T.C.
BALIKESİR ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK - ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

ROBOTLAR ARASINDA HIZLI VE GÜVENLİ HABERLEŞMENİN SAĞLANMASI

YÜKSEK LİSANS TEZİ

İbrahim ÖZCAN

Balıkesir, EYLÜL–2010

T.C.
BALIKESİR ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK - ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

ROBOTLAR ARASINDA HIZLI VE GÜVENLİ HABERLEŞMENİN SAĞLANMASI

YÜKSEK LİSANS TEZİ

İbrahim ÖZCAN

Tez Danışmanı: Yrd. Doç. Dr. Davut AKDAŞ

Sınav Tarihi: 17.09.2010

Jüri Üyeleri: Yrd. Doç. Dr. Mehmet TERZİ (BAÜ)

Yrd. Doç. Dr. Bayram ESEN (BAÜ)

Yrd. Doç. Dr.Davut AKDAŞ (Danışman-BAÜ)

Balıkesir, EYLÜL–2010

ÖZET

ROBOTLAR ARASINDA HIZLI VE GÜVENLİ HABERLEŞMENİN SAĞLANMASI

İbrahim ÖZCAN
Balıkesir Üniversitesi, Fen Bilimleri Enstitüsü,
Elektrik - Elektronik Mühendisliği Anabilim Dalı

(Yüksek Lisans Tezi / Tez Danışmanı: Yard. Doç. Dr. Davut AKDAŞ)

Balıkesir, Türkiye, 2010

Askeri sistemlerde haberleşme için gerekli olan niteliklerden biri de iletişimin hızlı ve güvenli sağlanabilmesidir. Bilginin güvenliği açısından haberleşme sırasında bilgiye ulaşmak için oluşabilecek saldırılarda saldırganın gerçek bilgiye ulaşmasını engellemek için verinin hızlı ve şifrelenmiş bir şekilde iletilmesi gerekmektedir.

Gerçekleştirilen çalışmada 4x4' lük tuş takımından, 16 tabanlı sayı sistemindeki sayılarla 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 getirilip 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

PROVIDING A FAST AND SECURE COMMUNICATION AMONG ROBOTS

İbrahim ÖZCAN
Balıkesir University, Institute of Science,
Department of Electric - Electronic Engineering

(Master Thesis / Supervisor: Assistant Prof. Dr. Davut AKDAŞ)

Balıkesir-Turkey, 2010

Another qualification that is required for communication in military systems is that the communication is provided fast and secure. In terms of information security, during communication, in attacks aimed at reaching information, the data must be delivered both fast and encoded so as to prevent the attacker to reach the information.

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 *Dynamic 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

İÇİNDEKİLER

Sayfa

KISALTMALAR LİSTESİ

Adı	Tanımı
LCD	Likit Kristal Ekran
LXOR	Bitsel Özel Veya
BMP	Windows işletim sisteminde kullanılan standart resim dosyası
JPEG	Kayıplı resim sıkıştırma formatı
RLE	Ardışıl karakter sıkıştırma
LZ77	Metin tabanlı veri sıkıştırma
LZ78	Metin tabanlı veri sıkıştırma
SSDC	Semi-Static Diagram Coding (Çift geçişli yarı-statik algoritma)
ISSDC	Iterativ Semi-Static Diagram Coding (Tekrarlama sayısı kadar geçişli yarı statik algoritma)
FPGA-VHDL	Donanımsal şifreleme modülü
CFB	Şifre geribesleme metodu
AES	Advanced Encryption Standart (Gelişmiş şifreleme standardı)
DES	Data Encryption Stantard (Veri şifreleme standardı)
KEM	Key Encapsulation Mechanism (Anahtar kapsülleme mekanizması)
DEM	Data Encapsulation Mechanism (Veri kapsülleme mekanizması)
Mp3	Mpeg-1 Layer Audio Layer III (sıkıştırılmış ses biçimi)
Hz	Hertz (Frekans Birimi) saniye başına düşen devir sayısı
KHz	Kilo Hertz. 1000/s anlaminda frekans birimi
Mb	MegaBayt.1.048.576 bayt anlamına gelen bir ölçü birimi
Wma	Windows Media Audio (Windows medya ses dosyası)
Aac	Advanced Audio Coding (Gelişmiş ses kodlama)
Gif	Graphics Interchange Format (Grafik değiştirme biçimi)
Tiff	Tagged Image File Format (Sıkıştırma yapmadan çekim imkanı sağlayan bir resim formatı)
PNG	Portable Network Graphics (Taşınabilir Ağ Grafiği)
GIMP	Image Manipulation Program (Görüntü işleme programı)
MPEG	Moving Pictures Experts Group (Görüntü sıkıştırma formatı)
DVD	Digital Versatile Disc (Çok yönlü sayısal disk)
HDTV	High Definition TeleVision (Yüksek çözünürlü televizyon)
exe	Çalıştırılabilir program dosyaları
Dpcm	Differential Pulse Code Modulation (Fark darbe kod modülasyonu)
Adpcm	Adaptive Differential Pulse Code Modulation (Uyarlanabilir diferansiyel darbe kodu modülasyonu)
Mpeg-4 Als	Audio Lossless Coding (Kayıpsız ses kodlama)
aiff	Unix tarafından kullanılan sıkıştırılmamış ses dosya formatı
au	Sun' ın sıkıştırılmamış ses dosya formatı
bwf	ebu tarafından sıkıştırılmamış ses dosya formatı
raw	Sensörden gelen sayısal verilen doğrudan belleğe yazılmasıyla oluşan özel format
Blowfish	anahtarlanmış, simetrik bir Block Cipher (öbek şifreleyici)
ECC	Elliptic Curve Cryptography (Düşük anahtar uzunluklu şifreleme)
Diffie-Hellman	Açık anahtar kripto sistemleri
SSP	Senkron seri port
USART	Donanımsal adreslenebilir asenkron seri alıcı verici

Adı	Tanımı
ADC	Analog dijital çevirici
ICSP	Devre üzerinde seri olarak programlanabilme
WDT	Watchdog Timer (Mikrodenetleyici sistemi resetleme birimi)
RTF	Zengin Metin Dosyası
RS232	Seri İletişim
UART	RS232 haberleşme donanım modulü
DVT	Discrete Wavalet Transform (Ayrık Dalgacık Dönüşümü)
DCT	Discrete Cosine Conversion (Ayrık Kosinüs Dönüşümü)

ŞEKİL LİSTESİ

Şekil Adı **Sayfa**

ÇİZELGE LİSTESİ

Çizelge

Numarası Adı Sayfa

ÖNSÖZ

Çalışmamın en başından en sonuna kadar beni yönlendiren değerli danışmanım Yrd. Doç. Dr. Davut AKDAŞ‘ a yüksek lisans çalışması sırasında karşılaştığım problemlerin çözümünde yardımcı olduğu için teşekkür ederim. Ayrıca işlerinin yoğunluğu arasında bana yardımcı olan değerli arkadaşım Arş. Gör. Sabri BIÇAKÇI’ ya teşekkür ederim.

Kaynak konusunda ve tez yazım sırasında desteğini benden esirgemeyen değerli meslektaşım Öğr. Gör. Ersin AKYÜZ’ e teşekkür ederim.

Çalışmamın en başından beri, en sıkıntılı zamanlarımda bana destek olan, kendi kıymetli zamanlarınını bile benim için harcayan değerli eşim Hülya ÖZCAN’ a teşekkürü bir borç bilirim. Onun gibi bir eşe sahip olduğum için gerçekten çok şanslıyım.

Balıkesir, 2010 İbrahim ÖZCAN

1. GİRİŞ

Haberleşme yöntemlerinden sayısal haberleşme günümüz teknolojisinde en çok tercih edilen yöntemlerden biridir. İnternet teknolojisinin gelişmesiyle birlikte haberleşmenin internet üzerinden yapıldığı günümüzde, haberleşme sırasında iletilecek verilerin hızlı bir şekilde iletilmesi bir amaç haline gelmiştir. Çeşitli tekniklerle iletilecek veriler sıkıştırılarak iletim zamanını kısaltabilir. Sayısal haberleşme de veriler çok değişik yöntemlerle sıkıştırılabilir. Veri sıkıştırma yöntemleri kayıplı ve kayıpsız sıkıştırma yöntemleri olarak ele alınabilir.

Kayıplı sıkıştırma yöntemleri, özellikle multimedya verilerinde kullanılmaktadır. Buna örnek olarak son 10 yıldır çok moda olan bir mpeg standardı olan mp3 formatı verilebilir. Bu yöntemlerde sıkıştırılmış veri geri açılmak istenildiğinde fark edilemeyecek kadar kayıp vardır. Mesela mp3 formatında insan kulağının duyamayacağı 20 Hz altı ve 20 Khz üzeri sesler kayda alınmaz. Bu sesleri duymadığımıza göre kayıt etmenin gereği yoktur. Aynı şekilde kayıplı veri sıkıştırma yöntemlerine jpeg standardı verilebilir. Bu yöntemde renk tonları arasındaki gözün göremeyeceği ayrıntılar silinir. Mesela siyah bir zemindeki siyah-gri arası tonlar silinir, bu ayrıntı resme ancak çok yakından bakıldığına seçilebilir. Örneğin 10 Mb lık yer kaplayan *bmp* uzantılı bir resmi *jpeg* uzantılı hale dönüştürdüğümüzde yaklaşık 2,5 Mb lık bir yer kaplar. Böylelikle haberleşme sırasında gönderilecek olan resim için 4 kat bir zaman kazanılmış olur.

Kayıpsız sıkıştırma yöntemleri, saklanmak istenen verinin ya da iletilmek istenen verinin tekrar açıldığında eski haline yani orijinal haline dönmesi istenildiğinde 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.

Bilginin gizlenmesi ve güvenilir hale getirilmesi ile uğraşan bilime kriptoloji (şifrebilimi) denir [1]. Kriptoloji, Yunanca krypto's (saklı) ve lo'gos (kelime) kelimelerinin birleştirilmesinden oluşturulmuştur ve iletişimde gizlilik bilimi olarak değerlendirilmektedir [2]. Tarih boyunca bilginin istenmeyen kişilerin eline geçmesini engellemek için çeşitli şifreleme yöntemleri kullanılmıştır. Sistemler arası bağlantılarda ya da herhangi iki nokta arasındaki haberleşmede verinin güvenli bir şekilde gittiğinden emin olmak gerekir. Bunun sağlanması ise gönderilen verinin şifrelenmesi ile olur. Böylece açık haberleşme kanalları kullanılarak verinin güvenli bir şekilde ulaştırılması sağlanır. İletişimde, açık bir haberleşme kanalı kullanılıyorsa gizli tutulmak istenen bilginin yetkisiz bir kişi tarafından dinlenebileceği veya haberleşme kanalına girip (araya girme) veriyi bozabileceği ya da değiştirebileceği (yanlış verinin gönderilmesi) düşüncesi her zaman için önemli bir problem oluşturur [3].

Kriptoloji esas olarak iki bölüme ayrılır; kriptografi (şifreleme) ve kriptanaliz (şifre çözme). Kriptografi, bir bilginin istenmeyen taraflarca anlaşılmayacak bir hale dönüştürülmesinde kullanılan tekniklerin bütünüdür. Kriptografi gizlilik, bütünlük, kimlik denetimi, inkar edememe gibi bilgi güvenliği kavramlarını sağlamak için çalışan matematiksel yöntemleri içermektedir. Kriptanaliz, şifrelenmiş, yani anlamsız bir metinden doğru metni bulma tekniklerinin tümüdür [4].

Devlet işleri olsun, askeri işler olsun güvenliğin ön planda olduğu kurumlar da haberleşme için verinin gizliliğinin korunması çok önemlidir. Tezin dördüncü bölümünde haberleşme sırasında verinin güvenli bir şekilde

iletilmesini sağlayan kriptoloji yapısından ve kullanılan tekniklerden de bahsedilecektir.

Tezin beşinci bölümünde tasarlanan sistem ve elde ettiğimiz sonuçlar hakkında bilgi verilmektedir. Değişik uzunlukta ki veriler girildikten sonra, girilen verinin nasıl sıkıştırıldığı ve şifrelendiği hakkında bilgi verilmiştir. Sıkıştırılan ve şifrelenen verinin haberleşilen mikrodenetleyiciye gönderilmesi ve orada gönderilen verinin orijinal haline nasıl getirildiği detaylı bir şekilde anlatılmıştır. Son bölümde elde edilen sonuçlar ve önerilerden bahsedilmiştir.

2. LİTERATÜR

Literatür taraması yaparken metin sıkıştırma teknikleri ile birlikte sıkıştırma oranlarının en verimli hale getirilmesi konusunda çeşitli çalışmasalar yer almaktadır. Aynı şekilde şifreleme teknikleri ile ilgili güvenlik ve hız göz önünde bulundurularak oluşturulmuş tekniklerle ilgili çalışmalar yer almaktadır. Bu bölümde sıkıştırma ve şifleleme ile ilgili araştırmalardan bilgiler sunulacaktır.

Çölkesen (2005), kitabında sayısal ortamda saklanan her tür veri, en azından bir tür fazlalık içerdiğinden bahsetmiştir. Bu fazlalıkları, yöreselliğe dayanılarak alanda, sıklıkta ve zamanda olarak 3 gruba ayırmıştır. Huffman kodlamasının sıklıkta yöreselliğe dayanan fazlalığı azalttığını öngörmüştür. Alanda yöreselliğe dayanan fazlalığı azaltan teknikler boşluk sıkıştırma, ardışıl karakter sıkıştırma (RLE), yarım sekizli paketleme, karakter yapıştırma, desen tanıma ve bağıl kodlama olarak belirtmiştir. Zamanda yöreselliğe dayanan fazlalığı azaltan tekniklerin neredeyse tamamının Lempel ve Ziv tarafından önerilen iki algoritmaya, LZ77 ve LZ78' e ve onların türevlerine dayandığını öngörmüştür. Bir veri üzerinde aynı türden tekniklerin zincirleme bir şekilde kullanılmasıyla toplam başarımın artmayacağını aksine çoğu durumda azalabileceğini belirtmiştir. Ancak farklı türden teknikler zincirleme şeklinde kullanılırsa sıkıştırma başarımının artacağını belirtmiştir [5].

Mesut (2006), "Veri Sıkıştırmada Yeni Yöntemler" isimli doktora tezinde, eskiden geliştirilmiş veri sıkıştırma yöntemleriyle yakın zamanda geliştirilmiş yöntemler arasındaki farkları incelemiştir. Yeni yöntemlerin kendinden önceki yöntemleri ne yönde geliştirdiğine dair araştırmalar yapmıştır. Ayrıca var olan sözlük tabanlı yöntemlere alternatif olabilecek SSDC ve ISSDC isminde yeni yöntemler geliştirmiş ve bu yöntemlerle diğer veri sıkıştırma yöntemlerini örnek verilerle karşılaştırmıştır [6].

Algan (2004), “Huffman Veri Sıkıştırma Algoritması ve Uygulaması” isimli makalesinde, Huffman veri sıkıştırma hakkında örnek bir ağaç yapısı oluşturmuş ve aynı zamanda bir uygulama geliştirmiştir [7].

Öcal (2006), “Güvenli İletişim İçin FPGA Kullanarak Şifreleme Sistemi Tasarımı ve Gerçekleştirilmesi” isimli yüksek lisans tezinde, haberleşme sırasında verinin daha hızlı ve daha güvenli iletilebilmesi için şifreleme tekniği olarak donanım tasarımını seçmiştir. FPGA – VHDL elemanında uygulamasını gerçekleştirmiştir. İleri şifreleme standardı (AES) tercih edilmiş, şifreleme metodu olarak “şifre geribesleme metodu (Cipher Feedback, CFB)” ve donanım elemanı olarak XILINK’ in ürünü SPARTAN kullanılmış. FPGA ile gerçekleştirdiği donanım uygulamasında en hızlı yazılım uygulamasından 13,9 kat daha hızlı olduğunu bulmuştur [8].

Yerlikaya (2006), “Yeni Şifreleme Algoritmalarının Analizi” isimli doktora tezinde, günümüzde yaygın olarak kullanılan simetrik ve asimetrik şifreleme algoritmalarını incelemiş. Simetrik algoritmalardan AES ve DES üzeirnde durmuş, asimetrik algoritmalardan RSA ve Eliptik eğri şifreleme algoritmalarının uygulamasını gerçekleştirmiştir. Eliptik eğri şifreleme algoritmasının daha düşük anahtar değeriyle RSA ile aynı güvenliği sağladığını belirlemiştir [9].

Buluş (2006), “Temel Şifreleme Algoritmaları ve Kriptanalizlerinin İncelenmesi” isimli yüksek lisans tezinde, temel şifreleme algoritmaları ve kriptanalizlerini incelemiştir. Türk alfabesindeki şifrelemeden sonra kriptanaliz sırasında oluşan hataları, algoritmaları Türk alfabesine uyarlayarak gidermiştir [1].

Yıldırım (2006), “Veri Şifrelemesinde Simetrik ve Asimetrik Anahtarlama Algoritmalarının Uygulanması (Hybrid Şİfreleme)” isimli yüksek lisans tezinde, simetrik ve asimetrik algoritmaların her ikisinin avantajlı yanlarını birleştiren Hybrid şifreleme algoritmasını kullanmıştır. Hybrid

şifreleme de kullanılan yöntemlerden klasik key encapsulation mechanism (KEM) ve data encapsulation mechanism (DEM) yerine, ki bunların herhangi birinde zayıflık olduğunda kolayca şifrenin kırılabileceği belirtilmiş, karıştırıcı algoritmayla KEM ve DEM üzerinde işlemler yapılıp daha sağlam bir yapı kullanılmıştır [10].

3. VERİ SIKIŞTIRMA TEKNİKLERİ

Dijital veriler çeşitli tekniklerle çeşitli donanım ürünlerinde saklanmaktadır. Bu veriler 1 ve 0' lardan oluşan bitler halinde saklanır. 1 ve 0 diye bahsedilen "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 metinler uygun teknik kullanarak %90' a kadar sıkıştırılabilir. Böylece saklama belleğinden veya haberleşme kanalı band genişliğinden büyük ölçüde tasarruf sağlanmaktadır.

Veri sıkıştırma tekniklerini kayıplı ve kayıpsız teknikler olarak iki ana başlık altında toplanabilir.

3.1 Kayıplı Sıkıştırma

Kayıplı sıkıştırma yöntemleri, çıkartılması verinin bütünlüğünü en az düzeyde etkileyecek olan veri kümelerini çıkartarak, geriye kalan veri kümelerinin de kayıpsız sıkıştırmaya tâbi tutulması temeline dayanır. Bir veri kayıplı bir sıkıştırma yöntemi ile sıkıştırılırsa, verinin tamamı değil, sadece belirli bir kısmı geri getirilebilir. Veri birebir aynı şekilde geri getirilemediği için bu tür yöntemlere kayıplı yöntemler denir. Kayıplı veri sıkıştırma genellikle belirli bir miktar veri kaybının insan gözü ve kulağı tarafından

hissedilemeyeceği durumlarda, örneğin fotoğraf görüntüleri, video ve ses için kullanılır. İnsan gözü ve kulağı yüksek frekans değerlerine daha az hassasiyet gösterdiği için, genellikle veri eleme işlemi yüksek frekans değerlerinin simgeleyen veriler üzerinde yapılır [11].

Şekil 3-1 Sıkıştırmaya elverişli çeşitli formatlar [28]

Şekil 3-1' de bilgi teknolojilerinde kullanılan çeşitli sıkıştırma formatları görülmektedir. Bu formatların oluşmasına yardımcı olan yöntemlerin bazılarına ses, görüntü ve hareketli görüntü ana başlıkları altında değinilmektedir.

3.1.1 Kayıplı Ses Sıkıştırma Yöntemleri

3.1.1.1 Mpeg-1 Layer (Mp3)

1987 yılında geliştirilen mp3, bugün dünyanın en yaygın kayıplı ses formatı konumunda. Bundaki en önemli etken CD' deki standart bir müzik parçasının dosya boyutunun 10-14 kat arasında 'küçültülebilmesi', böylece herhangi bir dijital kayıt ortamına (CD, flash bellek, mp3 çalar vb.) daha çok

parça sığdırılabilmesi. Bu etkene, boyut küçüldüğü için sanal âlemde dosya transferinin çok daha kısa zaman almasını da eklemek gerek. Bu format hakkında bilinmesi gereken temel konu; sıkıştırma oranını yükselttikçe dosya boyutunun küçüleceği ancak sesin kalitesinden taviz verileceğidir. Bu nedenle sıkıştırma oranını mp3' ü kullanım amacına göre belirlemek yerinde olabilir [11].

3.1.1.2 Wma - Windows Media Audio

Windows' un gidişatı görüp mp3' e cevap olarak geliştirdiği bu formatın mp3' e göre iki temel farkından/avantajından söz etmek mümkün. İlki, aynı ses kalitesini daha düşük veri oranları ile sunabilmesi (128 kbps mp3 = 64 kbps WMA gibi) ve dijital kopyalamaya karşı koruma içermesi. 2000' den sonraki tüm Windows işletim sistemlerinde yer alan Windows Medya Çalar yazılımına entegre edilmiş olarak kullanılabilen bu format, internet üzerinden duraksız veri akışına da (streaming) olanak sağlıyor [12].

3.1.1.3 Aac - Advanced Audio Coding

AAC (Advanced Audio Coding- Gelişmiş Ses Kodlaması) için dijital ses dünyasının 'yükselen değeri' şeklinde bir tanımlama yapmak yanlış olmaz. AAC başta dijital radyo ve TV yayınları olmak üzere şimdiden dijital sesin kullanıldığı birçok alan/uygulamada standart olarak kabul edilmiştir. Format asıl çıkışını "iPod çılgınlığı" ve internetin ilk ve en organize ücretli parça indirme yazılımı "itunes" ile yapmıştır. Itunes üzerinden satışı yapılan parçaların tamamı AAC (bir başka ifade ile mp4 ya da m4a) formatındadır [12].

3.1.2 Kayıplı Görüntülü Sıkıştırma Yöntemleri

3.1.2.1 JPEG Standardı

JPEG, Joint Photographic Experts Group (*Birleşik Fotoğraf Uzmanları Grubu*) tarafından standartlaştırılmış bir sayısal görüntü kodlama biçimidir [29].

JPEG, ayarlanabilir kayıplı sıkıştırma kullanır, dolayısıyla JPEG verisinden okunan görüntü ile veriyi yaratmak için kullanılan görüntü aynı değildir. Ancak, kayıplar insan görme sisteminin daha az önem verdiği detaylarda gerçekleştiği için çoğu zaman fark edilmez [29].

İnsan retinası, yapısı nedeniyle bir görüntüdeki renk verisini parlaklık verisine göre daha düşük çözünürlükte görür. Dolayısıyla renk verisinin parlaklığa göre daha düşük bir çözünürlükte örneklenmesi, çoğunlukla hissedilir bir değişikliğe neden olmaz. JPEG, yatayda ve/veya düşeyde renk verisinin parlaklığın yarısı çözünürlükte örneklenmesine imkân verir [29].

3.1.2.2 JPEG2000 Standardı

JPEG2000 standardı, JPEG standardının kısıtlamalarını gidermek ve düşük bit oranlarında yüksek kalitede görüntüler elde etmek amacıyla tasarlanmıştır. Ayrık Dalgacık Dönüşümü (Discrete Wavalet Transform - DVT) teknolojisini temel alarak, bilinen en iyi sıkıştırma teknolojilerinin kullanılmasıyla oluşturulmuş bir kodlama sistemidir [6].

JPEG' teki en büyük dezavantajlarından biri; görüntü önce 8x8' lik bloklara ayrılıp daha sonra Ayrık Kosinüs Dönüşümü (Discrete Cosine Conversion - DCT) uygulandığı için, özellikle yüksek sıkıştırma oranlarında bloklar arası geçişin keskinleşmesi ve gözle fark edilir hale gelmesidir.

Özellikle 1/30' dan yüksek oranlarda sıkıştırılmış görüntülerde bloklar belli olur. JPEG2000' de böyle bir bloklara bölme olmadığı için bu dezavantaj ortadan kalkmıştır [6].

3.1.3 Kayıplı Hareketli Görüntü Sıkıştırma Yöntemleri

3.1.3.1 Mpeg-1 Ve Mpeg-2

MPEG (Moving Pictures Experts Group), 1988 yılında hareketli görüntü ve sesin sayısal saklama birimlerinde farklı sıkıştırma oranları ile saklanabilmesi için kurulmuştur. MPEG-1 tamamlanmadan 1990 yılında MPEG-2 projesi de başlatılmıştır. MPEG-1 1993 yılında, MPEG-2'nin ilk sürümü de 1994 yılında bitirilmiştir. Sonraki yıllarda MPEG-2 standardına yeni özellikler eklenerek geliştirilmesine devam edilmiştir [11].

MPEG-1 standardı, 1-1,5Mbps aralığında çalışması için genellikle 352x288 çözünürlükte 25fps (frame-per-second; kare/saniye) ile veya 352x240 çözünürlükte 30fps ile kullanılır. VHS kalitesini sağlayabildiği için VCD'lerin temel standardı haline gelmiştir [11].

MPEG-2 standardı, MPEG-1 gibi temel bir uygulama alanını hedeflemediği için uygulamadan bağımsız, genel bir standart olmuştur. MPEG-2, Sayısal Uydu Alıcılarının, DVD'nin (Digital Versatile Disc) ve HDTV'nin (High Definition TeleVision) temel standardı olmuştur [11].

3.1.3.2 Mpeg-4

MPEG-4 1993 yılında geliştirilmeye başlamıştır. 1999'da ilk sürümü (v1) tamamlan MPEG-4'ün, 2000'de ikinci (v2), 2001'de üçüncü (v3)

sürümleri tamamlandı. DivX, XviD, QuickTime, WMV gibi günümüzde en çok kullanılan video sıkıştırma formatları MPEG-4 tabanlıdır [11].

3.2 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 özetlenirse, kayıplı sıkıştırmalarda olduğu gibi insan gözü ve kulağının hassasiyetiyle ilgisi olmayan, metin belgeleri, çalıştırılabilir program dosyaları (.exe), kaynak kodları gibi dosyalar kayıpsız sıkıştırılmak zorundadır. Kayıplı sıkıştırma yöntemlerinde bahsedilen ses, görüntü ve hareketli görüntü tipindeki dosyalar, kayıpsız sıkıştırma yöntemleri kullanılarak da sıkıştırılabilir. Ancak bu yöntemde ki sıkıştırma oranı kayıplı sıkıştırma yöntemine göre düşüktür.

3.2.1 Kayıpsız Ses Sıkıştırma Yöntemleri

3.2.1.1 Dpcm (Differential Pulse Code Modulation)

DPCM Bell Laboratuarlar' ında İkinci Dünya Savaşı'nın bitiminden birkaç sene sonra geliştirilmiştir. Konuşma kodlama sistemi olarak çok popüler olan DPCM, halen sayısal telefon iletişiminde geniş çaplı olarak kullanılmaktadır [6].

Basit fark kodlama yöntemleri her örneğinin bir önceki örneğe olan farkını kodlarken, DPCM ise, kodlanacak örnek değeri önceki örneklerin yardımı ile tahmin ederek, örneğin gerçek değeri ile tahmin edilen değer

arasındaki farkı kodlar. Öngörü yapmanın ana fikri fark değerlerini daha da küçültmektir. Örneğin, eğer önceki 3 sembol 2, 5 ve 8 ise, her seferinde 3 birimlik artış olduğu ve bir sonraki değerin 11 olabileceği tahmin edilebilir. Eğer bir sonraki değer 12 çıkarsa, 8 ile 12 arasındaki fark olan 4 değerini kodlamak yerine, 11 ile 12 arasındaki daha küçük olan 1 değerini kodlamak daha etkili sıkıştırma sağlayacaktır [6].

3.2.1.2 Adpcm (Adaptive Differential Pulse Code Modulation)

Öngörü işleminden önce, ses verisi blok ya da çerçeve olarak isimlendirilen yapılara bölünür. Konuşma kodlaması yapılırken blok uzunluğu genellikle 16 veya 20ms olarak belirlenir. Konuşma genellikle 8 kHz ile kodlandığı için bir blok 128 ya da 160 örnekten oluşur. İleri yönde uyarlanır öngörüde bir çerçeve işlenmeden önce, o çerçeve için en uygun öngörü ve niceleme katsayıları belirlenir ve bu değerler öngörü işleminin sonuç değerleri ile birlikte yan bilgi olarak aktarılır. Geri yönde uyarlanır öngörüde ise, en uygun öngörü ve niceleme bir önceki çerçeveye dayalı olarak belirlendiği için, kod çözücü de aynı öngörüyü yapabileceğinden, sadece çerçeve bilgisinin gönderilmesi yeterlidir. Geri yönde uyarlanır öngörüde yan bilgi gönderilmediği için veri oranı daha düşüktür, fakat bir önceki çerçeve düşünülerek tahmin yapıldığı için tahmin hataları daha büyüktür [6].

Örnekleme bit oranı gereğinden daha düşük belirlenirse, oluşacak olan örnekleme gürültüsü nedeniyle DPCM ve ADPCM kayıplı olabilir [6].

3.2.1.3 Mpeg-4 Als (Audio Lossless Coding)

Etkili ve hızlı kayıpsız ses sıkıştırma yapan MPEG-4 ALS standardında diğer kayıpsız sıkıştırma yapan yöntemlerde olmayan birçok özellik mevcuttur:

• 32-bit PCM çözünürlüğünü ve 192kHz frekansı destekler.(CD-DA' da 16-bit ve 44.1 kHz)
• 65536 kanala kadar çoklu kanal desteği vardır. (CD-DA'da 2)
• IEEE754 biçimindeki 32-bit kayan noktalı ses verisini destekler. (CD-DA'da tamsayı)
• wav, aiff, au, bwf ve raw gibi birçok sıkıştırılmamış ses formatını sıkıştırabilir.
• Kodlanmış verinin her bölgesine hızlı erişim sağlar.
• MPEG-4 Video sıkıştırması ile kullanılabilir.
• Kodlayıcı parametrelerinin sahip olduğu esneklik birçok uygulama için elverişlidir [6].

3.2.2 Kayıpsız Görüntü Sıkıştırma Yöntemleri

Kayıpsız görüntü sıkıştırma, sadece olasılık tabanlı ya da sözlük tabanlı bir yöntem ile yapılabileceği gibi, ikisini bir arada kullanan bir yöntem ile de yapılabilir. Örneğin çok popüler olan PNG görüntü sıkıştırma yöntemi, LZ77 ve Huffman sıkıştırma algoritmalarını bir arada kullanan, DEFLATE sıkıştırma yöntemini kullanır. JPEG-LS, JBIG gibi özellikle kayıpsız görüntü sıkıştırmak için tasarlanmış yöntemler de vardır [6].

3.2.2.1 Rle (Run Length Eencoding)

Her tür veri için kullanılabilir bir algoritma olsa da, aynı sembolün ardışık olarak çok defa tekrar etmesi durumunda iyi bir sıkıştırma oranı sağladığı için, genellikle görüntü sıkıştırmada kullanılır. BMP, PCX ve TIFF görüntü dosya formatları, RLE ile sıkıştırma yapabilir [6].

3.2.2.2 Jpeg-Ls

Görüntüleri kayıpsız veya az kayıplı sıkıştırabilen JPEG-LS, 1998 yılının sonlarında tamamlanmıştır [11].

3.2.2.3 Jbig Standardı

JBIG (Joint Bi-level Image Experts Group), 1988'de ISO/IEC JTC1/SC29/WG1 grubu ile CCITT SGVIII grubunun birleşmesi ile oluşturulmuştur. 1-bit derinlikli (bi-level) görüntü verisini kayıpsız olarak sıkıştırmak için kullanılabilecek bir yöntem bulmak amacıyla kurulan grup, 1993'te JBIG standardını tamamlamış ve yayınlamıştır [ISO/IEC, 1993]. Bu tarihten sonra JBIG standardı MR ve MMR'nin yerini alarak, faks iletiminde yaygın olarak kullanılan bir standart haline gelmiştir [6].

3.2.2.4 Jbig – 2

ISO ve ITU'nun ortak geliştirdiği bir standarttır. ITU tarafında T.88 olarak, ISO tarafında ise 14492 kod numarası ile projelendirilen JBIG-2 standardı, 2000 yılında tamamlanmıştır [ITU, 2000 ve ISO, 2001]. Önceki JBIG standardı gibi, çoğunlukla faks iletiminde kullanılan 1-bit renk derinliğine sahip görüntüler için geliştirilmiş olan JBIG- 2, kayıpsız sıkıştırma yapabildiği gibi kayıplı sıkıştırma da yapabilir. [6]

3.2.3 Kayıpsız Sıkıştırma Kullanan Görüntü Dosya Formatları

3.2.3.1 Gif (Graphics Interchange Format)

1987'de CompuServe tarafından yaratılan algoritmalardan biridir. En fazla 8-bit renk derinliğine (28 = 256 renk) sahip görüntülere izin verir. Bu sebepten dolayı fotoğraf görüntülerinin sıkıştırılmasında yetersiz kalsa da, birkaç rengin çoğunlukta olduğu grafiksel gösterimler ve basit şekiller gibi görüntülerin kayıpsız olarak sıkıştırılmasında halen kullanılmaktadır [11].

3.2.3.2 Tiff (Tagged Image File Format)

Grafik, fotoğraf gibi dosyalar için kullanılan bir biçimdir. Aldus isimli şirket tarafından üretilip 1986 yılında ilk sürümü duyurulmuştur. 1994 yılında Aldus Corp ile Adobe Systems' in birleşmesinden sonra TIFF 6.0 geliştirilmiş ve bir çok yeni özellikler eklenmiştir. JPEG ve PNG gibi TIFF de yüksek renk derinliği olan görüntülerde kullanılır. Photoshop, GIMP gibi görüntü işleme programları TIFF biçimini destekler [13].

3.2.3.3 Png (Portable Network Graphics)

PNG, "Taşınabilir Ağ Grafiği" anlamındaki *(Portable Network Graphics)*' in kısaltmasıdır ve kayıpsız sıkıştırarak görüntü saklamak için kullanılan bir saklama biçimidir. PNG biçiminde paletli ya da gerçek renkte görüntüler seçimlik bir saydamlık kanalıyla saklanabilir [14].

3.2.4 Kayıpsız Hareketli Görüntü Sıkıştırma Yöntemleri

Kayıpsız hareketli görüntü sıkıştırma yöntemleri genellikle gerçek zamanlı video yakalama gibi hızlı kodlama yapmanın gerekli olduğu durumlarda kullanılmaktaydı. Pratikte, kayıpsız bir yöntem ile sıkıştırılmış olan bir hareketli görüntü çok yer kapladığı için ve saklama ünitelerinin kapasiteleri 10 sene öncesine kadar fazla büyük olmadığı için, saklama amacıyla daha sonra kayıplı bir yöntem ile tekrar sıkıştırılmaktaydı. Son yıllarda üretilen yüksek hızlı işlemciler sayesinde, kayıplı bir yöntem ile de gerçek zamanlı video yakalama işlemleri yapılabilmektedir. Bu nedenle günümüzde hareketli görüntü yakalama işlemlerinde de genellikle kayıplı kodlayıcılar tercih edilmektedir [11].

Huffyuv, SheerVideo, CorePNG, MSU Lossless Video Codec, LCL, Lagarith, SKYUV, Lossless Motion JPEG gibi birçok kayıpsız sıkıştırma yöntemi vardır [6].

3.2.4.1 Huffyuv

Ben Rudiak-Gould tarafından 90' lı yıllarda geliştirilmiş olan Huffyuv, en çok kullanılan kayıpsız hareketli görüntü (video) sıkıştırma algoritmalarından biridir. Sıkıştırma yapmayan YUV'un alternatifi olmuştur. Sıkıştırma ve açma işlemlerinde hızlı bir algoritmadır. Kaynak kodu açık, dağıtımı serbesttir [6].

3.2.4.2 Msu

"MSU (Moscow State University) Graphics and Media Lab." tarafından geliştirilmiştir. Her çerçeveyi ayrı bir resim gibi kodlamayıp, çerçeveler arasındaki yüksek oranda benzerliklerden faydalanarak çerçeveler-arası (inter-frame) öngörüsü kullanan bu yöntem, Huffyuv' a göre daha yavaş

çalışmasına rağmen daha iyi sıkıştırma oranı sağlamaktadır. Kayıplı sıkıştırma seçeneği de bulunan MSU, istenirse Huffyuv gibi, tüm çerçeveleri ayrı bir resim gibi de kodlayabilir [6].

3.2.5 Kayıpsız Karakter Tipli Sıkıştırma Teknikleri

3.2.5.1 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 kullanıldığının bilinmesi 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 ayrılabilir. Bilgisayar hafızasında bulunan herhangi bir dosyayı sıkıştırmak için statik huffman algoritması kullanılır. Çünkü sıkıştırma yapmadan önce dosya içindeki bütün sembolleri tarayarak dosya içinde hangi sembolün kaç adet bulunduğunu tespit edip frekans tablosunu rahatlıkla oluşturulabilir. Dinamik huffman tekniğinde ise frekans tablosuna önceden ihtiyaç duyulmaz. Frekans tablosu o an gelen sembollerle oluşturulur. Özellikle haberleşme sistemleri gibi önceden ne geleceği belli olmayan sistemlerde *dinamik huffman* tekniği kullanılır.

Static huffman algoritması kullanılacaksa bunun için de iki yaklaşım vardır. Birinci yaklaşım, metin dosyasının diline göre sabit bir frekans tablosunu kullanmaktır. Örneğin; Türkçe bir metin dosyasında "a" ve "e"

harflerine çok sık rastlanırken "ğ" harfine çok az rastlanır. Dolayısıyla "ğ" harfi daha fazla bitle "a" ve "e" harfi daha az bitle kodlanır. Frekans tablosunu elde etmek için kullanılan diğer bir yöntem ise metni baştan sona tarayarak her bir karakterin frekansını bulmaktır [7].

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 [7].

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

3.2.5.1.1 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 oluşturulduğu aşağıdaki adımlarda gösterilmektedir. Örnekte kullanılacak frekans tablosu Tablo 3.1' de verilmiştir.

Tablo 3.1' den şu anlaşılmalıdır: Önceden oluşturulmuş olan metin dosyasında "e" karakteri 55 kez, "r" karakteri 40 kez, …., "i" karakteri 9 kez kullanılmış. İşte bu tablo kullanılarak, hangi karakterin hangi bit dizisiyle temsil edileceği bunmaktadır.

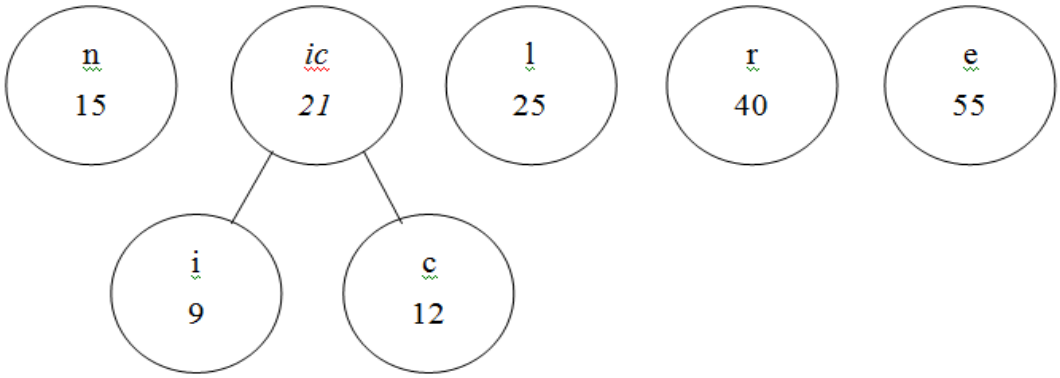
Tablo 3-1 Frekans Tablosu

SEMBOL (KARAKTER)	SEMBOL FREKANSI
e	55
r	40
l	25
n	15
c	12
i	9

Şekil 3-2 Sembollerin frekanslarına göre sıralanması

1. İlk önce bütün semboller frekanslarına göre küçükten büyüğe doğru sıralanır.

2. Ağaç yapısı oluşturmak için ilk adım olarak 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. Örneğin Şekil 3.3' de "i" ve "c" sembolleri toplanarak "21" frekanslı bir "ic" düğümü oluşturulur. 21 frekanslı bir sembol Şekil 3.3' te "n" ve "l" sembolleri arasına yerleşir. "i" ve "c" düğümleri ise yeni oluşturulan düğümün dalları şeklinde kalır.

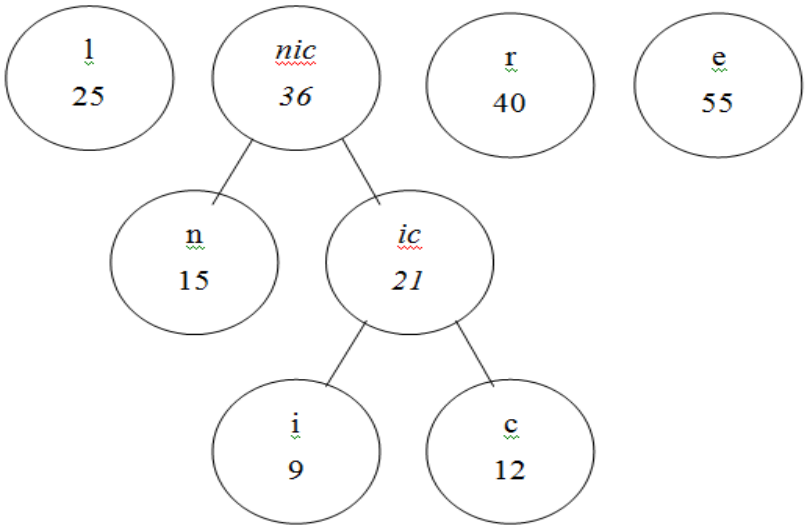


Şekil 3-3 İlk Düğümün Oluşturulması

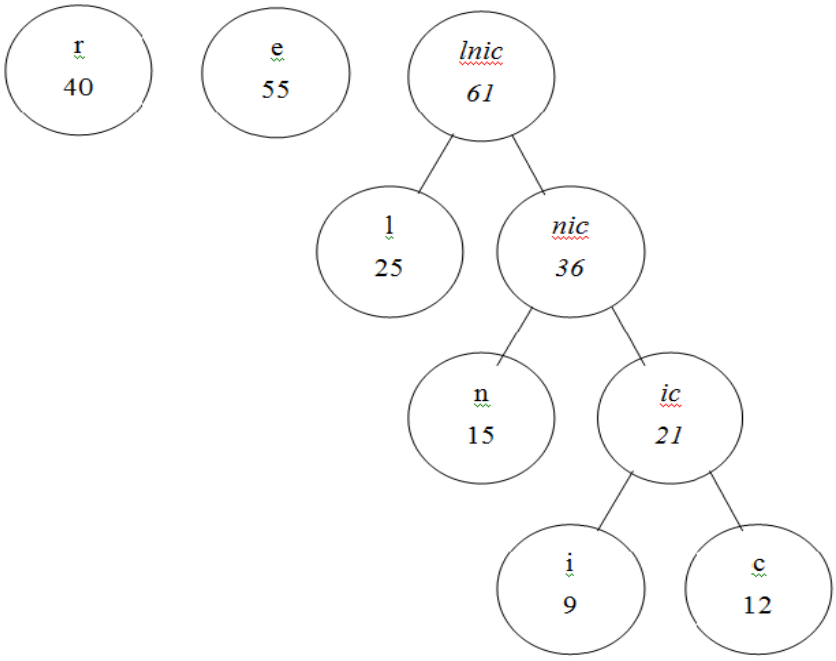
3. 2. adımdaki işlem tekrarlanarak en küçük frekanslı iki düğüm tekrar toplanır ve yeni bir düğüm oluşturulur. Bu yeni düğümün frekansı 36 olacağı için "l" ve "r" düğümleri arasına yerleşecektir. Yeni dizi aşağıdaki Şekil 3.4' te ki gibi olacaktır.

4. 2. adımdaki işlem tekrarlanarak en küçük frekanslı iki düğüm tekrar toplanır ve yeni bir düğüm oluşturulur. Bu yeni düğümün frekansı 61 olacağı için "e"' nin sağına yerleşecektir. Yani, en son düğüm frekansı en yüksek düğüm olacaktır. Yeni dizi Şekil 3.5' de ki gibi olacaktır. Dikkat edelirse her dalın en ucunda semboller bulunmaktadır. Dalların ucundaki düğümlere özel olarak *yaprak(leaf)* denilmektedir. Sona yaklaştıkça bilgisayar bilimlerinde önemli bir veri yapısı olan Tree(ağaç) veri yapısına yaklaşıldığı görülmektedir.

5. 2. adımdaki işlem tekrarlanarak en küçük frekanslı iki düğüm tekrar toplanır ve yeni bir düğüm oluşturulur. Bu yeni düğümün frekansı 95 olacağı için "lnic" düğümünden sonra yerleşecektir. Yeni dizi Şekil 3.6' daki gibi olacaktır. Dikkat edilirse her bir düğümün frekansı o düğümün sağ ve sol düğümlerinin frekanslarının toplamına eşit olmaktadır. Aynı durum düğüm sembolleri içinde geçerlidir.

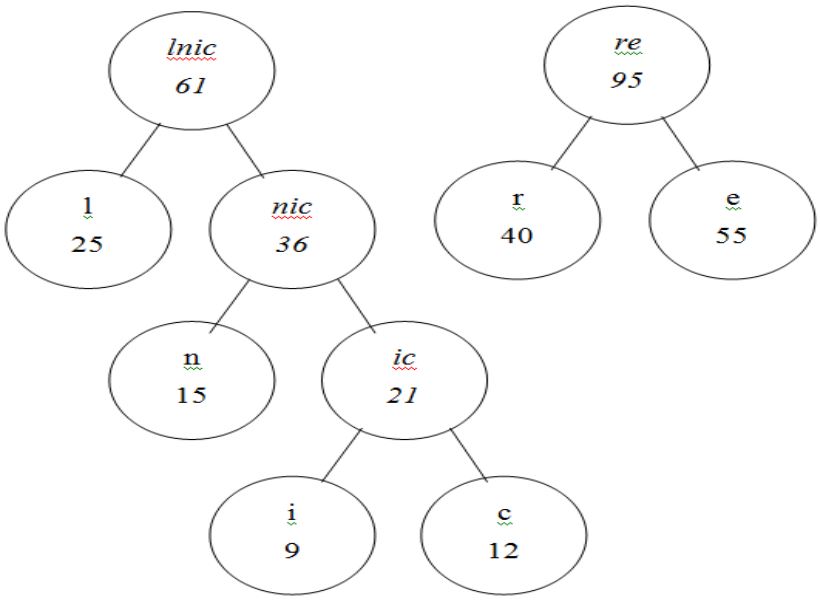


Şekil 3-4 ikinci Düğümlerin Oluşturulması

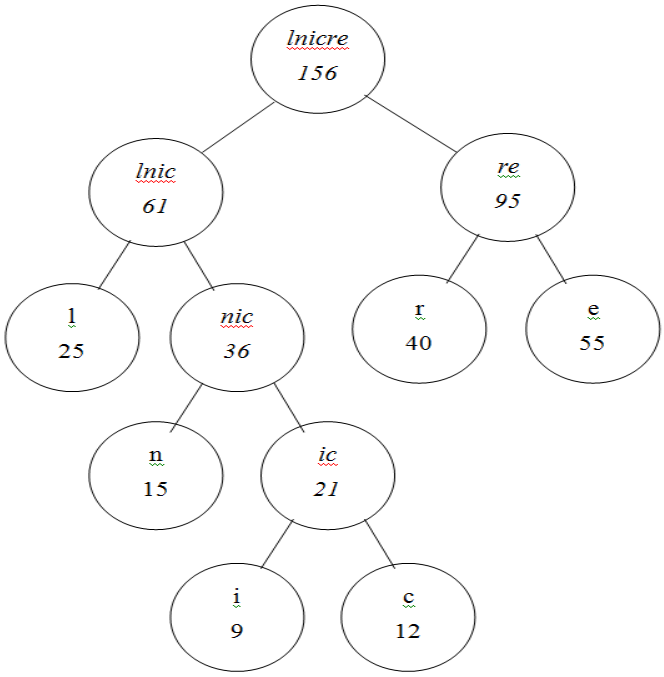


Şekil 3-5 Üçüncü Düğümlerin Oluşturulması

6. 2. adımdaki işlem en tepede tek bir düğüm kalana kadar tekrar edilir. En son kalan düğüm Huffman ağacının kök düğümü (root node) olarak adlandırılır. Son düğümün frekansı 156 olacaktır. Böylece huffman ağacının son hali Şekil 3.7' de ki gibi olacaktır.

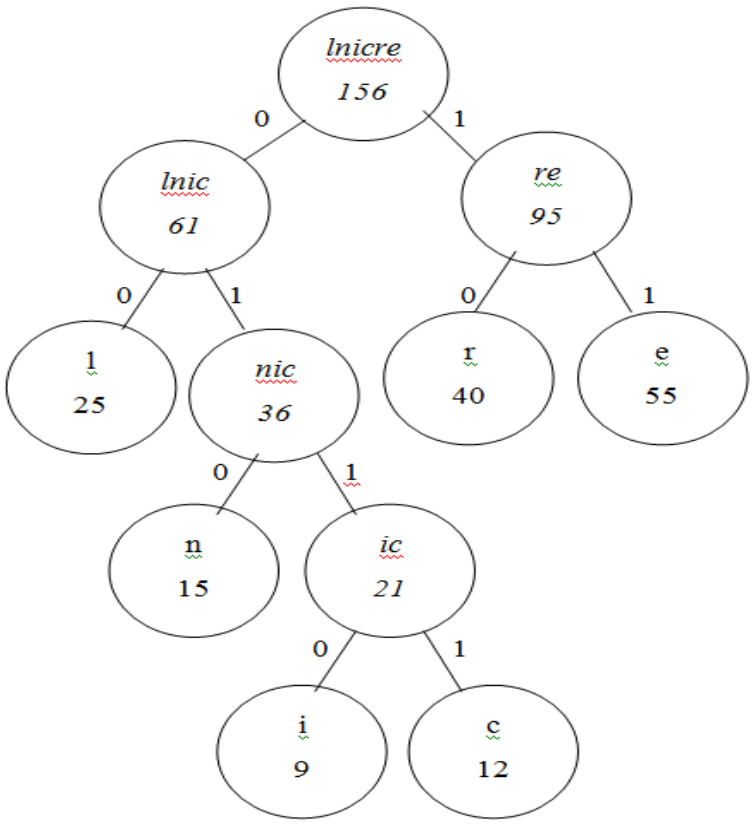


Şekil 3-6 Dördüncü Düğümlerin Oluşturulması



Şekil 3-7 Huffman ağacının kök düğümü(root node)

Artık huffman ağacını oluşturuldu. Şimdi her bir sembole karşılık gelen kod oluşturuluyor. Bu kodları oluştururken ağacın en tepesinden yani kök düğümden başalamak gerekir. Kök düğümün sağ ve sol düğümlerine giden dala sırasıyla "0" ve "1" kodları verilir. Bunun tam tersini yapılabilir, yani “1” ve “0” şeklinde de kodlar verilebilir. Ancak dikkat edilmesi gereken nokta ilk hangi tarafa “0” veya “1” verildiyse aynı şekilde devam edilmesi gerektiğidir. “lnic” düğümüne gelen dal “0”, “re” düğümüne gelen dal “1” seçilir ve alt düğümlere de aynı şekilde uygulanırsa dalların kodlarla işaretlenmiş hali Şekil 3.8’ de ki gibi olacaktır.



Şekil 3-8 Huffman Ağaç Yapısının Oluşturulması

Kodlarla dalları işaretledikten sonra hangi sembolün hangi bit dizisiyle kodlanacağı bulunur. Dikkat edilirse 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 3.2’ de bütün sembollere karşılık gelen kodlar gösterilmektedir.

Dikkat edilirse 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 orijinal verinin dışında başka bir veri elde etme ihtimali sıfırdır (Tabi kodlamanın doğru yapıldığını varsayılıyor) [7].

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

Tablo 3-2 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
i	9	4	0110

Sıkıştırma öncesi gereken bit sayısını bulanacak olursa: Her bir karakter eşit uzunlukta yani 8 bit ile temsil edildiğinden toplam karakter sayısı olan (55+40+25+15+12+9) = 156 ile 8 sayısının çarpılması gerekiyor. Orijinal veri sıkıştırılmadan saklanırsa *156*8 = 1248* bit gerekmektedir.

Huffman algoritması kullanılarak sıkıştırma yapılırsa, kaç bitlik bilgiye ihtiyaç duyulduğu hesaplanır: 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 duyulur. (Tablo 2.2) 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 ihtiyaç 369 bite indirildi. Yani yaklaşık olarak %70 gibi bir sıkıştırma gerçekleştirilmiş oldu. 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.

3.2.5.1.2 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 elde frekans tablosu ve sıkıştırılmış veri dizisi varsa işlemlerin tersini yaparak orijinal veri elde edilebilir. 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 [7]. Bu da aslında *kayıpsız sıkıştırmanın* bir sonucudur.

3.2.5.2 Shannon-Fano

"*Shannon-Fano kodlaması Huffman kodlamasına benzer; yalnızca kodlama ağacı farklı şekilde kurulur. Bu yöntemde de karakterlerin kullanım sıklığı olasılıkları hesaplanır ve olasılığı en büyük olan yukarıda olacak biçimde büyükten küçüğe doğru sıralanır. Kodlama ağacının oluşturulması için ilk önce kümedeki karakterler iki alt kümeye bölünür; her iki alt kümede bulunan karakterlerin olasılıklarının toplamının eşit olmasına dikkat edilir. Yani önemli olan karakter sayısı değil, olasılıklar toplamıdır. Yukarıda kalan altküme içinde bulunan karakterlere atanacak kodun ilk biti 0, aşağıda kalan altkümedekilere ise 1 atanır. Daha sonra bu iki alt küme de aynı şekilde altkümelere bölünerek karakterlere atanacak kodların bitleri elde edilir. Bölme işlemi altkümelerde 1 karakter kalana kadar sürdürülür* [5,s.402]."

Huffman tekniğinde olduğu gibi sembollerin ve kullanım sıklıklarının bulunduğu C = {g:10, f:10, e:5, d:20, c:15, b:15, a:25} şeklinde bir küme ele alındığında, bu sembol kümesiyle Shannon-Fano kodlama ağacınının nasıl oluşturulacağı açıklanıyor. Yukarıda ki tanımda da belirtildiği gibi öncelikle semboller olasılıklarına göre büyükten küçüğe doğru sıralanıyor. İlk adımda, semboller, toplam olasılıkları olabildiğince eşit şekilde iki parçaya ayrılır. Bu şekilde sembollere verilecek kodların ilk bitleri bulunmuş olur. Ayrılan parçalardan üstte kalan kısma 0, altta kalan kısma 1 atanır; bunlar sembollere verilen kodların ilk bitleridir. Bu şekilde parçaların her birinde tek bir sembol kalana kadar adımlar sürdürülürse Tablo 3.3' deki kodlama ağacı ve kodlar elde edilmiş olur.

Tablo 3-3 Shannon-Fano kodlama ağacı

Semboller	Olasılıklar								Atanan Kodlar
a	0.25	0	*0.45	0					00
d	0.20	0		1					01
c	0.15	1	*0.55	0	*0.30	0			100
b	0.15	1		0		1			101
f	0.10	1		1	*0.25	0	*0.20		110
g	0.10	1		1		1	*0.15	0	1110
e	0.05	1		1		1		1	1111

*: altkümeler ve toplam olasılıkları

3.2.5.3 Aritmetik Kodlama

"Huffman ve Shannon-Fano kodlamasında olduğu gibi bir kodlama ağacı kurulması gerektirmez. Ancak karakterlerin kullanım sıklığı olasılıkları bilinmelidir. Aritmetik kodlama ELIAS tarafından önerilmiş olup ilk kez ABRAMSON tarafından sunulmuştur (1963). Daha sonra, aritmetik kodlamanın gerçeklenmesi RISSANEN (1976), PASCO (1976) tarafından geliştirilmiştir [5,s.402]."

"Aritmetik kodlamanın temeli, kodlanacak veri parçasını 0 ile 1 arasında gerçel sayı aralığı ile temsil etmeye dayanır. Veri parçasındaki her karakter bu gerçel sayı aralığını daraltır. Aralık ne kadar darsa, onu temsil etmek için gerekli bit sayısı artar; genişse azalır. Kullanma sıklığı olasılığı küçük olan karakterler bu aralığı hızlı şekilde azaltırken, büyük olanlar daha az daraltır. Eğer veride fazlalık çoksa, sayı aralığı az daralacağı için, veri az sayıda bit ile kodlanabilir. Kodlama işlemine geçmeden önce kümedeki karakterler, olasılıkları, (kendi ve kendinden önceki karakterlerin) toplanmış olasılıkları ve her karakterin sayısal aralığından oluşan bir tablo oluşturulur; kodlama işlemi, bu tablodaki değerlerin aritmetik işlemlerde kullanılmasıyla gerçeklenir. Amaç aralığın hesaplanacağı iki gerçel sayıyı bulmaktır; birine

sağdaki, diğerine soldaki sayı denir. Aralığın hesaplanası için aşağıdaki bağıntılar kullanılır [5,s.403]."

$sol_i = sol_{i-1} + aralık_{i-1}$ * önceki toplam olasılık (2.1.)

$sağ_i = sol_i + aralık_{i-1}$ * o andaki karakterin olasılığı (2.2.)

Aralığı gösteren sol ve sağın başlangıç değeri : [0,1)

"Aritmetik kodlamanın ardında yatan temel düşünce, basit bir örnekle kolayca anlaşılabilir. Karakter kümesinin A, B, C, D ve ^ (NULL) karakterlerini içerdiği ve olasılıkların sırasıyla 0.20, 0.50, 0.10, 0.15 ve 0.05 olduğu varsayılırsa, Tablo 3-4' te ki gibi bir tablo oluşturulur [5,s.403]."

"Verinin BACA^ olduğu varsayılırsa, 2.1. ve 2.2. bağıntıları uyarınca kodlama işlemi sonucu aşağıdaki adımlarla elde edilir [5,s.403]:"

1- *"Başlangıçta aralığı gösteren sayılar [0.0, 1.0)' dır. Verinin ilk karakteri olan B, bu aralığı kendi aralığına çeker; yani aralık [0.2, 0.7) olur. Burada sol_1=0.2, $sağ_1$=0.7 ve $aralık_1$=0.5' dir [5,s.403]."*

2- *"Verideki ikinci karakter olan A' nın aralığı daraltması ilgili bağıntılara göre şöyle hesaplanır [5,s.403]:"*

$sol_2 = sol_1 + aralık_1$ * A' ya kadar olan toplam olasılık = 0.2 + 0.5*0.0 = 0.2

$sağ_2 = sol_2 + aralık_1 * O_2$ = 0.2 + 0.5*0.2 =0.3

Bunlara göre sol_2 = 0.2, $sağ_2$ = 0.3 ve $aralık_2$ = 0.1 çıkar.

$[sol_2, sağ_2)$ = [0.2, 0.3)

Tablo 3-4 Aritmetik Kodlama [5,s.403]

Karakter	olasılığı	toplam olasılık	aralık
A	0.20	0.2	**[0.0,0.2)**
B	0.50	0.7	**[0.2,0.7)**
C	0.10	0.8	**[0.7,0.8)**
D	0.15	0.95	**[0.8,0.95)**
^	0.05	1.0	**[0.95,1.0)**

3- *"Üçüncü karakter olan C' nin aralığı daraltması da, benzer işlemlerle şöyle hesaplanır [5,s.404]:"*

*sol_3 = sol_2 + $aralık_2$ * C' ye kadar olan toplam olasılık = 0.2 + 0.1*0.7 = 0.27*

*$sağ_3$ = sol_3 + $aralık_2$ * O_3 = 0.27 + 0.1*0.1 =0.28*

Bunlara göre sol_3 = 0.27, $sağ_3$ = 0.28 ve $aralık_2$ = 0.01 çıkar.

[sol_3, $sağ_3$) = [0.27, 0.28)

4- *"Dördüncü karakter ve ikinci kez bulunan A' nın aralığı daraltması da şöyle hesaplanır [5,s.404]:"*

*Sol_4 = sol_3 + $aralık_3$ * A' ya kadar olan toplam olasılık = 0.27 + 0.01*0.0 = 0.27*

*$Sağ_4$ = sol_4 + $aralık_3$ * O_4 = 0.27 + 0.01*0.2 =0.272*

Bunlara göre sol_4 = 0.27, $sağ_4$ = 0.272 ve $aralık_2$ = 0.002 çıkar.

[sol_4, $sağ_4$) = [0.27, 0.272)

5- *"Verideki beşinci karakter (^) aynı yöntemle hesaplanırsa [5,s.404],"*

*Sol_5 = sol_4 + $aralık_4$ * ^' ya kadar olan topla m olasılık = 0.27 + 0.002*0.95 = 0.2719*

*$Sağ_5$ = sol_5 + $aralık_4$ * O_5 = 0.2719 + 0.002*0.05 =0.272*

[sol_5, $sağ_5$) = [0.2719, 0.272)

olarak bulunur.

"Eğer BACA^ olan veri BABA^ olsaydı; yani üçüncü karakteri C yerine (kullanma olasılığı daha büyük olan) B olsaydı, aralığı oluşturulan değerler [0.22, 0.27) olarak hesaplanırdı. Buradan da, olasılığı büyük olan karakterlerin aralığı daha yavaş daralttığı görülür; C iken 0.01 olan aralık, B iken 0.05 olmaktadır [5,s.404]."

"Aritmetik kodlama sonucunda elde edilen kodun bit uzunluğu tam olarak "entropy" ye eşit olur; ancak gerçeklerken bazı kontrol karakteri kullanılacağından bu eşitlik bozulur. Huffman ve Shannon-Fano kodlamasında ortalama kod uzunluğunun "entropy" ye eşit olması, çoğu zaman elde edilemez [5,s.404]!"

3.2.5.4 Dinamik Sözlük Yaklaşımı

En çok kullanılan dinamik sözlük yaklaşımları LZ77, LZ78 ve onun türevi olan LZW'dir [10].

3.2.5.4.1 Lz77

Abraham Lempel ve Jakob Ziv tarafından geliştirilen ve 1977 yılında yayınladıkları "A Universal Algorithm for Data Compression" isimli makalelerinde tanımladıkları bu yöntem, o yıllarda tüm dünyada büyük ilgi görmüştür. Algoritmanın eksik yönleri zaman içinde farklı bilim adamları tarafından geliştirilmiştir. Sonraları yeni geliştirilen algoritmaların hepsine LZ77 ya da LZ1 ailesi denilmiştir [6].

LZ77 algoritmasında pencere (window) olarak adlandırılan bellek alanı ve katar uyuşma yordamı kullanılır. Kodlanacak veriye ait karakterler bu pencere içine sokulur. Herhangi bir anda, kodlanacak veriye ait bir hecenin pencere içinde olup olmadığına bakılır. Eğer pencere içinde varsa, orada bulunduğu yerin adresi (a) ve uyuşma gösteren hecenin uzunluğu (l) kodlamada kullanılır. Bu iki değişken, (a,l) biçiminde gösterilir. Ve işaretçi olarak adlandırılır; çünkü uyuşma gösteren heceyi işaret eder. Örneğin işaretçi değeri (33,23) ise, 33. karakterden sonra 23 karakter uzunluğunda; (12,2) ise 12. Karakterden sonra 2 karakter uzunluğunda uyuşma oldu anlamlarına gelir [5].

LZ77 algoritmasında, kodlama için kullanılan temel yordam aşağıda verildiği gibidir. Pencere olarak doğrusal dizi kullanılırsa, uyuşan hecenin bulunması için gerekli arama işlemi zaman alır; dolayısıyla, kodlama zamanı uzun sürer. Ancak çözme zamanı oldukça kısadır. LZ77 algoritması 1 kez kodlanıp daha sonra defalarca çözme gerektiren uygulamalara çok uygundur. Örneğin, elektronik kitapların, paket programların yardım dosyalarının saklanması gibi [5]...

While (kodlanacak veri olduğu süre) {
En uzun uyuşma gösteren heceye, a ve l' yi bul;
if (l < en küçük uyuşma uzunluğu)
yalnızca sıradaki ilk karakteri gönder ve pencereyi 1kr. kaydır;
else
işaretçiyi, (a,l)' yi gönder ve pencereyi l kadar kaydır; }[5]

Yukarıdaki yordama göre yeri "annebababababaannebabaanne" ise, kodlama işlemi sonucu "anneba(6,2)(8,4)(1,4)" biçiminde olur. Görüldüğü gibi kodlama işleminde uyuşma göstermeyen karakterler olduğu gibi kalırken, gösterenler işaretçiyle yer değiştirilmektedir. Uygulamada, kodda bulunan işaretçileri karakterlerden ayırmak için bayrak biti veya özel desen kullanılması gerekir [5].

LZ77 ailesi metin tabanlı veri sıkıştırmada büyük aşama kaydedilmesinin yolunu açmış, PKZip, Zip, Lharc (LHA) ve ARJ gibi 80'li ve 90'lı yılların popüler sıkıştırma paketleri değişken uzunluklu kodlayıcı ile desteklenen LZ77 tabanlı algoritmalar kullanmışlardır [6].

3.2.5.4.2 Lz78

Lempel ve Ziv, LZ77'de yer alan tampon büyüklüğünün sınırlı olmasının yarattığı sıkıntıları ortadan kaldırmak için, tampon kullanmayan çok farklı bir yöntem geliştirerek 1978'in Eylül ayında yayınlanan "Compression of Individual Sequences via Variable-Rate Coding" isimli makalelerinde bu algoritmalarına. LZ77'den çok farklı bir yapıda olması nedeniyle, bu algoritma ve geliştirilmiş biçimleri, LZ78 (veya LZ2) ailesi olarak adlandırılmıştır. LZ77'den farklı olarak, sadece metin tabanlı sıkıştırmada değil, bitmap gibi farklı sayısal veriler üzerinde de başarıyla uygulanabilmiştir [6].

LZ78 algoritmasında sözlük olarak adlandırılan bellek alanı, sözlükleri heceleme ve katar uyuşma yordamları kullanılır. Verideki karakterler, kodlama işlemi anında, heceler ve sözcükler şeklinde sözlükte saklanır ve onlara birer kod verilir. Herhangi bir anda, karakter düzeyinde gelen veri

parçası (w) sözlükle aranır. Eğer bulunursa (uyuşma gösteriyorsa) bir sonraki karakter (K) o anki sözcüğe eklenerek (w) yeni sözcük (wK) oluşturulur ve arama işlemine devam edilir. Bu işlem en uzun sözcüğü bulana kadar sürdürülür. Yeni oluşturulan sözcük (wK) sözlükte yoksa, bir önceki sözcüğün kodu kullanılır ve yeni sözcük kendisine yeni kod eklenerek sözlüğe eklenir. Bu işlem sözlük dolana kadar sürer; dolduktan sonra yeni eklemeler yapılmadan kodlamaya devam edilir. Yani sözlük dondurulur (FLUSH yöntemi olarak anılır). Fakat LZ78' in türevleri olan algoritmalarda sözlük dondurulmadığı zaman sıkıştırma başarımının arttığı görülmektedir; sözlüğü dondurma yerine boşaltıp yeniden baştan başlama, iki sözlük oluşturulup doldurulduğunda diğeriyle kodlama işlemine geçme, dolduğunda az kullanılanı çıkartıp yerine yeni sözcük ekleme gibi yöntemler daha iyi sıkıştırma başarımı vermektedir. Ancak sıkıştırma başarımıyla kodlama ve çözme zamanları da oldukça önemlidir; bunlar arasında denge kurulmalıdır. İyi sıkıştırma başarımı verip kodlama için sonsuz zaman gerektiren bir algoritmanın uygulama bulamayacağı açıktır [5]!

```
w = başlangıçta hiçbir karakter yok;
while (kodlanacak veri olduğu süre) {
        K = sıradaki karakter;
        if (eğer yeni oluşturulan sözcük wK sözlükte varsa)
              W = wK;
         else
              w' nin kodunu kullan, wK' yı sözlüğe ekle ve w=K atamasını
yap;
}
```
[5]

Yukarıdaki yordama göre veri "babaababaababababak" ise, sözlüğün kurulması ve kodlama için kullanılacak değerler Tablo 3.5.' de gösterildiği gibi bulunur. Görüldüğü gibi, sözlüğe yeni heceler koyulmaktadır [5].

Eğer sözlük için ayrılan bellek alanı dolarsa, izlenecek yol birçok çalışmaya kaynak olmuştur. Yalın LZ78 algoritmasında sözlük dondurulur [5].

Veri: babaababaababak

sözlüğe eklenen heceler ve atanan kodlar

LZ78' de sözlüğün kurulması için sözlük ağacı (trie tree) kullanılırsa iyi zaman maliyeti olur; ancak gerekli bellek alanı çok büyür [5].

3.2.5.4.3 LZW

Terry Welch 1984'te Unisys (o zamanki adı Sperry Corporation idi) için çalışırken, LZ78 yaklaşımını yüksek performanslı disk ünitelerine uyarlamış ve ortaya çıkan yeni algoritma LZW olarak kabul görmüştür [Welch, 1984]. LZW hem sıkıştırma hem de açma performansı açısından LZ78 ailesinin en iyisi olmayı başarmıştır. Her tip veri üzerinde iyi sonuçlar veren bir algoritma olduğu için, kendisinden sonra gelen birçok algoritma LZW'yi temel almıştır. 1985 yılından beri Unisys LZW'nin patentini elinde bulundurmaktadır [6].

Tablo 3-5 Lz78 algoritması için sözlüğün kurulması

Sözlük	b	ba	baa	bab	a	aa	baba	bak
	1	2	3	4	5	6	7	8
kodlamada kullanılan		(0,b)	(1,a)	(2,a) (2,k)	(2,b)	(0,a)	(1,a)	(4,a)
sonuç					**(4,a):4 nolu hecenin arkasında 'a' ekle anlamındadır.**			
a								**bab**

4. KRİPTOLOJİ

Kriptoloji, Yunanca krypto's (saklı) ve lo'gos (kelime) kelimelerinin birleştirilmesinden oluşturulmuştur ve iletişimde gizlilik bilimi olarak değerlendirilmektedir [9].

Ticari ilişkilerde, devlet işlerinde, askeri işlerde ve personel ilişkilerinde güvenli iş çalışması yapmak büyük bir sorundur [9].

Sistemler arası bağlantılarda ya da herhangi iki nokta arasındaki haberleşmede verinin güvenli bir şekilde gittiğinden emin olmak gerekir. Bunun sağlanması ise gönderilen verinin şifrelenmesi ile olur. Böylece açık haberleşme kanalları kullanılarak verinin güvenli bir şekilde ulaştırılması sağlanır. İletişimde, açık bir haberleşme kanalı kullanılıyorsa gizli tutulmak istenen bilginin yetkisiz bir kişi tarafından dinlenebileceği veya haberleşme kanalına girip (araya girme) veriyi bozabileceği ya da değiştirebileceği (yanlış verinin gönderilmesi) düşüncesi her zaman için önemli bir problem oluşturur [9].

Kriptoloji esas olarak iki bölüme ayrılır: Kriptografi (şifreleme) ve kriptoanaliz (şifre çözme). Gönderilmek istenen orijinal mesaj açık mesaj (plain text) ve bu mesajın şifrelenmiş hali şifreli mesaj (cipher text-cryptograph) olarak adlandırılır [9].

4.1 Kriptografi (Cryptography)

Bir bilginin istenmeyen taraflarca anlaşılmayacak bir hale dönüştürülmesinde kullanılan tekniklerin bütünüdür. Kriptografi gizlilik,

bütünlük, kimlik denetimi, inkar edememe gibi bilgi güvenliği kavramlarını sağlamak için çalışan matematiksel yöntemleri içermektedir [4].

- Gizlilik: Bilgi istenmeyen kişiler tarafından anlaşılmamalıdır.
- Bütünlük: Bilginin iletimi sırasında hiç değiştirilmediği doğrulanmalıdır.
- Kimlik Denetimi: Gönderici ve alıcı birbirlerinin kimliklerini doğrulamalıdır.
- İnkar Edememe: Gönderici bilgiyi gönderdiğini ve alıcı bilgiyi aldığını inkar edememelidir [4].

4.2 Kriptanaliz (Cryptanalysis)

Şifrelenmiş, yani anlamsız bir metinden doğru metni bulma tekniklerinin tümüdür [4].

Şifreleme işlevinin güvenli bir şekilde gerçekleştirilmesi şifreleme sırasında kullanılan tüm yöntem ve bilgilerin gizliliğine dayanır. Ancak herhangi bir nedenle şifreleme işlevlerinin açığa çıkabileceği düşünülerek iletişim güvenliği, şifreleme anahtarı denen ek bilgi ile arttırılmıştır. Bu durumda şifreleme işlemi sırasında açık mesaj, şifre anahtarı aracılığı ile şifrelenir. Bir başka deyişle açık mesaj ile şifrelenmiş mesaj arasındaki geçişler şifreleme algoritmasına bağlı olduğu kadar kullanılan anahtar bilgisine de bağlıdır [9].

4.3 Anahtar (Key)

Anahtar bir sayıdır ve kriptografik algoritma ile çalışır. Anahtarlar temel olarak çok büyük sayılardır. Örneğin 2048 bit gibi. Asimetrik anahtar sistemlerinde büyük anahtar kullanılması daha güvenli şifrelenmiş bilgi oluşturulmasını sağlar. Bir simetrik 80 bit anahtar, 1024 bit açık anahtar gücüne sahiptir [10].

Anahtarları doğru büyüklükte seçmek çok önemlidir. Büyük anahtarlar sağlam güvenlik sağlarken, küçük anahtarlar işlem zamanından tasarruf sağlarlar. Büyük anahtarların kırılması uzun zaman gerektirir. Şifrelenmiş verilerin yıllarca saklanması düşünülüyorsa büyük anahtar kullanılabilir. Tabii gelecekte bilgisayarların ne kadar güçlü ve etkili olabileceklerini kim bilebilir ki. Anahtarlar da şifrelenmiş olarak saklanırlar [10].

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

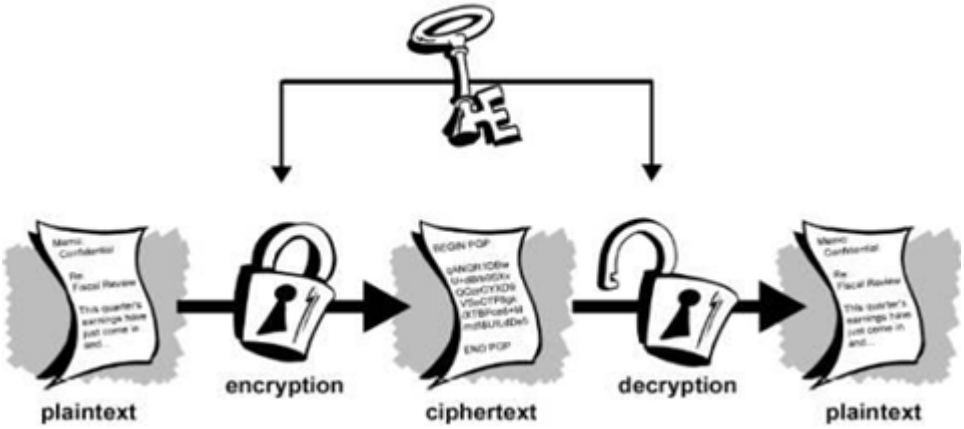
4.4 Simetrik Anahtarlı Sistemler

Genel yapısı şekil 4.1' de gösterilen simetrik anahtarlı algoritmalarda şifreleme ve şifre çözme için aynı anahtar kullanılır. Bu anahtara gizli anahtar (secret key) denir. Bu gizli anahtar iki tarafça da (gönderici ve alıcı) bilinir [10].

Projede oluşturulan devrelerde kullanılan pic mikrodenetleyicilerin haberleşmeleri sırasında şifreleme ve şifre çözme işlemlerinde kullanılan anahtar bu tip yani gizli anahtardır. Her bir pic mikrodenetleyicinin içinde aynı olan gizli anahtarlar bulunmaktadır.

Simetrik algoritmalar asimetrik algoritmalara nazaran daha hızlı çalışırlar. Bununla beraber, asimetrik algoritmalara nazaran saldırıya karşı daha az dirençlidirler. Simetrik algoritmalara örnek olarak AES, DES, 3DES, Blowfish, IDEA ve RC4 algoritmaları verilebilir [15,16].

Şifreleme ve çözücü algoritmasının anahtarları haberleşen kişiler tarafından bilinmesi gerekmektedir. Eğer şifreleme anahtarı biliniyor ise bilginin elde edilmesi oldukça kolaydır. 1970'den önceki bütün kripto sistemleri simetrikti. Bu tür sistemlere örnek verecek olursak; DES (Data Encryption Stantard) [17] ve AES [18] (Advanced Encryption Standart).



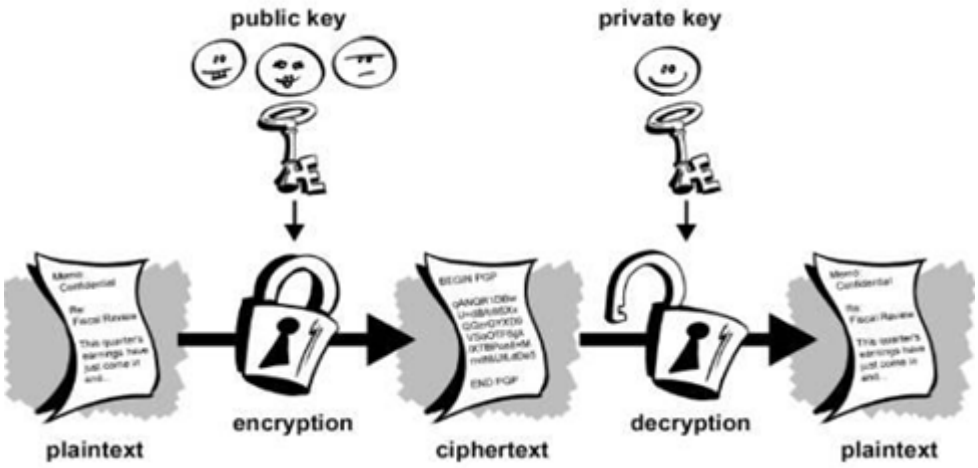
Şekil 4-1 Simetrik Anahtarlı Sistem

Burada önemli olan sorun anahtarın kimseye ulaşmadan alıcıya ulaşmasıdır [6].

Simetrik anahtarlama sistemlerinin en basit uygulamalarından biri olan yerine koyma algoritması olarak da bilinen Sezar şifresidir. Bu algoritmada bir parça bilgi, diğerinin yerine koyulur. Bu tür algoritmalar alfabe kaydırma işlemi esas alınır. Fakat bu teknolojide oldukça zayıf kalan bir algoritmadır. Simetrik anahtarlı sistemlerin avantaj ve dezavantajları şöyle sıralanabilir[10];

Avantajları;

• Çok hızlıdır. Şifrelenmiş verinin bir yere gönderilmediği durumlarda kullanışlıdır.
• Şifrelenmiş verinin gönderilmesi sırasında şifrenin gizli tutulması maliyeti artıran bir uygulamadır.
• Bu tasarımda gönderici ve alıcı bir anahtar üzerinde anlaşmalı ve bu anahtar aralarında gizli kalmalıdır.



Şekil 4-2 Asimetrik Anahtarlı Sistemler

Dezavantajları;

• Eğer kullanıcılar farklı yerlerde ise gizli anahtarın gönderilmesi sırasında hatların güvenliğinden emin olunmalıdır.
• Herhangi birisi anahtar değerini dinleyebilir ya da değiştirebilir.
• Burada ana problem anahtar dağıtma problemidir [10].

4.5 Asimetrik Anahtarlı Sistemler

Şifreleme ve şifre çözme için ayrı anahtarlar kullanılır. Bu anahtarlardan birine açık anahtar (public key), diğerine özel anahtar (private key) denir ve asimetrik anahtarlı sistemlerin genel yapısı şekil 3.11' de gösterilmiştir. Kullanılacak bu iki anahtar birlikte üretilirler. Bununla birlikte bu anahtarlardan herhangi birine sahip olan bir şahıs, diğer anahtarı üretemez, bu matematiksel olarak imkansız denebilecek derecede zordur [10].

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 [10].

Mesaj gönderebilecek kullanıcıların sayısı arttıkça, elde edilmesi gereken açık anahtar sayısı da artacaktır. Sistemde 100 kullanıcı varsa, her bir kullanıcının ayrı bir açık anahtarı olacağından, tüm bu açık anahtarlar, erişilebilir olmalıdır. Bu problem de sayısal sertifikalar teknolojisi yardımı ile çözülebilmektedir [19].

Simetrik sistemlerde özellikle geniş ağ yönetiminde anahtar kullanımı ve dağıtımı oldukça zordur. Bu sistem ile anahtar kullanımı ve anahtar yönetimi problemleri ortadan kaldırılmıştır [20].

Özel anahtarlar gizli tutularak açık anahtarlar tüm dünyayla paylaşılabilir. Açık anahtara sahip bir kişi bilgiyi sadece şifreleyebilir fakat çözemez. Yalnızca özel anahtara sahip olan kişi bilgiyi okuyabilir [10].

Asimetrik Anahtar sisteminde gönderici ve alıcının gizli anahtarları paylaşmaları gereksinimi ortadan kalkmıştır. Tüm iletişimler sadece açık anahtar üzerinden gerçekleştirilir. Özel anahtar hiç bir şekilde paylaşılmaz ya da gönderilmez. Bazı açık anahtar kripto sistemleri örnekleri Elgamal, RSA, ECC, Diffie-Hellman ve DSA' dır [10].

5. 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 [21].

5.1 Mikrodenetleyiciyi Seçim Ölçütleri

Bir uygulamaya başlamadan önce hangi firmanın ürünü kullanılacağına, daha sonra da hangi parça numaralı mikrodenetleyicinin kullanılacağına karar vermek gerekir. Bunun için mikrodenetleyici gerektiren uygulamada hangi özelliklerin olması gerektiği önceden bilinmesi gereklidir. Buna göre aşağıda sıralanan özelliklerin sistem üzerindeki gereksinimleri ve ileride yapılabilecek gelişmeleri de karşılayıp karşılamadığı araştırıldıktan sonra seçim yapılmalıdır [30]:

– Programlanabilir sayısal paralel giri /çıkı ucu sayısı
– Programlanabilir analog giriş/çıkış ucu sayısı
– Seri giriş/çıkış (senkron, asenkron ve cihaz denetimi gibi) ucu sayısı
– Analog karşılaştırıcının var olup olmadığı
– Motor veya servo kontrol için saat sinyali çıkışı
– Harici giriş vasıtasıyla kesme yapılıp yapılamayacağı
– Zamanlayıcı vasıtasıyla ile kesme yapılıp yapılamayacağı – Harici bellek arabiriminin varlığı

– Harici veriyolu arabiriminin (PC ISA gibi) varlığı
–Program belleği tipi (ROM, EPROM, PROM, FLASH ve EEPROM) ve kapasiteleri
–Program belleği üzerinde kod koruması yapılıp yapılamayacağı – Dahili RAM kapasitesi
–Dahili EEPROM' un var olup olmadığı ve kapasitesi – Reel sayı hesaplamasının varlığı
–Osilatör frekans değeri (Güç tüketiminde önemli rol oynar [30].)

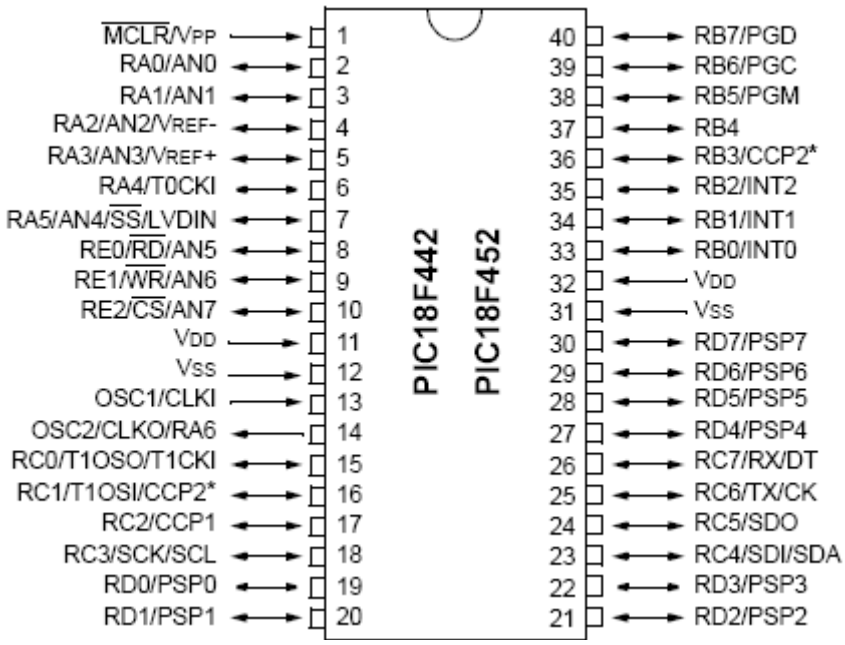
Bu listeye eklenecek özellikler artırılabilir [30].

5.2 PIC Mikrodenetleyicisinin tercih edilme nedenleri

PIC, kaynaklarına en kolay ulaşılabilen ve amatör çalışmalar için fiyatı oldukça uygun olan bir işlemcidir. Hatta profesyonel çalışmalarda bile kullanılabilecek kadar güvenilir kod korumasına sahiptir. Mesela 16F877 tipi bir işlemci flash program hafızasına sahiptir ve 1 milyondan fazla silinip tekrar programlanabilmesine karşın fiyatı sadece 3-4 TL civarındadır [30].

PIC, Harvard mimarisi temelli 8 bitlik bir mikro denetleyicidir. Bu bellek ve veri için ayrı yerleşik veri yolunun bulunduğu anlamına gelir. Böylelikle akış miktarı veriye ve program belleğine erişim sayesinde arttırılmış olur. Geleneksel mikrodenetleyicilerde veri ve programı taşıyan bir tek yerleşik veriyolu bulunur. Böylelikle, PIC' le diğer mikrodenetleyiciler karşılaştırıldığında işlem hızında en az 2 katlık performans üstünlüğü sağlanır [30].

PIC 16F877 piyasada en çok kullanılan üzerinde çok çalışılmış ve dolayısı ile kaynağı çok olan bir mikrodenetleyicidir. Tezde 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.



Şekil 5-1 PIC18F452 mikrodenetleyicinin pin diyagramı görünüsü

5.3 Pic18f452 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 [22].

5.4 PIC18F452 Mikrodenetleyicinin Temel Özelikleri

1. 32 Kbyte dogrusal program hafızası adresleme
2. 1,5 Kbyte dogrusal veri hafızası adresleme
3. 16 bit komut genisligi ve 8 bit veri genisligi
4. Osilatör frekansı 40 Mhz'dir ve 10 Mhz komut çalısma frekansı 200 ns'dir.
5. 3 harici kesme girisi

6. 8 bit Timer0 yazmacı/zamanlayıcı
7. 16 bit Timer1 yazmacı/zamanlayıcı
8. 8 bit Timer2 yazmacı/zamanlayıcı
9. 16 bit Timer3 yazmacı/zamanlayıcı
10. Çift Capture/Compare, PWM (CCP) modülü
11. Senkron seri port (SSP) ile SPITM ve I2CTM
12. Donanımsal adreslenebilir asenkron seri alıcı verici (USART)
13. 10 bit, 8 kanal analog dijital çevirici (ADC)
12. Çift analog karsılastırıcı modülü
13. Devre üzerinde seri olarak programlanabilme (ICSP)
14. Program hafızasını okuma ve yazma girisi
15. Düsük gerilim ile programlanabilme
14. Watchdog Timer (WDT)
15. Programlanabilir kod koruması
16. Güç koruma modu
17. Seçilebilir osilatör opsiyonu
18. Düsük güçlü yüksek hızlı Flash/EEPROM teknolojisi
19. Genis çalısma gerilim aralıgı (2V- 5.5V) [23].

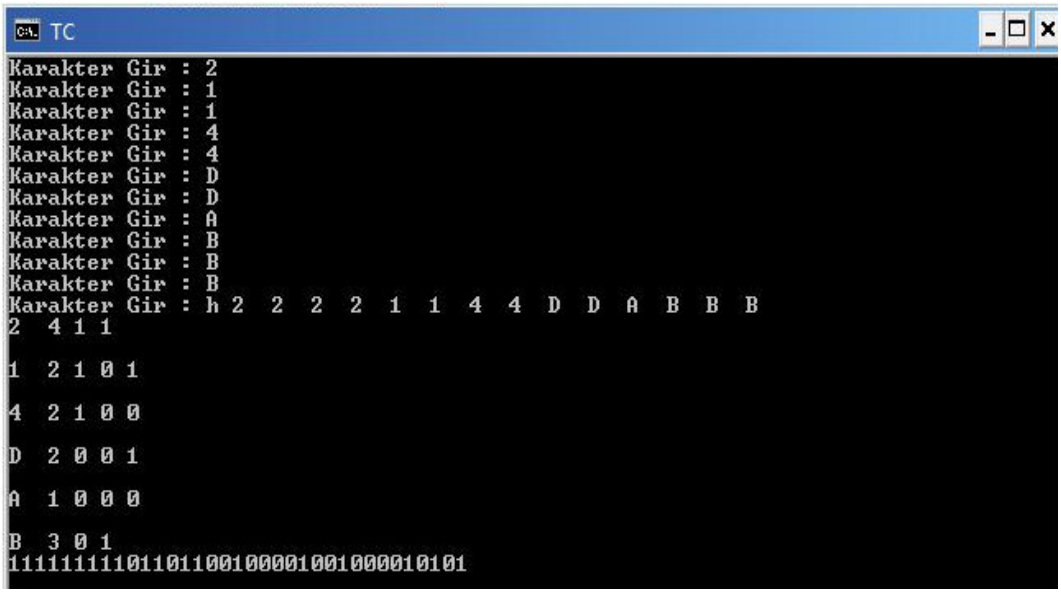
5.5 PIC18F452' yi Programlamak İçin Gerekli Araçlar

PIC18F452' yi programlamak için öncelikle yazılan programı HEX koduna çeviren derleyiciye ihtiyaç vardır. Bu tezde programı yazmak için C dili kullanıldığı için Ccs C derleyicisi tercih edilmiştir. PIC' i programlayacak olan elektronik devrenin bilgisayar ara yüzü çalıştırılır. Programcının bilgisayar arayüzüne derlenen HEX dosyası'nın yeri gösterilir. Böylelikle programlama işlemi yapılabilir. Sonuçta bir PIC' i programlamak için; program (.c), derlenmiş program (.hex), programcı ara yüzü ve programlama devresi (kartı) gerekir.

6. PICLER ARASI HIZLI VE GÜVENLİ HABERLEŞME

Tezin konusu olan robotlar arası hızlı ve güvenli iletişimi gerçekleştirmek için öncelikle hız konusu ele alınmıştır. İletilecek verinin hızlı bir şekilde iletilebilmesi için sıkıştırılması gerekmektedir. Önceki bölümlerde de bahsedildiği gibi bu sıkıştırma işlemi kayıplı veya kayıpsız veri sıkıştırma tekniklerinden biri kullanılarak yapılması gerekmektedir.

Projede verinin kayıpsız bir şekilde iletilmesini istediğimiz için kayıpsız veri sıkıştırma tekniği kullanılmıştır. Bu teknik içinde projeye en uygun algoritma olan Huffman algoritması kullanılmıştır. Girilen karakterlerin Huffman algoritmasıyla düzgün bir şekilde sıkıştırıldığını görebilmek için bu algoritma öncelikle Turbo C derleyicisinde oluşturulmuştur. İkinci bölümde Huffman algoritması oluştururken bahsedilen ağaç yapısı, C programlama dilinde pointer (işaretçi) ve structer(yapı) kullanılarak kök ve yapraklar olarak oluşturulmuştur. Şekil 6.1' deki ekran çıktısında 'h' tuşuna basılana kadar girilen karakterlerle, Huffman algoritması kullanılarak ağaç yapısı oluşturulmuştur. Böylece girilen karakterler yerine onları '0' ve/veya '1' lerden oluşan bitler temsil ederek sıkıştırma işlemi gerçekleştirilmiştir.



Şekil 6-1 Huffman Algoritmasıyla Verinin Sıkıştırılması

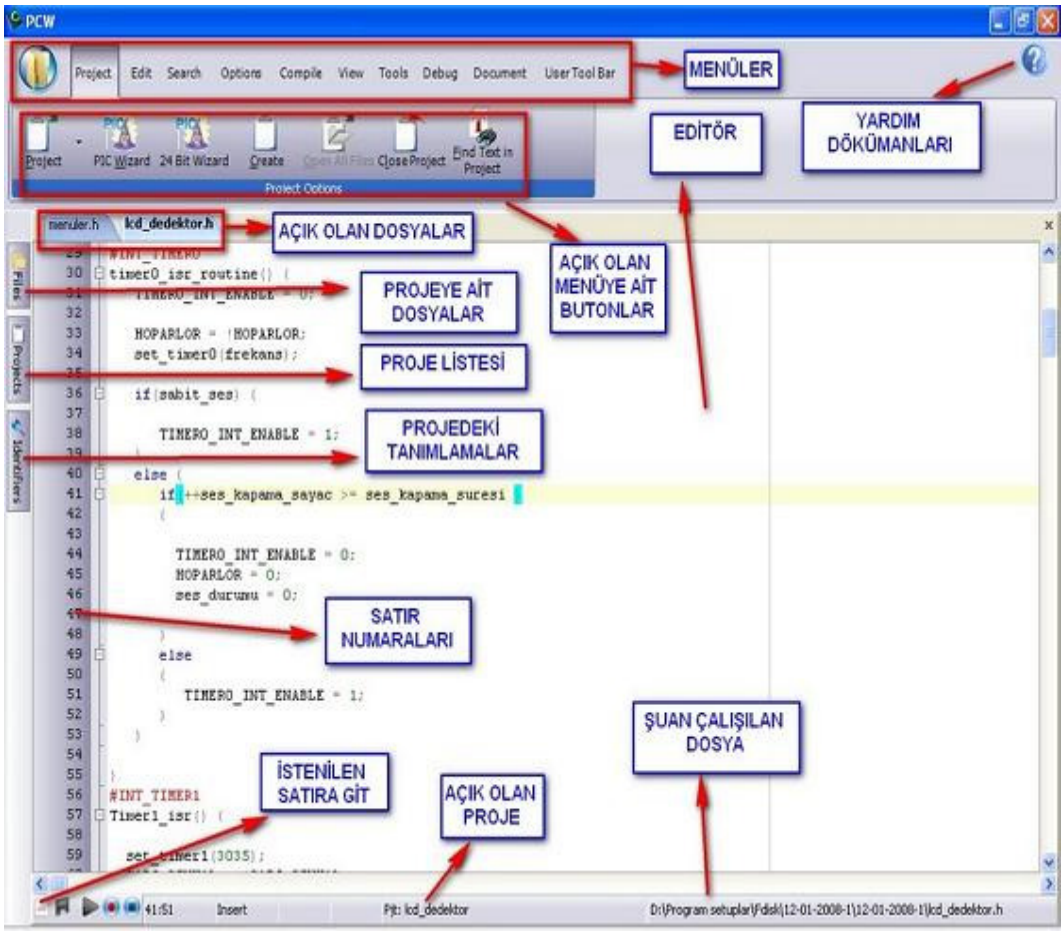
Huffman algoritmasıyla sıkıştırdığımız verilerin, güvenliğini arttırmak, veri iletimi sırasında istenmeyen kişilerin eline geçmesini engellemek için şifreleme işlemini gerçekleştirmek gerekmektedir. Ancak bu projede amaç veriyi sadece sıkıştırıp şifrelemek değil, veriyi gönderdikten sonra veriyi alan tarafın bu veriyi orijinal haline getirebilmesini de sağlamaktır. Bunu sağlayabilmek için haberleşen her iki tarafın da bildiği gizli bir bilgi olması lazım. İşte şifreleme tekniklerinde kullanılan anahtar yöntemlerinden birinin burada kullanılması gerekmektedir. Bu anahtarlar kendi algoritmamız ile oluşturulan ve hiçbir şekilde paylaşılmayan yani 'private key' olarak isimlendirilen anahtarlardır. Şifreleme işlemi için ise bu anahtar ile sıkıştırılan veri XOR şifreleme yöntemiyle işleme sokulur. Bu şekilde veri daha kısa ve şifrelenmiş hale gelir. Şekil 6.2' de girilen verilerin sıkıştırmış hali ve şifrelenmiş hali verilmektedir.

Şekil 6-2 Private Key Kulllanarak Xor yöntemiyle Sıkıştırılmış Verinin Şifrelenmesi

Turbo C derleyicisinde gerçekleştirilen bu uygulamalar Huffman algoritmasının ve XOR şifreleme yönteminin düzgün ve doğru bir şekilde çalışıp çalışmadığını kontrol etmek amacıyla yapılmış uygulamalardı. Projedeki amaç mikrodenetleyiciler arası veri haberleşmesi olduğu için, uygulama mikrodenetleyicinin programlanabileceği bir derleyiciye aktarılması gerekmektedir. Turbo C derleyicisiyle mikrodenetleyiciler programlanamadığı için, CCS firmasının PIC mikrodenetleyicilerin C dilinde programlanabilmesini sağlamak amacıyla ürettiği CCS C isminde C derleyicisi kullanılmıştır.

6.1.1 Ccs C Derleyicisi

CCS-C derleyicisi PIC10, PIC12, PIC14, PIC16, PIC18, PIC24 ve dsPIC serilerini desteklemektedir. CCS-C derleyicisinde 1Bit, 8 Bit, 16 Bit ve 32 bit tamsayı değişkenler ve 32 bit virgüllü sayı değişkenleri kullanılabilmektedir. #byte ve #bit önişlemci direktifleri sayesinde 1 Byte veya 1 Bitlik değişkenler PIC içerisindeki saklayıcılara bağlanabilir [25].

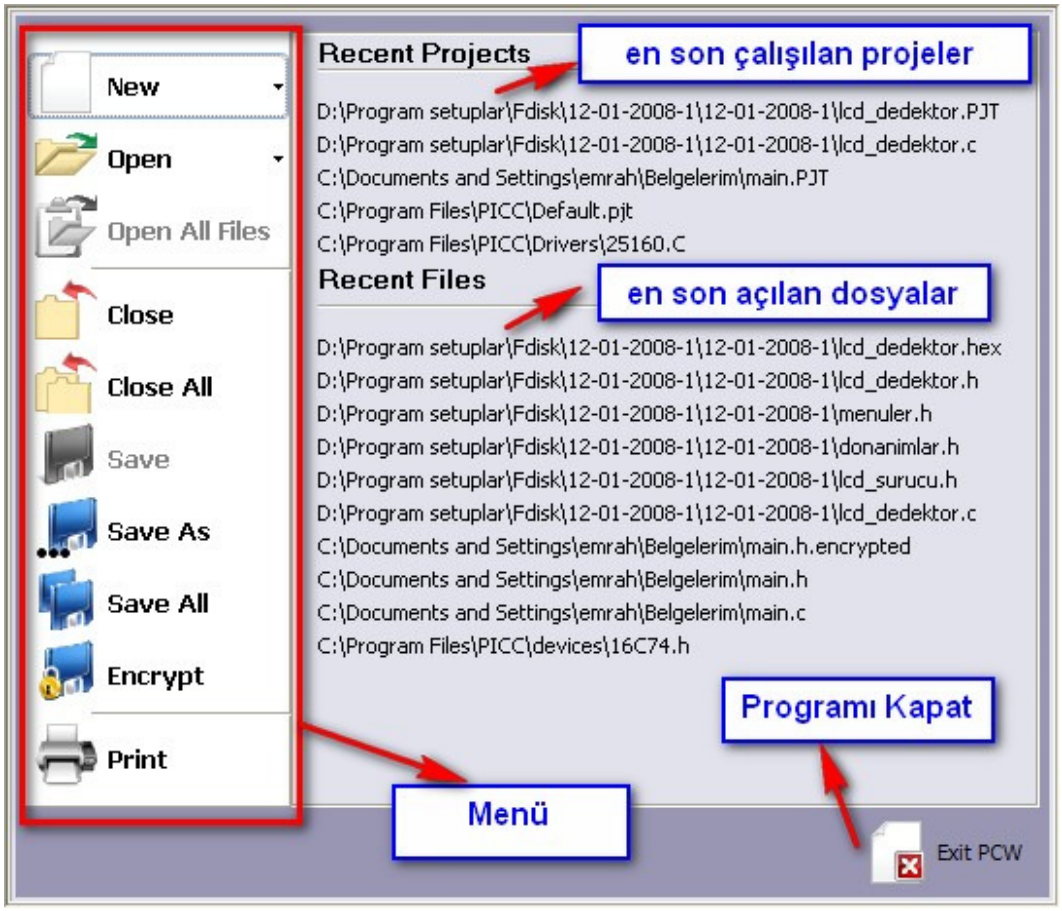


Şekil 6-3 CCS-C Programının ana görüntüsü

6.1.2 Ccs-C Ide Arabirim Tanıtımı

6.1.2.1 Ana Ekran

Ana ekranla ilgili açıklamalar Şekil 6.3 üzerinde yapılmıştır.



Şekil 6-4 CCS-C Programının Dosya Menüsü

6.1.2.2 Dosya Menüsü

New->;Source File: Yeni bir kaynak (*.c,*.h) kod dosyası oluştur

New->;Project Wizard: Otomatik proje oluşturma sihirbazı

New->;Project Manual: Manuel proje oluşturma

New->;RTF File: Yeni RTF Dosyası oluştur (Zengin Mtin Dosyası), RTF Editörü açılır.

New->;Flow Chart: Akış diagramı oluştur, Akış diagramı editörü açılır

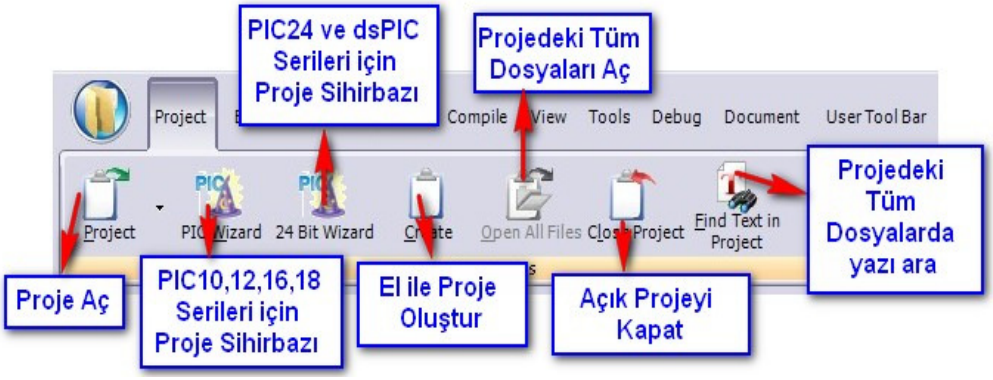
Open->;Any File: Herhangi bir dosya aç

Open->;Source File: Kaynak dosyası aç

Open->;Project: Proje Aç

Open->;Output File: Proje Çıkış dosyası aç

Open->;As Hex File: Hex dosayı açar



Şekil 6-5 CCS-C Programının Proje Menüsü (Project)

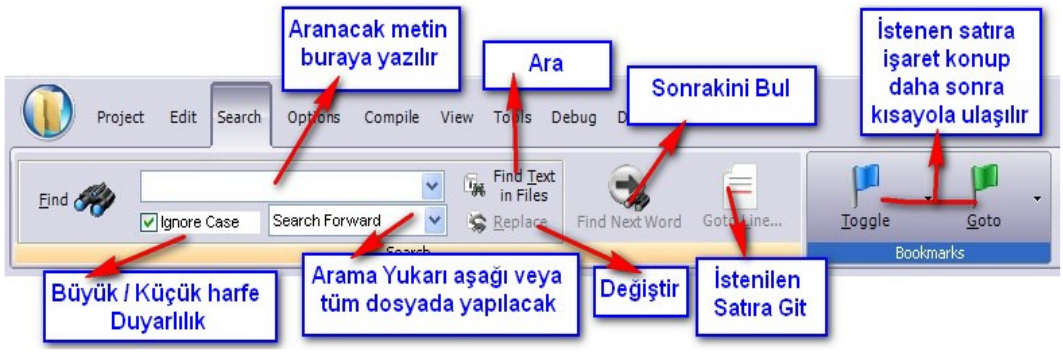
Şekil 6-6 CCS-C Programının Düzenleme Menüsü (Edit)

6.1.2.3 Proje Menüsü

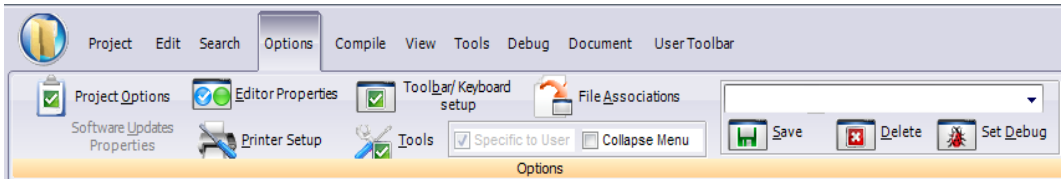
Şekil 6.5' teki proje menüsünü kullanarak, yapabilecekler şunlar. Varolan projeleri açmak, Yeni bir proje sihirbazı başlatmak, manuel proje oluşturmak, projeyi kapatmak, projedeki tüm dökümanları açmak ve proje dosyalarının tümünde yazı aramak.

6.1.2.4 Düzenleme Menüsü

Şekil 6.6' daki düzenleme menüsü olan menüde Kes, Kopyala, Yapıştır, Tümünü Seç gibi tüm programlardan alışık olunan menü vardır.



Şekil 6-7 CCS-C Programının Arama, Değiştirme Menüsü (Search)



Şekil 6-8 CCS-C Programının Ayarlar Menüsü (Options)

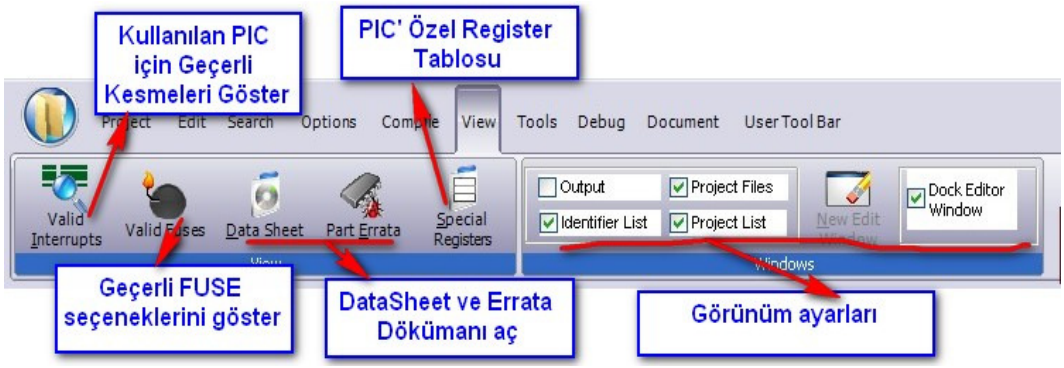
6.1.2.5 Arama, Değiştirme Menüsü

Şekil 6.7' deki menüyü kullanarak kodlar içerisinde arama ve değiştirme yapılabilir, satır numarası girerek kod içerisinde istenilen satıra gidilebilir. Bu menüden ve normal klavye kısa yollarıyla kullanabilecek bir diğer özellik ise yer imi oluşturma ve yer imine gitme. Yani istenilen satırlara işaretler koyup sonra bu işaretler arasında dolaşılabilir. 10 adet yer imi kullanılabilir, yer imi koymak için Shift+Ctrl+0…9, yer imine gitmek için Ctrl+0…9. 0…9 rakamları için harflerin üstündeki sayılar sırası kullanılır.

6.1.2.6 Ayarlar Menüsü

Şekil 6.8' deki ayarlar menüsünü kullanarak proje ile ilgili ayarlamalar yapılabilir, programın genel görünümü ve kod editöründe kullanılan yazı tip ve renk paletleri değiştirilebilir. Yine buradan kullanıcı araç çubuğundaki komutlar düzenlenebilir ve istenilen komuta klavye kısa yolu eklenebilir.

Şekil 6-9 CCS-C Programının Derleme Menüsü (Compile)



Şekil 6-10 CCS-C Programının Görünüm Menüsü (View)

6.1.2.7 Derleme Menüsü

Şekil 6.9' da ki derleme menüsünde yazılan kodu derlemek, projeyle ilgili tüm çıkış dosyalarını üretmek, çipe programı yüklemek, donanımsal hata ayıklayıcıyı başlatmak ve proje için oluşturulan çıktı dosyalarına bakmak için bu menü kullanılabilir. Örneğin yazılan C koduna karşılık üretilen ASM kodları merak edilirse "C/ASM List" butonuna tıklamak yeterli olacaktır.

6.1.2.8 Görünüm Menüsü

Şekil 6.10' daki görünüm menüsünde ise kullanacak PIC ile ilgili faydalı bilgilere ulaşılabilir. Örneğin "Valid Interrupts" butonuna basıldığında açılan pencerede kullanmak istenilen PIC' i seçerek hangi Interrupt (Kesme) ların bu PIC ile kullanılabileceği görülebilir. Aynı şey FUSE ayarları için de geçerli. Ana ekranda solda görünen üç adet butonun ekranda görünme ayarı da yine bu menüden yapılabilecekler arasında. Butonları kaybedildiğinde eğer ihtiyaç duyulursa buradan görünmesi sağlanabilir. Bu menüdeki bir diğer faydalı özellik ise "Special Registers" butonuna basıldığında açılan ekran. Bu ekranda istenilen PIC' e ait register' lara ve bu register' ların adreslerine ulaşılabilir.

6.1.2.9 Araçlar Menüsü

Araçlar menüsünde ihtiyaç duyulabilecek yardımcı yazılımlar vardır. Açıklamalar Şekil 6.11 üzerinde mevcuttur.

6.1.2.10 Araçlar Menüsü

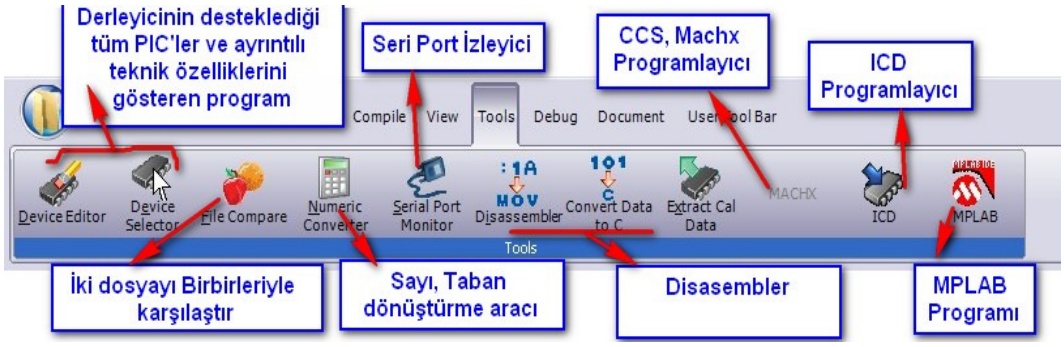
Eğer ICD programlayıcısı var ise, Şekil 6.11 yardımıyla Donanımsal Hata Ayıklama işlevini gerçekleştirilebilir. Normal yazılımsal simulasyonlar da program simulator tarafından işletilerek hata aranmaya çalışılır. Donanımsal hata ayıklamada ise program PIC de çalıştırılır, böylece daha doğru, sonuçlar daha pratik bir şekilde elde edilir.

6.1.2.11 Belge Menüsü

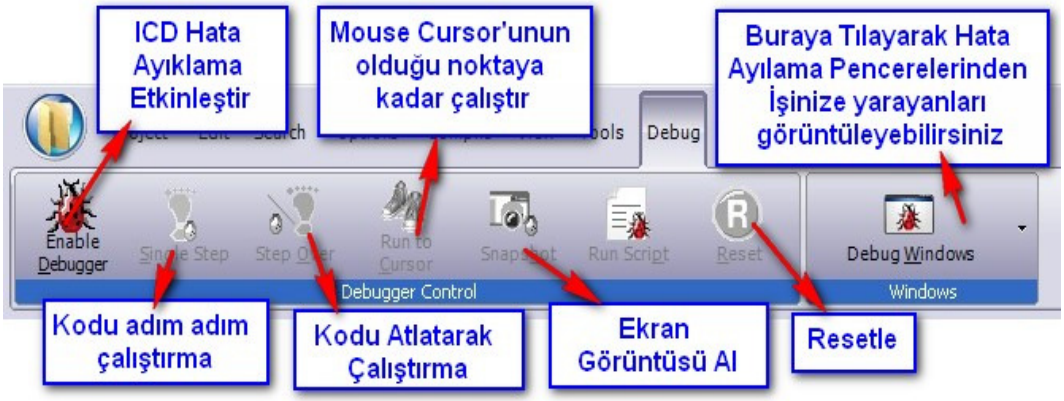
Şekil 6.13 proje ile ilgili dokümantasyon oluşturulabileceği, metin editörü ve akış diyagramı editörüne de kısa yollar içeren bir menüdür.

6.1.2.12 Kullanıcı Araç Çubuğu Menüsü

Şekil 6.12' deki menü keyfe göre düzenleyenebilecek bir menü, burada yer alacak butonlar, Ayarlar menüsünden "Toolbar/Keyboard Setup" kısmına girerek düzenlenebilir.



Şekil 6-11 CCS-C Programının Araçlar Menüsü (Tools)



Şekil 6-12 CCS-C Programının Hata Ayıklama Menüsü (Debug)

Şekil 6-13 CCS-C Programının Belge Menüsü (Document)

Şekil 6-14 CCS-C Kullanıcı Araç Çubuğu (User Tool Bar)

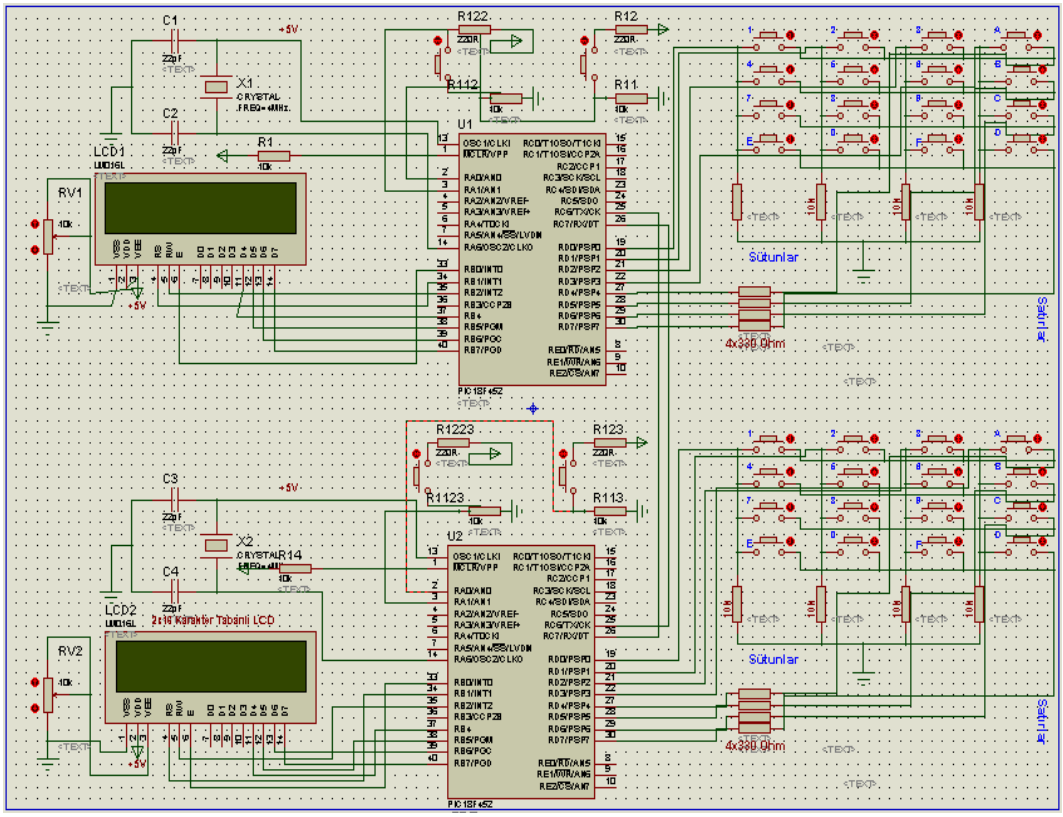
6.2 Proteus ile Devre Şemasının Oluşturulması

Bir üst konuda Ccs c derleyicisinden bahsedildi. Bu derleyicide kodları oluşturabilmek 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 simulasyonunu yapabilen yetenekli bir devre çizimi, simulasyonu, 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 şeması Proteus yardımıyla bilgisayar ortamında denenebilir. Devredeki elemanların değerleri değiştirilip yeniden çalıştırır ve sonuç gözlemlenebilir. Bu program, binlerce elektronik eleman içeren devre tasarımlarının üretiminde bile kullanılabilmektedir. Elektriksel hata raporu hazırlayabilmekte, malzeme listesini çok düzenli bir şekilde verebilmektedir [26].

.



Şekil 6-15 ISIS Programı ile Çizilmiş Haberleşme Devresi

PROTEUS programı ISIS ve ARES olmak üzere iki alt programdan oluşur. ISIS' ta elektronik devre çizimi gerçekleştirilirken, bunun yanında devrenin analizi de yapılabilmektedir [30]. Yukarıda bahsedilen sanal ortamda oluşturulan elektronik devre ISIS ile gerçekleştirilmektedir.

Şekil 6.15' te ki devrede 2 adet PIC18F452 mikrodenetleyici, 2 adet 2x16 Karakter Tabanlı LCD, 2 adet 4x4 (16 adet butondan oluşturulmuş) keypad, 2 adet girilen karakterlerin sıkıştırılıp şifrelenmesini sağlayan buton, 2 adet haberleşmeyi başlatmayı sağlayan buton ve devrenin elektrik akımını sağlayan diğer elemanlar yer almaktadır.

Projede PIC18F452 kullanılmasının nedeni PIC16F877' de yaşanan hafıza yetersizliği ve bacak sayısının PIC16F877 ile aynı olmasıdır. Şekil 6.15' te oluşturulan devre çalıştırıldığında şu sonuç elde edilmeli. Hexadecimal sayı formatı şeklinde oluşturulmuş 16 adet butonlara basılır. Tabi bu botonlara rastgele basılmayacak. Diğer mikrodenetleyiciye gönderilmek istenilen mesajın karşılığı olan buton değerleri girilecek. Veri girme işlemi tamamlandıktan sonra verinin sıkıştırılıp şifrelenmesi işlemini yapmak için RA0 pini lojik-1 olması gerekmektedir. Veri sıkıştırılıp şifrelendikten sonra diğer PIC18F452 mikrodenetleyicisine gönderilmesi gerekmektedir. PIC' ler arası haberleşme için çeşitli yollar vardır. Bu uygulamada rs232 portu üzerinden haberleşilecektir. Verinin gönderilebilmesi için RA1 pini lojik-1 yapılmalıdır.

.

Devrenin çalışma prensibi genel hatlarıyla bu şekilde olması gerekmektedir. Haberleşme sırasında, verinin diğer mikrodenetleyicide anlamlı hale yani orijinal hale dönüştürülmesi için hangi verilerin gönderildiğini programın nasıl kodlandığının anlatıldığı konu başlığı altında verilmektedir.

6.3 Ccs C İle Pic18f452 Mikrodenetleyicilerin Programlanması

Proteus ISIS programıyla oluşturulan devrenin istenilen biçimde çalışabilmesini sağlamak için PIC18F452 mikrodenetleyicilerin programlanması gerekmektedir. Bunun için de Ccs C derleyicisi kullanılmıştır. Bölüm 6.1' de bu derleyiciden bahsedilmiştir.

Öncelikle kodlamayı yaparken PIC18f452' nin bacaklarına bağlanan elemanlara göre pinlerin ilk durumları belirlenmiştir;

set_tris_b(0x00) // B portu komple çıkış
set_tris_d(0x0F) // 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 [27]. *set_tris_d(0x0F)* komutu ile yüksek değerlikli 4 bit çıkış, düşük değerlikli 4 bit giriş olarak kullanılacağını bildirir.

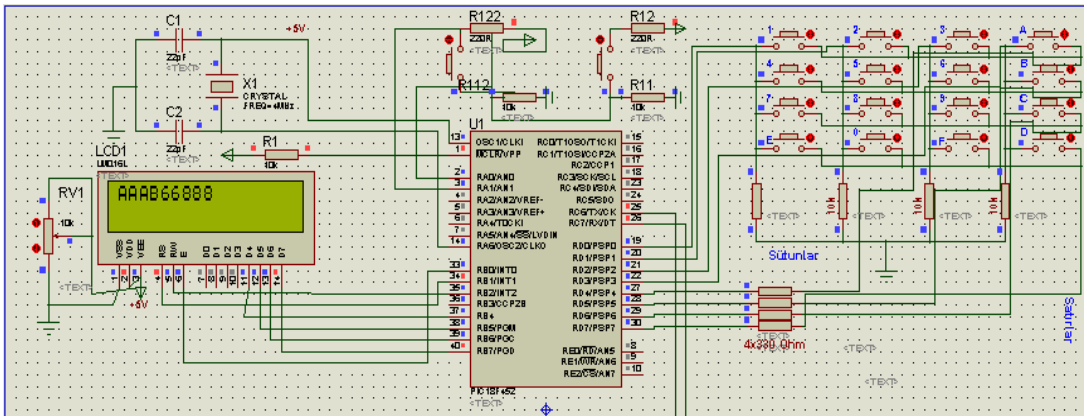
Tasarlanan devrede ilk durumda D portuna bağlı butonların hiç birisi basılı halde değil. Bu durumda D portu çıkışı *output_d(0x00)* komutuyla sıfırlanması gerekmektedir. Herhangi bir tuşa basıldığında hangi tuşa basıldığını anlamak için 4x4 şeklinde satır, sütun yapısı oluşturan butonlar sırasıyla kontrol edilir. Bunu yapabilmek için sırasıyla öncelikle satırları yani düşük değerlikli bitleri output_high(pin_dx) ile lojik-1 yapılır. Devamında lojik-1 yapılan 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 belirlenen 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>*

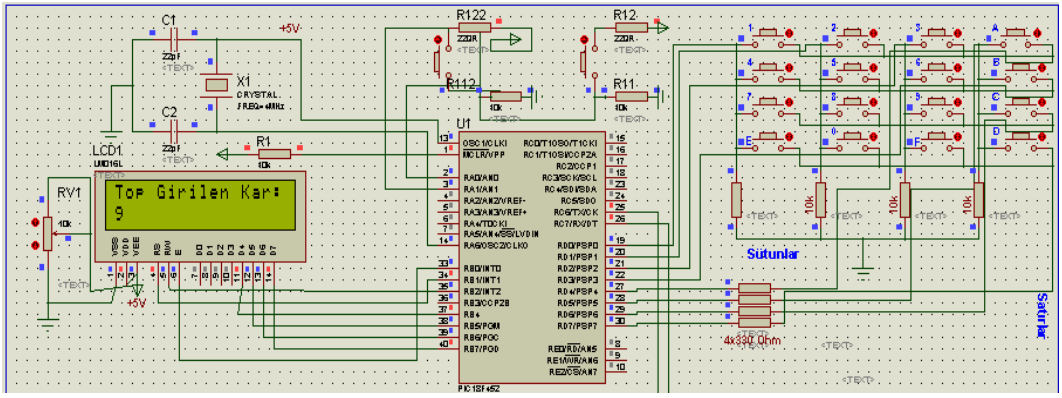
komutu eklenmektedir. *printf(lcd_putc,"%c",keypad_oku())* komutu ile keypad_oku() fonksiyonundan dönen buton değeri lcd ekranına yazdırılır.

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.

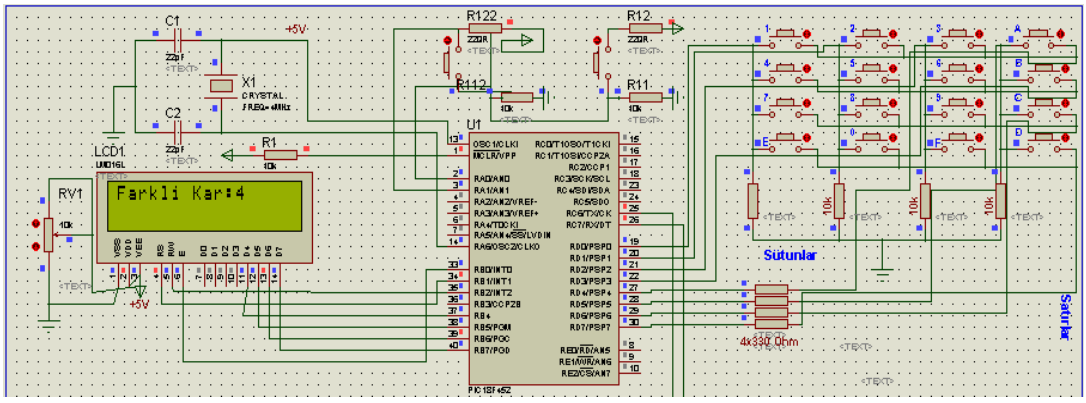
Aşağıdaki şekiller de yukarıdaki paragrafta bahsedilen adımlar bir örnekle gerçekleştirilmiştir. Şekil 6.16' da girilen tüm karakterler görülmektedir. Şekil 6.17' de girilen karakterlerin toplam sayısı görülmektedir. Şekil 6.18' de kaç farklı karakter girildiği ekranda belirtilmektedir. Şekil 6.19' da girilen karakterlerin kaç adet girildiği görülmektedir. Şekil 6.20' de girilen karakterlerin yerine kullanılacak bitler görülmektedir. Şekil 6.21' de girilen karakterlerin Huffman algoritmasıyla sıkıştırılmış halinin lcd ekranında görüntülenmesi sağlanmaktadır. Şekil 6.22 girilen karakterlerin Huffman algoritmasıyla sıkıştırılmış veri ile belli bir algoritmayla oluşturulan anahtarın XOR ile şifrelenmiş halinin ekrana yazdırılması görülmektedir.



Şekil 6-16 Tuş Takımından Girilen Karakterlerin Ekrana Yazdırılması



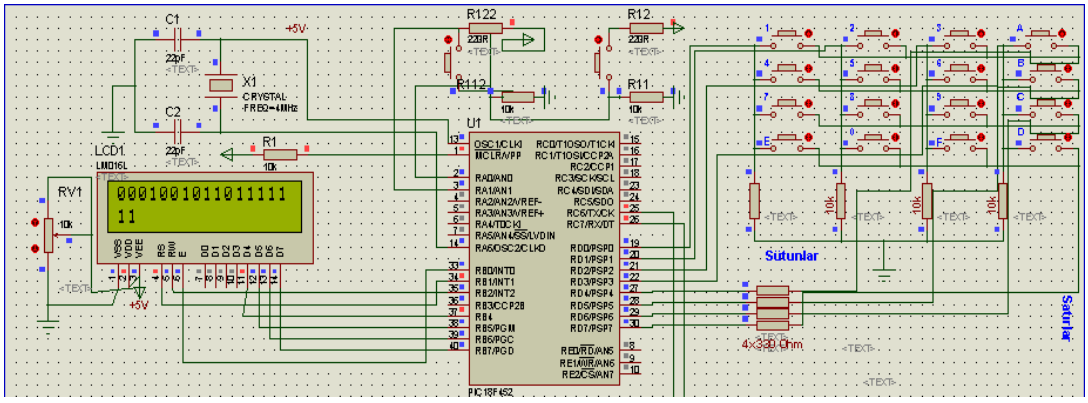
Şekil 6-17 Tuş Takımından Girilen Toplam Karakterlerin Sayısının Yazdırılması



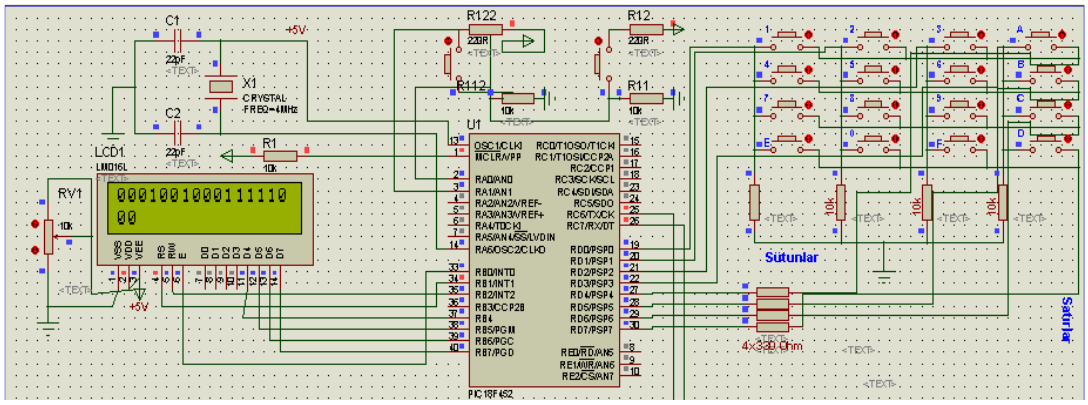
Şekil 6-18 Tuş Takımından Girilen Farklı Karakterlerin Sayısının Yazdırılması

Şekil 6-19 Tuş Takımından Girilen Karakterlerin Kaç Adet Girildiğinin Yazdırılması

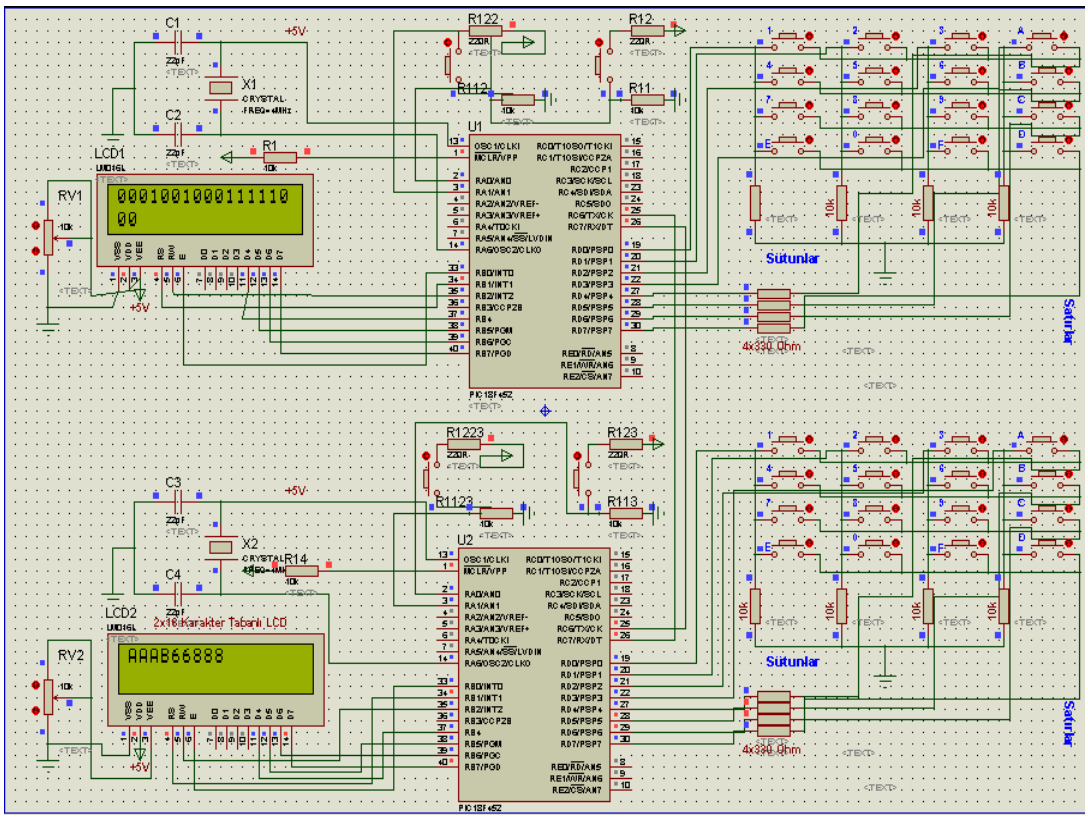
Şekil 6-20 Tuş Takımından Girilen Karakterlerin Yerine Kullanılacak Bitlerin Ekrana Yazdırılması



Şekil 6-21 Tuş Takımından Girilen Karakterlerin Huffman Algoritmasıyla Sıkıştırılmış Halinin Ekrana Yazdırılması



Şekil 6-22 Tuş Takımından Girilen Karakterlerin Huffman Algoritmasıyla Sıkıştırılmış Veri ile Anahtarın XOR ile Şifrelenmiş Halinin Ekrana Yazdırılması



Şekil 6-23 Şifreli Şekilde Gönderilen Verinin Orijinal Haline Çevrilmesi

Şekil 6.23' 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 6.23' de üstteki lcd ekranında verinin sıkıştırılmış ve şifrelenmiş hali alta ki lcd ekranında da verinin orijinal hali görülmektedir.

Öncelikle bu durum açıklanmadan önce verileri gönderilirken Ccs-c' de seçilen haberleşme birimini yani rs232 hakkında bilgi verilmektedir.

6.3.1 Ccs C İle Rs232 Seri İletişim

"*CCS C derleyicisi içinde RS232 seri iletişimi kolayca kontrol etmeyi sağlayan fonksiyonlar bulunmaktadır. CCS C programı hem donanımsal hem de yazılımsal RS232 seri iletişimi destekler. Yani CCS C ile ister içinde UART donanım modulünü barındıran denetleyiciler ile donanımsal veya yazılımsal, isterse de içinde UART donanım modulü bulunmayan denetleyiciler ile yazılımsal RS232 seri iletişim kurulabilir. Böylece içinde gerekli donanımı olan veya olmayan denetleyicilerde istenen pinlerle kolayca RS232 seri iletişim sağlanabilir. Fakat RS232 seri iletişim kesmeleri donanımsal pinlerin kullanılmadığı yazılımsal RS232 seri iletişimlerde çalışmamaktadır. Bazı PIC denetleyicilerde birden fazla UART modulü (UART1,UART2) bulunmaktadır [27,s.432]."*

6.3.1.1 #Use Rs232() Fonksiyonu

"Bu komut derleyiciye hangi pinlerin seri iletişim için kullanılacağını, seri iletişim hızının ne olacağını, iletişim için parity bitinin ne olacağı gibi seri iletişim ile ilgili ayarları derleyiciye bildiren ön işlemci komutudur. Fonksiyondaki "sabitler" kısmına aşağıda verilen sabitlerden kullanılması gerekenler yazılır. Her sabit virgül ile birbirinden ayrılır. PIC18F452' nin içinde Rs232 iletişimi için UART donanım modulü vardır. Bu modulü kullanarak Rs232 iletişimi belli ve sabit pinlerle yapılır. Fakat CCS C derleyicisi istenilen pinleri kullanarak Rs232 iletişimi de sağlar [27,s.432]."

#use rs232(sabit,sabit,sabit,…) [27,s.432]

#USE RS232 Sabitleri

*"**baud=x** RS232 iletişim hızını belirten sabittir. (baud=9600 gibi)*
***xmit=pin** RS232 veri gönderme ucunu belirten sabittir. (xmit=pin_A2 gibi)*
***rcv=pin** RS232 veri alma ucunu belirten sabittir. (rcv=pin_A3 gibi)*
***parity=x** RS232 iletişimi eşlik biti durumunu bildirir. X yerine N (eşlik biti yok), E (eşli biti çift), O (eşlik biti tek) yazılabilir. (parity=N gibi)*
***stop=x** RS232 stop bitinin kaç bit olduğunu belirtir. X yerine 1 veya 2 yazılır. Varsayılan stop biti 1'dir. (stop=1 gibi)*
***invert** RS232 iletişim pinlerinin polaritesini değiştirir. Yani xmit ile rcv pinleri yer değiştirir.*
***ENABLE=pin** Belirtilen pin veri gönderme boyunca lojik-1 olur. RS485 iletişimde veri göndermeyi başlatma ucu olarak kullanılabilir. (enable=pin_B2)*
***FORCE_SW** Yazılımsal RS232 iletişimin kullanılacağını belirtir. Bu parametre seçildiğinde, iletişim uçları olarak donanımsal uçlar seçilse dahi iletişim yazılımsal olarak yapılır.*
***UART1** xmit ve rcv pinleri olarak denetleyici içinde bulunan birinci UART modulü pinlerinin kullanılacağını belirtir.*
***UART2** xmit ve rcv pinleri olarak denetleyici içinde bulunan ikinci UART modulü pinlerinin kullanılacağını belirtir.*
***STREAM=isim** #use rs232() fonksiyonu ile yapılan ayarların belli bir isim altında kaydedilmesini sağlar. Bu sayede program başında birden fazla #use rs232() fonksiyonu tanıtılarak, değişik iletişim ayarlarında istenen sayıda RS232 iletişim protokolü belirlenir. Program içinde RS232 iletişimi komutlarını kullanırken gönderilecek veya alınacak verinin hangi RS232 iletişim ayarlarında alınıp veya verileceği seçilebilinir.*
***RESTART_WDT** getc() fonksiyonu veri okumayı beklerken WDT' ı sıfırlayarak, WDT' ın denetleyiciyi resetlemesini önler* [27,s.432].*"*

6.3.1.2 Set_Uart_Speed() Fonksiyonu

"Bu fonksiyon çalışma anında RS232 iletişim hızını (baud rate) değiştirme işini sağlayan fonksiyondur. Bu fonksiyon sadece UART donanım birimi barından denetleyicilerde geçerlidir [27,s.433].*"*

set_uart_speed(baud hızı)

set_uart_speed(baud hızı, STREAM ismi)

```
set_uart_speed(9600);    //Seri iletişim baud hızı 9600
set_uart_speed(19200);  // Seri iletişim baud hızı 19200
set_uart_speed(2400,seri_1);  // seri_1 ile kaydedilen seri iletişim baud
                                // hızı 2400 olarak değişir
```
[27,s.433].

6.3.1.3 Rs232 Seri İletişim Kesmeleri (#Int_Rda)

"#INT_RDA kesmesi seri denetleyicinin seri haberleşme veri alma ucuna (RX) bilgi geldiği zaman aktif olur. Kesme fonksiyonu çıkışında, kesme pasif hale getirilmelidir. Aksi takdirde program hep kesmeye gidecektir. Bazı denetleyicilerde INT_RDA0, INT_RDA1, INT_RDA2 kesmeleri de bulunmaktadır. Aynı zamanda gönderme tamponu boş olduğunda meydana gelen #INT_TBE (Transmit Buffer Empty) kesmesi de mevcuttur [27,s.434]*."*

*"**NOT:** Rs232 kesmesi seri iletişim için PIC' in kendi donanımsal uçları kullanılırsa geçerlidir. İletişim için diğer uçlar kullanıldığında çalışmaz* [27,s.434]*."*

6.3.1.4 Rs232 Giriş/Çıkış Fonksiyonları

"RS232 iletişimi sırasında karakter alımı – gönderimi, string (ifade) alımı – gönderimi gibi işlemlerin yapılmasını sağlayan fonksiyonlardır. Bu fonksiyonların kullanılabilmesi için programın başında #use rs232 ön işlemci fonksiyonunun tanıtılması gerekmektedir. Temel C dilinde olan ve CCS C' de kullanılan Giriş/Çıkış fonksiyonları Tablo 6.1' de verilmiştir [27,s.434]*."*

- ***GETC(), GETCH(), GETCHAR() FGETCH()***

"Bu fonksiyonlar RS232 ara yüzünün alma bitinden (RX) bir karakterin gelmesini bekler, karakter geldiğinde o karakterin değeri ile geri döner. Geri dönüş değeri 8 bitlik değerdir [27,s.435]*."*

değişken ismi = getc();
değişken ismi = getch();
değişken ismi = getchar();
değişken ismi = fgetch(STREAM ismi) [27,s.435]*;*

- ***GETS(), FGETS()***

"Bu fonksiyon string ifadelerin RS232 üzerinden alınmasını sağlar. [27,s.435]."

gets(değişken ismi);

fgets(değişken ismi,STREAM ismi) [27,s.435];

Tablo 6-1 CCS C derleyicisi RS232 Giriş/Çıkış fonsiyonları [27,s.434]

Komut	Açıklama
getc(), getch(), getchar(), fgetc()	*RS232 iletişiminde RX pininden bir karakter gelmesini bekler. Karakter geldiğinde gelen karakter değeri ile geri döner.*
gets(), fgets()	*RS232 iletişimde RX pini üzerinden string ifade alımı için kullanılır. Enter tuşuna basıldığında o ana kadar yazılan string değeri ile geri döner.*
putc(), putchar(), fputc()	*RS232 iletişimde TX pini üzerinden bir karakter gönderir.*
puts(), fputs()	*RS232 iletişimde TX pini üzerinden string ifade gönderir.*
printf(),fprintf()	*RS232 iletişimde TX pini üzerinden istenilen karakter veya string ifadelerin belirli bir formatta gönderilmesini sağlar.*
kbhit()	*RS232 iletişimi veri alma kontrol fonksiyonu*

Örnek:

#use rs232 (baud=9600, xmit=PIN_C6, rcv=PIN_C7, stop=1, parity=N)

```
char sifre[20];
printf("Sifre Gir:      ");
gets(sifre);  //Girilen string değer RX pini üzerinden
              //alınarak "şifre" değişkenine atanıyor
```
[27,s.435]

- ***PUTC(), PUTCHAR(), FPUTC()***

"Bu fonksiyonlar, RS232 gönderme biti üzerinden (TX) bir karakter gönderme işini yaparlar. Fonksiyonlardaki "data" kısmı 8 bitliktir [27,s.436]*."*

putc(data);
putchar(data);
fputc(data,STREAM ismi) [27,s.436]*;"*

Örnek:

```
#use rs232 (baud=9600, xmit=PIN_C6, rcv=PIN_C7, stop=1, parity=N)

char k;
putc('s');      // 's' karakteri TX pini üzerinden gönderilir.
putchar('s'); // 's' karakteri TX pini üzerinden gönderilir.

#use rs232 (baud=2400, xmit=PIN_B3, rcv=PIN_B4, stop=1, parity=N,
STREAM=iletisim_1)
fputc('s',iletisim_1)  // 's' karakteri TX pini üzerinden
                       //iletisim_1 ayarlarına göre gönderilir [27,s.436].
```

- ***PUTS(), FPUTS()***

"RS232 üzerinden string ifadelerin gönderilmesini sağlar. String ifade fonksiyon içinde çift tırnak içinde (" ") gösterilir. Ayrıca string ifade olarak bir string dizisi ismi yazılabilir. Böylece o dizinin tüm elemanları gönderilmiş olunur [27,s.436]*."*

puts(string);
fputs(string,STREAM ismi) [27,s.436];

Örnek:

```
#use rs232 (baud=9600, xmit=PIN_C6, rcv=PIN_C7, stop=1, parity=N)
char data[]="ccs c";
puts("---Menu---"); //"---Menu---" string ifadesi TX pini üzerinden
gönderilir.
puts(data);   //data dizisi içeriği (ccs c) TX pini üzrinden gönderilir.

#use rs232 (baud=2400, xmit=PIN_B3, rcv=PIN_B4, stop=1, parity=N,
STREAM=iletisim_1)
fputs("---Menu---",iletisim_1)   //"---Menu---" string ifadesi TX pini
üzerinden iletisim_1 ayarlarına göre gönderilir [27,s.437].
```

- ***PRINTF(), FPRINTF()***

"Bu fonksiyon string ifadelerin belli bir formatta gönderilmesi veya fonksiyonların belirli bir formatta çıktı vermesini sağlar. Fonksiyon genel olarak aşağıda verilen formatlarda kullanılır [27,s.437]*."*

printf(string);
printf(string,değişken/değişkenler);
printf(fonksiyon ismi,string,değişken/değişkenler);
printf(STREAM ismi,string,değişken/değişkenler [27,s.437];

Örnek:

#use rs232 (baud=9600, xmit=PIN_C6, rcv=PIN_C7, stop=1, parity=N)

printf("Sifreyi giriniz="); //"Şifreyi giriniz=" string ifadesi
//TX pini üzerinden gönderilir
printf("\nSicaklik %f C",th); //Bu komutla %d sabiti yerine ondalıklı tipte gösterilmek üzere "th" değişkeni değeri yazılır. Bir satır atlanır (\n parametesi sayesinde) [27,s.437].

CCS C ile RS232 seri iletişim hakkında detaylı bilgiler verildi. Bu konudan bahsedilmesinin nedeni uygulamada PIC18F452 mikrodenetleyicilerinin haberleşmesi için RS232 seri iletişimi kullanılmasındandır. PIC18F452 donanımsal olarak UART donanım modulünü barındırmaktadır. Seri iletişim sırasında veri alma ve gönderme işlemleri için RC6/TX ve RC7/RX pini mevcuttur. Diğer pinlerin de seri iletişim için kullanılabildiği bir önceki başlıklarda belirtildi. Fakat uygulamada #INT_RDA seri iletişim kesmesi kullanıldığı için seri iletişimde RC6/TX ve RC7/RX pinleri kullanılmak zorundadır. Bu koşullar dikkate alınarak CCS C derleyicisine aşağıdaki kod eklenmiştir.

#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7, parity=N)

CCS C ile RS232 seri iletişim konusunu ele alınmadan önce ilk mikrodenetleyiciye girilen verinin, o veri gönderilmediği halde aynı yani orijinal haliyle diğer mikrodenetleyicinin lcd ekranında nasıl elde edildiği ele alınmıştı. Tabi ki bu çalışmada ki amaç girilen verinin gönderilirken istenmeyen üçüncü şahısların eline geçmemesiydi. Dolayısıyla verinin hızlı

ve güvenli bir şekilde gitmesi için sıkıştırma ve şifreleme algoritmaları kullanılarak orijinaliyle alakası olmayan gizli bir veri elde edildi. Ancak sadece bu veriyi göndererek karşı taraftaki mikrodenetleyici bu bilgiyi çözemez. Girilen karakterlere ve kaç adet girildiğine de ihtiyaç duymaktadır. Bu durumda RS232 seri iletişimle gönderilecek üç çeşit veri ortaya çıktı. Bunlar tuş takımından girilen her bir farklı karakter, bu karakterlere kaç defa basıldığı ve son olarak sıkıştırılıp şifrelenmiş olan veri. Amaç hızlı iletişim olduğu için burada üç farklı veri bir defa da gönderilmektedir.

Verilerin bir araya toplanması için bir dizi değişkeni tanımlanır. Bu dizi değişkenine önce farklı karakterler sırasıyla eklenir. Farklı karakterlerin bittiğini anlayabilmek için program içinde belirlenen bir karakterde bu diziye eklenir. Daha sonra bu farklı karakterlerin frekans değerleri yani veri içinde kaç defa geçtiği bilgileri diziye eklenir. Burada da frekans değerlerinin yazımının bitiminde program içinde belirlenen bir karakterde bu diziye eklenir. En son Huffman algoritmasıyla sıkıştırılıp, program içinde oluşturulan anahtarla XOR şifreleme metoduyla şifrelenen veri bu diziye eklenir. Şimdi oluşturulan veri programda *puts(gonderVeri)* ile RS232 seri iletişim metoduyla gönderilmektedir. Gönderilen veri diğer PIC18F452 mikrodenetleyicinin seri veri alma ucuna (RX) geldiği zaman programda oluşturulan #INT_RDA kesmesi aktif olur. Bu kesmenin aktif olabilmesi için programın main() bloğunda *enable_interrupts(GLOBAL)* ve *enable_interrupts(int_rda)* komutları tanımlanmıştır. 'GLOBAL' aktif edilen tüm kesmelere izin verilmesini, 'int_rda' da RS232 seri iletişim kesmesi olan int_rda kesmesinin aktif edilmesini belirtmektedir.

Kesme işlemine giren program sırasıyla gelen verilere 3 gruba ayırıp onları tanımlanan dizilerin içine aktarmaktadır. *sembol[kar] = gonderVeri[kar]* komutu ile semboller, sembol[] dizisinin içine, *frekans[a] = gonderVeri[kar]* komutu ile frekanslar, frekans[] dizisinin içine aktarılmaktadır.

Diğer şifreli veride ise karakterler birer birer kripto[] dizisinin içine aktarılmaktadır ve aynı anda da *kriptanaliz[b] = anahtar[b]^kripto[b]*

oluşturulan anahtar[] dizisiyle XOR şifreleme yöntemiyle tekrar işleme sokularak veri, şifrelenmeden önceki haline dönüştürülmektedir. Yani sıkıştırıldıktan sonraki haline döndürülür.

Huffman sıkıştırma algoritmasıyla oluşturulan veri, girilen karakterler yerine onları karşılayan bitsel (0 ve/veya 1) ifadeler yazılarak oluşturulmuştu. Burada da sembol[] ve frekans[] dizisine aktardığımız verilerle, aynı Huffman sıkıştırma algoritmasını kullanarak, o karakterlere karşılık gelen bitsel ifadeler bulunarak orijinal veri elde edilmektedir. O bitlerle sıkıştırılmış veri birer birer karşılaştırılarak koşulu sağlayan sembol *printf(lcd_putc,"%c",sembol[i])* komutuyla lcd ekranına yazdırılmaktadır.

Böylelikle hiçbir anlam ifade etmeyecek şekilde gönderilen veri, Huffman algoritması, diğer mikrodenetleyicinin içinde yer alan aynı algoritmaya oluşturulmuş ve aynı değerleri içeren anahtar ve XOR şifreleme yöntemi kullanılarak orijinal haline dönüştürülmüştür.

7. SONUCLAR

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

8. ÖNERİLER

Bu uygulamada düşünülüp henüz gerçekleştirilemeyen uygulamanın devresi bir sonraki aşamada gerçekleştirilecektir. 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.

EKLER:

EK A. PiC18f452 Mikrodenetleyicinin Pin İsimleri ve Açıklamaları

Tablo 8-1 PIC18F452 PortA pinlerinin açıklanması

PIN Adı (PortA)	No	I/O/P Tipi	Buffer Tipi	Açıklama
OSC1/CLKIN	13	I	ST/CMOS(4)	Osilatör Clock girişi (Kristal ya da harici kaynak)
OSC2/CLKOUT/RA6	14	O	-	Osialatör çıkış ucu.Kristal veya resonator'a osilatör bağlantısı ile sağlanır.(¼ frekans OSC1 oranı) Digital giriş/çıkış
MCLR/V_{PP}	1	I/P	ST	Resetleme girişi, programlama anında programlama gerilimi girişi. (mikro denetleyicinin resetlenmesi için, bu pin lojik 0 yapılmalıdır)
RAO/ANO	2	I/O	TTL	PortA iki yönlü giriş/çıkış portudur. Anolog olarak kullanılabilir
RA1/AN1	3	I/O	TTL	Anolog olarak kullanılabilir
RA2/AN2/V_{REF-}	4	I/O	TTL	Anolog olarak kullanılabilir
RA3/AN3/V_{REF+}	5	I/O	TTL	Anolog olarak kullanılabilir
RA4/TOCKI	6	I/O	ST	Bu pin istenirse TMRO için bir clock girişi olabilir. SSP Slave seçme pini veya anolog giriş/çıkış olabilir.
RA5/SS/AN4/LVDIN	7	I/O	TTL	

Tablo 8-2 PIC18F452 PortA pinlerinin açıklanması

PIN Adı (PortA)	No	I/O/P Tipi	Buffer Tipi	Açıklama
OSC1/CLKIN	13	I	ST/CMOS(4)	Osilatör Clock girişi (Kristal ya da harici kaynak)
OSC2/CLKOUT/RA6	14	O	-	Osialatör çıkış ucu.Kristal veya resonator'a osilatör bağlantısı ile sağlanır.(¼ frekans OSC1 oranı) Digital giriş/çıkış
MCLR/V_{PP}	1	I/P	ST	Resetleme girişi, programlama anında programlama gerilimi girişi. (mikro denetleyicinin resetlenmesi için, bu pin lojik 0 yapılmalıdır)
RAO/ANO	2	I/O	TTL	PortA iki yönlü giriş/çıkış portudur. Anolog olarak kullanılabilir
RA1/AN1	3	I/O	TTL	Anolog olarak kullanılabilir
RA2/AN2/V_{REF-}	4	I/O	TTL	Anolog olarak kullanılabilir
RA3/AN3/V_{REF+}	5	I/O	TTL	Anolog olarak kullanılabilir
RA4/TOCKI	6	I/O	ST	Bu pin istenirse TMRO için bir clock girişi olabilir. SSP Slave seçme pini veya anolog giriş/çıkış olabilir.
RA5/SS/AN4/LVDIN	7	I/O	TTL	

Tablo 8-3 PIC18F452 PortB pinlerinin açıklanması

PIN Adı (PortB)	No	I/O/P Tipi	Buffer Tipi	Açıklama
RBO INT0	33	I/O	TTL/ST(1)	Giriş konumunda iken dahili pull-up devresi aktifleşebilir. Dış kesme giriş olarak seçilebilir.
RB1/INT1	34	I/O	TTL	Harici giriş
RB2/INT2	35	I/O	TTL	Harici giriş
RB3/CCP2*	36	I/O	TTL	PWM2 çıkışı olarak
RB4	37	I/O	ST	kullanılabilir. Kesme girişi olarak kullanılabilir.
RB5/PGM	38	I/O	TTL	Düşük akımla programlamada kullanılabilir. Kesme girişi olarak kullanılabilir.
RB6/PGC	39	I/O	TTL/ST(2)	Kesme girişi olarak kullanılabilir.Seçme programlamada clock pinidir.
RB7/PGD	40	I/O	TTL/ST(2)	Kesme girişi olarak kullanılabilir. Seçme programlamada data pinidir.

Tablo 8-4 PIC18F452 PortC pinlerinin açıklanması

PIN Adı (PortC)	No			I/O/P Tipi	Buffer Tipi	Açıklama
RCO/T1OSO/T1CK1	15	16	32	I/O	ST	Timer1 osc. Çıkışı veya saat girişi olarak kullanılabilir.
RC1/T1OSI/CCP2	16	18	35	I/O	ST	Timer1 osc. Girişi veya capture2 girişi/compare2çıkışı/PWM1 çıkışı olarak kullanılabilir.
RC2/CCP1	17	19	36	I/O	ST	Timer1 osc. Girişi veya capture1 girişi/compare1çıkışı/PWM1çıkışı olarak kullanılabilir
RC3/SCK/SCL	18	20	37	I/O	ST	SPI ve I modunda , seri saat giriş/çıkışında kullanılabilir.
RC4/SDI/SDA	23	25	42	I/O	ST	SPA moda , SPI girişverisi veya Ic modda I/O için.
RC5/SD0	24	26	43	I/O	ST	SPA modda , SPI çıkış verisi için seçilebilir.
RC6/TX/CK	25	27	44	I/O	ST	USART asenkron gönderme ya da senkron saat için…
RC7/RX/DT	26	29	1	I/O	ST	USART senkron alma ya da senkron veri için…

Tablo 8-5 PIC18F452 PortD pinlerinin açıklanması

PIN Adı (PortD)	No			I/O/P Tipi	Buffer Tipi	Açıklama
RD0/PSP0	19	21	38	I/O	ST/TTL(3)	PSP 0. biti olarak kullanılır.
RD1/PSP1	20	22	39	I/O	ST/TTL(3)	PSP 1. biti olarak kullanılır.
RD2/PSP2	21	23	40	I/O	ST/TTL(3)	PSP 2. biti olarak kullanılır.
RD3/PSP3	22	24	41	I/O	ST/TTL(3)	PSP 3. biti olarak kullanılır.
RD4/PSP4	27	30	2	I/O	ST/TTL(3)	PSP 4. biti olarak kullanılır.
RD5/PSP5	28	31	3	I/O	ST/TTL(3)	PSP 5. biti olarak kullanılır.
RD6/PSP6	29	32	4	I/O	ST/TTL(3)	PSP 6. biti olarak kullanılır.
RD7/PSP7	30	33	5	I/O	ST/TTL(3)	PSP 7. biti olarak kullanılır.

Tablo 8-6 PIC18F452 Port E pinlerinin açıklanması

PIN Adı PORT E	No			I/O/P Tipi	Buffer Tipi	Açıklama
RE0/RD/AN5	8	9	25	I/O	ST/TTL(3)	Anolog olarak ya da PSP okuma kontrolü kullanılabilir.
RE1/WR/AN6	9	10	26	I/O	ST/TTL(3)	
RE2/CS/AN7	10	11	27	I/O	ST/TTL(3)	

Tablo 8-7 PIC18F452 besleme pinlerinin açıklanması

PIN Adı (PortE)	No			I/O/P Tipi	Buffer Tipi	Açıklama
V_{SS}	12,31	13,84	6,29	P	-	Ground (toprak) uç.
V_{DD}	11,32	12,35	7,28	P	-	Pozitif kaynak ucu.
NC	-	1,17, 28,40	12,13 33,34		-	Bu pinler içeride kontrol edilmiyor. Bağlı değiller.

KAYNAKLAR

[1] Buluş, H. N., TEMEL ŞİFRELEME ALGORİTMALARI VE KRİPTANALİZLERİNİN İNCELENMESİ, Yüksek Lisans Tezi, Trakya Üniversitesi Fen Bilimleri Enstitüsü , Edirne, (2006), Sayfa No:1

[2] Schneider B. ”Applied Cryptography Second Edition”, John Wiley & Sons, Inc., New York, 1996

[3] Koltuksuz A., “Elektronik Ticarette Güvenlik, Özgürlük Denetimi, Doğruluk-Bütünlük ve Sayısal İmza“, 4.Türkiye İnternet Konferansı, İstanbul, Türkiye, (1998)

[4] Akyıldız, E., “ODTÜ’ de Kriptoloji Konusunda Yapılan Araştırmalar ve Uygulamalar”,ulusal e-devlet konferansı,4-5 Kasım 2008,Ankara

[5] Çölkesen, R., Bilgisayar Programlama ve Yazılım Mühendisliğinde veri yapıları ve algoritmalar, Papatya Yayıncılık, 2005, Sayfa No:402 - 404, 406

[6] Mesut, A., VERİ SIKIŞTIRMADA YENİ YÖNTEMLER, Doktora Tezi, Trakya Üniversitesi Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Anabilim Dalı, Edirne, (2006), Sayfa No:41-47 ,54 - 57

[7] http://www.csharpnedir.com/makalegoster.asp?MId=189

[8] Öcal F., GÜVENLİ İLETİŞİM İÇİN FPGA KULLANARAK ŞİFRELEME SİSTEMİ TASARIMI VE GERÇEKLEŞTİRİLMESİ, Yüksek Lisans Tezi, Gazi Üniversitesi Fen Bilimleri Enstitüsü, Haziran 2006

[9] Yerlikaya T., Yeni Şifreleme Algoritmalarının Analizi, Doktora Tezi, Trakya Üniversitesi Fen Bilimleri Enstitüsü, (2006)

[10] Yıldırım K., Veri Şifrelemesinde Simetrik Ve Asimetrik Anahtarlama Algoritmalarının Uygulanması (Hybrid Şifreleme), Yüksek Lisans Tezi, Kocaeli Üniversitesi Fen Bilimleri Üniversitesi, Kocaeli, (2006), Sayfa No: 8,9,10,11,12,13

[11] ORKAN, A. L., ÇALIŞKAN, B, Sıkıştırma ve Şifreleme , Bilişimin Temel Kavramları/13.Hafta, Marmara Üniversitesi İletişim Fakültesi Gazetecilik Bölümü / 2009-2010 Güz Dönemi

[12] http://forum.diyaudiotr.com/makaleleriniz/dijital-ses-formatlari-t966.html

[13] http://tr.wikipedia.org/wiki/TIFF

[14] http://tr.wikipedia.org/wiki/Portable_Network_Graphics

[15] Schneier, B., “Applied Cryptology, Second Edition: Protocols, Algorithms, and Source Code in C”, *Wiley Publishing*, (1996)

[16] Salomaa, A., “Public-Key Cryptography”, *Springer Verlag*, New York, (1990)

[17] Brown, L. P., “Analysis of DES and the Design of the LOKI Encryption Scheme, for Doctor of Philosophy”, Department of Computer Science, University College, *University of new South Wales*, Australia, April 1991.

[18] Daemen, J., Rijmen, V., “The Design of Rijndael AES-The Advanced Encryption Standard Series: Information Security and Cryptography”, *Springer Verlag 2002*.

[19] http://www.pro-g.com.tr/whitepapers/bilisim-guvenligi-v1.pdf

[20] Salomaa, A., “Public-Key Cryptography”, *Springer Verlag*, New York, (1990)

[21] Altınbasak, O., Mikrodenetleyiciler ve PIC programlama, Altas Kitapevi, (2001)

[22] Keleş,F.,MİKRODENETLEYİCİ KONTROLLÜ REDRESÖR TASARIM VE GERÇEKLENMESİ, Yüksek Lisans Tezi, Dumlupınar Üniversitesi Fen Bilimleri Enstitüsü, Subat-2006, Safa No: 32,33

[23] Ayyıldız S., Jal ile PIC Programlama, *Alta*s *Basım*, İstanbul, 361-362 (2006).

[24] BOZKURT N., MİKRODENETLEYİCİ KONTROLLÜ SERVO GERİLİM REGULATÖRÜNÜN TASARIM VE UYGULAMASI, Yüksek Lisans Tezi, Gazi Üniversitesi Fen Bilimleri Enstitüsü, Ekim 2006, Sayfa No: 35-40

[25] http://www.teknobakis.com/ccs-c-ile-pic-programlama-ccs-c-compiler-resimli-anlatim

[26] http://tr.wikipedia.org/wiki/Proteus

[27] Çiçek, S., CCS C ile Pic Programlama, *Atlaş*, (2009), Sayfa No 170,432-437

[28] http://e-bergi.com/2010/Nisan/Veri-Sikistirma

[29] http://tr.wikipedia.org/wiki/JPEG

[30] Yücel R., UZAKTAN KONTROLLÜ MİKRODENETLEYİCİLİ PROTOTİP, Yüksek Lisans Tezi, Sakarya Üniversitesi Fen Bilimleri Enstitüsü, Eylül 2007, Sayfa No: 6,7