlarda çalışma zorunluluğu oldu durumlarda öncelikle alınması gereken önlem patlama üçgeninin oluşumun engellemektir. Bu konuda "iş güvenliği ve işçi sağlığı" kuralları ciddi önlemler koymaktadır. Patlayıcı ortamı engellemek için alınacak ilk tedbirler birincil tedbirlerdir.

4.4.1. Birincil (Primer) Önlemler

Bu önlemlerin temelinde patlama üçgenindeki oksijen ve patlayıcı maddeyi ortamdan uzaklaştırmak yatar. Bu ortamlarda kullanılacak malzemenin korumalı olmasından öte diğer koşulları engellemek birincil önlemlere girmektedir. Ana başlıklar altında alınacak önlemler aşağıdaki gibidir.

1. Enerji kaynağını patlayıcı ortadan uzak tutmak, en çok kullanılan yöntemdir. Özellikle petrol ve kimya sanayide tercih edilen başlıca yöntemlerdendir. Örneğin akaryakıt

istasyonlarında şalterler ve trafo odaları güvenli bölgelerde yer almakta patlama riski oluşması engellenmektedir. Ve enerji kaynağı patlayıcı ortamın dışında bırakılmıştır.

2. Havanın oksijenini azaltmak, çok rahat kullanılamasa da ortama azot, karbonmonoksit gibi gazlar pompalayarak ortamdaki oksijen seviyesini %10 altına düşürerek, patlama üçgeninin oluşmasını engellemektir. Canlıların bulunduğu ortamlar için kullanımı mümkün olmasa da kullanım alanları bulunmaktadır.
3. Patlayıcı madde oranını değiştirmek, "alt patlama sınırının" altında veya "üst patlama sınırını" yukarısında tutulması temeline dayanır. Uygulama alanı bulabilen bir yöntem olarak kullanımı güvenlidir.
4. Havalandırma, patlayıcı madde oranını düşüreceği için patlama üçgeni oluşumunu engelleyecektir. Doğal havalandırma olabileceği gibi patlayıcı maddenin uzaklaştırılması fan gibi aletlerle zorlayarak da yapılabilir. Maden ocaklarında grizu patlaması ve çalışanların oksijen ihtiyacından dolayı mecburidir. Metan gazı havalandırmalarla dışarı atılır.
5. Patlayıcı ve yanıcı olan sıvıların içine patlama noktasını yükseltmek için farklı maddeler ilave edilerek de patlama engellenebilir.
6. Patlama olmadan değil patlama sonrası tahribatı önleyici tedbirlerde yine birincil önlemlerden sayılabilir. Bunun için patlama dayanıklı veya tahribat önleyici dizaynlar kullanılır.
    - **a.)** Basınç tahliye vanaları (relief valve) patlama oluştuğu anda açılarak basınç güvenli bölgeye yönlendirilir.
    - **b.)** Patlamayı bastırma (explosion suppression) tertibatları temeli oluşan patlamanın enerjisini yok etmeye dayanır. Toz patlamalarında kullanılır.
7. "Buhar bariyeri" ile patlayıcı gaz veya buhar oluşması önlenmektedir. Sıvı yakıtlarda yaygın olarak kullanılmaktadır (Sarı K., 2019).

4.4.2. İkincil (Sekonder) Önlemler

Yukarıda bahsedilen birincil önlemler uygulanamıyorsa, uygulanması durumunda bile risk devam ediyor ise ikincil önlemler kullanılır. Bu önlemler tehlikeli ortamda çalışabilecek ekipman ve alet kullanımı ile alınır. Burada kullanılan aletler exproof aletlerdir.

4.5. Patlayıcı Ortamların Sınıflandırılması

Her çalışma ortamının aynı olmaması ve kaynakların doğru kullanılması gerekliliklerinden dolayı her ortamda kullanılacak standart bir sistem kullanılamayacağından dolayı uzmanlar patlayıcı ortamları tehlike boyutlarına göre sınıflandırmışlardır.

Başta ekonomik nedenlerden ve bunun ardından bakım kolaylığı sebeplerinden dolayı nadiren patlama riski olan ortamlarla sürekli patlayıcı ortam olan yerler arasında alınacak önlemler arasında farklılıklar vardır. Bu farklılıklar kullanılacak aletler arasında da geçerlidir. Bu farklılıkları da gruplamak için sınıflandırma yapılmış, bu sınıflara ZON denilmiştir. Ülkemizde de bu sınıflara uluslararası kullanım ZON kelimesi kullanılmıştır. Bu sınıflandırmalarda iki görüş hakimdir, bunlar Batı Avrupa Görüşü ZON sistemi, Kuzey Amerikan Görüşü ve Division Sistemi ve uygulamasıdır.

Batı Avrupa genel standartlar da birleşmiş EN (euro norm) adı ile standartlarını yayınlamaktadır. Üye ülkeler bu standartlara tabi olmaktadırlar ve ülkemiz de Avrupa Birliği'ne uyum sürecinde olduğundan dolayı ülkemizde de önemli yer tutmaktadır. Ve ZON sistemi Kuzey Amerika Ülkeleri ve ABD hariç tüm dünyada kabul görmektedir ve IEC (IEC = International Electrical Commission) aynıdır (Sarı K., 2019).

4.5.1. Batı Avrupa görüşü ve uygulaması ZON sistemi

"ZON tanımları IEC 79-10 ve EN 50 014 de yer almaktadır. Son hali ATES 137 de (Avrupa Parlamentosu talimatı 99/92) düzenlenmiştir ve IEC' den farkı yoktur."

ZON 0: Sürekli olarak çalışma şartlarında patlayıcı ortam olan (veya ihtimali yüksek olan) ve patlayıcı ortam oluştuğunda uzun süre devam eden yerler ZON 0 olarak adlandırılır. Patlayıcı bulunan kapların içleri, patlayıcı üzerine çalışan aparatların içi gibi yerler ZON 0 olarak sınıflandırılır. Bu ZON da yüksek güvenlikli ve emniyetli aletler (ATEX100a' uygun) kullanılır. Kendinden emniyetli, a kategorisindeki devreler de burada kullanılabilir. (Ex-ia sertifikalı sistemler).

ZON 1: Sürekli olarak çalışma şartlarında patlayıcı ortam olma ihtimali az olan (veya ihtimali hiç olmayan) ve patlayıcı ortam oluştuğunda kısa süre devam eden yerler ZON 1 olarak adlandırılır. Arıza veya normal olmayan çalışma koşullarında veya denk gelişle patlayıcı ortam oluşturan veya ihtimal yaratan yerlerdir. Yani bu ortamlar patlama oluşma ihtimalsiz ve kısa süreli oluşan yerlerdir. ZON0 dışında kalan yakın mesafe yerler, pompa istasyonları, vana yakınları, akaryakıt istasyonların bazı bölgeleri bu gruba girer. Patlayıcı

ortamların çoğu bu gruptadır. Bu ZON da ikinci kategori emniyetli aletler (ATEX100a' uygun) kullanılır. Ex sertifikalı hemen hemen tüm aletler burada kullanılabilir.

ZON 2: Sürekli olarak çalışma şartlarında patlayıcı ortam olma olasılığı olamayan ve patlayıcı ortam oluştuğunda çok kısa süre devam eden yerler ZON 2 olarak adlandırılır. Arıza veya normal olmayan çalışma koşullarında veya denk gelişle patlayıcı ortam oluşturma ihtimali çok az olan yerlerdir. Yani bu ortamlar patlama oluşma ihtimalsiz ve kısa süreli oluşan yerlerdir

Yalnızca kaynaklı boru bağlantıları bulunan tesis veya tesisin kısımları, doğal gaz ve petrol boru hatları bu gruba girer. Bu ZON da üçüncü kategori emniyetli aletler (ATEX100a' uygun) kullanılır. Ex-n sertifikalı hemen hemen tüm aletler burada kullanılabilir.

Gaz ve buharla aynı olsa da tozlar için ayrı ZON'lar tanımlanmıştır (EN50.028). Tozlar için ZON10, ZON11, ZON12 isimleri kullanılmıştır. ATEX137 de bu isimler ZON20, ZON21, ZON22 olarak değiştirilmiştir.

ZON 20: Sürekli olarak çalışma şartlarında toz ve lif ortamı olan (veya ihtimali yüksek olan) ve oluştuğunda uzun süre devam eden yerler ZON 20 olarak adlandırılır.

ZON 21: Sürekli olarak çalışma şartlarında toz ve lif ortamı olma ihtimali az olan (veya ihtimali hiç olmayan) oluştuğunda kısa süre devam eden yerler ZON 21 olarak adlandırılır.

ZON 22: Sürekli olarak çalışma şartlarında toz ve lif ortamı olma olasılığı olamayan ve oluştuğunda çok kısa süre devam eden yerler ZON 22 olarak adlandırılır.

Tıbbi ortamlar ise Zon G ve Zon M iki sınıfa ayrılmaktadır.

ZON G: "Kapalı medikal gaz sistemi" olarak bilinir. Sürekli veya gelişi güzel, patlayıcı karışım (patlayıcı ortamdan farklı olarak) üretilen, taşınan veya küçük hacimlerde uygulanan yerlerdir. Bu yerlerin tamamen kapalı olması gerekmez, ufak köşe ve oyuklar bu kapsama girer.

ZON M: "Medikal ortam" olarak bilinir. Ağrı kesici madde veya tıbbı deri temizleme, dezenfekte, antiseptik ilaç kullanımı gibi olaylarda, küçük miktarda ve kısa süreli patlayıcı ortam oluşan ve oluşma ihtimali oyan yerleri kapsar. Zon sisteminde patlayıcı gazlar G ve tozlar da D harfi ile belirlenir. Tıbbi ortamlardaki zon tarifi ile karıştırılmamalıdır. Gazlar IEC ve EN de aşağıdaki gruplara ayrılmaktadır.

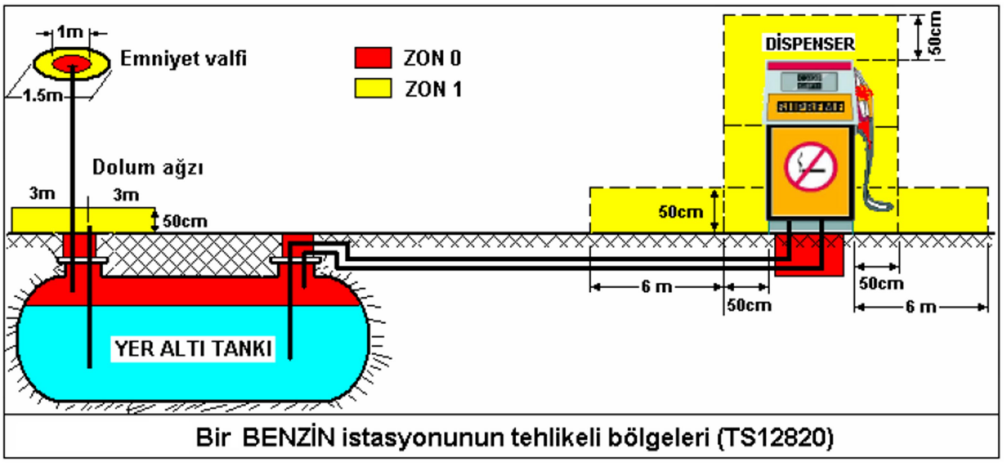
Gaz Grupları ise IEC ve EN gazları iki patlama grubunda toplamış, metan gazına (grizulu madenleri) I.grubta yer vermiştir.. Başka bir ifade ile EN madenleri diğer sanayi dallarını ayırmıştır (Sarı K., 2019).

PATLAMA GRUBU I: METAN

PATLAMA GRUBU II A: Propan, bütan, aseton, kereson, hexan, trimat, hylamin, v.b.

PATLAMA GRUBU II B: Etilen, karbon monoksit, hidrojen sülfit, etil-, -metil, -eter, v.b.

PATLAMA GRUBU IIC: Hidrojen, Asetilen ve karbon disülfit

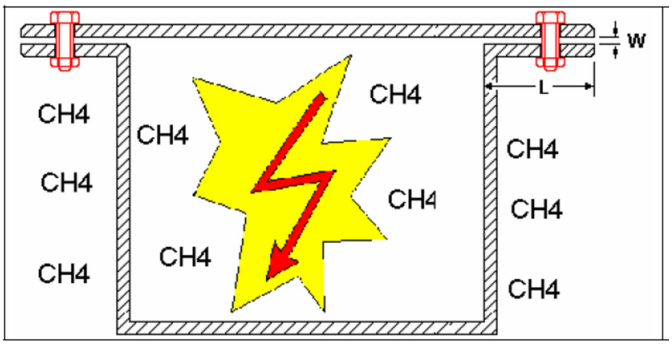


Şekil 4.5. Benzin istasyonunun tehlike bölgeleri

4.6. Koruma Tipleri

4.6.1. d tipi koruma (EN 50018, IEC 60079-1, TS 3380)

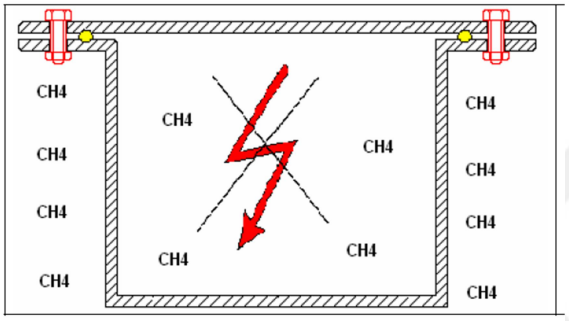
Bu koruma yönteminin temel mantığı enerji kaynağı (transformatör, kesici, yol verici v.b.) olan cihazların basınca dayanıklı bir korumanın içine konulmasıdır. Koruyucun içine patlayıcı gazlar sızma yapabilir, patlamaya sebep olabilir lakin içeride olan patlama dışarıdaki patlama ortamına gereken enerjiyi vermez. Bu özelliğinden dolayı, "ALEV SIZMAZ KORUMA" olarak da adlandırılır. Çok geniş bir uygulama alanına sahiptir.



Şekil 4.6. d-tipi koruma sistemi

4.6.2. e tipi koruma (EN 50019, IEC 60079-7, TS 3385)

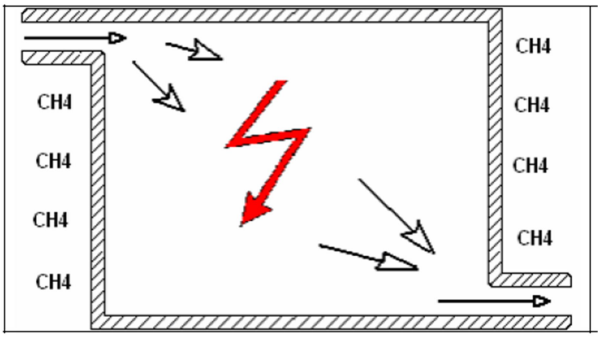
D-tipi korumanın güvenliğinin daha da artırılmış halidir. Sürekli çalışma koşullarında ark üretmeyecek klemens kutuları, sincap kafes asenkron motor, kablo bağlantıları, küçük transformatör v.b. cihazların yanlış kullanım veya arıza durumunda ark çıkarma ihtimaline karşı kullanılır. Yukarıdan da anlaşılabildiği gibi bu koruma tipinde korumanın içine sürekli çalışma koşullarında ark üreten cihazlar kullanılmaz.



Şekil 4.7. e-tipi koruma sistemi

4.6.3. p tipi koruma (EN 50016, IEC 60079-2)

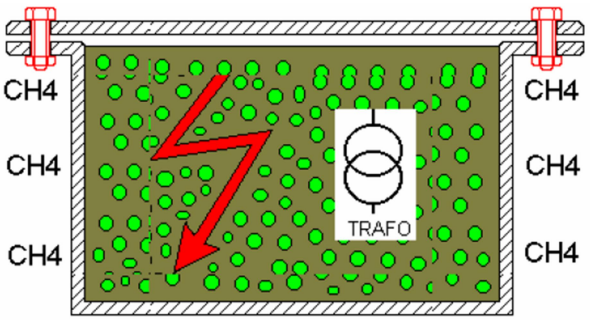
Basınçla sağlanan bir koruma tipidir. Koruma bölgesindeki basınç patlayıcı gazın veya buharın bulunduğu ortamdan daha yüksek tutularak bu gaz ve buharların ortama girmesi engellenir. D tipi koruma uygulanan yerlerde uygulanabilir. Örnek olarak bilezikli asekron motorların fırçaların olduğu bölgede ark çıkma ihtimali yüksek olup bu bölge basınç altında tutularak patlayıcı gaz veya buharın buralara girmesi engellenir.



Şekil 4.8. p-tipi koruma sistemi

4.6.4. q tipi koruma (EN 50017, IEC 60079-5)

Dar kullanım alanı olmakla beraber Fransa'da kullanımı bulunmaktadır. Genel olarak kuvars kumu ve toz doldurarak gazın girmesini istemediğimiz yerleri bloke ederiz. Transformatörlerde kullanıldığı gibi elektronik devrelerde de hem soğutma amacı ile hemde engelleyici olarak kullanılmaktadır.



Şekil 4.9. q-tipi koruma sistemi

4.6.5. o tipi koruma (EN 50015, IEC 60079-6)

Standartlardan çıkarılmasa da kullanımı yasaklamış bu yöntem ark çıkaran veya ısınan cihazlar yağ içine konularak patlayıcı ortamdan korunmaktadır. Trafo ve kesicilerde 70'li yıllarda kullanılmış bu yöntem herhangi bir hata ile patlamada patlayıcı gazın patlamasından daha çok zarar vermektedir. Bu sebeple kullanımı yasaklanmıştır. Sadece soğutma ihtiyacından dolayı dirençlerde, bazı büyük ebatlı transformatörlerde ve kesicilerde kullanılır. Taşınabilir ve küçük cihazlarda kullanılmamaktadır (Sarı K., 2019).

4.6.6. m tipi koruma (EN 50014, EN 50028, IEC 60079-18, IEC 61241-6 Ex-mD)

Bu koruma yönteminde enerji kaynakları özel reçinelerin içine gömülmekte ve patlama tehlikesi ortadan kaldırılır. Bu reçine dökülerek uygulandığı için bu yöntemin kullanılmasında mani olmayan elektronik baskı devrelerde, sadece bobinden oluşan selenoidler, mini röleler ve küçük güç transformatörlerde kullanılabilmektedir. Ama bu yöntem genellikle kendinden emniyetli uygulamalarda kullanılır.

4.6.7. n tipi koruma (EN 50021)

1999'dan sonra Avrupa standartlarına giren bu koruma tipi ZON2 ortamları için öngörülmüş bir koruma tipidir. Amerikan standartlarından "Non-sparking" olarak bilinir. Amerikan standartlarından "non encendive" a benzer nA, nC, nL, NP, nR, nL şeklinde ayrılmışlardır.

nA = Ark üretmez olarak adlandırılır, sürekli çalışma koşullarında ark üretmeyen aletler bu tip koruma yöntemi ile patlayıcı ortama karşı korunabilirler. "Ex-e tipi (artırılmış emniyet) korumanın yönteminin hafifletilmiş halidir.

nC = sürekli çalışma koşullarında ark üreten aletler, nC tipi korunarak ZON2 ortamlarda kategori3 sınıfı aletler olarak kullanılabilir. Hafifletilmiş Ex-d koruma veya değişik bir versiyonu olarak da düşünülebilir. Ark çıkaran, aşırı ısınan kısımların patlayıcı ortamdan çıkarılması ile önleme sağlanmış olur. Bu işlem ark çıkaran bölümlerin kaynak, döküm, lehim, saire v.b. yöntemlerle kapılması ile sağlanmaktadır. Örnek olarak reed switchler verilebilir. Bu koruma tipinde sürekli çalışmada ark çıkaran aletler de kısıtlama bulunmakta ve bunlar 690 volt ve 16 Amperi aşmamaları ve 20cm3 ü geçmemelidir. Ve ısı olarak da çıkardığı ısının 10K üstüne kadar dayanmalıdır. Yani küçük hacme sahip ısınmayan cihazlar aletlere Ex-nC tipi koruma uygulanabilmektedir.

nR = Özel bir sızdırma deney düzeneği ile denenen, ark çıkaran kısmın havalandırılması sayesinde patlayıcı gaz ve buharın buraya girmesi engellenmiştir.

nP = Basınçlı koruma yönteminin basitleştirilmiş veya uygulama şartları hafifleştirilmiş halidir. Zon2 bölgelerde kullanılır. Patlayıcılık güvenlik seviyesi düşük kategori 3 aletlere uygundur.

nL = Enerji seviyesi fazla olmayan aletler için uygundur, Kendinden emniyetliliğin hafifletilmiş şeklidir (Sarı K., 2019).

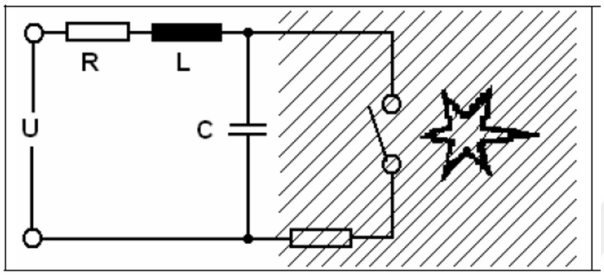
4.6.8. s tipi koruma

Bilinen patlayıcı ortam korunma yöntemlerinin kullanılamadığı ve farklı bir yöntem gerektiği durumlarda kullanılır. Örneğin madenci baş lambaları, dinamit manyetoları. Dinamit manyetoları yukarıda anlatılan veya kendinden emniyetli olarak imal edilemez çünkü dinamiti patlatmak için belirli bir enerji gerekmektedir. Burada çözüm olarak verilen enerjinin süresi kısaltılarak emniyeti sağlamak mümkün olmuştur.

4.6.9. Ex-td tipi koruma (IEC 61241-1-1)

Bu yöntem ise d sınıfı koruman biraz daha gelişmiş halidir. Aletin gövdesi sayesinde en küçük zerreler bile muhafazanın içine giremez ve ısı yeterince yükselmez. En az IP60 koruma uygulanır.

4.6.10. i tipi koruma (EN50020)



Şekil 4.10. i-tipi koruma sistemi

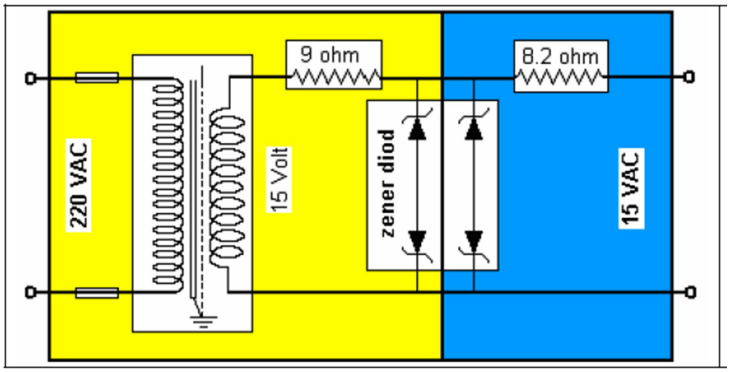
Bu koruma yöntemi "INTRINSICALLY SAFE" den kısaltılmıştır. Devre yapısı itibari ile kendinden emniyetlidir. Sürekli çalışma, bakım, arıza durumunda tehlikeli bölgedeki kısım gerekli büyüklükte enerji sağlayamaz ve bu sayede oluşan ark veya ısı patlamaya sebep olmaz. Söz konusu sadece alet değil, aletin beraberindeki devrenin tamamıdır.

Bu koruma yöntemine sahip aletlerin çıkaracağı enerjinin küçük olması kullanım alanlarını kısıtlamakta genel olarak kumanda devreleri, otomasyon devreleri gibi düşük güçlü devrelerdir. Bu yöntem arıza ve bakım durumunda da güvenli olduğundan dolayı en güvenilir koruma yöntemidir. ZON0'da da çalışabilen bu aletlerin tamiratı mümkündür. D sınıfı koruma sahip aletlerde yapılan denemeler ardı ardına yapılan altı patlamayı baz alırken kendinden korumalı aletlerde bu sayı yüzün üzerindedir. Çünkü d koruma sınıfına ait aletler ısının soğuması için belli bir süreye ihtiyaç duyarken kendinden korumalı devreler fazla ısınmamaktadır

Bu korunma yöntem birçok tedbir alınmasına rağmen İngiltere'deki yüksek can kayıplı maden ocaklarındaki yaşanan kazalardan sonra İngiliz akademisyenler tarafından geliştirilmiştir.

Kendinden emniyetli devreleri besleyen güç kaynakları da kendinden emniyetli veya d tipi korumaya sahip olabilir. Kendinden emniyetli çıkışı olan güç kaynağı her tip alete bağlanabilmekte ve bu aletler kondansatör veya bobin gibi enerji depolayıcı özelikte olmamalıdır. Kendinden emniyetli devrelerin imalatçıları bağlantı şemalarını gereken yerlere

verir ve bu aletler rastgele bağlanamazlar. Ve bu aletlerin bağlanacağı çevre elemanları da dikkatli olarak belirlenmeli ve kendinden emniyeti bozmaması gerekir.



Şekil 4.11. i-tipi koruma sistemi detaylı görünüm

Kendinden emniyetli aletin oluşturduğu enerji, gazları patlatamayacak kadar az bir enerji olduğuna göre gazların alt enerji seviyesi olup olmadığı düşünülmektedir. Eğer böyle bir enerji sınırı var ise bu seviyenin altında enerji üreten aletlerin otomatik olarak kendinden emniyetli olması gerekmektedir. Yalnız ark oluşmasında pek çok etmen olduğu için böyle bir enerjiden bahsedilememiştir. Ve ark üretmeyen bir gerilim seviyesi olmadığı için herhangi bir kendinden emniyetli aletlerde belirli bir değerin altındaki gerilim değeri uygulanırsa bu devre kendinden emniyetli denilemez. Ama üst sınır yirmi dört volt olarak belirlenmiştir (Sarı K., 2019).

5. ŞİFRELEME ALGORİTMALARI

5.1. Giriş

Son yıllarda askeri ve WLAN uygulamalarında yaşanan artışla birlikte kablosuz çözümlerde güvenlik daha önemli bir hale gelmiştir. Birçok askeri ve WLAN uygulamaları için üç temel tehditten bahsedebiliriz. Bunlar bilginin üçüncü partilere geçme riski, ağın doğrudan bloke edilme tehlikesi ve kablosuz ağa yetkisiz giriş ihtimali olarak tanımlanabilir. Özellikle büyük şirket ağlarında IEEE802.11 ve WLAN uygulamalarının kullanılmaya başlanması ağ güvenlik uygulamalarını sorgulanır hale gelmesine sebep olmuştur.

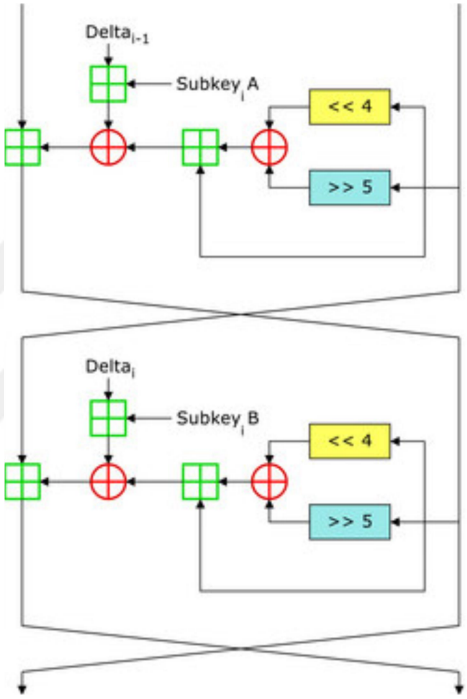
Tarihi süreçte IEEE 802.11'de güvenlik için WEP- Wired Equivalent Privacy kullanılagelmekteydi. Fakat bugün IEEE 802.11'i gibi yakın vadedeki standartlarda kullanımda olan WEP protokollerinin gelişmiş bir versiyonu olarak da görülen TKIP-Temporal Key Integrity Protocol ve orta vadede CBC-MAC gibi AES destekli protokoller kullanılacağı öngörülmektedir. Çünkü IEEE 802.11'in bu standartla protokolün yetkisiz girişleri önlemek üzere authentication, veri gizliliğini ve çalınmasını önlemek üzere privacy özelliklerinin, kablolu sistemlerdeki ile eşdeğer olması beklenirken durumun öyle olmadığı, WEP key diye adlandırılan anahtarların ve WEP IV diye adlandırılan başlatma vektörleri uzunluklarının yetersiz olduğu görüldü.

Genelde RF modul MAC (media access control) katmanı yazılımı üzerinde yapılan şifreleme algoritmalarının (Data Encryption Algorithm) seçiminde başka birçok konuda olduğu gibi; işlem hızı, code boyutu ve güvenlik arasında bir dengeleme yapmak gereklidir. Aşağıdaki tabloda yaygın olarak bilinen ve kullanılmakta olan şifreleme algoritmalarının işlem hızı ve code boyu olarak kıyaslamalarını görebilirsiniz. Uygulamanızda hız güvenlik ihtiyacından daha önemli bir etkense muhtemelen PRBS algoritması en iyi tercih olacaktır. Eğer uygulamanızın güvenlik ihtiyacı code boyutu, maliyet ve işlem hızından daha önemli bir gereklilik ise XTEA 32 (veya daha fazla) veya AES algoritmalarını kullanmak daha doğru olacaktır (Udea, 2019).

5.2. Tiny Encryption Algorithm

Tiny Encryption Algorithm (TEA), çok kısa olan kod boyutu ve basit algoritması sayesinde özellikle kod boyutunun oldukça sınırlı olduğu gömülü sistemlerde oldukça popüler olan bir şifreleme algoritmasıdır. TEA 1995 yılında, Roger M. Needham ve David J. Wheeler

tarafından yayınlandı. 1997 yılında algoritma üzerinde iyileştirme yapılarak XTEA, algoritması duyuruldu. http://www.cix.co.uk/~klockstone/ adresinden XTEA ve TEA hakkında detaylı bilgiye ve dökümanların orjinallerine ulaşılabilir. XTEA algoritması basit bit kaydırma ve toplama işlemlerinin tekrarlanmasından oluşan bir algoritmadır. Şekil 5.1'de algoritmanın birbirini tekrar eden 2 bloğu görülmektedir. Bu blokların sayısı artıkça yapılan şifrelemenin çözülmesi gittikçe daha zorlaşmaktadır. XTEA'nın geliştiricileri sağlam bir şifreleme için bu işlemin en az 64 kez yapılaması gerektiğini belirtmekte fakat 32 tekrarında çoğu durumda yeterli olacağını belirtmişlerdir (Udea, 2019).



Şekil 5.1. Tiny algoritması sembol gösterimi

5.2.1. Kaynak Kodu

```
tean( long * v, long * k, long N)
{
        unsigned long y=v[0], z=v[1], DELTA=0x62112345;
        if (N>0)
        {
        /* coding */
                unsigned long limit = DELTA * N, sum = 0 ;
                while (sum != limit)
                y += (z << 4 ^ z >> 5) + z ^ sum + k[ sum & 3 ],
                sum += DELTA,
                z += (y << 4 ^ y >> 5) + y ^ sum + k[sum >> 11 & 3] ;
        }
```

```
        Else
        {
        /* decoding */
                unsigned long sum = DELTA * ( -N ) ;
                while (sum)
                z -= (y << 4 ^ y >> 5) + y ^ sum + k[sum >> 11 & 3];
                sum -= DELTA,
                y -= (z << 4 ^ z >> 5) + z ^ sum + k[ sum & 3 ] ;
        }
        v[ 0 ] = y, v[ 1 ] = z ;
        return ;
}
```

Kaynak kodunu incelediğimizde ‘k’, ‘v2, ‘N’ olmak üzere 3 değişken kullanıldığını görmektedir. ‘v’ şifrelenecek verinin adresidir ve 64 bitlik bir bloğu gösterir, ‘k’ şifrelemede kullanılacak anahtardır. 128 bitlik bir bloğu gösterir. ‘N’ ise şifreleme işleminin kaç kez tekrar edileceğini gösterir. Fonksiyon ‘N’ pozitifse şifreleyici, negatif ise şifre çözücü olarak çalışır (Udea, 2019).

5.3. Algoritmalar Kıyaslama Tablosu

Çizelge 5.1. Algoritma hız çizelgesi

Algoritma	**Tekrarlama (Iteration) Sayısı**	**Güvenlik Seviyesi**	**Encoding Cycles Per Byte (Yaklaşık)**	**Decoding Cycles Per Byte (yaklaşık)**	**Encode Inst./Sec Byte/Sec**	**Decode Bytes/Sec**
PRBS XOR encryption with skipping key	KeyJump=1	Zayıf	92-146 (**,*)	92-146 (**,*)	68493	68493
XTEA (Tiny Encryption Algorithm)	16 iteration	Yüksek	1075 (*,***)	1280 (*,***)	9302	7813
XTEA (Tiny Encryption Algorithm)	32 iteration	Yüksek/ Çok Yüksek	2133 (*,***)	2194 (*,***)	4688	4558
AES(Rijndael Algorithm)		Yüksek/ Çok Yüksek	2153	2940 (****)	4645	3401

*Data uzunluğuna bağlıdır, **Kullanılan anahtara bağlıdır, ***Tekrarlamaya bağlıdır, ****Decode anahtarının gömülü veya üretiliyor olmasına bağlı

Çizelge 5.2. Algoritma kod boyutları

Algoritma	ROM	RAM
PRBS XOR (encryption with skipping key)	226	12*
XTEA (also referred to as TEAN or TEA-N)	1950	38*
AES (Rijndael Algorithm	6104	33*

* Şifrelenecek datayı içermemektedir

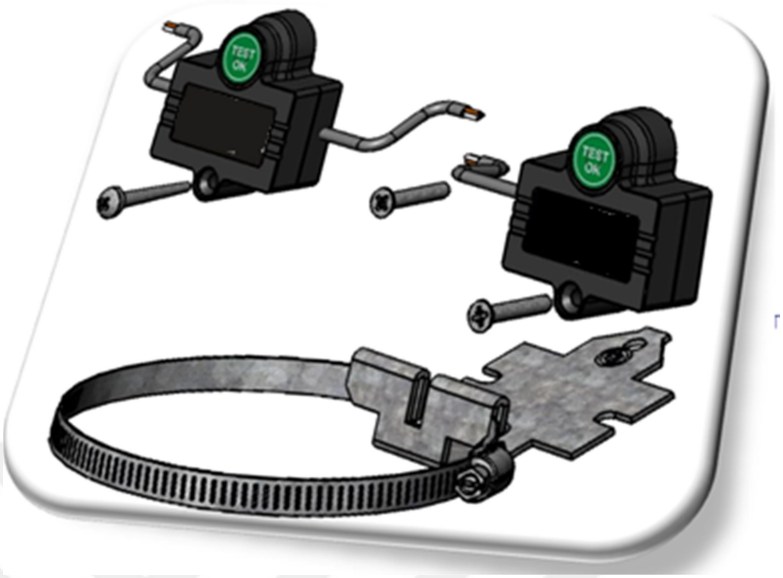
6. UHF ARAÇ TANIMA EKİPMANLARI TASARIMI

6.1. Giriş

Genel işleyiş olarak okuyucu ünite, öncelikle kendisinin bağlı bulunduğu (aynı kanalda çalışan) merkezi toplayıcı birimden bundan sonraki haberleşmelerde anahtar olarak kullanılacak anahtar isteme mesajı ile anahtar ister. Eğer anahtarı alabilir ise aracın deposuna konulduğunda araçta takılı olan ve araç için özel bir kimlik numarası taşıyan kimlik birimine kimlik numarasını isteyen bir mesaj gönderir. Bu mesaja cevap olarak araç kimlik birimi içerisinde kayıtlı olan kimlik numarasını ve okuyucu üniteden gelen sinyalin gücü ile daha önce kendisine kaydedilmiş sinyal gücünü karşılaştırır, iki değer arasındaki farkı okuyucu üniteye mesaj olarak iletir. Eğer okuyucu üniteye kaydedilen fark değerine uygun ise bu kimlik numarası okuyucu ünite tarafından daha önce belirlenen anahtar ile şifrelenerek merkezi toplayıcı birime iletilir. Eğer bu değer uygun değil ise tekrar sorgulama yapar. Bu kimlik numarası otomasyon sisteminin sorgu mesajı sırasında gönderdiği anahtar yardımı ile merkezi toplayıcı birime sorgusuna cevap olarak şifreli olarak iletilir. . Bu şekilde taşıt tanıma sistemi görevini yerine getirmiş olur. Otomasyon sistemi bu kimlik numarasını veritabanının bulunduğu server'dan doğrulamasını yaparak yakıt ikmaline izin verir veya izin vermez.

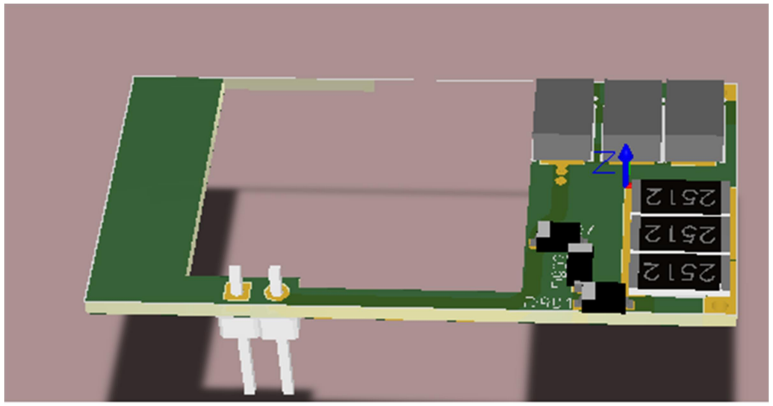
Şekil 6.1.UHF araç tanıma sistemi genel yapısı

6.2. Kimlik Birimi Tasarımı

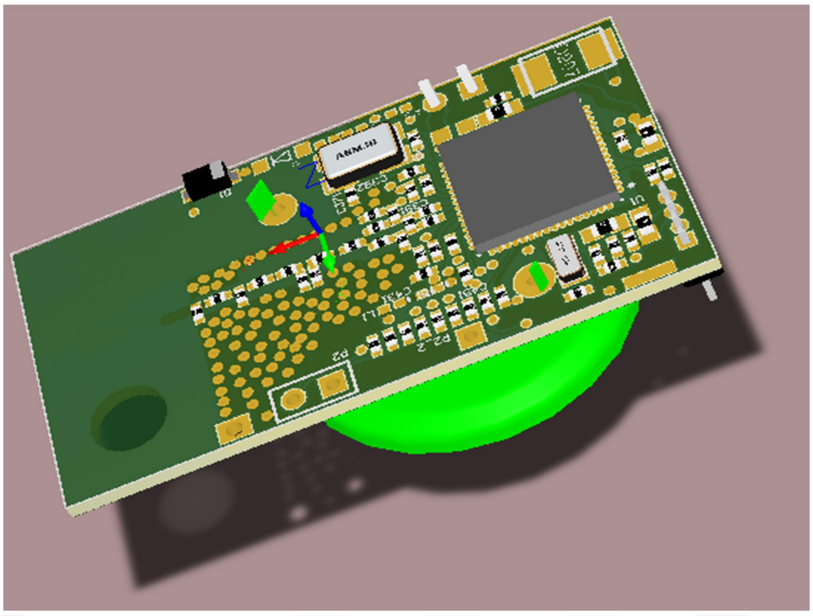


Şekil 6.2. Kimlik birimi bileşenleri

Kimlik birimi patlayıcı ortamda çalışacağı için akım ve voltaj sınırlamasını sağlayan bir alt kart ve tüm cihazlarda kullanılan, farklı programlar sayesinde değişik özellikler kazanan modül kartı olarak iki karttan oluşmaktadır. Bu kartlar komple bir bütünü oluşturmasına rağmen ayrı ayrı olma sebebi cihazın daha küçük hacme sığdırma gereksimi ve modül kartının diğer cihazlarda da kullanılacak olmasıdır. Bu cihaz birçok çeşit araca takılacağı için küçük boyutlarda olması gerekmektedir. Kartların çizimlerinin üç boyutlu olarak görünümleri aşağıdaki gibidir.

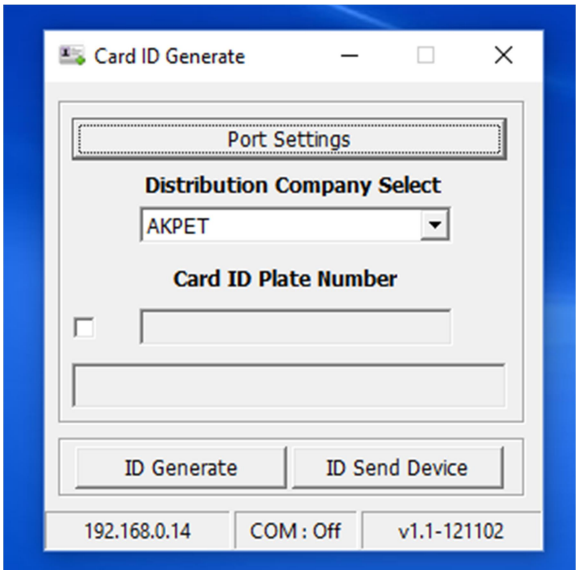


Şekil 6.3. Kimlik birimi alt kart



Şekil 6.4. Kimlik birimi üst kart

Üretim aşamasında bu iki kart pin headerlar ile birleştirilerek sistemin çalışması için gerekli kimlik numarası cihaza verilir. Bu kimlik numarası merkez server'dan tarih saat alarak ve hangi şirkete ait olacağını seçilmesinin ardından bir kimlik numarası üreten bilgisayar programı tarafından üretilir. Bu numara kimlik birimine merkezi toplayıcı birim ile aynı donanımı kullanan sadece üretim aşamasında kullanılan, aşağıda resmi bulunan ve daha sonra bahsedilecek olan prosedürlerde kullanılmak üzere tasarlanan, programı yazılan "Kimlik Numarası Verici" cihaz ile verilir. Bu cihazın bilgisayar ile bağlantısı RS232-RS485 çevirici veya USB-RS485 çevirici ile yapılır. Ayrıca bu bilgisayarın merkez server'a bağlanmak için mutlaka internet bağlantısı olmak zorundadır. Ekran görüntüsü aşağıdaki gibi olan bu program sayesinde kimlik birimine verilecek olan kimlik numaralarının içerisine hangi dağıtım şirketine ait olduğu bilgisi de konulabilmektedir. Bu program Delphi programlama dilinde MLB Mekatronik yazılım ekibince taşıt tanıma ekipmanlarında kullanılmak üzere yazılmıştır.



Şekil 6.5. Kimlik numara verici program

Şekil 6.6. Kimlik numara verici cihaz

Bu program sayesinde üretilen ve üretim aşamasında kimlik numarası verici cihaz ile kablosuz olarak kimlik birimine gönderilen kimlik numarası kullanılan işlemcinin "Flash Memory" sine kaydedilir. Bu kimlik numarası başka bir eşi olamaması, bundan sonra takılacağı araç ile özdeşlecek olması gerekliliğinden dolayı bu kimlik numarası tarih ve saate bağlı olarak üretilerek kimlik verme işlemi sırasında hata ile aynı kimlik numarası verme ihtimali ortadan kaldırılmıştır. Ve verilen kimlik numaraları, bu numaraları üreten bilgisayar programı tarafından saklanır. Daha sonra buradan kontroller yapılabilir.

Bu verilen kimlik numarası cihazda Flash Memory'nin korunaklı olan kısmında saklanır bu sayede elektriksel veya programın bilinmeyen hatalardan bu kimlik numarasının zarar görmesi veya silinmesi engellenmiş olur.

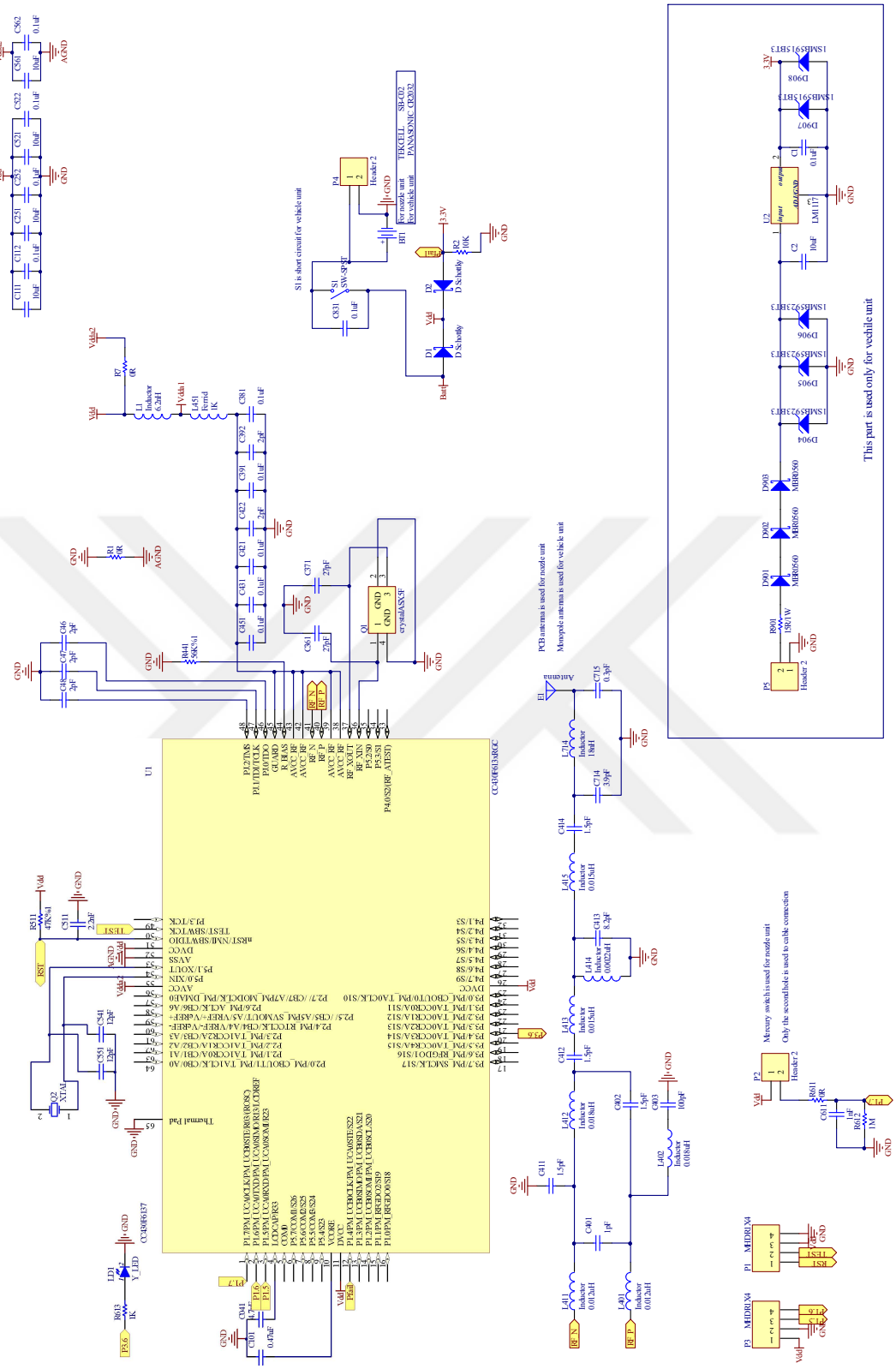
Kimlik birimi sökülüp başka bir araca takılmaması gerekliliğinden dolayı bu birimin yerinden söküldüğünde artık kimlik numarasını göndermemesini gerekmektedir. Bu ise kimlik birimi bağlantılarındaki üçüncü bir kablonun yerinden sökülmesi ile sağlanmaktadır.

Bu işlemin cihazın kesildikten sonra kontrolünün devamını sağlamak için kimlik birimi üzerinde ayrıca pil bulunmaktadır. Bu pil cihazın normal çalışmasında enerji sağlamadığı için uzun sürelerce kimlik biriminin bu özelliği korunabilmektedir.

Kimlik birimi üzerinde RSSI ölçümü olduğu için ve bu ölçümün kimlik birimine yaklaşılan her yönden eşit değişmesini sağlamak için bu birim üzerinde kutu içerisine uygun, montajı sırasında üretimi yavaşlatmayacak şekilde denemeler yapılarak özellikleri belirlenen monopol anten kullanılmıştır. Monopol anten seçilmesinin sebebi ise kullanılacağı yer de her yönden eşit algılama yapmasıdır.

Kimlik biriminde ve diğer ekipmanlarda TEXAS INSTRUMENTS firmasının üretmiş olduğu CC430 serisi mikrodenetleyiciler kullanılmıştır. Bu denetleyicinin kullanımla sebeplerinden biri dahili olarak 868 MHZ yayını sağlayan radyo modülünün olmasıdır. Küçük alanda uygulama yapmak için ve çevre elemanları azalmak amacı ile tercih edilmiştir. Ayıca bu denetleyiciler aşırı düşük güç tüketimine sahip oldukları için çok az akım çekmekteler ve uyku modlarında birkaç mikroampere kadar güç tüketimleri düşmektedir. Özellikle tabanca ekipmanında bu özelliği çok önemli olmaktadır.

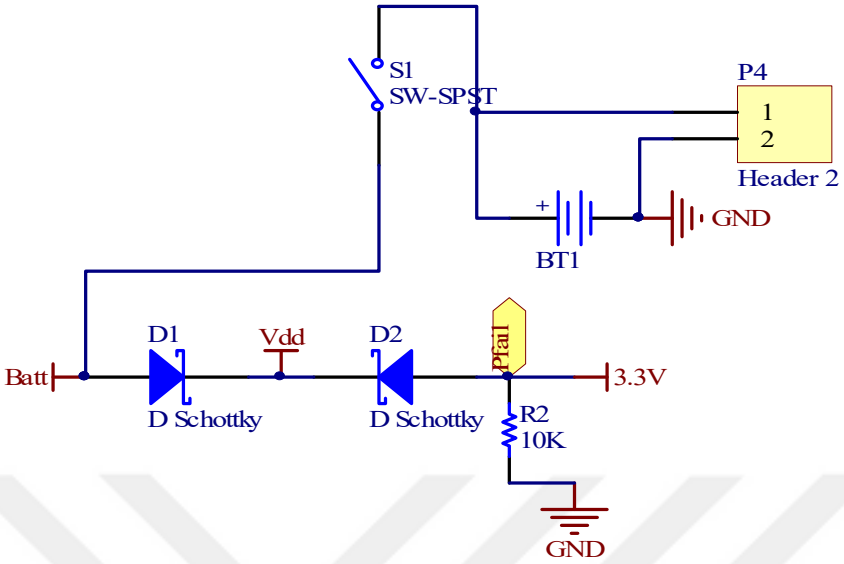
Kimlik biriminin tasarımındaki şematik çizimleri aşağıda görüldüğü gibidir.



Şekil 6.7. Kimlik birimi şematik çizimi

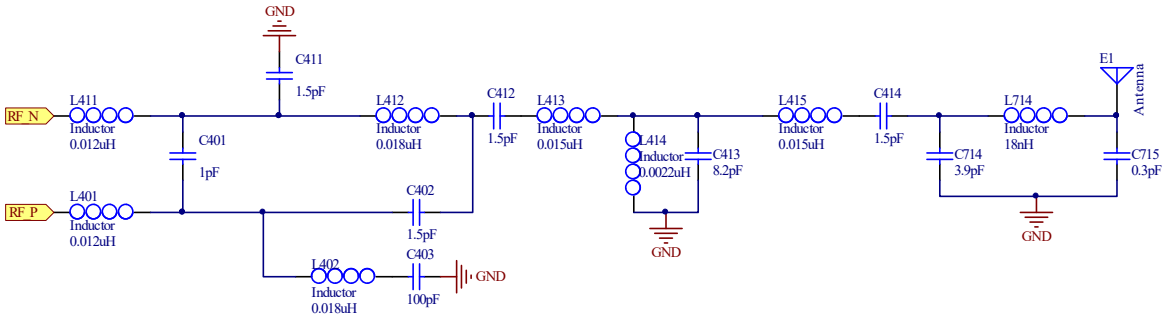
Bu tasarımı bölüm bölüm incelersek;

Güç bölümü aşağıdaki şekildedir



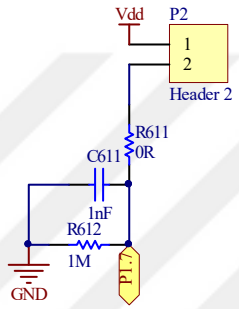
Şekil 6.8. Kimlik birimi besleme bölümü şematik çizimi

Burada modül ortak kullanıldığı için S1 anahtarı kısa devre edilerek üretilmektedir. Devredeki schotky diyotlar sayesinde güç ünitesinde 3,3 volt olduğu sürece 3 voltluk BT1 pilinden güç harcanmamış olur. PFAİL olarak adlandırılan kısım ise mikrodenetleyicinin ilgili ucuna gitmekte, güç ünitesinden güç alınamadığı sürede sistem uyku moduna geçer ve güç tüketimini minimuma çeker. Bu süreçte cihaz güç gelene kadar sadece yerinden sökülünce kimlik biriminin pasif olacağı ucunun ve gücünün tekrar gelip gelmediğinin kontrolünü yapar. Radyo haberleşmeleri ve diğer fonksiyonlar tamamen kapatılır. Bu sayede sistem mikroamperler seviyesinde güç tüketimine geçer. Buda sistemin 225 mAh'lik pilinin yıllarca yeterli olmasını sağlar.



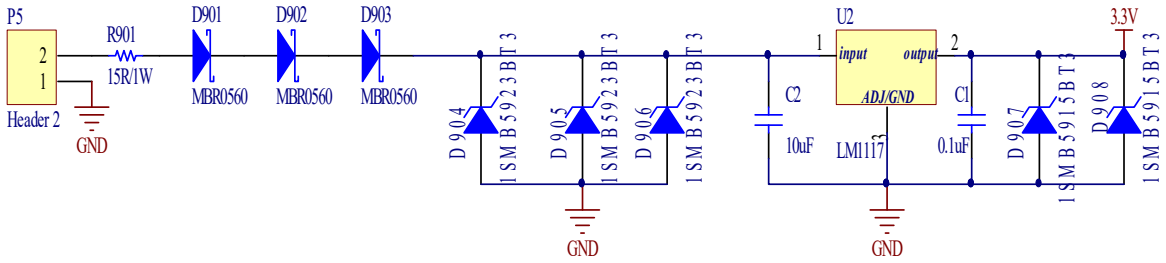
Şekil 6.9. Kimlik birimi anten bölümü şematik çizimi

Yukarıdaki devre ise mikrodenetleyicinin RF match devresidir. Buradaki değerler kullanılan antenin ve yolların görünür değerinin uygun olabilmesi için uzun süreli denemeler sonucunda bulunmuştur. Burada değerler bulunması için çıkış gücünün maksimum olması sağlanmaya çalışılmıştır. Bu modül ortak kullanıldığı için tabanca ekipmanında PCB anten kullanılırken araca takılan kimlik biriminin her yönden eşit güç alabilmesi gereksiminden dolayı monopol anten kullanılmıştır. Burada bu antenin şekli ve uzunluğu kullanılacak kutunun şekli ile uygun olarak belirlenmiştir. Burada kimlik biriminde maksimum güç transferi yerine asıl olarak değişik bölgelerdeki eşit uzaklıklardaki güç ölçümleri yakın olması hedeflenmiştir.



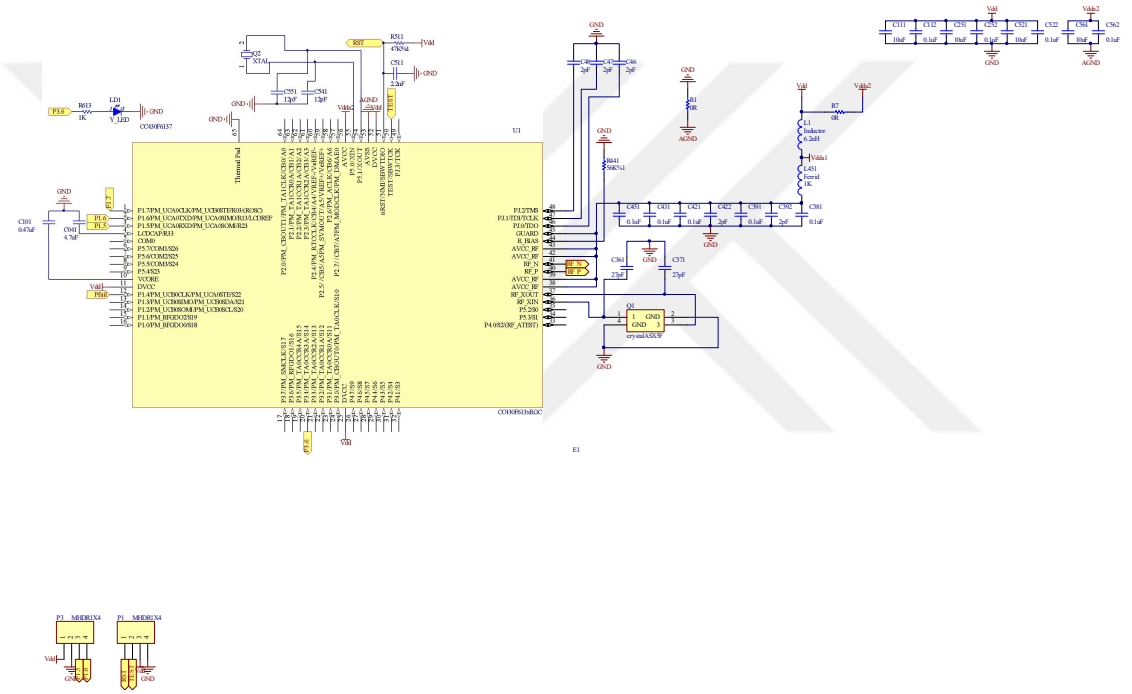
Şekil 6.10. Kimlik birimi kopma tespit bölümü şematik çizimi

Yukarıdaki devre ise cihazın yerinden söküldüğünü anlamak için kullanılmıştır. Cihazın P2-2 ucundan mikrodenetleyici kablonun bağlı olduğu bilgisini alır. Ve buradaki devrede logic 0 gördüğünde ise cihaz kablo bağlantısının ayrıldığını bir interrupt ile algılamakta ve aktivasyonunu iptal etmektedir. Burada direnç değeri düşük akım çekmesi için yüksek kullanılmıştır.



Şekil 6.11. Kimlik birimi zener bariyer bölümü şematik çizimi

Bu devre ise kimlik biriminde aküye bağlı zener bariyer güç ünitesinden gelen voltajı LM1117-3,3 voltaj regülatörü ile 3,3 volt seviyesine çevirmektedir. Giriş kısmındaki 15R/1W direnç akım sınırlamakta ardarda olan MBR05060 diyotlar ise oluşabilecek herhangi bir ters akımı engellemek içindir. 1SMB5923BT3 8.2V 3W zener diyot olup girişte oluşabilecek aşırı voltajları önleyerek regülatörü korumaktadır. Devrenin giriş kısmında görülen birden çok diyotun sebebi ise ATEX kurallarına göre herhangi ikisi zarar görürse sistemin devamını sağlamasıdır. Çıkış kısmında ise 1SMB5915BT3 3.9V 3W zener diyot ise 3,3 volt çıkışını korumak içindir. Bu devre okuyucu ünite pil ile çalıştığı ve dış dünyadan reçine ile ayrıldığı için kullanılmamıştır.

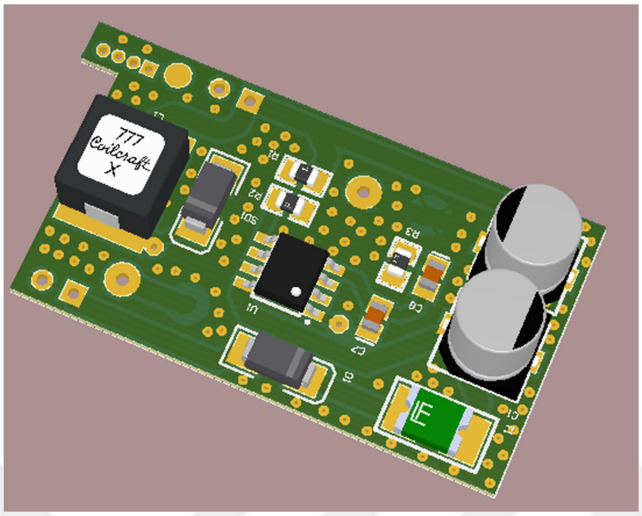


Şekil 6.12. Kimlik birimi sistemin gereksimleri bölümü şematik çizimi

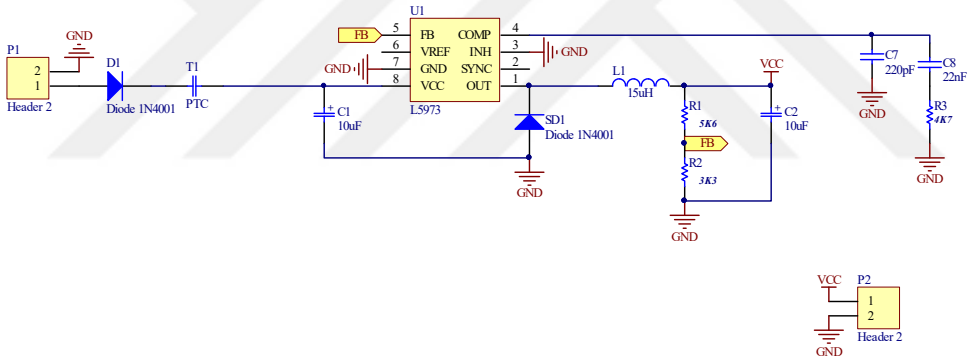
Bu devre ise mikrodenetleyicinin bağlantılarını göstermektedir. Burada devrenin çalışması için gerekli minimum gereksinimler ve sistemin gereksimlerini sağlaması için olan bağlantılar gösterilmektedir.

Yukarıda bahsedilen bu cihaz ile birlikte kullanılan güç ünitesi ise LM1117 in giriş voltajının datasheetlerde maximum 20 volt olarak verilmesi ve büyük araçlarda aküden en kolay 24 volt alınabilmesidir. Ve kademeli olarak düşürülen voltaj sayesinde LM1117-3,3 üzerinde aşırı güç olmayacak ve ısınma gibi bir durum olamayacaktır. Burada kullanılan step

down switching regulatörün giriş voltajı maksimum 40 volta kadar çıkabilmektedir. Bu entegre 2,25 amper akım sağlanmasına izin vermekte bu da sistem için yeterli olmaktadır.



Şekil 6.13. Kimlik birimi güç ünitesi üç boyutlu görünümü



Şekil 6.14. Kimlik birimi güç ünitesi şematik çizimi

Yukarıda kartın üç boyutlu görünümü ve şematik çizimi bulunmaktadır. Burada girişteki diyot sayesinde bağlantı yaparken ters bağlantılarda cihazın zarar görmesi engellenmiştir ve bunun takibinde bulunan resetlenebilen sigorta sayesinde aşırı akım çekmesi engellenmiştir.

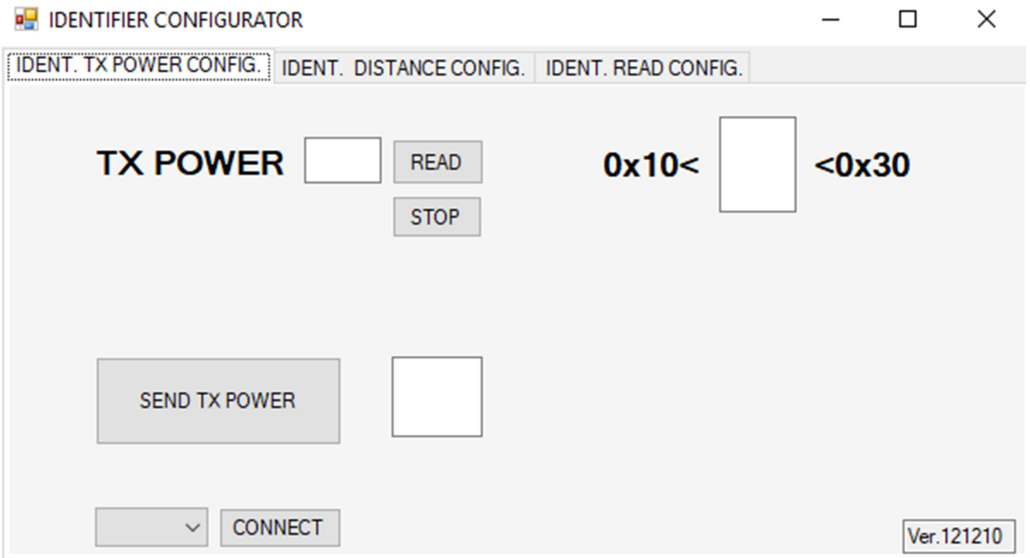
Devredeki diğer elemanlar ise L5973 entegresinden istenilen voltajı almak için kullanılan elemanlardır.

Cihazların radyo çıkış güçlerinin belli bir aralıkta olmasını sağlamak için üretim aşamasında çıkış gücü ölçümü sabit bir platforma yerleştirilerek yapılmakta bilgisayar ekranında bu değerin uygunluğu kontrol edilmektedir. Düzenek aşağıdaki resimdeki gibidir.

Kimlik biriminin kullanılacağı yere mekanik montajından sonra kimlik biriminin gereksinimleri sağlamak için bazı ayarlamalar yapılmalıdır.

Şekil 6.15. Kimlik birimi mekanik montajı

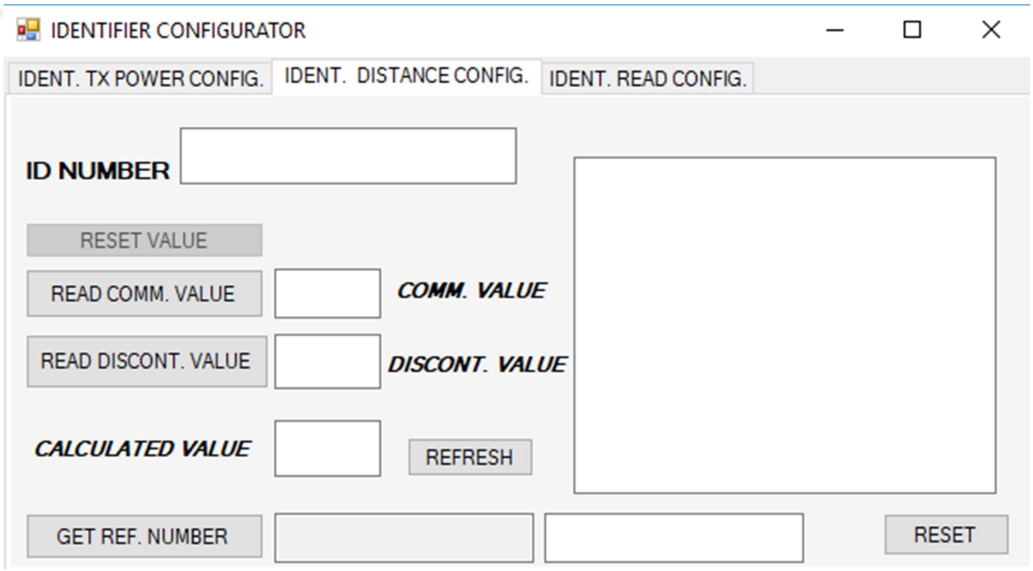
Bu ayarlamaları yapabilmesi için kendi yazmış olduğum "IDENTIFIER CONFIGURATOR" programı ve yardımcı ekipmanlar kullanılmalıdır. Bu programın genel görüntüsü aşağıdaki gibidir.



Şekil 6.16. Kimlik birimi konfigüratör programı

Bu program sayesinde belirli bir standartta üretilen kimlik birimleri montajının yapıldığı yerdeki metal yoğunluğu yansımalar v.b. şeylerden dolayı gösterdiği güç değişimleri bu program sayesinde gözlenebilir ve doğru kalibrasyon yapılabilir.

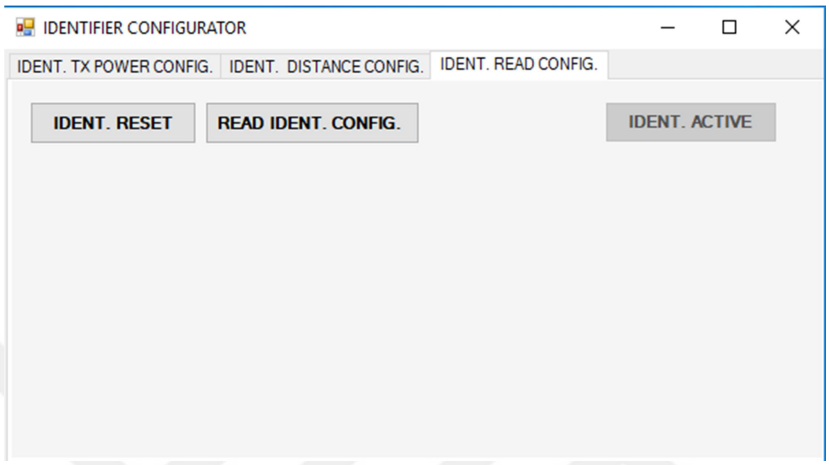
Programda ilk sekmede öncelikle programa bağlı çalışacak ekipmanın hangi port üzerinden sağlanacağı seçilir ve bağlantı sağlanır. Daha sonra bu sekme sayesinde kimlik biriminin çıkış gücü ayarlanır. Bu ayarla kimlik birimi aşırı yüksek yayın yaparak etrafa sistemi rahatsız edecek şekilde yayın yapmaz çok düşük güçte yayın yaparak da okuyucu ünite ile iletişiminde kopmalar olmasını engellemektedir. İstenilen değer 0x10 ile 0x30 arasında olmalı mümkünse 0x20 olarak ayarlanmalıdır.



Şekil 6.17. Kimlik birimi konfigüratör programı mesafe verme ekranı

Bu sekmede ise kimlik biriminin gelen sinyalin alımındaki gücü ölçülmekte ve ayarları yapılmaktadır. Öncelikle READ COMM. VALUE butonu ile tabancanın depoda normal duruşundaki değerler alınır. Daha sonra READ DISCONT. VALUE butonu ile de tabancanın depodan tam çıkma anındaki değer alınır. Ve bu iki değere göre kimlik birimi ile tabanca arasındaki sinyal gücüne bağlı olarak tabancanın uygun pozisyonda olup olmadığına dair karar mekanizması çalışır. Burada kimlik birimi okuyucu üniteye olması gereken değer ve şu an için aradaki güç farkını gönderir buna bağlı olarak okuyucu ünite kimlik biriminin kimlik numarasını gönderip göndermemeye karar verir. Bu kararın okuyucu ünite tarafından yapılmasının sebebi ise ilerinde yapılacak değişiklikler için okuyucu ünitenin akaryakıt istasyonlarda sabit olması ama kimlik birimlerinin ise araçların üzerinde sürekli ulaşmanın

mümkün olmamasıdır. Bu sekmenin en altında bulunan GET REF.NUMBER butonu sayesinde ise programın ürettiği rastgele bir sayıya karşılık gelecek anahtar sayının girilmeden kimlik biriminde müdahale engellenmiştir. Anahtar sayı firmanın internet sitesi üzerinden şifreli girişle alınabilmekte ve firma istediği takdirde o kullanıcının erişimini engelleyebilmektedir.



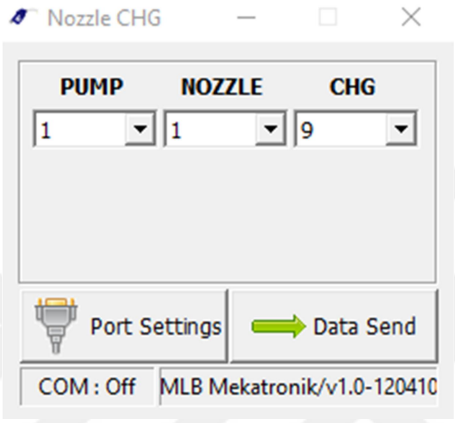
Şekil 6.18. Kimlik birimi konfigüratör programı değer okuma ekranı

Programın bu son sekmesinde ise kimlik birimin daha önce yüklenen değerleri okunabilir resetlenebilir ve kimlik birimi aktive edilerek artık kullanıma açılabilir.

6.3. Okuyucu Ünite Tasarımı

Şekil 6.19. Okuyucu ünite genel görünümü

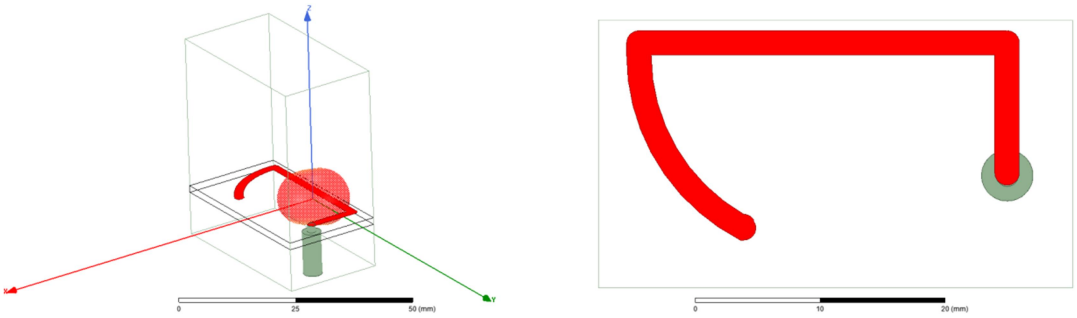
Okuyucu ünite merkezi toplayıcı birim ile kimlik birimi arasındaki iletişimi sağlar. Buna ilaveten yakıt deposunun ağzına yakın bir yere montajı yapılmış kimlik biriminden belirtilen güç eksilmesi kadar uzaklaştığında kimlik biriminin kimlik numarasını göndermeyi durdurur. Daha önce belirtildiği gibi kimlik numarasının gönderilip gönderilmemesi okuyucu ünite içerisine kayıtlı değişim miktarına göre yapılır. Bu değişim miktarı MLB mekatronik tarafından yazılan Nozzle CHG programı ve ID verici cihaz ile birlikte yapılmaktadır.



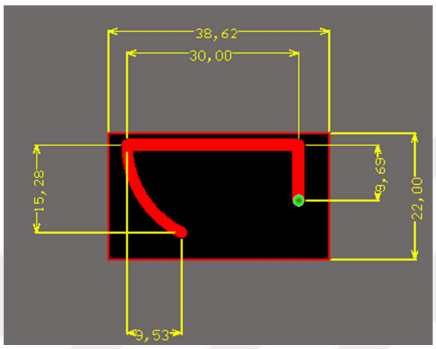
Şekil 6.20. Okuyucu ünite ayarlama programı

Bu cihaz güvenlik için merkezi toplayıcı birim ile sürekli değişen bir şifreleme anahtarı ile şifrelenmiş dataları göndererek haberleşir. Bu sayede yasadışı dolumlar engellenmiş olur.

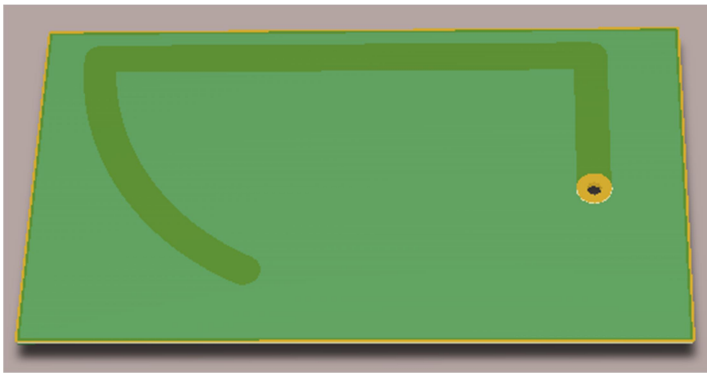
Bu cihaz da kullanılan anten aşağıdaki çizimdeki gibidir. Bu anten kimlik birimi üzerindeki monopol anten ile karşılıklı olarak geldiğinde uzaklaşmaya bağlı RSSI değerlerinde düzenli değişim gösterebilmesini sağlamış ve merkezi toplayıcı birim ile haberleşmesinde herhangi bir çekim gücü ile ilgili problem yaşanmamıştır. Anten tasarımı için birçok model denenmiş kullanılacağı yer itibari ile en iyi sonucu bu geometri vermiştir. Aşağıdaki tüm ölçüler mm olarak verilmiştir.



Şekil 6.21. Okuyucu ünite anten HFSS çizim



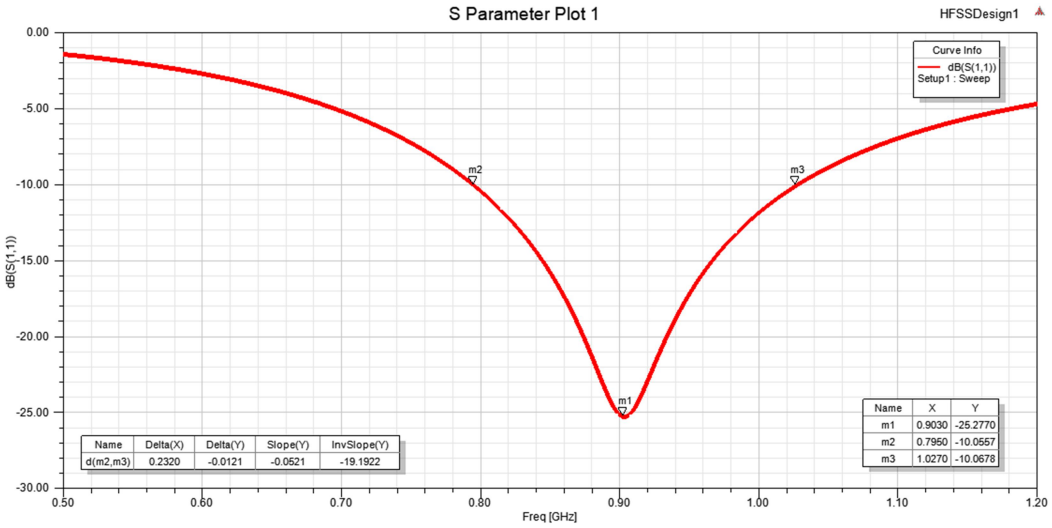
Şekil 6.22. Okuyucu ünite anten çizimi boyutlandırma



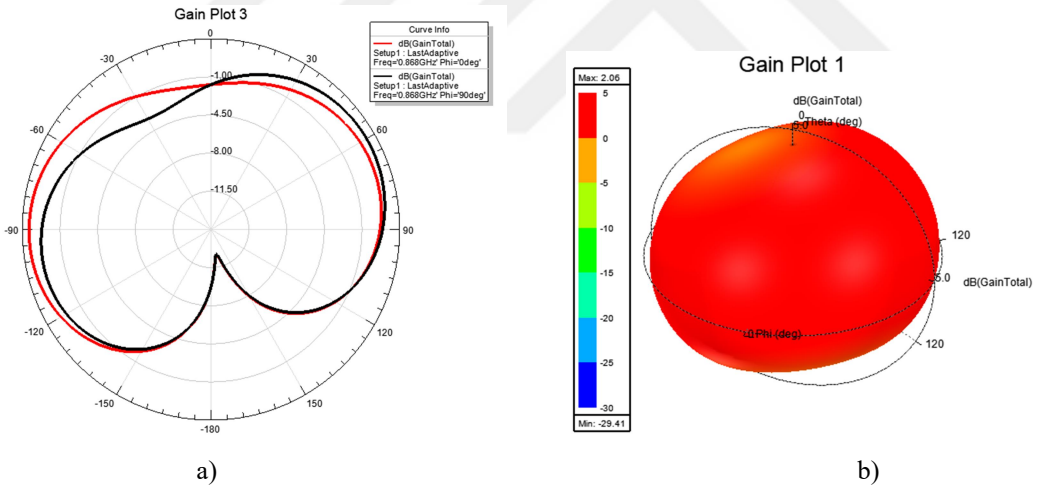
Şekil 6.23. Okuyucu ünite anten üç boyutlu görünümü

Bu anten deseni seçilirken Texas Instruments firmasının AN058 Application Note adlı dökümanında yer alan anten şekillerinden yola çıkılarak çalışacağı alan ve özelliklere uygun olacak şekilde hemen hemen her anten ve varyasyonları denenmiş, sisteme en uygun çalışan olmasından, üretim kolaylığından dolayı yukarıda ölçüleri verilen anten seçilmiştir. Bu aşamada denenen antenlerin uygun görülmeme sebebi ise radyasyon patenlerinin iki cihazın uzaklaşması veya sırasında düzgün bir değişim göstermemesine sebep olmasıdır. Tasarımı

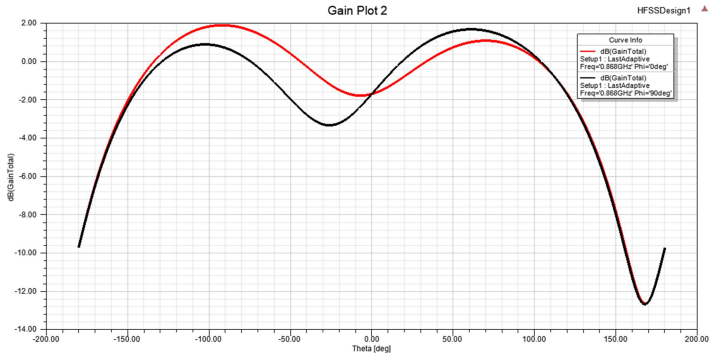
yapılan gerçekleştirilen antenin performansları sırasıyla S_{11} geri dönüş kaybı, kazancı, rezonatör direnci, duran dalga oranı ve elektrik alan performansı (Şekil 6.24, 6.25, 6.26, 6.27, 6.28) aşağıdaki şekillerde verilmiştir.



Şekil 6.24. S_{11} geri dönüş kaybı

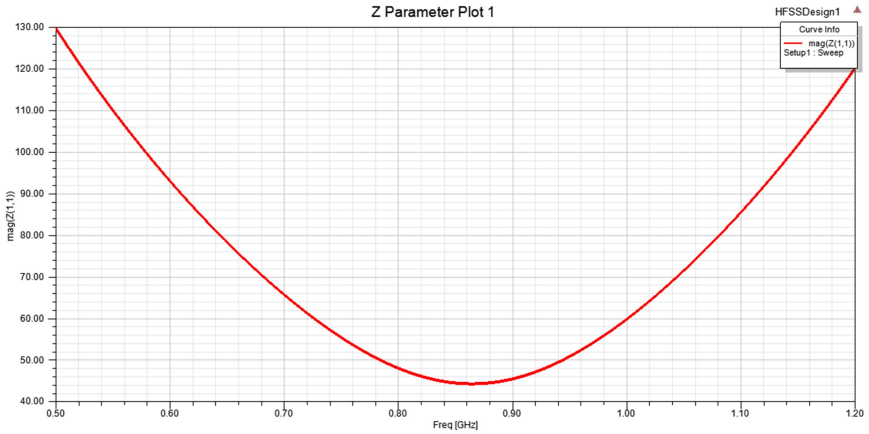


a) b)

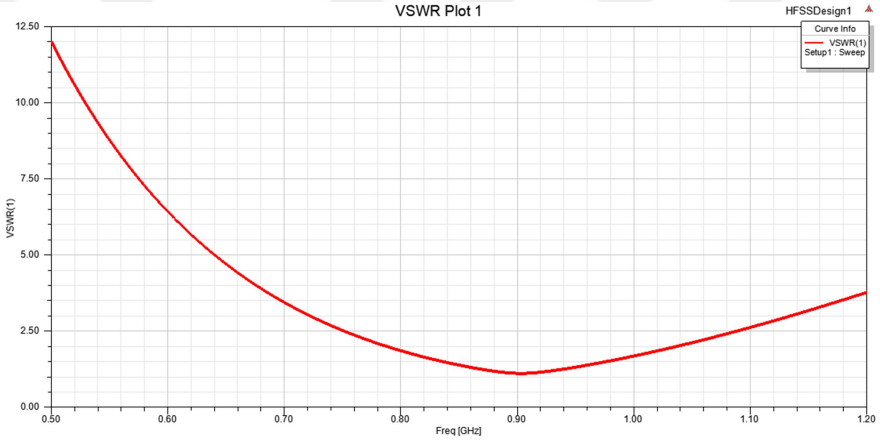


c)

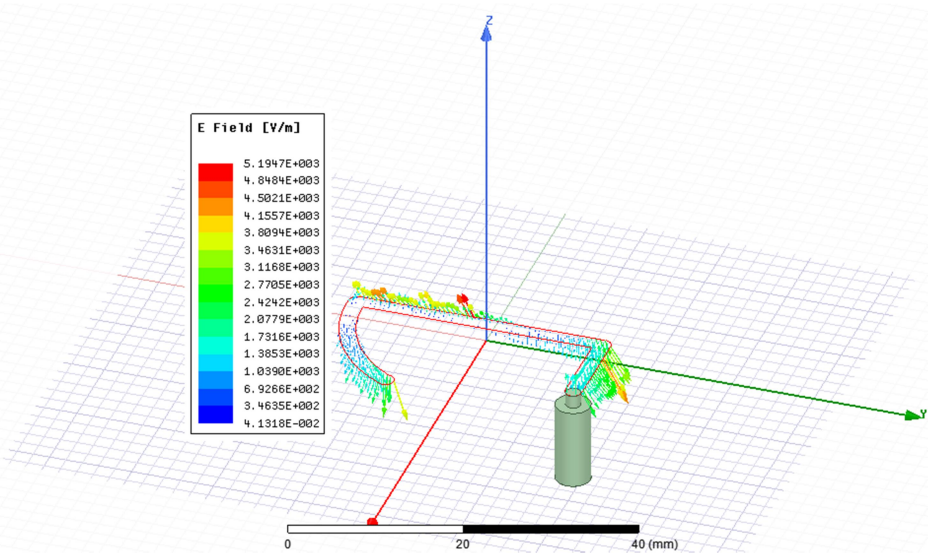
Şekil 6.25. Kazanç a) Işıma eğrisi, b) Üç boyut gösterim, c) Grafiği



Şekil 6.26. Rezonatör direnci

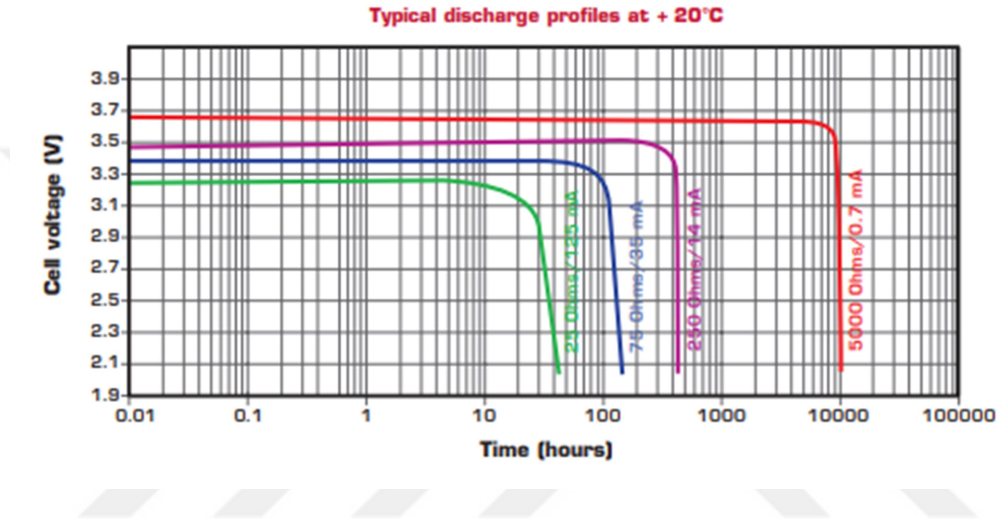


Şekil 6.27. Duran dalga oranı (VSWR)



Şekil 6.28. Duran dalga oranı (VSWR)

Ayrıca bu cihaz da patlayıcı ortamda çalışacağı için ATEX kurallarına uygun olarak tasarlanmıştır. Cihaz üretim aşamasında kıvılcımı engellemesi için reçine ile doldurulmaktadır. Ve burada 3,6 voltluk SAFT firmasına ait LS26500 kısa devre korumalı boşalma grafiği aşağıdaki gibi olan bir pil kullanılmıştır. Bu sayede cihazın pilin voltajının düşmesine bağlı olarak RF çıkış gücü değerlerinde değişme olamamıştır ve patlayıcı ortamda herhangi bir yanlışlıkla ısınma ihtimali ortadan kalkmıştır.



Şekil 6.29. Okuyucu ünite pil boşalma eğrisi

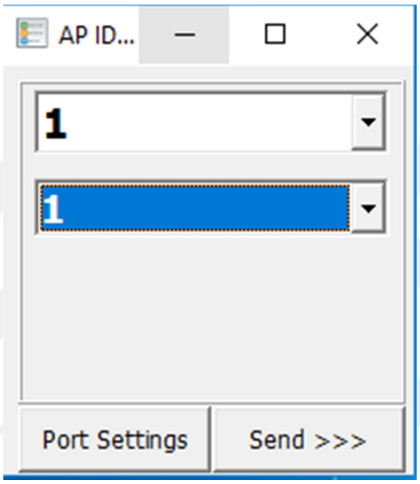
Cihazın bu şekilde çalışmasını sağlayan devre daha önce de belirtildiği gibi tasarlanan modül devresi ve bu cihaza uygun yazılımdır.

6.4. Merkezi Toplayıcı Birim

Şekil 6.30. Merkezi toplayıcı birim genel görünümü

Bu birim akaryakıt istasyonlarındaki yakıt dispanserlerin üzerine bulunur ve okuyucu birimden gelen şifreli dataları alır şifreyi çözer daha sonra otomasyon sisteminden gelen şifreleme anahtarları ile tekrar şifreli olarak kimlik biriminin kimlik numarasını gönderir ve ayrıca mifare kartları okuyup otomasyon sistemine iletir. Bu birimin montaj edildiği yer patlayıcı ortam olmadığından dolayı patlayıcılık için herhangi bir önleme ihtiyaç yoktur.

Bu cihazın içinde yine tasarlanan modül kullanılmıştır. Bu birimin çalışacağı kanal ve seri haberleşen otomasyon sistemi için adresini belirlemek için aşağıda resmi bulunan program kullanılmaktadır.



Şekil 6.31. Merkezi toplayıcı birim ayarlama ekranı

Bu birim otomasyon ile haberleşme için ise RS485 haberleşme kullanmakta ve MEPSAN firmasının STAWİZ otomasyon sisteminin protokolünü kullanmaktadır.

7.SONUÇ VE ÖNERİLER

7.1. Sonuç

Günümüzde taşımacılık artan insan nüfusu ve taşınacakların farklı yerlede olmasından dolayı devamlı büyüyen bir hizmet sektörü haline gelmiştir. Büyüme araç filolarındaki artış ile paralellik göstermektedir. Ve bu araç filolarının akaryakıt sarfiyatlarının takibi gün geçtikçe zorlaşmaktadır. Buna ek olarak akaryakıtın değerli oluşu filo sahiplerinin takip istediğini arttırmıştır.

Bu takibi sağlamak için değişik yöntemler denense de teknolojinin girmediği, insan faktörünün azalmadığı takipler sorun çıkarmakta ve filo sahiplerine beklenmedik zararlar vermektedir.

Önceleri anlaşmalı petrol istasyonlar ile çalışan firmalar daha sonra RFID kartlarla yakıt ikmali yapmaya başlamış ve şehir dışına yapılan taşımalarda bu kullandıkları sistem daha çok akaryakıt istasyonundan alışveriş yapmaya imkan vermiştir. Bu yapıda ise güvenlik ile ilgili açıklar bulunmakta ve sıkı takip yapılmaz ise kötü niyetli insanların akaryakıt hırsızlığı yapmasına olanak vermektedir. Taşıt tanıma sistemi yukarıdaki etmenlerden dolayı geliştirilmiş akaryakıtın sadece tanımlı aracın deposuna yakıt vermeyi hedeflemiştir. Yalnız bu sistemde ise araç üzerine sabitleme düzgün yapılmaz ise yine güvenlik açığı verebilmektedir. Araç üstüne sabitleyememenin ve düzgün yapılmamasının esas sebebi ise okunacak kimlik birimi ile okuyucu ünite arasındaki mesafenin kısa olmasından kaynaklanmaktadır.

Bu çalışmanın amacı yukarıda belirtilen ihtiyaçları karşılamak ve kimlik birimi ile okuyucu ünite arasındaki haberleşme mesafesinin arttırılmasıdır. Bunun için kimlik birimi enerjilendirilmiş ve okuyucu ünite ile 868 MHz frekansında haberleşme yapılarak aracın tanınması sağlanmıştır. Burada kimlik birimi üzerinde monopol anten kullanılmıştır. Ve okuyucu üniteden gelecek sinyalin uzaklaşmaya bağlı değişimi düzgün olması hedeflenmiştir. Ve kimlik biriminin sökülmesi durumunda pasif hale geçerek yakıt ikmalini engellemektedir. Şifreli olarak haberleşen kimlik birimi araya girebilecek başka bir cihaz ile kaçak yakıt ikmaline izin vermemektedir. Kimlik birimi bu özellikleri ile güvenliği sağlamakta ve daha uzak mesafelerde haberleşme yapmaktadır.

Okuyucu ünite, merkezi toplayıcı birim ile yine şifreli olarak haberleşmektedir. Bu sayede farklı bir cihazın sanki okuyucu ünite gibi davranması engellenmekte ve sistemde güvenlik açığı vermemektedir. Kablosuz yapısı sayesinde montaj kolaylığı sağlamaktadır. Okuyucu ünite kimlik birimi ve merkezi toplayıcı birim ile haberleşme PCB üzerine

tasarlanan monopol anten sayesinde yapılmaktadır. Bu antenin tasarımı yapılırken gerektiği kadar mesafeden haberleşecek ve kimlik birimi üzerindeki antenden uzaklaşırken düzgün bir güç azalması kriterlerini sağlayacak şekilde uzun denemeler sonucunda sonuçlandırılmıştır. Seçilen pilin boşalma eğrisinin yapısı buradaki devrenin akım yetersizliğinden beklenmedik hatalar vermesini engellemektedir.

Merkezi toplayı birim ise bilginin otomasyon sistemine aktarılması ile görevli olup otomasyon sistemi ile RS485 üzerinden uygun protokolle haberleşmektedir. Bu haberleşme yine şifreli olmakta ve başka cihazların merkezi toplayıcı birim gibi gösterilmesi engellenmektedir. Merkezi toplayıcı birim üzerinde hazır bir anten kullanılmış ve denemeler sonucu sisteme uygunluğu görülmüştür.

Genel olarak tasarlanan cihazların kullanım amacını karşılamakta ve günümüzdeki diğer sistemlerden daha güvenli olduğu görülmüştür.

7.2. Öneriler

Bu teze konu olan taşıt tanıma sistemi bileşenlerinden kimlik birimi pil ömrü uzun olacak şekilde pilli olarak tasarlanması düşünülebilir. Yalnız bu durumda pil durumun kontrolüne önem verilmesi gerekecektir. Bunun haricinde gelişen teknoloji ve maliyetlerin düşmesi ile kimlik birimi 868 MHz pasif, yerinden söküldüğünde zarar görecek şekilde RFID devrelere dönüşebilir. Bu dönüşüm ile tabanca üzerindeki okuyucu ünite içerisine 868 MHz RFID okuyucu modül veya okuyucu entegre devreler eklenmesi ve anten tasarımı daha yüksek güç yayacak şekle değiştirilmesi gerekecektir. Daha fazla güç kullanılacak olur ise de bu sisteme şarjlı pil yapısı eklenmeli ve burada sökülüp takılacak bu kısmın patlayıcı ortam gereksinimleri sağlanmalıdır.

Yine gelişen teknoloji sayesinde aşırı düşük güç tüketimli mikro denetleyicilerin işlem hızlarının artması ile şifreleme algoritmaları daha karmaşık hale getirilebilir.

8. KAYNAKLAR

United States Patent,1997, Vehicle Identification System For a Feul Dispenser,1.

EM4095 Data Sheet,2005 Rev F., Read Only Contactless Identification Device,2.

Özbayoğlu, A.M. ve Kök, İ.Ş., 2011, Sinyal gücüne bağlı bina içi konum tespiti modelleri karşılaştırılması, Elektrik-Elektronik ve Bilgisayar Sempozyumu 2011, Elâzığ, 99-104.

Bekçiaşı, U. ve Tenruh.M, 2012, Kablosuz Algılayıcı Ağlarda Konum Saptama Teknikleri ve Mesafe Bağımlı Tekniklerde Dördüncü Çapa Yaklaşımı, Akademik Bilişim'12- XIV. Akademik Bilişim Konferansı Bildirileri, Uşak,140-146.

Günay, F.B. ve Çavdar, T. Kablosuz Duyarga Ağlarda RSSI, TOA ve TDOA yardımıyla Gezgin Filo Lokalizasyon Modeli, 2014 IEEE 22nd Signal Processing and Comunications Applications Conference (SIU 2014), Trabzon,1431,1434.

Low, M. Ve Sylla, I.,2005, ISM-Band and Short Range Device Antennas, Texas Instrments Incorporated, Dallas.

Wallace, R.,2010, Antenna Selection Guide adlı Application Note AN058, Texas Instrments Incorporated, Dallas.

Wallace, R.,2013, Monopole PCB Antenna with Single or Dual Band Option adlı Design Note DN024, Texas Instruments Incorporate, Dallas.

Nordic Semiconductor, 2005, λ/4 printed monopole antenna for 868/915MHz, Norveç.

Suyabatmaz, B.B.,2006, Kablosuz veri iletimi için alıcı-verici geliştirme kartı tasarımı ve örnek bir uygulama, Yüksek Lisans Tezi, Gebze.

Başçifci, N., 2011 Zigbee Tabanlı Mobil Sağlık İzleme Sistem Tasarımı ve uygulaması, Yüksek Lisans Tezi, Konya.

Texas Instruments, 2009, CC430 Family User's Guide SLAU259E, Texas Instruments Incorporated, Dallas.

Texas Instruments, 2009, MSP430™ SoC With RF Core SLAS554H, Texas Instruments Incorporated, Dallas.

Wikipedia, 2011, Ultra_high_frequency, [online], https ://en.wikipedia.org /wiki/ Ultra_high_frequency [Ziyaret Tarihi: 2011].

Özdemir A., Yazıcı İ., Kunduz T., 2005, Mikrodenetleyici Tabanlı Kablosuz Kontrol ve Kumanda Sistemi Tasarımı, III. Otomasyon Sempozyumu, Denizli.

Yarıkkaş B.,2009, Cerrahi operasyonlar sonrası hasta vücudunda unutulan cerrahi malzemelerin tespitine yönelik cihaz tasarımı ve imalatı, Yüksek lisan tezi, Ankara.

Kabalcı, E., 2006, PC tabanlı kablosuz EKG biyotelemetri sistemi tasarımı ve yapımı, Yüksek Lisans Tezi, Ankara.

Türkyılmaz A., 2010, Su Sayacı Tercihinde Dikkat Edilecek Hususlar, Türkiye Cumhuriyeti Bilim, Sanayi ve Teknoloji Bakanlığı Metroloji ve Standardizasyon Genel Müdürlüğü, Ankara.

Sarı K., [online], http://www.taskomuru.gov.tr/file/ALSz/ExproofGuide_ Kemal%20SARI.pdf [Ziyaret Tarihi: 09.Haziran 2019].

Elektroport, Antenler ve Çeşitleri [online], https://www.elektrikport.com/makale-detay/antenler-ve-cesitleri/16763#ad-image-0 [Ziyaret Tarihi: 05. Mayıs 2019].

Yıldız, S., 2011, Bir mikrodalga antenin ışıma ana huzmesinin ayarlı bir toprak düzlemi ile yönlendirilmesi, Yüksek Lisans Tezi, İstanbul.

Udea, Uygulama Notu: UN-0506v01_ Şifreleme, [online] https://www.udea.com.tr/ home_assets/documents/UN-0506v01 Sifreleme.pdf [Ziyaret Tarihi: 02. Mayıs 2019].

Wallace, R., Dunbar S., 2010, 2.4 GHz YAGI PCB Antenna Application Note DN034, Texas Instruments Incorporated, Dallas.

Wallace, R., 2010, Antenna Kit Board 6 868_2440_MHz DN603, Texas Instruments Incorporated, Dallas.

Wallace, R., 2010, Antenna Kit Board 10 868_915_MHz DN606, Texas Instruments Incorporated, Dallas.

Wallace, R., 2010, Antenna Kit Board 5 868_915_MHz DN602, Texas Instruments Incorporated, Dallas.

EKLER

EK-1 Kimlik Birimi Programı

EK-1.1. main.c

```
#include"D:\ccs projects\UHFVISTAG_1\UHFVISTAG_1.h"
#include "D:\ccs projects\cc430f6137.h"
#include "D:\ccs projects\include\string.h"
#include "D:\ccs projects\include\stdio.h"

void main (void)
{

 WDTCTL = WDTPW + WDTHOLD;
 for(main_k=0;main_k<0xdff00;main_k++)
 {
        main_k++;
 }


 if(!(P1IN & BIT4))
 {
        __bis_SR_register(  GIE );

        P1DIR &= ~BIT4;
        P1REN &=~BIT4;
        P1IES &=~ BIT4;
        P1IFG = 0;
        P1OUT |= BIT4;
        P1IE  |= BIT4;
        on_bat=1;
        P1DIR &= ~BIT7;
        P1REN |=BIT7;
        P1IES &= ~ BIT7;
        P1IFG = 0;
        P1OUT |= BIT7;
        P1IE  |= BIT7;
        if(radio_state!=RF_SXOFF)
        {
                Strobe(RF_SIDLE);
                Strobe(RF_SXOFF);
        }
        radio_state=RF_SXOFF;
        TA1CCTL0 = ~CCIE;
        __bis_SR_register(LPM3_bits );
        on_bat=0;
 }



 read_MY_SEG();

 temp_crc_value=0;temp_crc_value1=0;
        temp_crc_value=calcrc(MY_ID,22,0);
        temp_crc_value1=  temp_crc_value;
        temp_crc_value&=0x000000ff;
        temp_crc_value1>>=8;
        temp_crc_value1&=0x000000ff;
        if(MY_ID[22]==  temp_crc_value1  &&  MY_ID[23]==
temp_crc_value)
        {
                RX_POWER=MY_ID[18];
                RX_POWER_MAX=MY_ID[20];
                if(RX_POWER_MAX > RX_POWER)
                {

TEMP_RX_POWER=RX_POWER_MAX;

TEMP_RX_POWER_MAX=RX_POWER;
                                WRONG_POWER=1;

                }
                read_MY_SEGD();
                temp_crc_value=0;temp_crc_value1=0;
                temp_crc_value=calcrc(MY_ID,22,0);
                temp_crc_value1=  temp_crc_value;
                temp_crc_value&=0x000000ff;
                temp_crc_value1>>=8;
                temp_crc_value1&=0x000000ff;
                if(MY_ID[22]==      temp_crc_value1      &&
MY_ID[23]== temp_crc_value)
                {
                        if(MY_ID[24]!=0x55              ||
MY_ID[25]!=0x11)
                        {
                                copy_C2D();
                                read_MY_SEG();
                        }
                }
                else
                {
                        copy_C2D();
                        read_MY_SEG();
                }
        }
        else
        {
                read_MY_SEGD();
                temp_crc_value=0;temp_crc_value1=0;
                temp_crc_value=calcrc(MY_ID,22,0);
                temp_crc_value1=  temp_crc_value;
                temp_crc_value&=0x000000ff;
                temp_crc_value1>>=8;
                temp_crc_value1&=0x000000ff;
                if(MY_ID[22]==      temp_crc_value1      &&
MY_ID[23]== temp_crc_value)
                {
                        RX_POWER=MY_ID[18];
                        RX_POWER_MAX=MY_ID[20];
                        if(RX_POWER_MAX              >
RX_POWER)
                        {

TEMP_RX_POWER=RX_POWER_MAX;

TEMP_RX_POWER_MAX=RX_POWER;
                                WRONG_POWER=1;
                        }
                        copy_D2C();
                }
        }


        read_MY_SEG();


        if(MY_ID[26]==(0xff-MY_ID[27]))
        {
                TX_POWER=MY_ID[26];
                read_MY_SEGD();
                if(!(MY_ID[26]==(0xff-MY_ID[27])))
                {
                        copy_C2D();
                }
                read_MY_SEG();
        }
        else
        {
                 read_MY_SEGD();
                 if(MY_ID[26]==(0xff-MY_ID[27]))
                 {
                        TX_POWER=MY_ID[26];
                        copy_D2C();
                 }
                 else      if      (MY_ID[26]==0xff      &&
MY_ID[27]==0xff)
                 {
                        TX_POWER=0x40;
                 }
                 else
                 {
                        TX_POWER=0x40;
                 }

        }

        read_MY_SEG();


        rfSettings.channr=0;
        rfSettings.mdmcfg4=0xca;
        TX_POWER_1=TX_POWER;
        RX_POWER1_=RX_POWER;
        WRONG_POWER=0;
        SetVCore(2);
```

```
        ResetRadioCore();
        InitRadio();
        InitButtonLeds();
        buttonPressed=0;
        read_MY_SEG();
        if(MY_ID[24]!=0x55 )
        {
                if(MY_ID[25]==0x11)
                {
                        read_MY_SEG();
                        MY_ID[24]=0x55;
                        write_MY_SEG();
                        {WDTCTL                 =
WDTPW+WDTCNTCL+WDTIS2;}
                        while(1);

                }
                else
                {
                        read_MY_SEGD();
                        if(MY_ID[24]==0x55 )
                          {

if(MY_ID[25]==0x11)
                                        {

read_MY_SEG();

MY_ID[24]=0x55;

write_MY_SEG();

{WDTCTL = WDTPW+WDTCNTCL+WDTIS2;}
                                                        while(1);

                                        }
                          }
                        Actived_failed=1;
                        rfSettings.channr=0;
                        SetVCore(2);
                        ResetRadioCore();
                        InitRadio();


        }
        else
        {
                if(MY_ID[25]!=0x11)
                {
                        read_MY_SEG();
                        MY_ID[25]=0x11;
                        write_MY_SEG();
                        {WDTCTL                 =
WDTPW+WDTCNTCL+WDTIS2;}
                        while(1);

                }
                if((P1IN & BIT7) )
                {
                        buttonPressed = 1;
                        Enable_Start_Timer(100);
                        read_MY_SEG();
                        MY_ID[24]=0;
                        MY_ID[25]=0;
                        write_MY_SEG();
                        copy_C2D();
                        read_MY_SEG();
                        Actived_failed=1;
                }
        }
        for(main_i=0;main_i<18;main_i++)
        {
                ACKMSG[6+main_i]=MY_ID[main_i];
        }
        for(main_i=0;main_i<0xff;main_i++)
        {
                RxBuffer[main_i]=0;
        }
        ReceiveOn();
        receiving = 1;
        transmitting=0;
        radio_state=RF_SRX;
        while (1)
        {

                __bis_SR_register(  GIE );
                if(radio_state== RF_SRX)
                {
                        if(!(RF1AIES&BIT9)              &&
!(RF1AIE&BIT9))
                        {
                                RF1AIES |= BIT9;
                                RF1AIFG &= ~BIT9;
                                RF1AIE  |= BIT9;
                        }
                }

                if(         (RX_POWER1_!=RX_POWER)          ||
MY_CHN!=0    ||    second_talk>1    ||    first_talk>1    ||transmitting>1    ||
receiving>1 || (TX_POWER!=TX_POWER_1) )
                {
                        {WDTCTL                 =
WDTPW+WDTCNTCL+WDTIS2;}
                        while(1);
                }

                if((P1IN & BIT7) && !Actived_failed)
                {
                        active_fail_count++;
                        if(active_fail_count>0xdff0)
                        {
                                Actived_failed=1;
                                active_fail_count=0;
                                timer_over=1;
                        }
                        else
                        {
                                Actived_failed=0;

                        }
                }
                if(!(P1IN & BIT7))
                {
                        active_fail_count=0;
                }


                if (buttonPressed && !transmitting)
                {

                        P1IFG = 0;
                        buttonPressed=0;
                }

                else  if(   wait_for_clear   ||   (timer_over   &&
!transmitting))
                {
                        wait_for_clear=0;
                        read_MY_SEG();

                        MY_ID[24]=0x00;
                        MY_ID[25]=0x00;
                        write_MY_SEG();
                        copy_C2D();
                        Actived_failed=1;
                        P1IFG = 0;
                        P1IE=1;
                        TA1CCTL0 |= ~CCIE;
                        timer_over=0;
                        InitButtonLeds();

                }
                else if(Message_Waiting && !transmitting)
                {
                        if(radio_state!=RF_SFRX)
                        {
                                ReceiveOff();
                                transmitting=1;
                                receiving=0;
                                receiving=0;

radio_state=RF_SFRX;
                        }
                        if(first_talk)
                        {
                                Message_Waiting=0;
                          if(Actived_failed)
                                {
```

```
for(main_i=0;main_i<2;main_i++)
                                        {
        ACKMSG[main_i+6]=0xEE;
                                        }
                                }

Transmit_TAG(ACKMSG,25);
                                transmitting=1;
                                P3OUT |= BIT6;
                                P3OUT |= BIT4;
                        }
                        else if(second_talk)
                        {
                                Message_Waiting=0;

                                if(Actived_failed)
                                {
        for(main_i=0;main_i<2;main_i++)
                                        {
        ACKMSG[main_i+6]=0xEE;
                                        }
                                }

Transmit_TAG(ACKMSG,25);
        transmitting=1;
                                        P3OUT   |=
BIT6;
                                        P3OUT   |=
BIT4;
                        }
                        else if(Power_TEST)
                        {
                                Message_Waiting=0;
                                Power_TEST=0;
                                {

POWERMSG[6]=(char)RX_POWER;

POWERMSG[7]=(char)RX_POWER_MAX;

Transmit(POWERMSG,25);

transmitting=1;
                                                P3OUT |=
BIT6;
                                                P3OUT |=
BIT4;
                                }
                        }
                        else if(Send_READCNFG_MSG)
                        {
                                Message_Waiting=0;

Send_READCNFG_MSG=0;
                                read_MY_SEG();

for(main_i=0;main_i<8;main_i++)
                                {

READCNFG_MSG[main_i+6]=MY_ID[main_i];
                                }

READCNFG_MSG[14]=MY_ID[18];

READCNFG_MSG[15]=MY_ID[19];

READCNFG_MSG[16]=MY_ID[26];

READCNFG_MSG[17]=MY_ID[27];

READCNFG_MSG[18]=MY_ID[24];

READCNFG_MSG[19]=MY_ID[25];

READCNFG_MSG[20]=MY_ID[20];

READCNFG_MSG[21]=MY_ID[21];

Transmit(READCNFG_MSG,25);
                                transmitting=1;

                                P3OUT |= BIT6;
                                P3OUT |= BIT4;

                        }
                        else if(Reinit_radio)
                        {
                                Message_Waiting=0;
                                Reinit_radio=0;
                                {

POWERMSG[6]=(char)RX_POWER;

POWERMSG[7]=(char)RX_POWER_MAX;

POWERMSG[8]=TX_POWER;

Transmit(POWERMSG,25);

transmitting=1;
                                                P3OUT |=
BIT6;
                                                P3OUT |=
BIT4;
                                }
                        }
                        else if(Actived)
                        {
                                Message_Waiting=0;
                                Actived=0;
                                actived_reg=0x55;

                                if(MY_ID[24]!=0x55
&& MY_ID[25]!=0x11)
                                {

ACTMSG[4]=ACTMSG[4]+1;

ACTMSG[6]=0x55;
                                read_MY_SEG();

                                if(MY_ID[18]==0xff)
                                {

MY_ID[24]=0x55;

MY_ID[25]=0x11;

Actived_failed=0;

                                }
                                MY_ID[18]=((char
)RX_POWER);

MY_ID[19]=((char)~RX_POWER);
                                MY_ID[20]=((char
)RX_POWER_MAX);

MY_ID[21]=((char)~RX_POWER_MAX);


MY_ID[26]=TX_POWER;
                                MY_ID[27]=0xff-
TX_POWER;

                                write_MY_SEG();
                                copy_C2D();
                                read_MY_SEG();
                        }
```

```
Transmit_TAG(ACTMSG,25);
transmitting=1;
                                        P3OUT |=
BIT6;
                                        P3OUT |=
BIT4;
                                }
                                else
                                {
ACTMSG[4]=ACTMSG[4]+1;
ACTMSG[6]=0x55;
Transmit_TAG(ACTMSG,25);
transmitting=1;
                                        P3OUT |=
BIT6;
                                        P3OUT |=
BIT4;
                               }
                                read_MY_SEG();
                                MY_ID[24]=0x55;
                                MY_ID[25]=0x11;
                                write_MY_SEG();
                                copy_C2D();
                                read_MY_SEG();
                                Actived_failed=0;
for(main_i=0;main_i<18;main_i++)
                                {
ACKMSG[6+main_i]=MY_ID[main_i];
                                }
                        }
                        else if(reset_KB)
                        {
                                reset_KB=0;
                                Message_Waiting=0;
Transmit(RESETMSG,25);
for(main_k=0;main_k<0xf00;main_k++)
                                {
main_k++;
                                }
                                {WDTCTL          =
WDTPW+WDTCNTCL+WDTIS2;}
                                while(1);
                        }
                        else if(id_change)
                        {
                                RX_POWER=0xff;
RX_POWER_MAX=0xff;
MY_ID[18]=(char)RX_POWER;
MY_ID[19]=(char)~RX_POWER;
MY_ID[20]=(char)RX_POWER_MAX;
MY_ID[21]=(char)~RX_POWER_MAX;
                                MY_ID[24]=0xff;
                                write_MY_SEG();
                                copy_C2D();
                                Message_Waiting=0;
                                id_change=0;
for(main_i=0;main_i<17;main_i++)
                                {
TAGIDMSG[6+main_i]=MY_ID[main_i];
                                }
                                Message_Waiting=0;
Transmit(TAGIDMSG,25);
                                transmitting=1;
                                P3OUT |= BIT6;
                                P3OUT |= BIT4;
                                can_sleep=0;
                                //write_MY_ID();
                        }
                        else if(plate_change)
                        {
MY_ID[18]=(char)RX_POWER;
MY_ID[19]=(char)~RX_POWER;
MY_ID[20]=(char)RX_POWER_MAX;
MY_ID[21]=(char)~RX_POWER_MAX;
                                MY_ID[24]=0xff;
                                write_MY_SEG();
                                copy_C2D();
for(main_i=0;main_i<17;main_i++)
                                {
TAGIDMSG[6+main_i]=MY_ID[main_i];
                                }
                                Message_Waiting=0;
Transmit(TAGIDMSG,25);
                                transmitting=1;
                                P3OUT |= BIT6;
                                P3OUT |= BIT4;
                                can_sleep=0;
                                plate_changed=0;
                        }
                        else
                        {
                                Message_Waiting=0;
if(radio_state!=RF_SRX)
                                {
ReceiveOn();
receiving=1;
radio_state=RF_SRX;
                                }
                                can_sleep=1;
                        }
                }
                else if(!transmitting)
                {
                        if(radio_state!=RF_SRX)
                        {
                                ReceiveOn();
                                receiving = 1;
                                radio_state=RF_SRX;
                        }
                        can_sleep=1;
                }
        }
}

////////////////////////////////////////////////////////////////////////////
//                          INTERRUPTS                                    //
////////////////////////////////////////////////////////////////////////////
#pragma vector=CC1101_VECTOR

__interrupt void CC1101_ISR(void)

{

  switch(__even_in_range(RF1AIV,32))
```

```
{
 case  0: break;
 case  2: break;
 case  4: break;
 case  6: break;
 case  8: break;
 case 10: break;
 case 12: break;
 case 14: break;
 case 16: break;
 case 18: break;
 case 20:
  if(receiving )
  {
   RxBufferLength = ReadSingleReg( RXBYTES );
                ReadBurstReg(RF_RXFIFORD,
RxBuffer,RxBufferLength);

                if(RxBuffer[26]<128 )
                {
                        RSSI_DEC=(RxBuffer[26]/2)-74;
                }
                else
                {
                        RSSI_DEC=RxBuffer[26];
                        RSSI_DEC=(RSSI_DEC-256)/2-74;
                }
                RxBufferLength=0x1c;


        make_for_Crypto_buf(RxBuffer,RxBufferLength-4);
                Decode(Crypto_BUF,25/4);
                make_for_buf(RxBuffer,(RxBufferLength-4));

                if(RxBuffer[5]=='S')
              {
                temp_crc_value=calcrc(RxBuffer,(RxBufferLength-3)-
3,0);
                temp_crc_value1=  temp_crc_value;
                temp_crc_value&=0x000000ff;
                temp_crc_value1>>=8;
                temp_crc_value1&=0x000000ff;
                if((RxBuffer[(RxBufferLength-3)-
1]==temp_crc_value)     &&     (RxBuffer[(RxBufferLength-3)-2]     ==
temp_crc_value1) &&( RxBuffer[2]==0x30))
                {
                      P1OUT |= BIT0;
                      __no_operation();
                   for(int_rec_i=0;int_rec_i<18;int_rec_i++)
                   {
                      temp_id[int_rec_i]=RxBuffer[6+int_rec_i];
                   }
                   make_buf_for_key(RxBuffer);
                   Generate_Delta();

                   switch(RxBuffer[3])
                   {
                   case 0x10:
                   {
                        if(!WRONG_POWER)
                        {
                                Message_Waiting=1;
                                first_talk=1;
                                second_talk=0;
                                ACKMSG[1]=RxBuffer[1];

if(RxBuffer[26]>RX_POWER)ACKMSG[4]=RxBuffer[26]                 -
RX_POWER;
                                else  {ACKMSG[4]=RX_POWER
- RxBuffer[26];}



                        }
                        else
                        {

                                {
                                        Message_Waiting=1;
                                        first_talk=1;
                                        second_talk=0;
ACKMSG[1]=RxBuffer[1];

if(RxBuffer[26]>RX_POWER)

ACKMSG[4]=RxBuffer[26] - RX_POWER;
                                        else
{ACKMSG[4]=RX_POWER - RxBuffer[26];}

                                }
                        }
                        break;
                   }
                   case 0x11:
                   {

                        {
                                if(!strncmp(temp_id,MY_ID,18))
                                {
                                        Message_Waiting=1;
                                        first_talk=0;
                                        second_talk=1;

ACKMSG[1]=RxBuffer[1];

if(ACKMSG[4]>250)ACKMSG[4]=0x31;
                                        else ACKMSG[4]++;
                                }
                        }
                   }
                   case 0x20:
                   {

                        if(RX_POWER_COUNTER>14          &&
(!strncmp(temp_id,MY_ID,8)) )
                        {

for(int_rec_i=0;int_rec_i<13;int_rec_i++)
                                {

av_RSSI_DEC_=RSSI_DEC_BUF[0+int_rec_i];

av_RSSI_DEC_1=RSSI_DEC_BUF[1+int_rec_i];

av_RSSI_DEC_2=RSSI_DEC_BUF[2+int_rec_i];


if(av_RSSI_DEC_<av_RSSI_DEC_1)
                                        {

RSSI_DEC_BUF[0+int_rec_i]=av_RSSI_DEC_;

RSSI_DEC_BUF[1+int_rec_i]=av_RSSI_DEC_1;

RSSI_DEC_BUF[2+int_rec_i]=av_RSSI_DEC_2;
                                        }
                                        else
                                        {

RSSI_DEC_BUF[0+int_rec_i]=av_RSSI_DEC_1;

RSSI_DEC_BUF[1+int_rec_i]=av_RSSI_DEC_;

RSSI_DEC_BUF[2+int_rec_i]=av_RSSI_DEC_2;
                                        }

if(av_RSSI_DEC_2<RSSI_DEC_BUF[0+int_rec_i])
                                        {

RSSI_DEC_BUF[0+int_rec_i]=av_RSSI_DEC_2;

RSSI_DEC_BUF[1+int_rec_i]=av_RSSI_DEC_;

RSSI_DEC_BUF[2+int_rec_i]=av_RSSI_DEC_1;
                                        }


av_RSSI_DEC_1=RSSI_DEC_BUF[1+int_rec_i];

av_RSSI_DEC_2=RSSI_DEC_BUF[2+int_rec_i];


if(av_RSSI_DEC_1<av_RSSI_DEC_2)
```

```
                                        {
RSSI_DEC_BUF[1+int_rec_i]=av_RSSI_DEC_1;
RSSI_DEC_BUF[2+int_rec_i]=av_RSSI_DEC_2;
                                        }
                                        else
                                        {
RSSI_DEC_BUF[1+int_rec_i]=av_RSSI_DEC_2;
RSSI_DEC_BUF[2+int_rec_i]=av_RSSI_DEC_1;
                                        }
                                        int_rec_i++;
                                        int_rec_i++;
                                }

RX_POWER=(RSSI_DEC_BUF[1]+RSSI_DEC_BUF[4]+RSSI_DEC_B
UF[7]+RSSI_DEC_BUF[10]+RSSI_DEC_BUF[13])/5;
                                RX_POWER1_=RX_POWER;
                                RSSI_DEC_=0;
                                RX_POWER_COUNTER=0;
                                Message_Waiting=1;
                        Reinit_radio=1;
                        first_talk=0;
                        second_talk=0;

                        can_sleep=0;
                        }
                        else
                        {

                                RX_POWER_COUNTER=0;
                                Message_Waiting=0;
                                Strobe(RF_SRX);
                                receiving=1;
                                transmitting=0;
                                radio_state=RF_SRX;
                        }
                        break;
                }
                case 0x80:
                {
                        RX_POWER=RxBuffer[6];
                        RX_POWER1_=RX_POWER;
                        RSSI_DEC_=0;
                        RX_POWER_COUNTER=0;
                        Message_Waiting=1;
                        Reinit_radio=1;
                        first_talk=0;
                        second_talk=0;
                        can_sleep=0;
                        break;
                }
                case 0x81:
                {
                        Message_Waiting=1;
                        TX_POWER=RxBuffer[6];
                        TX_POWER_1=TX_POWER;
                        RSSI_DEC_=0;
                        RX_POWER_COUNTER=0;
                        Reinit_radio=1;
                        first_talk=0;
                        second_talk=0;
                        can_sleep=0;

                        break;
                }

                case 0x82:
                {
                        Message_Waiting=1;
                        Send_READCNFG_MSG=1;
                        first_talk=0;
                        second_talk=0;
                        can_sleep=0;

                        break;
                }


                case 0x21:
                {

                        if(RX_POWER_COUNTER<30          &&
(!strncmp(temp_id,MY_ID,8)))
                        {
RSSI_DEC_BUF[RX_POWER_COUNTER]=RxBuffer[26];
                                RX_POWER_COUNTER++;

                        }

                        Strobe(RF_SRX);
                        radio_state=RF_SRX;
                        transmitting=0;
                        receiving=1;

                        break;
                }
                case 0x50:
                {
                        Message_Waiting=1;
                        if(RX_POWER_COUNTER>14)
                        {

for(int_rec_i=0;int_rec_i<13;int_rec_i++)
                                        {
av_RSSI_DEC_=RSSI_DEC_BUF[0+int_rec_i];
av_RSSI_DEC_1=RSSI_DEC_BUF[1+int_rec_i];
av_RSSI_DEC_2=RSSI_DEC_BUF[2+int_rec_i];

if(av_RSSI_DEC_<av_RSSI_DEC_1)
                                        {
RSSI_DEC_BUF[0+int_rec_i]=av_RSSI_DEC_;
RSSI_DEC_BUF[1+int_rec_i]=av_RSSI_DEC_1;
RSSI_DEC_BUF[2+int_rec_i]=av_RSSI_DEC_2;
                                        }
                                        else
                                        {
RSSI_DEC_BUF[0+int_rec_i]=av_RSSI_DEC_1;
RSSI_DEC_BUF[1+int_rec_i]=av_RSSI_DEC_;
RSSI_DEC_BUF[2+int_rec_i]=av_RSSI_DEC_2;
                                        }

if(av_RSSI_DEC_2<RSSI_DEC_BUF[0+int_rec_i])
                                        {
RSSI_DEC_BUF[0+int_rec_i]=av_RSSI_DEC_2;
RSSI_DEC_BUF[1+int_rec_i]=av_RSSI_DEC_;
RSSI_DEC_BUF[2+int_rec_i]=av_RSSI_DEC_1;
                                        }

av_RSSI_DEC_1=RSSI_DEC_BUF[1+int_rec_i];
av_RSSI_DEC_2=RSSI_DEC_BUF[2+int_rec_i];

if(av_RSSI_DEC_1<av_RSSI_DEC_2)
                                        {
RSSI_DEC_BUF[1+int_rec_i]=av_RSSI_DEC_1;
RSSI_DEC_BUF[2+int_rec_i]=av_RSSI_DEC_2;
                                        }
                                        else
                                        {
RSSI_DEC_BUF[1+int_rec_i]=av_RSSI_DEC_2;
```

```
RSSI_DEC_BUF[2+int_rec_i]=av_RSSI_DEC_1;
                                        }
                                        int_rec_i++;
                                        int_rec_i++;
                                }


RX_POWER_MAX=(RSSI_DEC_BUF[1]+RSSI_DEC_BUF[4]+RSSI_
DEC_BUF[7]+RSSI_DEC_BUF[10]+RSSI_DEC_BUF[13])/5;
                                RSSI_DEC_=0;
                                RX_POWER_COUNTER=0;
                                Reinit_radio=1;
                                first_talk=0;
                                second_talk=0;
                                can_sleep=0;
                        }


                        Strobe(RF_SRX);
                        receiving=1;
                        transmitting=0;
                        radio_state=RF_SRX;

                        break;
                }
                case 0x51:
                {


RSSI_DEC_BUF[RX_POWER_COUNTER]=RSSI_DEC;

                        RX_POWER_COUNTER++;
                        RSSI_DEC_+=RSSI_DEC;
                        Message_Waiting=0;
                        Strobe(RF_SRX);
                        radio_state=RF_SRX;
                        transmitting=0;
                        receiving=1;

                        break;
                }

                case 0x30:
                {
                        Message_Waiting=1;
                        for(int_rec_i=0;int_rec_i<18;int_rec_i++)
                        {

MY_ID[int_rec_i]=RxBuffer[6+int_rec_i];

TAGIDMSG[int_rec_i+6]=RxBuffer[6+int_rec_i];
                        }
                        id_change=1;
                        first_talk=0;
                        second_talk=0;
                        can_sleep=0;
                        break;
                }

                case 0x35:
                {
                        Message_Waiting=1;
                        read_MY_SEG();
                        for(int_rec_i=0;int_rec_i<8;int_rec_i++)
                        {

MY_ID[int_rec_i+8]=RxBuffer[6+int_rec_i+8];

TAGIDMSG[int_rec_i+6+8]=RxBuffer[6+int_rec_i+8];
                        }
                        plate_change=1;
                        first_talk=0;
                        second_talk=0;
                        can_sleep=0;
                        break;
                }

                case 0x60:// Active
                {
                        Message_Waiting=1;

                        Actived=1;
                        first_talk=0;
                        second_talk=0;
                        can_sleep=0;
                        Actived_failed=0;
                        break;
                }

                case 0x70:
                {
                        Message_Waiting=1;

                        Power_TEST=1;
                        first_talk=0;
                        second_talk=0;
                        can_sleep=0;
                        break;
                }
                case 0x90:
                {
                        reset_KB=1;
                        Message_Waiting=1;

                        Power_TEST=0;
                        first_talk=0;
                        second_talk=0;
                        can_sleep=0;
                        break;
                }

                default:
                {

                        Message_Waiting=0;
                        Strobe(RF_SRX);
                        radio_state=RF_SRX;
                        transmitting=0;
                        receiving=1;
                        break;
                }

                }
              }
              else
              {

                        Message_Waiting=0;
                        Strobe(RF_SRX);
                        radio_state=RF_SRX;
                        transmitting=0;
                        receiving=1;
              }
            }
            else
            {

                    Message_Waiting=0;
                    Strobe(RF_SRX);
                    radio_state=RF_SRX;
                    transmitting=0;
                    receiving=1;
            }
          }
          else if(transmitting && !receiving)

          {
            RF1AIE &= ~BIT9;
            P3OUT &= ~BIT6;
            P3OUT &= ~BIT4;

            transmitting = 0;
          }
          else
          {
                    Message_Waiting=0;
                    Strobe(RF_SRX);
                    radio_state=RF_SRX;
                    transmitting=0;
                    receiving=1;

          }
          break;
        case 22: break;
```

```
  case 24: break;
  case 26: break;
  case 28: break;
  case 30: break;
  case 32: break;
 }
 can_sleep=0;
 P1OUT&=~BIT0;
 __bic_SR_register_on_exit(LPM3_bits);
}
#pragma vector=PORT1_VECTOR
__interrupt void PORT1_ISR(void)
{
 switch(__even_in_range(P1IV, 16))
 {
  case  0: break;
  case  2: break;                 // P1.0 IFG
  case  4: break;                 // P1.1 IFG
  case  6: break;                 // P1.2 IFG
  case  8: break;                 // P1.3 IFG
  case 10:
                can_sleep=0;
                __bic_SR_register_on_exit(LPM3_bits);

                Power_DOWN=0;
                {WDTCTL                                 =
WDTPW+WDTCNTCL+WDTIS2;}
                while(1);


                break;
  case 12: break;
  case 14:
                can_sleep=0;
                __bic_SR_register_on_exit(LPM3_bits);

                Power_DOWN=0;
                {WDTCTL                                 =
WDTPW+WDTCNTCL+WDTIS2;}
                while(1);

                break;

  case 16:
   if(on_bat)
   {
                wait_for_clear=1;

   }
   else
   {
                P1IE = 0;
                buttonPressed = 1;
                Enable_Start_Timer(50);
                can_sleep=0;
                __bic_SR_register_on_exit(LPM3_bits);
   }
   break;
  default :
                can_sleep=0;

__bic_SR_register_on_exit(LPM3_bits);

                Power_DOWN=0;
                {WDTCTL                                 =
WDTPW+WDTCNTCL+WDTIS2;}
                while(1);

 }
}


#pragma vector=TIMER1_A0_VECTOR
__interrupt void TIMER1_A0_ISR(void)
{

 if(timer_cycle>2000)
 {
                if(!(BIT7 & P1IN))
                {
                                buttonPressed = 0;
                                P1IFG = 0;
                                P1IE=1;
                                TA1CCTL0 |= ~CCIE;
                                timer_over=0;
                                InitButtonLeds();
                }
                else
                {
                                timer_over= 1;
                }
                timer_cycle=0;
 }
 else
 {
  TA1CCR0 += Timer_Count;
  timer_cycle++;
 }
 can_sleep=0;
 __bic_SR_register_on_exit(LPM3_bits);
}
```

EK-1.2. crypto.c

```
#include <stdio.h>
#include <stdlib.h>

long DELTA = ?;
long key[4]={?,?,?,?};
char NUM_ITERATIONS=?;

long RCV_DELTA = ?;
long RCV_key[4]={?,?,?,?};


long Crypto_BUF[16];
void Generate_Key(void)
{
                RCV_key[0]=rand()*rand()*rand();
                RCV_key[0]+=rand()*0x00010000;
                RCV_key[1]=rand()*rand()*rand();
                RCV_key[1]+=rand()*0x00010000;
                RCV_key[2]=rand()*rand()*rand();
                RCV_key[2]+=rand()*0x00010000;;
                RCV_key[3]=rand()*rand()*rand();
                RCV_key[3]+=rand()*0x00010000;

}

void make_key_for_buf(char *ptr)
{
                ptr[6]=(char)RCV_key[0]&0x000000ff;
                ptr[7]=(char)((RCV_key[0]&0x0000ff00)/0x00000100);
                ptr[8]=(char)((RCV_key[0]&0x00ff0000)/0x00010000);
                ptr[9]=(char)((RCV_key[0]&0xff000000)/0x01000000);

                ptr[10]=(char)RCV_key[1]&0x000000ff;
                ptr[11]=(char)((RCV_key[1]&0x0000ff00)/0x00000100);
                ptr[12]=(char)((RCV_key[1]&0x00ff0000)/0x00010000);
                ptr[13]=(char)((RCV_key[1]&0xff000000)/0x01000000);

                ptr[14]=(char)RCV_key[2]&0x000000ff;
                ptr[15]=(char)((RCV_key[2]&0x0000ff00)/0x00000100);
                ptr[16]=(char)((RCV_key[2]&0x00ff0000)/0x00010000);
                ptr[17]=(char)((RCV_key[2]&0xff000000)/0x01000000);

                ptr[18]=(char)RCV_key[3]&0x000000ff;
                ptr[19]=(char)((RCV_key[3]&0x0000ff00)/0x00000100);
                ptr[20]=(char)((RCV_key[3]&0x00ff0000)/0x00010000);
                ptr[21]=(char)((RCV_key[3]&0xff000000)/0x01000000);

}

void make_buf_for_key(char *ptr)
{
                long
temp_make_buf=0,temp_make_buf1=0,temp_make_buf2=0,temp_make_
buf3=0;
                RCV_key[0]=0;
                RCV_key[1]=0;
                RCV_key[2]=0;
                RCV_key[3]=0;

                temp_make_buf=0;temp_make_buf1=0;temp_make_buf2=0;t
emp_make_buf3=0;
```

```
		temp_make_buf = ptr[6];
		temp_make_buf &= 0x000000ff;
		temp_make_buf1 = ptr[7]*0x00000100;
		temp_make_buf1 &= 0x0000ff00;
		temp_make_buf2 = ptr[8]*0x00010000;
		temp_make_buf2 &= 0x00ff0000;
		temp_make_buf3 =ptr[9]*0x01000000;
		temp_make_buf3 &=0xff000000;
		RCV_key[0]=temp_make_buf+temp_make_buf1+temp_mak
e_buf2+temp_make_buf3;

		temp_make_buf=0;temp_make_buf1=0;temp_make_buf2=0;t
emp_make_buf3=0;
		temp_make_buf = ptr[10];
		temp_make_buf &= 0x000000ff;
		temp_make_buf1 = ptr[11]*0x00000100;
		temp_make_buf1 &= 0x0000ff00;
		temp_make_buf2 = ptr[12]*0x00010000;
		temp_make_buf2 &= 0x00ff0000;
		temp_make_buf3 =ptr[13]*0x01000000;
		temp_make_buf3 &=0xff000000;
		RCV_key[1]=temp_make_buf+temp_make_buf1+temp_mak
e_buf2+temp_make_buf3;

		temp_make_buf=0;temp_make_buf1=0;temp_make_buf2=0;t
emp_make_buf3=0;
		temp_make_buf = ptr[14];
		temp_make_buf &= 0x000000ff;
		temp_make_buf1 = ptr[15]*0x00000100;
		temp_make_buf1 &= 0x0000ff00;
		temp_make_buf2 = ptr[16]*0x00010000;
		temp_make_buf2 &= 0x00ff0000;
		temp_make_buf3 =ptr[17]*0x01000000;
		temp_make_buf3 &=0xff000000;
		RCV_key[2]=temp_make_buf+temp_make_buf1+temp_mak
e_buf2+temp_make_buf3;

		temp_make_buf=0;temp_make_buf1=0;temp_make_buf2=0;t
emp_make_buf3=0;
		temp_make_buf = ptr[18];
		temp_make_buf &= 0x000000ff;
		temp_make_buf1 = ptr[19]*0x00000100;
		temp_make_buf1 &= 0x0000ff00;
		temp_make_buf2 = ptr[20]*0x00010000;
		temp_make_buf2 &= 0x00ff0000;
		temp_make_buf3 =ptr[21]*0x01000000;
		temp_make_buf3 &=0xff000000;
		RCV_key[3]=temp_make_buf+temp_make_buf1+temp_mak
e_buf2+temp_make_buf3;

		RCV_key[3]=ptr[18] + ptr[19]*0x00000100 +
ptr[20]*0x00010000  + ptr[21]*0x01000000;
}

void Generate_Delta(void)
{
		RCV_DELTA=0;
		if((RCV_key[0]&0x0f000000)>5){RCV_DELTA+=RCV_ke
y[0]&0x000000ff;}
		else{RCV_DELTA+=RCV_key[1]&0x000000ff;}

		if((RCV_key[1]&0x0000f000)>9){RCV_DELTA+=RCV_ke
y[2]&0x0000ff00;}
		else{RCV_DELTA+=RCV_key[3]&0x0000ff00;}

		if((RCV_key[2]&0x00f00000)>8){RCV_DELTA+=RCV_ke
y[0]&0x00ff0000;}
		else{RCV_DELTA+=RCV_key[1]&0x00ff0000;}

		if((RCV_key[3]&0xf000000)>3){RCV_DELTA+=RCV_key
[2]&0xff000000;}
		else{RCV_DELTA+=RCV_key[3]&0xff000000;}
RCV_DELTA=0x55555555;
}
void Encode(long* data, char dataLength)
{
  char i=0;
  long x1=0,x1a=0,x1b=0,sum1a,dat;
  long x2=0,x2a=0,x2b=0;
  long sum;
  char iterationCount;

  while(i<dataLength)
  {
    sum = 0;
    x1=data[i];
    x2=data[i+1];

    iterationCount = NUM_ITERATIONS;
    //x1=0x10000000;
    //x2=0x10000000;


      while(iterationCount > 0)
    {
      x2a=(x2<<4);
      x2b=(x2>>5);
      x1 += (((x2a) ^ (x2b)) + x2) ^ (sum + key[(sum&0x03)]);
      sum+=DELTA;
      x1a=(x1<<4);
      x1b=(x1>>5);
      sum1a=sum>>11;
      x2 += (((x1a) ^ (x1b)) + x1) ^ (sum + key[((sum>>11)&0x03)]);
      dat=(sum + key[((sum>>11)&0x03)]);
      iterationCount--;
    }


    data[i]=x1;
    data[i+1]=x2;
    i+=2;
  }
}

void Encode_TAG(long* data, char dataLength)
{
  char i=0;
  long x1=0,x1a=0,x1b=0,sum1a,dat;
  long x2=0,x2a=0,x2b=0;
  long sum;
  char iterationCount;

  while(i<dataLength)
  {
    sum = 0;
    x1=data[i];
    x2=data[i+1];

    iterationCount = NUM_ITERATIONS;
    //x1=0x10000000;
    //x2=0x10000000;


      while(iterationCount > 0)
    {
      x2a=(x2<<4);
      x2b=(x2>>5);
      x1 += (((x2a) ^ (x2b)) + x2) ^ (sum + RCV_key[(sum&0x03)]);
      sum+=RCV_DELTA;
      x1a=(x1<<4);
      x1b=(x1>>5);
      sum1a=sum>>11;
      x2 += (((x1a) ^ (x1b)) + x1) ^ (sum +
RCV_key[((sum>>11)&0x03)]);
      dat=(sum + RCV_key[((sum>>11)&0x03)]);
      iterationCount--;
    }


    data[i]=x1;
    data[i+1]=x2;
    i+=2;
  }
}


/*************************************************************
**********
* Function:     void Decode(int32* data, int8 dataLength)
*          Decodes data pointed to by *data. dataLength is
*          measured in int32s, and must be an even number. This
*          means the minimum number of bytes to be decoded is 8.
*          eg decoding 8 bytes:
*            int8 some_bytes[8];
*            Decode(some_bytes,2);
*
*          uses the key[] variable - ensure it is appropriately loaded
```

```
*
************************************************************
*********/

void Decode(long* data, char dataLength)
{
  char i=0;
  long x1;
  long x2;
  long sum;
  char iterations;

  iterations = NUM_ITERATIONS;

  while(i<dataLength)
  {
    sum = DELTA*iterations;
    x1=data[i];
    x2=data[i+1];

    while(sum != 0)
    {
      x2 -= (((x1<<4) ^ (x1>>5)) + x1) ^ (sum + key[((sum>>11)&0x03)]
);
      sum-=DELTA;
      x1 -= (((x2<<4) ^ (x2>>5)) + x2) ^ (sum + key[(sum&0x03)]);
    }
    data[i]=x1;
    data[i+1]=x2;
    i+=2;
  }
}

void Decode_TAG(long* data, char dataLength)
{
  char i=0;
  long x1;
  long x2;
  long sum;
  char iterations;

  iterations = NUM_ITERATIONS;

  while(i<dataLength)
  {
    sum = RCV_DELTA*iterations;
    x1=data[i];
    x2=data[i+1];

    while(sum != 0)
    {
      x2 -= (((x1<<4) ^ (x1>>5)) + x1) ^ (sum +
RCV_key[((sum>>11)&0x03)] );
      sum-=RCV_DELTA;
      x1 -= (((x2<<4) ^ (x2>>5)) + x2) ^ (sum +
RCV_key[(sum&0x03)]);
    }
    data[i]=x1;
    data[i+1]=x2;
    i+=2;
  }
}


int calcrc(char *ptr,char length,long crc_ini)
{
 long crcresult;
 char kcrc,icrc,tempcrc;
 char carry_in=0,carry_out=0;
 crcresult=crc_ini;
 for(icrc=0;icrc<length;icrc++)
 {
   tempcrc=ptr[icrc];
   for(kcrc=0;kcrc<8;kcrc++)
   {
     if((crcresult&0x01)!=(tempcrc &0x01) )
     {
       carry_in=1;
       crcresult^=0x4002;
     }
     else carry_in=0;
     if((crcresult&0x01))carry_out=1;
     else carry_out=0;
     crcresult>>=1;tempcrc>>=1;
     if(carry_in)crcresult|=0x8000;
     if(carry_out)tempcrc|=0x80;
   }
 }
 return(crcresult);
}

void make_for_Crypto_buf(char *ptr, char length)
{
 char make_for_Crypto_buf_i;
 long
temp_Crypto_BUF,temp_Crypto_BUF1,temp_Crypto_BUF2,temp_Crypt
o_BUF3;
 for
(make_for_Crypto_buf_i=0;make_for_Crypto_buf_i<15;make_for_Crypt
o_buf_i++)
 {
   Crypto_BUF[make_for_Crypto_buf_i]=0;
 }
 for
(make_for_Crypto_buf_i=0;make_for_Crypto_buf_i<(length/4);make_for
_Crypto_buf_i++)
 {
   temp_Crypto_BUF=0;
   temp_Crypto_BUF1=0;
   temp_Crypto_BUF2=0;
   temp_Crypto_BUF3=0;
   temp_Crypto_BUF=ptr[(make_for_Crypto_buf_i*4)+1];
   temp_Crypto_BUF1=ptr[(make_for_Crypto_buf_i*4)+2];
   temp_Crypto_BUF1<<=8;
   temp_Crypto_BUF2=ptr[(make_for_Crypto_buf_i*4)+3];
   temp_Crypto_BUF2<<=16;
   temp_Crypto_BUF3=ptr[(make_for_Crypto_buf_i*4)+4];
   temp_Crypto_BUF3<<=24;

Crypto_BUF[make_for_Crypto_buf_i]=temp_Crypto_BUF+temp_Crypto
_BUF1+temp_Crypto_BUF2+temp_Crypto_BUF3;

 }

}
void make_for_buf(char *ptr, char length,char start)
{
 char make_for_Crypto_buf_i;
 long temp_BUF,temp_BUF1,temp_BUF2,temp_BUF3;
 for
(make_for_Crypto_buf_i=1;make_for_Crypto_buf_i<length;make_for_Cr
ypto_buf_i++)
 {
   ptr[make_for_Crypto_buf_i]=0;
 }
 for
(make_for_Crypto_buf_i=0;make_for_Crypto_buf_i<(length/4);make_for
_Crypto_buf_i++)
 {
   temp_BUF=Crypto_BUF[make_for_Crypto_buf_i];
   temp_BUF1=Crypto_BUF[make_for_Crypto_buf_i];
   temp_BUF2=Crypto_BUF[make_for_Crypto_buf_i];
   temp_BUF3=Crypto_BUF[make_for_Crypto_buf_i];

   ptr[(make_for_Crypto_buf_i*4)+1]=(char)0x000000ff & temp_BUF;
   temp_BUF1>>=8;
   ptr[(make_for_Crypto_buf_i*4)+2]=(char)0x000000ff & temp_BUF1;
   temp_BUF2>>=16;
   ptr[(make_for_Crypto_buf_i*4)+3]=(char)0x000000ff & temp_BUF2;
   temp_BUF3>>=24;
   ptr[(make_for_Crypto_buf_i*4)+4]=(char)0x000000ff & temp_BUF3;

 }

}
```

EK-1.3. Flashfunctions.c

```
#include "cc430x613x.h"
#include "D:\ccs projects\cc430f6137.h"

char MY_ID[50];
```

```
long flash_crc_value=0,flash_crc_value1=0;
extern int calcrc(char *ptr,char length,long crc_ini);
void write_TX_POWER(char value)
{

  char * Flash_ptr;                  // Initialize Flash pointer
  Flash_ptr = (char *) 0x1880;
  __bic_SR_register(  GIE );               // 5xx Workaround:
Disable global
                                     //  interrupt  while  erasing.  Re-
Enable
                                     // GIE if needed
  FCTL3 = FWKEY;                     // Clear Lock bit
//  FCTL1 = FWKEY+ERASE;                   // Set Erase bit
  *Flash_ptr = 0;                    // Dummy write to erase
Flash seg
  FCTL1 = FWKEY+WRT;                 // Set WRT bit for
write operation


   *Flash_ptr = value;               // Write value to flash

  FCTL1 = FWKEY;                     // Clear WRT bit
  FCTL3 = FWKEY+LOCK;                // Set LOCK bit
  __bis_SR_register(  GIE );
}

char read_TX_POWER(void)
{
  char value;
  char *Flash_ptr;
  Flash_ptr = (char *) 0x1880;
  __bic_SR_register(  GIE );               // 5xx Workaround:
Disable global
                                     //  interrupt  while  erasing.  Re-
Enable
                                     // GIE if needed
  FCTL3 = FWKEY;                     // Clear Lock bit
//  FCTL1 = FWKEY+ERASE;                   // Set Erase bit
  //*Flash_ptr = 0;                  // Dummy write to erase
Flash seg
  FCTL1 = FWKEY+WRT;                 // Set WRT bit for
write operation
  value=*Flash_ptr;                  // Read value to flash

  FCTL1 = FWKEY;                     // Clear WRT bit
  FCTL3 = FWKEY+LOCK;                // Set LOCK bit
  __bis_SR_register(  GIE );
  return value;
}

void write_MY_CTX(char value)
{

  char * Flash_ptr;                  // Initialize Flash pointer
  Flash_ptr = (char *) 0x1881;
  __bic_SR_register(  GIE );               // 5xx Workaround:
Disable global
                                     //  interrupt  while  erasing.  Re-
Enable
                                     // GIE if needed
  FCTL3 = FWKEY;                     // Clear Lock bit
  //FCTL1 = FWKEY+ERASE;                   // Set Erase bit
  *Flash_ptr = 0;                    // Dummy write to erase
Flash seg
  FCTL1 = FWKEY+WRT;                 // Set WRT bit for
write operation


   *Flash_ptr = value;               // Write value to flash

  FCTL1 = FWKEY;                     // Clear WRT bit
  FCTL3 = FWKEY+LOCK;                // Set LOCK bit
  __bis_SR_register(  GIE );
}

char read_MY_CTX(void)
{
  char value;
  char *Flash_ptr;
  Flash_ptr = (char *) 0x1881;
  __bic_SR_register(  GIE );               // 5xx Workaround:
Disable global
                                     //  interrupt  while  erasing.  Re-
Enable
                                     // GIE if needed
  FCTL3 = FWKEY;                     // Clear Lock bit
  //FCTL1 = FWKEY+ERASE;                   // Set Erase bit
  //*Flash_ptr = 0;                  // Dummy write to erase
Flash seg
  FCTL1 = FWKEY+WRT;                 // Set WRT bit for
write operation
  value=*Flash_ptr;                  // Read value to flash

  FCTL1 = FWKEY;                     // Clear WRT bit
  FCTL3 = FWKEY+LOCK;                // Set LOCK bit
  __bis_SR_register(  GIE );
  return value;
}

void write_MY_ACT(char value)
{

  char * Flash_ptr;                  // Initialize Flash pointer
  Flash_ptr = (char *) 0x1882;
  __bic_SR_register(  GIE );               // 5xx Workaround:
Disable global
                                     //  interrupt  while  erasing.  Re-
Enable
                                     // GIE if needed
  FCTL3 = FWKEY;                     // Clear Lock bit
  //FCTL1 = FWKEY+ERASE;                   // Set Erase bit
  *Flash_ptr = 0;                    // Dummy write to erase
Flash seg
  FCTL1 = FWKEY+WRT;                 // Set WRT bit for
write operation


   *Flash_ptr = value;               // Write value to flash

  FCTL1 = FWKEY;                     // Clear WRT bit
  FCTL3 = FWKEY+LOCK;                // Set LOCK bit
  __bis_SR_register(  GIE );
}

char read_MY_ACT(void)
{
  char value;
  char *Flash_ptr;
  Flash_ptr = (char *) 0x1882;
  __bic_SR_register(  GIE );               // 5xx Workaround:
Disable global
                                     //  interrupt  while  erasing.  Re-
Enable
                                     // GIE if needed
  FCTL3 = FWKEY;                     // Clear Lock bit
  //FCTL1 = FWKEY+ERASE;                   // Set Erase bit
  //*Flash_ptr = 0;                  // Dummy write to erase
Flash seg
  FCTL1 = FWKEY+WRT;                 // Set WRT bit for
write operation
  value=*Flash_ptr;                  // Read value to flash

  FCTL1 = FWKEY;                     // Clear WRT bit
  FCTL3 = FWKEY+LOCK;                // Set LOCK bit
  __bis_SR_register(  GIE );
  return value;
}



void write_MY_ID(void)
{

  char * Flash_ptr,flash_i=0;                  // Initialize Flash
pointer

  Flash_ptr = (char *) 0x1882;
  __bic_SR_register(  GIE );               // 5xx Workaround:
Disable global
                                     //  interrupt  while  erasing.  Re-
Enable
                                     // GIE if needed
  FCTL3 = FWKEY;                     // Clear Lock bit
  FCTL1 = FWKEY+ERASE;                   // Set Erase bit
  *Flash_ptr = 0;                    // Dummy write to erase
Flash seg
```

```
  FCTL1 = FWKEY+WRT;                    // Set WRT bit for
write operation
  //*Flash_ptr = 0x55;
                  // Write value to flash
  for(flash_i=0;flash_i<18;flash_i++)
  {
    //Flash_ptr++;
    *Flash_ptr++=MY_ID[flash_i+2];
  }
  FCTL1 = FWKEY;                        // Clear WRT bit
  FCTL3 = FWKEY+LOCK;                   // Set LOCK bit
  __bis_SR_register( GIE );
}


void read_MY_ID(void)
{
  char flash_i=0;
  char *Flash_ptr;
  Flash_ptr = (char *) 0x1882;
  __bic_SR_register( GIE );             // 5xx Workaround:
Disable global
                                        // interrupt while erasing. Re-
Enable
                                        // GIE if needed
  FCTL3 = FWKEY;                        // Clear Lock bit
  //FCTL1 = FWKEY+ERASE;                // Set Erase bit
  //*Flash_ptr = 0;                     // Dummy write to erase
Flash seg
  FCTL1 = FWKEY+WRT;                    // Set WRT bit for
write operation
  //value=*Flash_ptr;                   // Read value to flash
  for(flash_i=0;flash_i<18;flash_i++)
  {
    //Flash_ptr++;
    MY_ID[flash_i]=*Flash_ptr++;
  }
  FCTL1 = FWKEY;                        // Clear WRT bit
  FCTL3 = FWKEY+LOCK;                   // Set LOCK bit
  //return value;
  __bis_SR_register( GIE );
}



void write_MY_SEG(void)
{

  char * Flash_ptr,flash_i=0;           // Initialize Flash
pointer


  flash_crc_value=calcrc(MY_ID,22,0);
  flash_crc_value1=  flash_crc_value;
  flash_crc_value&=0x000000ff;
  flash_crc_value1>>=8;
  flash_crc_value1&=0x000000ff;
  MY_ID[22]=(char)flash_crc_value1;
  MY_ID[23]=(char)flash_crc_value;


  Flash_ptr = (char *) 0x1880;
  __bic_SR_register( GIE );             // 5xx Workaround:
Disable global
                                        // interrupt while erasing. Re-
Enable
                                        // GIE if needed
  FCTL3 = FWKEY;                        // Clear Lock bit
  FCTL1 = FWKEY+ERASE;                  // Set Erase bit
  *Flash_ptr = 0;                       // Dummy write to erase
Flash seg
  FCTL1 = FWKEY+WRT;                    // Set WRT bit for
write operation
  //*Flash_ptr = 0x55;
                  // Write value to flash
  for(flash_i=0;flash_i<29;flash_i++)
  {
    //Flash_ptr++;
    *Flash_ptr++=MY_ID[flash_i];
  }
  FCTL1 = FWKEY;                        // Clear WRT bit
  FCTL3 = FWKEY+LOCK;                   // Set LOCK bit
  __bis_SR_register( GIE );
}


void read_MY_SEG(void)
{
  char flash_i=0;
  char *Flash_ptr;
  Flash_ptr = (char *) 0x1880;
  __bic_SR_register( GIE );             // 5xx Workaround:
Disable global
                                        // interrupt while erasing. Re-
Enable
                                        // GIE if needed
  FCTL3 = FWKEY;                        // Clear Lock bit
  FCTL1 = FWKEY+ERASE;                  // Set Erase bit
  //*Flash_ptr = 0;                     // Dummy write to erase
Flash seg
  FCTL1 = FWKEY+WRT;                    // Set WRT bit for
write operation
  //value=*Flash_ptr;                   // Read value to flash
  for(flash_i=0;flash_i<29;flash_i++)
  {
    //Flash_ptr++;
    MY_ID[flash_i]=*Flash_ptr++;
  }
  FCTL1 = FWKEY;                        // Clear WRT bit
  FCTL3 = FWKEY+LOCK;                   // Set LOCK bit
  //return value;
  __bis_SR_register( GIE );
}


void read_MY_SEGD(void)
{
  char flash_i=0;
  char *Flash_ptr;
  Flash_ptr = (char *) 0x1800;
  __bic_SR_register( GIE );             // 5xx Workaround:
Disable global
                                        // interrupt while erasing. Re-
Enable
                                        // GIE if needed
  FCTL3 = FWKEY;                        // Clear Lock bit
  FCTL1 = FWKEY+ERASE;                  // Set Erase bit
  //*Flash_ptr = 0;                     // Dummy write to erase
Flash seg
  FCTL1 = FWKEY+WRT;                    // Set WRT bit for
write operation
  //value=*Flash_ptr;                   // Read value to flash
  for(flash_i=0;flash_i<29;flash_i++)
  {
    //Flash_ptr++;
    MY_ID[flash_i]=*Flash_ptr++;
  }
  FCTL1 = FWKEY;                        // Clear WRT bit
  FCTL3 = FWKEY+LOCK;                   // Set LOCK bit
  //return value;
  __bis_SR_register( GIE );
}



void copy_C2D(void)
{
  unsigned int flash_i;
  char *Flash_ptrC;
  char *Flash_ptrD;

  Flash_ptrC = (char *) 0x1880;         // Initialize Flash
segment C ptr
  Flash_ptrD = (char *) 0x1800;         // Initialize Flash
segment D ptr

  __bic_SR_register( GIE );             // 5xx Workaround:
Disable global
                                        // interrupt while erasing. Re-
Enable
                                        // GIE if needed
  FCTL3 = FWKEY;                        // Clear Lock bit
  FCTL1 = FWKEY+ERASE;                  // Set Erase bit
  *Flash_ptrD = 0;                      // Dummy write to erase
Flash seg D
  FCTL1 = FWKEY+WRT;                    // Set WRT bit for
write operation

  for (flash_i = 0; flash_i < 128; flash_i++)
```

```
  {
    *Flash_ptrD++ = *Flash_ptrC++;              // copy value segment C to seg D
  }

  FCTL1 = FWKEY;                    // Clear WRT bit
  FCTL3 = FWKEY+LOCK;               // Set LOCK bit
}

void copy_D2C(void)
{
  unsigned int flash_i;
  char *Flash_ptrC;
  char *Flash_ptrD;

  Flash_ptrC = (char *) 0x1880;             // Initialize Flash segment C ptr
  Flash_ptrD = (char *) 0x1800;             // Initialize Flash segment D ptr

  __bic_SR_register( GIE );                 // 5xx Workaround: Disable global
                                            // interrupt while erasing. Re-Enable
                                            // GIE if needed
  FCTL3 = FWKEY;                    // Clear Lock bit
  FCTL1 = FWKEY+ERASE;              // Set Erase bit
  *Flash_ptrC = 0;                  // Dummy write to erase Flash seg D
  FCTL1 = FWKEY+WRT;                // Set WRT bit for write operation

  for (flash_i = 0; flash_i < 128; flash_i++)
  {
    *Flash_ptrC++ = *Flash_ptrD++;              // copy value segment C to seg D
  }

  FCTL1 = FWKEY;                    // Clear WRT bit
  FCTL3 = FWKEY+LOCK;               // Set LOCK bit
}
```

EK-1.3. hal_pmm.c

```
#include "msp430.h"
#include "hal_pmm.h"

#define _HAL_PMM_DISABLE_SVML_
#define _HAL_PMM_DISABLE_SVSL_
#define _HAL_PMM_DISABLE_FULL_PERFORMANCE_

//*************************************************************************//
#ifdef _HAL_PMM_DISABLE_SVML_
#define _HAL_PMM_SVMLE SVMLE
#else
#define _HAL_PMM_SVMLE 0
#endif
#ifdef _HAL_PMM_DISABLE_SVSL_
#define _HAL_PMM_SVSLE SVSLE
#else
#define _HAL_PMM_SVSLE 0
#endif
#ifdef _HAL_PMM_DISABLE_FULL_PERFORMANCE_
#define _HAL_PMM_SVSFP SVSLFP
#else
#define _HAL_PMM_SVSFP 0
#endif
//*************************************************************************//
// Set VCore
//*************************************************************************//
unsigned int SetVCore (unsigned char level)
{
  unsigned int actlevel;
  unsigned int status = 0;
  level &= PMMCOREV_3;                      // Set Mask for Max. level
  actlevel = (PMMCTL0 & PMMCOREV_3);        // Get actual VCore

  while (((level != actlevel) && (status == 0)) || (level < actlevel))
                                            // step by step increase or decrease
  {
    if (level > actlevel)
      status = SetVCoreUp(++actlevel);
    else
      status = SetVCoreDown(--actlevel);
  }
  return status;
}

//*************************************************************************//
// Set VCore Up
//*************************************************************************//
unsigned int SetVCoreUp (unsigned char level)
{
  unsigned int PMMRIE_backup,SVSMHCTL_backup;

  // Open PMM registers for write access
  PMMCTL0_H = 0xA5;

  // Disable dedicated Interrupts to prevent that needed flags will be cleared
  PMMRIE_backup = PMMRIE;
  PMMRIE &= ~(SVSMHDLYIE | SVSMLDLYIE | SVMLVLRIE | SVMHVLRIE | SVMHVLRPE);
  // Set SVM highside to new level and check if a VCore increase is possible
  SVSMHCTL_backup = SVSMHCTL;
  PMMIFG &= ~(SVMHIFG | SVSMHDLYIFG);
  SVSMHCTL = SVMHE | SVMHFP | (SVSMHRRL0 * level);
  // Wait until SVM highside is settled
  while ((PMMIFG & SVSMHDLYIFG) == 0);
  // Disable full-performance mode to save energy
  SVSMHCTL &= ~_HAL_PMM_SVSFP ;
  // Check if a VCore increase is possible
  if ((PMMIFG & SVMHIFG) == SVMHIFG){
      //-> Vcc is to low for a Vcore increase
      // recover the previous settings
      PMMIFG &= ~SVSMHDLYIFG;
      SVSMHCTL = SVSMHCTL_backup;
      // Wait until SVM highside is settled
      while ((PMMIFG & SVSMHDLYIFG) == 0);
      // Clear all Flags
      PMMIFG &= ~(SVMHVLRIFG | SVMHIFG | SVSMHDLYIFG | SVMLVLRIFG | SVMLIFG | SVSMLDLYIFG);
      // backup PMM-Interrupt-Register
      PMMRIE = PMMRIE_backup;

      // Lock PMM registers for write access
      PMMCTL0_H = 0x00;
      return      PMM_STATUS_ERROR;
// return: voltage not set
  }
  // Set also SVS highside to new level
      //-> Vcc is high enough for a Vcore increase
  SVSMHCTL |= SVSHE | (SVSHRVL0 * level);
  // Set SVM low side to new level
  SVSMLCTL = SVMLE | SVMLFP | (SVMLRRL0 * level);
  // Wait until SVM low side is settled
  while ((PMMIFG & SVSMLDLYIFG) == 0);
  // Clear already set flags
  PMMIFG &= ~(SVMLVLRIFG | SVMLIFG);
  // Set VCore to new level
  PMMCTL0_L = PMMCOREV0 * level;
  // Wait until new level reached
  if (PMMIFG & SVMLIFG)
  while ((PMMIFG & SVMLVLRIFG) == 0);
  // Set also SVS/SVM low side to new level
  PMMIFG &= ~SVSMLDLYIFG;
  SVSMLCTL |= SVSLE | (SVSLRVL0 * level);
  // wait for lowside delay flags
  while ((PMMIFG & SVSMLDLYIFG) == 0);

 // Disable SVS/SVM Low
 // Disable full-performance mode to save energy
```

```
            SVSMLCTL                                    &= ~(_HAL_PMM_DISABLE_SVSL_+_HAL_PMM_DISABLE_SVML_+_HAL_PMM_SVSFP );

            // Clear all Flags
            PMMIFG  &=  ~(SVMHVLRIFG  |  SVMHIFG  |  SVSMHDLYIFG | SVMLVLRIFG | SVMLIFG | SVSMLDLYIFG);
            // backup PMM-Interrupt-Register
            PMMRIE = PMMRIE_backup;

            // Lock PMM registers for write access
            PMMCTL0_H = 0x00;
            return PMM_STATUS_OK;                       // return: OK
          }

          //*************************************************************************//
          // Set VCore down (Independent from the enabled Interrupts in PMMRIE)
          //*************************************************************************//
          unsigned int SetVCoreDown (unsigned char level)
          {
            unsigned int PMMRIE_backup;

            // Open PMM registers for write access
            PMMCTL0_H = 0xA5;

            // Disable dedicated Interrupts to prevent that needed flags will be cleared
            PMMRIE_backup = PMMRIE;
            PMMRIE  &=  ~(SVSMHDLYIE  |  SVSMLDLYIE  |  SVMLVLRIE | SVMHVLRIE | SVMHVLRPE);

            // Set SVM high side and SVM low side to new level
            PMMIFG &= ~(SVMHIFG | SVSMHDLYIFG | SVMLIFG | SVSMLDLYIFG);
            SVSMHCTL  =  SVMHE  |  SVMHFP  |  (SVSMHRRL0  *  level);
            SVSMLCTL  =  SVMLE  |  SVMLFP  |  (SVSMLRRL0  *  level);
            // Wait until SVM high side and SVM low side is settled
            while ((PMMIFG & SVSMHDLYIFG) == 0 || (PMMIFG & SVSMLDLYIFG) == 0);

            // Set VCore to new level
            PMMCTL0_L = PMMCOREV0 * level;

            // Set also SVS highside and SVS low side to new level
            PMMIFG &= ~(SVSHIFG | SVSMHDLYIFG | SVSLIFG | SVSMLDLYIFG);
            SVSMHCTL |= SVSHE | SVSHFP | (SVSHRVL0 * level);
            SVSMLCTL |= SVSLE | SVSLFP | (SVSLRVL0 * level);
            // Wait until SVS high side and SVS low side is settled
            while ((PMMIFG & SVSMHDLYIFG) == 0 || (PMMIFG & SVSMLDLYIFG) == 0);
            // Disable full-performance mode to save energy
            SVSMHCTL &= ~_HAL_PMM_SVSFP;
          // Disable SVS/SVM Low
            // Disable full-performance mode to save energy
            SVSMLCTL                                    &= ~(_HAL_PMM_DISABLE_SVSL_+_HAL_PMM_DISABLE_SVML_+_HAL_PMM_SVSFP );

            // Clear all Flags
            PMMIFG  &=  ~(SVMHVLRIFG  |  SVMHIFG  |  SVSMHDLYIFG | SVMLVLRIFG | SVMLIFG | SVSMLDLYIFG);
            // backup PMM-Interrupt-Register
            PMMRIE = PMMRIE_backup;
            // Lock PMM registers for write access
            PMMCTL0_H = 0x00;

            if ((PMMIFG & SVMHIFG) == SVMHIFG)
              return PMM_STATUS_ERROR;
                                                        // Highside is still to low for the adjusted VCore Level
            else return PMM_STATUS_OK;
                                                        // Return: OK
          }
```

EK-1.4. RF1A.c

```
          #include "RF1A.h"
          #include "cc430x613x.h"

          //
************************************************************
*****************
          // @fn          Strobe
          // @brief       Send a command strobe to the radio. Includes workaround for RF1A7
          // @param       unsigned char strobe       The strobe command to be sent
          // @return      unsigned char statusByte   The status byte that follows the strobe
          //
************************************************************
*****************
          unsigned char Strobe(unsigned char strobe)
          {
            unsigned char statusByte = 0;
            unsigned int  gdo_state;
            //WDTCTL = WDTPW ;
            WDTCTL = WDTPW+WDTCNTCL+WDTIS2;
            // Check for valid strobe command
            if((strobe == 0xBD) || ((strobe >= RF_SRES) && (strobe <= RF_SNOP)))
             {
              // Clear the Status read flag
              RF1AIFCTL1 &= ~(RFSTATIFG);

              // Wait for radio to be ready for next instruction
              while( !(RF1AIFCTL1 & RFINSTRIFG))
              {
                  if(WDTCTL&0x0080)       {WDTCTL       = WDTPW+WDTCNTCL+WDTIS2;}
              }

              // Write the strobe instruction
              if ((strobe > RF_SRES) && (strobe < RF_SNOP))
              {
               gdo_state  =  ReadSingleReg(IOCFG2);          // buffer IOCFG2 state
               WriteSingleReg(IOCFG2,  0x29);           // chip-ready to GDO2
               WDTCTL = WDTPW+WDTCNTCL+WDTIS2;
               RF1AINSTRB = strobe;
               if ( (RF1AIN&0x04)== 0x04 )          // chip at sleep mode
               {
                if ( (strobe == RF_SXOFF) || (strobe == RF_SPWD) || (strobe == RF_SWOR) ) { }
                 else
                 {
                  while ((RF1AIN&0x04)== 0x04){
                      if(WDTCTL&0x0080)       {WDTCTL       = WDTPW+WDTCNTCL+WDTIS2;}
                  }    // chip-ready ?
                  // Delay for ~810usec at 1.05MHz CPU clock, see erratum RF1A7
                   __delay_cycles(850);
                 }
               }
               WriteSingleReg(IOCFG2,   gdo_state);          // restore IOCFG2 setting
               WDTCTL = WDTPW+WDTCNTCL+WDTIS2;
               while( !(RF1AIFCTL1 & RFSTATIFG) ){
                      if(WDTCTL&0x0080)       {WDTCTL       = WDTPW+WDTCNTCL+WDTIS2;}
               }
              }
              else                                  // chip active mode (SRES)
              {
               RF1AINSTRB = strobe;
              }
              statusByte = RF1ASTATB;
             }
             WDTCTL = WDTPW + WDTHOLD;
             return statusByte;
            }

          //
************************************************************
*****************
```

```
//  @fn          ReadSingleReg
//  @brief       Read a single byte from the radio register
//  @param       unsigned char addr       Target radio register address
//  @return      unsigned char data_out  Value of byte that was read
//
*************************************************************************
unsigned char ReadSingleReg(unsigned char addr)
{
  unsigned char data_out;
  WDTCTL = WDTPW+WDTCNTCL+WDTIS2;
  // Check for valid configuration register address, 0x3E refers to PATABLE
  if ((addr <= 0x2E) || (addr == 0x3E))
    // Send address + Instruction + 1 dummy byte (auto-read)
    RF1AINSTR1B = (addr | RF_SNGLREGRD);
  else
    // Send address + Instruction + 1 dummy byte (auto-read)
    RF1AINSTR1B = (addr | RF_STATREGRD);

  while (!(RF1AIFCTL1 & RFDOUTIFG) )
  {
                if(WDTCTL&0x0080)       {WDTCTL       = WDTPW+WDTCNTCL+WDTIS2;}
  }
  data_out = RF1ADOUTB;                 // Read data and clears the RFDOUTIFG
  WDTCTL = WDTPW + WDTHOLD;
  return data_out;
}

//
*************************************************************************
//  @fn          WriteSingleReg
//  @brief       Write a single byte to a radio register
//  @param       unsigned char addr       Target radio register address
//  @param       unsigned char value     Value to be written
//  @return      none
//
*************************************************************************
void WriteSingleReg(unsigned char addr, unsigned char value)
{
                WDTCTL = WDTPW+WDTCNTCL+WDTIS2;
  while (!(RF1AIFCTL1 & RFINSTRIFG)){
                if(WDTCTL&0x0080)       {WDTCTL       = WDTPW+WDTCNTCL+WDTIS2;}
  }       // Wait for the Radio to be ready for next instruction
  RF1AINSTRB = (addr | RF_SNGLREGWR);       // Send address + Instruction
  RF1ADINB = value;                         // Write data in
  WDTCTL = WDTPW + WDTHOLD;
  __no_operation();
}

//
*************************************************************************
//  @fn          ReadBurstReg
//  @brief       Read multiple bytes to the radio registers
//  @param       unsigned char addr       Beginning address of burst read
//  @param       unsigned char *buffer   Pointer to data table
//  @param       unsigned char count     Number of bytes to be read
//  @return      none
//
*************************************************************************
void ReadBurstReg( char addr,  char *buffer,  char count)
{
  unsigned int i;
  if(count > 0)
  {
                WDTCTL                  = WDTPW+WDTCNTCL+WDTIS2;
    while (!(RF1AIFCTL1 & RFINSTRIFG)){
                if(WDTCTL&0x0080)       {WDTCTL       = WDTPW+WDTCNTCL+WDTIS2;}
    }       // Wait for INSTRIFG
    RF1AINSTR1B = (addr | RF_REGRD);        // Send addr of first conf. reg. to be read
                                            // ... and the burst-register read instruction
    for (i = 0; i < (count-1); i++)
    {
      while (!(RFDOUTIFG&RF1AIFCTL1)){
                if(WDTCTL&0x0080)       {WDTCTL       = WDTPW+WDTCNTCL+WDTIS2;}
      }             // Wait for the Radio Core to update the RF1ADOUTB reg
      buffer[i] = RF1ADOUT1B;               // Read DOUT from Radio Core + clears RFDOUTIFG
                                            // Also initiates auo-read for next DOUT byte
    }
    buffer[count-1] = RF1ADOUT0B;           // Store the last DOUT from Radio Core
  }
  WDTCTL = WDTPW + WDTHOLD;
}

//
*************************************************************************
//  @fn          WriteBurstReg
//  @brief       Write multiple bytes to the radio registers
//  @param       unsigned char addr       Beginning address of burst write
//  @param       unsigned char *buffer   Pointer to data table
//  @param       unsigned char count     Number of bytes to be written
//  @return      none
//
*************************************************************************
void WriteBurstReg( char addr,  char *buffer,  char count)
{
  unsigned char i;
  WDTCTL = WDTPW+WDTCNTCL+WDTIS2;
  if(count > 0)
  {
    while (!(RF1AIFCTL1 & RFINSTRIFG)){
                if(WDTCTL&0x0080)       {WDTCTL       = WDTPW+WDTCNTCL+WDTIS2;}
    }       // Wait for the Radio to be ready for next instruction
    RF1AINSTRW = ((addr | RF_REGWR)<<8 ) + buffer[0]; // Send address + Instruction

    for (i = 1; i < count; i++)
    {
      RF1ADINB = buffer[i];                 // Send data
      //  WDTCTL = WDTPW+WDTCNTCL+WDTIS2;
      while (!(RFDINIFG & RF1AIFCTL1)){
                if(WDTCTL&0x0080)       {WDTCTL       = WDTPW+WDTCNTCL+WDTIS2;}
      }       // Wait for TX to finish
      // WDTCTL = WDTPW + WDTHOLD;
    }
    i = RF1ADOUTB;                          // Reset RFDOUTIFG flag which contains status byte
  }
  WDTCTL = WDTPW + WDTHOLD;
}

//
*************************************************************************
//  @fn          ResetRadioCore
//  @brief       Reset the radio core using RF_SRES command
//  @param       none
//  @return      none
//
*************************************************************************
void ResetRadioCore (void)
{
  Strobe(RF_SRES);                      // Reset the Radio Core
  Strobe(RF_SNOP);                      // Reset Radio Pointer
}
```

```
//
*************************************************************
*****************
// @fn          WriteRfSettings
// @brief       Write the minimum set of RF configuration
register settings
// @param       RF_SETTINGS *pRfSettings  Pointer to the
structure that holds the rf settings
// @return      none
//
*************************************************************
*****************
void WriteRfSettings(RF_SETTINGS *pRfSettings) {
  WriteSingleReg(FSCTRL1,  pRfSettings->fsctrl1);
  WriteSingleReg(FSCTRL0,  pRfSettings->fsctrl0);
  WriteSingleReg(FREQ2,    pRfSettings->freq2);
  WriteSingleReg(FREQ1,    pRfSettings->freq1);
  WriteSingleReg(FREQ0,    pRfSettings->freq0);
  WriteSingleReg(MDMCFG4,  pRfSettings->mdmcfg4);
  WriteSingleReg(MDMCFG3,  pRfSettings->mdmcfg3);
  WriteSingleReg(MDMCFG2,  pRfSettings->mdmcfg2);
  WriteSingleReg(MDMCFG1,  pRfSettings->mdmcfg1);
  WriteSingleReg(MDMCFG0,  pRfSettings->mdmcfg0);
  WriteSingleReg(CHANNR,   pRfSettings->channr);
  WriteSingleReg(DEVIATN,  pRfSettings->deviatn);
  WriteSingleReg(FREND1,   pRfSettings->frend1);
  WriteSingleReg(FREND0,   pRfSettings->frend0);
  WriteSingleReg(MCSM0 ,   pRfSettings->mcsm0);
  WriteSingleReg(FOCCFG,   pRfSettings->foccfg);
  WriteSingleReg(BSCFG,    pRfSettings->bscfg);
  WriteSingleReg(AGCCTRL2, pRfSettings->agcctrl2);
  WriteSingleReg(AGCCTRL1, pRfSettings->agcctrl1);
  WriteSingleReg(AGCCTRL0, pRfSettings->agcctrl0);
  WriteSingleReg(FSCAL3,   pRfSettings->fscal3);
  WriteSingleReg(FSCAL2,   pRfSettings->fscal2);
  WriteSingleReg(FSCAL1,   pRfSettings->fscal1);
  WriteSingleReg(FSCAL0,   pRfSettings->fscal0);
  WriteSingleReg(FSTEST,   pRfSettings->fstest);
  WriteSingleReg(TEST2,    pRfSettings->test2);
  WriteSingleReg(TEST1,    pRfSettings->test1);
  WriteSingleReg(TEST0,    pRfSettings->test0);
  WriteSingleReg(FIFOTHR,  pRfSettings->fifothr);
  WriteSingleReg(IOCFG2,   pRfSettings->iocfg2);
  WriteSingleReg(IOCFG0,   pRfSettings->iocfg0);
  WriteSingleReg(PKTCTRL1, pRfSettings->pktctrl1);
  WriteSingleReg(PKTCTRL0, pRfSettings->pktctrl0);
  WriteSingleReg(ADDR,     pRfSettings->addr);
  WriteSingleReg(PKTLEN,   pRfSettings->pktlen);
}

//
*************************************************************
*****************
// @fn          WritePATable
// @brief       Write data to power table
// @param       unsigned char value                 Value     to
write
// @return      none
//
*************************************************************
*****************
void WriteSinglePATable(unsigned char value)
{
                WDTCTL = WDTPW+WDTCNTCL+WDTIS2;
 while( !(RF1AIFCTL1 & RFINSTRIFG)){
                if(WDTCTL&0x0080)       {WDTCTL       =
WDTPW+WDTCNTCL+WDTIS2;}
 }
 RF1AINSTRW = 0x3E00 + value;               // PA Table
single write

 while( !(RF1AIFCTL1 & RFINSTRIFG)){
                if(WDTCTL&0x0080)       {WDTCTL       =
WDTPW+WDTCNTCL+WDTIS2;}
 }
 RF1AINSTRB = RF_SNOP;                  // reset PA_Table
pointer
 WDTCTL = WDTPW + WDTHOLD;
}

//
*************************************************************
*****************
// @fn          WritePATable
// @brief       Write to multiple locations in power table
// @param       unsigned char *buffer Pointer to the table of
values to be written
// @param       unsigned char count    Number of values to be
written
// @return      none
//
*************************************************************
*****************
void WriteBurstPATable(unsigned char *buffer, unsigned
char count)
{
 volatile char i = 0;
 WDTCTL = WDTPW+WDTCNTCL+WDTIS2;
 while( !(RF1AIFCTL1 & RFINSTRIFG)){
                if(WDTCTL&0x0080)       {WDTCTL       =
WDTPW+WDTCNTCL+WDTIS2;}
 }
 RF1AINSTRW = 0x7E00 + buffer[i];           // PA Table
burst write

 for (i = 1; i < count; i++)
 {
  RF1ADINB = buffer[i];              // Send data
  while (!(RFDINIFG & RF1AIFCTL1)){
                if(WDTCTL&0x0080)       {WDTCTL       =
WDTPW+WDTCNTCL+WDTIS2;}
  }     // Wait for TX to finish
 }
 i = RF1ADOUTB;                     // Reset RFDOUTIFG
flag which contains status byte

 while( !(RF1AIFCTL1 & RFINSTRIFG)){
                if(WDTCTL&0x0080)       {WDTCTL       =
WDTPW+WDTCNTCL+WDTIS2;}
 }
 RF1AINSTRB = RF_SNOP;                  // reset PA Table
pointer
 WDTCTL = WDTPW + WDTHOLD;
}
```

EK-1.5. RFfunctions.c

```
#include "cc430x613x.h"
#include "D:\ccs projects\cc430f6137.h"
#include "RF1A.h"
//#include                "C:\Program                Files\IAR
Systems\projects\UHFVIS\HAL\RF1A.h"
//#include                "C:\Program                Files\IAR
Systems\projects\UHFVIS\HAL\hal_pmm.h"
//#include                "C:\Program                Files\IAR
Systems\projects\UHFVIS\HAL\crypto.c"
//#include                "C:\Program                Files\IAR
Systems\projects\UHFVIS\HAL\RfRegSettings.c"
#include <string.h>



/*******************
 * Function Definition
 */
/*
void Transmit(unsigned char *buffer, unsigned char length);
void ReceiveOn(void);
void ReceiveOff(void);

void InitButtonLeds(void);
void InitRadio(void);
*/


#define  PACKET_LEN        (25)       // PACKET_LEN <=
61
#define  RSSI_IDX          (PACKET_LEN+1)  // Index of
appended RSSI
#define  CRC_LQI_IDX       (PACKET_LEN+2)  // Index of
appended LQI, checksum
#define  CRC_OK            (BIT7)          // CRC_OK bit
#define  PATABLE_VAL       (0)//(0x51)          // 0 dBm
output
```

```
extern RF_SETTINGS rfSettings;
extern int calcrc(char *ptr,char length,long crc_ini);
extern void make_for_Crypto_buf(char *ptr, char length);
extern void Encode(long* data, char dataLength) ;
extern void Decode(long* data, char dataLength) ;
extern void Encode_TAG(long* data, char dataLength);
extern void make_for_buf(char *ptr, char length);
extern long temp_sta;
extern char radio_state;
extern long Crypto_BUF[16];
unsigned char packetReceived;
unsigned char packetTransmit;

unsigned char RxBuffer[250];
unsigned char RxBufferLength = 0;
//                CHN , to,  MSGTYPE/ACK
char                                    FIRSTSETUP[9]=
{9,0x00,0x31,0x20,0x31,0x53,'F','S','T'};
char    ACKMSG[25]={25,0x00,    0x31,0x11,0x31,    0x53,
0x4D, 0x45, 0x50, 0x53, 0x41,0x4E, 0x30, 0x30, 0x34, 0x32, 0x20, 0x42,
0x56, 0x4B, 0x20, 0x38, 0x31,0x00,0x00};
char    TAGIDMSG[40]={25,0x00,    0x31,0x31,0x31,    0x53,
0x4D, 0x45, 0x50, 0x53, 0x41,0x4E, 0x30, 0x30, 0x34, 0x32, 0x20, 0x42,
0x56, 0x4B, 0x20, 0x38, 0x31,0x00,0x00};
char   POWERMSG[25]={25,0x00,   0x31,0x20,0x31,   0x53,
0x4D, 0x45, 0x50, 0x53, 0x41,0x4E, 0x30, 0x30, 0x34, 0x32, 0x20, 0x42,
0x56, 0x4B, 0x20, 0x38, 0x31,0x00,0x00};
char   ACTMSG[25]=     {25,0x00,   0x31,0x61,0x31,   0x53,
0x4D, 0x45, 0x50, 0x53, 0x41,0x4E, 0x30, 0x30, 0x34, 0x32, 0x20, 0x42,
0x56, 0x4B, 0x20, 0x38, 0x31,0x00,0x00};
char    READCNFG_MSG[25]={25,0x00,    0x30,0x82,0x31,
0x53, 0x4D, 0x45, 0x50, 0x53, 0x41,0x4E, 0x30, 0x30, 0x34, 0x32, 0x20,
0x42, 0x56, 0x4B, 0x20, 0x38, 0x31,0x00,0x00};
char   RESETMSG[25]={25,0x00,   0x30,0x90,0x31,   0x53,
0x4D, 0x45, 0x50, 0x53, 0x41,0x4E, 0x30, 0x30, 0x34, 0x32, 0x20, 0x42,
0x56, 0x4B, 0x20, 0x38, 0x31,0x00,0x00};

char TxBuffer[63];
unsigned char buttonPressed = 0;
unsigned int i = 0;
int TX_POWER =0,TX_POWER_1=0;
long RX_POWER,RX_POWER1_;
long RX_POWER_MAX;
unsigned char transmitting = 0;
unsigned char receiving = 0;

void InitButtonLeds(void)
{
 // Set up the button as interruptible
        P1DIR &= ~BIT7;//// input
        P1REN |=BIT7;////pull up
        P1IES &= ~ BIT7;
        P1IFG = 0;
        P1OUT |= BIT7;
        P1IE  |= BIT7;

        P1DIR &= ~BIT4;//// input
        P1REN &=~BIT4;////pull up
        P1IES |= BIT4;
        P1IFG = 0;
        P1OUT |= BIT4;
        P1IE  |= BIT4;

/*      P1DIR &= ~BIT6;//// input
        P1REN &=~BIT6;////pull up
        P1IES &=~ BIT6;
        P1IFG = 0;
        P1OUT |= BIT6;
        P1IE  |= BIT6;

        P1DIR &= ~BIT5;//// input
        P1REN |=BIT5;////pull up
        P1IES |= BIT5;
        P1IFG = 0;
        P1OUT |= BIT5;
        P1IE  |= BIT5;
*/
          // Initialize Port J
        PJOUT = 0x00;
        PJDIR = 0xFF;

        // Set up LEDs
        P1OUT &= ~BIT0;
        P1DIR |= BIT0;
        P3OUT &= ~BIT6;
        P3DIR |= BIT6;
        P3OUT &= ~BIT4;
        P3DIR |= BIT4;
}

//////////////////////////////////////////////////////7


//////////////////////////////////////////////////////
void InitRadio(void)
{
 // Set the High-Power Mode Request Enable bit so LPM3
can be entered
 // with active radio enabled
 PMMCTL0_H = 0xA5;
 PMMCTL0_L |= PMMHPMRE_L;
 PMMCTL0_H = 0x00;

 WriteRfSettings(&rfSettings);

 WriteSinglePATable(TX_POWER);
}

void Transmit( char *buffer,  char length)
{
 int temp_crc_value=0;
 int temp_crc_value1=0;
 char temp_buffer_transmit[63];
                // Enable TX end-of-packet interrupt
 memcpy(temp_buffer_transmit,buffer,length);


//if(temp_buffer_transmit[2]<250)temp_buffer_transmit[2]++;
 //else temp_buffer_transmit[2]=0x31;

 temp_crc_value=calcrc(temp_buffer_transmit,length-3,0);
 temp_crc_value1=  temp_crc_value;
 temp_buffer_transmit[length-
1]=(char)temp_crc_value&0x00ff;
 temp_crc_value1>>=8;
 temp_buffer_transmit[length-
2]=(char)temp_crc_value1&0x00ff;

 make_for_Crypto_buf(temp_buffer_transmit,length-1);
 Encode(Crypto_BUF,length/4);
 make_for_buf(temp_buffer_transmit,length-1);
 make_for_Crypto_buf(temp_buffer_transmit,length-1);


 RF1AIES |= BIT9;
 RF1AIFG &= ~BIT9;                        // Clear pending
interrupts
 RF1AIE |= BIT9;
 WriteBurstReg(RF_TXFIFOWR,      temp_buffer_transmit,
length+1);
 Strobe(RF_SIDLE);
 Strobe( RF_STX );                // Strobe STX
 radio_state=RF_STX;

 //Decode(Crypto_BUF,length/4);
 //make_for_buf(temp_buffer_transmit,length-1);
 //transmitting=0;

}


void Transmit_TAG( char *buffer,  char length)
{
 int temp_crc_value=0;
 int temp_crc_value1=0;
 char temp_buffer_transmit[63];
                // Enable TX end-of-packet interrupt
 memcpy(temp_buffer_transmit,buffer,length);


//if(temp_buffer_transmit[4]<250)temp_buffer_transmit[4]++;
 //else temp_buffer_transmit[4]=0x31;

 temp_crc_value=calcrc(temp_buffer_transmit,length-3,0);
 temp_crc_value1=  temp_crc_value;
 temp_buffer_transmit[length-
1]=(char)temp_crc_value&0x00ff;
```

```
            temp_crc_value1>>=8;
            temp_buffer_transmit[length-
2]=(char)temp_crc_value1&0x00ff;


            make_for_Crypto_buf(temp_buffer_transmit,25);
            Encode_TAG(Crypto_BUF,25/4);
            make_for_buf(temp_buffer_transmit,25);
            //make_key_for_buf(temp_buffer_transmit);
            //make_for_Crypto_buf(temp_buffer_transmit,4,2);
            // Decode_TAG(Crypto_BUF,1);
            // make_for_buf(temp_buffer_transmit,4,2);

            RF1AIES |= BIT9;
            RF1AIFG &= ~BIT9;                       // Clear pending
interrupts
            RF1AIE |= BIT9;
            // RF1AIFG &= ~BIT9;
            WriteBurstReg(RF_TXFIFOWR,      temp_buffer_transmit,
length+1);
            Strobe(RF_SIDLE);
            Strobe( RF_STX );                       // Strobe STX
            radio_state=RF_SFTX;

            //Decode(Crypto_BUF,length/4);
            //make_for_buf(temp_buffer_transmit,length-1);
            //transmitting=0;

          }

          void ReceiveOn(void)
          {
            RF1AIES |= BIT9;                        // Falling edge of
RFIFG9
            RF1AIFG &= ~BIT9;                       // Clear a pending
interrupt
            RF1AIE  |= BIT9;                        // Enable the interrupt
            //RF1AIFG &= ~BIT9;
            // Radio is in IDLE following a TX, so strobe SRX to enter
Receive Mode
            ///Strobe( RF_SIDLE );
            temp_sta=Strobe( RF_SRX );//orginali
            //Strobe( RF_SIDLE );
          }

          void ReceiveOff(void)
          {
            RF1AIE &= ~BIT9;                        // Disable RX interrupts
            RF1AIFG &= ~BIT9;                       // Clear pending IFG

            // It is possible that ReceiveOff is called while radio is
receiving a packet.
            // Therefore, it is necessary to flush the RX FIFO after
issuing IDLE strobe
            // such that the RXFIFO is empty prior to receiving a packet.
            Strobe( RF_SIDLE );
            Strobe( RF_SFRX  );
          }
```

EK-1.6. RFRegSettings.c

```
          #include "RF1A.h"
          #define MHZ_868
          #ifdef MHZ_915

          // Chipcon
          // Product = CC430Fx13x
          // Chip version = C   (PG 0.7)
          // Crystal accuracy = 10 ppm
          // X-tal frequency = 26 MHz
          // RF output power = 0 dBm
          // RX filterbandwidth = 101.562500 kHz
          // Deviation = 19 kHz
          // Datarate = 38.383484 kBaud
          // Modulation = (1) GFSK
          // Manchester enable = (0) Manchester disabled
          // RF Frequency = 914.999969 MHz
          // Channel spacing = 199.951172 kHz
          // Channel number = 0
          // Optimization = -
          // Sync mode = (3) 30/32 sync word bits detected
          // Format of RX/TX data = (0) Normal mode, use FIFOs for
RX and TX
          // CRC operation = (1) CRC calculation in TX and CRC
check in RX enabled
          // Forward Error Correction =
          // Length configuration = (1) Variable length packets, packet
length configured by the first received byte after sync word.
          // Packetlength = 61
          // Preamble count = (2)  4 bytes
          // Append status = 1
          // Address check = (0) No address check
          // FIFO autoflush = 0
          // Device address = 0
          // GDO0 signal selection = ( 6) Asserts when sync word has
been sent / received, and de-asserts at the end of the packet
          // GDO2 signal selection = (41) RF_RDY
          RF_SETTINGS rfSettings = {
             0x08,   // FSCTRL1   Frequency synthesizer control.
             0x00,   // FSCTRL0   Frequency synthesizer control.
             0x23,   // FREQ2     Frequency control word, high byte.
             0x31,   // FREQ1     Frequency control word, middle byte.
             0x3B,   // FREQ0     Frequency control word, low byte.
             0xCA,   // MDMCFG4   Modem configuration.
             0x83,   // MDMCFG3   Modem configuration.
             0x93,   // MDMCFG2   Modem configuration.
             0x22,   // MDMCFG1   Modem configuration.
             0xF8,   // MDMCFG0   Modem configuration.
             0x00,   // CHANNR    Channel number.
             0x34,   // DEVIATN   Modem deviation setting (when FSK
modulation is enabled).
             0x56,   // FREND1    Front end RX configuration.
             0x10,   // FREND0    Front end TX configuration.
             0x18,   // MCSM0     Main Radio Control State Machine
configuration.
             0x16,   // FOCCFG    Frequency Offset Compensation
Configuration.
             0x6C,   // BSCFG     Bit synchronization Configuration.
             0x43,   // AGCCTRL2  AGC control.
             0x40,   // AGCCTRL1  AGC control.
             0x91,   // AGCCTRL0  AGC control.
             0xE9,   // FSCAL3    Frequency synthesizer calibration.
             0x2A,   // FSCAL2    Frequency synthesizer calibration.
             0x00,   // FSCAL1    Frequency synthesizer calibration.
             0x1F,   // FSCAL0    Frequency synthesizer calibration.
             0x59,   // FSTEST    Frequency synthesizer calibration.
             0x81,   // TEST2     Various test settings.
             0x35,   // TEST1     Various test settings.
             0x09,   // TEST0     Various test settings.
             0x47,   // FIFOTHR   RXFIFO and TXFIFO thresholds.
             0x29,   // IOCFG2    GDO2 output pin configuration.
             0x06,   // IOCFG0    GDO0 output pin configuration. Refer
to SmartRF® Studio User Manual for detailed pseudo register
explanation.
             0x04,   // PKTCTRL1  Packet automation control.
             0x04,   // PKTCTRL0  Packet automation control.
             0x00,   // ADDR      Device address.
             0x78    // PKTLEN    Packet length.
          };

          #elif defined MHZ_868

          // Chipcon
          // Product = CC430Fx13x
          // Chip version = C   (PG 0.7)
          // Crystal accuracy = 10 ppm
          // X-tal frequency = 26 MHz
          // RF output power = 0 dBm
          // RX filterbandwidth = 101.562500 kHz
          // Deviation = 19 kHz
          // Datarate = 38.383484 kBaud
          // Modulation = (1) GFSK
          // Manchester enable = (0) Manchester disabled
          // RF Frequency = 867.999939 MHz
          // Channel spacing = 199.951172 kHz
          // Channel number = 0
          // Optimization = -
          // Sync mode = (3) 30/32 sync word bits detected
          // Format of RX/TX data = (0) Normal mode, use FIFOs for
RX and TX
          // CRC operation = (1) CRC calculation in TX and CRC
check in RX enabled
          // Forward Error Correction =
          // Length configuration = (1) Variable length packets, packet
length configured by the first received byte after sync word.
          // Packetlength = 61
          // Preamble count = (2)  4 bytes
```

```
            // Append status = 1
            // Address check = (0) No address check
            // FIFO autoflush = 0
            // Device address = 0
            // GDO0 signal selection = ( 6) Asserts when sync word has been sent / received, and de-asserts at the end of the packet
            // GDO2 signal selection = (41) RF_RDY
            RF_SETTINGS rfSettings = {
               0x08,   // FSCTRL1   Frequency synthesizer control.
               0x00,   // FSCTRL0   Frequency synthesizer control.
               0x21,   // FREQ2     Frequency control word, high byte.
               0x62,   // FREQ1     Frequency control word, middle byte.
               0x76,   // FREQ0     Frequency control word, low byte.
               0xCA,   // MDMCFG4   Modem configuration.
               0x83,   // MDMCFG3   Modem configuration.
               0x93,   // MDMCFG2   Modem configuration.
               0x22,   // MDMCFG1   Modem configuration.
               0xF8,   // MDMCFG0   Modem configuration.
               0x00,   // CHANNR    Channel number.///// orginal=0;
               0x34,   // DEVIATN   Modem deviation setting (when FSK modulation is enabled).
               0x56,   // FREND1    Front end RX configuration.
               0x10,   // FREND0    Front end TX configuration.
               0x18,    // MCSM0      Main Radio Control State Machine configuration.
               0x16,    // FOCCFG      Frequency Offset Compensation Configuration.
               0x6C,   // BSCFG     Bit synchronization Configuration.
               0x43,   // AGCCTRL2  AGC control.
               0x40,   // AGCCTRL1  AGC control.
               0x93,   // AGCCTRL0  AGC control.
               0xE9,   // FSCAL3    Frequency synthesizer calibration.
               0x2A,   // FSCAL2    Frequency synthesizer calibration.
               0x00,   // FSCAL1    Frequency synthesizer calibration.
               0x1F,   // FSCAL0    Frequency synthesizer calibration.
               0x59,   // FSTEST    Frequency synthesizer calibration.
               0x81,   // TEST2     Various test settings.
               0x35,   // TEST1     Various test settings.
               0x09,   // TEST0     Various test settings.
               0x47,   // FIFOTHR   RXFIFO and TXFIFO thresholds.
               0x29,   // IOCFG2    GDO2 output pin configuration.
               0x06,   // IOCFG0    GDO0 output pin configuration. Refer to SmartRF® Studio User Manual for detailed pseudo register explanation.
               0x04,   // PKTCTRL1  Packet automation control.
               0x04,   // PKTCTRL0  Packet automation control.
               0x00,   // ADDR      Device address.
               0x1a    // PKTLEN    Packet length.
            };

            #endif

            #if !defined (MHZ_868) && !defined (MHZ_915)
            #error "Please select MHZ_868 or MHZ_915 as the active project configuration"
            #endif
```

EK-1.7. Timerfunctions.c

```
            #include "cc430x613x.h"
            #include "D:\ccs projects\cc430f6137.h"
            int Timer_Count=0;
            void Enable_Start_Timer(int timer_count)
            {
              Timer_Count=timer_count;
              TA1CCTL0 = CCIE;                          // CCR0 interrupt enabled
              TA1CCR0 = timer_count;
              TA1CTL = TASSEL_2 + MC_2 + TACLR+ID_3;          // SMCLK, contmode, clear TAR
            }
```

EK-2 Okuyucu Ünite

EK-2.1. main.c

```
            /*S
             * main.c
             */

            #include "D:\ccs projects\UHFVISNOZZLE_03082012\UHFVISNOZZLE_03082012.h"
            #include "D:\ccs projects\cc430f6137.h"
            #include "D:\ccs projects\include\string.h"
            #define Tx_power_for_acc          0xc0
            #define Tx_power_for_tag          0xc0//ad

            void main (void)
            {

                  WDTCTL = WDTPW+WDTHOLD;

            ////////////////////////////////////////////////////////////////////////////////////////////////////
                  ////////////////////////////////////// Config regleri okuma//////////////////////////////////////////////////////
                  read_MY_SEG();
                  temp_crc_value=0;temp_crc_value1=0;
                  temp_crc_value=calcrc(MY_ID,22,0);
                  temp_crc_value1=  temp_crc_value;
                  temp_crc_value&=0x000000ff;
                  temp_crc_value1>>=8;
                  temp_crc_value1&=0x000000ff;
                  if(MY_ID[22]==  temp_crc_value1  && MY_ID[23]== temp_crc_value)
                  {
                        my_fn=MY_ID[0]*0x10  + MY_ID[1];
                        ACCMSG[1]=my_fn;

            if((MY_ID[0])>6)MY_CHN=((MY_ID[0])-6);
                        else MY_CHN=(MY_ID[0]);
                        change_value=MY_ID[2];

                        read_MY_SEGD();

            temp_crc_value=0;temp_crc_value1=0;

            temp_crc_value=calcrc(MY_ID,22,0);
                        temp_crc_value1=  temp_crc_value;
                        temp_crc_value&=0x000000ff;
                        temp_crc_value1>>=8;
                        temp_crc_value1&=0x000000ff;
                        if(MY_ID[22]==  temp_crc_value1 && MY_ID[23]== temp_crc_value)
                        {

                        }
```

```
                                else
                                {
                                        copy_C2D();
                                        read_MY_SEG();
                                }
                        }
                        else
                        {
                                read_MY_SEGD();
temp_crc_value=0;temp_crc_value1=0;
temp_crc_value=calcrc(MY_ID,22,0);
                                temp_crc_value1=
temp_crc_value;
                                temp_crc_value&=0x000000ff;
                                temp_crc_value1>>=8;
                                temp_crc_value1&=0x000000ff;
                                if(MY_ID[22]==  temp_crc_value1
&& MY_ID[23]== temp_crc_value)
                                {
my_fn=MY_ID[0]*0x10 + MY_ID[1];
                                        ACCMSG[1]=my_fn;
if((MY_ID[0])>6)MY_CHN=((MY_ID[0])-6);
                                        else
MY_CHN=(MY_ID[0]);
change_value=MY_ID[2];
                                         copy_D2C();
                                }
                        }


                        //7my_fn=0x0b;
                         //my_fn=MY_ID[0]*0x10 + MY_ID[1];

                        /*
                         * for without acc
                         */
                        ////////////////////for                        without
acc////////////////////////////////////
                        /* my_fn=0xB3;
                         MY_ID[0]=0x0B;
                         MY_ID[1]=0x03;
                         MY_ID[2]=12;*/
                        ////////////////////////////////////////////////////
                         //my_fn=0x11;
                        // power_up=0x55;
                         ACCMSG[1]=my_fn;
                         if((MY_ID[0])>6)MY_CHN=((MY_ID[0])-6);
                         else MY_CHN=(MY_ID[0]);
                                change_value=MY_ID[2];

                        ////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////

                                ////////////////////////////////dont        work        when
broken//////////////////////////////////
                                /*
                                if(power_up!=0x55)
                                {
                                        for(main_i=0;main_i<20;main_i++)
                                        {
                                                MY_ID[main_i]=0xff;
                                        }
                                        my_fn=MY_ID[0]*0x10             +
MY_ID[1];
                                        ACCMSG[1]=my_fn;

                        if((MY_ID[0])>6)MY_CHN=((MY_ID[0])-6);
                                        else MY_CHN=(MY_ID[0]);
                                        change_value=MY_ID[2];
                                        __bis_SR_register(  GIE );
                                        write_MY_SEG();
                                        copy_C2D();
                                        __bic_SR_register(  GIE );
                                }


                                */


////////////////////////////////////////////////////////////////////////////

                                 TX_POWER=0xc0;
                                rfSettings.channr=0;
                                rfSettings.mdmcfg4=0xca;
                                SetVCore(2);
                                ResetRadioCore();
                                InitRadio();
                                InitButtonLeds();
                                ReceiveOff();
                                receiving=0;
                                radio_state=RF_SFRX;
                                buttonPressed=0;
                                first_talk=1;
                                //Get_KEY=0;

                                //power_up=0x55;//////////////////for without acc

                                for(main_i=0;main_i<250;main_i++)
                                {
                                        RxBuffer[main_i]=0;
                                }
                                if(power_up!=0x55)
                                {
                                        P3OUT|=BIT4;
                                        power_up=0;
                                        rfSettings.channr=0;
                                        rfSettings.mdmcfg4=0xca;
                                        ResetRadioCore();
                                        InitRadio();
                                        ReceiveOn();
                                        radio_state=RF_SRX;
                                        receiving=1;
                                        transmitting=0;
                                        timer_over=0;
                                        __bis_SR_register(  GIE );
                                        Enable_Start_Timer(500);
                                }

                                else
                                {
                                        timer_over=0;

                                        first_talk=0;
                                        multi_timer_over=0;
                                        second_talk=0;
                                        check_button=1;

                                        Enable_Start_Timer(10);
                                        can_sleep=0;
                                }

                                while (1)
                        {

                                __bis_SR_register(  GIE );///uykuyu ekle

                                if(id_change)
                                {
                                        // MY_ID[18]=(char)RX_POWER;
//MY_ID[19]=(char)~RX_POWER;

//MY_ID[20]=(char)RX_POWER_MAX;

//MY_ID[21]=(char)~RX_POWER_MAX;
                                         //MY_ID[24]=0xff;
                                        //
                                         __bis_SR_register(                GIE
);///uykuyu ekle
                                         Message_Waiting=0;
                                         ReceiveOff();
                                         radio_state=RF_SFRX;

                                         id_change=0;

for(main_i=0;main_i<17;main_i++)
                                         {
```

```
NOZIDMSG[6+main_i]=MY_ID[main_i];
                }
                Message_Waiting=0;

Transmit_NO_key(NOZIDMSG,25);
                transmitting=1;
                P3OUT |= BIT6;
                P3OUT |= BIT4;
                can_sleep=0;
                write_MY_SEG();
                copy_C2D();
                {WDTCTL                 =
WDTPW+WDTCNTCL+WDTIS2;}
                while(1);

        }

////////////////////////////////////////////////////////////

        if(can_sleep && power_up==0x55)
        {
//              TA1CCTL0 = ~CCIE;

                if(radio_state!=RF_SXOFF)
                {
                        Strobe(RF_SIDLE);
                        Strobe(RF_SXOFF);
                }
                radio_state=RF_SXOFF;
                __bis_SR_register(LPM3_bits );
                can_sleep=0;
                Get_KEY_count=0;
                multi_timer_over=0;
                transmitting=0;
        }
        Get_KEY=1;
        first_meet=1;
        //Get_KEY=0;////////////////for without acc
       //  buttonPressed=1;
        power_msg_got=0;
        while (buttonPressed)
        {

                if(power_up==0x55
&&((MY_CHN&0xf0)!=0 || second_talk>1 || first_talk>1 ))
                {
                        {WDTCTL         =
WDTPW+WDTCNTCL+WDTIS2;}
                        while(1);
                }
                if((P1IN        &       BIT7)       &&
(!check_button_up))
                {
                        check_button_up=1;

Enable_Start_Timer(100);
                        //Timer_Count=100;
                        //TA1CCR0           =
Timer_Count;
                }

////////////////////////////////////////////////////////////////////////
                if(id_change)
                {
                        //
MY_ID[18]=(char)RX_POWER;

//MY_ID[19]=(char)~RX_POWER;

//MY_ID[20]=(char)RX_POWER_MAX;

//MY_ID[21]=(char)~RX_POWER_MAX;
                        //MY_ID[24]=0xff;
                        __bis_SR_register(
GIE );///uykuyu ekle
                        Message_Waiting=0;
                        ReceiveOff();

radio_state=RF_SFRX;
                        receiving=0;
                        id_change=0;

for(main_i=0;main_i<17;main_i++)
                        {

NOZIDMSG[6+main_i]=MY_ID[main_i];
                        }
                        Message_Waiting=0;

Transmit_NO_key(NOZIDMSG,25);
                        transmitting=1;
                        P3OUT |= BIT6;
                        P3OUT |= BIT4;
                        can_sleep=0;
                        while(transmitting);
                          write_MY_SEG();
                          copy_C2D();
                        {WDTCTL         =
WDTPW+WDTCNTCL+WDTIS2;}
                        while(1);

                        }

////////////////////////////////////////////////////////////////////////

                ////////////////////    Herşey
Başlıyor//////////////////////////
                if(timer_over && !transmitting)
                {
                        timer_over=0;
                                                ////    hüseyin     abinin
algoritmayı da kullanabilirim//////////////////
                        if(receive_timeout)
                        {

stop_receive=1;

receive_timeout=0;

if(radio_state!=RF_SFRX)
                                {

                        ReceiveOff();

                        radio_state=RF_SFRX;
                                }

Strobe(RF_SIDLE);

Strobe(RF_SXOFF);

radio_state=RF_SXOFF;


receiving=0;

if(first_meet)
                                {

                        Timer_Count=300;

                        TA1CCR0 = Timer_Count;
                                }
                                else
                                {


Timer_Count=1300;

TA1CCR0 = Timer_Count;
                                }

//send_acc=1;
                        }

////////////////////////////////////////////////////////////////////
                                                //
Get_KEY=0;////////////////////for without acc
                        if(power_up!=0x55)
                        {
                                P3OUT |=
BIT4;

if(power_up_count>100)
                                {
```

```
            multi_timer_over=29;
            power_up=0x55;
            power_up_count=0;
            P3OUT &=~ BIT4;
                                    }
                                    else
                                    {

multi_timer_over=0;
power_up_count++;
Enable_Start_Timer(300);
/*TA1CCTL0 |= CCIE;
Timer_Count=300;
TA1CCR0 += Timer_Count;*/
                                    }

                                }

////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////

                                else if(Get_KEY  &&
power_up==0x55)
                                {

if(radio_state!=RF_SFRX)
                                    {
            ReceiveOff();
            radio_state=RF_SFRX;
                                    }

receiving=0;
rfSettings.channr=MY_CHN;
rfSettings.mdmcfg4=0xc5;
TX_POWER=Tx_power_for_acc;
ResetRadioCore();
InitRadio();
            P3OUT |=
BIT6;
            P3OUT |=
BIT4;
ACCMSG[3]=0x40;

Transmit_NO_ENCODE(ACCMSG,25);

transmitting=1;
Wait_KEY=1;
Wait_KEY=0;//// for without acc;
Get_KEY=0;/// for without acc
Get_KEY_count++;
if(Get_KEY_count>20)
                                    {
            multi_timer_over=29;
                                    }
                                    else

multi_timer_over=0;

Timer_Count=200;
TA1CCR0 += Timer_Count;
TA1CCTL0 = CCIE;
power_up=0x55;
                                //
Get_KEY=0;//// for without access point
                                    }
                                    else if(first_talk)/// ilk
karşılaşma
                                    {

if(rfSettings.channr!=0x00 ||       rfSettings.mdmcfg4!=0xcA)
                                        {
            rfSettings.channr=0x00;
            rfSettings.mdmcfg4=0xcA;
            TX_POWER=Tx_power_for_tag;
            ResetRadioCore();
            InitRadio();
                                        }

if(radio_state!=RF_SFRX)
                                        {
            ReceiveOff();
            radio_state=RF_SFRX;
                                        }

receiving=0;
            P3OUT |=
BIT6;
            P3OUT |=
BIT4;
            WDTCTL
= WDTPW+WDTCNTCL+WDTIS2;

////////////////////////////////////////////

if(power_msg_got)
                                        {
            Generate_Key();
            Generate_Delta();
            make_key_for_buf(ACKMSG);
            WDTCTL = WDTPW + WDTHOLD;
            Transmit(ACKMSG,25);
                                        }
                                        else
                                        {
            WDTCTL = WDTPW + WDTHOLD;
            POWERMSG[2]=0x30;
            Transmit(POWERMSG,25);

                                        }
            ////////////////////////////////////////////

transmitting=1;
time_out_for_receive=1;
//Get_KEY_count=0;
```

```
                                        //
TA1CCTL0 = CCIE;                        // CCR0 interrupt enabled
multi_timer_over++;
multi_timer_over_tag++;
stop_receive=0;
Timer_Count=500;
TA1CCR0 += Timer_Count;
power_up=0x55;
                                    }
                                    else
if(second_talk)//daha sonraki karşılaşma
                                    {
if(rfSettings.channr!=0x00 ||       rfSettings.mdmcfg4!=0xcA)
                                                {
        rfSettings.channr=0x00;
        rfSettings.mdmcfg4=0xcA;
        TX_POWER=Tx_power_for_tag;
        ResetRadioCore();
        InitRadio();
                                                }
if(radio_state!=RF_SFRX)
                                                {
        ReceiveOff();
        radio_state=RF_SFRX;
                                                }
                                                //
Get_KEY_count=0;
receiving=0;
                                                P3OUT |=
BIT6;
                                                P3OUT |=
BIT4;
                                                WDTCTL
= WDTPW+WDTCNTCL+WDTIS2;
Generate_Key();
Generate_Delta();
make_key_for_buf(ACKMSG);
                                                WDTCTL
= WDTPW + WDTHOLD;
Transmit(ACKMSG,25);
transmitting=1;
time_out_for_receive=1;
multi_timer_over++;
multi_timer_over_tag++;
                                        //
TA1CCTL0 = CCIE;                        // CCR0 interrupt enabled
stop_receive=0;
Timer_Count=500;
TA1CCR0 += Timer_Count;
power_up=0x55;
                                    }
if(multi_timer_over_tag>1 && (multi_timer_over_tag%10)==0)/// 10 sn
ve katları
                                    {
for(main_i=0;main_i<17;main_i++)
                                                {
        ACKMSG[6+main_i]=0xff;
        ACCMSG[6+main_i]=0xff;
                                                }
if(first_data_came)
                                                {
        first_data_came=0;
        multi_timer_over=10;
        ////////fuar//////////
        multi_timer_over_tag=6;
        ////////////////////
                                                }
first_talk=1;
second_talk=0;
if(!first_meet)send_acc=1;
                                    }
if(((multi_timer_over>1   &&   (multi_timer_over)>28))   ||   (button_up))//
(multi_timer_over%29)==0) || (button_up)) //30 sn doldu bye bye
                                                {
wrong_alarm=0;
if(button_up)
                                                        {
        while(main_i1<0xfffb)
        {
                if(!(P1IN&BIT7))
                {
                        wrong_alarm=1;
                        main_i1=0xfffd;
                }
                main_i1++;
        }
        main_i1=0;
        button_up=0;
       // return;
                                                }
if(!wrong_alarm || ((multi_timer_over)>28) )
                                                {
        button_up=0;
        InitButtonLeds();
        buttonPressed=0;
        check_button_up=0;
        button_up=0;
        TA1CCTL0 = ~CCIE;
//Get_KEY_count=0;
        if(radio_state!=RF_SFRX)
```

```
        {
                ReceiveOff();
                radio_state=RF_SFRX;
        }
        receiving=0;
        can_sleep=1;
                                }
                        }
                }
                else if(Get_KEY && !transmitting)
                {
if(radio_state!=RF_SRX)
                        {
ReceiveOn();
radio_state=RF_SRX;
receiving=1;
transmitting=0;
                        }
                }
                else if(send_acc && !transmitting )
                {
if(radio_state!=RF_SFRX)
                        {
ReceiveOff();
radio_state=RF_SFRX;
                        }
                        receiving=0;
rfSettings.channr=MY_CHN;
rfSettings.mdmcfg4=0xc5;
TX_POWER=Tx_power_for_acc;
                        ResetRadioCore();
                        InitRadio();
                        P3OUT |= BIT6;
                        P3OUT |= BIT4;
                        ACCMSG[3]=0x11;
Transmit_ACC(ACCMSG,25);
                        transmitting=1;
                        send_acc=0;
                        stop_receive=1;
                        can_sleep=0;
                        Timer_Count=500;
                        TA1CCR0 += Timer_Count;
                }
                else if(!transmitting && !send_acc)
                {
if(rfSettings.channr!=0)
                {
if(radio_state!=RF_SFRX)
                        {
                ReceiveOff();
                radio_state=RF_SFRX;
                        }
receiving=0;
rfSettings.channr=0x00;
rfSettings.mdmcfg4=0xcA;
TX_POWER=Tx_power_for_tag;
ResetRadioCore();
InitRadio();
                }
                if(!stop_receive)
                {
if(radio_state!=RF_SRX)
                        {
                ReceiveOn();
                radio_state=RF_SRX;
                receiving=1;
                transmitting=0;
                        }
                }
                else
                        {
if(radio_state!=RF_SFRX)
                                {
                ReceiveOff();
                Strobe(RF_SXOFF);
                radio_state=RF_SXOFF;
                                }
receiving=0;
                        }
                        //can_sleep=1;
                }
        }
        }

        //////////////////////////////////////////////////////////////////////
        //                      INTERRUPTS                                  //
        //////////////////////////////////////////////////////////////////////
        #pragma vector=CC1101_VECTOR

        __interrupt void CC1101_ISR(void)

        {
          long temp_crc_value=0,temp_crc_value1=0;

          switch(__even_in_range(RF1AIV,32))        // Prioritizing
Radio Core Interrupt
          {
           case  0: break;                          // No RF core interrupt
pending
           case  2: break;                  // RFIFG0
           case  4: break;                  // RFIFG1
           case  6: break;                  // RFIFG2
           case  8: break;                  // RFIFG3
           case 10: break;                  // RFIFG4
           case 12: break;                  // RFIFG5
           case 14: break;                  // RFIFG6
           case 16: break;                  // RFIFG7
           case 18: break;                  // RFIFG8
           case 20:                         // RFIFG9
             if(receiving )                                 //      RX
end of packet
              {
                      ReadBurstReg(RF_RXFIFORD,
RxBuffer,0xff);// RxBufferLength);
                      if(RxBufferLength!=0x1c)
                      {
```

```
                __no_operation();
            }
            RxBufferLength=0x1c;
            if(                !(power_msg_got)&&
rfSettings.channr==0x00)
            {

                WDTCTL                =
WDTPW+WDTCNTCL+WDTIS2;

        make_for_Crypto_buf(RxBuffer,RxBufferLength-4);

        Decode(Crypto_BUF,25/4);//(RxBufferLength-3)/4);

        make_for_buf(RxBuffer,(RxBufferLength-4));
                WDTCTL    =    WDTPW    +
WDTHOLD;
            }

            else    if(rfSettings.channr==0x00    &&
(power_up==0x55))
            {
                WDTCTL                =
WDTPW+WDTCNTCL+WDTIS2;


        make_for_Crypto_buf(RxBuffer,RxBufferLength-4);

        Decode_TAG(Crypto_BUF,25/4);//(RxBufferLength-3)/4);

        make_for_buf(RxBuffer,(RxBufferLength-4));
                WDTCTL    =    WDTPW    +
WDTHOLD;
            }

            else
            {

                WDTCTL                =
WDTPW+WDTCNTCL+WDTIS2;

        make_for_Crypto_buf(RxBuffer,RxBufferLength-4);

        Decode(Crypto_BUF,25/4);//(RxBufferLength-3)/4);

        make_for_buf(RxBuffer,(RxBufferLength-4));
                WDTCTL    =    WDTPW    +
WDTHOLD;
            }

            if( RxBuffer[5]=='S')
          {
            temp_crc_value=calcrc(RxBuffer,(RxBufferLength-3)-
3,0);
            temp_crc_value1=  temp_crc_value;
            temp_crc_value&=0x000000ff;
            temp_crc_value1>>=8;
            temp_crc_value1&=0x000000ff;
            if((RxBuffer[(RxBufferLength-3)-
1]==temp_crc_value)    &&    (RxBuffer[(RxBufferLength-3)-2]    ==
temp_crc_value1) &&( RxBuffer[2]==0x31))
            {
              __no_operation();
              P1OUT ^= BIT0;              // Toggle LED1
              ///// Message correct
              for(int_rec_i=0;int_rec_i<17;int_rec_i++)
              {
                  temp_id[int_rec_i]=RxBuffer[6+int_rec_i];
              }

              switch(RxBuffer[3])
              {
              case 0x11:
              {

                  if(first_talk)
                  {
                        id_get=0;
for(int_rec_i=0;int_rec_i<17;int_rec_i++)
                        {
last_read_id[int_rec_i]=RxBuffer[6+int_rec_i];
ACCMSG[int_rec_i+6]=RxBuffer[int_rec_i+6];
ACKMSG[int_rec_i+6]=RxBuffer[int_rec_i+6];
if((RxBuffer[int_rec_i+6])!=0xff) id_get=1;
                        //else id_get=0;
                        }

                        ACCMSG[6]=RxBuffer[6];
                      //    ACCMSG[6]=RxBuffer[4];//for
debug
                        if(!big_rx_power_got)
                        {
if(((RxBuffer[4])<(change_value))&&(RxBuffer[26]>0x10    &&
RxBuffer[26]<0x30))//  if(id_get )//
                        {
first_meet=0;
Message_Waiting=1;
send_acc=1;
time_out_for_receive=0;
multi_timer_over=0;
multi_timer_over_tag=6;
first_talk=0;
second_talk=1;
                        }
                        }
                        else
                        {
if((abs(temp_rx_power-
RxBuffer[4])<(change_value))&&(RxBuffer[26]>0x10    &&
RxBuffer[26]<0x30))
                        {
first_meet=0;
Message_Waiting=1;
send_acc=1;
time_out_for_receive=0;
multi_timer_over=0;
multi_timer_over_tag=6;
first_talk=0;
second_talk=1;
                        }
                        }

                        //
if(ACKMSG[4]>250)ACKMSG[4]=0x31;
                        //  else ACKMSG[4]++;

                  }
                  else
                  {
                        {
                              id_get=0;
for(int_rec_i=0;int_rec_i<17;int_rec_i++)
                        {
last_read_id[int_rec_i]=RxBuffer[6+int_rec_i];
```

```
ACCMSG[int_rec_i+6]=RxBuffer[int_rec_i+6];

ACKMSG[int_rec_i+6]=RxBuffer[int_rec_i+6];

if((RxBuffer[int_rec_i+6])!=0xff) id_get=1;
                                        }

                                        ACCMSG[6]=RxBuffer[6];


                                    //    ACCMSG[6]=RxBuffer[4];///for
debug
                                        if(!big_rx_power_got)
                                        {

if(((RxBuffer[4])<(change_value))&&(RxBuffer[26]>0x10          &&
RxBuffer[26]<0x30))//  if(id_get )//

                                                {

first_meet=0;

Message_Waiting=1;

send_acc=1;

time_out_for_receive=0;

multi_timer_over=0;

multi_timer_over_tag=6;

first_talk=0;

second_talk=1;
                                                }
                                        }
                                        else
                                        {

if((abs(temp_rx_power-
RxBuffer[4])<(change_value))&&(RxBuffer[26]>0x10          &&
RxBuffer[26]<0x30))
                                                {

first_meet=0;

Message_Waiting=1;

send_acc=1;

time_out_for_receive=0;

multi_timer_over=0;

multi_timer_over_tag=6;

first_talk=0;

second_talk=1;
                                                        }
                                                }
                                                //send_acc=1;
                                               //
if(ACKMSG[4]>250)ACKMSG[4]=0x31;
                                                //              else
ACKMSG[4]++;
                                                }
                                        }
                                        first_data_came=1;
                                        break;

                                }
                                case 0x20: ///TX_POWER
                                {
                                        power_msg_got=1;
                                        if(RxBuffer[6]>128)
                                        {
                                                big_rx_power_got=1;
                                                temp_rx_power=RxBuffer[6]-
(0xff-RxBuffer[6]);
                                        }
                                        else
                                        {
                                                big_rx_power_got=0;
                                        }
                                                Message_Waiting=1;
                                        //TX_POWER=RxBuffer[6];
                                        //Reinit_radio=1;
                                        //first_talk=0;
                                        //second_talk=0;
                                        //can_sleep=0;
                                        break;
                                }
                                case 0x30:// ID_CHANGE
                                {
                                        Message_Waiting=1;
                                        for(int_rec_i=0;int_rec_i<18;int_rec_i++)
                                        {

MY_ID[int_rec_i]=RxBuffer[6+int_rec_i];

NOZIDMSG[int_rec_i+6]=RxBuffer[6+int_rec_i];
                                        }
                                        id_change=1;
                                        power_up=0x55;
                                        first_talk=0;
                                        second_talk=0;
                                        can_sleep=0;
                                        timer_over=1;
                                        buttonPressed=1;
                                        break;
                                }
                                case 0x40://GET KEY
                                {
                                        make_buf_for_key_ACC(RxBuffer);
                                        Generate_Delta();
                                        Get_KEY=0;
                                        Wait_KEY=0;
                                        break;
                                }
                                default:
                                {
                                        Message_Waiting=0;
                                        break;
                                }

                                }
                        }
                        else
                        {
                                Message_Waiting=0;
                        }
                }
                else
                {
                        Message_Waiting=0;
                }
                }
                else if(transmitting && !receiving)
                // TX end of packet
                {
                 RF1AIE &= ~BIT9;                // Disable TX end-of-
packet interrupt
                 P3OUT &= ~BIT6;                 // Turn off LED after
Transmit
                 P3OUT &= ~BIT4;
                 transmitting = 0;
                }
                else
                {
                        rfSettings.channr=0;
                        ResetRadioCore();
                        InitRadio();
                        ReceiveOn();
                        receiving = 1;
                        transmitting=0;

                }// trap
                break;
              case 22: break;           // RFIFG10
              case 24: break;           // RFIFG11
              case 26: break;           // RFIFG12
              case 28: break;           // RFIFG13
```

```
    case 30: break;                 // RFIFG14
    case 32: break;                 // RFIFG15
   }
   can_sleep=0;
   __bic_SR_register_on_exit(LPM3_bits);
}

//////////////////////////////////////////////////////////////////////////////
#pragma vector=PORT1_VECTOR
__interrupt void PORT1_ISR(void)
{
 switch(__even_in_range(P1IV, 16))
 {
   case  0: break;
   case  2: break;                 // P1.0 IFG
   case  4: break;                 // P1.1 IFG
   case  6: break;                 // P1.2 IFG
   case  8: break;                 // P1.3 IFG
   case 10: break;                 // P1.4 IFG
   case 12: break;                 // P1.5 IFG
   case 14: break;                 // P1.6 IFG
   case 16:                        // P1.7 IFG
    P1IE = 0;                      // Debounce by disabling
buttons
    check_button=1;
    Enable_Start_Timer(1000);
    multi_timer_over=0;
    __bic_SR_register_on_exit(LPM3_bits); // Exit active

    break;
 }
}

//////////////////////////////////////////////////////////////////////////////
#pragma vector=TIMER1_A0_VECTOR
__interrupt void TIMER1_A0_ISR(void)
{
        /*if(timer_cycle>1)
        {
                timer_cycle=0;
                P3OUT^=BIT4;
                TA1CCR0 += Timer_Count;
        }
        else
        {
                TA1CCR0 += Timer_Count;
                timer_cycle++;
        }*/

 if(timer_cycle>200)
 {
        timer_cycle=0;
        timer_over=1;
        if(time_out_for_receive)///receive     yi     kapat
uyumaya geç
        {
                timer_over=1;
                receive_timeout=1;
                time_out_for_receive=0;
__bic_SR_register_on_exit(LPM3_bits);
                if(check_button_up)
                {
                        Get_KEY_count=0;
                //
check_button_up=1;
                        timer_over=1;

                        button_up=1;
                }
                 return;
        }
        multi_timer_over++;
        can_sleep=0;
        if(check_button)//  &&  !(BIT7  &  P1IN)  )///
civalı switche göre ayarlanacak
        {
                timer_over=1;
                buttonPressed=1;
                check_button=0;
                multi_timer_over=0;
                TA1CCTL0 = ~CCIE;

                //Timer_Count=10;
                //TA1CCR0 += Timer_Count;
                Get_KEY_count=0;
                //Get_KEY=1;//unutma
                first_talk=1;
                can_sleep=0;
        }
        else if(check_button)//////////////yeni algoritmada
burası ölmüş haberim yok
        {

                timer_over=0;
                check_button=0;
                buttonPressed=0;
                multi_timer_over=0;
                TA1CCTL0 = ~CCIE;
                P1IFG = 0;
                InitButtonLeds();
                buttonPressed=0;
                first_talk=0;
                P1IFG = 0;
                can_sleep=1;
        }
        if(check_button_up)
        {
                Get_KEY_count=0;
        //      check_button_up=1;
                timer_over=1;
                button_up=1;
        }
        __bic_SR_register_on_exit(LPM3_bits);       //
Exit active
 }
 else
 {
  TA1CCR0 += Timer_Count;
  timer_cycle++;
  //can_sleep=1;
 }

}
```

EK-3 Merkezi Birim

EK-3.1. main.c

```
/*S
 * main.c
 */

#include "D:\ccs projects\UHFVISacc0207\UHFVISNOZZLE_051211.h"
#include "D:\ccs projects\cc430f6137.h"
#include "D:\ccs projects\include\string.h"
#include <stdio.h>
#include <stdlib.h>
#define Tx_power_for_acc 0x51
#define Tx_power_for_tag 0x02

////////////////////////////////////////////////////////////////////////////////
////////
void Uart_Send(void)
{

////////////////////////////////////////for debug////////////////////////////////////////////////////////

make_for_Crypto_buf_USC(Temp_BUF,16);

Encode_USC(Crypto_BUF,4);

make_for_buf_USC(Temp_BUF,16);
////////////////////////////////////////////////////////////////////////////////
////////////////////////////77

////////////////////////////////////////////////////////////////////////////////
//////////////////////////

uart_int_k=5;

for(uart_int_i=0;uart_int_i<16;uart_int_i++)

{

Uart_TX[uart_int_i+uart_int_k]=(((Temp_BUF[uart_int_i]>>4)&0x0f)+0x30);

uart_int_k++;

Uart_TX[uart_int_i+uart_int_k]=(((Temp_BUF[uart_int_i])&0x0f)+0x30);

//uart_int_k++;

}

//////////////////adding CRC////////////////////////////////

temp_crc_value=calcrc(Uart_TX,37,0);

temp_crc_value1= temp_crc_value;

temp_crc_value1>>=8;

temp_crc_value1 &= 0x00ff;

temp_crc_value &= 0x00ff;

if(((char)temp_crc_value&0x00ff)!=0xfa)

{

Uart_TX[37]=((char)temp_crc_value&0x00ff);

if(((char)temp_crc_value1&0x00ff)!=0xfa)

{

Uart_TX[38]=((char)temp_crc_value1&0x00ff);

Uart_TX[39]=0x03;

Uart_TX[40]=0xfa;

tx_int_count=0;

tx_count=40;

UCA0TXBUF = Uart_TX[0];

UCA0IE |= UCTXIE;

}

else

{

Uart_TX[38]=0x10;

Uart_TX[39]=((char)temp_crc_value1&0x00ff);

Uart_TX[40]=0x03;

Uart_TX[41]=0xfa;

tx_count=41;

tx_int_count=0;

UCA0TXBUF = Uart_TX[0];

UCA0IE |= UCTXIE;

}

}

else

{

Uart_TX[37]=0x10;//((char)temp_crc_value&0x00ff);

Uart_TX[38]=((char)temp_crc_value&0x00ff);

if(((char)temp_crc_value1&0x00ff)!=0xfa)

{
```

```
Uart_TX[39]=((char)temp_crc_value1&0x00ff);

Uart_TX[40]=0x03;

Uart_TX[41]=0xfa;

tx_count=41;


UCA0TXBUF = Uart_TX[0];
                                        UCA0IE |=
UCTXIE;

                                }

                                else

                                {


Uart_TX[39]=0x10;

Uart_TX[40]=((char)temp_crc_value1&0x00ff);

Uart_TX[41]=0x03;

Uart_TX[42]=0xfa;

tx_count=42;

tx_int_count=0;


UCA0TXBUF = Uart_TX[0];
                                        UCA0IE |=
UCTXIE;

                                }

                        }
                                //
make_for_Crypto_buf_USC(Temp_BUF,16);

                                //              Decode_USC(Crypto_BUF,4);

                                //
make_for_buf_USC(Temp_BUF,16);

}


////////////////////////////////////////////////////////////////////////////////
/////////


void test_RXMSG(void)
{


                        make_for_Crypto_buf(RxBuffer,RxBufferLengt
h-4);
    Decode_NOZ(Crypto_BUF,25/4);//(RxBufferLength-3)/4);
    make_for_buf(RxBuffer,(RxBufferLength-4));


                        if( RxBuffer[5]=='S')
    {
      temp_crc_value=calcrc(RxBuffer,(RxBufferLength-3)-
3,0);
      temp_crc_value1=  temp_crc_value;
      temp_crc_value&=0x000000ff;
      temp_crc_value1>>=8;
      temp_crc_value1&=0x000000ff;
      if((RxBuffer[(RxBufferLength-3)-1]==temp_crc_value)
&&  (RxBuffer[(RxBufferLength-3)-2]  ==  temp_crc_value1)  &&  (
RxBuffer[2]==0x32)                                          &&
(((RxBuffer[1]&0xf0)>>4)==((my_fn&0xf0)>>4)))               //
(RxBuffer[1]==my_fn))//(((RxBuffer[1]&0xf0)>>4)==MY_CHN))
      {
        __no_operation();
        P1OUT ^= BIT0;                  // Toggle LED1
        ///// Message correct

        receive_poll=0;
        switch(RxBuffer[3])
        {
                case 0x11:
                {

/*for(int_rec_k=0;int_rec_k<10;int_rec_k++)
                                {

for(int_rec_i=0;int_rec_i<17;int_rec_i++)
                                        {

nozzle_ids[int_rec_i][int_rec_k]=0;
                                        }
                                }
        */
                                int_rec_k=(RxBuffer[1]&0x0f);


for(int_rec_i=0;int_rec_i<int_rec_k;int_rec_i++)
                                {

for(int_rec_l=0;int_rec_l<17;int_rec_l++)
                                        {

nozzle_ids[int_rec_l][int_rec_i]=0;
                                        }
                                }


for(int_rec_i=int_rec_k+1;int_rec_i<10;int_rec_i++)
                                {

for(int_rec_l=0;int_rec_l<17;int_rec_l++)
                                        {

nozzle_ids[int_rec_l][int_rec_i]=0;
                                        }
                                }


for(int_rec_i=0;int_rec_i<17;int_rec_i++)
                                {

nozzle_ids[int_rec_i][int_rec_k]=RxBuffer[6+int_rec_i];
                                }



ReceiveOff();

radio_state=RF_SFRX;

receiving=0;

TX_POWER=0xc0;//Tx_power_for_tag;

rfSettings.channr=MY_CHN;

rfSettings.mdmcfg4=0xc5;
```

```
                                                //
TX_POWER=Tx_power_for_tag;

ResetRadioCore();

InitRadio();

ReceiveOn();

transmitting=0;

receiving=1;

P3OUT&=~BIT6;

P3OUT&=~BIT4;

radio_state=RF_SRX;


multi_timer_over=0;

timer_over=0;


TA1CCTL0 =~ CCIE;

timer_over=0;

timer_cycle=0;

Enable_Start_Timer(delay_time);///// burayı yükseltebilirim
                                                        break;
                        }
                        case 0x20: ///TX_POWER
                        {
                                Message_Waiting=1;
                                TX_POWER=RxBuffer[6];
                                Reinit_radio=1;
                                first_talk=0;
                                second_talk=0;
                                can_sleep=0;
                                break;
                        }
                        case 0x30:// ID_CHANGE
                        {
                                Message_Waiting=1;

for(int_rec_i=0;int_rec_i<17;int_rec_i++)
                                {

MY_ID[int_rec_i]=RxBuffer[6+int_rec_i];
                                }
                                id_change=1;
                                first_talk=0;
                                second_talk=0;
                                can_sleep=0;
                                break;
                        }
                        default:
                        {
                                Message_Waiting=0;
                                ReceiveOn();
                                receiving=1;
                                transmitting=0;
                                P3OUT&=~BIT6;
                                P3OUT&=~BIT4;
                                break;
                        }

                  }
                }
                else
                {
                        Message_Waiting=0;
                        ReceiveOn();
                        receiving=1;
                        transmitting=0;
                        P3OUT&=~BIT6;
                        P3OUT&=~BIT4;
                }
              }
              else
              {
                        Message_Waiting=0;
                        ReceiveOn();
                        receiving=1;
                        transmitting=0;
                        P3OUT&=~BIT6;
                        P3OUT&=~BIT4;
              }


        }


        void main (void)
        {

                        WDTCTL = WDTPW+WDTHOLD;
                        WDTCTL = WDTPW + WDTHOLD;
// Stop WDT

                        read_MY_SEG();

                        if(!(MY_ID[1]==0xff && MY_ID[2]==0xff &&
MY_ID[3]==0xff && MY_ID[4]==0xff))
                        {
                                if(MY_ID[2]!=(0xff-MY_ID[1]))
                                {
                                        if(MY_ID[4]==(0xff-
MY_ID[3]))
                                        {

                read_MY_SEG();

                MY_ID[1]=MY_ID[3];

                MY_ID[2]=MY_ID[4];

                write_MY_SEG();
                                                {WDTCTL
= WDTPW+WDTCNTCL+WDTIS2;}
                                                while(1);
                                        }
                                        else    ////    ne    yapam
bilemedim
                                        {
                                                while(1);
                                        }
                                }
                                else
                                {
                                        if(MY_ID[4]!=(0xff-
MY_ID[3]))
                                        {

                read_MY_SEG();

                MY_ID[3]=MY_ID[1];

                MY_ID[4]=MY_ID[2];

                write_MY_SEG();
                                                {WDTCTL
= WDTPW+WDTCNTCL+WDTIS2;}
                                                while(1);
                                        }
                                }
                                my_fn=MY_ID[1];
                                ACCMSG[1]=my_fn;

                if(((MY_ID[1]&0xf0)>>4)>6)MY_CHN=(((MY_ID[1]&0xf
0)>>4)-6);
                                else
MY_CHN=(MY_ID[1]&0xf0)>>4;
                        }
                        if(MY_ID[5]==0xff && MY_ID[6]==0xff &&
MY_ID[7]==0xff && MY_ID[8]==0xff)
                        {
                                delay_time=3500;
                        }
                        else
                        {
                                if(MY_ID[6]!=(0xff-MY_ID[5]))
                                {
```

```
                                        if(MY_ID[8]==(0xff-
MY_ID[7]))
                                        {

        read_MY_SEG();

        MY_ID[5]=MY_ID[7];

        MY_ID[6]=MY_ID[8];

        write_MY_SEG();
                                                {WDTCTL
= WDTPW+WDTCNTCL+WDTIS2;}
                                                while(1);
                                        }
                                        else    ////    ne    yapam
bilemedim
                                        {

        delay_time=3500;
                                                //while(1);
                                        }
                                }
                                else
                                {
                                        if(MY_ID[8]!=(0xff-
MY_ID[7]))
                                        {

        read_MY_SEG();

        MY_ID[7]=MY_ID[5];

        MY_ID[8]=MY_ID[6];

        write_MY_SEG();
                                                {WDTCTL
= WDTPW+WDTCNTCL+WDTIS2;}
                                                while(1);
                                        }
                                }
                                delay_time=270*MY_ID[5];
                        }

                        TX_POWER=0xc0;//Tx_power_for_tag;
                        rfSettings.channr=MY_CHN;
                        rfSettings.mdmcfg4=0xc5;
                        SetVCore(2);
                        ResetRadioCore();
                        InitRadio();
                        InitButtonLeds();
                        ReceiveOn();
                        receiving=1;
                        transmitting=0;
                        radio_state=RF_SRX;
                        buttonPressed=0;
                        PMAPPWD = 0x02D52;                      // Get
write-access to port mapping regs
                        P1MAP5 = PM_UCA0RXD;                    //
Map UCA0RXD output to P2.6
                        P1MAP6 = PM_UCA0TXD;                    // Map
UCA0TXD output to P2.7
                        PMAPPWD = 0;                            // Lock port
mapping registers
                        P1DIR |= BIT6;                          // Set P2.7 as
TX output
                        P1SEL |= BIT5 + BIT6;                   // Select
P2.6 & P2.7 to UART function
                        P1REN &=~ BIT6;
                UCA0CTL0 |=UCPEN;
                UCA0CTL1 |= UCSWRST;                    // **Put state
machine in reset**
                UCA0CTL1 |= UCSSEL_2;                   // SMCLK
                UCA0BR0 = 6;    /// burayı 7 yapmak takdiri ilahi idi //
1MHz 9600 (see User's Guide)
                UCA0BR1 = 0;                            // 1MHz 9600
                UCA0MCTL = UCBRS_0 + UCBRF_13 + UCOS16;    //
Modln UCBRSx=0, UCBRFx=0,
                                        // over sampling
                UCA0CTL1 &= ~UCSWRST;                   // **Initialize
USCI state machine**
                UCA0IE |= UCRXIE;                       // Enable USCI_A0
RX interrupt

                for(main_i=0;main_i<250;main_i++)
                {
                        RxBuffer[main_i]=0;
                }

                Enable_Start_Timer(2000);

                while (1)
                {
                        __bis_SR_register(  GIE );///uykuyu ekle

                                if(  receiving>1  ||  transmitting>1  ||
MY_CHN > 10 )// || receive_poll>20)//||(MY_CHN)!=1 || my_fn!=0x11 )
                                {
                                        //receive_poll=0;
                                        {WDTCTL                 =
WDTPW+WDTCNTCL+WDTIS2;}
                                        while(1);
                                }
                        ////////////////////////////////////////////////////////////////////////////////
////////////////////
                                if(message_will_be_test)
                                {

message_will_be_test=0;
                                        test_RXMSG();
                                }
                                ////////////////// Herşey
Başlıyor//////////////////////////
                                if(!transmitting && Send_KEY)
                                {
                                        ReceiveOff();
                                        receiving=0;

radio_state=RF_SFRX;
                                        P3OUT|=BIT6;
                                        P3OUT|=BIT4;

Transmit(KEYMSG,25);

                                        transmitting=1;
                //                      Send_KEY=0;
                                }

//////////////////////////////////////////////////////////////
                                if(id_change)
                                {
                                        write_MY_SEG();
                                        copy_C2D();
                                        // read_MY_SEG();
                                        {WDTCTL                 =
WDTPW+WDTCNTCL+WDTIS2;}
                                        while(1);

                                }

//////////////////////////////////////////////////////////////
                                if(timer_over                   &&
(!message_will_be_test))
                                {
                                        P1OUT&=~BIT0;
                                        timer_over=0;

                                        TA1CCTL0 =~ CCIE;
                                        ////    hüseyin    abinin
algoritmayı da kullanabilirim//////////////////


for(main_k=0;main_k<10;main_k++)                                {

        for(main_i=0;main_i<17;main_i++)

        {

                nozzle_ids[main_i][main_k]=0;

        }
                                }
```

```
//////////////////////////////////////////////////////////////////////
                                                {
        ReceiveOff();
        radio_state=RF_SFRX;
        receiving=0;
        TX_POWER=0xc0;//Tx_power_for_tag;
        rfSettings.channr=MY_CHN;
        rfSettings.mdmcfg4=0xc5;
       // TX_POWER=Tx_power_for_tag;
        ResetRadioCore();
        InitRadio();
        ReceiveOn();
        transmitting=0;
        receiving=1;
        P3OUT&=~BIT6;
        P3OUT&=~BIT4;
        radio_state=RF_SRX;
                                                }
                                }
                                if(!Send_KEY && !transmitting)
                                {
if(radio_state!=RF_SRX)
                                                {

ReceiveOn();
transmitting=0;
receiving=1;
P3OUT&=~BIT6;
P3OUT&=~BIT4;
radio_state=RF_SRX;
                                                }
                                }
                                if(receive_poll>20              &&
!transmitting)
                                {
                                        receive_poll=0;
                                        ReceiveOff();
radio_state=RF_SFRX;
                                        receiving=0;
TX_POWER=0xc0;//Tx_power_for_tag;
rfSettings.channr=MY_CHN;
rfSettings.mdmcfg4=0xc5;
                                        ResetRadioCore();
                                        InitRadio();
                                        ReceiveOn();
                                        transmitting=0;
                                        receiving=1;
                                        P3OUT&=~BIT6;
                                        P3OUT&=~BIT4;
                                        radio_state=RF_SRX;
                                }
                                if(uart_data_will_send )
                                {
uart_data_will_send=0;
                                        Uart_Send();

                                }
        }

        }
        //////////////////////////////////////////////////////////////////////
        //                      INTERRUPTS                              //
        //////////////////////////////////////////////////////////////////////
        #pragma vector=CC1101_VECTOR
        __interrupt void CC1101_ISR(void)

        {
          long temp_crc_value=0,temp_crc_value1=0;

          switch(__even_in_range(RF1AIV,32))        // Prioritizing
Radio Core Interrupt
           {
            case  0: break;                         // No RF core interrupt
pending
            case  2: break;                 // RFIFG0
            case  4: break;                 // RFIFG1
            case  6: break;                 // RFIFG2
            case  8: break;                 // RFIFG3
            case 10: break;                 // RFIFG4
            case 12: break;                 // RFIFG5
            case 14: break;                 // RFIFG6
            case 16: break;                 // RFIFG7
            case 18: break;                 // RFIFG8
            case 20:                        // RFIFG9
             if(receiving )                                     //      RX
end of packet
              {

               // Read the length byte from the FIFO
               RxBufferLength = ReadSingleReg( RXBYTES );
                       ReadBurstReg(RF_RXFIFORD,
RxBuffer,RxBufferLength);// RxBufferLength);

                       if(RxBufferLength!=0x1C)
                       {

                               __no_operation();
                               ResetRadioCore();
                               InitRadio();
                               ReceiveOn();
                               receiving=1;
                               transmitting=0;
                               radio_state=RF_SRX;
                               return;
                               //while(1);
                       }
                       RxBufferLength=0x1c;
                       //DELTA = 0x9e3779b9;
                       P1OUT ^= BIT0;
                       if(RxBuffer[5]=='S'  &&  RxBuffer[3]==  0x40
&&                                              RxBuffer[0]==25
&&(((RxBuffer[1]&0xf0)>>4)==((my_fn&0xf0)>>4)))
//(((RxBuffer[1]&0xf0)>>4)==MY_CHN))//
((RxBuffer[1]&0xf0)>>4)==MY_CHN )
                       {
                               ////////////GET_KEY/////////////////
                               Generate_Key();
                               make_key_for_buf(KEYMSG);
                               Generate_Delta();
                               receive_poll=0;
                               Send_KEY=1;
                               return;
                       }
                       message_will_be_test=1;


              }
             else if(transmitting && !receiving)
            // TX end of packet
             {
```

```
                RF1AIE &= ~BIT9;                // Disable TX end-of-
packet interrupt
                P3OUT &= ~BIT6;                 // Turn off LED after
Transmit
                P3OUT &=~BIT4;
                transmitting = 0;
                Send_KEY=0;
                //ReceiveOn();
                //receiving=1;
               }
               else
               {
                        // MY_CHN=0;
                        rfSettings.channr=MY_CHN;
                        ResetRadioCore();
                        InitRadio();
                        ReceiveOn();
                        receiving = 1;
                        transmitting=0;
                        P3OUT&=~BIT6;
                        P3OUT&=~BIT4;

               }// trap
               break;
             case 22: break;                    // RFIFG10
             case 24: break;                    // RFIFG11
             case 26: break;                    // RFIFG12
             case 28: break;                    // RFIFG13
             case 30: break;                    // RFIFG14
             case 32: break;                    // RFIFG15
            }
            can_sleep=0;
            __bic_SR_register_on_exit(LPM3_bits);
           }

           ////////////////////////////////////////////////////////////////////
           #pragma vector=PORT1_VECTOR
           __interrupt void PORT1_ISR(void)
           {
            switch(__even_in_range(P1IV, 16))
            {
             case  0: break;
             case  2: break;                    // P1.0 IFG
             case  4: break;                    // P1.1 IFG
             case  6: break;                    // P1.2 IFG
             case  8: break;                    // P1.3 IFG
             case 10: break;                    // P1.4 IFG
             case 12: break;                    // P1.5 IFG
             case 14: break;                    // P1.6 IFG
             case 16:                           // P1.7 IFG
              P1IE = 0;                         // Debounce by disabling
buttons
              //buttonPressed = 1;
              check_button=1;
              Enable_Start_Timer(20);
              //can_sleep=0;
              multi_timer_over=0;
              __bic_SR_register_on_exit(LPM3_bits); // Exit active

              break;
            }
           }

           ////////////////////////////////////////////////////////////////////
           #pragma vector=TIMER1_A0_VECTOR
           __interrupt void TIMER1_A0_ISR(void)
           {

            if(timer_cycle>600)
            {
                        timer_cycle=0;
                        timer_over=1;

                        multi_timer_over++;
                        can_sleep=0;
                         __bic_SR_register_on_exit(LPM3_bits);       //
Exit active
            }
            else
            {
             TA1CCR0 += Timer_Count;
             timer_cycle++;
             can_sleep=1;
            }


           }

           ////////////////////////////////////////////////////////////////////////////
           #pragma vector=USCI_A0_VECTOR
           __interrupt void USCI_A0_ISR(void)
           {

                        switch(__even_in_range(UCA0IV,4))
            {
            case 0:break;                       // Vector 0 - no interrupt
            case 2:                             // Vector 2 - RXIFG
                        //if(WDTCTL&0x0080)       {WDTCTL       =
WDTPW+WDTCNTCL+WDTIS2;}
                        //while
(!(UCA0IFG&UCTXIFG));//{if(WDTCTL&0x0080)     {WDTCTL     =
WDTPW+WDTCNTCL+WDTIS2;}}                // USCI_A0 TX buffer
ready?
                        //WDTCTL = WDTPW+WDTHOLD;

                        if(uart_rx_count==0)
                        {
                                    if(UCA0RXBUF!=0x02)
                                    {
                                                uart_rx_count=0;
                                                UCA0IE |= UCRXIE;
                                                return;
                                    }
                                    else
                                    {
Uart_RX[uart_rx_count]=UCA0RXBUF;
                                                uart_rx_count++;

                                    }
                        }

                        else
                        {
Uart_RX[uart_rx_count]=UCA0RXBUF;
                                    uart_rx_count++;
                                    if(uart_rx_count>17)
                                    {
                                                Uart_RX[0]=0;
                                                Uart_RX[1]=0;
                                                uart_rx_count=0;
                                                UCA0IE |= UCRXIE;
                                                return;
                                    }
                                    if(uart_rx_count==2)
                                    {
                                                if(Uart_RX[1]!=0x52)
                                                {

if(Uart_RX[1]==0x02)
                                                            {

uart_rx_count=0;

Uart_RX[0]=0x02;

uart_rx_count++;

UCA0IE |= UCRXIE;

return;

                                                            }
                                                            else
                                                            {

Uart_RX[0]=0;

Uart_RX[1]=0;

uart_rx_count=0;

UCA0IE |= UCRXIE;

return;
                                                            }
                                                }
```

```
                }
                else if(uart_rx_count==5)
                {
                        if(Uart_RX[4]<0x31)
                        {
        uart_rx_count=0;                                        UCA0IE |=
UCRXIE;
        Uart_RX[0]=0;
        Uart_RX[1]=0;
        Uart_RX[2]=0;
        Uart_RX[3]=0;
        Uart_RX[4]=0;
                                                        return;
                        }
                }
                else if(uart_rx_count>5)
                {



        if(Uart_RX[uart_rx_count-1]==0xfa)
                                        {
        if(Uart_RX[uart_rx_count-2]==0x03)
                                                {


        ////////////////////////message handler//////////////////////////

        temp_crc_value=calcrc(Uart_RX,uart_rx_count-
4,0);///////CRC CHECK

        temp_crc_value1=  temp_crc_value;

        temp_crc_value&=0x000000ff;

        temp_crc_value1>>=8;

        temp_crc_value1&=0x000000ff;

        if(Uart_RX[uart_rx_count-4]==temp_crc_value              &&
Uart_RX[uart_rx_count-3]==temp_crc_value1)

        {


        ////////////////////////////////////////////////////////////////////////
```

```
                temp_delta    =    Uart_RX[5]*0x01000000    +
Uart_RX[6]*0x00010000 + Uart_RX[7]*0x00000100 + Uart_RX[8];

                nozzle_number=Uart_RX[4]-0x30;

                ////////////////artık                    sıra
bende////////////////////////////////////

                if(nozzle_number==0x0B                  &&
Uart_RX[5]==0xDC && Uart_RX[6]==0xBD && Uart_RX[7]==0xae)

                {

                        P3OUT ^= BIT4;

                        Message_Waiting=1;

                        MY_ID[1]=Uart_RX[8];

                        MY_ID[2]=0xff-Uart_RX[8];

                        MY_ID[3]=Uart_RX[8];

                        MY_ID[4]=0xff-Uart_RX[8];

                        MY_ID[5]=Uart_RX[10];

                        MY_ID[6]=0xff-Uart_RX[10];

                        MY_ID[7]=Uart_RX[10];

                        MY_ID[8]=0xff-Uart_RX[10];


                        id_change=1;

                        uart_rx_count=0;

                        UCA0IE |= UCRXIE;

                        return;

                }

                else

                {


temp_add_buf[0]=(((my_fn>>4)&0x0f)/10)+0x30;


temp_add_buf[1]=(((my_fn>>4)&0x0f)%10)+0x30;
```

```
        if((!(Uart_RX[2]==temp_add_buf[0]                       &&
temp_add_buf[1]==Uart_RX[3])))


        //if(((Uart_RX[2]&0x0f)*10                              +
Uart_RX[3]&0x0f)!=((my_fn>>4)&0x0f))

                        {

                                Uart_RX[0]=0;

                                Uart_RX[1]=0;

                                Uart_RX[2]=0;

                                Uart_RX[3]=0;

                                Uart_RX[4]=0;

                                uart_rx_count=0;

                                UCA0IE |= UCRXIE;

                                return;

                        }

                }

                receive_poll++;

        //      P1DIR |= BIT6;                  // Set P2.7 as
TX output

        //      P1SEL |= BIT5 + BIT6;           // Select
P2.6 & P2.7 to UART function

                P3OUT ^= BIT4;

                Uart_TX[2]=Uart_RX[2];

                Uart_TX[3]=Uart_RX[3];

                Uart_TX[4]=Uart_RX[4];

                //Uart_TX[5]=Uart_RX[5];

                uart_rx_count=0;

                Uart_RX[0]=0;

                Uart_RX[1]=0;
```

```
                Uart_RX[2]=0;
                Uart_RX[3]=0;
                Uart_RX[4]=0;
                //Uart_RX[5]=0;

                DELTA_USC=temp_delta;
                for(uart_int_i=0;uart_int_i<17;uart_int_i++)
                {

        Temp_BUF[uart_int_i]=nozzle_ids[uart_int_i][nozzle_numbe
r];//uart_int_i+1;//

                }
                //if(Temp_BUF[6]==0xff                    ||
Temp_BUF[6]==0x00)
                //No_id=1;
                //else No_id=0;

                uart_data_will_send=1;

        }//////////////////////////////////////////////


                                                }
                                                else
if(Uart_RX[uart_rx_count-2]==0x10)
                                                {

        Uart_RX[uart_rx_count-2]=0xfa;
        uart_rx_count--;

                                                }
                                                else
                                                {

        uart_rx_count=0;
        UCA0IE |= UCRXIE;
                Uart_RX[0]=0;
                Uart_RX[1]=0;
                Uart_RX[2]=0;
                Uart_RX[3]=0;
                Uart_RX[4]=0;
                return;
                                                        }
                                                }
                                        }
                                }

        //if(UCA0RXBUF==0x12)P1OUT^=BIT0;
          //UCA0TXBUF = UCA0RXBUF;                // TX ->
RXed character
          break;
          case 4:
                tx_int_count++;
                WDTCTL                                  =
WDTPW+WDTCNTCL+WDTIS2;
                if(tx_int_count>tx_count)
                {
                        tx_int_count=0;
                        tx_count=0;
                        WDTCTL                          =
WDTPW+WDTHOLD;
                        /*uart_rx_count=0;
                        UCA0IE |= UCRXIE;
                        Uart_RX[0]=0;
                        Uart_RX[1]=0;
                        Uart_RX[2]=0;
                        Uart_RX[3]=0;
                        Uart_RX[4]=0;

                return;
                        */
                        //UCA0IE &=~ UCTXIE;
        //              UCA0IFG &=~UCTXIFG;
                }
                else
                {
                        UCA0TXBUF                       =
Uart_TX[tx_int_count];
                  UCA0IE |= UCTXIE;

                }
                break;                  // Vector 4 - TXIFG
          default: break;
          }
        }
```

EK-4 Mesafe Verici

EK-4.1. main.c

```
        /*S
         * main.c
         */

        #include          "D:\ccs          projects\MESAFYENI
ACC\UHFVISNOZZLE_051211.h"
        #include "D:\ccs projects\cc430f6137.h"
        #include "D:\ccs projects\include\string.h"
        #include <stdio.h>
        #include <stdlib.h>
        #define Tx_power_for_acc        0x51
        #define Tx_power_for_tag        0x02

        void main (void)
        {
                WDTCTL = WDTPW+WDTHOLD;
                WDTCTL    =    WDTPW    +    WDTHOLD;
// Stop WDT




                read_MY_SEG();
         /*
         *
         *
         *
         *
         *
         *
         *
         */


        // my_fn=0;
         //ACCMSG[1]=my_fn;

//if(((MY_ID[1]&0xf0)>>4)>6)MY_CHN=(((MY_ID[1]&0xf0)>>4)-6);
        //else MY_CHN=(MY_ID[1]&0xf0)>>4;
        //my_fn=0x11;
        TX_POWER=0x51;//Tx_power_for_tag;
        MY_CHN=0;/////////
        rfSettings.channr=0;
        SetVCore(2);
        ResetRadioCore();
        InitRadio();
        InitButtonLeds();
        ReceiveOn();
        receiving=1;
        transmitting=0;
```

```
            radio_state=RF_SRX;
            buttonPressed=0;
            //first_talk=1;
            //read_MY_ID();
            PMAPPWD = 0x02D52;                  // Get write-access
to port mapping regs

            P1MAP5 = PM_UCA0RXD;                // Map
UCA0RXD output to P2.6

            P1MAP6 = PM_UCA0TXD;                // Map
UCA0TXD output to P2.7

            PMAPPWD = 0;                        // Lock port mapping
registers



            P1DIR |= BIT6;                      // Set P2.7 as TX output

            P1SEL |= BIT5 + BIT6;               // Select P2.6 & P2.7
to UART function
            P1REN &=~ BIT6;


          //    UCA0CTL0|=UCMODE0+UCMODE1;
                UCA0CTL1 |= UCSWRST;            //
**Put state machine in reset**

            UCA0CTL1 |= UCSSEL_1;               // CLK = ACLK

            UCA0BR0 = 0x03;                     // 32kHz/9600=3.41
(see User's Guide)

            UCA0BR1 = 0x00;                     //

            UCA0CTL0 |=UCPEN;
            UCA0CTL0 &= ~UCSPB;
            UCA0MCTL = UCBRS_3+UCBRF_0;         //
Modulation UCBRSx=3, UCBRFx=0

            UCA0CTL1 &= ~UCSWRST;               // **Initialize
USCI state machine**

            UCA0IE |= UCRXIE;                   // Enable USCI_A0
RX interrupt


           /*
            UCA0CTL0 |=UCPEN;
            UCA0CTL1 |= UCSWRST;                // **Put state
machine in reset**
            UCA0CTL1 |= UCSSEL_2;               // SMCLK
            UCA0BR0 = 6;    /// burayı 7 yapmak takdiri ilahi idi //
1MHz 9600 (see User's Guide)
            UCA0BR1 = 0;                        // 1MHz 9600
            UCA0MCTL = UCBRS_0 + UCBRF_13 + UCOS16;    //
Modln UCBRSx=0, UCBRFx=0,
                                                // over sampling
            UCA0CTL1 &= ~UCSWRST;               // **Initialize
USCI state machine**
            UCA0IE |= UCRXIE;
           */
           for(main_i=0;main_i<250;main_i++)
           {
                RxBuffer[main_i]=0;
           }
           for(main_i=0;main_i<27;main_i++)
            {
                Uart_RX[main_i]=0;
            }

          //  buttonPressed=1;

           //WDTCTL = WDTPW ;


           while (1)
           {
                __bis_SR_register(  GIE );///uykuyu ekle

                //while (buttonPressed)

                        //temp_random=rand();
                        if((MY_CHN&0xf0)!=0             ||
receiving>1 || transmitting>1 )
                        {
                                {WDTCTL             =
WDTPW+WDTCNTCL+WDTIS2;}
                                while(1);
                        }
                        ////////////////                Herşey
Başlıyor//////////////////////////
                        if(!transmitting && Send_id)
                        {

                                ReceiveOff();
                                receiving=0;

radio_state=RF_SFRX;
                                P3OUT|=BIT6;
                                P3OUT|=BIT4;

Transmit(TAGIDMSG,25);

                                transmitting=1;
                                Send_id=0;
                        }
                        if(!transmitting && Send_Plate)
                        {

                                ReceiveOff();
                                receiving=0;

radio_state=RF_SFRX;
                                P3OUT|=BIT6;
                                P3OUT|=BIT4;

Transmit(PLATEMSG,25);

                                transmitting=1;
                                Send_Plate=0;
                        }
                        if(!transmitting                &&
Send_Nozzle_id)
                        {

                                ReceiveOff();
                                receiving=0;

radio_state=RF_SFRX;
                                P3OUT|=BIT6;

Transmit(NOZIDMSG,25);

                                transmitting=1;
                                Send_Nozzle_id=0;
                        }

                        if(!transmitting                &&
Send_TAG_POWER)
                        {

                                ReceiveOff();
                                receiving=0;

radio_state=RF_SFRX;
                                P3OUT|=BIT6;

Transmit(TAGTXMSG,25);

                                transmitting=1;

Send_TAG_POWER=0;
                        }
                        if(!transmitting && Get_Power)
                        {
                                Get_Power=0;
                                ReceiveOff();
                                receiving=0;

radio_state=RF_SFRX;
                                P3OUT|=BIT6;

Transmit(POWERMSG,25);

                                transmitting=1;

//Send_TAG_POWER=0;
                        }
                        if(!transmitting                &&
Send_RX_PowerMSG)
                        {
```

```
Send_RX_PowerMSG=0;                                     ReceiveOff();
                                                        receiving=0;

radio_state=RF_SFRX;
                                                        P3OUT|=BIT6;

Transmit(RXPOWERMSG,25);
                                                        transmitting=1;

//Send_TAG_POWER=0;
                                                }
                                                if(!transmitting && Send_KEY)
                                                {

                                                        ReceiveOff();
                                                        receiving=0;

radio_state=RF_SFRX;
                                                        P3OUT|=BIT6;
                                                        P3OUT|=BIT4;

Transmit(KEYMSG,25);
                                                        transmitting=1;
                                                        Send_KEY=0;
                                                }
                                                if(!transmitting)
                                                {
                                                        //
if(radio_state!=RF_SRX)
                                                        {

ReceiveOn();

transmitting=0;

receiving=1;

radio_state=RF_SRX;
                                                        }


                                                }

        }



}
////////////////////////////////////////////////////////////////////////////
//                      INTERRUPTS                                        //
////////////////////////////////////////////////////////////////////////////
#pragma vector=CC1101_VECTOR
__interrupt void CC1101_ISR(void)
{
//   long temp_crc_value=0,temp_crc_value1=0;

  switch(__even_in_range(RF1AIV,32))        // Prioritizing
Radio Core Interrupt
  {
    case  0: break;                         // No RF core interrupt
pending
    case  2: break;                         // RFIFG0
    case  4: break;                         // RFIFG1
    case  6: break;                         // RFIFG2
    case  8: break;                         // RFIFG3
    case 10: break;                         // RFIFG4
    case 12: break;                         // RFIFG5
    case 14: break;                         // RFIFG6
    case 16: break;                         // RFIFG7
    case 18: break;                         // RFIFG8
    case 20:                                // RFIFG9
      if(receiving )                                    //      RX
end of packet
      {


        // Read the length byte from the FIFO
        RxBufferLength = ReadSingleReg( RXBYTES );
                ReadBurstReg(RF_RXFIFORD,
RxBuffer,RxBufferLength);// RxBufferLength);
                if(RxBufferLength!=0x1C)
                {

                        __no_operation();
                        ResetRadioCore();
                        InitRadio();
                        ReceiveOn();
                        receiving=1;
                        transmitting=0;
                        radio_state=RF_SRX;
                        return;
                        //while(1);
                }
                RxBufferLength=0x1c;
                //DELTA = 0x9e3779b9;

                if(RxBufferLength==0x1C && RxBuffer[3]==0x40)
                {

                        ////////////GET_KEY/////////////////
                        Generate_Key();
                        make_key_for_buf(KEYMSG);
                        Generate_Delta();
                ///     receive_poll=0;
                        Send_KEY=1;
                        return;
                }
                else if(RxBuffer[5]=='S')
                {
                        }
                        else
                        {

                make_for_Crypto_buf(RxBuffer,RxBufferLength-4);
                        Decode(Crypto_BUF,25/4);//(RxBufferLength-3)/4);
                        make_for_buf(RxBuffer,(RxBufferLength-4));
                        }


                        if( RxBuffer[5]=='S')
                {
                  temp_crc_value=calcrc(RxBuffer,(RxBufferLength-3)-
3,0);
                  temp_crc_value1=  temp_crc_value;
                  temp_crc_value&=0x000000ff;
                  temp_crc_value1>>=8;
                  temp_crc_value1&=0x000000ff;
                  if((RxBuffer[(RxBufferLength-3)-
1]==temp_crc_value)     &&     (RxBuffer[(RxBufferLength-3)-2]     ==
temp_crc_value1) )
                  {
                    __no_operation();
                    P1OUT ^= BIT0;                  // Toggle LED1
                    ///// Message correct


                    switch(RxBuffer[3])
                    {
                    case 0x11:
                    {

                        Uart_TX[0]=0x02;
                        Uart_TX[1]=0x52;
                        Uart_TX[2]=0x30;
                        Uart_TX[3]=0x31;
                        Uart_TX[4]=0x31;
                        Uart_TX[5]=0x34;
                        //Uart_TX[6]=RxBuffer[8];
                        for(int_rec_i=0;int_rec_i<17;int_rec_i++)
                        {
                                        //
TAGIDMSG[int_rec_i+6]=RxBuffer[6+int_rec_i];

Uart_TX[int_rec_i+6]=RxBuffer[6+int_rec_i];
                        }

                        temp_crc_value=calcrc(Uart_TX,22,0);
                        temp_crc_value1=  temp_crc_value;
                        temp_crc_value1>>=8;
                        temp_crc_value1 &= 0x00ff;
                        temp_crc_value &= 0x00ff;
```

```
                    if(((char)temp_crc_value&0x00ff)!=0xfa)
                    {
Uart_TX[22]=((char)temp_crc_value&0x00ff);
if(((char)temp_crc_value1&0x00ff)!=0xfa)
                        {
Uart_TX[23]=((char)temp_crc_value1&0x00ff);
                                Uart_TX[24]=0x03;
                                Uart_TX[25]=0xfa;
                                tx_count=25;
                                UCA0TXBUF       =
Uart_TX[0];
                                UCA0IE |= UCTXIE;
                        }
                        else
                        {
                                Uart_TX[23]=0x10;
Uart_TX[24]=((char)temp_crc_value1&0x00ff);
                                Uart_TX[25]=0x03;
                                Uart_TX[26]=0xfa;
                                UCA0TXBUF       =
Uart_TX[0];
                                UCA0IE |= UCTXIE;
                                tx_count=26;
                        }
                }
                else
                {
Uart_TX[22]=0x10;//((char)temp_crc_value&0x00ff);
Uart_TX[23]=((char)temp_crc_value&0x00ff);

if(((char)temp_crc_value1&0x00ff)!=0xfa)
                        {
Uart_TX[24]=((char)temp_crc_value1&0x00ff);
                                Uart_TX[25]=0x03;
                                Uart_TX[26]=0xfa;
                                tx_count=26;
                                UCA0TXBUF       =
Uart_TX[0];
                                UCA0IE |= UCTXIE;
                        }
                        else
                        {
                                Uart_TX[24]=0x10;
Uart_TX[25]=((char)temp_crc_value1&0x00ff);
                                Uart_TX[26]=0x03;
                                Uart_TX[27]=0xfa;
                                tx_count=27;
                                UCA0TXBUF       =
Uart_TX[0];
                                UCA0IE |= UCTXIE;
                        }
                }
                break;
        }
        case 0x20: ///TX_POWER
        {
                Uart_TX[0]=0x02;
                Uart_TX[1]=0x52;
                Uart_TX[2]=0x30;
                Uart_TX[3]=0x31;
                Uart_TX[4]=0x31;
                Uart_TX[5]=0x33;
                Uart_TX[6]=RxBuffer[8];
                Uart_TX[7]=RxBuffer[6];
                Uart_TX[8]=RxBuffer[7];
                Uart_TX[9]=RxBuffer[26];
                for(int_rec_i=0;int_rec_i<14;int_rec_i++)
                {
                        //
TAGIDMSG[int_rec_i+6]=RxBuffer[6+int_rec_i];
Uart_TX[int_rec_i+10]=0xff;//RxBuffer[6+int_rec_i];
                }

        temp_crc_value=calcrc(Uart_TX,22,0);
        temp_crc_value1=  temp_crc_value;
        temp_crc_value1>>=8;
        temp_crc_value1 &= 0x00ff;
        temp_crc_value &= 0x00ff;

        if(((char)temp_crc_value&0x00ff)!=0xfa)
        {
                Uart_TX[22]=((char)temp_crc_value&0x00ff);
                if(((char)temp_crc_value1&0x00ff)!=0xfa)
                {
Uart_TX[23]=((char)temp_crc_value1&0x00ff);
                        Uart_TX[24]=0x03;
                        Uart_TX[25]=0xfa;
                        tx_count=25;
                        UCA0TXBUF = Uart_TX[0];
                        UCA0IE |= UCTXIE;
                }
                else
                {
                        Uart_TX[23]=0x10;
Uart_TX[24]=((char)temp_crc_value1&0x00ff);
                        Uart_TX[25]=0x03;
                        Uart_TX[26]=0xfa;
                        UCA0TXBUF = Uart_TX[0];
                        UCA0IE |= UCTXIE;
                        tx_count=26;
                }
        }
        else
        {
Uart_TX[22]=0x10;//((char)temp_crc_value&0x00ff);
                Uart_TX[23]=((char)temp_crc_value&0x00ff);
                if(((char)temp_crc_value1&0x00ff)!=0xfa)
                {
Uart_TX[24]=((char)temp_crc_value1&0x00ff);
                        Uart_TX[25]=0x03;
                        Uart_TX[26]=0xfa;
```

```
                    tx_count=26;

                    UCA0TXBUF = Uart_TX[0];
                    UCA0IE |= UCTXIE;
                }
                else
                {
                    Uart_TX[24]=0x10;

Uart_TX[25]=((char)temp_crc_value1&0x00ff);
                    Uart_TX[26]=0x03;
                    Uart_TX[27]=0xfa;
                    tx_count=27;
                    UCA0TXBUF = Uart_TX[0];
                    UCA0IE |= UCTXIE;
                }
            }
                break;
            }
            case 0x31:// ID_CHANGE
            {
                Message_Waiting=1;
               /*   for(int_rec_i=0;int_rec_i<22;int_rec_i++)
                  {
Uart_TX[int_rec_i]=Uart_RX[int_rec_i];
                  }
            */
                  for(int_rec_i=0;int_rec_i<17;int_rec_i++)
                  {
TAGIDMSG[int_rec_i+6]=RxBuffer[6+int_rec_i];
Uart_TX[int_rec_i+6]=RxBuffer[6+int_rec_i];
                  }

            temp_crc_value=calcrc(Uart_TX,22,0);
temp_crc_value1=  temp_crc_value;
temp_crc_value1>>=8;
temp_crc_value1 &= 0x00ff;
temp_crc_value &= 0x00ff;

if(((char)temp_crc_value&0x00ff)!=0xfa)
{
        Uart_TX[22]=((char)temp_crc_value&0x00ff);
        if(((char)temp_crc_value1&0x00ff)!=0xfa)
        {
Uart_TX[23]=((char)temp_crc_value1&0x00ff);
            Uart_TX[24]=0x03;
            Uart_TX[25]=0xfa;
            tx_count=25;
            UCA0TXBUF = Uart_TX[0];
            UCA0IE |= UCTXIE;
        }
        else
        {
            Uart_TX[23]=0x10;
Uart_TX[24]=((char)temp_crc_value1&0x00ff);
            Uart_TX[25]=0x03;
            Uart_TX[26]=0xfa;
            UCA0TXBUF = Uart_TX[0];
            UCA0IE |= UCTXIE;

            tx_count=26;
        }
}
else
{
Uart_TX[22]=0x10;//((char)temp_crc_value&0x00ff);
        Uart_TX[23]=((char)temp_crc_value&0x00ff);

        if(((char)temp_crc_value1&0x00ff)!=0xfa)
        {
Uart_TX[24]=((char)temp_crc_value1&0x00ff);
            Uart_TX[25]=0x03;
            Uart_TX[26]=0xfa;
            tx_count=26;

            UCA0TXBUF = Uart_TX[0];
            UCA0IE |= UCTXIE;
        }
        else
        {
            Uart_TX[24]=0x10;

Uart_TX[25]=((char)temp_crc_value1&0x00ff);
            Uart_TX[26]=0x03;
            Uart_TX[27]=0xfa;
            tx_count=27;
            UCA0TXBUF = Uart_TX[0];
            UCA0IE |= UCTXIE;
        }
}
              break;
```

```
                }
                default:
                {
                        Message_Waiting=0;
                        ReceiveOn();
                        receiving=1;
                        transmitting=0;
                        break;
                }

                }
              }
              else
              {
                        Message_Waiting=0;
                        ReceiveOn();
                        receiving=1;
                        transmitting=0;

              }
            }
            else
            {
                    Message_Waiting=0;
                    ReceiveOn();
                    receiving=1;
                    transmitting=0;
                    P3OUT=~BIT6;
            }
           }
           else if(transmitting && !receiving)
          // TX end of packet
           {
            RF1AIE &= ~BIT9;                // Disable TX end-of-
packet interrupt
            P3OUT &= ~BIT6;                 // Turn off LED after
Transmit
            transmitting = 0;
            //ReceiveOn();
            //receiving=1;
           }
           else
           {
                    MY_CHN=0;
                    rfSettings.channr=MY_CHN;
                    ResetRadioCore();
                    InitRadio();
                    ReceiveOn();
                    receiving = 1;
                    transmitting=0;

           }// trap
           break;
          case 22: break;                   // RFIFG10
          case 24: break;                   // RFIFG11
          case 26: break;                   // RFIFG12
           case 28: break;                  // RFIFG13
           case 30: break;                  // RFIFG14
           case 32: break;                  // RFIFG15
          }
          can_sleep=0;
          __bic_SR_register_on_exit(LPM3_bits);
         }

         ////////////////////////////////////////////////////////////////////////////
         #pragma vector=PORT1_VECTOR
         __interrupt void PORT1_ISR(void)
         {
          switch(__even_in_range(P1IV, 16))
          {
           case  0: break;
           case  2: break;                  // P1.0 IFG
           case  4: break;                  // P1.1 IFG
           case  6: break;                  // P1.2 IFG
           case  8: break;                  // P1.3 IFG
           case 10: break;                  // P1.4 IFG
           case 12: break;                  // P1.5 IFG
           case 14: break;                  // P1.6 IFG
           case 16:                        // P1.7 IFG
            P1IE = 0;                       // Debounce by disabling
buttons
            //buttonPressed = 1;
            check_button=1;
            Enable_Start_Timer(20);
            //can_sleep=0;
            multi_timer_over=0;
            __bic_SR_register_on_exit(LPM3_bits); // Exit active

            break;
          }
         }

         ////////////////////////////////////////////////////////////////////////////
         #pragma vector=TIMER1_A0_VECTOR
         __interrupt void TIMER1_A0_ISR(void)
         {

          if(timer_cycle>600)
          {
                    timer_cycle=0;
                    timer_over=1;

                    multi_timer_over++;
                    can_sleep=0;
                     __bic_SR_register_on_exit(LPM3_bits);       //
Exit active
          }
          else
          {
           TA1CCR0 += Timer_Count;
           timer_cycle++;
           can_sleep=1;
          }

         }


         ////////////////////////////////////////////////////////////////////////////
         #pragma vector=USCI_A0_VECTOR
         __interrupt void USCI_A0_ISR(void)
         {
                    char uart_int_i;
          switch(__even_in_range(UCA0IV,4))
          {
          case 0:break;                     // Vector 0 - no interrupt
          case 2:                           // Vector 2 - RXIFG
                    //if(WDTCTL&0x0080)       {WDTCTL       =
WDTPW+WDTCNTCL+WDTIS2;}
                    //while
(!(UCA0IFG&UCTXIFG));//{if(WDTCTL&0x0080)     {WDTCTL     =
WDTPW+WDTCNTCL+WDTIS2;}}                    // USCI_A0 TX buffer
ready?
                    //WDTCTL = WDTPW+WDTHOLD;
                    if(uart_rx_count==0)
                    {
                              if(UCA0RXBUF!=0x02)
                              {
                                        uart_rx_count=0;
                                        //UCA0IE |= UCRXIE;
                                        return;
                              }
                              else
                              {

Uart_RX[uart_rx_count]=UCA0RXBUF;
                                        uart_rx_count++;

                              }
                    }

                    else
                    {

Uart_RX[uart_rx_count]=UCA0RXBUF;
                              uart_rx_count++;
                              if(uart_rx_count>30)
                              {

for(uart_int_i=0;uart_int_i<27;uart_int_i++)
                                        {

Uart_RX[uart_int_i]=0;

                                        }
                                        uart_rx_count=0;

                                        //UCA0IE |= UCRXIE;
                                        return;
```

```
                }
                if(uart_rx_count==2)
                {
                        if(Uart_RX[1]!=0x52)
                        {
if(Uart_RX[1]==0x02)
                                {
uart_rx_count=0;
Uart_RX[0]=0x02;
uart_rx_count++;                                //
UCA0IE |= UCRXIE;
return;
                                }
                                else
                                {
Uart_RX[0]=0;
Uart_RX[1]=0;
uart_rx_count=0;
//UCA0IE |= UCRXIE;
return;
                                }
                        }
                }
                else if(uart_rx_count==5)
                {
                        if(Uart_RX[4]!=0x31)
                        {
uart_rx_count=0;
                                //UCA0IE
|= UCRXIE;
Uart_RX[0]=0;
Uart_RX[1]=0;
Uart_RX[2]=0;
Uart_RX[3]=0;
Uart_RX[4]=0;
                                return;
                        }
                }
                        else if(uart_rx_count>15)
                        {
if(Uart_RX[uart_rx_count-1]==0xfa)
                                {
if(Uart_RX[uart_rx_count-2]==0x03)
                                {
temp_crc_value_=0;
////////////////////////message handler//////////////////////////
temp_crc_value_=calcrc(Uart_RX,uart_rx_count-
4,0);///////CRC CHECK
temp_crc_value1_=  temp_crc_value_;
temp_crc_value_&=0x000000ff;
temp_crc_value1_>>=8;
temp_crc_value1_&=0x000000ff;
if(Uart_RX[uart_rx_count-4]==temp_crc_value_          &&
Uart_RX[uart_rx_count-3]==temp_crc_value1_)
{
P3OUT^=BIT4;
switch(Uart_RX[5])
{
case 0x30:
{
for(uart_int_i=0;uart_int_i<17;uart_int_i++)
{
TAGIDMSG[uart_int_i+6]=Uart_RX[uart_int_i+6];
}
for(int_rec_i=0;int_rec_i<22;int_rec_i++)
{
Uart_TX[int_rec_i]=Uart_RX[int_rec_i];
}
for(uart_int_i=0;uart_int_i<27;uart_int_i++)
{
Uart_RX[uart_int_i]=0;
}
uart_rx_count=0;
Send_id=1;
break;
}
case 0x31:
{
for(uart_int_i=0;uart_int_i<17;uart_int_i++)
{
NOZIDMSG[uart_int_i+6]=Uart_RX[uart_int_i+6];
}
for(int_rec_i=0;int_rec_i<22;int_rec_i++)
{
Uart_TX[int_rec_i]=Uart_RX[int_rec_i];
}
for(uart_int_i=0;uart_int_i<27;uart_int_i++)
{
```

```
Uart_RX[uart_int_i]=0;

        }
                uart_rx_count=0;
                uart_rx_count=0;
                Send_Nozzle_id=1;
                break;

            }
            case 0x32:
            {

for(uart_int_i=0;uart_int_i<17;uart_int_i++)
                {

ACCIDMSG[uart_int_i+6]=Uart_RX[uart_int_i+6];

                }
                uart_rx_count=0;
                Send_ACC_id=1;
                break;
            }
            case 0x33:
            {

for(uart_int_i=0;uart_int_i<17;uart_int_i++)
                {

TAGTXMSG[uart_int_i+6]=Uart_RX[uart_int_i+6];

                }

for(int_rec_i=0;int_rec_i<22;int_rec_i++)
                {

Uart_TX[int_rec_i]=Uart_RX[int_rec_i];
                }

for(uart_int_i=0;uart_int_i<27;uart_int_i++)
                {

Uart_RX[uart_int_i]=0;

                }
                uart_rx_count=0;

Send_TAG_POWER=1;
                break;

                break;
        }
        case 0x34:
        {
                ask_id=1;
                break;
        }
        case 0x35:
        {

for(uart_int_i=0;uart_int_i<17;uart_int_i++)
                {

PLATEMSG[uart_int_i+6]=Uart_RX[uart_int_i+6];

                }

for(int_rec_i=0;int_rec_i<22;int_rec_i++)
                {

Uart_TX[int_rec_i]=Uart_RX[int_rec_i];
                }

for(uart_int_i=0;uart_int_i<27;uart_int_i++)
                {

Uart_RX[uart_int_i]=0;

                }
                uart_rx_count=0;
                Send_Plate=1;
                break;
        }
        case 0x36:
        {
                uart_rx_count=0;
                Get_Power=1;
                break;
                break;
        }
        case 0x37:
        {
                uart_rx_count=0;
```

```
                                Get_ID_POWER=1;
                                break;
                                //break;
                        }
                        case 0x38:
                        {
                                uart_rx_count=0;
RXPOWERMSG[6]=Uart_RX[6];

Send_RX_PowerMSG=1;
                                break;
                        }
                        case 0xdd:
                        {
                                //MY_CHN=
rfSettings.channr=MY_CHN;
                                ResetRadioCore();
                                InitRadio();
                                ReceiveOn();
                                receiving = 1;
                                transmitting=0;

                                break;
                        }
                        default:
                        {

for(uart_int_i=0;uart_int_i<27;uart_int_i++)
                                {
Uart_RX[uart_int_i]=0;

                                }
                                uart_rx_count=0;
                                break;
                        }
                }
                else
                {
                        for(uart_int_i=0;uart_int_i<27;uart_int_i++)
                        {
                                Uart_RX[uart_int_i]=0;

                        }
                        uart_rx_count=0;
                }
        else
        {
                for(uart_int_i=0;uart_int_i<27;uart_int_i++)
                {
                        Uart_RX[uart_int_i]=0;

                }
                uart_rx_count=0;
        }
if(Uart_RX[uart_rx_count-2]==0x10)
        {
                Uart_RX[uart_rx_count-2]=0xfa;
                uart_rx_count--;
        }
                }
        }
        //if(UCA0RXBUF==0x12)P1OUT^=BIT0;
          //UCA0TXBUF = UCA0RXBUF;                // TX ->
RXed character
          break;
         case 4:
                tx_int_count++;
                if(tx_count==0)
                {
                        //UCA0IE |= UCRXIE;
                }
                if(tx_int_count>tx_count)
                {
                        tx_int_count=0;
                        tx_count=0;
                        //UCA0IE |= UCRXIE;
                        //UCA0IE &=~ UCTXIE;
        //              UCA0IFG &=~UCTXIFG;
                }
                else
                {
                        UCA0TXBUF               =
Uart_TX[tx_int_count];
                  UCA0IE |= UCTXIE;

                }
                break;                  // Vector 4 - TXIFG
         default: break;
         }
        }
```

EK-4 Mesafe Verici Bilgisayar Programı

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO.Ports;


namespace WindowsFormsApplication4
{
  public partial class Form1 : Form
  {
    bool reset_click = false;
    byte[]          reset_array     =       new
byte[50];//={0x02,0x52,0x30,0x31,0x31,0x38,0x11,0x22,0x33,0x44,0x55
,0x66,0x77,0x11,0x22,0x33,0x44,0x55,0x66,0x77,0x88,0xC7,0xDE,0x03
,0xFA}
```

```
        byte[] kb_reset_array = new byte[50];//={0x02,0x52,0x30,0x31,0x31,0x38,0x11,0x22,0x33,0x44,0x55,0x66,0x77,0x11,0x22,0x33,0x44,0x55,0x66,0x77,0x88,0xC7,0xDE,0x03,0xFA}
        byte[] kb_config_array = new byte[50];//={0x02,0x52,0x30,0x31,0x31,0x38,0x11,0x22,0x33,0x44,0x55,0x66,0x77,0x11,0x22,0x33,0x44,0x55,0x66,0x77,0x88,0xC7,0xDE,0x03,0xFA}
        byte[] kb_active_array = new byte[50];
        byte[] get_ref_array = new byte[50];

        byte[] okuma_temp =new byte[20];
        byte[] kesme_temp = new byte[20];
        byte[] okuma_temp1 = new byte[20];
        byte[] kesme_temp1 = new byte[20];
        int[] Msg = new int[50];
        int buffer_offset = 0;
        bool data_lenght_ok = false, get_ref=false;
        string id;
        string id1;
        int temp_i = 0, timer_count = 0, nom_value = 0, max_value = 0, value = 0;
        Int32 crcresss = 0;
        bool Config_Gonder = false, KB_RESET = false, KB_RESET1 = false, serial_reseted = false, KB_ACTIVE = false, KB_ACTIVE1=false,send_password=false;
        bool okuma_degeri = false,kesme_degeri=false,deger_gonder=false,tx_powersend=false;
        byte sent_value = 0;
        /// <summary>
        ///
        ///
        ///
        /// </summary>
        /// <param name="ptr"></param>
        /// <param name="length"></param>
        /// <param name="crc_ini"></param>
        /// <returns></returns>
        ///
        void Add_to_Textbox(int textbox_input)
        {
            switch (textbox_input)
            {
                case 0:
                    id = id + "0";
                    break;
                case 1:
                    id = id + "1";
                    break;
                case 2:
                    id = id + "2";
                    break;
                case 3:
                    id = id + "3";
                    break;
                case 4:
                    id = id + "4";
                    break;
                case 5:
                    id = id + "5";
                    break;
                case 6:
                    id = id + "6";
                    break;
                case 7:
                    id = id + "7";
                    break;
                case 8:
                    id = id + "8";
                    break;
                case 9:
                    id = id + "9";
                    break;
                case 10:
                    id = id + "A";
                    break;
                case 11:
                    id = id + "B";
                    break;
                case 12:
                    id = id + "C";
                    break;
                case 13:
                    id = id + "D";
                    break;
                case 14:
                    id = id + "E";
                    break;
                case 15:
                    id= id+ "F";
                    break;
            }
        }

        void Add_to_Textbox1(int textbox_input)
        {
            switch (textbox_input)
            {
                case 0:
                    id1 = id1 + "0";
                    break;
                case 1:
                    id1 = id1 + "1";
                    break;
                case 2:
                    id1 = id1 + "2";
                    break;
                case 3:
                    id1 = id1 + "3";
                    break;
                case 4:
                    id1 = id1 + "4";
                    break;
                case 5:
                    id1 = id1 + "5";
                    break;
                case 6:
                    id1 = id1 + "6";
                    break;
                case 7:
                    id1 = id1 + "7";
                    break;
                case 8:
                    id1 = id1 + "8";
                    break;
                case 9:
                    id1 = id1 + "9";
                    break;
                case 10:
                    id1 = id1 + "A";
                    break;
                case 11:
                    id1 = id1 + "B";
                    break;
                case 12:
                    id1 = id1 + "C";
                    break;
                case 13:
                    id1 = id1 + "D";
                    break;
                case 14:
                    id1 = id1 + "E";
                    break;
                case 15:
                    id1 = id1 + "F";
                    break;
            }
        }

        void calc_value()
        {
            if(nom_value>max_value)
            {
                if(Math.Abs(max_value - nom_value) > 0 && Math.Abs(max_value - nom_value)<3)
                {
                    value = nom_value + 5;
                }
                else if (Math.Abs(max_value - nom_value) < 5)
                {
                    value = nom_value + 3;
                }
                else if (Math.Abs(max_value - nom_value) < 7)
                {
                    value = nom_value + 1;
                }
                else if (Math.Abs(max_value - nom_value) < 9)
```

```
                {
                    value = nom_value;
                   // nom_value = nom_value;
                }
                else if (Math.Abs(max_value - nom_value) < 11)
                {
                    value = nom_value - 1;
                }
                else if (Math.Abs(max_value - nom_value) < 13)
                {
                    value = nom_value - 3;
                }
                else if (Math.Abs(max_value - nom_value) < 15)
                {
                    value = nom_value - 5;
                }
                else
                {
                    value =nom_value-5;

                }
                MessageBox.Show("RECOMMENDED    VALUE
WAS DETERMINED");//("ÖNERİLEN DEĞER BELİRLENDİ");
                textBox5.Visible = true;
                textBox5.Text = "";

                id = "";
                Add_to_Textbox((value & 0xf0) >> 4);
                Add_to_Textbox((value & 0x0f));
                textBox5.Visible = true;
                textBox5.Text = id;
            }
            else if (nom_value < max_value)
            {
                if (Math.Abs(max_value - nom_value) > 0 &&
Math.Abs(max_value - nom_value) < 3)
                {
                    value = nom_value - 5;
                }
                else if (Math.Abs(max_value - nom_value) < 5)
                {
                    value = nom_value - 3;
                }
                else if (Math.Abs(max_value - nom_value) < 7)
                {
                    value = nom_value - 1;
                }
                else if (Math.Abs(max_value - nom_value) < 9)
                {
                    value = nom_value;
                    // nom_value = nom_value;
                }
                else if (Math.Abs(max_value - nom_value) < 11)
                {
                    value = nom_value + 1;
                }
                else if (Math.Abs(max_value - nom_value) < 13)
                {
                    value = nom_value + 3;
                }
                else if (Math.Abs(max_value - nom_value) < 15)
                {
                    value = nom_value + 5;
                }
                else
                {
                    value = nom_value+5 ;

                }
                MessageBox.Show("RECOMMENDED    VALUE
WAS DETERMINED");//("ÖNERİLEN DEĞER BELİRLENDİ");
                textBox5.Visible = true;
                textBox5.Text = "";

                id = "";
                Add_to_Textbox((value & 0xf0) >> 4);
                Add_to_Textbox((value & 0x0f));
                textBox5.Visible = true;
                textBox5.Text = id;
            }
            else
            {
                value = nom_value - 5;
                MessageBox.Show("RECOMMENDED    VALUE
WAS DETERMINED");//("ÖNERİLEN DEĞER BELİRLENDİ");
                textBox5.Visible = true;
                textBox5.Text = "";

                id = "";
                Add_to_Textbox((value & 0xf0) >> 4);
                Add_to_Textbox((value & 0x0f));
                textBox5.Visible = true;
                textBox5.Text = id;
            }
            degerleri_gonder.Visible = true;
        }

        int calcrc(string ptr, int length, int crc_ini)
        {
            int crcresult;
            int kcrc, icrc, tempcrc;
            int carry_in = 0, carry_out = 0;
            crcresult = crc_ini;
            //WDTCTL = WDTPW + WDTCNTCL + WDTIS2;
            for (icrc = 0; icrc < length; icrc++)
            {
                tempcrc = (reset_array[icrc]);
                for (kcrc = 0; kcrc < 8; kcrc++)
                {
                    if ((crcresult & 0x01) != (tempcrc & 0x01))
                    {
                        carry_in = 1;
                        crcresult ^= 0x4002;
                    }
                    else carry_in = 0;
                    if    (Convert.ToBoolean(crcresult    &    0x01))
carry_out = 1;
                    else carry_out = 0;
                    crcresult >>= 1; tempcrc >>= 1;
                    if   (Convert.ToBoolean(carry_in))   crcresult   |=
0x8000;
                    if   (Convert.ToBoolean(carry_out))   tempcrc   |=
0x80;
                }
            }
            //WDTCTL = WDTPW + WDTHOLD;
            return (crcresult);
        }

        int calcrc2(string ptr, int length, int crc_ini)
        {
            int crcresult;
            int kcrc, icrc, tempcrc;
            int carry_in = 0, carry_out = 0;
            crcresult = crc_ini;
            //WDTCTL = WDTPW + WDTCNTCL + WDTIS2;
            for (icrc = 0; icrc < length; icrc++)
            {
                tempcrc = (kb_active_array[icrc]);
                for (kcrc = 0; kcrc < 8; kcrc++)
                {
                    if ((crcresult & 0x01) != (tempcrc & 0x01))
                    {
                        carry_in = 1;
                        crcresult ^= 0x4002;
                    }
                    else carry_in = 0;
                    if    (Convert.ToBoolean(crcresult    &    0x01))
carry_out = 1;
                    else carry_out = 0;
                    crcresult >>= 1; tempcrc >>= 1;
                    if   (Convert.ToBoolean(carry_in))   crcresult   |=
0x8000;
                    if   (Convert.ToBoolean(carry_out))   tempcrc   |=
0x80;
                }
            }
            //WDTCTL = WDTPW + WDTHOLD;
            return (crcresult);
        }

        int calcrc1(string ptr, int length, int crc_ini)
        {
            int crcresult;
            int kcrc, icrc, tempcrc;
            int carry_in = 0, carry_out = 0;
            crcresult = crc_ini;
            //WDTCTL = WDTPW + WDTCNTCL + WDTIS2;
```

```
                for (icrc = 0; icrc < length; icrc++)
                {
                    tempcrc = (Msg[icrc]);
                    for (kcrc = 0; kcrc < 8; kcrc++)
                    {
                        if ((crcresult & 0x01) != (tempcrc & 0x01))
                        {
                            carry_in = 1;
                            crcresult ^= 0x4002;
                        }
                        else carry_in = 0;
                        if    (Convert.ToBoolean(crcresult    &    0x01))
carry_out = 1;
                        else carry_out = 0;
                        crcresult >>= 1; tempcrc >>= 1;
                        if   (Convert.ToBoolean(carry_in))   crcresult   |=
0x8000;
                        if   (Convert.ToBoolean(carry_out))   tempcrc   |=
0x80;
                    }
                }
                //WDTCTL = WDTPW + WDTHOLD;
                return (crcresult);
            }

            public Form1()
            {
                InitializeComponent();

                foreach (string s in SerialPort.GetPortNames())
                {
                    // MessageBox.Show(s);

                    Comport.Items.Add(s);
                    //Console.WriteLine("   {0}", s);
                }
                progressBar1.Visible = false;
                progressBar2.Visible = false;
                okuma.Visible = true;
                kesme.Visible = true;
                degerleri_gonder.Visible = false;
                textBox10.Visible = false;
                textBox11.Visible = false;
                textBox12.Visible = false;
                textBox13.Visible = false;
                textBox14.Visible = false;
                label8.Visible = false;
                label9.Visible = false;
                label10.Visible = false;
                label11.Visible = false;
                label12.Visible = false;
                label13.Visible = false;
                label14.Visible = false;
                mesafe_reset.Enabled = false;
                degerleri_gonder.Enabled = false;
                button4.Enabled = false;
//  textBox1.Visible = false;
 // textBox2.Visible = false;
 // textBox5.Visible = false;
  ////////////////////////////////
 reset_array[0] = 0x02;
 reset_array[1] = 0x52;
 reset_array[2] = 0x30;
 reset_array[3] = 0x31;
 reset_array[4] = 0x31;
 reset_array[5] = 0x38;
 reset_array[6] = 0xff;
 reset_array[7] = 0x22;
 reset_array[8] = 0x33;
 reset_array[9] = 0x44;
 reset_array[10] = 0x55;
 reset_array[11] = 0x66;
 reset_array[12] = 0x77;
 reset_array[13] = 0x11;
 reset_array[14] = 0x22;
 reset_array[15] = 0x33;
 reset_array[16] = 0x44;
 reset_array[17] = 0x55;
 reset_array[18] = 0x66;
 reset_array[19] = 0x77;
 reset_array[20] = 0x88;

 reset_array[21] = 0xa9;
 reset_array[22] = 0x92;
 reset_array[23] = 0x03;
 reset_array[24] = 0xfa;
 ////////////////////////////////
 kb_reset_array[0] = 0x02;
 kb_reset_array[1] = 0x52;
 kb_reset_array[2] = 0x30;
 kb_reset_array[3] = 0x31;
 kb_reset_array[4] = 0x31;
 kb_reset_array[5] = 0x39;
 kb_reset_array[6] = 0xff;
 kb_reset_array[7] = 0x22;
 kb_reset_array[8] = 0x33;
 kb_reset_array[9] = 0x44;
 kb_reset_array[10] = 0x55;
 kb_reset_array[11] = 0x66;
 kb_reset_array[12] = 0x77;
 kb_reset_array[13] = 0x11;
 kb_reset_array[14] = 0x22;
 kb_reset_array[15] = 0x33;
 kb_reset_array[16] = 0x44;
 kb_reset_array[17] = 0x55;
 kb_reset_array[18] = 0x66;
 kb_reset_array[19] = 0x77;
 kb_reset_array[20] = 0x88;

 kb_reset_array[21] = 0x68;
 kb_reset_array[22] = 0x02;
 kb_reset_array[23] = 0x03;
 kb_reset_array[24] = 0xfa;

 ////////////////////////////////////////
 kb_config_array[0] = 0x02;
 kb_config_array[1] = 0x52;
 kb_config_array[2] = 0x30;
 kb_config_array[3] = 0x31;
 kb_config_array[4] = 0x31;
 kb_config_array[5] = 0x3A;
 kb_config_array[6] = 0xff;
 kb_config_array[7] = 0x22;
 kb_config_array[8] = 0x33;
 kb_config_array[9] = 0x44;
 kb_config_array[10] = 0x55;
 kb_config_array[11] = 0x66;
 kb_config_array[12] = 0x77;
 kb_config_array[13] = 0x11;
 kb_config_array[14] = 0x22;
 kb_config_array[15] = 0x33;
 kb_config_array[16] = 0x44;
 kb_config_array[17] = 0x55;
 kb_config_array[18] = 0x66;
 kb_config_array[19] = 0x77;
 kb_config_array[20] = 0x88;

 kb_config_array[21] = 0x28;
 kb_config_array[22] = 0xf3;
 kb_config_array[23] = 0x03;
 kb_config_array[24] = 0xfa;
 ////////////////////////////////////////
 kb_active_array[0] = 0x02;
 kb_active_array[1] = 0x52;
 kb_active_array[2] = 0x30;
 kb_active_array[3] = 0x31;
 kb_active_array[4] = 0x31;
 kb_active_array[5] = 0x3B;
 kb_active_array[6] = 0x11;
 kb_active_array[7] = 0x22;
 kb_active_array[8] = 0x33;
 kb_active_array[9] = 0x44;
 kb_active_array[10] = 0x55;
 kb_active_array[11] = 0x66;
 kb_active_array[12] = 0x77;
 kb_active_array[13] = 0x11;
 kb_active_array[14] = 0x22;
 kb_active_array[15] = 0x33;
 kb_active_array[16] = 0x44;
 kb_active_array[17] = 0x55;
 kb_active_array[18] = 0x66;
 kb_active_array[19] = 0x77;
 kb_active_array[20] = 0x88;

 kb_active_array[21] = 0x87;
 kb_active_array[22] = 0x2f;
 kb_active_array[23] = 0x03;
 kb_active_array[24] = 0xfa;
```

```
////////////////////////////////////
get_ref_array[0] = 0x02;
get_ref_array[1] = 0x52;
get_ref_array[2] = 0x30;
get_ref_array[3] = 0x31;
get_ref_array[4] = 0x31;
get_ref_array[5] = 0x3c;
get_ref_array[6] = 0x11;
get_ref_array[7] = 0x22;
get_ref_array[8] = 0x33;
get_ref_array[9] = 0x44;
get_ref_array[10] = 0x55;
get_ref_array[11] = 0x66;
get_ref_array[12] = 0x77;
get_ref_array[13] = 0x11;
get_ref_array[14] = 0x22;
get_ref_array[15] = 0x33;
get_ref_array[16] = 0x44;
get_ref_array[17] = 0x55;
get_ref_array[18] = 0x66;
get_ref_array[19] = 0x77;
get_ref_array[20] = 0x88;

get_ref_array[21] = 0xc5;
get_ref_array[22] = 0x1d;
get_ref_array[23] = 0x03;
get_ref_array[24] = 0xfa;
////////////////////////////////////
Msg[0] = 0x02;
Msg[1] = 0x52;
Msg[2] = 0x30;
Msg[3] = 0x31;
Msg[4] = 0x31;
Msg[5] = 0x33;
Msg[6] = 0x11;
Msg[7] = 0x22;
Msg[8] = 0x33;
Msg[9] = 0x44;
Msg[10] = 0x55;
Msg[11] = 0x66;
Msg[12] = 0x77;

Msg[13] = 0x11;
Msg[14] = 0x22;
Msg[15] = 0x33;
Msg[16] = 0x44;
Msg[17] = 0x55;
Msg[18] = 0x66;
Msg[19] = 0x77;
Msg[20] = 0x88;

Msg[21] = 0x4d;
Msg[22] = 0x06;
Msg[23] = 0x03;
Msg[24] = 0xfa;
////////////////////////////////////
```

```
}

private void mesafe_reset_Click(object sender,
EventArgs e)
{
   okuma.Visible = true;
   reset_click = true;
   label2.Visible = false;
   label3.Visible = false;
   label4.Visible = false;
   reset_array[6] = 0xff;
   textBox5.Visible = false;
   crcresss = calcrc(Convert.ToString(reset_array), 21,
0);

   reset_array[21] = (byte)(crcresss & 0x000000ff);
   reset_array[22] = (byte)((crcresss & 0x0000ff00) >>
8);

   reset_array[23] = 0x03;
   reset_array[24] = 0xfa;
   if (serialPort1.IsOpen)
   {
      serialPort1.Write(reset_array, 0, 26);
      timer1.Interval = 1000;
      timer1.Start();
   }
   else
   {
      serialPort1.PortName = Comport.Items.ToString();
      serialPort1.Close();
      serialPort1.Open();
      serialPort1.Write(reset_array, 0, 26);
      timer1.Start();
   }
}



private void timer1_Tick(object sender, EventArgs e)
{
   if (KB_RESET1 || KB_ACTIVE1)
   {
      KB_RESET = false;
      KB_RESET1 = false;
      label13.Visible = false;
      label14.Visible = false;

         // timer1.Interval = 5000;
      timer1.Stop();
      Application.Restart();
   }
  // timer1.Stop();
```

```
                if (serialPort1.BytesToRead > 0)//////////24 yaparak da
olabilir
                {
                   Boolean data_found = false;
                 //  Int32 deneme;
                   int j,k;
                   byte[] bfr = new byte[40];
                   serialPort1.Read(bfr, 0, bfr.Length);
                   for (j = 0; j < 40; j++)
                   {
                      if (bfr[j] == 0xfa && bfr[j - 1] == 0x03)
                      {
                         if (j >= 24 && bfr[0] == 0x02 && bfr[1] ==
0x52) { data_lenght_ok = true; buffer_offset = 0; break; }
                         else { data_lenght_ok = false; buffer_offset =
0; }
                      }
                      //else if (serialPort1.BytesToRead > 28) {
buffer_offset = 0; data_lenght_ok = false; }
                      //else if (bfr.Length < 24) { buffer_offset =
bfr.Length + 1; data_lenght_ok = false; }
                   }
                      //if (bfr[bfr.Length] == 0xfa &&
bfr.Length>=24) { data_lenght_ok = true; buffer_offset = 0; }
                      //else { buffer_offset = bfr.Length +
1;data_lenght_ok=false; }
                   UTF8Encoding enc = new UTF8Encoding();
                   string income = enc.GetString(bfr, 0, bfr.Length);
                   //deneme = Convert.ToInt32(income[7]);
                   ////////////////////////////////////////////////////////
                   if (reset_click)
                   {
                      data_lenght_ok = false;
                      buffer_offset = 0;
                      if (0xff == bfr[7])
                      {
                         timer1.Stop();
                         reset_click = false;
                         MessageBox.Show("IDENTIFIER
COMMUNICATION VALUE RESET IS SUCCESSFUL");///("KB
Mesafe Değeri Resetlendi");
                         //IDENTIFIER COMMUNICATION VALUE
RESET SUCCESFULLY


                      }
                      else
                      {
                         reset_click = false;
                         timer1.Stop();
                         MessageBox.Show("IDENTIFIER
COMMUNICATION VALUE RESET IS NOT SUCCESSFUL");

                      }

                   }
```

```
            else if (KB_RESET)
            {
                buffer_offset = 0; data_lenght_ok = false;
                if (bfr[5] == 0x90 && bfr[1] == 0x52)
                {
                    KB_RESET = false;
                    KB_RESET1 = true;
                    label13.Visible = true;
                    timer1.Interval = 7000;
                }
                else
                {
                    //serial_reseted = false;
                    MessageBox.Show("COMMUNICATION
ERROR");
                    for (j = 0; j < 40; j++)
                        bfr[j] = 0;
                    KB_RESET = false;
                    KB_RESET1 = false;
                    serialPort1.Close();
                    serialPort1.Open();
                }
            }
            else if (KB_ACTIVE)
            {
                buffer_offset = 0;
                data_lenght_ok = false;
                if (bfr[5] == 0x61 && bfr[1] == 0x52)
                {
                    KB_ACTIVE = false;
                    KB_ACTIVE1 = true;
                    label14.Visible = true;
                    timer1.Interval = 7000;
                }
                else
                {
                    //serial_reseted = false;
                    MessageBox.Show("COMMUNICATION
ERROR");
                    for (j = 0; j < 40; j++)
                        bfr[j] = 0;
                    KB_ACTIVE = false;
                    KB_ACTIVE1 = false;
                    serialPort1.Close();
                    serialPort1.Open();
                }
            }
            else if (deger_gonder)
            {
                buffer_offset = 0;
                data_lenght_ok = false;
                deger_gonder = false;
                if (sent_value == bfr[7])
                {
                    MessageBox.Show("VALUE SEND IS
SUCCESSFUL");
                    timer1.Stop();
                    Application.Restart();
                }
                else
                {
                    MessageBox.Show("VALUE SEND IS NOT
SUCCESSFUL");
                    timer1.Stop();
                }
                reset_click = false;

            }
            else if (tx_powersend)
            {
                buffer_offset = 0;
                data_lenght_ok = false;
                tx_powersend = false;
                if (Msg[6] == Convert.ToInt32(income[6]))
                {
                    textBox8.BackColor = Color.Green;
                }
                else
                {
                    textBox8.BackColor = Color.Red;

                }
                Msg[6] = 0;
            }
            ////////////////////////////////////////////////////
            else if (Config_Gonder)
            {
                buffer_offset = 0;
                data_lenght_ok = false;
                timer1.Stop();
                id1 = "";
                if (bfr[5] == 0x82 && bfr[1] ==
0x52)//////düzeltecem
                {
                    int i;
                    Config_Gonder = false;
                    textBox10.Visible = true;
                    textBox11.Visible = true;
                    textBox12.Visible = true;
                    textBox13.Visible = true;
                    textBox14.Visible = true;
                    label8.Visible = true;
                    label9.Visible = true;
                    label10.Visible = true;
                    label11.Visible = true;
                    label12.Visible = true;
                    for (i = 0; i < 8; i++)
                    {

                        Add_to_Textbox1((bfr[i + 6] & 0xf0) >> 4);
                        Add_to_Textbox1((bfr[i + 6] & 0x0f));
                        textBox10.Text = id1;
                        textBox10.BackColor = Color.Aqua;
                    }
                    if (bfr[14] == (0xff - bfr[15]))
                    {
                        id1 = "";
                        Add_to_Textbox1((bfr[14] & 0xf0) >> 4);
                        Add_to_Textbox1((bfr[14] & 0x0f));
                        textBox11.Width = 40;
                        textBox11.Text = id1;
                        //textBox11.BackColor = Color.Aqua;
                    }
                    else
                    {
                        textBox11.Width = 200;
                        textBox11.BackColor = Color.Red;
                        //textBox11.Text = Color.Black;
                        textBox11.Text = "FAILED ";
                        Add_to_Textbox1((bfr[14] & 0xf0) >> 4);
                        Add_to_Textbox1((bfr[14] & 0x0f));
                        Add_to_Textbox1((bfr[15] & 0xf0) >> 4);
                        Add_to_Textbox1((bfr[15] & 0x0f));

                    }

                    if (bfr[16] == (0xff - bfr[17]))
                    {
                        id1 = "";
                        Add_to_Textbox1((bfr[16] & 0xf0) >> 4);
                        Add_to_Textbox1((bfr[16] & 0x0f));
                        //textBox12.BackColor = Color.Aqua;
                        textBox12.Width = 40;
                        textBox12.Text = id1;
                    }
                    else
                    {
                        //textBox12.BackColor =
Color.AntiqueWhite;
                        textBox12.Width = 120;
                        textBox12.Location = new Point(129, 134);
                        textBox12.Text = "DEFAULT";
                    }

                    if (bfr[20] == (0xff - bfr[21]))
                    {
                        id1 = "";
                        //textBox13.BackColor = Color.Aqua;
                        Add_to_Textbox1((bfr[20] & 0xf0) >> 4);
                        Add_to_Textbox1((bfr[20] & 0x0f));
                        textBox13.Width = 40;
                        textBox13.Text = id1;
                    }
                    else
                    {
                        textBox12.Text = "DEFAULT";
                    }
                    if (bfr[18] == 0x55 && bfr[19] == 0x11)
```

```
                {
                    id1 = "OK";
                    //Add_to_Textbox1((bfr[14] & 0xf0) >> 4);
                    // Add_to_Textbox1((bfr[14] & 0x0f));
                    textBox14.Width = 40;
                    textBox14.Text = id1;
                    //textBox11.BackColor = Color.Aqua;
                }
                else
                {
                    textBox14.Width = 90;
                    textBox14.BackColor = Color.Red;
                    //textBox11.Text = Color.Black;
                    textBox14.Location = new Point(173, 208);
                    textBox14.Text = "FAILED ";


                }
            }

            else
            {
                MessageBox.Show("COMMUNICATION
ERROR");

                for (j = 0; j < 40; j++)
                    bfr[j] = 0;
                serialPort1.Close();
                serialPort1.Open();
                textBox10.Visible = false;
                textBox11.Visible = false;
                textBox12.Visible = false;
                textBox13.Visible = false;
                textBox14.Visible = false;

            }
        }
        ////////////////////////////////////////////////////////////
        else if (get_ref)
        {
            get_ref = false;
            send_password = true;
            textBox15.Text = "";
            for (j = 0; j < 10; j++)
            {
                textBox15.Text = textBox15.Text +
Convert.ToString(bfr[j + 6]);
            }
            button5.Text = "Send Password";                    bfr[6]);

                }
        ////////////////////////////////////////////////////////////
        //  Msg[6] = 0;
        else if (data_lenght_ok)
        {
            buffer_offset = 0;
            data_lenght_ok = false;

            int[] temp_buf = new int[50];
            int temp_deger = 0;
            int i;
            id = textBox3.Text;
            id = "          ";
            id = " xxx";

            for (i = 0; i < 8; i++)
            {
                Add_to_Textbox((bfr[i + 8] & 0xf0) >> 4);
                Add_to_Textbox((bfr[i + 8] & 0x0f));
            }
            textBox3.Text = id;

            id = id + "     ";
            i = 0;
            temp_i = 0xff - bfr[6];
            Add_to_Textbox((temp_i & 0xf0) >> 4);
            Add_to_Textbox((temp_i & 0x0f));
            //  textBox4.Text += id;

            id = id + "     ";
            i = 0;
            temp_i = bfr[6];
            Add_to_Textbox((temp_i & 0xf0) >> 4);
            Add_to_Textbox((temp_i & 0x0f));
            textBox4.Text += id;

            id = "";
            temp_i = bfr[7];
            Add_to_Textbox((temp_i & 0xf0) >> 4);
            Add_to_Textbox((temp_i & 0x0f));
            textBox7.Text = id;
            if (!((0x10 < temp_i) && (temp_i < 0x30)))
            {

                textBox6.BackColor = Color.Red;
            }
            else
            {
                textBox6.BackColor = Color.Green;
            }
            timer1.Start();
            if (okuma_degeri)
            {
                nom_value += (0xff - bfr[6]);
                okuma_temp[timer_count]   =   (byte)(0xff   -

                timer_count++;
                progressBar1.Value++;
                if (timer_count > 9)
                {
                    temp_deger = okuma_temp[0];
                    ////////////////////////////////////


                    for (j = 0; j < 9; j++)
                    {
                        temp_deger = okuma_temp[0];
                        for (k = 0; k < 9; k++)
                        {
                            if (temp_deger > okuma_temp[k + 1])
                            {
                                temp_deger = okuma_temp[k + 1];
                            }
                        }
                        okuma_temp1[j] = (byte)temp_deger;
                        for (k = 0; k < 10; k++)
                        {
                            if (temp_deger == okuma_temp[k] &&
!data_found)
                            {
                                okuma_temp[k] = 0xff;
                                data_found = true;
                            }
                        }
                        data_found = false;
                    }
                    ////////////////////////////////////////
                    nom_value = 0;
                    nom_value += (okuma_temp1[3]);
                    nom_value += (okuma_temp1[4]);

                    nom_value += (okuma_temp1[5]);
                    nom_value += (okuma_temp1[6]);

                    nom_value = nom_value / 4;
                    id = "";
                    Add_to_Textbox((nom_value   &   0xf0)   >>
4);

                    Add_to_Textbox((nom_value & 0x0f));
                    textBox1.Visible = true;
                    textBox1.Text += id;
                    timer_count = 0;
                    progressBar1.Value = 0;
                    okuma_degeri = false;
                    timer1.Stop();
                    MessageBox.Show("COMMUNICATION
VALUE WAS DETERMINED");
                    kesme.Visible = true;
                }

            }
            if (kesme_degeri)
            {
                max_value += (0xff - bfr[6]);
                okuma_temp[timer_count]   =   (byte)(0xff   -
bfr[6]);

                timer_count++;
                progressBar2.Value++;
                if (timer_count > 9)
                {
                    ////////////////////////////////////////
```

```
                    for (j = 0; j < 9; j++)
                    {
                        temp_deger = okuma_temp[0];
                        for (k = 0; k < 9; k++)
                        {
                            if (temp_deger > okuma_temp[k + 1])
                            {
                                temp_deger = okuma_temp[k + 1];
                            }
                        }
                        okuma_temp1[j] = (byte)temp_deger;
                        for (k = 0; k < 10; k++)
                        {
                            if (temp_deger == okuma_temp[k] &&
!data_found)
                            {
                                okuma_temp[k] = 0xff;
                                data_found = true;
                            }
                        }
                        data_found = false;
                    }

                    ////////////////////////////////////////

                    max_value = 0;
                    max_value += (okuma_temp1[3]);
                    max_value += (okuma_temp1[4]);

                    max_value += (okuma_temp1[5]);
                    max_value += (okuma_temp1[6]);

                    max_value = max_value / 4;


                    textBox2.Text = "";
                    //max_value = max_value / 10;
                    id = "";
                    Add_to_Textbox((max_value  &  0xf0)  >>
4);
                    Add_to_Textbox((max_value & 0x0f));
                    textBox2.Visible = true;
                    textBox2.Text += id;

                    progressBar2.Value = 0;
                    kesme_degeri = false;
                    timer1.Stop();
                    MessageBox.Show("DISCONTINUATION
VALUE WAS DETERMINED");
                    calc_value();
                    //Kesme_degeri.Visible = true;
                }

            }

        }
    }
    else
    {
        if (Config_Gonder || KB_RESET || KB_ACTIVE ||
reset_click)
        {
            data_lenght_ok = false;
            timer1.Stop();
            serial_reseted = true;
            MessageBox.Show("COMMUNICATION
ERROR");

            //for (j = 0; j < 40; j++)
            //   bfr[j] = 0;
            serialPort1.Close();
            serialPort1.Open();
            //serialPort1.BytesToRead = 0;
            textBox10.Visible = false;
            textBox11.Visible = false;
            textBox12.Visible = false;
            textBox13.Visible = false;

            textBox14.Visible = false;

            Config_Gonder = false;

        }
       //  timer1.Stop();
        // serialPort1.Close();
         //serialPort1.Open();
        //  MessageBox.Show("İŞLEM BAŞARISIZ");

    }

}

private    void    textBox2_TextChanged(object    sender,
EventArgs e)
{
  // max_value = Convert.ToByte(textBox2.Text);
}

private    void    degerleri_gonder_Click(object    sender,
EventArgs e)
{
    Int16 i = 0, j = 0;
    string b;
    string[] has = new string[50];
    b = textBox5.Text;
    for (i = 0; i < textBox5.Text.Length; i++)
    {


        has[i] = (b.Substring(i, 1));
        switch (has[i])
        {
            case "A":
                has[i] = (10).ToString();
                break;
            case "B":
                has[i] = (11).ToString();
                break;
            case "C":
                has[i] = (12).ToString();
                break;
            case "D":
                has[i] = (13).ToString();
                break;
            case "E":
                has[i] = (14).ToString();
                break;
            case "F":
                has[i] = (15).ToString();
                break;

        }
        //    j = Convert.ToByte(has[i]);
        /* if (Convert.ToByte(has[i]) > 0x39)
         {
            has[i] = (Convert.ToByte(has[i]) - 7).ToString();
         }*/
        //has[i] = Convert.ToByte(textBox1.Text);
    }
    value    =    0x10    *    (Convert.ToInt16(has[0]))    +
Convert.ToInt16(has[1]);
    //value = Convert.ToByte(textBox5.Text);
    reset_array[6] =Convert.ToByte( value);
    crcresss  =  calcrc(Convert.ToString(reset_array),  21,
0);
    reset_array[21] = (byte)(crcresss & 0x000000ff);
    reset_array[22]  =  (byte)((crcresss  &  0x0000ff00)  >>
8);
    reset_array[23] = 0x03;
    reset_array[24] = 0xfa;
    serialPort1.Write(reset_array, 0, 26);
    timer1.Interval = 1000;
    timer1.Start();
    deger_gonder = true;
    sent_value = reset_array[6];

}

private void YENILE_Click(object sender, EventArgs e)
{
    Int16 i = 0,j=0;
    string b;
    string[] has = new string[50];
    b = textBox2.Text;
    for (i = 0; i < textBox2.Text.Length; i++)
    {
```

```
                has[i] = (b.Substring(i, 1));
                switch(has[i])
                {
                   case "A":
                      has[i] = (10).ToString();
                      break;
                   case "B":
                      has[i] = (11).ToString();
                      break;
                   case "C":
                      has[i] = (12).ToString();
                      break;
                   case "D":
                      has[i] = (13).ToString();
                      break;
                   case "E":
                      has[i] = (14).ToString();
                      break;
                   case "F":
                      has[i] = (15).ToString();
                      break;

                }
             //   j = Convert.ToByte(has[i]);
               /* if (Convert.ToByte(has[i]) > 0x39)
                {
                   has[i] = (Convert.ToByte(has[i]) - 7).ToString();
                }*/
                //has[i] = Convert.ToByte(textBox1.Text);
             }
             max_value = 0x10 * (Convert.ToInt16(has[0])) +
Convert.ToInt16(has[1]);

             b = textBox1.Text;
             for (i = 0; i < textBox1.Text.Length; i++)
             {


                has[i] = (b.Substring(i, 1));
                switch (has[i])
                {
                   case "A":
                      has[i] = (10).ToString();
                      break;
                   case "B":
                      has[i] = (11).ToString();
                      break;
                   case "C":
                      has[i] = (12).ToString();
                      break;
                   case "D":
                      has[i] = (13).ToString();
                      break;
                   case "E":
                      has[i] = (14).ToString();
                      break;
                   case "F":
                      has[i] = (15).ToString();
                      break;

                }
                /*  if (Convert.ToByte(has[i])>0x39)
                {
                   has[i] =(Convert.ToByte( has[i] )- 7).ToString();
                }*/
                //has[i] = Convert.ToByte(textBox1.Text);
             }
             nom_value = 0x10 * (Convert.ToInt16(has[0])) +
Convert.ToInt16(has[1]);
             calc_value();
             label4.Visible = false;
             degerleri_gonder.Visible = true;
          //   value = Convert.ToByte(textBox5.Text);
          }

          private void button1_Click(object sender, EventArgs e)
          {
             byte[] bfr = new byte[serialPort1.BytesToRead];
             serialPort1.Read(bfr, 0, bfr.Length);
             label2.Visible = false;
             label4.Visible = false;
             //serialPort1.Close();
             //serialPort1.Open();
             progressBar1.Visible = true;
             timer1.Interval = 4000;
             timer1.Start();
             okuma_degeri = true;
             progressBar1.Value = 0;
             textBox4.Text = "";
             textBox1.Text = "";
             nom_value = 0;
             kesme_degeri = false;
          }

          private void cg_deger_gonder_Click(object sender,
EventArgs e)
          {
             Int16 i = 0;
             string b;
             string[] has = new string[50];
             //byte[] has= new byte[50];
             // int i = 0;
             b = textBox8.Text;
             for (i = 0; i < textBox8.Text.Length; i++)
             {


                has[i] = (b.Substring(i, 1));

                //has[i] = Convert.ToByte(textBox1.Text);
             }
             Msg[6] = 0x10 * (Convert.ToInt16(has[0])) +
Convert.ToInt16(has[1]);
             byte[] send = new byte[50];
             crcresss = calcrc1(Convert.ToString(Msg), 22, 0);
             Msg[22] = crcresss & 0x000000ff;
             Msg[23] = (crcresss & 0x0000ff00) >> 8;
             Msg[24] = 0x03;
             Msg[25] = 0xfa;
             for (i = 0; i < 26; i++)
             {


                send[i] = Convert.ToByte(Msg[i]);//
Convert.ToInt16(a.Substring(i, 1));

                //has[i] = Convert.ToByte(textBox1.Text);
             }
             textBox8.BackColor = Color.Yellow;
             serialPort1.Write(send, 0, 26);
             tx_powersend = true;
             timer1.Interval = 1000;
             timer1.Start();
          }

          private void STOP_Click(object sender, EventArgs e)
          {
             timer1.Stop();
             textBox4.Text = "";
             textBox1.Text = "";
             nom_value = 0;
             progressBar1.Value = 0;
          }

          private void button1_Click_1(object sender, EventArgs
e)
          {
             Application.Restart();
          }

          private void okuma_Click_1(object sender, EventArgs e)
          {
            //  private void okuma_Click(object sender, EventArgs
e)
          //{
             byte[] bfr = new byte[serialPort1.BytesToRead];
             serialPort1.Read(bfr, 0, bfr.Length);
             label2.Visible = false;
             label4.Visible = false;
             textBox5.Visible = false;
             //serialPort1.Close();
             //serialPort1.Open();
             progressBar1.Visible = true;
             timer1.Interval = 500;//////////////unutma 4000 di
             timer1.Start();
             okuma_degeri = true;
             progressBar1.Value = 0;
```

```
            textBox4.Text = "";
            textBox1.Text = "";
            nom_value = 0;
            kesme_degeri = false;

        //}
        }

        private void kesme_Click_1(object sender, EventArgs e)
        {
          //  private void kesme_Click(object sender, EventArgs
e)
        // {
            byte[] bfr = new byte[serialPort1.BytesToRead];
            serialPort1.Read(bfr, 0, bfr.Length);
            label3.Visible = false;
            label4.Visible = false;
          //  serialPort1.Close();
          // serialPort1.Open();
            timer1.Start();
            progressBar2.Visible = true;
            progressBar2.Value = 0;
            textBox4.Text = "";
            textBox2.Text = "";
            max_value = 0;
            kesme_degeri = true;
            okuma_degeri = false;
            timer_count = 0;
        //}
        }

        private void connect_Click_1(object sender, EventArgs
e)
        {
         //  private void connect_Click(object sender, EventArgs
e)
        //{
            //serialPort1.PortName                    =
Comport.SelectedItem.ToString();
            serialPort1.Close();
            serialPort1.Open();
        //}
        }
        private void Comport_SelectedIndexChanged_1(object
sender, EventArgs e)
        {
         //  private void Comport_SelectedIndexChanged(object
sender, EventArgs e)
        //{
            serialPort1.Close();
            serialPort1.PortName                    =
Comport.SelectedItem.ToString();
        //}
        }
```

```
e)
        private void button2_Click_1(object sender, EventArgs
        {
            textBox10.Visible = false;
            textBox11.Visible = false;
            textBox12.Visible = false;
            textBox13.Visible = false;
            textBox14.Visible = false;
            label8.Visible = false;
            label9.Visible = false;
            label10.Visible = false;
            label11.Visible = false;
            label12.Visible = false;
            label13.Visible = false;
            label14.Visible = false;

            KB_RESET = true;
            kb_reset_array[23] = 0x03;
            kb_reset_array[24] = 0xfa;
            serialPort1.Write(kb_reset_array, 0, 26);
            timer1.Interval = 1000;
            timer1.Start();
            ///deger_gonder = true;
        }
        private void button3_Click(object sender, EventArgs e)
        {
            label14.Visible = false;
            kb_config_array[23] = 0x03;
            kb_config_array[24] = 0xfa;
            serialPort1.Write(kb_config_array, 0, 26);
            timer1.Interval = 500;
            timer1.Start();
            Config_Gonder = true;
        }
        private    void    textBox9_TextChanged(object    sender,
EventArgs e)
        {
        }
        private void button4_Click(object sender, EventArgs e)
        {
            textBox10.Visible = false;
            textBox11.Visible = false;
            textBox12.Visible = false;
            textBox13.Visible = false;
            textBox14.Visible = false;
            label8.Visible = false;
            label9.Visible = false;
            label10.Visible = false;
            label11.Visible = false;
            label12.Visible = false;
            label13.Visible = false;
            label14.Visible = false;
            KB_ACTIVE = true;
            crcresss = calcrc2(Convert.ToString(kb_active_array),
21, 0);

            kb_active_array[21] = (byte)(crcresss & 0x000000ff);
```

```
            kb_active_array[22] = (byte)((crcresss & 0x0000ff00)
>> 8);
            //reset_array[23] = 0x03;
            //reset_array[24] = 0xfa;
            kb_active_array[23] = 0x03;
            kb_active_array[24] = 0xfa;
            serialPort1.Write(kb_active_array, 0, 26);
            timer1.Interval = 1000;
            timer1.Start();
        }
        private void button5_Click(object sender, EventArgs e)
        {
            if (!send_password)
            {
                get_ref = true;
                send_password = false;
                serialPort1.Close();
                serialPort1.Open();
                serialPort1.Write(get_ref_array, 0, 26);
                timer1.Interval = 1000;
                timer1.Start();
            }
            else if(textBox16.TextLength>9)
            {
                string b;
                int[] has = new int[50];
                get_ref = false;
                send_password = false;
                button5.Text = "GET REF.NUMBER";
                Int16 i = 0, j = 0;

                b = textBox16.Text;
                for (i = 0; i < textBox16.TextLength; i++)
                {
                    has[i] = Convert.ToByte(b.Substring(i, 1));
                  //  has[i] = ((has[i]) - 0x30);
                    reset_array[i + 7] =Convert.ToByte( has[i]);
                    kb_active_array[i + 7] = Convert.ToByte(has[i]);
                }
                mesafe_reset.Enabled = true;
                degerleri_gonder.Enabled = true;
                button4.Enabled = true ;
            }
        }
        private void textBox16_KeyPress(object sender, KeyPressEventArgs
e)
        {
            if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar) &&
e.KeyChar != '.')
            {
                e.Handled = true;
            }
    }
```

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı : Hasan SELEK
Uyruğu : Türkiye Cumhuriyeti
Doğum Yeri ve Tarihi : Tekirdağ 1984
Telefon : 05543386576
Faks : ----------
E-Posta : hasanselek42@hotmail.com

EĞİTİM

Derece	Adı	İlçe	İl	Bitirme Yılı
Lise :	Meram Anadolu Lisesi	Meram	Konya	2002
Üniversite :	Mersin Üniversitesi	Çiftlikköy	Mersin	2006
Yüksek Lisans :	Konya Teknik Üniversitesi	Selçuklu	Konya	2019

İŞ DENEYİMLERİ

Yıl	Kurum	Görevi
2006-2014	Mepsan A.Ş.	Uzman Proje Yöneticisi
2014-	Butkon Asansör	Genel Müdür Yrd.

UZMANLIK ALANI

Elektronik devre tasarımı, gömülü sistem yazılımı, antenler

YABANCI DİLLER

İngilizce

YAYINLAR

Uzer D., Dündar Ö., Gültekin S.S., Ülker Y., **Selek H.**, 2010, U-Yarık Dikdörtgen Mikroserit Yama Antenler için Fiziksel Yarık Parametrelerinin Yapay Sinir Ağları ile Modellenmesi, 2. Ulusal Konya Ereğli Kemal Akman Meslek Yüksek Okulu Teblig Günleri, 1(2), sayfa 82-83.

Kontrol Edilecek Hususlar	**Evet**	**Hayır**
Sayfa yapısı uygun mu?	X	
Şekil ve çizelge başlık ve içerikleri uygun mu?	X	
Denklem yazımları uygun mu?	X	
İç kapak, onay sayfası, tez bildirimi, özet, abstract, önsöz ve/veya teşekkür uygun yazıldı mı?	X	
Tez yazımı; Giriş, Kaynak Araştırması, Materyal ve Yöntem (veya Teorik Esaslar), Araştırma Bulguları ve Tartışma, Sonuçlar ve Öneriler sıralamasında mıdır?	X	
Kaynaklar soyadı sırasına göre verildi mi?	X	
Kaynaklarda verilen her bir yayına tez içerisinde atıfta bulunuldu mu?	X	
Kaynaklar açıklanan yazım kuralına uygun olarak yazıldı mı?	X	
Tez içerisinde kullanılan şekil ve çizelgelerde kullanılan ifadeler Türkçe'ye çevrilmiş mi? (Latince ve Özel kelimeler hariçtir)	X	
Tezin içindekiler kısmı, tez içerisinde verilen başlıklara uygun hazırlanmış mı?	X	
[+]Tez Önerisi Formunun (LEE 22a veya 22b) ilk sayfası ile birlikte materyal ve yöntem kısımlarını içeren sayfaların fotokopisini tezinizin içindekiler sayfasından önce telli zımbalı formda koydunuz mu?	X	

Yukarıdaki verilen cevapların doğruluğunu kabul ediyorum.

	Unvanı Adı SOYADI	İmza
Öğrenci :	Hasan SELEK	……...
Danışman :	Doç. Dr. Seyfettin Sinan GÜLTEKİN	……

Tez tesliminde enstitü web sayfası veri tabanında yayınlanmasına **izin vermiyorum**.

Lisansüstü Eğitim Enstitüsü Onayı

Bu tez Konya Teknik Üniversitesi Lisansüstü Eğitim Enstitüsü Tez Yazım Kurallarına uygundur.

Onaylayan Adı SOYADI	Tarih	İmza
……	……	……..