MİKRODENETLEYİCİLER KULLANILARAK HIZLI VE GÜVENLİ HABERLEŞME SİSTEMLERİNİN İNCELENMESİ

İbrahim ÖZCAN[1], Davut AKDAŞ[2]
[1]ibrahimozcan@balikesir.edu.tr Balıkesir Üniversitesi, Balıkesir Meslek Yüksek Okulu, Bilgisayar Teknolojileri Bölümü, 10145 Balıkesir
[2]akdas@balikesir.edu.tr Balıkesir Üniversitesi, Mimarlık-Mühendislik Fakültesi, Elektrik-Elektronik Mühendisliği Bölümü, 10145 Balıkesir

ÖZET

Gerçekleştirilen çalışmada 4x4' lük keypad' den Hexadecimal olarak girilen bilginin, Dinamik Huffman yöntemiyle sıkıştırılıp, oluşturulan bir anahtarla XOR yöntemiyle şifrelenerek bir mikrodenetleyiciden diğer mikrodenetleyiciye hızlı ve güvenli bir şekilde gönderilmesi ve gönderildikten sonra gönderilen veri tekrar orijinal haline gerilip lcd ekranda görüntülenmesi amaçlanmıştır.

Microchip firmasının ürettiği PIC18F452 mikrodenetleyicisi yeterli hız ve hafızaya sahip olduğu için bu proje de tercih edilmiştir. Mikrodenetleyici C programa diliyle programlanmıştır.

Anahtar Kelimeler: Haberleşme, Dinamik Huffman, XOR, şifreleme PIC18F452, CCS C PIC Derleyicisi,

ABSTRACT

In the study carried out, it was aimed to monitor the information's delivery -which was entered via 4x4 keypad as hexadecimal- to compress it with Huffman method, encrypt with XOR method with a key generated, to one microcontroller to another in a fast and secure way, and after delivery, to convert the sent data back to its original form and show it on a lcd screen.

PIC18F452 micro controller that Microchip company produced had enough speed and memory, it was preferred in the project. Micro controller was programmed in C programming language.

Keywords: Communication, Dynamic Huffman, XOR, encryption, PIC18F452, CCS C PIC compiler

1. GİRİŞ

Tarih boyunca devlet işlerinde, askeri işlerde ve benzeri işlerde bilginin güvenli bir şekilde korunması ve/veya iletilmesi ile ilgili problemlere hep çözüm aranmıştır. Tarihin ilk şifreleme tekniklerinden biri olan "Monoalpabetic ciphers" tekniğini kullanan en ünlü teknik Roma imparatoru Sezar' ın kullandığı Sezar Şifresi' dir. Sezar şifrelemesinden sonra en tanınmış şifreleme yöntemlerinden biri Rotasyon Şifreleme (Rotor Machine) olmuştur. Bu makineye örnek ikinci dünya savaşı sırasında Nazi Almanya' sının kullandığı Enigma makinesi adlı cihazdır. Bu iki şifreleme yöntemi de çeşitli tekniklerle kırılmıştır. Daha sonraki geliştirilen tekniklerde şifreleme işlemlerinde

anahtar kullanımı ön plana çıkmıştır. Hatta şifreleme teknikleri anahtarların biçimine göre simetrik veya asimetrik olarak adlandırılmıştır. Simetrik şifreleme algoritmalarında şifreleme ve şifre çözme işlemlerinde aynı anahtar kullanılır. Asimetrik şifreleme algoritmalarında şifreleme ve şifre çözme işlemlerinin her biri için ayrı anahtar kullanılır. Her iki yönteminde avantajları ve dezavantajları vardır. Kullanılacakları alanlara göre biri diğerine tercih edilebilir. Simetrik şifreleme algoritmalarına verilebilecek en iyi örnekler DES ve AES şifreleme algoritmaları diyebiliriz. Asimetrik şifreleme algoritmalarına örnek olarak Elgamal, RSA, ECC, Diffie-Hellman ve DSA' yı verebiliriz.

Haberleşme söz konusu olduğunda sadece verinin güvenliği yeterli olmuyor. Günümüzde verinin güvenli bir şekilde iletilmesi ile birlikte hızda haberleşme için çok önemli bir etkendir. Verinin hızlı bir şekilde iletilmesi için verinin sıkıştırılması gerekmektedir. Çeşitli tekniklerle iletilecek veriler sıkıştırılarak iletim zamanını kısaltabiliriz. Sayısal haberleşme de veriler çok değişik yöntemlerle sıkıştırılabilir. Veri sıkıştırma yöntemlerini kayıplı ve kayıpsız sıkıştırma yöntemleri olarak iki ana başlık altında ele alabiliriz. Bu çalışmada iletilecek verinin kayıpsız şekilde iletilmesi gerekmektedir.

Kayıpsız sıkıştırma yöntemleri, saklanmak istenen verinin ya da iletilmek istenen verinin tekrar açıldığında eski haline yani orijinal haline dönmesini istediğimizde kullanılan yöntemlerdir. Örneğin iletilecek metin tabanlı mesajın karşı tarafta açıldığında aynen okunabilmesi için kayıpsız olarak sıkıştırılmalıdır. Çünkü açılan mesajda karakterlerde eksiklik varsa mesajın okunabilirliğinde ve anlamında hatalar oluşabilir. Özellikle askeri sistemlerde haberleşme sırasında şifreleme teknikleriyle birlikte kayıpsız sıkıştırma yöntemleri kullanılır ki verilen mesaj ya da emir doğru algılansın. Bir ya da birkaç bit sıkıştırma sırasında göz ardı edilirse mesaj karşı tarafta açıldığında orijinal haliyle açılmamış olur ve dolayısıyla istenilen mesaj iletilemediği için büyük problemlere neden olabilir.

Bu çalışmada değişik uzunlukta ki veriler girildikten sonra, girilen verinin nasıl sıkıştırıldığı ve şifrelendiği hakkında bilgi verilmiştir. Sıkıştırılan ve şifrelenen verinin haberleşme sağlanan mikrodenetleyiciye gönderilmesi ve orada gönderilen verinin orijinal haline nasıl getirildiği bahsedilmektedir.

2. VERİ SIKIŞTIRMA TEKNİKLERİ

Dijital verileri çeşitli tekniklerle çeşitli donanım ürünlerinde saklamaktayız. Ve biz bu verileri 1 ve 0' lardan oluşan bitler halinde saklıyoruz. 1 ve 0 diye bahsettiğimiz "var" ve "yok" tan kasıt devreden akım geçmesi ve geçmemesi anlamındadır. Yani belli bir volt değerini geçemeyen gerilimler bilgisayara ulaştığında bu durum "0" değerini teşkil etmektedir. Eşit ve aşan gerilim değerleri ise "1" değerine eşit sayılır. Dijital verilerin tümü işte bu 0 ve 1' ler üzerine kurulmuştur.

Verilerin, saklanması sırasında alandan tasarruf ve iletilmesi sırasında da zamandan tasarruf sağlamak amacıyla çeşitli tekniklerle sıkıştırılması yukarıda bahsedilen 0 ve 1' ler üzerinde yapılan işlemlerle gerçekleşmektedir. Bazı metinleri uygun teknik kullanarak %90' a kadar sıkıştırabiliriz. Böylece saklama belleğinden veya haberleşme kanalı band genişliğinden büyük ölçüde tasarruf sağlanmaktadır.

2.1 Kayıpsız Sıkıştırma

Kayıpsız sıkıştırma yöntemleri, elimizde bulunan verinin, bu metin belgesi, ses, video, fotoğraf olabilir, sıkıştırılıp tekrar açıldığında ilk halinin yani orijinal halinin elde edilmesi için kullanılır. Mesela bir metin belgesi kayıpsız sıkıştırılmalıdır. Çünkü sıkıştırılan veri eski haline geri getirildiğinde metin içinde ki eksik kelime veya karakterler metnin anlam bütünlüğünü bozabilir. Şu şekilde özetlersek, kayıplı sıkıştırmalarda olduğu gibi insan gözü ve kulağının hassasiyetiyle ilgisi olmayan, metin

belgeleri, çalıştırılabilir program dosyaları (.exe), kaynak kodları gibi dosyalar kayıpsız sıkıştırılmak zorundadır.

Uygulamada kayıpsız veri sıkıştırma tekniği kullanılmıştır. Karakter tipli sıkıştırma tekniklerinden Huffman algoritması kullanılmaktadır.

2.1.1 Huffman

Kayıpsız metin sıkıştırma tekniklerinin ilklerinden olan Huffman algoritması, ele alınan metin içinde çok geçen karakterlere kısa kodlar az geçen karakterlere uzun kodlar vererek sıkıştırma işlemini gerçekleştirir. Bir metni veya herhangi bir veri kümesini Huffman tekniği kullanarak sıkıştırabilmek için o ele alınan veri içinde hangi sembolün kaç adet kullanıldığını bilmemiz gerekir. Bunun için her bir sembolün veri kümesi içinde kaç adet kullanıldığını gösteren bir tablo oluşturulur. Bu tablonun ismine de *frekans tablosu* denilmektedir.

Huffman algoritmasını frekans tablosunun önceden oluşturulup oluşturulmamasına göre *statik huffman* ve *dinamik huffman* algoritması olmak üzere ikiye ayırabiliriz. Bilgisayar hafızasında bulunan herhangi bir dosyayı sıkıştırmak için statik huffman algoritmasını kullanırız. Çünkü sıkıştırma yapmadan önce dosya içindeki bütün sembolleri tarayarak dosya içinde hangi sembolün kaç adet bulunduğunu tespit edip frekans tablosunu rahatlıkla oluşturabiliriz. Dinamik huffman tekniğinde ise frekans tablosuna önceden ihtiyaç duymayız. Frekans tablosunu o an gelen sembollerle oluştururuz. Özellikle haberleşme sistemleri gibi önceden ne geleceği belli olmayan sistemlerde dinamik huffman tekniği kullanılır.

İkinci yöntem daha gerçekçi bir çözüm üretmekle beraber metin dosyasının dilinden bağımsız bir çözüm üretmesi ile de ön plandadır. Bu yöntemin dezavantajı ise sıkıştırılan verilerde geçen sembollerin frekansının da bir şekilde saklanma zorunluluğunun olmasıdır [1].

Metin dosyası içindeki sembollere yeni değerler yani kodlar verebilmek için oluşturduğumuz frekans tablosunu kullanarak bir "Huffman Ağacı" oluşturmamız gerekmektedir. Huffman ağacı hangi sembolün hangi kodlarla temsil edeceğini belirlememize yarar.

2.1.1.1 Huffman Ağacının Oluşturulması

Bir huffman ağacı oluşturabilmek için yukarıda da bahsedildiği gibi frekans tablosuna ihtiyaç duyulur. Algan (2004) 'nın makalesinden esinlenerek bir örnekle huffman ağacının nasıl oluşturulduğunu gösterelim. Örnekte kullanılacak frekans tablosu Tablo 1' de verilmiştir.

Tablo 1' den şunu anlamalıyız: Bizim elimizdeki metin dosyasında "e" karakteri 55 kez, "r" karakteri 40 kez, …., "i" karakteri 9 kez kullanılmış. İşte biz bu tabloyu kullanarak hangi karakteri hangi bit dizisiyle temsil edeceğiz onu bulacağız.

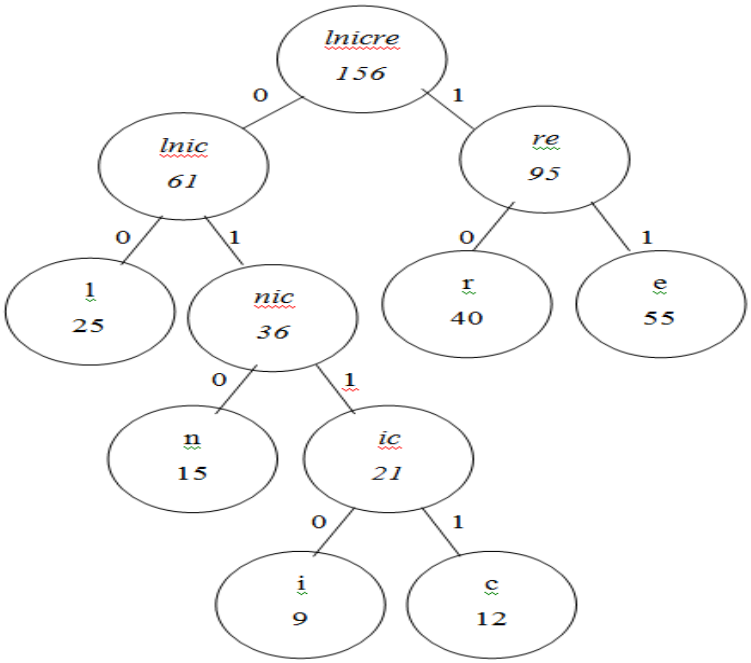
İlk önce bütün semboller frekanslarına göre küçükten büyüğe doğru sıralanır. Şekil 1' de gösterilmektedir.

Ağaç yapısını oluşturmaya başlayalım. Öncelikle en küçük frekansa sahip 2 sembolün frekansları toplanarak yeni bir düğüm oluşturulur. Oluşturulan bu yeni düğüm diğer düğümler arasına küçükten büyüğe doğru sıralanma kuralına uygun olarak yerleşir. Bu şekilde adım adım ağaç yapısı oluşturulur. En son aşağıdaki Şekil 2 deki hale gelir.

Tablo 1. Frekans tablosu

SEMBOL (KARAKTER)	SEMBOL FREKANSI
E	55
R	40
L	25
N	15
C	12
İ	9

Şekil 1. Sembollerin frekanslarına göre sıralanması



Şekil 2. Hufman ağaç yapısı

Kodlarla dalları işaretledikten sonra hangi sembolün hangi bit dizisiyle kodlanacağını bulalım. Dikkat ederseniz metin içinde bulunan bütün semboller dalların ucunda bulunduğu için kökten yaprağa gelene kadar dallardaki kodlar birleştirilip sembollerin kodları oluşturulur. Örneğin en yüksek frekansa sahip "e" sembolünün kodu "11", en düşük frekansa sahip "i" kodunun kodu "0110" olacaktır. Görüldüğü gibi metin içinde daha çok yer alan "e" sembolü daha küçük bit dizisiyle daha az yer alan "i" sembolü daha büyük bit dizisiyle temsil edilmektedir. Tablo 2' de bütün sembollere karşılık gelen kodlar gösterilmektedir.

Tablo 2. Sembollerin Huffman koduyla gösterilmesi

SEMBOL (KARAKTER)	SEMBOL FREKANSI	BİT SAYISI	HUFFMAN KODU
E	55	2	11
R	40	2	10
L	25	2	00
N	15	3	010
C	12	4	0111
İ	9	4	0110

Dikkat ederseniz hiçbir Huffman kodu bir diğer Huffman kodunun ön eki durumunda değildir. Örneğin ön eki "010" olan hiç bir Huffman kodu mevcut değildir. Aynı şekilde ön eki "00" olan hiç bir Huffman kodu yoktur. Bu Huffman kodları ile kodlanmış herhangi bir veri dizisinin *"tek çözülebilir bir kod"* olduğunu göstermektedir. Yani sıkıştırılmış veriden orjinal verinin dışında başka bir veri elde etme ihtimali sıfırdır (Tabi kodlamanın doğru yapıldığını varsayıyoruz) [1].

Hufman algoritmasıyla sembollerin yerine kodlar verdik. Gerçekten sıkıştırma işleminin faydasının olup olmadığını görmek için metnin içinde bulunan bu sembollerin ne oranda sıkıştırıldıklarına bir bakalım.

Sıkıştırma öncesi gereken bit sayısını bulacak olursak: Her bir karakter eşit uzunlukta yani 8 bit ile temsil edildiğinden toplam karakter sayısı olan (55+40+25+15+12+9) = 156 ile 8 sayısını çarpmamız lazım. Orjinal veriyi sıkıştırmadan saklarsak **156*8 = 1248** bit gerekmektedir.

Huffman algoritmasını kullanarak sıkıştırma yaparsak kaç bitlik bilgiye ihtiyaç duyacağımızı hesaplayalım: 55 adet "e" karakteri için 110 bit, 40 adet "r" karakteri için 80 bit, 25 adet "l" karakteri için 50 bit....9 adet "i" karakteri için 36 bite ihtiyaç duyarız (Tablo 2.2'ye bakınız). Sonuç olarak gereken toplam bit sayısı = 55*2 + 40*2 + 25*2 + 15*3 + 12*4 + 9*4 = 110 + 80 + 50 + 45 + 48 + 36 = 369 bit olacaktır.

Sonuç **:** 1248 bitlik ihtiyacımızı 369 bite indirdik. Yani yaklaşık olarak %70 gibi bir sıkıştırma gerçekleştirmiş olduk. Gerçek bir sistemde sembol frekanslarını da saklamak gerektiği için sıkıştırma oranı %70'ten biraz daha az olacaktır. Bu fark genelde sıkıştırılan veriye göre çok küçük olduğu için ihmal edilebilir.

2.1.1.2 Huffman Kodunun Çözülmesi

Örnekte verilen frekans tablosuna sahip bir metin içerisindeki "eeniirrrccile" veri kümesinin sıkıştırılmış hali her karakter ile karakterin kodu yer değiştirilerek aşağıdaki gibi elde edilir.

e e n i i r r r c c i l e

11 11 010 0110 0110 10 10 10 0111 0111 0110 00 11

➔ 11110100110011010101001110111011100011

Eğer elimizde frekans tablosu ve sıkıştırılmış veri dizisi varsa işlemlerin tersini yaparak orjinal veriyi elde edebiliriz. Şöyle ki; sıkıştırılmış verinin ilk biti alınır. Eğer alınan bit bir kod sözcüğüne denk geliyorsa, ilgili kod sözcüğüne denk düşen karakter yerine koyulur, eğer alınan bit bir kod sözcüğü değilse sonraki bit ile birlikte ele alınır ve yeni dizinin bir kod sözcüğü olup olmadığına bakılır. Bu işlem dizinin sonuna kadar yapılır ve huffman kodu çözülür. Huffman kodları tek çözülebilir kod

olduğu için bir kod dizisinden farklı semboller elde etmek olanaksızdır. Yani bir huffman kodu ancak ve ancak bir şekilde çözülebilir. [1] Bu da aslında *kayıpsız sıkıştırmanın* bir sonucudur.

3. KRİPTOLOJİ

Kriptolojiyi iki bölüme ayırabiliriz. Birinci bölüm şifreleme (Kriptografi) ikinci bölüm şifre çözme (kriptanaliz) şeklinde adlandırılır. Haberleşme esnasında gönderilmek istenen mesaja açık mesaj (plain text) ve bu mesajın şifrelenmiş hali şifreli mesaj (cipher text-cryptograph) olarak adlandırılır [2].

3.1 Kriptografi (Cryptography)

Bir bilginin istenmeyen kişiler tarafından ele geçirilmesini engellemek amacıyla anlaşılmayacak hale getirilmesi için kullanılan tekniklerin tümüne kriptografi denir. Kriptografi gizlilik, bütünlük, kimlik denetimi, inkar edememe gibi bilgi güvenliği kavramlarını sağlamak için çalışan matematiksel yöntemleri içermektedir [3].

3.2 Kriptanaliz (Cryptanalysis)

Şifrelenmiş verinin tekrar orijinal haline getirilmesi için kullanılan yöntemlerin tümüne kriptanaliz denir. Şifreleme ve şifre çözme işlemlerinde verinin iletim sırasında güvenliğini daha da artırmak için anahtarlar kullanılmaktadır. Yani bir verinin güvenliği sadece oluşturulan şifreleme algoritması değil anahtara ve bu anahtarın uzunluğuna da bağlıdır [2].

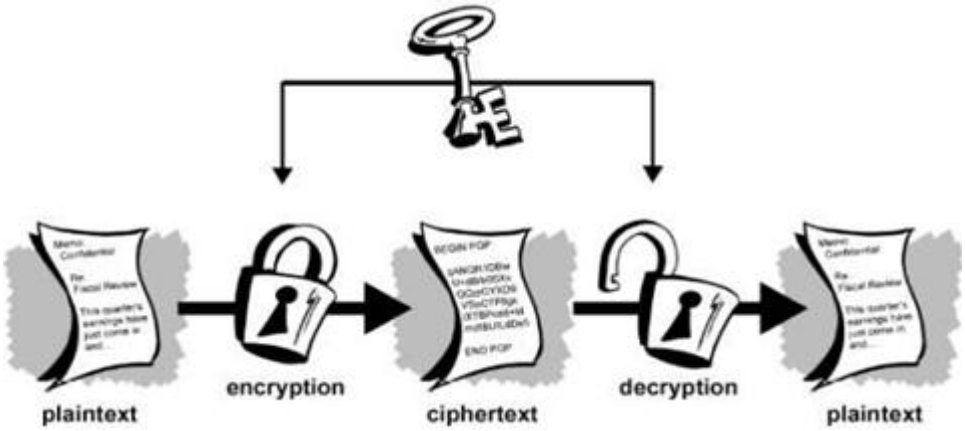
3.3 Anahtar (Key)

Anahtarlar sayılardan oluşur. Anahtarın uzunluğu verinin güvenliği ile doğru orantılıdır. Ancak anahtar çok büyük olursa iletim çok yavaş olur. O yüzden şifrelenecek veri iletilmeyecek sadece depolanacak ise çok büyük anahtarla avantaj sağlar. Eğer bu anahtarlar haberleşmede kullanılacak ise anahtar boyu daha kısa seçilmelidir.

Şifreleme ve şifre çözme için kullanılan anahtarın gizliliğine ve sayısına göre şifreleme sistemlerini iki ana başlık altında toplayabiliriz.

3.3.1 Simetrik Anahtarlı Sistemler

Simetrik anahtarlı sistemlerde gönderici ve alıcılarda aynı gizli anahtar bulunur. Yani bilgiyi şifrelemek için ve şifrelenmiş bilgiyi tekrar eski haline getirmek için aynı anahtar kullanılır. Uygulamada alıcının birden fazla olabileceği düşünülerek ve hızlı iletişim göz önünde bulundurularak şifreleme işleminde bu sistem tercih edilmiştir. Şekil 3' de simetrik anahtarlı sistemin genel yapısı gösterilmektedir [4].



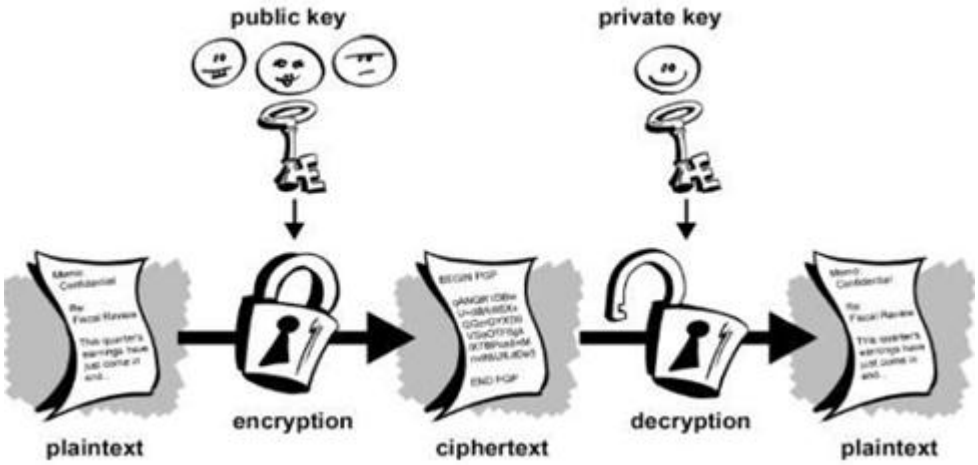
Şekil 3. Simetrik Anahtarlı Sistem

3.3.2 Asimetrik Anahtarlı Sistemler

Bu sistemde iki farklı anahtar kullanılır. Biri herkes tarafından görülebilen genel anahtar (public key) diğeri ise gizli anahtar' (private key) dır. Asimetrik anahtarlı sistemlerin genel yapısı Şekil 4' te görülmektedir. Bununla birlikte bu anahtarlardan herhangi birine sahip olan bir şahıs, diğer anahtarı üretemez, bu matematiksel olarak imkansız denebilecek derecede zordur [4].

Asimetrik algoritmalar, simetrik algoritmalara göre daha güvenli ve kırılması zor algoritmalardır. Bununla birlikte, başarımları (performans) simetrik algoritmalara göre oldukça düşüktür. Asimetrik algoritmalarda her şahsın bir anahtar çifti vardır. Bir şahsın özel anahtarı, yalnızca kendi kullanımı içindir ve başkalarının eline geçmemesi gerekir. Bu şahsın açık anahtarı ise, bu şahsa mesaj göndermek isteyen herhangi biri tarafından kullanılabilir. Gönderici mesajı, alıcının açık anahtarı ile şifreler. Alıcı, gelen mesajı kendi özel anahtarı ile açar [4].

Özel anahtarınızı gizli tutarak açık anahtarınızı tüm dünyaya yayabilirsiniz. Açık anahtara sahip bir kişi bilgiyi sadece şifreleyebilir fakat çözemez. Yalnızca özel anahtara sahip olan kişi bilgiyi okuyabilir [4].



Şekil 0. Asimetrik Anahtarlı Sistemler

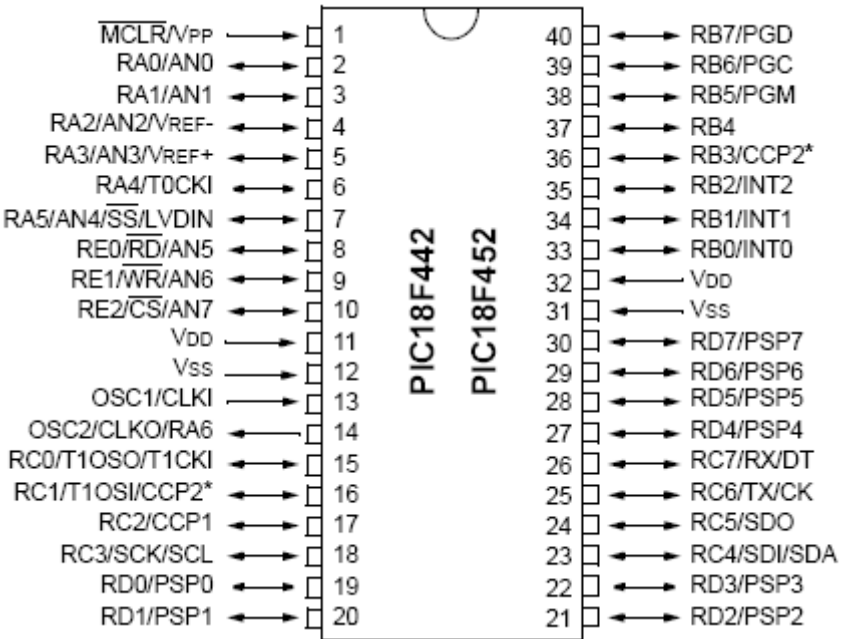
4. MİKRODENETLEYİCİLER

Bir bilgisayar içerisinde bulunması gereken temel bileşenlerden RAM, I/O ünitesinin tek bir çip içerisinde üretilmiş biçimine mikrodenetleyici (mikrokontrolör) denir. Bilgisayar teknolojisi gerektiren uygulamalarda kullanılmak üzere tasarlanmış olan mikrodenetleyiciler, mikroislemcilere göre çok daha basit ve ucuzdur. Günümüz mikrodenetleyicileri pek çok alanda kullanılmaktadır. Neredeyse her mikroişlemci üreticisinin ürettiği birkaç mikrodenetleyicisi bulunmaktadır. Bir uygulamaya başlamadan önce hangi özelliklere sahip mikrodenetleyicinin kullanılacağı önemlidir [5].

PIC 16F877 piyasada en çok kullanılan üzerinde çok çalışılmış ve dolayısıyla kaynağı çok olan bir mikrodenetleyicidir. Uygulamada ilk önce bu mikrodenetleyici kullanılmak istenmiş ancak hafızası yeterli gelmediği için aynı bacak yapısına sahip ve aynı özelliklere sahip daha üstün özellikleri de olan PIC 18F452 kullanılmıştır.

4.1 Pic 18f452 Mikrodenetleyici

PIC18F452 mikrodenetleyicisi 40 pinli bir mikrodenetleyicidir. Giriş çıkış olarak kullanılabilecek 33 adet I/O pini mevcuttur. Bu pinlerden 8 tanesi, 10 bitlik ADC (analogdigital dönüstürücü) pinidir [24,25]. PIC18F452 mikrodenetleyicisi, Amerikan Microchip firmasının üretmiş olduğu 8 bitlik CMOS FLASH yapısında bir mikrodenetleyicidir [6].



Şekil 0. PIC18F452 mikrodenetleyicinin pin diyagramı görünüşü

5. PROTEUS İLE DEVRE ŞEMASININ OLUŞTURULMASI

Mikrodenetleyiciyi programlayabilmek için CCS C programı kullanılmıştır. Kodların oluşturulabilmesi için öncelikle mikrodenetleyicilerin haberleşmesini sağlayacak elektronik devrenin kurulması gerekmektedir. Bu devrenin bilgisayar ortamında sanal olarak kurulabilmesini sağlayan Proteus programına ihtiyaç duyulmaktadır.

Labcenter Electronic firmasının bir ürünü olan Proteus görsel olarak elektronik devrelerin simülasyonunu yapabilen yetenekli bir devre çizimi, simülasyonu, animasyonu ve PCB çizimi programıdır. Klasik workbench' lerden en önemli farkı mikroişlemcilere yüklenen .HEX dosyalarını da çalıştırabilmesidir. Proteus gün geçtikçe genişleyen bir model kütüphanesine sahiptir. Proteus programı sanal bir laboratuvardır. Her türlü elektrik/elektronik devre şemasını Proteus yardımıyla bilgisayar ortamında deneyebilirsiniz. Devredeki elemanların değerlerini değiştirip yeniden çalıştırır ve sonucu gözlemleyebilirsiniz. Bu program, binlerce elektronik eleman içeren devre tasarımlarının üretiminde bile kullanılabilmektedir. Elektriksel hata raporu hazırlayabilmekte, malzeme listesini çok düzenli bir şekilde verebilmektedir [7].

5.1 Ccs C İle Pic18f452 Mikrodenetleyicilerin Programlanması

Proteus ISIS programıyla oluşturulan devrenin istediğimiz biçimde çalışabilmesini sağlamak için PIC18F452 mikrodenetleyicilerin programlanması gerekmektedir. Bunun için de Ccs C derleyicisi kullanılmıştır.

Öncelikle kodlamayı yaparken PIC18f452' nin bacaklarına bağlanan elemanlara göre pinlerin ilk durumları (1) ve (2) numaralı kodlarla belirlendi;

set_tris_b(0x00) (1)

set_tris_d(0x0F) (2)

set_tris_b() : B portu komple çıkış
set_tris_d() : Yüksek değerlikli 4 bit çıkış, düşük değerlikli 4 bit giriş

B portuna 2x16 Karakter Tabanlı LCD paneli bağlı ve görüntünlemek istenen veriler bu LCD panelde görüntüleneceği için *set_tris_b(0x00)* komutu kullanılır. Bu komut B portunu komple çıkış yapmaktadır. D portuna 16 butonla oluşturulan bir keypad bağlıdır. Bu butonlara basıldığında değerleri okuyabilmek için ilk durumda *set_tris_d(0x0F)* komutu kullanılır. Set_tris_x komutu port pinlerinin hangisinin giriş pini, hangisinin çıkış pini olacağını belirtir [8]. *set_tris_d(0x0F)* komutu ile yüksek değerlikli 4 bit çıkış, düşük değerlikli 4 bit giriş olarak kullanılacağını bildirir.

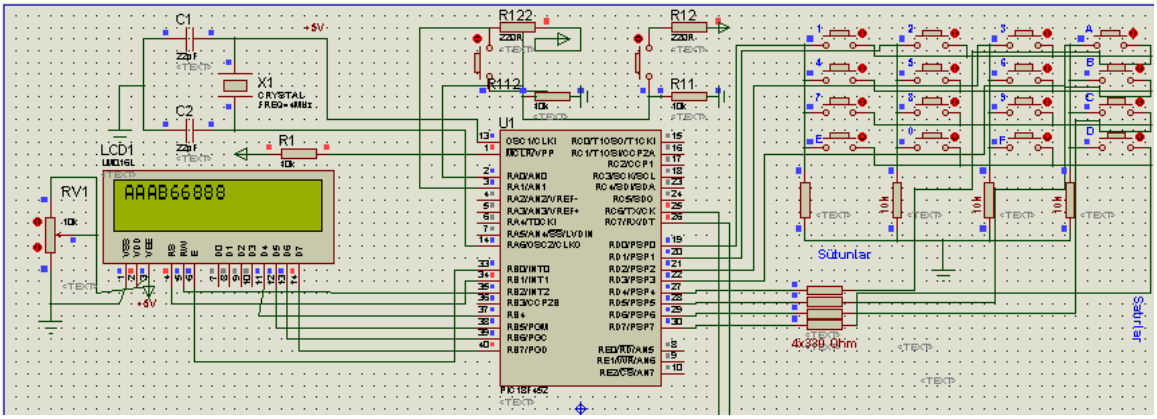
Tasarlanan devrede ilk durumda D portuna bağlı butonların hiç birisi basılı halde değil. Bu durumda D portu çıkışı *output_d(0x00)* komutuyla sıfırlanması gerekmektedir. Herhangi bir tuşa basıldığında hangi tuşa basıldığını anlamak için 4x4 şeklinde satır, sütun yapısı oluşturan butonlar sırasıyla kontrol edilir. Bunu yapabilmek için sırasıyla öncelikle satırları yani düşük değerlikli bitleri output_high(pin_dx) ile lojik-1 yapıyoruz. Devamında lojik-1 yaptığımız satırdaki sütunlara basılıp basılmadığı yani input(pin_dx) ile kontrol edilir. Eğer koşul doğruysa yani true döndüyse o satır ve sütuna karşılık gelen buton değeri belirlediğimiz bir değişkene aktarılır. Uygulamada birden fazla butonlara basılacağı için "keypad_oku" isminde bir fonksiyon oluşturulmuştur. Ve her butona basıldığında bu fonksiyon çağrılıp basılan tuşun değeri değişkene aktarılıp return deyimiyle ana fonksiyona tekrar geri döndürülmektedir. Ana fonksiyonda da *write_eeprom(q,tus)* komutuyla daha önceden basılan tuşların değerlerinin kaybolmaması için eeproma değerler yazılır.

Basılan her buton lcd ekranında gösterilmek istendiğinde lcd kodlarının program tarafından anlaşılır olabilmesi için *#include <lcd.c>* komutu eklenmektedir. *printf(lcd_putc,"%c",keypad_oku())* komutu ile keypad_oku() fonksiyonundan dönen buton değeri lcd ekranına yazdırılır.

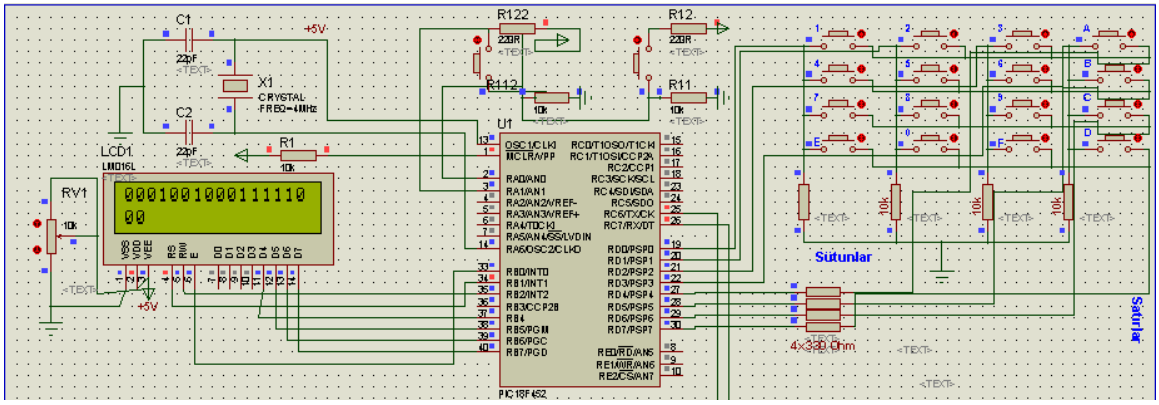
Girilen komutlar bittikten sonra RA0 pinine bağlı butona basılır. Bu butona basıldıktan sonra ilk önce hafızaya kaydedilmiş karakterler sırasıyla lcd ekranına yazdırılır. Daha sonra girilen toplam karakter sayısı, girilen farklı karakter sayısı, karakterlerin kaç adet girildiğini, huffman algoritmasıyla oluşturulan ağaç yapısıyla karakterlerin yerine kullanılacak '0' ve/veya '1' lerin lcd ekranına

yazdırılır. En son olarak, girilen bütün karakterlerin yerine her bir karakter yerine kullanılacak '0' ve '1' ler yazılır. Bu veriyle program içinde belli bir algoritmayla oluşturulan anahtar XOR şifreleme yöntemiyle şifrelenerek, şifrelenmiş veri lcd ekranına yazdırılır.

Şekil 6.' da girilen karakterler, Şekil 7.' de girilen karakterlerin Huffman algoritmasıyla sıkıştırılmış veri ile belli bir algoritmayla oluşturulan anahtarın XOR ile şifrelenmiş halinin ekrana yazdırılması görülmektedir.

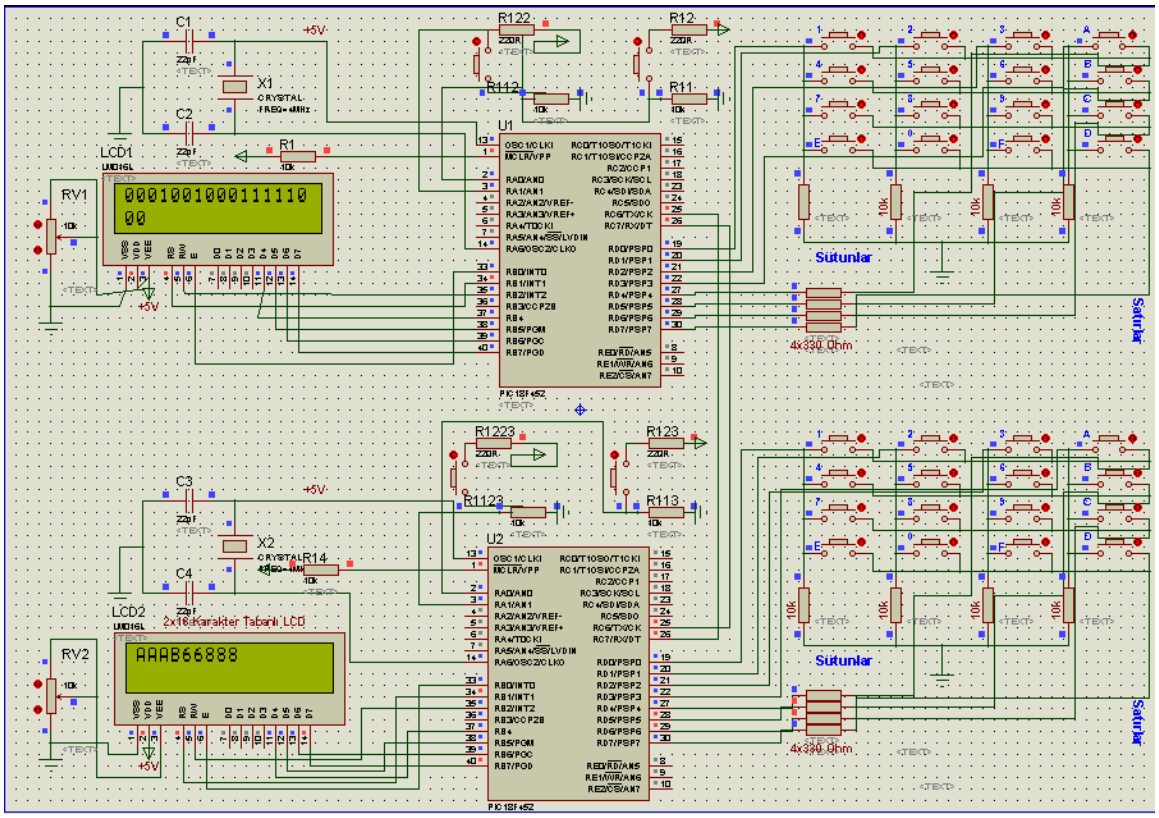


Şekil 0. Tuş Takımından Girilen Karakterlerin Ekrana Yazdırılması



Şekil 7. Tuş Takımından Girilen Karakterlerin Huffman Algoritmasıyla Sıkıştırılmış Veri ile Anahtarın XOR ile Şifrelenmiş Halinin Ekrana Yazdırılması

Şekil 8.' de bir mikrodenetleyiciden girilen karakterlerin sıkıştırma ve şifreleme işlemlerinden geçtikten sonra diğer mikrodenetleyiciye gönderilip, gönderilen mikrodenetleyici de verinin orijinale dönüştürülmüş hali görülmektedir. Bu nasıl mümkün olmaktadır. Verinin orijinal hali gönderilmediği halde o hale nasıl çevrilmektedir. Şekil 8.' de üstteki lcd ekranında verinin sıkıştırılmış ve şifrelenmiş hali alta ki lcd ekranında da verinin orijinal hali görülmektedir.



Şekil 8. Şifreli Şekilde Gönderilen Verinin Orijinal Haline Çevrilmesi

6. SONUÇLAR

Bu çalışmada, haberleşme sırasında gönderilmek istenen verinin istenmeyen kişilerin eline geçmesini engellemek amacıyla, verinin güvenli ve hızlı bir şekilde alıcıya gitmesi problemi ele alınmıştır. Bu problemin çözümü için sıkıştırma ve şifreleme teknikleri ele alınmıştır. Sıkıştırma tekniklerinden dinamik Huffman algoritması verinin hızlı bir şekilde iletilmesini sağlamak için seçilmiştir. Uygulama da girilen karakterlere göre her defasında o karakterlere farklı bir değer ataması sistemin güvenliği açısından da önem taşıdığı için Huffman algoritması seçilmiştir. Şifreleme işlemi için simetrik veya asimetrik şifreleme sistemlerinden birini kullandığımızda, mikrodenetleyicinin yavaş çalışmasına neden olacağı için AES içinde bulunan XOR ve anahtar yöntemi kullanılması uygun görülmüştür. Bu tezin amacına hızlı ve güvenli iletişim olduğu için mikrodenetleyicinin hızını yavaşlatacak bir şifreleme sistemi kullanılamaz.

Günümüz teknolojisinde güvenliğin ne kadar önemli olduğu, özellikle askeri haberleşme sistemlerinde, kişisel veri güvenliğinde, internet üzerinden dosya paylaşımında aşikardır.

Çalışmanın sonunda görüldü ki haberleşme sırasında orijinal veri alıcı tarafa gönderilmeden, alıcı taraf orijinal veriyi kriptanaliz yöntemiyle hızlı ve güvenli bir şekilde elde etti.

Bu çalışmadaki yenilik, PIC18F452 mikrodenetleyiciler arasında ki haberleşmeyi gerçekleştirirken sıkıştırma ve şifreleme tekniğini bir arada kullanılmasıdır.

7. ÖNERİLER

Burada ki amaç sadece iki mikrodenetleyiciyi haberleştirmek değildir. Asıl amaç askeri sistemler için düşünülen sürü robot mantığını gerçekleştirebilmektir. Bunun için yapılması gereken haberleşmenin koordinasyonunu sağlayacak lider bir robot yaratmaktır. Robotların birbiriyle haberleşmesi bu robot üzerinden sağlanacaktır. Haberleşmek isteyen robotun hangi robot olduğunu anlayabilmek için lider robotta diğer robotları tanıyacak kod bulunacaktır. Haberleşmek isteyen robot iletmek istediği veriye kendi kodunu ve kiminle haberleşmek istiyorsa onun kodunu ekleyecektir. Böylelikle bilginin hangi robottan geldiği ve nereye gideceği konusunda hata oluşmayacaktır.

Bu robotlar haberleşirken, bu tezde kullanılan sıkıştırma ve şifreleme algoritmalarıyla, veriler hızlı ve güvenli bir şekilde iletilecektir.

8. KAYNAKÇA

[1] http://www.csharpnedir.com/makalegoster.asp?MId=189 (Erişim tarihi:20.05.2012)

[2] YERLİKAYA T., Yeni Şifreleme Algoritmalarının Analizi, Doktora Tezi, Trakya Üniversitesi, Fen Bilimleri Enstitüsü, Edirne (2006)

[3] AKYILDIZ, E., ODTÜ' de Kriptoloji Konusunda Yapılan Araştırmalar ve Uygulamalar, Ulusal e-devlet konferansı, Ankara (4-5 Kasım 2008)

[4] YILDIRIM K., Veri Şifrelemesinde Simetrik Ve Asimetrik Anahtarlama Algoritmalarının Uygulanması (Hybrid Şifreleme), 8-13, Yüksek Lisans Tezi, Kocaeli Üniversitesi, Fen Bilimleri Üniversitesi, Kocaeli, (2006),

[5] ALTINBASAK, O., **Mikrodenetleyiciler ve PIC programlama**, Altaş Kitapevi, (2001)

[6] KELEŞ,F., Mikrodenetleyici Kontrollü Redresör Tasarım Ve Gerçeklenmesi,32,33 Yüksek Lisans Tezi, Dumlupınar Üniversitesi, Fen Bilimleri Enstitüsü, Kütahya (Subat-2006)

[7] http://tr.wikipedia.org/wiki/Proteus (Erişim tarihi:22.05.2012)

[8] ÇİÇEK, S., **CCS C ile Pic Programlama**, 170,432-437, Atlas Kitapevi, (2009)