

T.C.

GEBZE YÜKSEK TEKNOLOJİ ENSTİTÜSÜ
MÜHENDİSLİK VE FEN BİLİMLERİ ENSTİTÜSÜ

PROSES TASARIM, BENZETİM VE
KONTROLU İÇİN GENEL AMAÇLI BİR
SCADA PROGRAMI

Eşref DERİNKAYA

YÜKSEK LİSANS TEZİ

109941

BİLGİSAYAR MÜHENDİSLİĞİ

ANABİLİM DALI

T.C. YÜKSEKÖĞRETİM KURULU
DOKÜMANTASYON MERKEZİ

TEZ DANIŞMANI

Doç.Dr. E. Murat ESİN

GEBZE

2001

.....Esat DERİNKAYA.....'ın tez çalışması, G.Y.T.E. Mühendislik
ve Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 09.02.2001 tarih ve
2001/1.4 sayılı kararıyla oluşturulan jüri tarafından
.....Altınay Müh...... Anabilim Dalında YÜKSEK LİSANS tezi
olarak kabul edilmiştir.



JÜRİ

ÜYE
(Tez Danışmanı)

: Doç. Dr. E. Murat ESİN

Murattain

ÜYE

: Y. Doç. Dr. Hakan HOCAOĞLU

Hakan

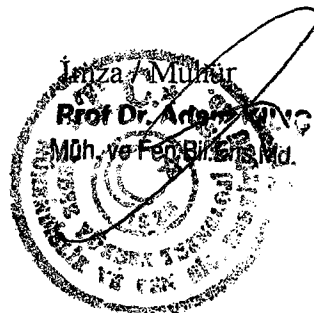
ÜYE

: Y. Doç. Dr. Ertuğrul KARAGÜHA

Ertugrul

ONAY

G.Y.T.E. Mühendislik ve Fen Bilimleri Enstitüsü Yönetim Kurulu'nun
29.05.2001 tarih ve 2001/1.3 sayılı kararı.



ÖZET



Bu çalışma PC ortamında bir Supervisory Control And Data Acquisition (SCADA, gözlemci kontrol ve veri toplama) yazılım paketi geliştirilmesini konu alır.

Paketin ismi olan SimKon SİMülasyon ve KONTrol kelimelerinin baş harflerinin alınması suretiyle oluşturulmuştur.

Farklı alanlarda gerçekleşen ve bilgisayar üzerinde modellenebilecek proseslerin benzetimi ve kontrolü için bir yazılım paketi geliştirilmiştir. Böylelikle proseslerin modellenmesi yapılarak bunun gerçek zamanlı kontrolünün yapılabilmesine olanak sağlamaktadır. Modüler bir yapıda tasarlanmıştır. Böylelikle programı kullanan kişilerin kendi hesaplama ve kontrol rutinlerinin bu programa sonradan eklenmesi mümkün olmaktadır.

Paket Borland C++ Builder kullanılarak Windows ortamı için geliştirilmiştir.

Birinci bölümde giriş bölümü yer almaktadır ve bilgisayar grafiklerinin gelişiminden bahsedilmiştir.

İkinci bölümde SCADA hakkında bilgi verilmiştir.

Üçüncü bölümde SCADA yazılım paketinin oluşturulmasından bahsedilmiştir.

Dördüncü bölümde örnek bir prosesin tasarım ve benzetimi gösterilmiştir.

Beşinci bölüm sonuç bölümüdür.

EK1’de SİMKON kullanım kılavuzu verilmektedir.

EK2’de SİMKON kaynak kodunun bulunduğu CD yer almaktadır

Sonuç olarak kullanılabilir bir SCADA paketi ortaya çıkmıştır.

SUMMARY



This work is based on a Supervisory Control And Data Acquisition(SCADA) software development.

Simkon, the name of the package, has been inspired from the words of “simulasyon” and “kontrol” which means, in Turkish, simulation and control, respectively.

The developed software package can simulate and control different types of processes that can be modelled on a computer. In addition, a real time controlling of the processes, modelled on the computer, is possible. The software package has a modular structure. Therefore control or calculation routines may be plugged in when requested.

This Software package has been developed with C++ Builder 5.0 for Windows 95 and above.

In Chapter 1, Evolution of the Computer Graphics is presented.

In Chapter 2, Information about SCADA is presented.

In Chapter 3, Development of SCADA software package is presented.

In Chapter 4, An example of process simulation and design is illustrated.

Chapter 5 is the conclusion part.

Appendix 1, The User Manual of SCADA.

Appendix 2 includes a CD with source code of SIMKON.

Appendix 3 The source code of the developed software (in the CD).

Finally a useful SCADA package is developed and presented in the disk provided with the thesis.

TEŞEKKÜR



Bu tez çalışmasının geliştirilmesi sırasında yardımlarını ve hoşgörüsünü esirgemeyen tez danışmanım Doç. Dr. E. Murat ESİN'e teşekkürlerimi sunarım. Bu çalışmayı, tezin başlangıcı sırasında meydana gelen yüzyılın felaketinde hayatını kaybeden, eski iş arkadaşım ve hocam Selim BOZKURT'a adamayı bir vefa borcu olarak görüyorum.

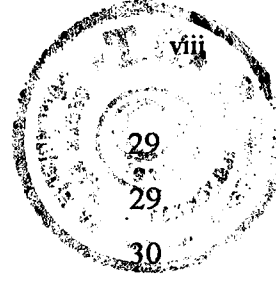


İÇİNDEKİLER DİZİNİ



Sayfa

ÖZET	IV
SUMMARY	V
TEŞEKKÜR	VI
İÇİNDEKİLER DİZİNİ	VII
ŞEKİLLER DİZİNİ	X
TABLolar DİZİNİ	XI
1. GİRİŞ	1
1.1. Bilgisayar Ortamında Grafikler	3
1.2. Grafik Kullanıcı Arayüzü	4
1.3. Bilgisayar Destekli Çizim Programları	4
2. SCADA	6
2.1. SCADA Uzak Terminal Birimleri (Remote Terminal Units, RTU)	9
2.1.1. Bir RTU'daki Donanım İşlevselliği	11
2.1.2. RTU'da Yazılım İşlevselliği	12
2.1.3. RTU'ların Çalışması	12
2.2. İletişim Ortamı	13
2.3. SCADA Sistemlerinde Kontrol	14
2.4. SCADA Sisteminden Beklentiler	15
3. SCADA PAKETİNİN OLUŞTURULMASI	17
3.1. Pakette Kullanılan Algoritmalar	17
3.1.1. Programın Genel Döngüsü	18
3.2. Veri Yapıları	24
3.2.1. Nesnelist Dizisi	24
3.2.2. Akımlist Dizisi	25
3.2.3. Tipler Listesi	25
3.2.4. Diğer Yapılar	26
3.3. Dosya Yapıları	27
3.3.1. Ayar Dosyası	28
3.3.2. Nesne Dosyası Yapısı	28



3.3.3. Akım Dosyası Yapısı	29
3.3.4. Kayıt Dosyası Yapısı	29
3.4. Yazılımın Geliştirilmesi	30
3.4.1. Çizim Formu	31
3.4.1.1. Timer1	32
3.4.1.2. Timer2	33
3.4.1.3. HintTimer	33
3.4.1.4. Timer3	33
3.4.1.5. OpenFileDialog ve SaveDialog1	33
3.4.1.6. ImageList1	33
3.4.1.7. MainMenu1	34
3.4.1.8. PopupMenu1, PopupMenu2 ve PopupMenu3	34
3.4.1.9. Memo1	34
3.4.1.10. Edit3	34
3.4.1.11. StatusBar1	34
3.4.1.12. Image1	34
3.4.2. Toolbar	35
3.4.2.1. Elemanlar	35
3.4.2.2. Bağlantılar, Katmanlar	36
3.4.2.3. Plug-in	37
3.4.2.4. Çizim Elemanları	37
3.4.3. Birim Editörü	37
3.4.3.1. Image 1	38
3.4.3.2. Panel 1	38
3.4.3.3. Panel 5	38
3.4.4. Özellik Tablosu	38
3.4.5. Çevirme ve Büyütme Formu	40
3.4.6. Plugin Ayarı	40
3.4.7. Diğer Formlar	41
3.5. Kontrol ve Benzetim	41
4. ÖRNEK TASARIM UYGULAMASI	43
4.1 Benzetim Programı	45
5.SONUÇ	47

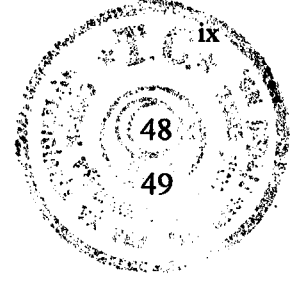
KAYNAKLAR

ÖZGEÇMİŞ

EKLER

EK-1

EK-2



ŞEKİLLER DİZİNİ



Şekil

	Sayfa
2.1. Bir SCADA Sistemi (12)	8
2.2. MTU'ya Olan Giriş ve Çıkışlar (12)	9
2.3. RTU'nun Giriş ve Çıkışları (12)	10
2.4. Kontrol Sistemi	14
3.1. SİMKON'un Yapısı	17
3.2. Tasarım Sırasında Çizim Formu ve İçerdiği Komponentler.	32
3.3. Tasarım Sırasında Araç Çubuğu Formu ve İçerdiği Komponentler.	36
3.4 Birim Editörünün Tasarım Sırasındaki Görüntüsü.	39
3.5. Inspector Table Formu	39
3.6. Büyütme ve Çevirme Formu	40
3.7. Plugin Ayar Formu	41
4.1 Birim Editörü	43
4.2. Örnek Tasarım Uygulaması	44
4.3. Benzetim Programının Gönderdiği Sonuç.	46

TABLÖLAR DİZİNİ



Tablo

2.1. Tipik Bir SCADA Sisteminden Beklentiler (1)

16

3.1. Nesne Tanımlamaları

24

3.2. Akım Tanımlamaları

25

3.3. Nesne Tipi Tanımlamaları

26

3.4. Diğer Değişkenler

26

3.5. Sabitler

27

3.6. Örnek Birim Dosyası

29

3.7. Örnek Akım Dosyası

29

3.8. Örnek Kayıt Dosyası

30

3.9. Paralel Porta Erişen Kod.

42

3.10. Windows NT İçin Port Erişim API'leri.

42

1.GİRİŞ



Bilgisayarlar, ilk kullanılmaya başlandığı zamandan bugüne kadar sürekli bir şekilde gerek donanım ve gerekse yazılım olarak gelişmiştir. İlk kullanılan bilgisayarların giriş ünitesi olarak delikli kartlar kullanılırken zamanla komut yorumlayıcılar geliştirilmiş ve artık günümüzde tamamen grafik tabanlı etkileşimli arabirimler kullanılarak bilgisayarı kullanmak oldukça kolaylaşmıştır. Günümüzde artık kullanıcı dostu bir arabirimi olmayan bilgisayarlar yavaş yavaş azalarak yerlerini kolay kullanılabilen arabirime sahip bilgisayarlara bırakmaya başlamıştır.

Eskiden her paket program için arabirim tamamen farklı olduğundan farklı arabirime sahip programları kullanmak zor olmuştur. Bugün Windows ve Xwindow gibi pencere yöneticilerinin de sayesinde artık standart bir arabirim getirilmeye başlanmıştır. Böylelikle bir paket programı kullanmayı bilen bir kişi, program tamamen farklı bir işi de yapıyor olsa diğer programı çok daha kolay bir şekilde kullanmayı öğrenebilmektedir. Örneğin Windows işletim sistemi altında çalışan programların arabirimleri birbirine oldukça benzerdir.

Bu çalışma ile değişik alanlardaki proseslerin tasarım ve benzetimi için tek bir arabirim geliştirilmek istenmektedir. Örneğin, birbirinden oldukça farklı alanlar olan kimyasal prosesler ile elektriksel proseslerin tasarımlarının tek bir arabirim altında yapılabilmesi mümkün hale gelmiştir. Böylelikle sadece kullanılan nesneler ve nesne arasındaki bağlantı ayarları değiştirilerek istenilen türde bir sistem tasarlanabilecek hale gelmiştir. Geliştirilen bu yazılım paketine simülasyon ve kontrol kelimelerinin başındaki üçer harf alınmak suretiyle SİMKON ismi verilmiştir. SİMKON ile birbirinden tamamen farklı çok değişik alanlardaki proseslerin tasarımı mümkün olabilmektedir. Tasarlanan bu sistemlerin benzetimi için sadece benzetim veya kontrol işini yapan ve bir arabirime ihtiyacı olmayan küçük programların yazılması yeterli olmaktadır. Yazılan bu benzetim veya kontrol programları sistem grafini değerlendirerek benzetim veya kontrol sonuçlarını SİMKON'a gönderir.



Bu konuda yurt dışında Hexatec Software'in geliştirdiği Saturn[1], National Instruments'in geliştirdiği LabView[2], University of Twente tarafından geliştirilen Twenty Sim[3] ve Wonderware Corporation tarafından geliştirilen Wonderware[4], Sysmac Productions tarafından geliştirilen Sysmac[5], Boeing Company tarafından geliştirilen Easyware[6], gibi oldukça değişik paket olmasına rağmen Türkiye'de bugüne kadar sadece MikroSay Yazılım firması tarafından TEOS isimli bir SCADA paketi geliştirilmiştir. [7]

TEOS paketi TÜBİTAK'ın katkılarıyla 10 yıllık bir endüstriyel deneyim ve 4 yıllık bir yazılım geliştirme çabası sonucu ortaya çıkmıştır. TEOS paketi Delphi 2.0 kullanılarak yazılmıştır. Paketin kendisine ait bir yorumlayıcı dili bulunmaktadır. Gelişmiş bir tasarım editörüne sahip olan bu pakette kullanıcıların kendi birimlerini oluşturma aracı yoktur. Bununla beraber benzetim için oldukça güzel bir arabirimi vardır. Ayrıca değişik donanımları kontrol etmek için birçok donanım sürücüsü vardır.

Dünya ile birlikte Türkiye'de de bir çok SCADA uygulamasına rastlamak mümkündür. Örneğin İstanbul Metrosunda bulunan yürüyen merdivenler, havalandırma fanları, aydınlatma sistemi, yangın ihbar ve koruma sistemleri ve enerji dağıtım sistemleri tamamen bilgisayarlarla izlenebilmekte ve gerekli müdahaleler merkezi bir kontrol odasından yapılabilmektedir. Enerji besleme noktalarından herhangi biri arızalandığı zaman SCADA sistemiyle 1-2 dakika içerisinde yedek besleme devreye alınarak ve enerji sürekliliği sağlanmaktadır. Herhangi bir yangın, kaza gibi olağanüstü durumlarda ise çevresel kontrol sistemi otomatik olarak devreye girerek yolculara taze havanın gelmesi ve dumanın güvenli bir şekilde dışarı alınmasıyla yangının söndürülmesi sağlanmaktadır.

Ayrıca Bolu'da yapılmakta olan deprem konutlarının elektrik dağıtım sistemlerinde de SCADA sisteminden faydalanılmaktadır. Botaş'ın doğal gaz hatlarında, TEK Elektrik üretim ve dağıtımında, Kayseri Su ve Kanalizasyon İdaresi tarafından içme suyu dağıtım şebekesinin kontrolünde, ASKİ tarafından su depoları, Pompa istasyonları ve ölçüm noktalarının kontrolünde, Antalya'da Jet yakıtı pompalayan ÇEKİSAN firmasında vs. SCADA sistemlerinden

faýdalanılmaktadır.[7],[8] Görüldüğü gibi bir çok farklı alanlarda SCADA sistemleri kullanılmaktadır.



1.1. Bilgisayar Ortamında Grafikler

Bilgisayar ortamında grafiklerin tarihi 1950'li yıllara kadar gider. Bu yıllarda ABD hava savunma sisteminde giriş için lightpen ve çıkış için CRT(Cathode Ray Tube) ünitesini kullanan etkileşimli grafiklerin basit bir versiyonunu kullanmaya başladı.

Modern bilgisayar grafikleri ise 1963 yılında MIT(Massachusetts Institute of Technology)'de doktora öğrencisi olan Ivan Sutherland'ın Sketchpad projesi ile başladı. Onun bu çalışması CRT display ile light pen'in kullanıldığı bir çizim sistemini temsil ediyordu.

1963 yılından sonra T.E. Johnson sketchpad'i nesnelerin değişik cephelerden görünüşü ve perspektifleri için uyarladı. Bugünün CAD (Computer Aided Design) sistemleri, hala sık sık bu projeden alıntılar yaparak yazılır.

1960'ların başında vektör tazelemeli ekranlar yapıldı. Bunlar çok fazla tampon bellek ve karmaşık işlemler gerektiriyordu.

1970'lerden sonra raster ekran yapımı başarılması, mikro ve mini bilgisayarların yapımı ve sistemden bağımsız programların yazılmaya başlandı. Böylece CRT display sistemleri, kendi işlemcilerine sahip olarak sisteminkinden bağımsız çalışabilir duruma gelip, esnek ve hızlı programların yolunu açmıştır.[9]

Bu gün ise her boyda bilgisayar için yazılmış CADD(Computer Aided Design and Drafting) yazılımı bulunabilmektedir.

1.2. Grafik Kullanıcı Arayüzü

Bilgisayarın ilk kullanılmaya başlandığı yıllarda grafikten söz etmek imkansızdı. Teknolojinin gelişmeye başlayıp CRT ekranların yaygınlaşmasından sonra bilgisayar grafiklerinde baş döndüren bir gelişme yaşandı. Önceleri metin tabanlı ve klavyeden yönetilebilen programlar yerini yavaş yavaş grafik ekranda çalışan uygulamalara bıraktı. Günümüzde ise artık grafik kullanıcı arabirimi kullanmayan yazılımların artık sayıca çok azalmıştır.

Kullanılması kolay programlar bilgisayar kullanmayı kolaylaştırdığından grafik kullanıcı ara yüzleri bilgisayarların yaygınlaşmasını sağlamıştır.

1.3. Bilgisayar Destekli Teknik Çizim Programları

Bilgisayar destekli teknik çizim programları nesnelerin teknik resim kurallarına uygun olarak ölçekli çizimine imkan verir. Ayrıca teknik resimde kullanılan kesikli çizgi, ölçek çizgisi ve benzeri sembolik gösterimlere sahiptir. Zoom ve kaydırma işlemlerinde çizgi kalınlıkları değişmez. Bu nedenle daha çok vektör tabanlı dosyalama yapılır. Bazı paketlerde istenirse raster (bitmap) tabanlı çizimle dönüşüm yapılabilir.

Alışılmış analiz teknikleriyle oluşturulan graf ve argümanların elle girildiği tasarım programları genel olarak grafik desteğinden yoksundur. Hazırlık çalışmalarından başlayarak uzman tasarımcılar tarafından kullanılabilen bu tür paketler, çoğunlukla konuyla ilgili gruplar tarafından oluşturulmuştur. Ticari dağıtım ve yaygın kullanım şansları sınırlıdır. Tasarıma konu olan prosese ilişkin çizimlerin yapılması ve buna ilişkin grafın belirlenmesi profesyonellik gerektirir. Bunlarda çizim aşamasında, bağımsız yazılım firmalarının genel amaçlar için ürettiği çizim paketleri kullanılır.

Grafik ortamdaki CADD programları ise tasarımın başladığı taslak çizme aşamasından itibaren bir grafik kullanıcı ara yüzü içinde çalışır. Son yıllarda

5

özellikle mikrobilgisayar kapasitelerinin artması ile yaygınlaşmıştır. Tasarımcının yapması gereken birçok ara işlemi üstlenmesi büyük kolaylıklar sağlamaktadır.

Belli bir alanda uygulama geliştirmeye yönelik olan bu tür paketler, bir çizim ortamı içinde proses elemanlarının gösterilmesi için önceden hazırlanmış sembol kütüphanesine sahiptir. İhtiyaç duyulan bileşenler buradan çağrılabilmekte ve ekranda uygun bir yere yerleştirilmektedir. Bunlar arasında fonksiyonel ilişkiler tanımlanarak matematik model oluşturulmaktadır. Daha sonra modelin tanımladığı prosesin argümanları verilerek asıl tasarım programı işletilmekte ve sonuçlar isteğe göre grafik ya da metin olarak aynı ortamdan alınabilmektedir.

2. SCADA



Dinamik sistemleri oluşturan elemanların haberleşmesi ve buna dayanarak belirli denge durumlarına gelmeleri matematiksel bir şekilde ifade edilmeye çalışılmıştır. Sibernetik adı verilen bilim dalı hemen her tür sistemin benzer denge arayışlarına sahip olduğunu, bu nedenle de benzer matematik denklem sistemleriyle ifade edilebileceklerini göstermiştir.[10]

Sibernetiği mühendislik alanına uygulayan sistemler teorisi de, ele alınan sistemin çalıştığı enerji ortamını birbirinden ayırt etmeden, hatta tanımadan çalışma biçimini matematik olarak ifade edebilmeyi amaçlamaktadır.

Sistemleri oluşturan elemanlar belirli matematik modellere sahiptir ve herhangi bir başka enerji ortamında aynı modele sahip elemanlar bulunur. Bu yaklaşıma benzeşim denir. Buradan yola çıkarak, bir disiplin altında yetişmiş ve benzeşim teorisini kavramış bir uzmanın küçük çabayla başka sistemlerin çalışma şartlarını anlaması, hatta açıklaması kolayca mümkün olur.

Dinamik sistem dendiğinde, içinde zaman ve dolayısıyla değişim bulunduran yapılar akla geldiğine göre, burada sözü edilen çalışma alanının içine her enerji ortamından mühendislik sistemi aynı bakış açısından değerlendirilebilir.

O halde uğraşılan sistemin türü ne olursa olsun, sonuçta bu sistemi incelemek için yapılması gereken, her bir elemanın varlığıyla ifade ettiği matematik modeli tanımlayabilmek ve sistemde yer tutuşlarına göre aralarındaki ilişkiyi kurabilmektir. Böylece herhangi bir sistem bir bütün halinde anlaşılmaya hazır hale gelir.

Sonuçta bugün gelinen teknoloji, SCADA(Supervisory Control And Data Acquisition) gözlemci kontrol ve veri toplama sistemleridir. SCADA sistemleri ölçme sistemleri, modelleme, benzetim, karar destek senaryoları, kumanda donanım ve yazılımlarından meydana gelir. Ayrıca bütün bunları bir arada ele almamızı ve kullanmamızı sağlayan endüstriyel grafik kullanıcı arabirim sistemi bütünler.

“Gözlemci kontrol ve veri toplama”, yani SCADA teriminin kullanıldığı ilk yayın, 1973 yılında yapılan ilk PICA (Power Industry Computer Application) konferansında yapılmıştır.[11]



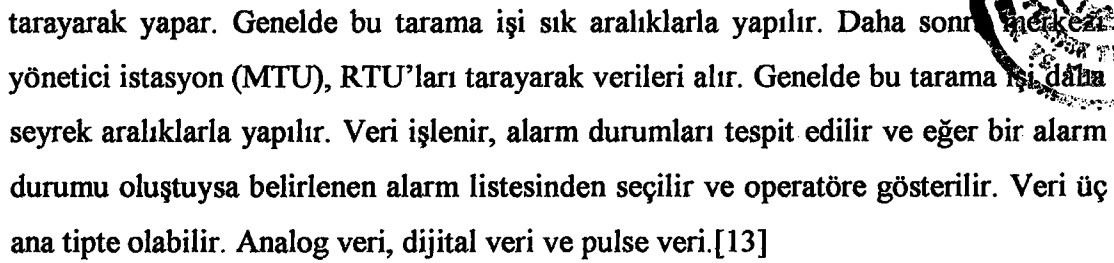
SCADA sistemleri, kısaca merkezi bir sunucu veya yönetici (genellikle ana istasyon, ana terminal birimi veya MTU olarak bilinir) ile bir veya daha fazla alanda bulunan veri toplama ve kontrol biriminin (genellikle uzak istasyonlar, uzak terminal birimi (RTU) şeklinde bilinir) bir takım standart veya özelleşmiş yazılım aracılığı ile uzaktaki veri elemanlarını kontrol edebilen bir endüstriyel ölçüm ve kontrol sistemidir. Aynı zamanda SCADA sistemleri ağırlıklı olarak açık döngü kontrol karakteristiğini ve yine ağırlıklı olarak uzak mesafe iletişimini kullanır.

Genellikle fabrikalarda, arıtım tesislerinde rastlanan SCADA benzeri sistemlerden daha çok Distributed Control Systems (DCS) olarak söz edilir. Genellikle SCADA sistemleri ile benzerlik gösterirler, fakat veri toplama veya kontrol üniteleri genellikle daha kapalı alanlarda yerleştirilir. Haberleşme bir yerel alan ağı (LAN, Local Area Network) kullanılarak yapılabilir ve normalde dayanıklı ve yüksek hızdadır. Bir DCS sistemi genellikle önemli miktarda kapalı döngü kontrolü kullanır.

Diğer taraftan SCADA sistemleri genel olarak yaygın bir coğrafi alanı kapsar ve doğal olarak bir LAN'dan daha az güvenilirlikteki bir çok değişik iletişim sistemlerine bağlıdır. Bu durumda kapalı döngü kontrolü daha az istenilen bir durumdur.

Bir SCADA sisteminde operatör, haberleşme, MTU ve RTU olmak üzere dört ana kısım vardır.[11]

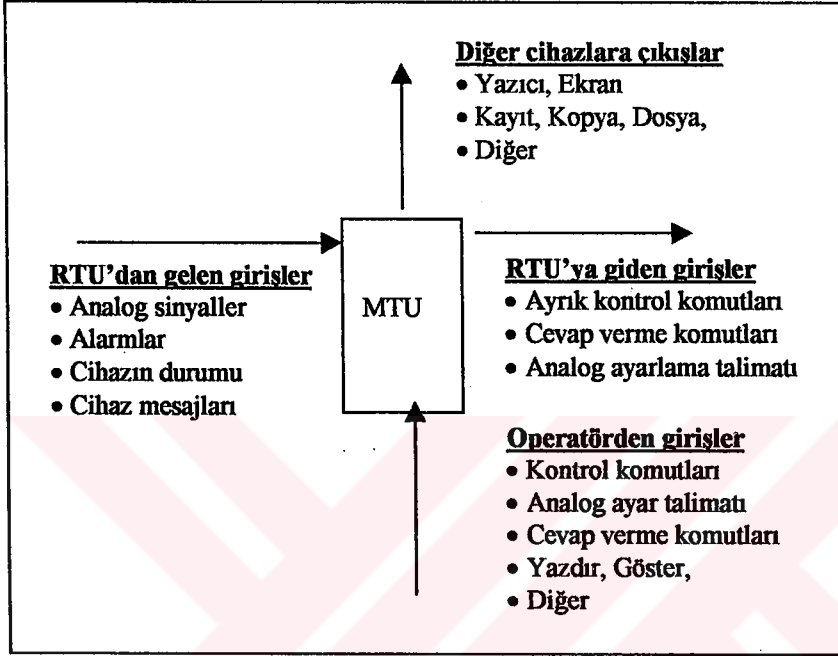
SCADA sistemleri bir teçhizat veya fabrikayı gözlemlemek ve kontrol etmek için kullanılır. Kontrol işlemi otomatik olabileceği gibi bir operatörün komutları ile de başlatılabilir. Veri toplama işini ilk önce RTU'lar kendine bağlı olan girişlerini



The diagram illustrates a radio network configuration. At the top, a horizontal line represents a power or communication line with four vertical poles. Below this, four Remote Terminal Units (RTU 1, RTU 2, RTU 3, RTU 4) are shown. Each RTU is connected to a modem (labeled 'MODEM'). The modems for RTU 1 and RTU 2 are connected to a central Master Terminal Unit (MTU) via radio links. The MTU is also connected to a radio link (labeled 'RADIO'). An operator is shown at a computer terminal connected to the MTU. The MTU is also connected to a radio link (labeled 'MODEM'). The modems for RTU 3 and RTU 4 are connected to radio links (labeled 'RADIO').

Şekil 2.1. Bir SCADA Sistemi [12]

Şekil 2.2’de tipik bir MTU’nun girişleri ve çıkışları gösterilmiştir. Bu şekilde görüldüğü üzere MTU’ya operatörden ve RTU’lardan giriş yapılabilmektedir. MTU bu girişleri değerlendirerek RTU’lara veya ekran, yazıcı, dosya gibi ortamlara çıkış verebilmektedir.



Şekil 2.2. MTU’ya Olan Giriş ve Çıkışlar, [12]

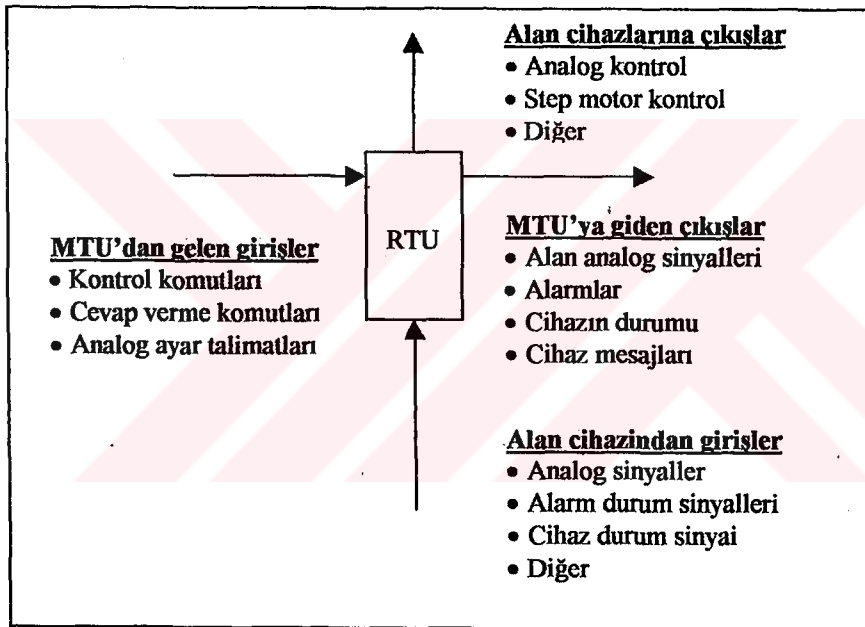
2.1. SCADA Uzak Terminal Birimleri (Remote Terminal Units, RTU)

Bu cihazlar SCADA sistemleri için anahtar pozisyonundadırlar. SCADA RTU'ları alandan bilgi toplamayı sağlayan sağlam ve küçük bilgisayarlardır. Merkezi yönetici SCADA'nın alan cihazları ile iletişimine izin verirler. RTU'lar bağımsız veri toplama ve kontrol ünitesidir. Görevi uzak bölgedeki proses cihazlarını kontrol etmek, bunlardan veri toplamak ve bu veriyi merkezi yönetici SCADA sistemine taşımaktır.

Merkezi kumanda ve izlemeyi sağlayabilmek için, RTU'lar tüm ölçüm sonuçları ile cihazların çalışma durumlarını (kesici açık-kapalı, vana açık-kapalı,

pompa çalışıyor-duruyor vb.) merkeze bildirir ve merkezden gelen komutları (kesici aç-kapat, vana aç-kapat, pompa durdur-çalıştır vb.) uygular. Böylece merkezi denetim biriminin başında buluna sistem operatörünün tüm ölçüm sonuçlarını görmesi ve komutlar göndererek sistemi yönetmesi sağlanır.

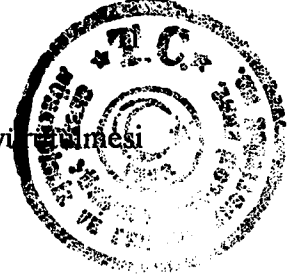
Şekil 2.3'te tipik bir RTU'nun olası giriş ve çıkışları gösterilmiştir. RTU'nun girişleri MTU'dan gelen komutlar ile alan cihazından okunan bilgiler olabilmektedir. RTU aldığı girişlere göre MTU'ya bilgi gönderebilmekte veya alan cihazının kontrolünü yapabilmektedir.



Şekil 2.3. RTU'nun Giriş ve Çıkışları [12]

İki temel tipte RTU vardır. Birincisi kompakt olan ve tüm giriş/çıkışların tek bir karttan yapılan single board RTU'lar. Bunlar sadece ölçüm yapar, merkeze bildirirler ve merkezden gönderilen basit komutları yerine getirirler. Karar verme yetkileri yoktur. Bunun dışında akla gelebilecek tüm işlevleri merkez yerine getirir.. Bu tür RTU'lar ile oluşturulan SCADA sistemlerinde bütün yük merkez bilgisayar üzerindedir. Merkez bilgisayar sürekli olarak RTU'ları tarayarak ölçüm değerlerini alır ve alarm durumu olup olmadığını sınar. Eğer alarm varsa gerekli müdahaleyi sağlayacak komutu RTU'ya gönderir. Alarm durumlarında ve elektrik, doğalgaz ve

su dağıtım sistemlerin yönetiminde bütün işlemlerin merkez tarafından yürütülmesi pratikte pek çok sorun çıkarmaktadır.



Diğeri ise ayrı bir CPU modülü olan ve başka modüllerin de eklenebileceği genişleme yuvası olan(PC ana kartına benzer yapıda) modüler RTU'lardır. Bunlar merkez bilgisayarın işlem yükünün bir kısmın üzerlerine alarak sistem verimliliğini ve performansını artırır. Ayrıca oluşacak tüm aykırı durumları sınıyarak alarm uyarıları üretebilir ve bu durumlarda ne yapılacağına kendi başlarına karar vererek anında müdahale sağlayabilir. Bunlar, kullanıcının özel gereksinimlerini yerine getirecek şekilde programlanabilir ve denetledikleri cihazların kullanıcı gereksinimlerini karşılayacak şekilde çalışmasını sağlarlar. Ayrıca, diğer RTU'lar ile iletişim kurarak istenilen işlevlerin yerine getirilmesini sağlayabilirler. Bu işlemler yürütülürken, merkeze sürekli olarak haber vererek sistemin genel işleyişinin ve son durumunun merkezden izlenmesine de olanak tanırlar.

Single board RTU'lar normalde sabit giriş/çıkışa sahiptir. Örneğin, 16 dijital giriş, 8 dijital çıkış, 8 analog giriş, ve 4 analog çıkış gibi. Bu sahip olduğu kapasiteyi geliştirmek mümkün değildir.

Modüler RTU'lar ilave modüllerin eklenerek genişletilmesine olanak verecek şekilde tasarlanmıştır. Tipik bir modül 8 analog giriş modülü veya 8 analog çıkış modülü olabilir. GPS modülü gibi bazı özelleşmiş modüller de olabilir.

2.1.1. Bir RTU'daki Donanım İşlevselliği

SCADA RTU'su küçük ve sağlam bir bilgisayardır. Aşağıdaki donanımsal özelliklere sahiptir :

CPU ve RAM

Programları ve verileri saklayabilmek için kalıcı bellek

Seri port, veya bir modem üzerinden iletişim yeteneği

Güvenli bir güç kaynağı

Watchdog timer

DI/DO/AI/AO'lar için I/O arabirimi
Real time clock

2.1.2. RTU'da Yazılım İşlevselliği

Bütün RTU'lar aşağıdaki işlevselliğe gereksinim duyarlar. Birçok RTU'da bunlar birleştirilmiş olurlar ve ayrı modüller halinde tanımlanmaya gereksinim duymazlar.

- Gerçek zamanlı işletim sistemi. Bu özel bir RTOS (Real Time Operating System) olabileceği gibi, girişleri tarayan ve iletişim portlarını gözetleyen tek ve büyük bir döngü olan kod olabilir.
- Merkezi SCADA ile haberleşme için bir driver
- I/O sistemi için ünite sunucuları.
- SCADA uygulaması. Örneğin girişleri tarayan, veriyi işleyen ve depolayan, merkezi SCADA'ya haberleşme ağı üzerinden cevap veren bir uygulama,
- Sistem kontrolü,

Ayrıca bazı RTU'ların bir dosya sistemi olabilir. Bu programları ve ayarlama dosyalarını kaydetmeyi sağlar.

2.1.3. RTU'ların Çalışması

RTU girişleri oldukça hızlı bir şekilde tarar. Değişikliklerin zamanını tutma, durum değişikliğini tutma gibi bazı işlemler yapar. Bazı RTU'lar başlangıç raporlarını merkezi SCADA'ya bildirme yeteneğine sahiptir. Daha çok rastlanan durum ise merkezi SCADA'nın değişiklikleri yapmadan önce RTU'lara değişiklikleri sormasıdır. Merkezi SCADA tarafından oylandığı zaman, RTU isteğe cevap vermelidir. Bu cevap basit bir şekilde "bana tüm veriyi yolla" olabileceği gibi, karmaşık bir kontrol fonksiyonun çalıştırılması şeklinde de olabilir.

RTU'lar kapasitelerine ve büyüklüklerine göre aşağıdaki gibi sınıflandırılabilirler.[11]

Küçük ve kendi başına çalışan sistemler. Bunlarda veri EPROM veya FLASH ROM içerisine kaydedilir ve sadece operatörün fiziksel erişimi ile veriye ulaşılabilir. Bu sistemler genelde minimum bellek içeren single chip işlemciler kullanır.

Orta büyüklükteki sistemlerde single board endüstriyel bilgisayarlar kullanılır.

Geniş sistemler fabrika veya daha üst düzeylerin kontrolünde kullanılır. Genellikle fabrikalardaki DCS (Distributed Control Systems) ve benzeri sistemlerde kullanılır. Yüksek hızda haberleşmeye olanak sağlayan LAN (Local Area Network) kullanılır. Zamanlama bu gibi sistemlerde oldukça kritik olabilir.[14]

2.2. İletişim Ortamı

SCADA sistemlerinde MTU ile RTU'lar arasındaki haberleşme ve RTU'ların kendi aralarındaki haberleşme pek çok yöntem ile sağlanabilir. Bunlardan bazıları aşağıda ifade edilmiştir.

- Telsiz
- PTT hatları
- Özel kablo hattı
- Fiber optik kablolar
- Uydu hatları

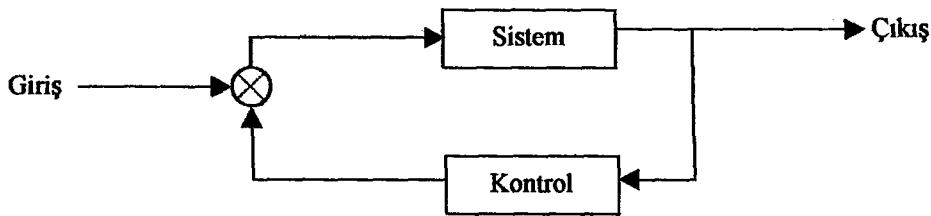
SCADA sistemlerinin uygulandığı yere göre yukarıdaki haberleşme ortamlarından veya bunların dışındaki bir ortam seçilebilir. Ayrıca bu ortamda kullanılacak olan veri iletişimi protokolü de önemlidir. İyi bir veri iletişim protokolü, gönderilen mesajların alıcıya tam ve doğru olarak ulaştığını garanti edebilmeli, fiziksel ortamdaki gürültülerden dolayı mesaj bozulmaları olduğunda hata kontrolü olmalı ve gerekirse ilgili mesajları tekrar göndererek istenilen işlevi tekrar sağlamalıdır. Önemli olan protokolün minimum hata ile güvenli bir iletişimi sağlayabilmesidir.

Ayrıca buradan tahmin edileceği üzere veri iletişimde güvenlik açıkları bulunabilmektedir. Özellikle geniş bir alanda kullanılacak SCADA sistemi tasarlanırken dışarıdan yapılması muhtemel saldırılara karşı koruma önlemleri mutlaka alınmalıdır. Ne yazık ki günümüzde yapılan SCADA uygulamalarında bu kısım pek fazla dikkate alınmamaktadır. Örneğin bir metro sisteminde haberleşme sistemine dışarıdan sızılabilmesi mümkün olmakla beraber bu konuda yeterli önlemler alındığı takdirde bu ihtimal oldukça azalmaktadır.

The Computer Emergency Response Team (CERT)'in 1989-1995 yılları arasında topladığı verilere göre henüz böyle bir saldırı yapılmamıştır. Bununla beraber 1995 yılında PCCIP (President's Commission on Critical Infrastructure Protection) tarafından bir yıl süreyle yapılan çalışma sonucu internette yapılması muhtemel siber saldırı 48.000 ile 44.000.000 olarak tahmin edilmiştir.[15],[16] SCADA sistemleri giderek büyüdükçe bu tehlikeye daha fazla maruz kalacaklardır. Ülke çapında bir sistemde SCADA sistemlerinde iletişim için kullanılan ortamın giderek karmaşıklaştığı düşünülürse bu konuda ciddi çalışmalar yapılması gerekmektedir.

2.3. SCADA Sistemlerinde Kontrol

Kapalı kontrol sistemleri iki ana bloktan meydana gelir. Bular sistemin modeli ve kontroldür. Kapalı bir sistemde girişler ve çıkışlar vardır. Bunların sayısı herhangi bir değer olabilir ve tamamen tasarlanan sistemle ilgilidir. Aşağıdaki şekilde geri beslemeli bir kontrol sistemi yer almaktadır.(Şekil 2.4)



Şekil 2.4. Kontrol Sistemi

Kontrol edilecek sistem için bir algılayıcı ve bir eylemlendirici olması yeterlidir. Böylece sistemde oluşan fiziksel olaylar algılayıcılar yardımıyla elektrik sinyaline dönüştürülüp bilgisayar için anlamlı hale getirildikten sonra eylemlendirici gereken şekilde kumanda edilerek sistemin kontrolünü sağlamak mümkündür.

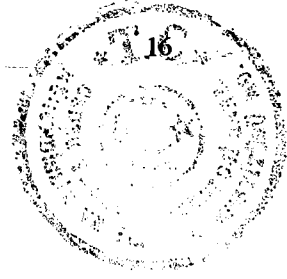
Kontrol sistemi, bir sonsuz çevrim içinde sürekli aynı işlemleri yapar. Bu işlemlerin şu aşamaları içerdiği kabul edilebilir.

- a. Ölçme
- b. Karar
- c. Kumanda
- d. Proses cevabı

2.4. SCADA Sisteminden Beklentiler

Bir SCADA sistemi sonsuz bir çevrim içinde veri toplama, benzetim programını çalıştırma, kontrol verilerini sisteme aktarma ile toplanan verileri süzerek kullanıcıya gösterir.

Bir SCADA sisteminden beklenenler, Tablo 2.1'deki gibi ifade edilmiştir. Burada tasarım ile benzetim ve kontrol olmak üzere iki temel bölüm vardır. Tasarım kısmında yer alan birim kısmında, tasarım sırasında kullanılan birimler ve bunlar üzerindeki işlemler gösterilmiştir. Akım kısmında birimler arasındaki bağlantı işlemleri yer almaktadır. Graf kısmında ise oluşturulan sistemin grafının çıkartılması yapılmaktadır. Benzetim ve kontrol bölümünde ise oluşturulan sistemin benzetim ve kontrol işlemleri yapılmaktadır.



Tablo 2.1. Tipik bir SCADA sisteminden beklentiler. [1]

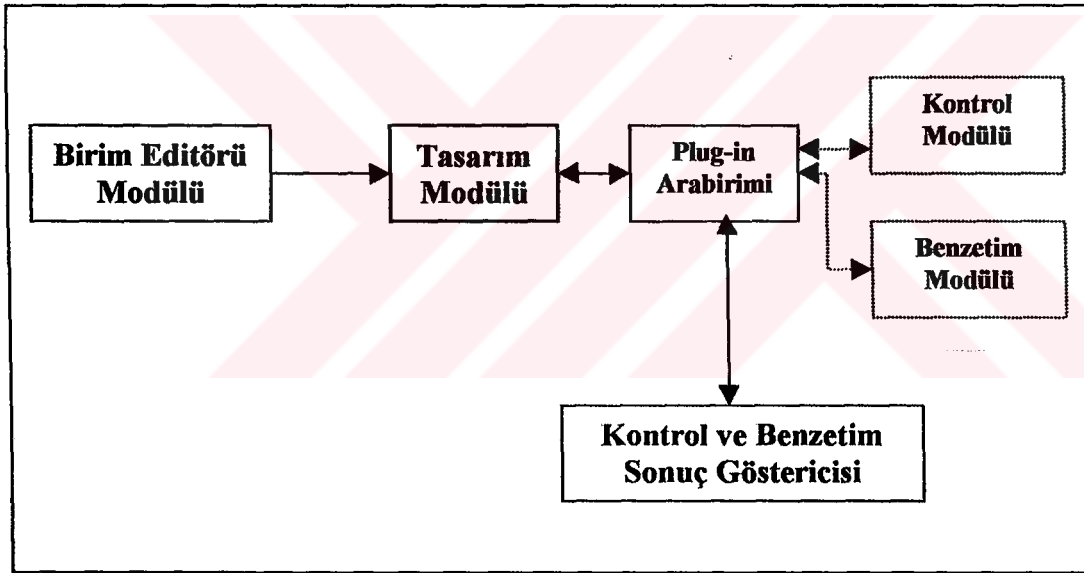
Tasarım	Birim	Görsel	Mantıksal
		Birim çizimleri kütüphanesinin oluşturulması	Birim argümanları dosyalarının oluşturulması
		Birimlerin görsel olarak sisteme dahil edilmesi	Birimlere isim verilmesi.
		Birimlerin silinmesi, taşınması, adlarının yazdırılması	Birimin özelliklerinin tayini
		Birim özellikleri penceresi	Özelliklerin gösterilmesi ve değişikliklerin yapılması
	Akım	Çalışma dosyası işlemleri	Çalışma dosyası işlemleri
		Akımların çizimi	Akımlara isim verilmesi
		Akımların silinmesi, adlarının ve yön oklarının gösterilmesi	Akım işlemleri
		Akım özellikleri penceresi	Özelliklerin gösterilmesi ve değişikliklerin yapılması
	Graf	Proses grafiğinin çizdirilmesi	
		İlişkisiz elemanların gösterilmesi	Proseste ilişkilerin testi
Benzetim ve kontrol		Proses resminin çizdirilmesi	
		Gösterge ekranının ayarlanması	Proseste yer alan birimlerin değişkenlerinden oluşan bağlantı dosyasının oluşturulması
		İterasyon sayısı ve sınırlarının belirlenmesi	Göstergede kullanılacak değişkenlerin seçilmesi
		Benzetim ve başlatılması	Hesaplama modülünün çalıştırılması
		Benzetim programının ürettiği verilerden seçilmişlerin anlık değerlerinin gösterilmesi	Benzetim ve/veya kontrolün sonu

3. SCADA PAKETİNİN OLUŞTURULMASI



Geliştirilen bu yazılım nesneye dayalı programlama teknikleri kullanılarak geliştirilmiştir. Tüm birimler ve akımlar farklı özellikleri olabilen nesneler olarak tasarlanmıştır. Böylece kodlama kısmı oldukça kolaylaşmıştır.

SİMKON modüler yapıda tasarlanmıştır. Paketi oluşturan modüllerin yaptıkları işler ve birbirleriyle iletişimi Şekil 3.1'deki gibi gösterilebilir. Birim modülünün çizdiği birimler tasarım modülünde kullanılarak bir sistem oluşturulmakta, plug in arabirimi ile kontrol ve benzetim modülleri ile ilişki kurularak sistemin benzetimi veya kontrolu yapılabilmektedir.



Şekil 3.1 SİMKON Yapısı

3.1. Pakette Kullanılan Algoritmalar

SİMKON'da tasarım ekranında gözüken her bir birim ve birimler arası bağlantıyı oluşturan akımlar aslında birer nesnedir ve bu nesnelerin kendine ait event'ları vardır. Bu nedenle tasarım programın genel döngüsü oldukça basittir. Sadece her bir döngüde tüm nesnelere parametre olarak mouse işlemleri ve tuş

kombinasyonları parametre olarak verilir. Temelde ana döngünün yaptığı tek iş budur. Her bir nesne kendi kendisini kontrol ederek gerekli tepkileri verir.



3.1.1. Programın Genel Döngüsü

SİMKON tasarım formunun genel döngüsü aşağıdaki gibi açıklanabilir.

1. Eğer fare hareket ettiyse farenin koordinatlarını yaz.
2. Nesne veya akım üzerindeki **konum kontrol algoritmasını(h, i)** çalıştır.
3. Tuşa basıldıysa **klavye kontrol algoritmasını(a)** çalıştır.
4. Fare tuşuna basılırsa **nesne kontrol algoritmasını(b)** çalıştır.
5. Fare kliklenirse, **tek-çift klik kontrol algoritmasını(c)** çalıştır.
6. Fare tuşu bırakılırsa **nesne kontrol algoritmasını(b)** çalıştır.
7. Gerekiyorsa menüden seçim yap.
8. 1'e git

Genel olarak program sürekli olarak yukarıdaki adımları kontrol eder. Gerekiyorsa sırasıyla aşağıda açıklanan algoritmalar çalıştırılır.

a. Klavye kontrol algoritması

Bu algoritma, klavyeden bir tuşa basıldığında çağrılır. Basılan tuş kodu değerlendirilir ve çağrılan noktaya geri döner.

1. Kısa yol tuşumu?
2. Eğer öyleyse ilgili kısa yolu çalıştır.
3. Algoritmadan geri dön.

b. Nesne kontrol algoritması

Form üzerinde fare tuşuna basıldığında devreye girer. Nesne taşınması veya bir dikdörtgen bölge seçilmesi konusunda karar verir.

1. Fare imleci bir nesnenin üzerinde mi?
2. Eğer öyleyse **taşıma algoritmasını(d)** çağır
3. Fare imleci nesne üzerinde değilse seçim yapılmak isteniyordur.
Bu durumda fare tuşu bırakılana kadar sürekli olarak fare koordinatını oku ve pmXor modunda basıldığı nokta ile fare imlecinin olduğu nokta arasında bir dikdörtgen çizdir.
4. Fare tuşu bırakıldığında seçme algoritmasını çalıştır.
5. Algoritmadan geri dön

c. Tek çift klik kontrol algoritması

Eğer bir nesne veya akım seçimi yapılmak istendiğinde ya da nesne veya akım özellikleri değiştirilmesi istendiği durumlarda bu algoritma çalıştırılır.

1. Eğer Timer başlatılmadıysa başlat, eğer daha önce başlatıldıysa bu bir çift click'tir, o zaman 7'ye git.
2. Timer kapanana kadar bekle.
3. Bu durumda tek click yapıldı, koordinatları al,
4. Eğer nesne çizim modundaysa **nesne ve akım oluşturma algoritmasını(e)** çalıştır.
5. Değilse alınan koordinatta nesne var mı? Eğer varsa ve seçili değil ise bunu seç, seçili ise seçimi kaldır ve 11. satıra git.
6. Alınan koordinatta akım var mı? Eğer varsa ve seçili değil ise bunu seç, seçili ise seçimi kaldır ve 11.satıra git.
7. 11'a git.
8. Çift click yapıldı, koordinatları al,
9. Alınan koordinatta nesne var mı? Eğer varsa o nesneye ait özellik tablosunu aç
10. Alınan koordinatta akım var mı? Eğer varsa o akıma ait özellik tablosunu aç
11. Algoritmadan geri dön

d. Taşıma algoritması

Bir nesne taşınmak istendiği zaman bu algoritma çalıştırılır. Nesne kontrol algoritması ile iç içedir.

1. Fare imlecinin üzerinde olduğu nesneyi, o nesnenin `hide()` metodunu kullanarak sakla.
2. Fare imlecinin bulunduğu noktaya bir `pmXor` modunda bir kare çiz.
3. Fare imlecinin bir önceki yerindeki kareyi, `pmXor` modunda tekrar çizerek sakla
4. Fare tuşu basılı ise 2'ye git
5. Bu durumda nesne yeni yerine sürüklenmiştir. Fare basılı olduğu anda nesne yerini gösteren kareyi sakla.
6. Nesne özelliklerinde sol ve üst değerlerini fare imlecinin bulunduğu noktaya göre ayarla.
7. Taşınmakta olan nesnenin `show()` metodunu kullanarak göster
8. Algoritmayı sona erdir.

e. Nesne ve akım oluşturma algoritması:

Programda çizim işlemleri için `TImage` nesnesi kullanılmıştır. Ekrana bir nesne yerleştirmek için soldaki `TListView` içinden yerleştirilmek istenen nesne seçildikten sonra fare imlecini ekranda nereye yerleştirmek isteniyorsa oraya tıklatarak yerleştirilebilir. Ayrıca iki nesne arasında bağlantı yapılmak isteniyorsa yine bu algoritma kullanılır. Burada kullanılan algoritma aşağıdaki gibidir:

1. Nesne seçimi yapıldı mı?
2. Evetse 5'e git
3. Akım çizdirilmek mi isteniyor?
4. Evetse 7'ye git.
5. Fare imlecinin bulunduğu noktayı al ve varsayılan nesne özelliklerini nesneye atarak nesneyi oluşturun.
6. 9'a git
7. Basılan yerde nesne varsa nesneyi seç
8. Eğer ikinci nesne seçildiyse seçili nesneler arasında akım oluşturun.
9. Algoritmayı sona erdir.

f. Akım çizdirme algoritması

Akım, iki nesne arasındaki ilişkiyi gösterir. C++ dilinde bir class olarak tanımlanmıştır. Bir akım sınıfı göstermek veya saklamak için `show()` ve `hide()` isimli

metodlar kullanılır. Bunlar içinde akımın çizim algoritması vardır. Akım çizilirken L şeklinde bağlantı yapılacak şekilde çizilir. Böylelikle akımların karışması engellenir.

Bir akım gösterilmek veya saklanması için yani *show()* ve *hide()* arasındaki tek fark akım çizgisinin rengi nedeniyle ortaya çıkan gösterme ve saklama şeklindedir. Bu iş için kullanılan algoritma ise şöyledir :

1. Birinci nesne koordinatlarını ve büyüklüğünü al,
2. İkinci nesne koordinatlarını ve büyüklüğünü al,
3. x1, y1, x2 ve y2 tanımla
4. x1'e solda olan nesnenin sol+en+tolerans değerini ata
5. x2'ye sağda olan nesnenin sol-tolerans değerini ata
6. y1'e üstte olan nesnenin üst+boy+tolerans değerini ata
7. y2'ye altta olan nesnenin üst-tolerans değerini ata
8. L şeklindeki bağlantının ilk kısmı, (x1,y1)-(x1,y2) noktaları arasında doğru çiz.
9. L şeklindeki bağlantının ikinci kısmı, (x1,y2)-(x2,y2) noktaları arasında doğru çiz.
10. Akım yönüne göre okları çizdir
11. Algoritma sonu

g. Nesne çizim algoritması

Dosyada vektör olarak saklanan nesneleri çizdirmek için birkaç değişik algoritma kullanılır. Bunlar vektör olarak tanımlı resmi bitmap'e çevirme, büyütme, çevirme gibi algoritmaların kullanılması gibi.

Aşağıdaki algoritma ile vektör olarak tanımlı nesne resmi bitmap formatına çevrilir:

1. Bellekte TBitmap tipinde bir alan oluştur.
2. Nesne tipindeki vektör sayısı kadar dön
3. Vektör tipini ve özelliklerini al.
4. Bitmap belleğe bu vektörü çiz.
5. Eğer vektör sayısı bittiye bellekteki resmi ImageList'e atılır. Böylece vektör tabanlı resimler bitmap'e dönüşüp ListView'da kullanılabilir.
6. Algoritma sonu.

Çizim formu üzerine bir nesne çizdirilmek isteniyorsa **nesne ve akım oluşturma algoritması(e)** çağrılır. Bu algoritmanın çalıştırılması sırasında nesnenin show() ve hide() metodlarından faydalanılır. Temel olarak show() ve hide() metodları arasında birisinin gösterip diğerinin saklamasından başka, yani renk değişikliğinden başka fark yoktur. Show() ve hide() metodu aşağıdaki algoritmayı kullanır:

1. Nesne tipine göre tipler dizisindeki vektör tabanlı resmi al.
2. $x1=sol+vector[i].x1 * zoomx$
3. $y1=ust+vector[i].y1 * zoomy$
4. $x2=sol+vector[i].x2 * zoomx$
5. $y2=ust+vector[i].y2 * zoomy$
6. Buraya kadar nesnenin büyütme oranına göre vektör büyütülüp form üzerine yerleştirilmeye hazır hale geldi. Şimdi nesnenin kaç derece çevrileceğini tutan rotation değişkenindeki değere göre $x1,y1,x2$ ve $y2$ noktalarının yerleri değiştirilir
7. Çevrilme ve büyütme işlemleri bitti, vektörü ekranda oluştur.
8. Vektör sayısı kadar 2'ye git
9. Algoritma sonu

h. Akım tutma algoritması

Bu kısım ile tasarım ekranında nesneleri birbirine bağlayan akımların tutulabilmesi için gerekli algoritma açıklanmaktadır. L şeklinde bağlantı yapıldığı için tutma algoritması oldukça kısadır. Programın geliştirilmesinin ilk aşamalarında L şeklinde bağlantı yerine doğrudan bağlantı yapıldığında seçme ve tutma kısmı için $(x1,y1)-(x2,y2)$ arasındaki doğruya uzaklığını hesaplayıp tolerans içindeki uzaklıklar için tutulma işlemi yapılmaktaydı. Oysa L şeklindeki bağlantıda aşağıdaki algoritma yeterli olmaktadır :

1. Fare imlecinin koordinatını al.
2. Birinci nesne koordinatlarını ve büyüklüğünü al,
3. İkinci nesne koordinatlarını ve büyüklüğünü al,
4. $x1, y1, x2$ ve $y2$ tanımla
5. $x1$ 'e solda olan nesnenin $sol+en+tolerans$ değerini ata
6. $x2$ 'ye sağda olan nesnenin $sol-tolerans$ değerini ata

7. y_1 'e üstte olan nesnenin üst+boy+tolerans değerini ata
8. y_2 'ye altta olan nesnenin üst-tolerans değerini ata

Şimdi $(x_1, y_1)-(x_1, y_2)$ ve $(x_1, y_2)-(x_2, y_2)$ noktaları arasında yer alan iki tane doğru var. Bu doğrular sırasıyla x ve y eksenine paralel için aşağıdaki kısım tutulma işlemini yapar.

9. Fare imlecinin koordinatı $(x_1, y_1)-(x_1, y_2)$ aralığında ise tutulma başarılı, 11'e git
10. Fare imlecinin koordinatı $(x_1, y_2)-(x_2, y_2)$ aralığında ise tutulma başarılı, 11'e git
11. Tutulma değerini geri döndürerek algoritmayı sonlandır

Yukarıdaki algoritma aynı zamanda akım sınıfının OnMouse() isimli metodunda da kullanılmaktadır. Şu farkla ki;

1. Fare imleci koordinatını alarak tutma algoritmasını çağır,
2. Eğer tutulma başarılı ise fare imleci akım üzerindedir. Akımın rengini değiştir
3. Eğer tutulma başarısız ise ve daha önce fare üzerinde ise akım rengini tekrar siyah olarak yap
4. Algoritmadan çık

i. İpucu algoritması

Bir nesne üzerinde fare imleci 2 saniyeden fazla durursa, ekranda bir kutu içerisinde o nesnenin ismi yazılır. Ayrıca fare imleci, nesnenin kenarlarına geldiğinde, geldiği kenara göre “\”, “/”, “|”, “—” ve “+” şekillerinden uygun olanı atanır. Algoritması şöyledir.

1. Fare imlecinin nesne üzerindeki yerine göre fare imlecinin şeklini değiştir ve HintTimer'ı çalıştır.
2. Timer dolduğunda eğer hala nesne üzerindeyse Edit3 nesnesinin Text özelliğine nesne ismini yaz ve nesneyi göster.
3. Eğer fare tuşuna basıldıysa Edit3 nesnesini sakla.



3.2. Veri yapıları

SİMKON geliştirilirken kullanılan yapıları nesnelist isimli birim listesi, akımlist isimli akım listesi, tip listesi ve diğer yapılar olarak dört kısımda incelemek mümkündür :

3.2.1. Nesnelist Dizisi

Bu tüm birimlerin listesinin yer aldığı bir dizidir. Her bir elemanı bir nesnedir. Yeni bir eleman eklendiğinde, bellekte bu eleman için yer ayarlanır ve ilk atamalar yapılarak diziye eklenir. Bir eleman silinmek istendiği zaman önce o elemana bağlı olan tüm akımlar silinir. Sonra eleman diziden çıkarılır ve o eleman için ayrılan bellek hemen boşaltılır.

Tanımlanış şekli ve her bir elemanın ne işe yaradığı aşağıdaki şekildedir:

Tablo 3.1. Nesne Tanımlamaları

```
class nesne {
int sol, ust;           // Soldan ve üstten uzaklığı, pixel cinsinden.
int en, boy;           // En ve boyunun uzunluğu, pixel cinsinden
int sec;              // secili olup olmadığını tutan değişken
float zoomx, zoomy;    // 0..1 küçültme, 1..n büyütme
int rotation;         // 0°..360° döndürme açısı.
public:
char *name;           // nesnenin ismi
char *type;           // nesnenin tipi
char *ozellik[max_ozellik]; // nesnenin özelliklerini tutan değişken
void select();        // nesne kendisini seçer
void unselect();      // seçimi kaldırır.
int selected(int x, int y, int mousestat); // seçilme kontrolünü yapar
int OnMouse(int x,int y); // Fare imleci kontrolünü yapar
void hide();          // nesne kendisini saklar
void show();          // kendisini gösterir
void create(int sol_, int ust_, int en_, int boy_); // nesne kendini oluşturur.
void setprop(int hangisi,char *_newprop); // özellik değerlerini atar.
};
```

nesne nesnelist[MAX_OBJECT+1]

// nesne grubu tanımlaması



3.2.2. Akımlist Dizisi

Akımlarla ilgili yapıların tutulduğu bir dizidir. Her bir elemanı akım isimli bir class olarak tanımlanmıştır. Yeni bir akım oluşturulduğunda sayac.akım isimli değişken değeri bir arttırılır ve yeni bir nesne oluşturularak diziye eklenir. Bir akım silindiği zaman sayac.akım isimli değişken değeri sabit tutulup, silinecek akımın silindi özelliği true yapılır. Böylece bellekten 200 byte boşa harcanırken hızdan kazanılır. İlk kayıt işlemi sırasında silinen nesne listeden tamamen çıkarılır.

Tanımlanış şekli ve her bir elemanın ne işe yaradığı aşağıdaki şekildedir:

Tablo 3.2 Akım Tanımlamaları

```
class akimlar {
    char *name;           // Akımın kendi ismi
    char *name1;          // Bağlı olduğu ilk nesne ismi
    char *name2;          // Bağlı olduğu ikinci nesne ismi, akım yönü 1'den ikiye
    int secili;           // Seçili olup olmadığını tutan değişken
    int yon;              // Akım yönü,
    int ozelliksayisi;    // Akımın özellik sayısı
    int silindi;          // Bir sonraki düzenlemeye kadar akımı saklar
    int mavi;             // Mouse üzerinde iken renklendirmeyi yapar
    char *ozellik[max_ozellik]; // Akımın özellik sayısı kadar özelliği
    int OnMouse(int Mx,int My); // Fare imleci üzerinde mi kontrolünü yapar.
    void select();        // Kendisini seçer.
    void unselect();      // Seçimi kaldırır
    void create(const char _name1, const char _name2,int _yon); // kendisini oluşturur.
    void selected(int x, int y, int mousestat); // seçilme kontrolü yapar.
    void show();          // kendisini gösterir
    void hide();          // kendisini saklar
};
```

Akimlar akimlist[MAX_OBJECT+1]; // Akımlar için dizi

3.2.3. Tipler Listesi

Nesne tipleri ile ilgili verilerin tutulduğu listedir. Program ilk çalıştırılmaya başlanırken nesne dosyası(5.3) içinden nesne tipleri okunarak bu listeye yüklenir. Liste sürekli bellekte tutulur ve her bir nesne çizim işlemi sırasında şekli bu listeden alınır. Ayrıca bir nesne oluşturulmak istendiğinde tipi bu listedeki tipler arasında bulunur ve özellikleri bu listeden alınır.

Tanımlanış şekli ve her bir elemanın ne işe yaradığı aşağıdaki şekildedir:



Tablo 3.3. Nesne tipi tanımlamaları

```
struct nesnetip
{
    char *name;           // tipin ismi
    int kactane;           // kaç özelliği var
    char *ozellik[ max_ozellik ]; // özelliklerin listesi
    int vektorsayisi;      // kaç vektörden oluştuğu
    vector icon[max_icon]; // vektör listesi
};

struct vector{
    int x1, y1, x2, y2;    // vektör koordinatları
    int type;              // vektör tipi, daire, çizgi, elips vs.
    TColor renk;           // vektör rengi
};

nesnetip tipler[MAX_TIP+1]; // Kullanılan nesneleri tutan dizi;
```

3.2.4. Diğer Yapılar

Programda yukarıdaki yapıların dışında yardımcı global değişkenler ve yapılar kullanılmıştır. Bunlar genel olarak şöyle gösterilebilir.

Tablo 3.4. Diğer değişkenler

```
struct objcounter{
    int resist;           // nesne listesi için sayaç
    int akim;             // akım listesi için sayaç
};

Objcounter sayac;        // nesne ve akım için sayaç değişkeni
int Ciz=Hicbiri;         // Çizim formunda o andaki çizim modu
int Baglanti=0;          // akım çizdirme modunu gösterir
int secant=0;            // seçili nesne sayısını tutar
int MoveStart=0;         // Nesne taşınma durumunu tutar, 0'dan farklı ise taşıma modunda
int nesnecopy[MAX_OBJECT]; // kes, kopyala ve yapıştır işlemleri için gereken diziler
int SecimYapiliyor=0;    // Dikdörtgen bölge seçimi yapıyor
//Nesne için yeniden boyutlandırma ve taşıma için gereken değişkenler ve sabitler
int Tolerans=2;          // Tutma toleransı piksel cinsinden
int Uzerinde=1;          // OnMouse için fare nesne üzerinde
int SolUstSagAlt=2;      // imleç sol-üst sağ-alt köşede
int SolAltSagUst=3;      // imleç sol-alt sağ-üst köşede
int SagSol=4;            // imleç sağ-sol kenarlarda
int UstAlt=5;            // imleç üst-alt kenarlarda
int akimlarin_ozelliksayisi=0; // Her bir akımda olabilecek özelliklerin listesi
char *akimozellik[max_ozellik]; // ve sayısını tutan liste ve değişkenler.
// bu liste program başlatılırken akim.dat dosyası içinden okutulur
int MouseBasildi=0;      // Mouse tuşuna basıldıysa 1, değilse 0 değerini alır.
char *NesneDosyasi;      // INI dosyasından okunan nesne dosyası
char *AkimDosyasi;       // INI dosyasından okunan akım dosyası
```

Yukarıdaki tabloda gösterilen global değişkenlerin yanı sıra program içinde sabit olarak tanımlanmış bazı değerler vardır. Bu sabitler programın limitlerini belirler. Örneğin programın tanıyabileceği maksimum tip sayısı 1000 tane'dir. Bu sayıyı istediğimiz kadar arttırıp azaltabiliriz. Bu sabitleri sınırlayan tek sorun bellek kapasitesidir.

Tablo 3.5. Sabitler

const MAX_OBJECT=5000; const MAX_TIP=1000; const Hicbiri=32000; const Birinci=0; const Ikinci=1; const Ucuncu=2; const Dorduncu=3; const Besinci=4; const Altinci=5; const MaxTextSize=20; const max_ozellik=20; const max_icon=40;	<i>// Vektör çizimi için gerekli tanımlamalar.</i> const Cizgi=1; const Elips=2; const Dikdortgen=3; <i>//const debug=0;</i> <i>//Klavye kodları....</i> const sc_delete=46; <i>//Akım yönü ile ilgili değişkenler...</i> const ayYonsuz=1; const ayCiftYon=2; const ayTekYon=3;
--	---

3.3. Dosya Yapıları

SİMKON'da beş tür kayıt şekli kullanılmaktadır.

1. Ayar dosyası – SİMKON.INI
2. Nesne dosyası,
3. Akım dosyası,
4. Kayıt dosyası(sistem graf çıktısı) yapılan çizimlerin tutulduğu dosya,
5. Benzetim sonuç dosyası,

Bütün bu dosyalar SİMKON'un doğru bir şekilde çalışabilmesi için gereklidir. SİMKON ile bir tasarım yapılabilmesi için en az bir nesne dosyasının oluşturulması gerekmektedir. Kullanılan dosyaların açıklamaları aşağıda belirtilmiştir:

3.3.1. Ayar Dosyası

Bu dosya adından da anlaşılacağı üzere program ile ilgili genel ayarlamaları tutar. Yapısı text tabanlıdır. İçinde 1. Satırda nesne dosyasının tam ismi, ikinci satırda akım dosyasının tam ismi olmak üzere iki veri bulundurulmaktadır. Bu dosyaya kullanıcı ayarları(araç çubuğu yeri, plugin ayarları) eklenerek programın özelleştirilmesi sağlanabilir.

3.3.2. Nesne Dosyası Yapısı

Bu dosya program ilk çalıştırıldığında ayar dosyasındaki konumuna göre otomatik olarak yüklenir. Dosyada bulunan // ile başlayan kayıtlar açıklama amacı ile kullanılır. Yapı oldukça esnek tasarlanmıştır. Bu sayede dosyada bulunan nesne yapıları ve özellikleri farklı sayıda olabilir. Dosyadaki nesne sayıları ve özellikleri ile ilgili tek sınırlama bellek miktarıdır. Ortalama olarak her bir kaydın yaklaşık 200 byte yer kapladığı düşünülürse 5000 değişik nesne için sadece belleğin 1 MB'lık kısmı yeterli olur.

Dosyadan ilk okunan kayıt dosyadaki birim sayısını verir. Bundan sonra nesne sayısı kadar aşağıdaki tipte kayıt bulunur:

- Nesne adı
- Özellik sayısı,
- Özellik sayısı kadar özellik ismi,
- Vektör sayısı,
- Vektör sayısı kadar aşağıdaki şekilde vektör bilgisi bulunur.
 - Vektör tipi (çizgi, dikdörtgen, elips vs.)
 - x1, y1, x2, y2 koordinatları
 - Rengi.

Tablo 3.6. Örnek Birim Dosyası

8 //birim_sayısı	çizgi 11 5 13 19 siyah	çizgi 8 5 8 19 mavi
Direnç //ilk_birim_adı	çizgi 13 19 15 5 siyah	çizgi 13 11 13 11 mavi
1 //birim_özellik_sayısı	çizgi 15 5 17 19 siyah	çizgi 13 12 24 12 mavi
Miktar //birim_özellik_ismi	çizgi 17 19 19 5 siyah	çizgi 13 19 13 5 mavi
11 //birim_vektör_sayısı	çizgi 19 5 20 12 siyah	
çizgi 0 12 3 12 siyah	çizgi 20 12 23 12 siyah	
çizgi 3 12 5 19 siyah	Kondansatör	
çizgi 5 19 7 5 siyah	1	
çizgi 7 5 9 19 siyah	Farad	
çizgi 9 19 11 5 siyah	çizgi 0 12 8 12 mavi	

3.3.3. Akım Dosyası Yapısı

Bu dosya program ilk çalıştırıldığında ayar dosyasındaki konumuna göre otomatik olarak yüklenir. Dosyada bulunan // ile başlayan kayıtlar açıklama amacı ile kullanılır. Dosyanın yapısı oldukça esnek ve basittir :

- Dosyanın ilk satırı özellik sayısını verir
- Bundan sonra dosyada özellik sayısı kadar özellik ismi vardır.

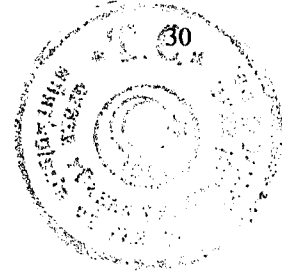
Tablo 3.7 Örnek Akım Dosyası

2 //Akımın_özellik_sayısını_verir. Direnç //Akımın_yoğunluğunu_verir.
--

3.3.4. Kayıt Dosyası Yapısı

Bu dosya yapısı yapılan çizimlerin saklanması amacıyla kullanılır. Yapısı oldukça esnektir. Ayrıca az yer tutar, her bir nesne ve akım başına ortalama 200 byte yer tutar. Bu durumda 6000 akım ve 6000 nesneden oluşan bir sistem için gereken disk ve bellek kapasitesi yaklaşık 1 MB' tır.

Bu kayıt yapısında dosyadaki ilk kayıt, dosyada bulunan nesne sayısını verir. Bundan sonra nesne sayısı kadar aşağıdaki kayıtlar bulunur :



- Nesne adı
- Nesne tipi
- Sol ve üstten uzaklıkları,
- En ve boyu,
- Çevirme açısı,
- X ve Y büyütme oranları,
- Ait olduğu tipin özellik sayısı kadar özellikleri

Dosyadan nesneler okunduktan sonra sıra akımların okunmasına gelir. Bunun için nesne yüklenme kısmı bittikten sonra gelen kayıt dosyadaki akım sayısını verir. Akım sayısı kadar aşağıdaki bilgiler okunur :

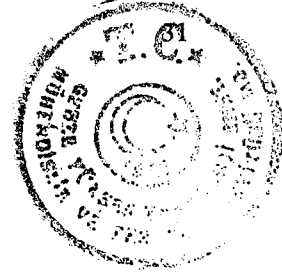
- Akım ismi
- Kaynak akım
- Hedef akım

Tablo 3.8. Örnek Kayıt Dosyası

5
Direnç1 Direnç 56 265 24 24 0 2.000000 2.000000 250
Direnç2 Direnç 177 436 24 24 90 2.000000 2.000000 250
Ampermetre3 Ampermetre 300 260 24 24 0 2.000000 2.000000
Voltmetre4 Voltmetre 165 574 24 24 90 2.000000 2.000000
Alternatif5 Alternatif 166 107 24 24 90 2.000000 2.000000 100 50 ...
5
1
Akım1 Direnç1 Alternatif5
1 Boş
Akım2 Ampermetre3 Alternatif5
1 Boş
Akım3 Ampermetre3 Direnç2
1 Boş
Akım4 Voltmetre4 Direnç2
1 Boş
Akım5 Direnç1 Direnç2
1 Boş

3.4. Yazılımın Geliştirilmesi

SİM KON, C++ dili kullanılarak yazılmıştır. Yazılımın geliştirildiği ortam ise aşağıdaki gibidir :



- Borland C++ Builder 5.0
- Windows 98 SE
- 64 MB RAM
- Intel Pentium 200MMX
- 2 GB HDD

SİMKON'nın çalıştırılabilmesi için ise gereken minimum konfigürasyon aşağıdaki gibidir.

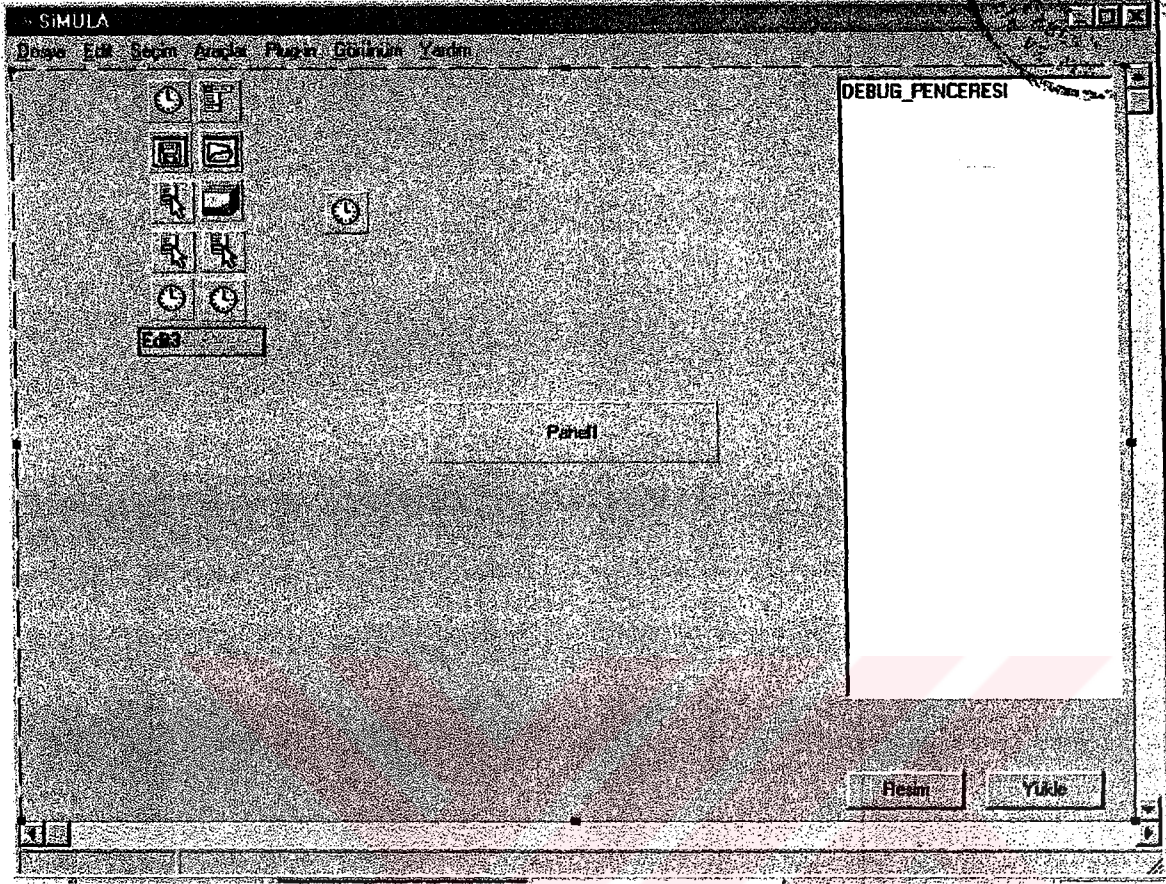
- 16 MB RAM
- Windows 95 ve üstü
- 20 MB boş harddisk alanı
- Pentium 100 ve üzeri işlemci.

SİMKON'da tüm modülleriyle beraber yaklaşık 4000 satır yazılmıştır. C++ getirdiği bir avantaj olan object oriented programlama tekniği kullanılmıştır. Windows içinde bulunan nesnelerin haricinde tasarım sırasında ekranda gözüken her bir birim programda bir nesne olarak yer almaktadır ve yine birim arasındaki bağlantıları gösteren akımlar da birer nesnedir.

Programda yer alan başlıca elemanlar ve ne işe yaradıkları :

3.4.1. Çizim Formu

Tüm nesne tasarım ve serbest el çizim işlemlerinin yapıldığı yerdir. TForm tipinde olan çizim formu üzerine TImage tipindeki Image1 nesnesi üzerinde çizim işlemleri gerçekleştirilir. Şekil 3.2'deki gibidir. Ayrıca bu form içinde bir çok nesne vardır. Bu nesnelerin özellikleri ve ne işe yaradıkları aşağıdadır :



Şekil 3.2. Tasarım Sırasında Çizim Formu ve İçerdiği Komponentler.

3.4.1.1. Timer1

Tipi TTimer'dır. Yaptığı iş ise çift tıklama kontrolünün sağlanmasıdır. Bir kez tıklandığında Timer başlatılır, belli bir süre içinde iki kez tıklandıysa programın çalışmasını çift tıklama kısmına yönlendirir. C++ Builder'da hiçbir işlem yapmadan OnClick() ve OnDblClick() event'larına bir atama yaptığımızda, OnDblClick() event'ı hiçbir zaman işletilmiyor. Hem tek tıklamayı, hem de çift tıklamayı sağlamak için parametre kontrolü yapılabilir. Ancak Timer kullanarak çift tıklama kontrolü yapılması çift tıklama hassasiyetini Windows'tan bağımsız hale getirebiliyor. Böylelikle kullanıcı program içinde ayarlamasını yaparak çift tıklama aralığını kendisi belirleyebiliyor.



Çalışma şeklini şöyle ifade etmek mümkündür : Fare bir kez tıklanıldığında OnClick() hemen event'ı devreye girmeyip Timer1 aktif hale getirilir. Timer1 belirtilen süre sonunda kendiliğinden pasif hale geçiyor. Eğer Timer1 pasif hale gelmeden bir kez daha tıklama olursa OnDblClick() event'ına atlanıyor. Eğer Timer1 pasif hale gelmeden fare tuşuna tıklanmazsa bu bir tek tıklamadır ve OnClick() işletilir.

3.4.1.2. Timer2

TTimer tipindedir ve hata ayıklama (debug) amacıyla kullanılmıştır. Program yazılması sırasında istenilen değişkenlerin o anki değerleri Memo1 içine yazdırılır.

3.4.1.3. HintTimer

TTimer tipindedir. Bir nesne üzerinde fare imleci 2 saniyeden fazla durursa fare imlecinin üzerindeki nesne ismi bir hint kutusuna yazdırılır.

3.4.1.4. Timer3

TTimer tipindedir. CizimFormu, ToolBarForm üzerindeki nesneleri kullandığı için ToolBarForm'dan daha sonra oluşturulması gerekmektedir. Bu nedenle ToolBarForm tamamen oluşturulmadan CizimFormu gösterilmemelidir. Bu timer bu gecikmeyi sağlar.

3.4.1.5. OpenFileDialog ve SaveDialog1

Yapılan çizimin dosyaya kaydedilmesi ve dosyadan yüklenirken izin ve dosya ismi seçimi amacıyla kullanılmıştır. TOpenDialog ve TSaveDialog tipindedirler.

3.4.1.6. ImageList1

Nesneler dosyada aşağıdaki şekilde vektör olarak saklanmaktadır.

çizgi 1 1 20 20 kırmızı

çizgi 20 1 1 20 yeşil

Program her çalıştırıldığında tüm nesnelerin vektör resimleri bitmap hale getirildikten sonra bu listede tutulur. Bu liste ile ListView1 nesneleri ilişkilendirilmiştir. ListView1 nesnesi düğmelerin resimlerini bu listeden okur.

3.4.1.7. MainMenu1

Programda kullanılan ana menü için gereken nesnedir. Tipi TMainMenu olarak tanımlanmıştır

3.4.1.8. PopupMenu1, PopupMenu2 ve PopupMenu3

Hepsi TPopupMenu tipindedir. Programda sağ tuşa basılınca çıkan menülerdir. PopupMenu1, ListView üzerinde, PopupMenu2 plugin işlemlerinde ve PopupMenu3 Image1 nesnesi üzerinde kullanılırlar.

3.4.1.9. Memo1

TMemo tipindedir. Program çalışması sırasında oluşan uyarı ve açıklama mesajları bu memo içinde yer alır. Debug amacıyla kullanıldığından normal program işleyişi sırasında gözükmez.

3.4.1.10. Edit3

TEdit tipindedir. HintTimer ile ortak çalışır. Bir nesne üzerinde 2 saniyeden fazla bekletildiğinde Edit3, fare imlecinin bulunduğu yere konumlandırılır ve üzerine nesne ismi yazılır.

3.4.1.11. StatusBar1

TStatusBar tipindedir. Burada programın işleyişi sırasında kullanıcıya mesaj vermek amacıyla tasarlanmıştır. Burada yazan mesajlar programın kullanılmasını kolaylaştırır. Ayrıca sürekli olarak fare imlecinin koordinatları yazılır.

3.4.1.12. Image1

Tasarım kısmında yapılan tüm çizim işlemleri bu nesne üzerinde yapılır. Çizim kısmında çizimlerin yapılma şekli açıklanacaktır.

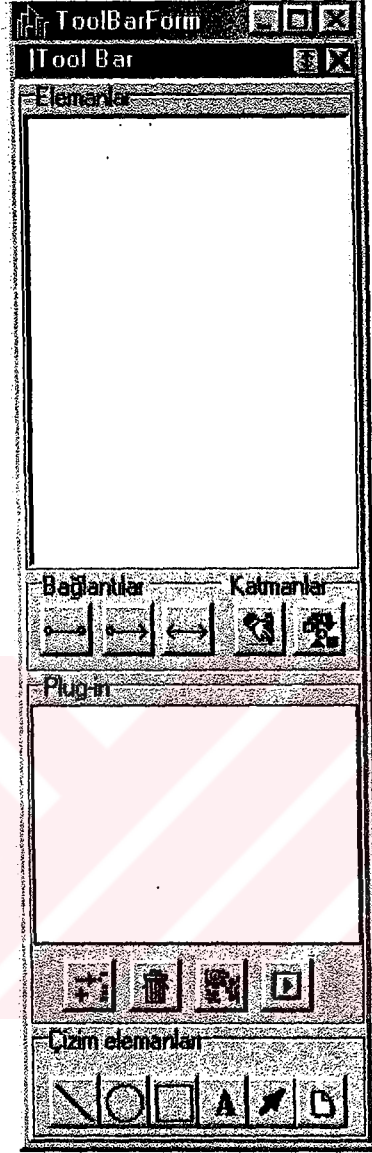
3.4.2. ToolBar

Tüm tasarım işlemlerinin yapıldığı araç çubuğudur. Kullanıcının isteklerine ve kullanım alışkanlıklarına göre bu araç çubuğunu ekranın istediği yerine getirebilir. Ayrıca taşınması için herhangi bir yerinden tutularak taşınabilir. Yine çizim sırasında gerekli olmayan düğme veya seçenek gruplarını kapatarak ekranı daha verimli kullanabilmeyi sağlar. Bu araç çubuğunun yapılması için form üzerine iki TPanel nesnesi yerleştirildi. Bunlardan birisinin ekranın üst tarafında ince bir bar halinde, diğeri tüm form boyunca yerleştirildi. Böylelikle Windows standardı daha işlevsel bir hale getirilerek esnek bir araç çubuğu tasarlanmış oldu (Şekil 3.3).

Araç çubuğunda her bir bölüm için bir TGroupBox nesnesi bulunmaktadır ve buna çift tıklandığında boyu küçültülüp büyütülebilmektedir. Araç çubuğunda yer alan nesneler şunlardır.

3.4.2.1. Elemanlar

Bu kısım tasarım sırasında nesne çizimi için kullanılır. TGroupBox içine TListView kullanılarak yapılmıştır. TListView, vektör tabanlı çizimleri desteklemediğinden program çalışmaya başlarken, SİM KON.INI dosyası içinde yer alan birim dosyasının yerini alır ve bu birim dosyası içinde vektör olarak tanımlı olan nesne şekillerini, bitmap formatına dönüştürerek buraya ekler. Bitmap dönüştürme için bellekte TBitmap tipli ve boyutları 24x24 olan bir alan tanımlanarak bu alanda vektörel çizim yapılarak bitmap haline getirilmiştir.



Şekil 3.3. Tasarım Sırasında Araç Çubuğu Formu ve İçerdiği Komponentler.

3.4.2.2. Bağlantılar, Katmanlar

Bir TGroupBox içinde TSpeedButton nesneleri kullanılarak yazılmıştır. Çizdirilecek akımın yönüne göre üç farklı düğme vardır. Bunlar yönsüz, tek yönlü ve çift yönlüdür. Ayrıca burada bulunan düğmeler aracılığı ile serbest çizim modundan, tasarım moduna geçiş yapılabilmektedir.

3.4.2.3. Plug-in

Bu kısımda programa eklenebilecek plugin'lerin listesi yer almaktadır. Bu kısma sağ tuşla tıklanarak yeni plugin eklenebilir. Plugin'lerin kullanılmak istenmesinin nedeni, ileride benzetim, test vs. gibi kısımlar yazılmak istendiğinde program kodunun değiştirilmesine gerek olmadan sadece bu kısma sağ tuşla tıklanarak gelen menüde ekle seçilerek ilgili plugin çalıştırılabilir. Böylelikle sadece benzetim işini yapan ve sonucu üreten küçük programlar yazılarak bir sistemin tasarımı ve benzetimi kolaylıkla yapılabilir.

3.4.2.4. Çizim elemanları

Bu kısımda serbest el çizimlerinin yapılabilmesi için gereken düğmeler bulunur. TGroupBox nesnesi içinde yer alan düğmelerden oluşur. Bu düğmeler TspeedButton türündedir ve bir seferde sadece bir tanesi seçilebilmektedir.

3.4.3. Birim Editörü

Programın temel bileşenlerinden birisidir. Programda kullanılan birimlerin çizilmesini ve her bir birimin özelliklerinin belirlenebilmesini sağlar. Bu program vektör tabanlı çizimler üretir. Ürettiği vektörler 25x25'lik bir uzayda gösterilirler. Birimlerde vektörler kullanıldığı için büyütme, küçültme ve döndürme işlemlerinin oldukça rahat bir şekilde yapılabilmesine imkan sağlamaktadır (Şekil 3.4).

Oluşturulan birimler için her türlü özellik tanımlaması da burada yapılır. Burada kaydedilen birimler Çizim Formu olarak adlandırılan tasarım formunda şekil olarak gösterilirler ve ayrıntı tablosunda birim için gösterilecek değerler burada belirlenir. Bir birim için yeni özellik tanımlanmak istediği zaman veya var olan özelliklerinden biri değiştirilmek isteniyorsa yine bu editörden faydalanılabilir.

Bu editör text formatında dosya üretir. Ürettiği dosyaların formatı 3.3.2 Nesne Dosyası isimli bölümde anlatılmaktadır. Eğer istenirse notepad gibi herhangi bir text editörü de kullanılarak birimler üzerinde işlem yapılabilir.

Kullanılan temel nesneleri ise aşağıdaki gibidir.

3.4.3.1. Image1

Tipi TImage olarak tanımlanmıştır. Nesne çizim işlemlerinin yapıldığı yerdir. 24x24 boyutundaki nesne şekli büyütülerek bir ızgara haline getirilmiştir. Image üzerindeki 4x4 boyutundaki her bir kutu aslında gerçekte bir pikseli temsil etmektedir. Çizimler vektörel olarak gösterildiği için pikseller bölünüyormuş gibi bir izlenim oluşturabilir. Ancak bu vektör tabanlı zoom yapıldığı için kalite kaybı olmadan büyütülmesinden kaynaklanmaktadır. Fare event'larına yazılan kodlar sayesinde çizim yapar.

3.4.3.2. Panel1

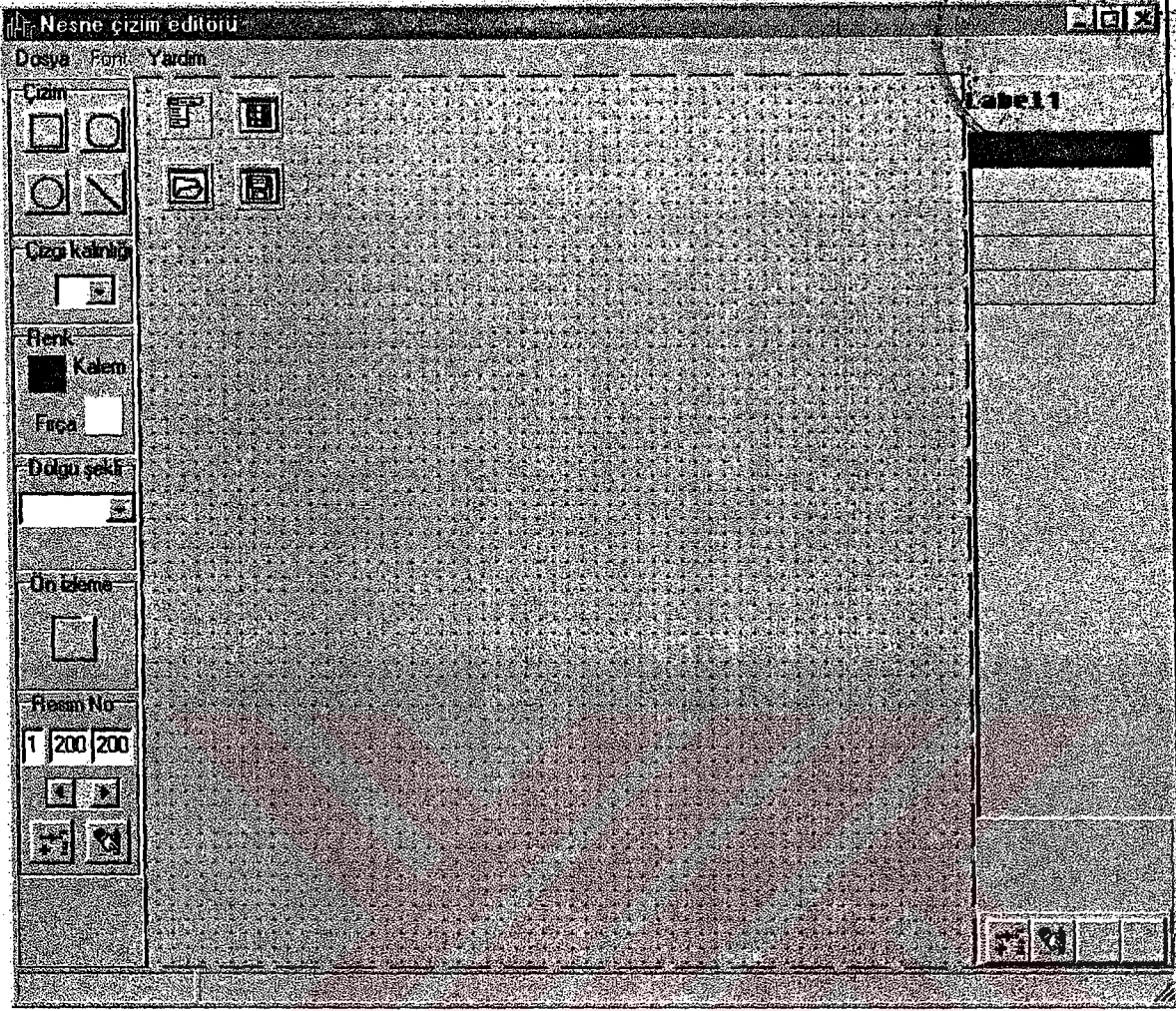
Tipi TPanel olarak tanımlanmıştır. Üzerinde çizim şeklini değiştirmeye yarayan düğmeler bulunur. Ayrıca çizilen şeklin gerçek boyutlardaki görünümü Image2 üzerinde anında gösterilmektedir.

3.4.3.3. Panel5

Sağ taraftaki paneldir. Üzerinde bulunan StringGrid1 ile nesne özellikleri işlemleri yapılabilir. Panelin alt tarafında bulunan Speed Button grubu ile oluşturulmak istenen birime yeni özellik eklenebilir.

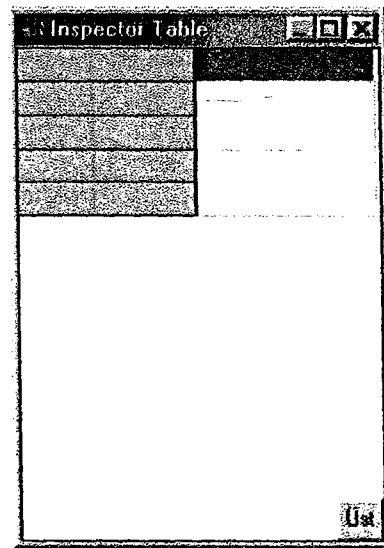
3.4.4. Özellik Tablosu

Sağdaki şekilde seçilen nesne veya akım için, seçilen akım veya nesnenin özelliklerini gösteren ve bu özellikleri değiştirebilmeye yarayan formun tasarım aşamasındaki şekli görülmektedir. Burada soldaki sütun sadece okunabilir olup, nesnenin veya akımın özelliklerinin ismini tutmaya yarar. Sağdaki sütunda ise yapmak istediğimiz değişiklikler yazılabilir (Şekil 3.5).



Şekil 3.4. Birim Editörünün Tasarım Sırasındaki Görüntüsü.

Form dialog tipinde hazırlanmıştır. Form üzerine bir TStringGrid nesnesi yayılmıştır. Form'un OnActivate() event'ı sırasında eğer nesnenin özellik sayısı form içine sığmıyorsa bir adet ScrollBar eklenmesini sağlar. OnDeactivate() ise CizimFormunda yer alan refresh() fonksiyonunu çağırarak değiştirilen özelliklerin aktif hale getirilmesini sağlar. Bu form hem nesne hem de akım özellikleri için kullanılmıştır.

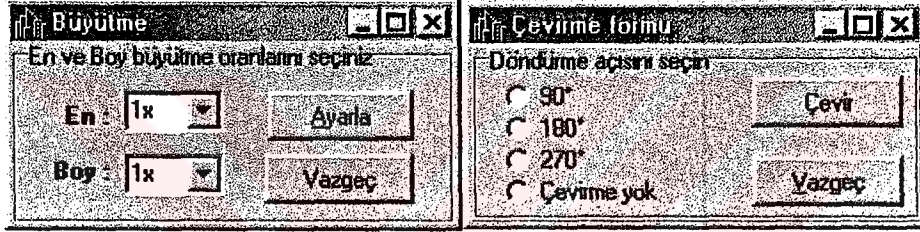


Şekil 3.5. Inspector Table Formu

3.4.5. Çevirme ve Büyütme Formu

Çevirme formunda 4 TRadioGroup cinsinde RadioButton vardır. Seçilen düğmeye göre formun rotation isimli değişkenine derece cinsinden çevirme değeri yazılır. Form kapandıktan sonra daha önce seçili olan bütün nesnelerin özellikleri bu işleminden etkilenir (Şekil 3.6).

Büyütme formu ise iki adet TComboBox tipindeki nesne ile en ve boy oranındaki seçilen değerler formun Zoom özelliğine aktarılır. Form kapatıldıktan sonra, daha önce seçilmiş durumda olan bütün nesnelerin özelliklerine bu değişiklikler uygulanır (Şekil 3.6).

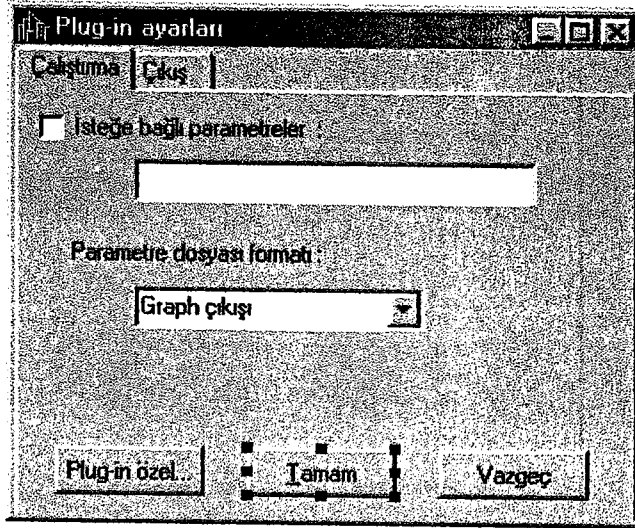


Şekil 3.6. Büyütme ve Çevirme Formu

3.4.6. Plugin Ayarı

Ayar formunda TTabbetNotebook kullanılmıştır. Bunda iki sayfa açılarak birisinde seçili olan plugin için parametreler ve diğer ayarlamalar yer almaktadır. Diğer sayfada ise seçili olan plugin çalıştırıldıktan sonra SİMKON'ya yollanan değerlerin ayarlanması yapılabilmektedir. Ayrıca eğer varsa plugin içinde bulunan kendi ayarlama menüsü çağrılabilir (Şekil 3.7).

Yine plugin için kullanılan PluginBekle isimli form, seçilen plugin çalıştırıldıktan sonra pluginin çalıştırılması bitene kadar bekler. Plugin çalışması bittikten sonra bir çıkış dosyası oluşturur. Bu form her saniye bu dosyanın oluşup oluşmadığına bakarak bekler. Eğer oluştuysa benzetim sonuçlarını gösteren form çağrılır.



Şekil 3.7. Plugin Ayar Formu

3.4.7. Diğer Formlar

Programda bir about formu, bir adet splash formu ve bir ipucu formu yer almaktadır. About formunda program hakkında bilgi verilmekte ve Image ve Timer özelliklerinden faydalanılarak zamanla değişen grafik çizimi yapılmaktadır. Ayrıca ekranın altıdan yukarıya doğru kayan yazı şeklinde program hakkında bilgi veren yazı geçmektedir.

Splash formunda ise programın yüklenmesi sırasında ekranda yaklaşık 2-3 saniye süreyle gözükmektedir. Bu form ekrandayken arka planda nesne yükleme, akım yükleme ve nesne resimlerinin vektörden oluşturulması yapılmaktadır.[17],[18]

3.5. Kontrol ve Benzetim

Tasarım programında dahili olarak benzetim ve kontrol özelliği yoktur. Ancak plug-in desteği sayesinde bu işlemleri yapılan uygulamaya özel eklenti programları ile yapabilmektedir. Eğer bir sistemin benzetimi veya kontrolü yapılmak isteniyorsa bu sistem ile ilgili olan hesaplama ve kontrol kısımlarının yapılması gerekmektedir. Bu kısım için başlangıçta bir yorumlayıcı düşünülmüştü. Ancak daha yorumlayıcı kısmı daha ilerideki çalışmalara bırakıldı.

SİMKON, plug-in modülü sayesinde her türlü donanıma erişebilir. Bunun için istenilen tasarım yapıldıktan sonra ilgili benzetim plug-in'i içinden portlara erişim mümkün olabilmektedir.

Paralel porttan veri alışverişi yapan örnek uygulamada ise Windows 9x altında portlara erişim uygulanmıştır. Böylece eğer benzetim sonucunda bazı değerler paralel porta gönderilmek isteniyorsa benzetim programı içinde bu kodun uygun yere eklenmesi yeterli olacaktır.

Tablo 3.9. Paralel Porta Erişen Kod.

<pre>typedef unsigned char byte; void GonderP(int Port,byte veri) { asm { mov dx,0x378; mov al,veri; out dx,al; } }</pre>	<pre>byte AIP(int Port) { byte veri; asm { mov dx,0x378; in al,dx; mov veri,ax; } return veri; }</pre>
---	--

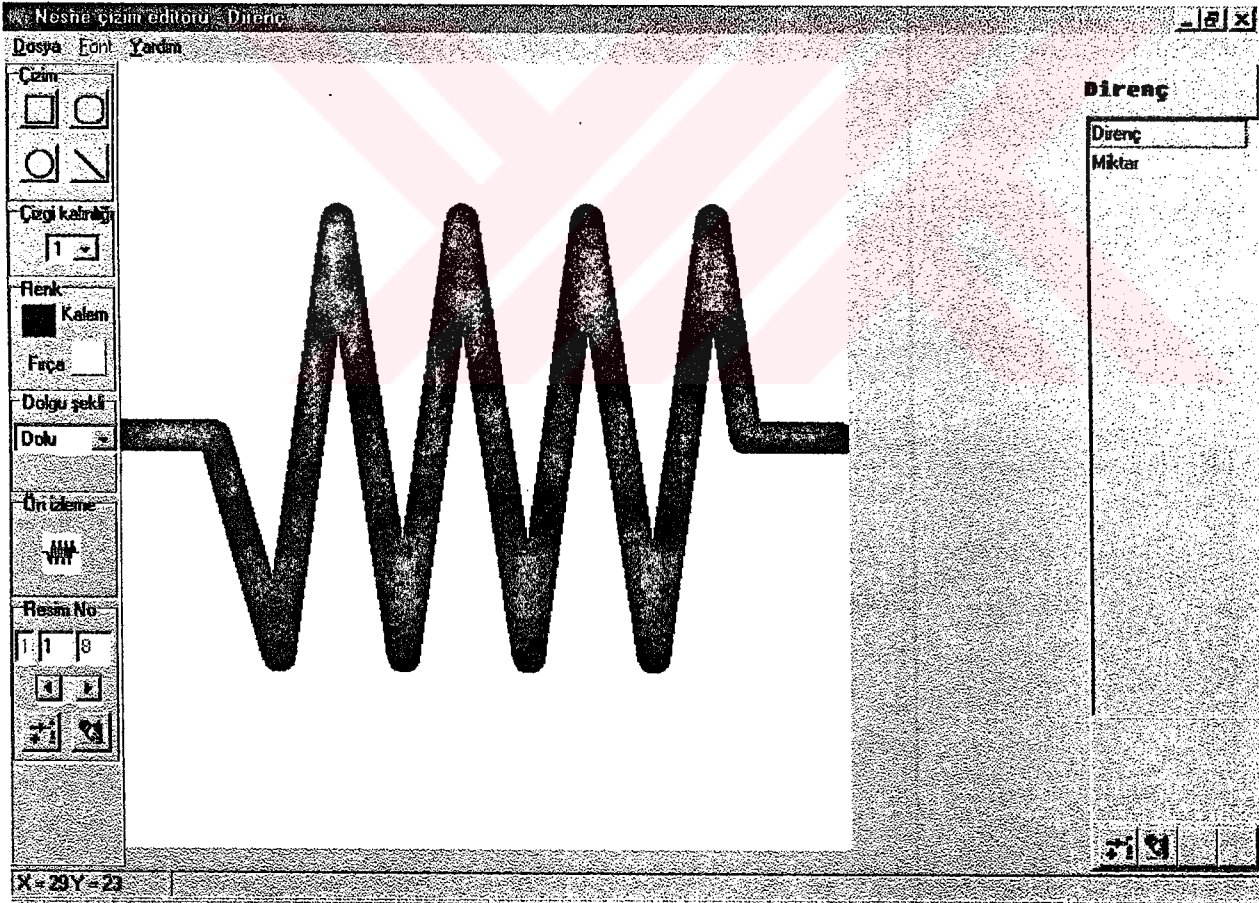
Eğer program Windows 9x dışındaki bir platformda çalıştırılacaksa assembly kullanmak yerine ilgili portu kullanan API'lerden faydalanmak en doğru çözüm olacaktır. Örneğin Windows NT altında seri portlara ulaşmak için aşağıdaki API'lerden faydalanmak en doğru çözüm olacaktır (Tablo 4.2).

Tablo 3.10. Windows NT İçin Port Erişim API'leri.

BuildCommDCB	GetCommMask	GetCommTimeouts
BuildCommDCBAndTimeouts	PurgeComm	GetDefaultCommConfig
ClearCommBreak	SetCommBreak	SetCommState
ClearCommError	SetCommConfig	SetCommTimeouts
CommConfigDialog	SetCommMask	SetDefaultCommConfig
DeviceIoControl	GetCommModemStatus	SetupComm
EscapeCommFunction	GetCommProperties	TransmitCommChar
GetCommConfig	GetCommState	WaitCommEvent

4. Örnek Tasarım Uygulaması

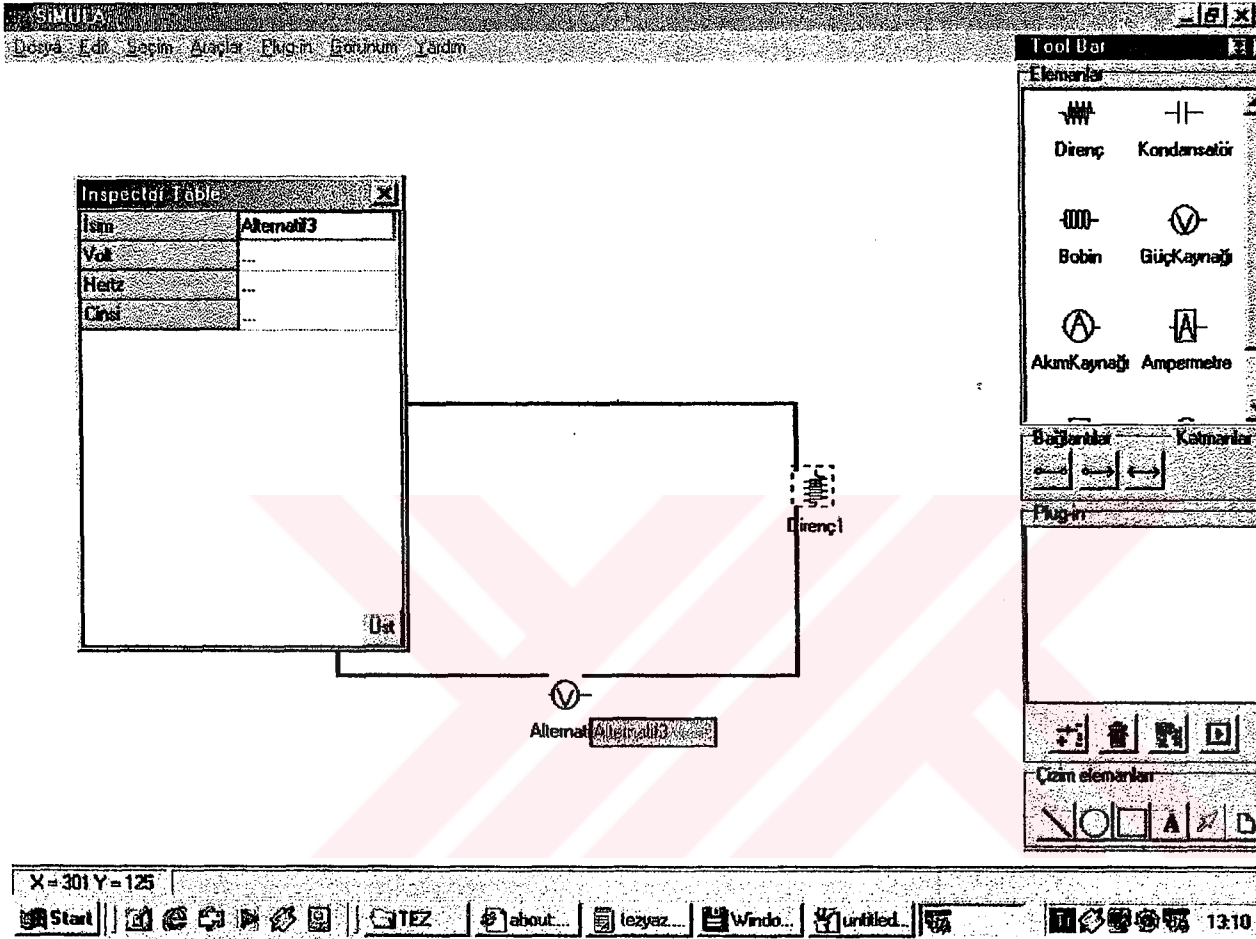
Bir uygulama yapmak istendiği zaman öncelikle o uygulama için gerekebilecek nesneler oluşturulmalıdır. Bunun için *Araçlar* menüsünden *Birim Editörü* çalıştırılmalıdır. Bundan sonra tasarımda kullanılacak elemanları tanımlayabiliriz. Örnek olması amacıyla bir lamba(direnç) ve bir alternatif akım kaynağından oluşan basit bir sistemi tasarlamak istediğimizi varsayalım. *Direnç* nesnesini oluşturmak için birim editöründe yeni bir sayfa açılması gerekmektedir. Açılan bu sayfada nesne ile ilgili özellikleri oluştururuz. Örnek uygulamada direnç için bizim önemli olan ve özelliğimiz sadece miktarıdır (Şekil 4.1). Eğer başka özellikleri de eklemek istersek ekranın sağ alt köşesinde yer alan ekle düğmesini seçerek istediğimiz kadar özelliği ekleyebiliriz.



Şekil 4.1 Birim Editörü

Benzer şekilde çözeceğimiz problemde kullanılacak diğer nesneler için benzer işlemler yapıldıktan sonra artık tasarım işine başlayabiliriz. Bunun için nesne

dosyasını bir isim vererek kaydediyoruz ve herhangi bir metin editörü ile SİMKON.ini dosyasının içine yaptığımız bu nesneleri tutan dosya isimlerini yazmak gerekmektedir.



Şekil 4.2. Örnek Tasarım Uygulaması

SİMKON'u yeniden başlattığımız zaman Şekil 4.2'de gözüktüğü gibi bir araç çubuğu karşımıza gelecek. Bu toolbar içinde oluşturduğumuz nesneleri birer ikon halinde görebiliriz. Artık simülasyonunu yapacağımız devreyi çizebiliriz. Bunun için nesneleri forma yerleştirip gerekli bağlantıyı yapmamız gerekli. Bunları yaptıktan sonra tasarım aşaması bitmiştir ve artık benzetime geçebiliriz.

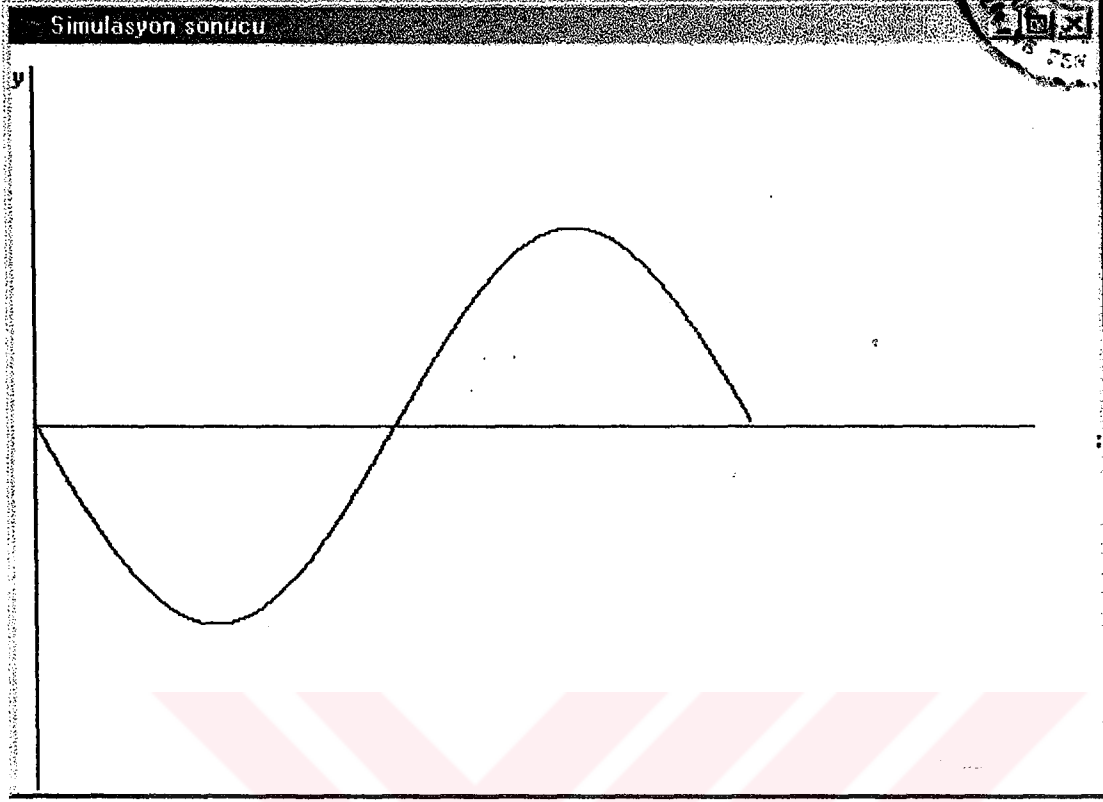
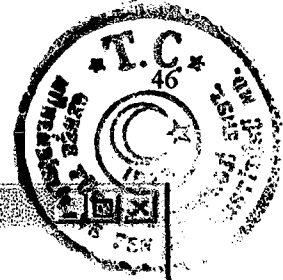
Çizdiğimiz şekillerin graf yapısı oluşturulup plug-in yönetimi aracılığı ile benzetim programına yollamamız gerekmektedir. Bunun için yeni plug-in ekle seçeneğini seçip çizdiğimiz devreyi analiz edecek programı belirtmek gerekir.



4.1. Benzetim Programı

Örnek olması açısından tek elemanlı elektrik devrelerini analiz eden benzetim programı yazılmıştır. Bu program basit bir devre analizi ile ampermetre veya voltmetre üzerindeki değerlerin zamana göre grafiğini çıktı olarak verir. Alternatif gerilim kaynağının düzgün sinüs dalgası olduğu varsayılmıştır. Bu devrenin tasarım aşamasındaki şeması Şekil 4.2’de verilmiştir.

Benzetim programı sadece devreyi alır, yorumlar ve çalışması bittiği zaman sonuçları bir text dosyası halinde kaydeder veya tcp/ip ile bunları grafik modülüne yollar. Bu işlerin sonucunda kontrol tekrar ana programa geçer. Ana programdan artık benzetim sonuçları grafik olarak izlenebilir (Şekil 4.3).



Şekil 4.3. Benzetim Programının Gönderdiği Sonuç.

Bu şekilde x eksenini zamanı, y eksenini de akım değerini göstermektedir. Beklendiği üzere ampermetre üzerinde okunan değerler sinüzoidal bir eğri oluşturmaktadır.

5. Sonuç



SİMKON ile her türlü mühendislik problemlerinin çözümü ve benzetimi amaçlanmıştır. Programın bu haliyle oldukça değişik konulardaki proseslerin tasarımını ve benzetimini yapabilecek durumdadır. Bir proses benzetimi için sadece hesaplama kısımlarının yazılması bu prosesin benzetimi için yeterli olmaktadır.

Ancak program her ne kadar tasarım ve benzetime izin veriyorsa da bir çok eksiği bulunmaktadır. Örneğin, tasarımlar sadece 2 boyutlu yapılabilmekte, akım çizimleri bazı tasarımlar için yetersiz olabilmektedir. Ayrıca ilerideki çalışmalarda benzetim programının harici olmasının getirdiği avantajların yanı sıra basit bir yorumlayıcı(interpreter) ile tasarım ortamı ile benzetim ortamının birleştirilebilmesi de düşünülebilir.

Olabilecek tüm SCADA sistemlerinin, sisteme özel program yazılmasına ihtiyaç bırakmadan tek bir paketle yazılması hedeflenmiştir. Ancak bu paket içinde yer alan kontrol kısmı bu iş için yeterince kullanışlı olamamıştır. Bunun nedeni SCADA sistemlerinde haberleşme için kullanılan birçok farklı haberleşme protokolü için sürücü yazılamamasıdır. Ancak yazılım modüler yapıda geliştirildiği için ilerideki çalışmalarda bu kısmın geliştirilebilmesi mümkündür. Ayrıca benzeri yazılım paketlerinin geliştirilme süreleri göz önüne alındığında yazılım oldukça kısa sayılabilecek bir sürede geliştirildiği düşünülürse programın çok daha fazla geliştirilebileceği ortadadır.

KAYNAKLAR

- 1 – Hexatec Software web sayfası, Saturn yazılımı, <http://www.hexatec.com/>, 2001
- 2 – Natianl Instruments web sayfası, Lab View yazılımı, 2001
<http://digital.ni.com/productpages/nilabview.nsf/main?readform>, 2001
- 3 – University of Twente, 20 Sim paketi, <http://www.20sim.com/>, 2001
- 4 – Wonderware Corporation web sayfası, Wonderware paketi,
<http://www.wonderware.com/home.htm>, 2001
- 5 – Sysmac productions web sayfası, SysMac paketi,
<http://www.boeing.com/assocproducts/easy5/>, 2001
- 6 – The Boeing Company web sayfası, Easyware paketi,
<http://www.boeing.com/assocproducts/easy5/>, 2001
- 7 – Mikrosay web sayfası, <http://www.mikrosay.com>, 2001
- 8 – SetKom web sayfası, <http://www.setkom.com.tr>, 2001
- 9 – E.Murat ESİN, Kimyasal proseslerin bilgisayar destekli benzetimi için bir grafik kullanıcı arayüzü, Doktora tezi, Ege Üniversitesi, Fen Bilimleri Enstitüsü, Bornova, İzmir 1995
- 10 – E.Murat ESİN PC Tabanlı proses kontrol için genel amaçlı bir türkçe scada ortamı, Proje, GYTE - Kocaeli, 2000
- 11 – The Scada Web, Ian WIESE, Web sayfası, <http://members.iinet.net.au/~ianw>, 2001,
- 12 – Boyer, Stuart, A. SCADA: Supervisory Control and Data Acquisition, Instrument Society of America, Research Triangle, NC. 1993.
- 13 – Ian WIESE, Scada Talks, Makale, Industrial Computing, page 34-36, December 1999
- 14 – Miri Engineering web sayfası, <http://www.miri.com.au>, 2001
- 15 – Captain Barry C. Ezell, Risks of Cyber Attack to Supervisory Control and Data Acquisition for Water Supply, Thesis, University of Virginia, May 1998
- 16 – Scadalist e-mail listesi ve arşivi, list@scadagospel.com,
<http://members.iinet.net.au/~ianw/archive>, 2001
- 17 – Microsoft Windows 32 SDK(Software Development Kit), Microsoft, 1999
- 18 – C++ Builder User's Manual, <http://www.inprise.com>, 1999

ÖZGEÇMİŞ

Adı Soyadı : Eşref DERİNKAYA
Doğum Tarihi : 25.06.1975
Medeni Hali : Bekar
e-mail adresi : derinkaya@yahoo.com

Eğitim durumu

Anadolu İlkokulu, Körfez - Kocaeli
İpraş (Tüpraş) 50.Yıl Lisesi, Körfez - Kocaeli
Kocaeli Üniversitesi – Endüstri Mühendisliği (2 Yıl), Kocaeli
Kocaeli Üniversitesi – Bilgisayar Bilimleri Mühendisliği, Kocaeli
GYTE – Bilgisayar Mühendisliği, Gebze- Kocaeli (Tez aşamasında)

Görevleri

- Kasım 1998 – Ağustos 2000 Yılına kadar Kocaeli Üniversitesi Bilgisayar Bilimleri Mühendisliği bölümünde Araştırma Görevlisi,
- Ağustos 2000 – Halen Tubitak UEKAE’de Araştırmacı,

Gerçekleştirdiği Çalışmalar

- Pascal ve C ile yazılmış DOS altında çalışan bir dosya yöneticisi, disket formatlama, disket kopyalama, FAT 12 ve FAT 16 için dosya listeleme/değiştirme, TSR ekran koruyucusu, TSR yardım, sistem bilgi ve benchmark programlarından oluşan bir utility, Ocak 1997,
- Araba lastiği satış bayileri için Delphi ile yazılmış Lastik Otomasyonu programı, Mart 1997,
- Assembler ile yazılmış TSR virus aşısı, COM dosyaların sonuna bir tür checksum ekleyerek dosya değiştirildiği zaman uyaran program, Eylül 1997
- Motorola MC68008 mikroişlemcisi kullanan bir mikroişlemci kartı ve çevre birimi hazırlanması, bunun seri porttan PC ile haberleştirilmesi. Ocak 1998,
- İnternet üzerinde sesli ve yazılı haberleşme programı. IRC benzeri yazışma, gerçek zamanlı bir sesli konuşma modülü ve çok kullanıcılı sanal bir beyaz tahta. Haziran 1998,
- Kandilli Rasathanesi kayıtlarında yer alan sismik kayıtların Marmara Bölgesi ve Türkiye haritası üzerinde gösterilmesi, Kasım 1999

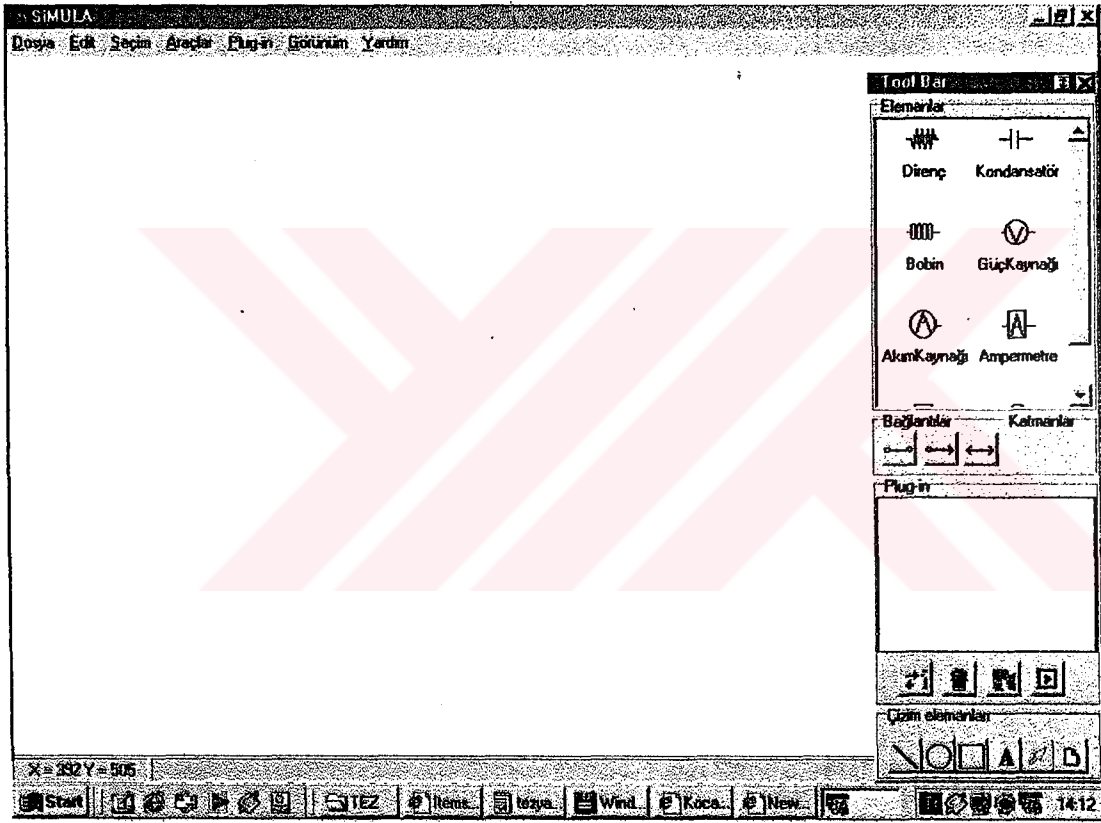
EKLER

Ek-1

Kullanma Kılavuzu

Çalıştırma

Program kurulduktan sonra SİMKON ikonuna çift tıklanarak program çalıştırılır. Program çalıştırıldığı zaman ekrana aşağıdaki ekran gelecektir (Şekil Ek-1.1).



Şekil Ek-1.1 SİMKON Ana Formu

Sağ taraftaki ToolBar içinde çizimde kullanılmak üzere hazırlanan nesneler vardır. ToolBar oldukça esnek tasarlanmıştır. Bu formu ekranda istediğiniz yere formun boş olan istediğiniz yerinden sürükleyerek götürebilirsiniz. Boyutunu ayarlamak için sağ üstteki küçültme ve büyütme düğmesini kullanabilirsiniz. Ayrıca buradaki her bir çerçevenin üzerine çift tıklama yapılarak seçili olan çerçevenin büyüklüğünün değiştirilmesi yapılabilmektedir.

Menülerin kullanımı

Dosya menüsü

Bu menü altında genel dosya işlemleri bulunmaktadır.

Yeni: Yeni bir tasarım yapılmak istendiğinde kullanılır. Var olan tasarım kapatılarak yeni bir sayfa açılır.

Kayıt: Yapılan tasarımı diske kayıt eder.

Yükle: Daha önceden yapılmış bir tasarımı yükler.

Edit menüsü

Bu menü altında çizim işlemleri bulunmaktadır.

Kes : Seçili olan nesneleri clipboard'a atarak siler.

Kopyala : Seçili olan nesneleri silmeden kopyalar

Yapıştır : Clipboard'da ki nesneleri çizim ekranına yapıştırır.

Çevir: Seçili olan nesneyi belirtilen açı kadar çevirir.

Zoom: Seçili olan nesneyi belirtilen oranlar kadar büyütür.

Seçim Menüsü

Hepsini seç : Tasarım formu üzerinde bulunan bütün nesneleri birden seçer.

Seçimi kaldır : Seçili nesnelerin seçimini kaldırır.

Diğerlerini seç : Seçili olmayan nesneleri seçer, seçili nesnelerin seçimini kaldırır.

Araçlar Menüsü

Bu menü SİMKON paketi içindeki araçlara kısa yoldan ulaşımı sağlar.

Plug-in Menüsü

Ekle : Yeni bir plug-in eklemek için kullanılır.

Sil : Var olan plug-in listeden çıkartılır.

Düzenle : Seçili plug-in için ayarlamalar yapılır.

Çalıştır : Seçili olan plug-in çalıştırılır.

Görünüm

Araç Çubuğu : Bu menü ile ToolBar gösterilip saklanır.

Büyük Mouse : Ekran boyunca mouse imleci gösterilir ve saklanır.

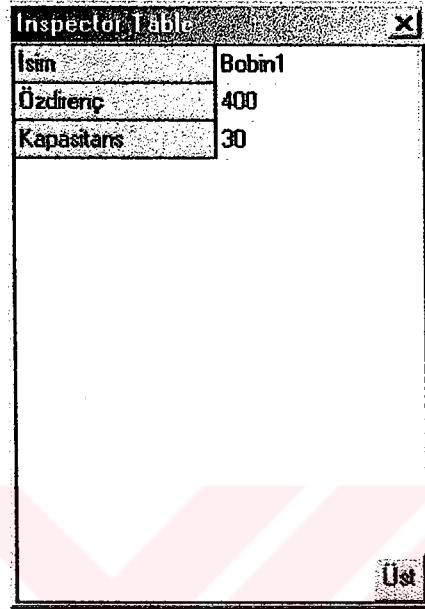
Yardım

Program hakkında genel yardım içerir.

İpuçları

- Programda çizime başlamak için önce toolbar'dan mouse ile bir nesne seçilir. Daha sonra çizim ekranı üzerinde yerleştirilmek istenen yere tıklanarak nesne yerleştirilir.
- Nesneler arasında bağlantı yapmak için çizim formuna en az iki nesne yerleştirdikten sonra ToolBar üzerindeki Bağlantılar kısmında bulunan bağlantı sembollerinden birisi seçilerek, yönlü, yönsüz ve çift yönlü bağlantı yapılabilir.
- İki nesneyi birbirine bağlamak için yapılacak bağlantı türü(yönlü, yönsüz, çift yönlü) seçilir. Önce kaynak nesneye tıklanır, sonra hedef nesne tıklanarak bağlantı kurulmuş olur.
- Bir nesnenin yerinin değiştirilmesi için nesne fare ile tutularak sürüklenir ve yeni yerine bırakılır.
- Nesneyi seçmek için nesneye tıklanır ve nesne seçilmiş olur. Eğer daha önceden seçili bir nesne ise seçim kaldırılır.

- Nesne özellikleri için nesneye çift tıklanır ve nesne özellikleri görüntülenir. Eğer özellikler değiştirilmek isteniyorsa, değişiklik işlemleri de buradan yapılır.



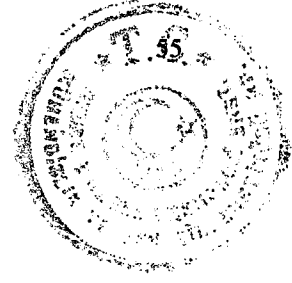
The image shows a window titled 'Inspector Table' with a close button (X) in the top right corner. Inside the window, there is a table with three rows and two columns. The first row has 'İsim' (Name) in the first column and 'Bobin1' in the second column. The second row has 'Özdirenç' (Resistance) in the first column and '400' in the second column. The third row has 'Kapasitans' (Capacitance) in the first column and '30' in the second column. Below the table, there is a button labeled 'Üst' (Up) in the bottom right corner.

İsim	
Bobin1	
Özdirenç	400
Kapasitans	30

Üst

Şekil Ek-1.2 Özellik Tablosu

- Akım üzerine gelindiği zaman akım mavi rengini alır. Bu durumda tek tıkladığımız zaman akım seçme işini yaparız. Eğer çift tıklarsak akım için özellikler tablosu görüntülenir (Şekil Ek-1.3).
- Plug-in eklemek için ToolBar'dan Plug-in çerçevesi veya menü kullanılabilir.
- Serbest vektör çizim yapmak için ToolBar'da aşağıdaki nesneler kullanılabilir. Silmek için en sağdaki düğme kullanılır.



Inspector Table	
İsim	Akım1
Kaynak	Ampemetre2
Hedef	Bobin1
Direnç	Boş
Üst	

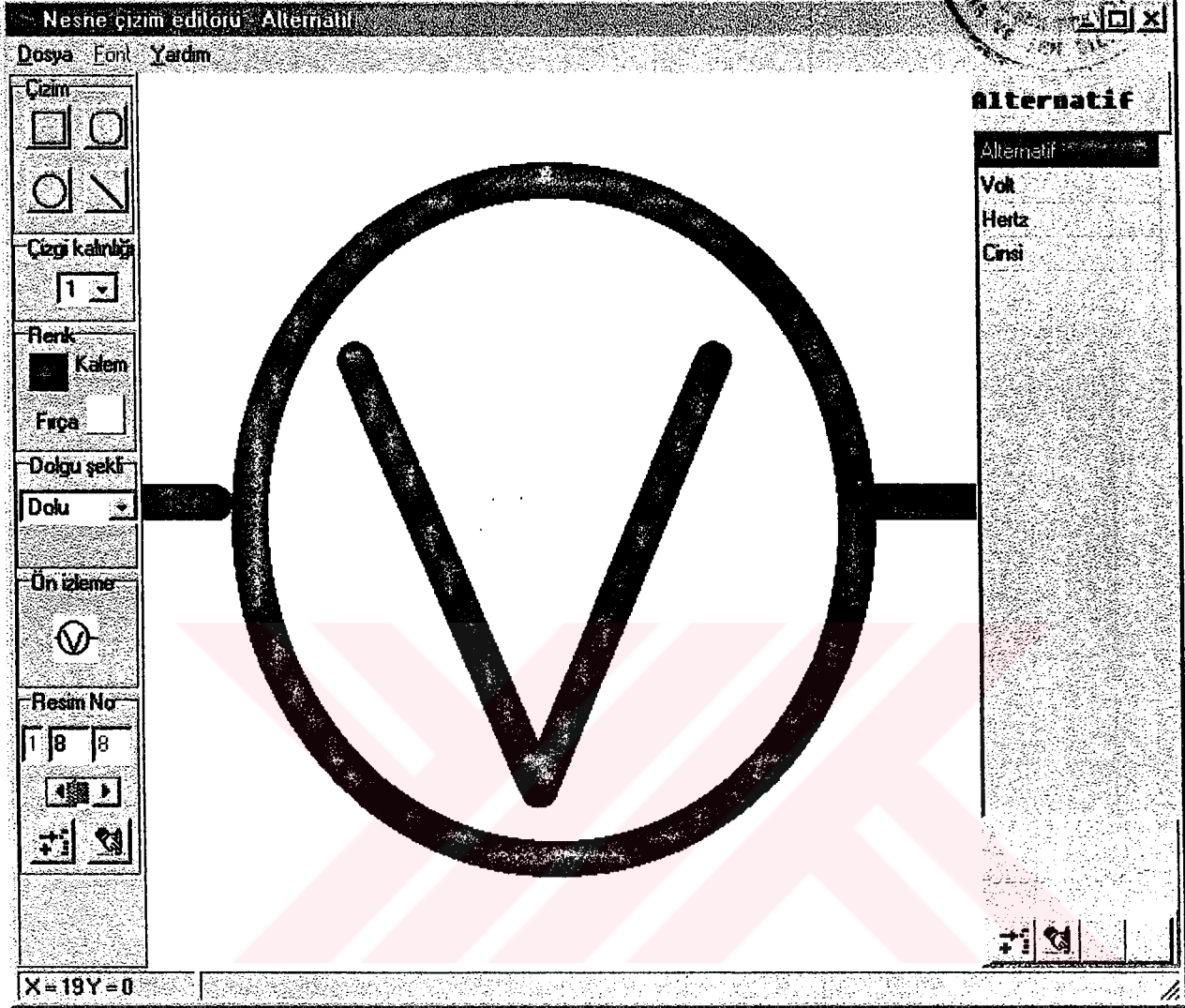
Şekil Ek-1.3. Akım İçin Özellikler Tablosu

Birim Editörü

Tasarımda kullanılan nesnelerin oluşturulması ve değiştirilmesi için kullanılır.

Dosya menüsünde dosya işlemleri bulunur. Bunlar Yeni dosya, kayıt ve çıkıştır.

- Çizim yapmak için çizim çerçevesindeki düğmelerden biri seçildikten sonra ızgara üzerine mouse ile çizilir.
- Çizgi kalınlığı kısmından çizimde kullanılan kalemin kalınlığı belirtilir.
- Renk kısmında dolgu rengi ve çizgi rengi belirtilir.



Şekil Ek-1.4. Birim Editörü

- Resim No yazan kısımdan, ilk ve son nesne numarasına ulaşılabilir ve o diğer nesnlere geçiş yapılabilir. Buradaki sağ ve sol oklar aracılığı ile bir önceki ve bir sonraki resim arasında geçiş yapılabilir.
- Yine Resim No yazan çerçeve içindeki soldaki düğme ile yeni nesne sayfa eklenebilir ve sağdaki düğme ile var olan sayfa çıkarılabilir.
- Çizim ızgarasının sağ tarafındaki listede En üstte nesne ismi olmak üzere nesnenin özelliklerinin isimleri yer alır. Direnç değeri, volt gibi değerler burada belirtilir.



- Alttaki düğmelerin soldaki ile yeni bir özellik eklenir ve sağdaki ile de var olan bir özellik çıkartılabilir. Özellik ismi değiştirmek için ismin üzerine tıklamak yeterlidir.
- Birim editörü ile kayıt edilen nesnelerin SİMKON tarafından kullanılabilmesi için SİMKON.INI içerisinde bu nesne dosyasının gösterilmesi gerekmektedir. Örneğin birim editörü ile yaptığımız çizimleri elonesne.dat isminde kaydederek, notepad veya herhangi bir metin editörü ile SİMKON.ini dosyası açılıp birinci satırı /kayit_yeri/elonesne.dat şeklinde ayarlanmalıdır. Bu işlem yapıldıktan sonra SİMKON yeniden başlatıldığında nesneler otomatik olarak aktif hale geçecektir.

