T.C. İSTANBUL KÜLTÜR ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

EAGLEYE SCADA DESIGNER

YÜKSEK LİSANS TEZİ

Emrah KÜÇÜKALİ

0309040008

Tezin Enstitüye Verildiği Tarih: 19 Haziran 2006
Tezin Savunulduğu Tarih: 10 Ağustos 2006

Tez Danışmanı: Prof .Dr. Behiç ÇAĞAL
Diğer Jüri Üyeleri: Yrd. Doç. Dr. Hikmet ÇAĞLAR
Yrd. Doç. Dr. Elif TARIM (Y.T.Ü)

AĞUSTOS 2006

ÖNSÖZ

Yüksek Lisans Tezimin hazırlanması esnasında bana rehberlik eden Prof. Dr. Behiç ÇAĞAL 'a, teknik konularda bilgi ve araç gereç temininde yardımcı olan 3E Yazılım çalışanlarına teşekkürü bir borç bilirim.

Haziran 2006 Emrah KÜÇÜKALİ

İÇİNDEKİLER

KISALTMALAR

SCADA : Supervisory Control And Data Acquisition–Uzaktan Kontrol ve Gözetleme Sistemi

PLC : Programmable Logic Controller

OPC : OLE for Process Control – Yönetim kontrol için OLE

OLE : Object Linked Embedded – Gömülü nesne bağlama

TAG : Değişkenlere verilen genel ad. (Örn: byte,word,Dword,Double v.s.)

DDE : Dynamic Data Exchange – Dinamik veri paylaşımı

CRC : Cyclic Redundancy Check

ŞEKİL LİSTESİ

ÖZET

Gelişen teknolojide hız ve güvenlik etkenlerinin en yüksek değerlerde istenmesi ve hatanın en az da tutulması için otomasyon sistemlerinde bilgisayarlar vazgeçilmez araçlar olmuştur. Kontrol sistemlerinin bilgisayarlara devredilmesi ve kullanılan yazılımlar ile verilerin analiz edilmesi suretiyle geleceğe yönelik kararların hızlı ve doğru alınması işletmelerin kar oranlarını yükseltmek için en önemli araçlar olmuştur.

Mekanik teknolojisinin bilgisayar kontrollü eğilime girmesi mühendislerin sistem tasarımlarında yazılım ihtiyacını vazgeçilmez kılmıştır.Kontrolü bilgisayara devrederek insan hatalarının ortadan kaldırılması ve operatörlerin iş yükünün azaltılması toplam kaliteyi artıran bir unsur olmuştur.

Bu projenin amacı kurulan mekanik sistemi kontrol edecek yazılımın programcılık bilgisi gerektirmeden ve cihazların haberleşme protokollerini bilmeden kolayca oluşturulması ve görsel tasarımın esnek ve her türlü otomasyon işi için uygun bir ortam yaratılmasıdır.

Bu sistemde sistemi kullanacak olan kişinin bir programcı değil sadece sistemi kuran tasarımcı olduğu gözetilmiş ve hiç kod yazmadan basit bir bilgisayar bilgisiyle kontrol yazılımı yazabilmesi hedeflenmiştir.

Günümüze kadar SCADA diye bilinen saha cihazları ile haberleşen yazılımların Visual Basic,Delphi,C++ gibi yüksek seviyeli dillerin uzman programcılar tarafından kodlandığı bir zamanda Eagleye SCADA Designer sistem tasarımcılarına programlama uzmanlığı yükünü ortadan kaldırmayı hedeflemiştir.

SUMMARY

In progressive technology, because of "speed and security factors desires" and "needs of minimum errors", computers became indispensable tools for automation systems. With assigning control systems to computers and analyzing data by using softwares which makes to decide fast and correct have become important to grow up benefit ratios.

With combine of computer controlled mechanic technology, engineers' needs of softwares in system planning is now indispensable. By assigning the control to computers consuming human faults and by taking operators' workload have become an element which grow up the total quality.

This project's aim is to create an environment without need of programming knowledge and communication protocols with a software which will control the set up mechanic system and make flexible visual design that is suitable for all automation jobs.

In this system people who use this system have been considered as a planner of the system but not a programmer, and aimed to control the software without coding, only with a simple knowledge of computer.

Until the present-day, softwares which were communicating with tools known as SCADA have been coded with professional coders by professional programing languages like Visual Basic, Delphi, C++ ; Eagleye SCADA Designer aimed to eliminate workloads of system planners.

1.GİRİŞ

1.1 Kontrol Sistemleri Nedir?

Teknoloji ilerledikçe insanların çalıştıkları alanlar daralmaktadır. Elektroniğin bir alt kolu olarak Endüstriyel kontrol'de bu alanlardan biridir. Otomasyon veya diğer adı ile endüstriyel kontrol alanları kontrol sistemleri teknolojisi ve süreç kontrol teknisyen ve teknikerlerinin uğraş alanına girmektedir.

1.2 SCADA Nedir?

SCADA terimi "Supervisory Control And Data Acquisition kelimelerinin baş harflerinden oluşturulmuştur. Türkçeye " Danışmalı Kontrol ve Veri Toplama Sistemi " veya "Uzaktan Kontrol ve Gözetleme Sistemi" olarak çevrilebilir. Kısaca bilgisayarlardan, haberleşme aletlerinden, algılayıcılardan veya diğer aygıtlardan oluşturulmuş denetlenebilen ve kontrol edilen bir sistemin genel adıdır. En genel olarak enerji scada'sı (Elektrik-Su-Doğalgaz) ve süreç scada'sı (fabrika-tesis otomasyonu) olarak ikiye ayrılır.

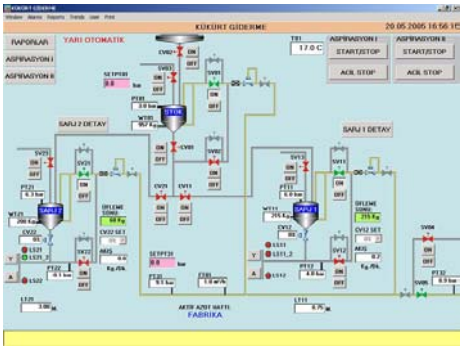
Bilgisayar ve iletişim teknolojilerindeki son gelişmeler ve bunlarla ilgili cihazlardaki maliyet düşüşleri, elektrik dağıtım sistemlerinin otomasyonunu teknik ve ekonomik olarak yapılabilir hale gelmiştir. Örneğin dağıtım ve su otomasyonu ile şebekenin uzaktan izlenmesi ve hızlı ve etkin bir şekilde kontrolünü sağlanır ve sonuçta daha güvenilir, sürekli ve kaliteli elektrik enerjisi ve su beslemesi olası duruma gelir. Sistem bilgileri arşivlenebilir ve istatistikî raporlar ile en etkin ve ekonomik işletme yönetimi sağlanır.

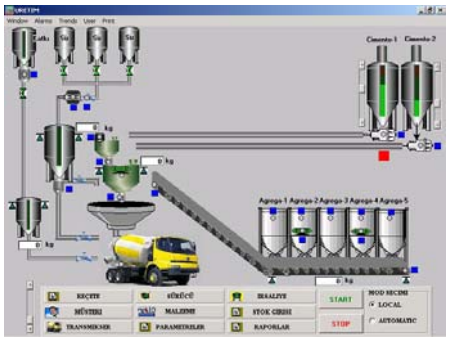
Süreçler için gözetleyici denetim ve veri toplama işlemlerini yapan sistemler için kullanılan SCADA sistemleri, fabrikadaki süreçlerin (hammadde, üretim ve mamul madde takibi vb.) denetiminde kullanılan çeşitli araçlarla (RTU, PLC vb.) birlikte fabrikanın üretim kontrolü ve takibine yönelik bir alt yapı oluştururlar. Bu alt yapının imkan verdiği ölçüde üretim kaynakları planlaması (MRPII) ve işletme kaynakları planlama (ERP) sistemleriyle gerekli bağlaşımlar kurularak ideal bir yapıya erişilebilir. Amaç en düşük maliyetle, daha kaliteli ve daha çok üretmek için gerekli yapıyı kurmaktır. İşletmedeki tesislerden en yüksek verimlilikle yararlanmak, yöneticilerin işletmeye ve üretim bilgilerine tam olarak hâkim olmasıyla sağlanabilir. SCADA yazılım paketleri endüstriyel tesislerde alt yapı yazılım görevini üstlenmeli ve fabrika içi ile dışındaki ağlara bağlanarak şirketin bütün katmanlarının uyum içerisinde çalışmasına imkân verir. SCADA işletme genelinde herkese, her zaman erişebilecekleri, gerçek zamanlı ve ayrıntılı bilgiyi sağlar.

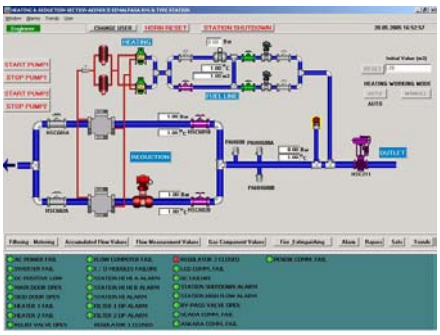
SCADA sistemi, hidroelektrik, nükleer güç üretimi, doğalgaz üretim ve işleme tesislerinde, gaz, yağ, kimyasal madde ve su boru hatlarında pompaların, valflerın ve akış ölçüm ekipmanlarının işletilmesinde, kilometrelerce uzunluktaki elektrik aktarım hatlarındaki açma kapama düğmelerinin kontrolü ve hatlardaki ani yük değişimlerinin dengelenmesi gibi çok farklı alanlarda kullanılabilmektedir.

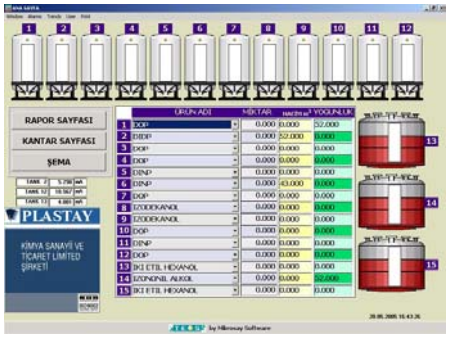
Günümüzde tüm tesis ve işletmeler SCADA sistemini kurmaya başlamışlardır; giderek de daha fazla ivme kazanacaktır.

SCADA programlarından örnek ekranlar aşağıdadır:









1.3 PLC Nedir?

Bir SCADA uygulamasının temelinde kontrol ve kumanda fonksiyonlarını yerine getiren PLC'li bir otomasyon panosu bulunur. PLC "Programmable Logic Controller" kelimelerinin kısaltılmış halidir. Günümüzde otomasyonun her alanında küçük ve kompakt olanından büyük ve modüler olanlarına kadar çeşitli tip ve markalarda PLC kullanılmakta ve geleneksel elektromekanik kumanda sistemleri çok daha az tercih edilmektedir. İlk PLC MODICON markadır.

PLC ile yapılan bir otomasyon sistemi daha güvenilir, daha fonksiyonel, kolay genişleyebilir ve değiştirilebilir bir yapıda olmaktadır. Çok karmaşık prosesler ve kontroller PLC ile rahatlıkla yapılabilmekte, remote I/O ve operatör panel gibi ekipmanlar sayesinde kullanım kolaylığı sağlanmaktadır.

PLC akış kontrolü, interlock, PID gibi bütün kumanda ve kontrol fonksiyonlarını üstlenmeli ve bir üst sistem olan SCADA'dan sadece ayar değerlerini alıp istenilen

bilgileri verebilmelidir. Kontrol ve kumanda için hayati önemi olan bu fonksiyonları hiç bir üst sisteme devretmemelidir.

Bu prensiplerin bilincinde olarak PLC ve SCADA sistemlerini tasarlamak ve devreye almak gerekir.

1.4 Protokol Nedir?

Protokoller iletişimin kurallarıdır. Bir network'taki iletişim kuralları protokoller tarafından düzenlenir. Diğer bir deyişle bilgisayarlar aynı ya da uyumlu protokolleri kullanıyorlarsa birbirleriyle iletişim kurabilirler.

Çok sayıda protokol vardır. Ancak her birinin değişik amaçları vardır. Genel olarak firmalar ürettikleri cihazlar için kendi protokollerini oluştururlar. Ama bazı protokoller üreticiler tarafından kolaylık ve globalliği sağlamak adına kabullenilebilir. Örneğin Fransız Modicon firmasının 80'li yıllarda oluşturduğu MODBUS protokolü bu gün birçok endüstriyel cihaz üreten firmalar tarafından kullanılmaktadır. Böylece birbirinden farklı markalarda ama aynı görevi yapan cihazlar tek hat üzerinde birbirleri ile veya bilgisayar ile haberleşebilmektedir.

Rekabet piyasasında bazı firmalar ise ürettikleri cihazlar için kendi protokollerini oluşturup kendi yazılımlarını satmak ve tekel konumuna gelmek istemişlerdir. Örneğin Siemens firmasının ürettiği PLC'lerin protokolleri açık değildir. Bu PLC'ler ile haberleşen SCADA programı yapabilmek için Siemens firmasının yazılımları gerekmektedir. Bu tekeli kaldırmak için cihaz üreticileri bir araya gelip OPC adında bir şirketler birliği oluşturmuşlardır. Bu şirketler birliğinin amacı protokolünü açmayan firmaların programcılara bir haberleşme kütüphanesi oluşturmasını şart koşmuştur. Böylece programcılar kendi yazılım ortamlarında bu kütüphaneyi kullanarak PLC'ler ile haberleşen programlar yazma imkânına kavuşmuştur.

1.5 Örnek bir protokol açılımı (MODBUS RTU)

Modbus RTU protokolü Master/Slave (Efendi/Köle) mantığında tasarlanmış bir protokoldür. Bu mantığa göre hat üzerinde bir tane cihaz Master diğerleri Slave olarak atanır. Master cihaz (ki bu genelde bilgisayardır) almak istediği bilgiler için Slave cihaza ilgili soruyu sorar ve Slave cihazdan cevap bekler. Modbus RTU protokolünde basitçe haberleşme şöyledir:

Bilgisayar tarafından yollanan cümle:

Şekil 1.1 – Modbus Protokolü 1

Slave cihazdan gelen cevap cümlesi:

Şekil 1.2 – Modbus Protokolü 2

Bu protokol deki standart şöyledir:

ID : (1 Byte) Bir haberleşme hattında birden fazla cihaz seri olarak birbirine bağlanmış olabilir. Bu cihazlara özel olarak bir numara verilir. Böylece Master cihazdan gönderilen soru cümlesinin hangi cihaz tarafından işleneceği belirlenmiş olur.

Function code : (1 Byte) Modbus RTU protokolünde soru cümlesinin tipini belirler. Örneğin hex 03 okuma isteğini belirler.

Start Address : (1 Word) Bilgiler cihazlar içinde belirli adreslerde tutulur. Bu adresleri indeks numarası olarak düşünebiliriz. Burada belirlenen adres hangi bilginin alınacağını belirler.

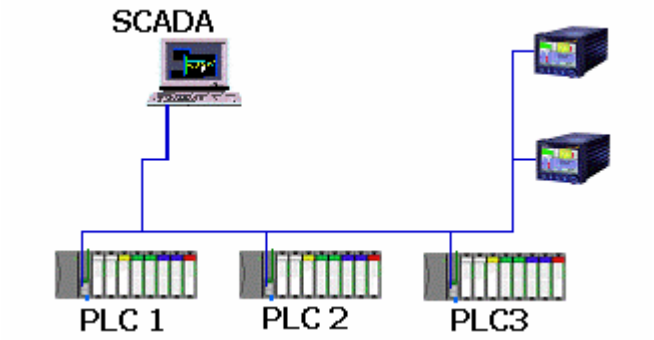
Count : (1 Word) Bir soru cümlesinde birden fazla değeri almak istendiğinde peş peşe gelen adresleri almak için 1 den büyük değer verilerek istenilen sayıda bilgi alınabilir.

Byte Count : (1 Word) Gelen cevaptaki byte sayısını verir.

Status : (N byte) İstenen verileri içeren byte dizisi.

CRC : (1 Word) Protokolün gürültüden etkilenip etkilenmediğini anlamak için eklenen kontrol sayısı. Cümledeki tüm byteların XOR işlemine sokularak elde edilir.

Bir haberleşme sisteminin şeması:



Şekil 1.3 Scada Sistemi

1.6 Bağlantı Tipleri

Cihazların birbirleri ile bağlantısında çeşitli fiziksel hatlar vardır.Bu bağlantılar dan en genel olanları RS232,RS485 ve RS422 dır.Bazı firmaların özel ürettikleri fiziksel hatlarda vardır.Bunlara en tanınmış örnek olarak PROFIBUS ‘ı verebiliriz.

RS 232: Seri iletişim portuna verilen isimdir. Bilgisayarlarda rastladığımız 9 pinli erkek girişler RS232 dir.

RS 485: 2 telli seri iletişim portudur.1200 m mesafeye veri taşımak mümkündür. Üretilen haberleşmeli cihazların büyük bir çoğunluğu bu fiziksel hatla haberleşir. Bilgisayarları bu hatta çıkartmak için RS232/RS485 çevirici cihazlar kullanılır.

RS 422: 4 telli seri iletişim portudur.

Ayrıca yeni üretilen cihazlarda fiziksel hat olarak ETHERNET yükselen yıldızdır. Ethernet’in network sistemlerinde kullanılması ve her türlü cihazın birbirine bağlanması (Örneğin bir PLC’nin networktaki yazıcıya bağlanması ve otomatik çıktı vermesi) trendi cihaz üreticilerini yaygın olan Ethernet’e yöneltmektedir.

1.7 SCADA Programlarını Oluşturan Temel Yapılar

1.7.1 Sayfa Tasarım

SCADA programlarının en önemli parçasıdır. SCADA’nın ruhu gereği sahayı ekrana taşımak ve kullanıcıya görsel bir arabirim yaratmak gerekir. Sahanın genel görünümü, operatörün parametre girişi yapabilmesini sağlayan giriş bölümleri sayfa tasarımları ile gerçekleştirilir.

1.7.2 Trend

Bilgisayar sistemlerinden beklenen bir önemli özellikte geçmişe yönelik verilerin grafiksel gösterimidir. Trend kavramı sahadan gelen bilgilerin -ki bu bilgiler sıcaklık, akım, voltaj, nem ve hız gibi zamana göre incelenen bilgilerdir – zamana bağlı grafiksel görünümüne verilen addır. Trendleri gösteren programlardaki grafiklere Chart adı verilir. Her bir eğri ise kalem olarak adlandırılır. Trend kavramının altında kayıtta vardır. Yani grafiği istenen verilerin geçmişteki durumu da kayıt altına alınıp daha sonra tekrar gösterilebilmelidir.

1.7.3 Alarm

Alarm sistemi SCADA programlarının olmazsa olmazıdır. Kritik durumlarda kullanıcıları uyarması ve anında müdahalenin sağlanabilmesi için bir uyarı sistemidir. PLC’den alınan bilgilerin belli sınırları geçmesi durumunda kullanıcıya uyarı verilmesi gerekir. Örneğin sıcaklık değerinin 1000 dereceyi geçmesi durumunda sisteme zarar gelecek bir işletmede sıcaklığın 950 dereceyi

geçmesi şartına bağlı bir alarm oluşturulabilir. Kullanıcının bilgisayar ekranında dikkati çekilerek sisteme gerekli müdahale yapması sağlanır.

1.7.4 Script

SCADA programlarında PLC den gelen değerlerin özel olarak işlenmesi ve kullanıcıya gösterimde yorumlanması gerekir. Bu gibi durumlarda kod yazmak kaçınılmazdır. SCADA programında yazılan ve özel bir yazıma sahip kodlara script denir.

1.7.5 Internal Tags (İç Değişkenler)

İç değişkenler PLC'den gelen bilgilerin yorumlandıktan sonra aktarıldığı değişkenlerdir. Bu değişkenlerin içerdiği bilgiler sadece SCADA'nın atadığı değerleri taşır. Bilgisayarın belleğinde tutulan alanlar olarak düşünülebilir.

1.7.6 External Tags (Dış Değişkenler)

İç değişkenlerin aksine PLC'den gelen değerlerin saklandığı değişkenlerdir. Program içinde gösterimlerde kullanılır.

2. PROJE (Eagleye SCADA Designer)

2.1 Projenin Amacı

Bir SCADA programının temelinde bulunan sayfa tasarımı, trend, alarm, script ve cihazla haberleşmek için gerekli protokolün kodlanması ancak uzman bir programcı tarafından hayata geçirilebilir. Bu süreç oldukça zor ve uzun olabilir.

Bu projede bir SCADA programı yapmak için gerekli olan araçlar kullanıcıya hazır sunulup protokol kodlama zahmetini ortadan kaldırmaktır. Bu program sayesinde sistemi kontrol eden ve izleyen bir SCADA programı kolayca hazırlanabilir.

Ayrıca proje Client/Server mimariyi destekler. Bu sayede birden fazla bilgisayar bir networkta birleştirilerek birden fazla izleme noktası oluşturulabilir.

2.2 Eagleye Scada Designer Nedir?

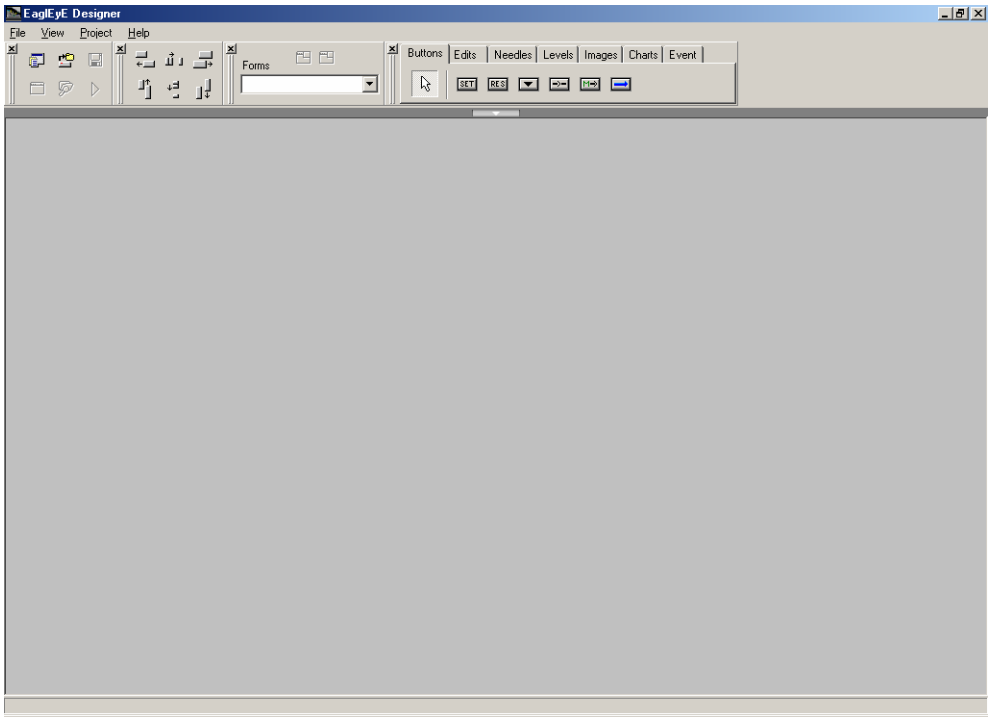
Eagleye Scada Designer Windows işletim sisteminde çalışmak için hazırlanmış kolay ve hızlı programlama olanağı veren bir yazılımdır. SCADA yazılımlarının ihtiyacı olan tüm yapıları içinde barındırır.

Eagleye Scada iki temel ayaktan oluşur. Bunlardan biri saha tasarımlarının yapıldığı, alarmların tanımlandığı, trendlerin oluşturulduğu ve scriptlerin yazıldığı kullanıcı arabiriminin hazırlandığı Tasarım programıdır.

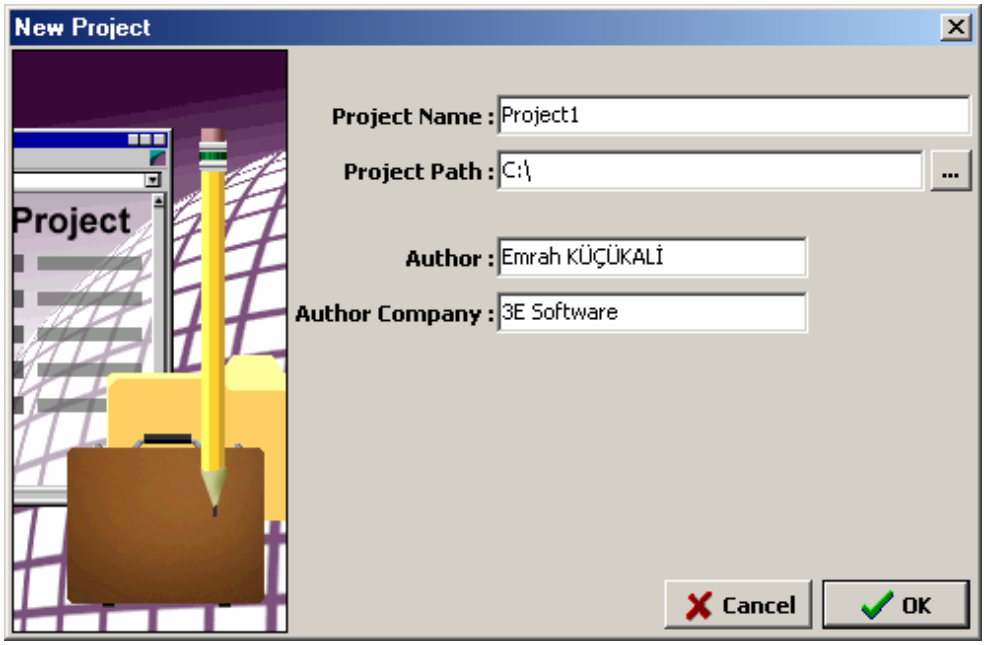
Diğer program ise PLC'ler ile haberleşip verileri tasarım programına gönderen "Eagleye External Tag Server" dır.Bu program sayesinde PLC'den alınan veriler birden fazla bilgisayara dağıtılabilir.

3. KULLANICI ARAYÜZLERİ VE KULLANIMLARI

3.1 Program Genel Yapısı



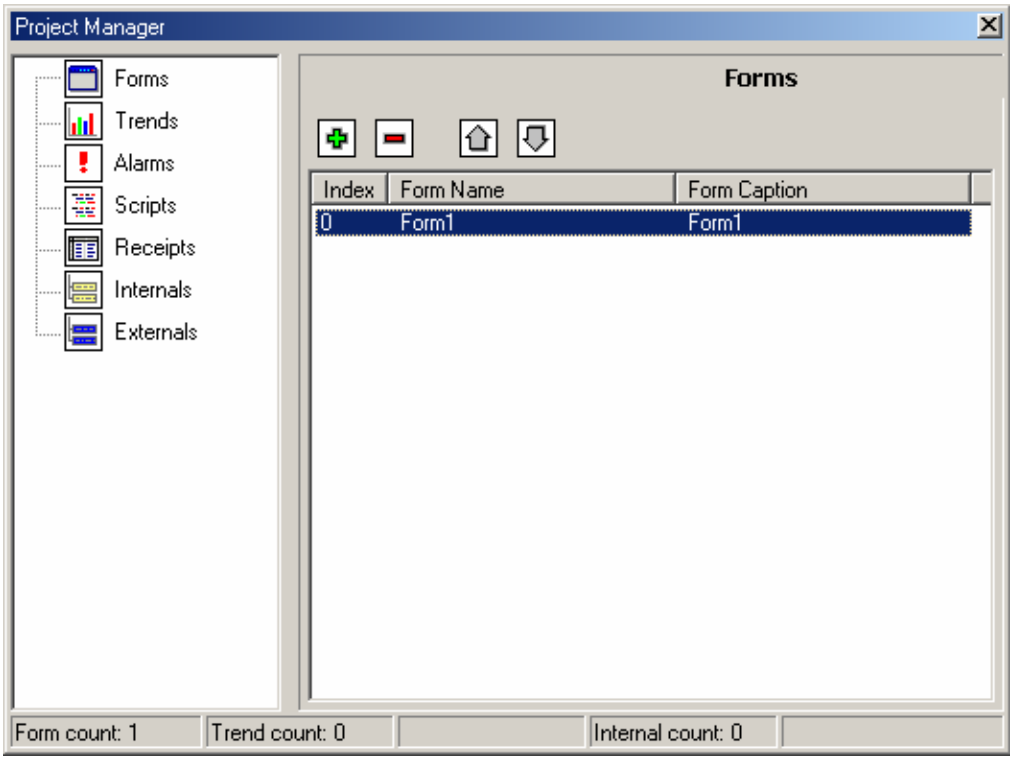
Eagleye Scada Designer ilk bakışta tanıdık Windows programları gibidir.Yeni bir proje başlatmak için File/New Project menüsünden “New Project” penceresi açılır.



“New Project” penceresinde projeye verilecek isim ve kaydedileceği dizin seçilir. Ayrıca programı yapan kişi hakkında da kısa bir bilgi alır. Bu bilgiler daha

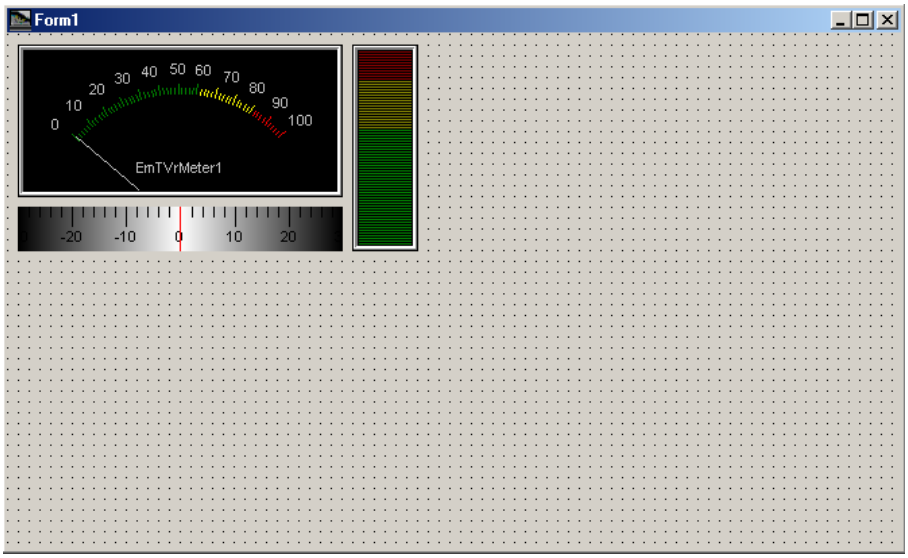
sonra program runtime çalıştığında ilk açılışta bilgi penceresinde (Splash) gösterilir.

"OK" tuşuna basıldıktan sonra "Project Manager" penceresinden projeye gerekli olan adımları ekleyebiliriz."Project Manager" penceresine F12 kısa yol tuşu veya "View/Project Manager" menüsünden ulaşılabilir.



Şekil 3.1 Project Manager

Yeni bir Form yaratarak birden fazla sayfaya sahip bir program yazabilirsiniz. Tasarım görünümündeki bir form aşağıdaki gibidir.



Şekil 3.2 Tasarım Formu

3.2 Kısa yol Tuşları

Projedeki önemli işlemlere bir tuşa basarak ulaşmanızı sağlar.

 Yeni proje başlatır. (File / New Project)

 Var olan bir projeyi açar. (File / Open Project)

 Projeyi kaydeder. (File / Save Project)

 Projeye yeni bir Form ekler.

 "Project Manager" sayfasın açar.

 Projeyi "Run" konumuna getirir.Proje çalışır.

3.3 Hizalama Araçları (Alignment Box)

Bu araçlar sayesinde Form üzerine yerleştirilen çok sayıda nesne bir tuşa basarak istenilen düzene getirilir.

 Seçilen tüm nesneleri sol tarafa yaslar.

 Seçilen tüm nesneleri sağ tarafa yaslar.

 Seçilen tüm nesneleri yukarı tarafa yaslar.

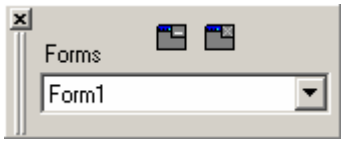
 Seçilen tüm nesneleri aşağı tarafa yaslar.

 Seçilen tüm nesneleri eşit aralıklarla yatayda sıralar.

 Seçilen tüm nesneleri eşit aralıklarla dikeyde sıralar.

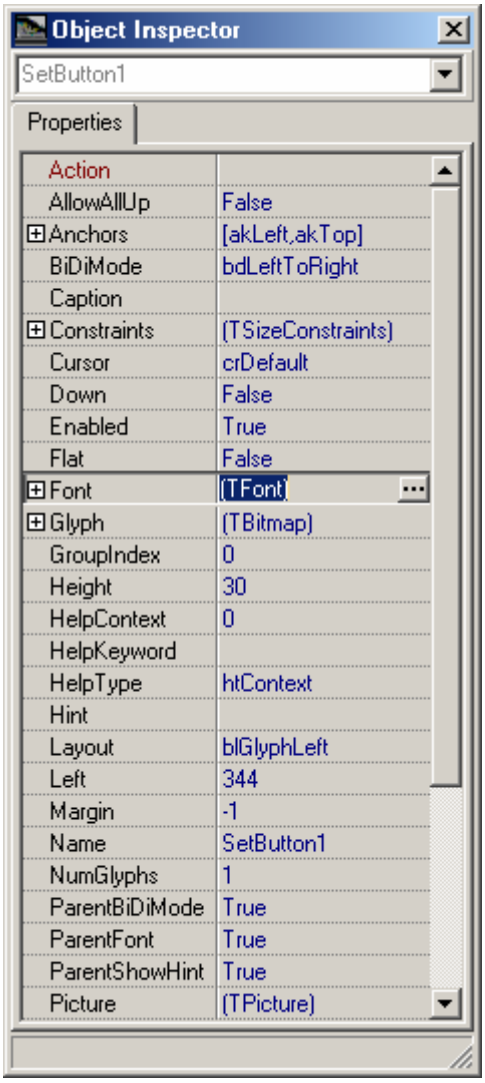
3.4 Tasarım Form Kutusu (Forms Box)

Bu araç sayesinde tasarım anında istenilen Form'a ulaşılabilir. İstenirse tüm tasarım Form'ları kapatılabilir.



3.5 Özellik Penceresi (Object Inspector)

"Form" üzerine konulan nesnelerin özelliklerini değiştirmeye yarayan penceredir. Seçilen nesnelerin özellikleri listelenip istenilen ayarlar yapılabilir.



Şekil 3.3 Object Inspector

Özellik	Açıklama
Caption	Nesne üzerindeki yazıyı taşır.
Font	Nesne üzerindeki yazının stilini,rengini ve boyutunu temsil eder.
Height	Nesnenin form üzerindeki yüksekliğini temsil eder.
Left	Nesnenin form üzerindeki sol tarafa olan uzaklığını temsil eder.
Name	Nesneye scripten ulaşmak için verilen ismi temsil eder.
Top	Nesnenin ekranın üstüne olan uzaklığını temsil eder.
Width	Nesnenin genişliğini temsil eder.
ValueTag	Nesnenin etki edeceği tagı temsil eder.
Visible	Nesnenin ekranda görünürlüğünü temsil eder.

3.6 Nesne Kutusu (Component Box)

"Form" üzerinde görsel tasarımlar yapmak için gerekli nesneleri barındırır. Çeşitli amaçlar için geliştirilmiş nesnelerle kullanıcı ara yüzleri oluşturulur. Bu nesneler çeşitli kategorilerde toplanmıştır.

Şekil 3.4 Nesne Kutusu

Set Button: Bu tuş sayesinde bağlı olduğu tag'ın değerini 1 'e set eder. Hiç bir kod yazmadan ValueTag özelliğine atanan tagın değerini değiştir. On/OFF kontrollü cihazların açılma komutunu verir.

Reset Button: Bu tuş bağlı olduğu tagın değerini 0'a set eder. ON/OFF kontrollü cihazların kapanma komutunu verir.


Maintained Button: Bu tuş kalıcı tuş olarak da adlandırılır. Tuş ilk bakışta 1 değerini set eder. Tekrar basılması durumunda 0 değerini set eder. ON/OFF kontrollü cihazlarda hem açma hem de kapatma işlemini tek tuşla yapmayı sağlar.


Momentary Button: Anlık tuş diye de adlandırılan bu tuş tetik görevi yapar. Basılı kaldığı sürece bağlı olduğu tag'a 1 değerini set eder. Bırakıldığında 0 değerini gönderir.


MultiState Button: Bir değişken için birden fazla set etme seçeneğini barındıran çok fonksiyonlu bir tuşdur.MultiType özelliğinde 3 seçenek bulunur.Bunlar,

- Const
  ConstValue özelliğine atanan değeri tuşa her basışta bağlı olduğu tag'a set eder.
- Increment
  StepValue özelliğine atanan değeri tuşa her basışta bağlı olduğu tag'a ekler. Tag değerinin Max değerini geçmesi durumunda tag değeri Min değerine döner.
  TagValue=Value + StepValue
- Decrement
  StepValue özelliğine atanan değeri tuşa her basışta bağlı olduğu tag'dan çıkarır. Tag değerinin Min değerinin altına düşmesi durumunda tag değeri Max değerine döner.
  TagValue=Value – StepValue

 **GoTo Button:** Birden fazla Form içeren uygulamalarda sayfa değiştirmek için kullanılan bir tuştur. GoPage özelliğine atanan Form'a geçmeyi sağlar.

 **Label:** Sayfa üzerinde açıklama yazılarının gösterilmesi için kullanılır. Caption özelliğine atanan alfa nümerik değer ile ekranda bilgi verilebilir.

 **ValueEdit:** Sayfa üzerinde bağlı olduğu tag'ın değerini gösterme ve tag'a istenen değeri aktarmayı sağlar.

 **ReadOnlyEdit:** Sayfa üzerinde bağlı olduğu tag'ın değeri gösterir. Ama tag değerine müdahale edilemez. Yalnızca izleme amaçlıdır.

 **Meter:** Tag değerini kadran şeklinde gösteren bir nesnedir. Verilen limit değerlerine göre renk değiştirir.

 **Needle:** Tag değerini kadran şeklinde gösteren bir nesnedir. Verilen limit değerlerine göre renk değiştirir.

 **JogMeter:** Tag değerini sayaç görünümünde kullanıcıya gösterir. Animasyonlu gösterim şekli vardır.

 **ThermoMeter:** Tag değeri termometre şeklinde gösterir. Sıcaklık değerlerinin gösteriminde kullanılır. Limit değerlerde farklı renkler alarak kullanıcıyı uyarır.

 **PowerMeter:** Tag değerini gösteren ve göstergesinde birim bulunduran görsel bir nesnedir.

 **LevelBar:** Tag değerini seviye görünümünde sunar. Su seviyesi bilgisini göstermekte kullanılır.

 **Slider:** Tag değerini seviye görünümünde sunar. Ayrıca tag değerini üzerindeki tuşla anında değiştirebilirsiniz.

 **Image:** Tag değerine göre resim gösterir.7 tane farklı resim saklayabilir. Tagın değerine göre istenen resim gösterilir. Örneğin tag değeri 1 iken 1. resim 2 ise 2. resim gösterilir. Birden fazla konuma sahip saha elemanlarının gösteriminde

kullanılır. Bir motorun çalışması, durması veya arıza konumunda olması farklı renkteki resimlerle gösterilebilir.

MobileImage: Bu nesnenin taşıdığı resim sayfa üzerinde hareket ettirilebilir. Böylece yürüyen bant gibi sistemlerin animasyonu yapılabilir. Bu hareketler için RegLeft,RegTop özellikleri kullanılır.

Chart: Oluşturulan trendlerin çevrimiçi olarak izlenmesini sağlayan bir nesnedir.8 kalem bilgiyi aynı anda ekranda gösterebilir. İstenilen zaman birimlerinde gösterim sağlanabilir.

Şekil 3.5 Trend gösterim nesnesi

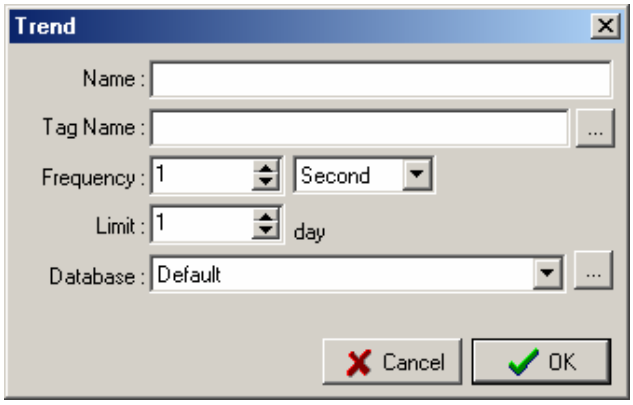
Bu nesne üzerinden görüntü 3 boyuta çıkartılabilir. Ayrıca "zoom" ve grafiği kaydırma mümkündür. İstenilen zamanda grafik görüntüsü yazdırılabilir veya Bitmap resim formatında bilgisayara kaydedilebilir. Seçilen zaman aralıklarında ok tuşları ile hareket edilebilir.

EventButton: "Script" olarak yazılan bir kodun bu tuşa basılmak suretiyle çalıştırılmasını sağlar. ClickEvent özelliğine bir kod atanabilir.

Timer: "Script" olarak yazılan bir kodun belli bir zaman aralıklarında düzenli olarak çalıştırılmasını sağlar.

3.7 Yeni Trend Oluşturma

"Project Manager" penceresinden "Trends" bölümüne gelinir ve "Add" tuşuna basılır.

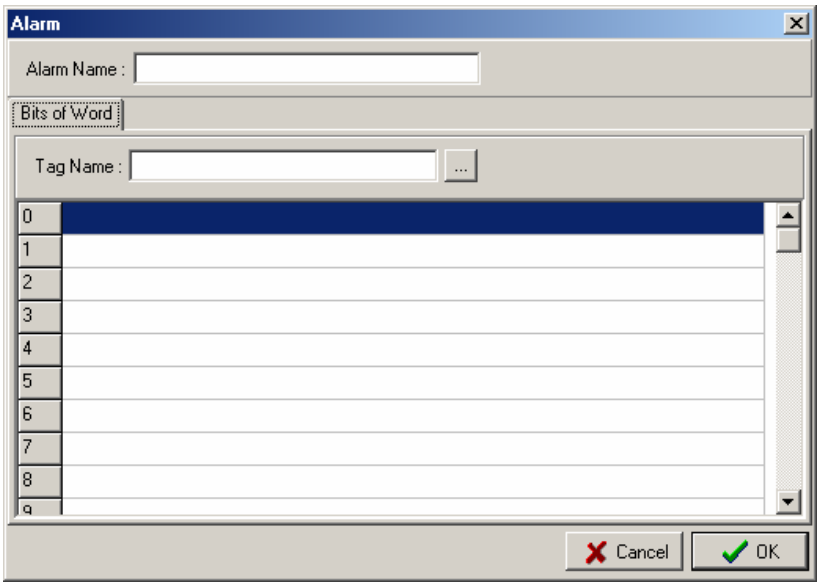


Şekil 3.6 Yeni Trend tanımlama

Name	Oluşturulan "Trend" e verilecek isimdir. Bu isim Chart nesnesine bağlamak için kullanılır.
Tag Name	Trendin kaydedeceği tag'ın adıdır. Yandaki tuşa basılarak seçim kutusundan daha önce yaratılan bir tag seçilir.
Frequency	Trend'e eklenen tagın ne kadarlık bir zaman sıklığı ile kaydedileceğini belirler.
Limit	Kayıt uzunluğunun kaç günlük olduğunu belirler. Kuyruk tipinde çalışır. Verilen limit kadar geri tarihte bilgi saklanır.
Database	Tag değerlerinin saklanacağı veritabanını belirler.

3.8 Yeni Alarm Oluşturma

"Project Manager" penceresinden "Alarms" bölümüne gelinir ve "Add" tuşuna basılır.



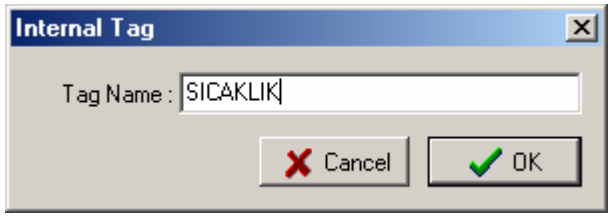
Şekil 3.7 Yeni Alarm tanımlama

Alarm Name	Oluşturulan alarmın adını belirler.
Tag Name	Alarmın oluşturulacağı tagı belirler. Yandaki tuşa basılarak seçim kutusundan daha önce yaratılan bir tag seçilir.

"Tag Name" özelliğine atanan tagın bir Word olması beklenir. Bu 1 Word lük değerin her bir bitine bir alarm mesajı bağlanabilir. Bu bitlerin aktif yani 1 olması durumunda SCADA bunun bir alarm olduğunu varsayar ve ilgili mesajı ekranda gösterir.

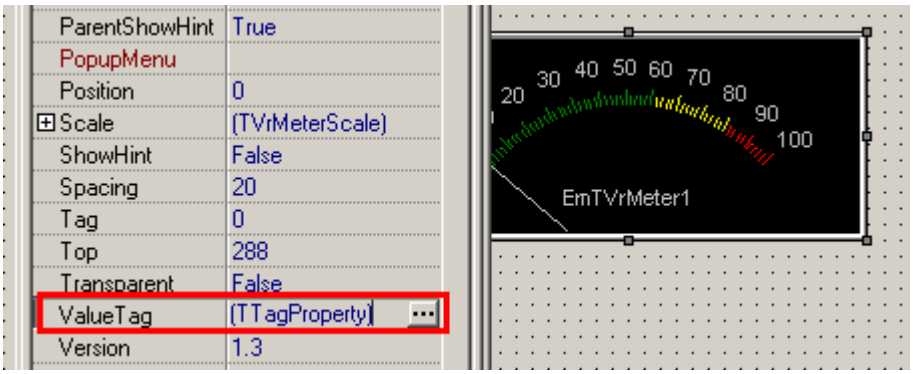
3.9 Yeni Internal Tag (İç Değişken) Oluşturma

"Project Manager" penceresinden "Internals" bölümüne gelinir ve "Add" tuşuna basılır.



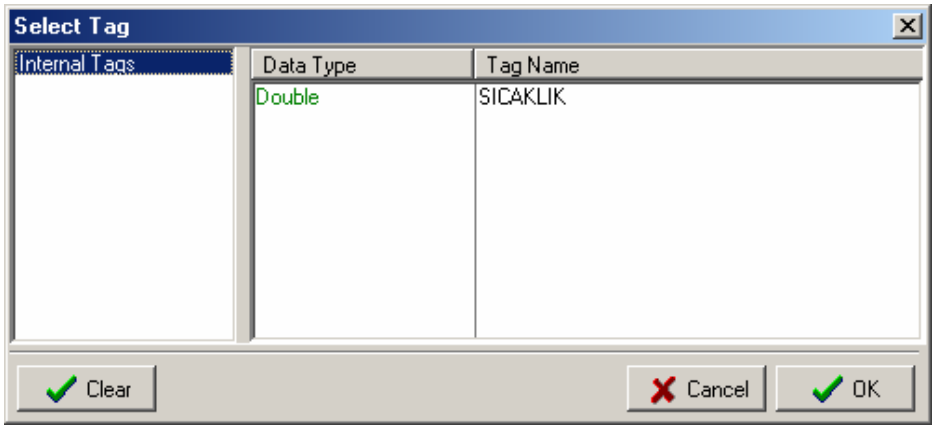
Şekil 3.8 Yeni Internal Tag Tanımlama

Bir isim verilerek istenildiği kadar “Internal Tag” yaratılabilir.Bu tagların tipi double dır.Sahadan gelen bir bilgi veya program içinden oluşturulacak bir değer bu değişkenlere atanabilir.Bu tagların nesnelere atanması ise “Select Tag Editor” ile gerçekleştirilir.



Şekil 3.9 Select Tag Editor Seçimi

“Object Inspector” penceresinde ilgili nesne seçildikten sonra “ValueTag” özelliğinin yanındaki “…” tuşuna basarak bu Editor açılabilir. Editor aşağıdaki gibidir:

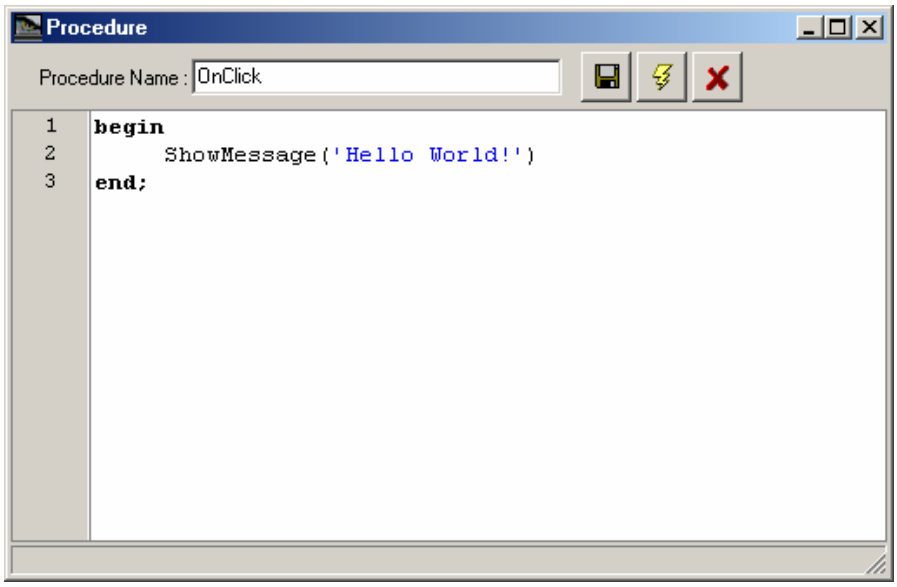


Şekil 3.10 Select Tag Editor

Sol tarafta Tag grupları listelenir.”Internal Tags” dışında ilerde anlatılacak “External Tags” larda burada grup olarak gösterilir. Bu gruplara Mouse ile tıklandığında sağ tarafta tanımlanmış olan taglar listelenir. Veri tipleri ve isimleri buradan kolaylıkla anlaşılır ve yine Mouse ile seçilerek “OK” tuşuna basılır ve atama gerçekleşir.

3.10 Yeni Script Oluşturma

"Project Manager" penceresinden "Scripts" bölümüne gelinir ve "Add" tuşuna basılır.



Şekil 3.11 Yeni Script Yazma

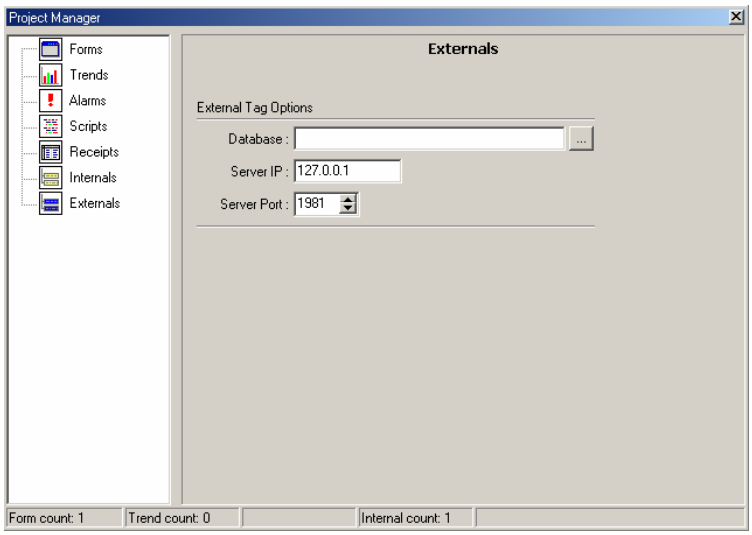
Birden fazla kod blokları oluşturulup nesnelere atanabilir. Script'de pascal sentaksı kullanılır. Eagleye Scada'nın kendine özel fonksiyonları da vardır. Bunlardan en önemlileri:

GetTagValue(**Tag**: string):Variant;
SetTagValue(**Tag**: string; **Value**: Variant);

fonksiyonlarıdır. Bu fonksiyonlar sayesinde kod içinden istenilen tagın değeri alınabilir veya yeni bir değer atanabilir.

3.11 External Tags (Dış Değişkenler) Oluşturma

"Project Manager" penceresinden "Externals" bölümüne gelinir ve ayarlar yapılır.

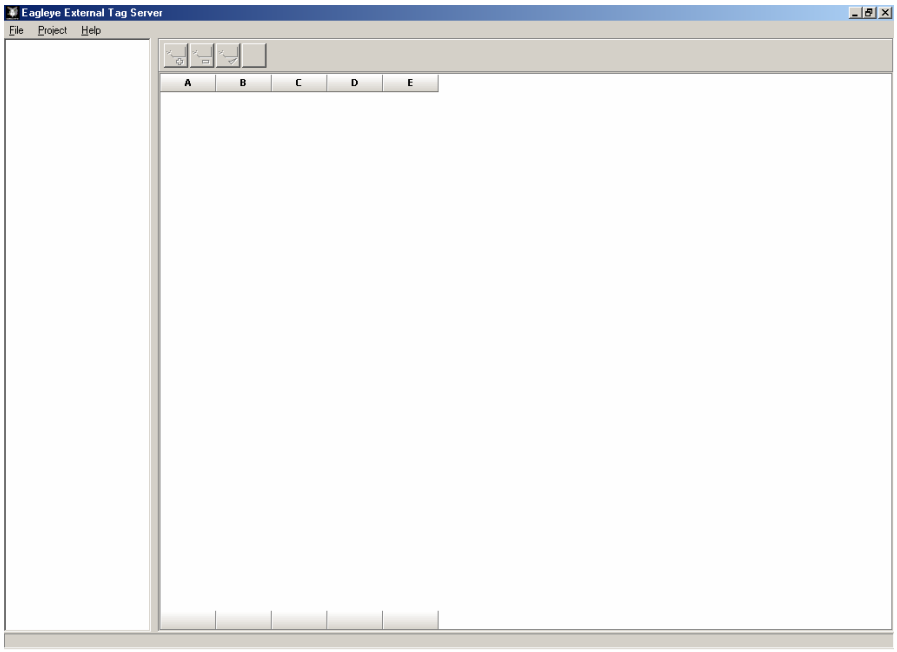


Şekil 3.12 External Tag Ayarları

External Taglar "Eagleye External Tag Server" programında tanımlanır ve bir Access dosyası olarak saklanır. Designer kısmında bu Access dosyası seçildiğinde tanımlanmış olan taglar projeye dâhil olur. İleride bahsedeceğimiz "Eagleye External Tag Server" adlı program Designer programlarına veriyi TCP/IP protokolü ile iletir. Böylece birden fazla SCADA programı tek sunucu programından verileri alabilir. TCP/IP protokolünün gereği olarak burada bir IP numarası ve Port tanımlanmalıdır. Eğer SCADA cihazlara bağlanan bilgisayar üzerinde ise bu IP numarası localhost yani 127.0.0.1 olmalıdır.

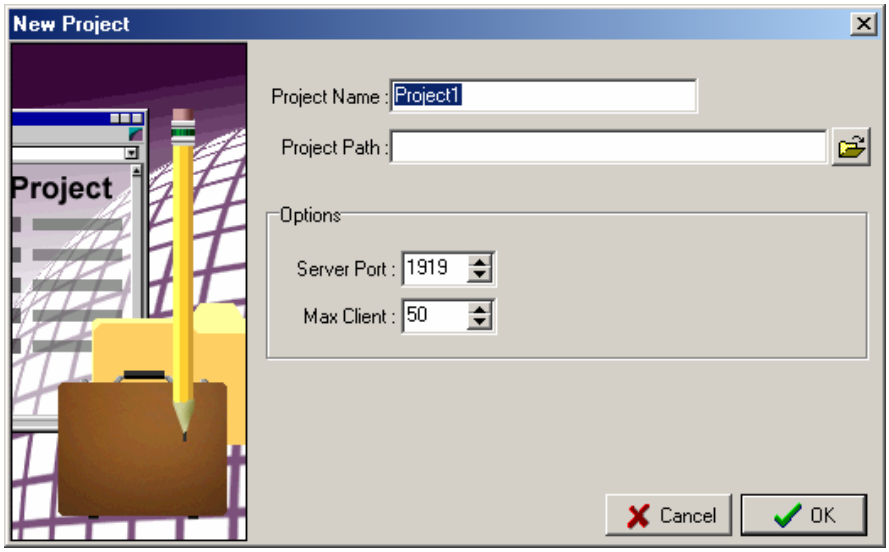
4. Eagleye External Tag Server

Bu program cihazlarla haberleşecek tagların tanımlanmasını ve gerekli protokol ayarlarının kolayca yapılmasını sağlar. Ayrıca Runtime anında verileri cihazlardan optimize bir şekilde okur ve Designer programı veya programlarına dağıtır.



Şekil 4.1 Eagleye External Tag Server

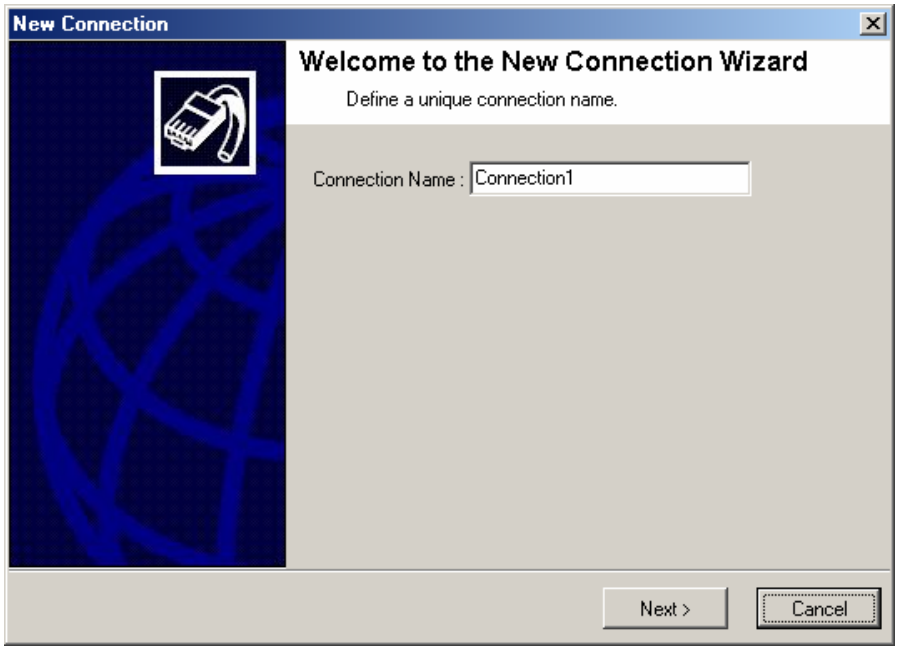
İlk başta File/New Project menülerinden yeni bir proje başlatılır.Bu proje bir Access dosyasıdır ve eklenecek olan protokol ve tag tanımlarını saklar.



Şekil 4.2 Yani Bağlantı Ekle

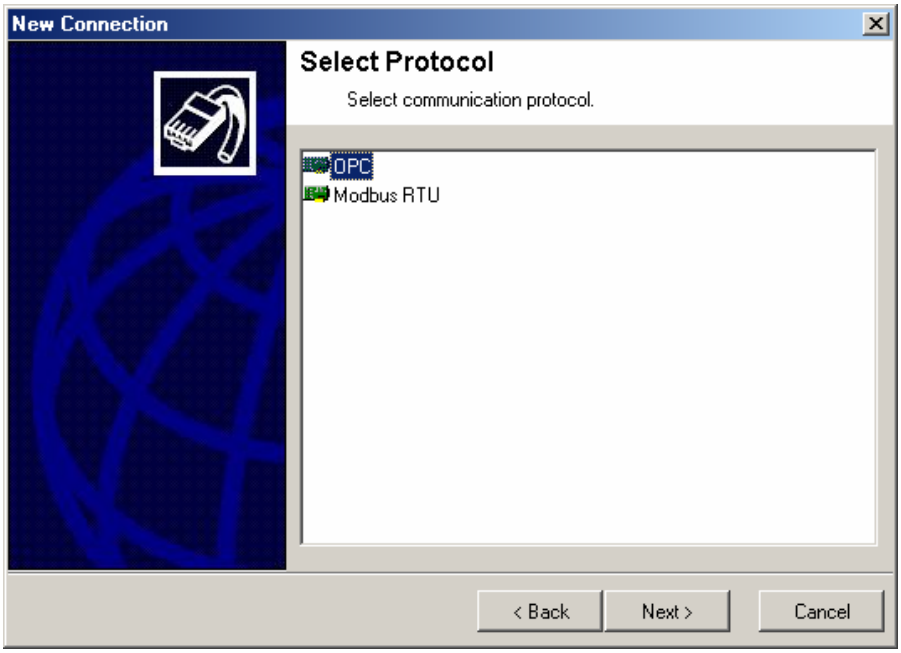
Projenin ismi ve kaydedileceği dizin buradan seçilir. Ayrıca eğer birden fazla Designer'a hizmet verecekse "Max Client" özelliğinden limitler belirlenebilir.

Yeni bir proje oluşturduktan sonra ki adım yeni bir "Connection" oluşturmaktır."Project/Add New Connection" menusu kullanılarak "New Connection Wizard" açılır.



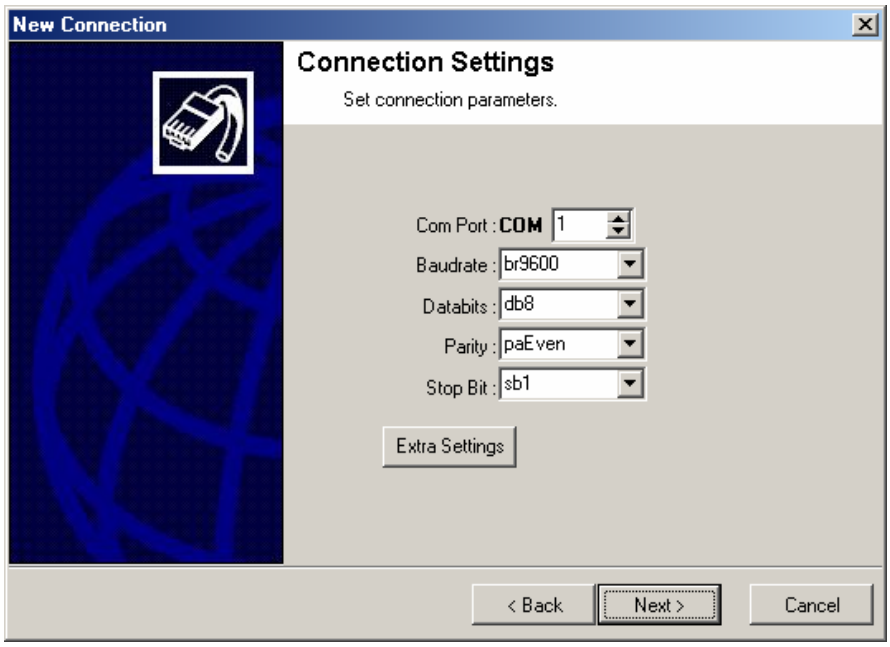
Şekil 4.3 Bağlantı Adı Tanımlama

Tanımlanacak olan "Connection" a bir isim verilir. Bu isim her bağlantı için farklı olmalıdır. Ayrıca boş bırakılamaz ve "Internal", "Connections" isimleri verilemez."Next" tuşuna basıldıktan sonra karşımıza protokol listesi çıkar:



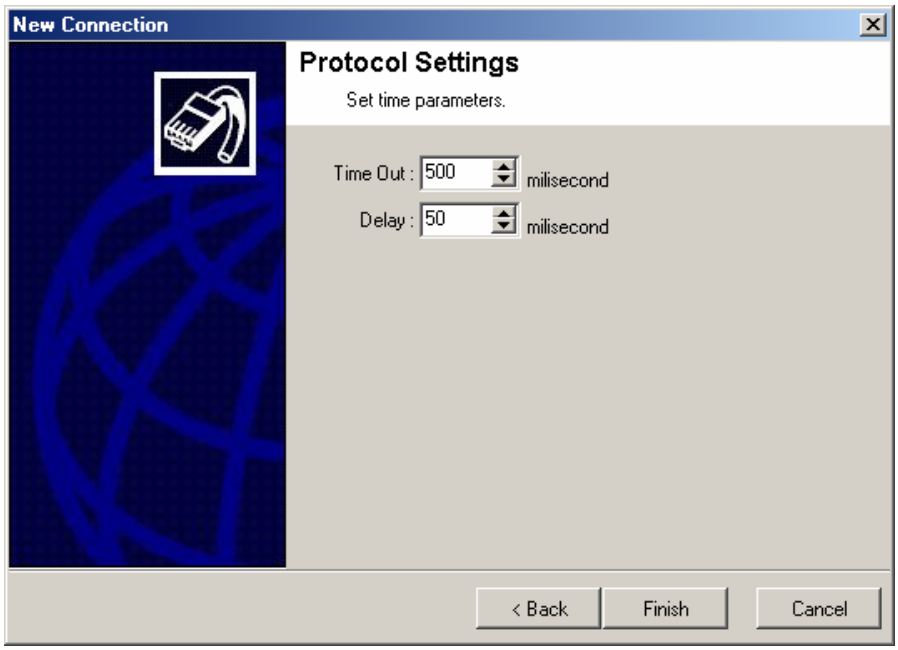
Şekil 4.4 Protokol Seçimi

Bu listeden istenilen protokol seçilir.”Next” tuşuna basarak bu protokolün ayarları yapılır.



Şekil 4.5 Protokol Ayarları

Protokol için gerekli olan fiziksel bilgiler buradan girilir. Hangi COMPORT ile haberleşeceği, baud rate, parity gibi bilgiler buradan ayarlanır.

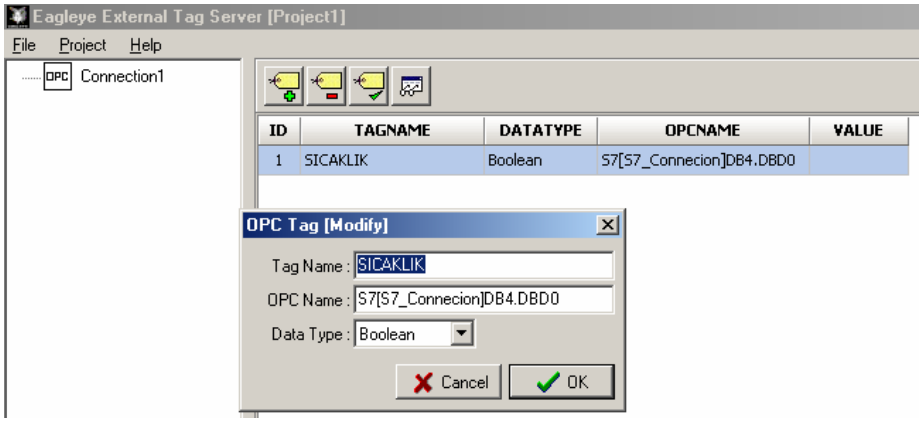


Şekil 4.6 Protokol Zaman Ayarları

Bir sonraki sayfada ise tüm protokollerdeki ortak özellik olan “TimeOut” ve “Delay” bilgileri girilir.

TimeOut: Bir bilgi istendikten sonra cevabın gelmesi için tahammül edilen süredir. Eğer bu süre içinde cevap gelmezse başka bir soruya geçilir.10 defa cevap gelmezse kullanıcıya bağlantı ile ilgili bir sorun olduğu alarmı verilir. Bunun sebebi büyük bir ihtimalle kablo bağlantısındaki kesikliktir.

Delay: Cihazların bir soruyu alıp karşılığında cevap vermesinden sonra yazılımın 2. soru için hazır konuma gelme süresidir.Genelde 50 ms cok yeterli bir süredir.Delay zamanı ile 2 soru arasındaki bekleme zamanı ayarlanır ve böylece sağlıklı bir iletişim sağlanmış olur.



Şekil 4.7 External Tag Tanımlama

Daha sonra oluşturulan bağlantıya yeni taglar eklenerek Designer programının kullanımına verilir.

5. Designer Kod Şablonu

5.1 Designer Nesnesi

```
TCustomDesigner = class(TComponent)
private
  FEng: Pointer;
  FSnapToGrid: Boolean;
  FActive: Boolean;
  FOnControlInserted: TNotifyEvent;
  FOnControlInserting: TELDesignerOnControlInsertingEvent;
  FGrid: TELDesignerGrid;
  FSelectedControls: TELDesignerSelectedControls;
  FDesignPanel: TELCustomDesignPanel;
  FActivating: Boolean;
  FOnModified: TNotifyEvent;
  FOnChangeSelection: TNotifyEvent;
  FOnValidateName: TELDesignerOnValidateNameEvent;
  FDesignControl: TWinControl;
  FOnControlHint: TELDesignerOnControlHintEvent;
  FShowingHints: TELDesignerHintTypes;
  FOnDesignFormClose: TCloseEvent;
  FPopupMenu: TPopupMenu;
  FOnContextPopup: TContextPopupEvent;
  FHandleClr: TColor;
  FMultySelectHandleClr: TColor;
  FHandleBorderClr: TColor;
  FInactiveHandleClr: TColor;
  FInactiveHandleBorderClr: TColor;
  FLockedHandleClr: TColor;
  FMultySelectHandleBorderClr: TColor;
  FClipboardFormat: string;
  FClipbrdFormatRegistered: Boolean;
  FClipboardFormatWord: Word;
  FOnKeyUp: TKeyEvent;
  FOnKeyDown: TKeyEvent;
  FOnKeyPress: TKeyPressEvent;
  FOnNotification: TELDesignerOnNotificationEvent;
  FOnGetUniqueName: TELDesignerOnGetUniqueName;
  FOnDblClick: TNotifyEvent;
  FOnDragDrop: TELDesignerDragDropEvent;
  FOnDragOver: TELDesignerDragOverEvent;
  procedure SetActive(const Value: Boolean);
  procedure SetDesignControl(const Value: TWinControl);
  procedure SetDesignPanel(const Value: TELCustomDesignPanel);
  function GetDesignControlVisible: Boolean;
  procedure SetDesignControlVisible(const Value: Boolean);
  procedure ChangeDesignPanel(const Value: TELCustomDesignPanel);
```

```
    procedure SetPopupMenu(const Value: TPopupMenu);
    procedure SetHandleBorderClr(const Value: TColor);
    procedure SetHandleClr(const Value: TColor);
    procedure SetInactiveHandleBorderClr(const Value: TColor);
    procedure SetInactiveHandleClr(const Value: TColor);
    procedure SetLockedHandleClr(const Value: TColor);
    procedure SetMultySelectHandleBorderClr(const Value: TColor);
    procedure SetMultySelectHandleClr(const Value: TColor);
    procedure SetClipboardFormat(const Value: string);
    procedure SetGrid(const Value: TELDesignerGrid);
    function DoKeyDown(var AMessage: TWMKey): Boolean;
    function DoKeyPress(var AMessage: TWMKey): Boolean;
    function DoKeyUp(var AMessage: TWMKey): Boolean;
    procedure GridParamsChanged;
    procedure RegisterClipboardFormat;
  protected
    procedure Notification(AComponent: TComponent; Operation: TOperation);
override;
    procedure CheckActive(AIsActiveNeeded: Boolean);
    function GetPopupMenu: TPopupMenu; dynamic;
    procedure DoModified; virtual;
    procedure ValidateName(const AName: string; var AIsValidName: Boolean);
virtual;
    procedure GetUniqueName(const ABaseName: string; var AUniqueName:
string); virtual;
    procedure ChangeSelection; virtual;
    procedure ControlInserting(var AControlClass: TControlClass); virtual;
    procedure ControlInserted; virtual;
    procedure DoNotification(AnObject: TPersistent; Operation: TOperation);
virtual;
    procedure ControlHint(AControl: TControl; var AHint: string); virtual;
    procedure ContextPopup; virtual;
    procedure DesignFormClose(var Action: TCloseAction); virtual;
    procedure KeyDown(var Key: Word; Shift: TShiftState); virtual;
    procedure KeyPress(var Key: Char); virtual;
    procedure KeyUp(var Key: Word; Shift: TShiftState); virtual;
    procedure DblClick; virtual;
    procedure DragOver(ASource, ATarget: TObject; AX, AY: Integer; AState:
TDragState;
      var AAccept: Boolean); dynamic;
    property Active: Boolean read FActive write SetActive;
    property DesignControl: TWinControl read FDesignControl write
SetDesignControl;
    property SelectedControls: TELDesignerSelectedControls read
FSelectedControls;
    property DesignControlVisible: Boolean read GetDesignControlVisible write
SetDesignControlVisible;
    property DesignPanel: TELCustomDesignPanel read FDesignPanel write
SetDesignPanel;
    property Grid: TELDesignerGrid read FGrid write SetGrid;
```

```
    property SnapToGrid: Boolean read FSnapToGrid write FSnapToGrid default
True;
    property ShowingHints: TELDesignerHintTypes read FShowingHints write
FShowingHints
      default [htControl, htSize, htMove, htInsert];
    property PopupMenu: TPopupMenu read FPopupMenu write SetPopupMenu;
    property HandleClr: TColor read FHandleClr write SetHandleClr default
clBlack;
    property HandleBorderClr: TColor read FHandleBorderClr write
SetHandleBorderClr default clBlack;
    property MultySelectHandleClr: TColor read FMultySelectHandleClr write
SetMultySelectHandleClr default clGray;
    property MultySelectHandleBorderClr: TColor read
FMultySelectHandleBorderClr write SetMultySelectHandleBorderClr default
clGray;
    property InactiveHandleClr: TColor read FInactiveHandleClr write
SetInactiveHandleClr default clGray;
    property InactiveHandleBorderClr: TColor read FInactiveHandleBorderClr
write SetInactiveHandleBorderClr default clBlack;
    property LockedHandleClr: TColor read FLockedHandleClr write
SetLockedHandleClr default clRed;
    property ClipboardFormat: string read FClipboardFormat write
SetClipboardFormat;
    property OnModified: TNotifyEvent read FOnModified write FOnModified;
    property OnValidateName: TELDesignerOnValidateNameEvent read
FOnValidateName write FOnValidateName;
    property OnGetUniqueName: TELDesignerOnGetUniqueName read
FOnGetUniqueName write FOnGetUniqueName;
    property OnControlInserting: TELDesignerOnControlInsertingEvent read
FOnControlInserting write FOnControlInserting;
    property OnControlInserted: TNotifyEvent read FOnControlInserted write
FOnControlInserted;
    property OnNotification: TELDesignerOnNotificationEvent read
FOnNotification write FOnNotification;
    property OnChangeSelection: TNotifyEvent read FOnChangeSelection write
FOnChangeSelection;
    property OnControlHint: TELDesignerOnControlHintEvent read
FOnControlHint write FOnControlHint;
    property OnDesignFormClose: TCloseEvent read FOnDesignFormClose write
FOnDesignFormClose;
    property OnContextPopup: TContextPopupEvent read FOnContextPopup write
FOnContextPopup;
    property OnKeyDown: TKeyEvent read FOnKeyDown write FOnKeyDown;
    property OnKeyPress: TKeyPressEvent read FOnKeyPress write
FOnKeyPress;
    property OnKeyUp: TKeyEvent read FOnKeyUp write FOnKeyUp;
    property OnDblClick: TNotifyEvent read FOnDblClick write FOnDblClick;
    property OnDragDrop: TELDesignerDragDropEvent read FOnDragDrop write
FOnDragDrop;
```

```
    property OnDragOver: TELDesignerDragOverEvent read FOnDragOver write
FOnDragOver;
  public
    constructor Create(AOwner: TComponent); override;
    destructor Destroy; override;
    procedure DragDrop(ASource, ATarget: TObject; AX, AY: Integer); dynamic;
    procedure Modified;
    procedure DeleteSelectedControls;
    procedure LockControl(AControl: TControl; ALockMode:
TELDesignerLockMode);
    procedure LockAll(ALockMode: TELDesignerLockMode);
    function GetLockMode(AControl: TControl): TELDesignerLockMode;
    function IsUniqueName(const AName: string): Boolean;
    function CanCopy: Boolean;
    function CanCut: Boolean;
    function CanPaste: Boolean;
    procedure Cut;
    procedure Copy;
    procedure Paste;
  end;
```

5.2 Object Inspector Nesnesi

```
 TELPropEditor = class
 private
   FPropList: PELPropEditorPropList;
   FPropCount: Integer;
   FOnModified: TNotifyEvent;
   FOnGetComponent: TELOnGetComponent;
   FOnGetComponentNames: TELOnGetComponentNames;
   FOnGetComponentName: TELOnGetComponentName;
   FDesigner: Pointer;
   function GetPropTypeInfo: PTypeInfo;
   function DoGetValue: string;
 protected
   procedure SetPropEntry(AIndex: Integer; AInstance: TPersistent; APropInfo:
PPropInfo);
   function GetComponent(const AComponentName: string): TComponent;
   procedure GetComponentNames(AClass: TComponentClass; AResult:
TStrings);
   function GetComponentName(AComponent: TComponent): string;
   function GetValue: string; virtual;
   procedure SetValue(const Value: string); virtual;
   function GetAttrs: TELPropAttrs; virtual;
   procedure GetValues(AValues: TStrings); virtual;
   procedure GetSubProps(AGetEditorClassProc: TELGetEditorClassProc;
AResult: TList); virtual; // Returns a list of TELPropEditor
   function GetPropName: string; virtual;
   function AllEqual: Boolean; virtual;
   procedure Edit; virtual;
   procedure ValuesMeasureHeight(const AValue: string; ACanvas: TCanvas; var
AHeight: Integer); virtual;
   procedure ValuesMeasureWidth(const AValue: string; ACanvas: TCanvas; var
AWidth: Integer); virtual;
   procedure ValuesDrawValue(const AValue: string; ACanvas: TCanvas; const
ARect: TRect; ASelected: Boolean); virtual;
 protected
   function GetPropInfo(AIndex: Integer): PPropInfo;
   function GetInstance(AIndex: Integer): TPersistent;
   function GetFloatValue(AIndex: Integer): Extended;
   function GetInt64Value(AIndex: Integer): Int64;
   function GetOrdValue(AIndex: Integer): Longint;
   function GetStrValue(AIndex: Integer): string;
   function GetVarValue(AIndex: Integer): Variant;
   procedure SetFloatValue(Value: Extended);
   procedure SetInt64Value(Value: Int64);
   procedure SetOrdValue(Value: Longint);
   procedure SetStrValue(const Value: string);
   procedure SetVarValue(const Value: Variant);
 public
   constructor Create(ADesigner: Pointer; APropCount: Integer); virtual;
```

```
    destructor Destroy; override;
    procedure Modified;
    property PropName: string read GetPropName;
    property PropTypeInfo: PTypeInfo read GetPropTypeInfo;
    property PropCount: Integer read FPropCount;
    property Value: string read DoGetValue write SetValue;
    property Designer: Pointer read FDesigner;
    property OnModified: TNotifyEvent read FOnModified write FOnModified;
    property OnGetComponent: TELOnGetComponent read FOnGetComponent
write FOnGetComponent;
    property OnGetComponentNames: TELOnGetComponentNames read
FOnGetComponentNames write FOnGetComponentNames;
    property OnGetComponentName: TELOnGetComponentName read
FOnGetComponentName write FOnGetComponentName;
  end;
```

6. SONUÇLAR

Eagleye SCADA, PLC eğitimi veren kurumlarda PLC ve SCADA entegrasyon konularında kullanılmaktadır.3E mühendislik şirketinin desteğinde piyasanın isteklerine cevap verecek şekilde günden güne gelişmektedir.Ürünün bir sonraki adımı 3D seviyesinde dünyada ilk SCADA olmaktır.3 Boyutlu mimariye geçişte önemli bir basamak ve alt yapıyı bünyesinde barındıran Eagleye SCADA teknoloji geliştikçe gelişmeye devam edecektir.

KAYNAKLAR

[1] www.tmssoftware.com, Tms Software VCL kütüphaneleri

[2] www.torry.net, Delphi kod kaynağı

[3] www.truevision3d.com,

[4] www.modbus.org

[5] www.opcconnect.org

[6] www.delphi.about.com

[7] www.swissdelphi.ch

[8] www.intel.com/imageprocessing

[9] www.delphi3000.com

[10] **Delphi Unleashed**,2004 Delphi Unleashed - Charlie Clarvert,