İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

SÜREÇ KONTROLUNDA NESNELERİN BAĞLAŞMASI VE İLİŞKİLENDİRİLMESİ (OPC) STANDARDI VE UYGULAMASI

YÜKSEK LİSANS TEZİ

Müh. Yusuf ÜNLÜ

Anabilim Dalı : ELEKTRİK MÜHENDİSLİĞİ

Programı : KONTROL VE OTOMASYON MÜHENDİSLİĞİ

EKİM 2007

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

SÜREÇ KONTROLUNDA NESNELERİN BAĞLAŞMASI VE İLİŞKİLENDİRİLMESİ (OPC) STANDARDI VE UYGULAMASI

YÜKSEK LİSANS TEZİ

Müh. Yusuf ÜNLÜ
504041120

Tezin Enstitüye Verildiği Tarih : 10 Eylül 2007
Tezin Savunulduğu Tarih : 2 Ekim 2007

Tez Danışmanı : Doç.Dr. Cevat ERDAL
Diğer Jüri Üyeleri Doç.Dr. M. Turan SÖYLEMEZ
Yrd. Doç. Dr. O. Kaan EROL

EKİM 2007

ÖNSÖZ

Gelişen teknolojik imkanlar ve donanım maliyetlerinde meydana gelen büyük düşüşler, endüstriyel alanda yapılan yatırımlarda donanım kısmına ayrılan payın azalmasını sağlarken, bilgiyi etkin biçimde kullanabilmeye ayrılan payın da giderek artmasına neden olmuştur. Bu açıdan bu sektörde bir çok yazılım geliştirilmiştir.

Bu çalışmada endüstriyel otomasyon dünyasında belki de en hızlı gelişen standart olan OPC Standardı'nın önemi ve uygulama yapısı hakkında bilgi vermeye çalıştım. Ayrıca bu çalışmanın akademik dünyada üretilen çözümlerin endüstriyel dünyaya daha rahat uygulanmasını sağlaması bakımından önemli bir başlangıç olacağını ümit etmekteyim.

Bu konu hakkında çalışmama değerli katkıları olan danışman hocam Doç. Dr. Cevat Erdal'a, uygulama safhasında mesaisini ve yardımlarını benden esirgemeyen Doç. Dr. M. Turan Söylemez hocama ve çalışmalarımla ilgilenen Prof. Dr. İbrahim Eksin, Prof. Dr. Müjde Güzelkaya ve Doç. Dr. Salman Kurtulan hocalarıma çok teşekkür ederim. Ayrıca teşvikleriyle ve anlayışı ile bu çalışmanın ortaya çıkmasında büyük pay sahibi olan ASP Otomasyon Ltd. Genel Müdürü sayın Yük. Müh. Bülent Vurgun'a yine teşekkürlerimi iletmek isterim.

Bu çalışmayı yaparken bana göstermiş oldukları destek ve anlayıştan dolayı, hayat arkadaşım ve değerli eşim Ayşe Ünlü ve diğer aile bireylerime çok teşekkür ederim. Onların olumlu yaklaşımları ve destekleri olmadan bu çalışmanın sonuçlanması mümkün olamazdı.

Eylül, 2007 Yusuf ÜNLÜ

İÇİNDEKİLER

KISALTMALAR

OLE	:	Object Linking and Embedding
OPC	:	OLE for Process Control
COM	:	Component Object Model
DCOM	:	Distributed Component Object Model
PLC	:	Programmable Logic Controller
DCS	:	Distributed Control System
HMI	:	Human Machine Interface
SCADA	:	Supervisory Control and Data Acquisition
MES	:	Manufacturing Execution System
ERP	:	Enterprise Resource Planning
UTP	:	Unshielded Twisted Pair
TCP/IP	:	Transmission Control Protocol / Internet Protocol
RTU	:	Remote Terminal Unit
PID	:	Proportional Integral Derivative
CPU	:	Central Processor Unit
DLL	:	Dynamic Link Library
IDL	:	Interface Definition Language
API	:	Application Programming Interface
CLSID	:	Class Identifier
IID	:	Interface Identifier
CATID	:	Category Identifier
ProgID	:	Program Identifier
AppID	:	Application Identifier

TABLO LİSTESİ

Sayfa No

ŞEKİL LİSTESİ

Sayfa No

SÜREÇ KONTROLUNDA NESNELERİN BAĞLAŞMASI VE İLİŞKİLENDİRİLMESİ (OPC) STANDARDI VE UYGULAMASI

ÖZET

Endüstriyel otomasyon dünyası özellikle 1990'lı yıllardan itibaren hızlı bir gelişim sürecine girmiştir. Bilgisayarların makine ile insan arasında arabirim olarak kullanılmaya başlanmasıyla birlikte artık donanım bazında gerçekelenemeyen veya gerçeklenmesi oldukça zahmetli olan bir takım konularda bilgisayar yazılımları devreye girmiştir.

Bilgisayarlar, her geçen gün devamlı olarak düşen maliyetleri ve işlemci ve bellek teknolojilerinde meydana gelen ilerlemeler nedeni ile otomasyon dünyasında daha fazla kullanılmaya başlanmışlardır. Microsoft firmasının Windows işletim sisteminin diğer işletim sistemlerine nazaran daha çok tercih edilmesiyle bir çok yazılım üreticisi çeşitli donanımlar için gerçek zamanlı otomasyon yazılımlarını (SCADA, Supervisory Control and Data Acquisition, Raporlama, Veri saklama vb.) bu platformu kullanarak üretmeye başlamışlardır. İşte bu noktada donanım ile yazılımlar arasında standart bir haberleşme ara yüzü ihtiyacı belirmiştir. Zira veri akışının güvenilir ve devamlı olması üretimin kesintiye uğramaması açısından büyük bir önem arz etmektedir.

Donanım ile yazılım arasında istenilen standart arayüzü sağlayan OPC, (OLE for Process Control - Süreç Kontrolunda Nesnelerin Bağlaşması ve İlişkilendirilmesi) donanımda meydana gelen değişimlerin yazılım üreticisini etkilememesini, aynı şekilde yazılım kısmında gerçekleşen yeniliklerin de donanım üreticisi için bir uyum sorunu oluşturmamasını sağlamaktadır. OPC, endüstride donanım üreticileri ile yazılım üreticileri arasına çizilmiş bir çizgi gibidir. Bu sayede her iki taraf da haberleşme sorunu yaşama endişesinden kurtulmuş bir şekilde ürünlerine daha fazla özellik eklemede kendilerini daha özgür hissetmektedirler.

Bu çalışmada, endüstrinin en hızlı kabul edilen ve gelişen standartlarından biri olan OPC tanıtılmış ve OPC kullanımına bir örnek olmak üzere MATLAB programının OPC Toolbox'ı kullanılarak küçük bir uygulama sunulmuştur. Bu uygulamanın amacı, akademik dünyada oldukça yaygın olarak kullanılan benzeşime dayalı MATLAB çözümlerinin çok kolay bir şekilde gerçek dünyaya aktarılıp test edilebilirliğini göstermek ve bu alanda yeni uygulamaların önünü açmaktır.

OBJECT LINKING AND EMBEDDING FOR PROCESS CONTROL (OPC) SPECIFICATION AND ITS APPLICATION

SUMMARY

Industrial automation world has been in a fascinating growth period since 1990's. With the use of computers as HMI (human-machine interface), it became possible to produce solutions which are difficult or impossible to do physically.

Increasing abilities on CPU and memory and decreasing cost of computer hardwares made possible to use them in every possible application in automation world. Most of real time applications (SCADA, Supervisory Control and Data Acquisition, report tools, data storage etc.) started to use Micrsoft's Windows operating system as common application platform. To maintain data consistency between software and hardware in order to prevent production losses, a standard interface requirement is needed since many application programmes are used in important fields of automation systems.

Real time data exchange and sharing requests between controllers and computers realized more standardized ways with OPC (OLE for Process Control – Object Linking and Embedding). Any changes/improvement in hardware components or software applications will have no negative impact on data communication when we use OPC standart interface for both sides. OPC draws a line between hardware manufacturers and software producers letting both sides feel free to add more functions to their products without communication problems.

In this study it is aimed to give information about automation world's fastest growing standard OPC. A small application is provided using MATLAB's OPC Toolbox functions to show its practical usage. MATLAB is selected due to its common usage in academic world and to show how easy to implement its simulation abilities to the real time applications. It is believed that this study will be the starting gate of this kind of applications.

1. OTOMASYON SİSTEMLERİ TARİHÇESİ

İnsanoğlu doğası gereği her zaman önündeki engelleri aşma arzusunda olmuş ve bu yönde de beynini ve becerisini ortaya koymuştur. Tarihteki bir çok buluş bu içgüdünün insanoğlunu sürüklemesi ile meydana gelmiştir. Ortaya çıkan her yeni buluş da kendinden sonra ki buluşlara ilham kaynağı olmuştur.

İnsanın bir sorun karşısında önce beyni devreye girer ve sorunu tanımlamaya çalışır. Beyin sorunun ne olduğunu ve bu soruna yol açan nedenleri anladıktan sonra, çözüm olabilecek yöntemleri tasarlamaya başlar. Tasarlanan yöntemler arasında en uygun olanı uygulamak üzere sinirlere gerekli emirler verilir ve bu sayede ön görülen yöntemin sorunu halledip halletmediğine bakılır. Eğer sorun ortadan kalkmış ise beyin bu yöntemi hafızasına kaydeder ve bir daha benzer bir sorun ile karşılaştığında yine aynı yöntemi uygulamaya çalışır. Eğer sorun çözülmediyse hafızasındaki diğer yöntemleri de sırasıyla deneyerek sorunu çözmeye çalışır.

Önceleri çözümün uygulanmasında insanoğlu kaynak olarak sadece kendisini kullanmış ve kendi el becerisi ile sorunu halletmeye çalışmıştır. Ancak daha sonraları etrafındaki nesneleri tanıdıkça ve onların özelliklerini anladıkça çözümlerinde bu nesnelerden de faydalanmayı öğrenmiştir. Bu sayede el becerisi ile gerçekleştiremediği çözümleri bile yerine getirmeyi başarmıştır.

İnsanoğlunun amacı, etrafındaki canlı ve cansız varlıkları tanımak, sonra da bunların bir çoğunu bir arada kullanarak sorunu çözmek için yöntemler üretmek ve en etkin biçimde sonuca ulaşmak olmuştur. Bunun için aynı probleme ait çeşitli yöntemler geliştirmiş ve bu yöntemlerin uygulanmasında kendi müdahalesi olmadan işlerin yürüyebilmesini sağlamaya çalışmıştır. Artık insanoğlu çözümü oluşturacak elemanları (canlı ve/veya cansız nesneler) anlamlı bir şekilde bir araya getirip –ki buna ileride sistem denilecektir, süreci takip etme ve sadece gerektiğinde sürece müdahil olup sonuca en etkin biçimde ulaşma hedefine geçmiştir.

Bu noktadan itibaren otomasyon kavramı insanoğlunun günlük hayatında önemli bir yer tutmaya başlamıştır. İnsanoğlu, en temel günlük ihtiyaçlarından, en karmaşık

işlerine kadar her alanda yapılması gerekenlerin olabildiğince kendi müdahalesi olmadan yerine getirilmesini sağlayacak yöntemler geliştirmiştir.

Günümüzde de bu durumun etkin olduğunu yaşantımıza bakarak kolaylıkla görebiliriz. En basit işlerden, hatta kimi zaman "bunun içinde mi?" dedirtecek türden çözümlerden, birçoğumuzun nasıl işlediğini bile anlayamadığımız karmaşık sistemlere kadar hemen tüm alanlarda, otomasyon hayatımızın her safhasında kendini göstermektedir.

1.1 Otomasyon Sistemi

Otomasyon, endüstride, yönetimde ve bilimsel işlerde insan aracılığı olmadan işlerin otomatik olarak yapılmasıdır. Sistem ise, bir sonuç elde etmeye yarayan yöntemler düzeni veya belirli bir görevi yerine getirme amacı ile, birlikte iş gören elemanlar topluluğudur.

Bu iki tanımdan yola çıkarak "Otomasyon Sistemi" şöyle tanımlanabilir;

Otomasyon Sistemi, endüstride, yönetimde ve bilimsel işlerde insan aracılığı olmadan işlerin otomatik olarak yapılmasını sağlayan ve birlikte iş gören elemanlar topluluğudur.

Bu teze konu olan kısım gereği bu noktadan itibaren Endüstriyel Otomasyon Sistemlerine ağırlık verilecektir.

1.2 Endüstriyel Otomasyon Sistemleri Tarihçesi

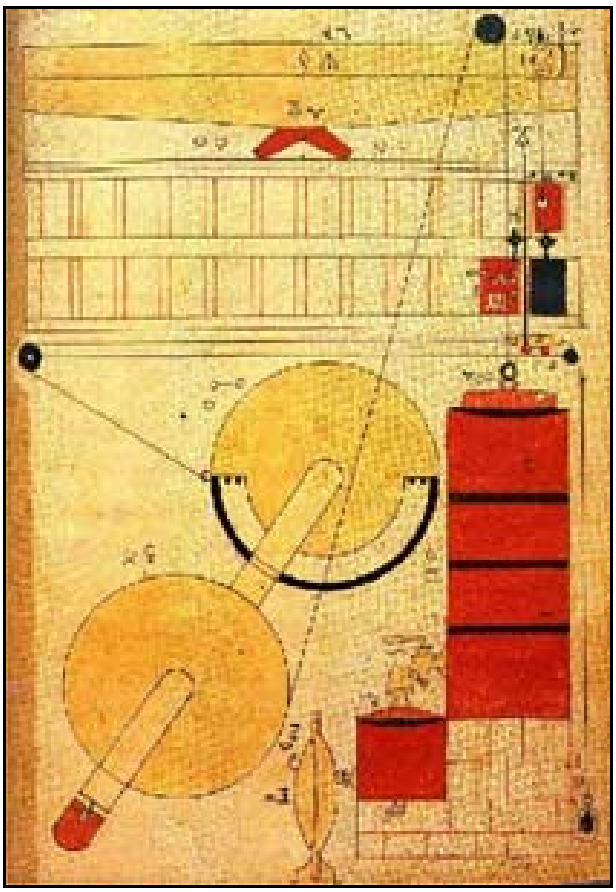
Endüstriyel otomasyon sistemlerinin geçmişi çok eski zamanlara kadar dayanır. İnsanlık bilerek veya farkında olmadan en eski zamanlardan bu yana otomasyon sistemleri kurmuş ve geliştirmiştir. Teknik yönden gerçekleştirildiği bilinen ya da belgelerde bilinçli olarak tasarlanmış ve çalışabilirliği denenmiş olan en eski otomatik kontrol düzenleri Helenistik Dönem'e kadar uzanmaktadır [1].

Otomasyon sistemlerini tarihte geçirdikleri gelişimle birlikte dört ana başlıkta toplamak mümkündür. İlk olarak, yakın geçmişe kadar her devirde yavaş yavaş gelişen Mekanik Otomasyon Sistemleri gösterilebilir. Daha sonra, elektriğin bulunmasıyla birlikte hız kazanan, sanayileşmenin de etkisiyle oldukça yaygın kullanım alanına sahip olan Elektriksel Otomasyon Sistemleri tarihte ikinci bir

dönem olarak sayılabilir. 20. yüzyılın ikinci yarısından itibaren çok hızlı bir gelişme gösteren zayıf akım tekniklerinin uygulandığı Analog Otomasyon Sistemleri ve günümüzde baş döndüren bir hızla gelişen sayısal elektronik sayesinde ortaya çıkan Sayısal Otomasyon Sistemleri, endüstri dünyasında üçüncü ve dördüncü nesil otomasyon sistemleri olarak adlandırılabilirler.

1.2.1 Mekanik Otomasyon Sistemleri

Mekanik otomasyon sistemlerine ait en eski uygulamalar su saatlerindeki debi kontroluna ilişkindir [1]. İskenderiye'de Ktesibios'un (M.Ö. III. Yüzyıl) geliştirmiş olduğu ve şekil 1.1 de gösterilen debi kontrolu düzeneği, modern otomobillerde yakıt akışını ayarlayan karbüratörlere benzemektedir.

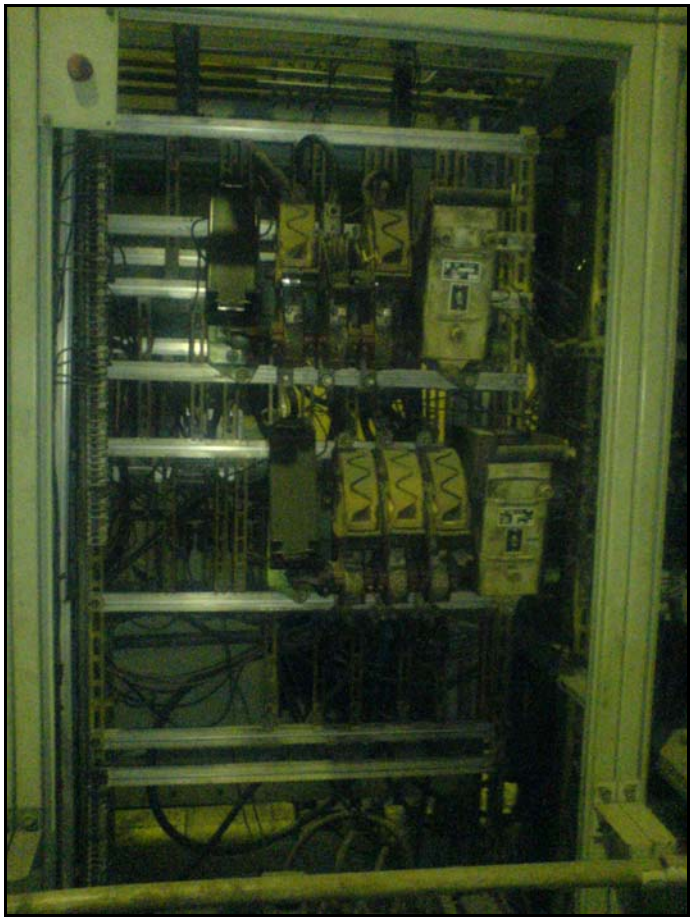


Şekil 1.1 : Cizreli Eb-ül-İz'in "Kıtab al Hiyal" kitabındaki debi kontrol sistemi.

Tarihte her dönemin gelişmiş uygarlıkları bu tip otomasyon sistemlerine önemli katkılar yapmış ve nihayet 19. yüzyıl'ın sonlarına doğru günümüz otomasyon sistemlerinde çok önemli bir kavram olan geri besleme kavramının keşfedilmesiyle birlikte otomasyon dünyası hızlı bir gelişim sürecine girmiştir [1].

1.2.2 Elektriksel Otomasyon Sistemleri

Özellikle ikili mantığın kabul görmesi ile birlikte 1920'li yıllarda elektriksel röle ve kontaktörlere dayanan otomasyon sistemleri tasarlanmaya başlamıştır. Bu şekilde tasarlanan sistemlerle makina otomasyonları oldukça geniş bir kullanım alanına sahip olmuştur.



Şekil 1.2 : İsdemir Demir Çelik fabrikasındaki röle otomasyon sistemi

Elektriksel otomasyon sistemleri, dayanıklılıkları ve iş görürlülükleri sayesinde uzunca bir süre endüstriyel çözümlerde kullanılmışlardır. Bunların günümüze kadar uzanan çalışır örneklerini bulmak hiç şaşırtıcı değildir. Bu tarz sistemlerin aktif olarak çalışan bir örneği İsdemir Demir Çelik fabrikası tel çubuk haddehanesi bölümünde bulunmaktadır ve bu sistem Şekil 1.2'de gösterilmiştir.

1.2.3 Analog Otomasyon Sistemleri

20. yüzyıl'ın ortalarında röle tabanlı otomasyon sistemleri ile gerçeklenmesi mümkün olmayan bazı kontrol çözümleri yaygın olarak kullanılmaya başlanmıştır. Direnç, indüktans ve kondansatör gibi elektronik komponentlerin kullanımı yanında

çeşitli işaret işleme tekniklerinin gelişmesi ve günümüzde dahi halen kontrol problemlerine etkin çözümler üreten PID algoritmalarının kullanılması, analog otomasyon sistemlerine dayanan çözümlerin gelişmesine yol açmıştır. Şekil 1.3'te bu çözümlerden bazı analog otomasyon sistemi birimleri görülmektedir.

Şekil 1.3 : OlmukSA Edirne kağıt fabrikasındaki analog otomasyon sistemleri

Analog otomasyon sistemlerinde ayrıca elektriksel olmayan yöntemlerle sağlanan çözümler de vardır. Bu tür çözümlere örnek olarak, basınçlı hava ile çalışan PID denetleyicilerini göstermek mümkündür.

Analog otomasyon sistemleri, röle tabanlı otomasyon sistemleri gibi işlevselliklerini günümüze kadar koruyarak sağlamlıklarını ispat etmişlerdir.

1.2.4 Sayısal Otomasyon Sistemleri

1970'li yılların başında yarı iletken teknolojisinin gelişmesi ile birlikte, bilgisayar ve yazılım sektörü, endüstrinin her alanında, yaygın olarak kullanılmaya başlanmıştır. Artık analog otomasyon sistemlerinden sayısal otomasyon sistemlerine geçiş sürecine girilmiştir.

İlk PLC (Programmable Logic Controller - Programlanabilir Lojik Denetleyici) çözümleri yine bu yıllarda uygulanmıştır. Başlarda sadece ikili (binary) süreçleri (bir anahtar konumuna göre motor çalıştırma-durdurma, vana açma-kapama v.b.) kontrol etmekte kullanılan PLC'ler, analog-sayısal ve sayısal-analog dönüştürücülerin kullanımı sayesinde, hem analog (belli bir seviye değerine bağlı olarak oransal bir vanaya açıklık verilmesi gibi) hem de ikili kontrol problemlerinde kullanılmışlardır.

PLC'lerin hızlı gelişimi 80'li yıllarda DCS'lerin (Distributed Control System - Dağıtılmış Kontrol Sistemi) ortaya çıkmasını sağlamıştır. DCS felsefesi, değişik bölgelerde yerleşmiş bulunan kontrol bölgelerinin tek bir merkezden yönetilmesinin sağlanması ilkesine dayanmaktadır. Bu sayede denetleyici bir merkezde olurken buna bağlı bir I/O (input/output – giriş/çıkış) istasyonu fabrikanın başka ve uzak bir bölgesinde olabilmektedir. I/O istasyonları sürecin belli bir bölgesindeki analog veya sayısal işaretlerin bir araya gelmesiyle oluşturulur. Bu sayede, çok büyük bir fabrika bu istasyonlar vasıtası ile dağıtılmış kontrol bölgelerine ayrılmış olur. Merkez ile I/O istasyonları arası haberleşme ise her iki tarafta bulunan modemler aracılığı ile sayısal olarak yapılır.

Şekil 1.4 : Sayısal otomasyon sistemlerinde kullanılan PLC ve bilgisayarlar

Şekil 1.4'te çeşitli PLC ve DCS sistem örnekleri görülmektedir. Önceleri raf tipi (rack) olan PLC / DCS ürünleri günümüzde artık ray tipi modüler bir yapıda

üretilmektedir. Ayrıca bilgisayarların bir PLC sistemi gibi kullanılmasını sağlayan SoftPLC uygulamalarının da günümüzde giderek artan bir tercih olduğunu vurgulamak gerekir.

Bilgisayarlar sayısal otomasyon sistemlerinin önemli bir birimidir. PLC programlanmasından operator ekranlarının tasarımına kadar her adımın yazılımı bilgisayarlar üzerinde gerçekleştirilir.

Analog otomasyon sistemlerinin geniş uygulama alanları ile elektriksel otomasyon sistemlerinin rahat ve kolay anlaşılabilirliklerinin bir arada kullanılabildiği sayısal otomasyon sistemleri, günümüzde tüm otomasyon sistemlerinin alt yapısını oluşturmaktadır. Bu teze konu olan kısım gereği bundan sonraki bölümlerde sadece sayısal otomasyon sistemlerinde kullanılan haberleşme teknikleri ele alınacaktır.

2. SAYISAL OTOMASYON SİSTEMLERİNDE HABERLEŞME

Sayısal otomasyon sistemleri genel olarak kontrol programının yürütüldüğü bir işlemci ünitesi ve hafızası, giriş/çıkış kartları, görüntüleme birimleri ve yardımcı birimlerden oluşmaktadır. Bu birimler arası tüm haberleşmeler sayısal olarak yapılmaktadır ve hepsine ait ayrı fiziksel alt yapılar ve bu fiziksel altyapılar üzerinde koşan protokoller bulunmaktadır.

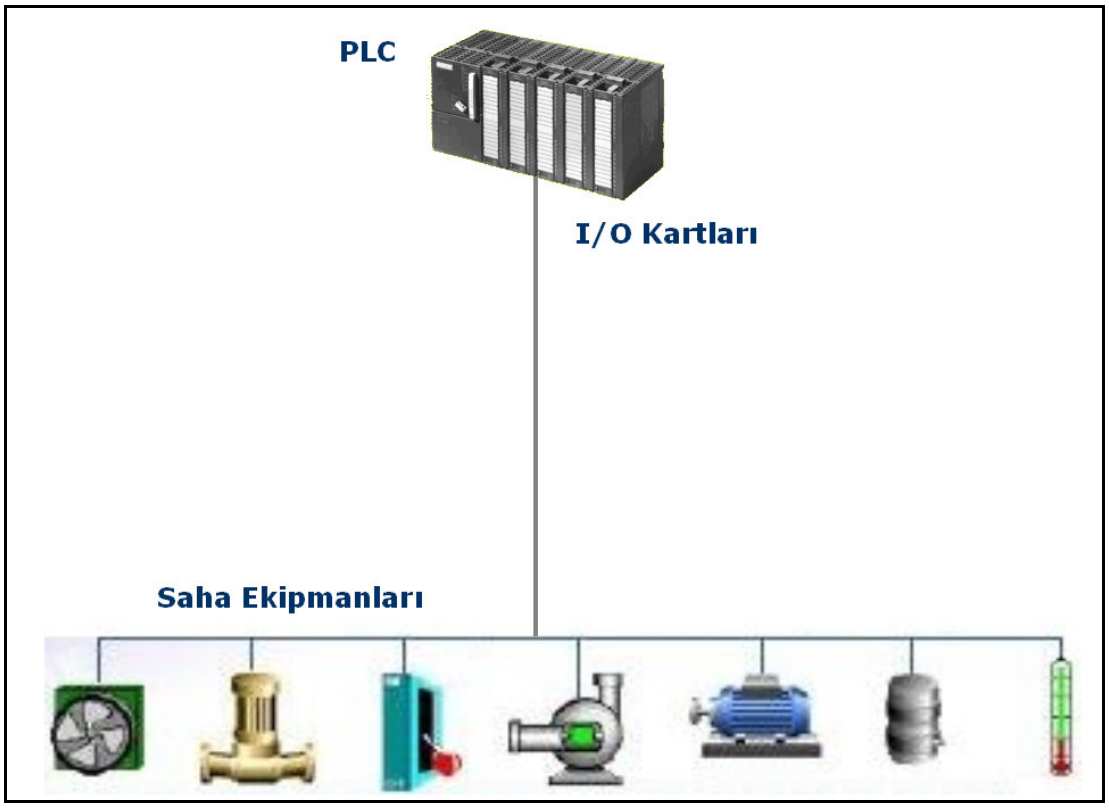
Sayısal otomasyon sistemlerinde haberleşmeyi dört başlık altında toplamak mümkündür. Bunlar;

- Saha ekipmanları ile olan haberleşme
- PLC'ler arası olan haberleşme
- PLC ile insan-makina arayüzleri HMI (Human Machine Interface) birimleri arası haberleşme
- PLC ile diğer uygulamalar arası olan haberleşmelerden oluşmaktadır.

2.1 Saha Ekipmanlarıyla Haberleşme

Saha ekipmanları ile olan haberleşmelerden kasıt, otomasyon sistemi tarafından kontrol edilecek olan fiziksel ekipmanlarla olan haberleşmedir. Şekil 2.1'de temsili bir resmi çizilen, günümüz otomasyon sistemleri tarafından kontrol edilen belli başlı ekipmanlar (motorlar, vanalar, klepeler, damperler ve benzeri cihazlar) görülmektedir. Ayrıca sistem tarafından okunan basınç dönüştürücüleri ve anahtarları, seviye dönüştürücüleri ve anahtarları, akım ve güç ölçümleri gibi çeşitli analog ve sayısal bilgi sağlayan enstrümanlarla da haberleşmeler vardır.

Ekipman ve enstrümanlardan gelen işaretler, türlerine göre, analog veya sayısal giriş kartlarına bağlanırken, kontrol edilecek ekipmanlara giden sayısal veya analog işaretler de PLC'nin ilgili analog veya sayısal çıkış (I/O) kartlarından sahaya gönderilir.



Şekil 2.1 : PLC ile saha ekipmanları arası haberleşme

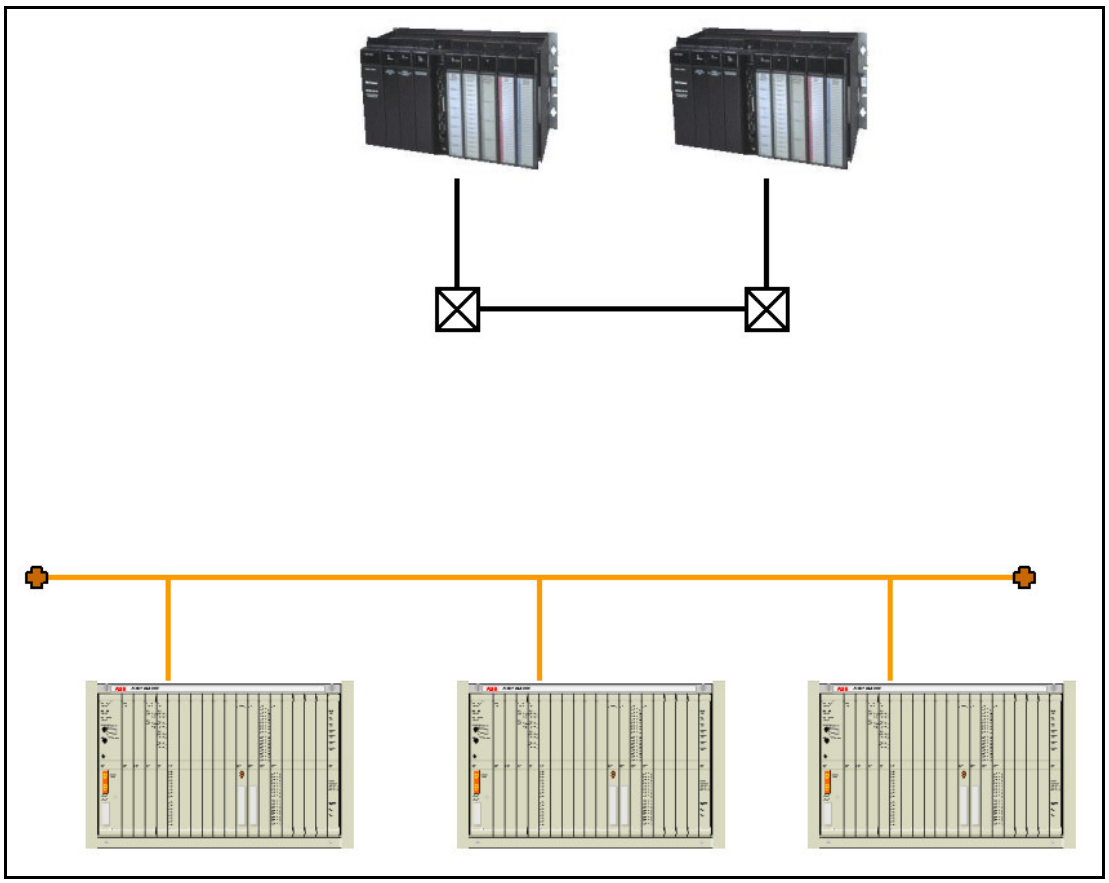
Giriş/Çıkış kartları genelllikle özel bir haberleşme hattı (bus) üzerinden PLC'ye bağlanır. Ayrıca günümüzde sahadan gelen işaretlerin, Profibus, Modbus gibi endüstri haberleşme standartları kullanılarak, PLC'ye aktarımı mümkündür. Bu durumda, klasik giriş/çıkış kartlarının yerine, PLC ve ekipman taraflarında ilgili standardın haberleşme modülleri bulunmakta ve veri alış verişleri yazılım tarafından belirlenmektedir.

2.2 PLC'ler Arası Haberleşme

Sayısal otomayon sistemleri kullanımının yaygınlaşması ile birlikte, PLC'lerden talep edilen iş yükü tek bir işlemci tarafından yerine getirilemeyecek kadar ağırlaşmıştır. Ayrıca, çoğu durumda, sürecin anlam bütünlüğü ve gereklilikleri, birden fazla PLC kullanımını zorunlu kılmaktadır. Bu nedenlerden dolayı, bu işlemciler (PLC'ler) arası haberleşmeyi sağlamak diğer bir ihtiyaç olarak ortaya çıkmıştır.

PLC'ler arası haberleşme genel olarak üretici firma tarafından geliştirilen özel bir kontrol ağı ile sağlanmaktadır. Her üretici kendi geliştirdiği özel protokoller ve bu protokolleri anlayabilecek özel donanımlar geliştirerek kendi ürünleri arasında veri

alış verişini mümkün hale getirmiştir. Ancak günümüzdeki eğilim ve endüstriyel kullanıcıların beklentisi, özel protokoller ve donanımlardan ziyade standart yazılım ve donanımların kullanılması yönündedir. Bu bağlamda, kontrol ağlarında tıpkı bilişim ağlarında olduğu gibi, Ethernet alt yapısında UTP (Unshielded Twisted Pair) kablolar üzerinde TCP/IP (Transmission Control Protocol / Internet Protocol) protokolünü görmek hiç şaşırtıcı değildir.



Şekil 2.2 : PLC'lerin kendi aralarındaki haberleşmeleri

Ancak her ne kadar alt yapı ve üzerindeki haberleşme protokolü değişsede PLC'ler arası veri alış verişi genelde şu üç mantık üzerinde yapılmaktadır.

- Özel yayın (peer-to-peer)
- Çoklu yayın (multicast)
- Genel yayın (broadcast)

Şekil 2.2'de, yukarıdaki üç haberleşme türü için, veri alış verişinde bulunacak olan birimler arasındaki yapı gösterilmektedir. Özel yayın türünde belli iki birim veri alış verişinde bulunurken, çoklu yayın türünde haberleşmede ikiden fazla belli birimler

haberleşir. Bu birimlerden biri veri gönderimi yaparken, kontrol ağındaki diğer belli birimler bu veriyi alırlar. Bu sayede, aynı veriye birden fazla birimde ihtiyaç olması durumunda çoklu yayın yöntemi, özel yayın yöntemine göre haberleşme verimliliği, kaynak birimin ulaşılabilirliğinin (availability) artması gibi önemli avantajlar sağlamaktadır. Son olarak, gönderilen verinin belli bir alıcısı olmadan ağdaki tüm birimlerin kullanımına sunulan haberleşme yöntemi olan, genel yayın yöntemi, çoklu yayın yönteminde veri alımı yapacak olan birimlerin artması durumunda kullanılan bir yöntemdir.

2.3 PLC ile HMI Arası Haberleşme

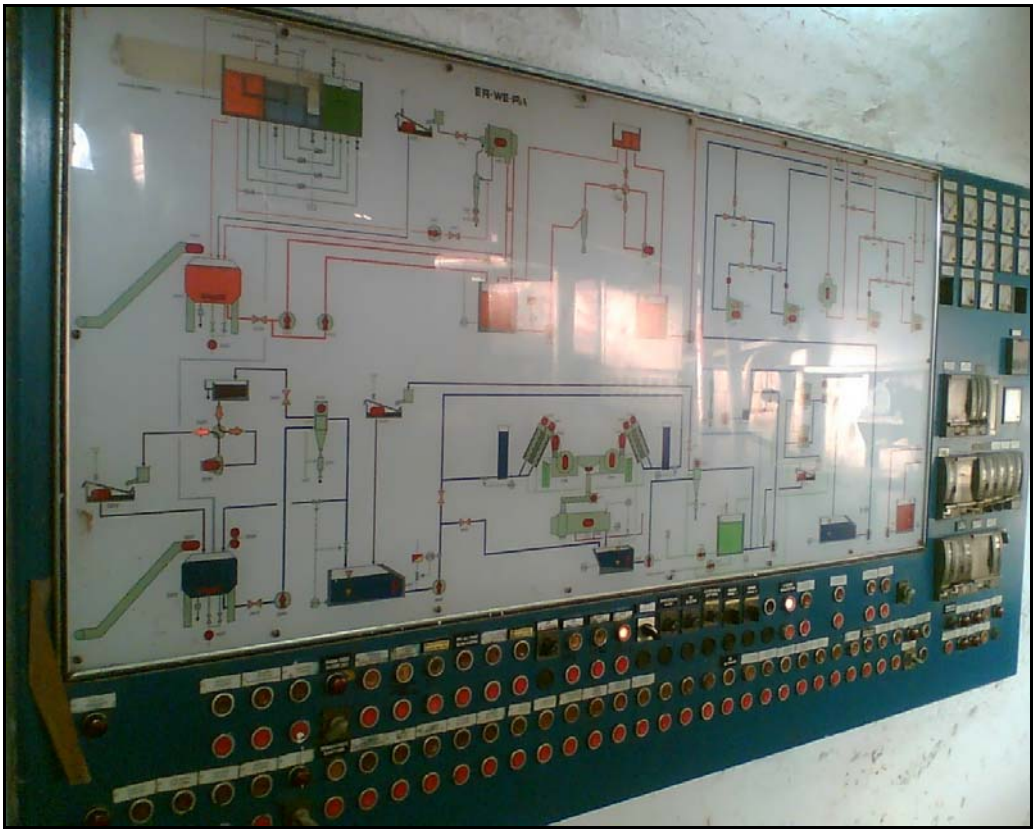
Sayısal kontrol sistemlerinin en önemli avantajlarından biri de güçlü görsel yanlarının olmasıdır. Bu tip sistemlerin kullanılmaya başlandığı ilk zamanlardan başlamak üzere her aşamada, PLC ile sistemi idare edecek olan operatör arasında etkin bir köprü kurma yükümlülüğü, HMI tasarımcılarına düşmüştür. Çoğu zaman anlaşılması çok kolay olmayan uzun PLC programları bir çok insan için karmaşık gelmiştir. Oysa HMI birimleri sayesinde –ki bunlar gelişim süreçleri ile birlikte ilerleyen paragraflarda verilecektir, programcı tarafından yazılan sayfalar dolusu lojik yerine gerekli yerlerin anlamlı bir sembol ve/veya uyarıcı bir yazı ile birlikte gösterilmesi mümkün olmuştur.

Böylece insan ile makine arasında bir arayüz oluşturulmuş ve operatör tarafından verilecek olan komutlar makine diline çevrilmiş (örneğin, bir motora çalış komutu), makine dilinde olan bir durum bilgisi anlamlı bir ifade olarak (örneğin, 4-20 mA aralığında olan bir basınç bilgisi bar biriminden basınç ifadesi olarak) operatöre sunulmuş olmaktadır.

İlk kullanılan HMI ürünleri geniş bir alan üzerine monte edilen ve ekipmanların durumlarını çeşitli ışık/lamba durumları ile açıklayan panolar olmuştur. Bu panolar üzerindeki lambalar ile durum bilgisi sağlanırken, tuşlar ile de komutlar makinaya gönderilmekteydi.

Şekil 2.3'te, OlmukSA Edirne kağıt fabrikasında halen bulunan ve geniş bir alanda çok sayıda tuşun ve lambaların olduğu bu tip bir ürün görülmektedir. Bu tip çözümler, ilk zamanlar ihtiyaçları karşılasa da, daha sonraları kontrol alanları genişledikçe kullanılabilirliklerini yitirmişlerdir. Zira, HMI ürününe yapılan her yeni

eklenti için ürüne fiziksel ilaveler ve değişiklikler yapmak gerekiyor ve her ilave sürecin izlenmesini zorlaştırıyordu.

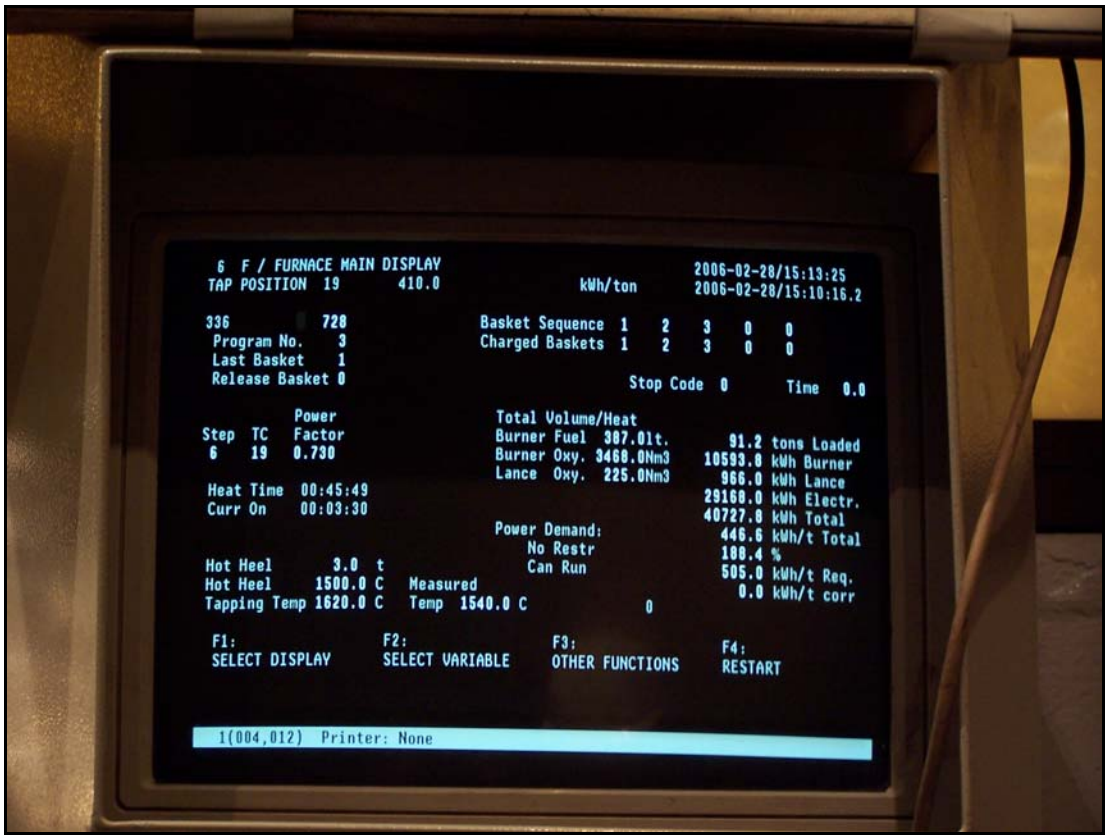


Şekil 2.3 : OlmukSA Edirne kağıt fabrikası HMI ürünü

Bundan bir adım sonra, yazı temelli ekranlar HMI ürünleri olarak ortaya çıkmaya başladı. Böylece, panolarda yeni bir ilave veya değişiklik yapmanın zorluğu aşılmış olsa da görsellik ve dolayısıyla sürecin anlaşılabilirliğinden ödün verilmiş oluyordu.

Sadece bir göstergeden ibaret olan bu tip ekranların bir örneği şekil 2.4'te gösterilmiştir. Yani, bu tip ürünler bildigimiz anlamdaki, işlemcisi, hafızası ve sabit diski olan bir bilgisayar değillerdir. Ekranlardaki her bir karakter, her bir bilgi (sabit veya canlı bilgi olsun) kontrol biriminden (PLC veya DCS) ekrana gönderilmektedir.

Bilgisayarların HMI ürünü olarak kullanılması, 1980'li yılların sonu ile 1990'lı yılların başında IBM ve DOS işletim sistemlerinin piyasaya çıkması ve bu işletim sistemleri üzerinde uygulamaların geliştirilmesiyle başlamıştır. Kontrol sistemi üreticileri, artık ar-ge çalışmalarında kaynaklarının önemli bir bölümünü yazılıma ayırmışlar ve kendi sistemleri ile haberleşecek SCADA (Supervisory Control and Data Acquisition) programları geliştirme safhasına girmişlerdir.



Şekil 2.4 : İzmir Demir Çelik fabrikası ark ocağı kısmı HMI ürünü

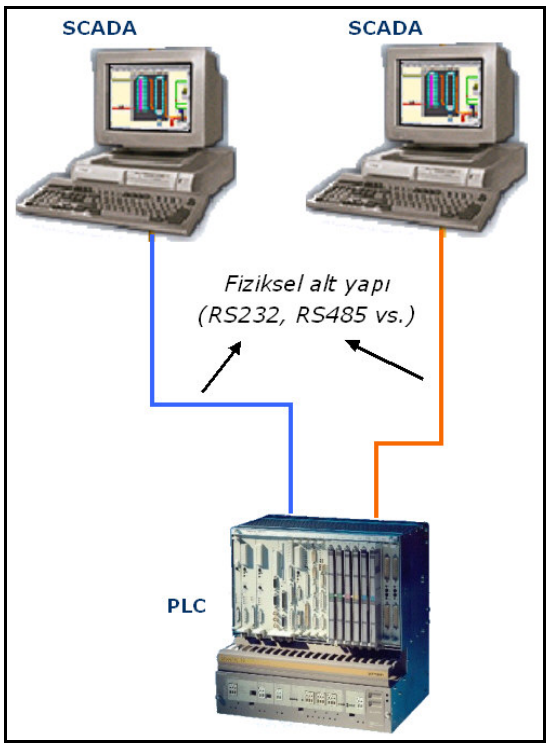
SCADA sistemleri, bilgisayarların HMI ürünü olarak kullanılmasıyla, endüstri dünyasına giren bir kavramdır. Bir SCADA programı, sadece saha ekipmanlarının durumunu gösterme ve operatörden verilen emirlerin PLC'ye aktarılmasında değil aynı zamanda da geçmişe dönük veri saklama, bu verilere ait zamana göre grafik çizimi, veri üzerinde çeşitli analiz imkanları v.b. yardımcı işleri de yerine getirebilmektedir.

Bu yüzden, günümüzde, makina ile insan arasındaki arayüz eğer bir bilgisayar programı tarafından sağlanıyorsa, bu birimlere SCADA ve özel panel ekranlar tarafından sağlanıyorsa, bunlara da HMI ürünü denilmektedir. Çalışmanın bundan sonra ki kısmında, SCADA kelimesi gerçek zamanlı bilgisayar programı için, HMI kelimesi ise panel ekranlar için kullanılacaktır.

SCADA programları, 1990'lı yıllarda, PLC / DCS üreticileri tarafından sağlanmakta idi. Bunun nedeni, kontrol sistemi üreticilerinin PLC ile olan haberleşmeyi genellikle kendilerinin geliştirmeleri ve dışa kapalı bir protokol üzerinden gerçekleştirmeleriydi. SCADA programının bulunduğu bilgisayar, çoğu zaman, bilgisayara takılan özel bir haberleşme kartı vasıtası ile bu protokolü kullanarak

kontrol ağına dahil olur ve haberleşme kartının sürücüsü (driver) sayesinde de canlı bilgilere ulaşırdı.

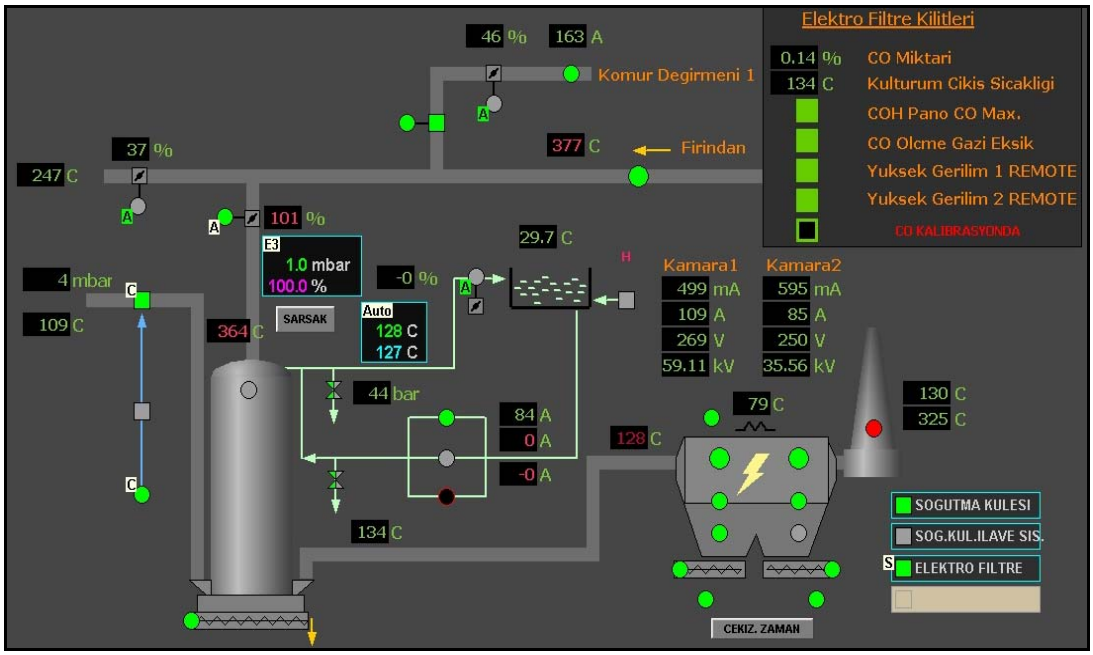
Kullanılan fiziksel altyapılar genellikle RS232, RS485, RS422 veya Ethernet altyapıları olmuştur. (Şekil 2.5) Bunların üzerinde koşan protokoller ise Allen-Bradley için DH1 ve DH+, Siemens için 3964R ve SinecH1, ABB için COMLI ve MasterBus vs. gibi PLC / DCS üreticilerinin geliştirdikleri, kendilerine has protokollerdi [2].



Şekil 2.5 : PLC ile SCADA ürünleri arası haberleşme

1990'lı yılların ikinci çeyreğinden itibaren, markalar arası standart bir haberleşme protokolü olan, ProfiBus ve ModBus gibi protokoller piyasaya çıkmıştır. Bugün kontrol sistemlerinin birbirleri ile olan haberleşmesinde yaygın olarak kullanılan bu protokoller sayesinde, farklı üreticiler tarafından üretilmiş kontrol sistemleri birbirleriyle veri haberleşmesinde bulunmaktadırlar. Böylece, bugün piyasadaki bir çok motor sürücüsü klasik yöntemle, yani giriş / çıkış kartları ile PLC veya DCS sistemine onlarca kablo üzerinden sayısal ve analog işaretler gönderip almak yerine yalnızca bir çift kablo ile ProfiBus üzerinden haberleşmektedirler. Yine, enerji sektöründe kullanılan birçok veri analizörünün ModBus protokolü kullanarak merkezi kontrol sistemleri ile haberleşmelerini buna örnek olarak gösterebiliriz.

İşletim sistemindeki ilerlemelere paralel olarak, SCADA programlarının da kapasiteleri artmıştır. Öyle ki, SCADA programları, kritik karar aşamalarında, karar verici modül olarak veya güçlü analiz özelliklerine sahip olmaları nedeni ile, detaylı bir raporlandırma programı gibi kullanılabilmektedirler. Bu yüzden de, endüstri dünyasının çok sık kullandığı, bir fabrika yönetim merkezi haline gelmişlerdir. Donanım üreticileri, yazılımlarını geliştirmek için, önceleri, IBM'in OS/2 işletim sistemini, daha sonra, Unix işletim sistemlerini ve günümüzde aktif olarak kullanılan, Microsoft'un NT tabanlı Windows işletim sistemlerini kendilerine taban olarak seçmişlerdir. Şekil 2.6'ta, NT tabanlı bir SCADA programı görüntüsü bulunmaktadır.



Şekil 2.6 : Windows NT tabanlı bir SCADA görüntüsü

2.4 PLC ile Diğer Uygulamalar Arası Haberleşme

Windows işletim sistemi üzerinde, gerek donanım üreticileri tarafından gerekse yazılım firmaları tarafından, birçok kontrol yazılımı geliştirilmiştir. Farklı amaçlar için kullanılan bu yazılımların sayısı arttıkça, endüstriyel kullanıcılar, farklı firmalarca sunulan çözümlerin birbiri ile veri alış verişinde bulunmasını veya bir yazılımda sahip oldukları verinin veya özelliğin diğer bir yazılım tarafından da kullanılabilmesi isteklerini çoğaltmaya başlamışlardır. Mevcut durumda, her bir yazılım haberleşeceği donanımla ilgili bir sürücü dosyasına sahip olmalıdır. Sürücü, fiziksel donanım ile haberleşen kısım olduğundan, bu yazılımların sayısı arttıkça ya

haberleşilecek donanıma kart ilavesi yapılmalı ya da bu yazılımların donanıma aynı anda erişmeleri kısıtlanmalıdır. Bu durum, endüstriyel yazılım firmalarının ürünleri arasında, standart bir haberleşme yönteminin geliştirilmesi zorunluluğunu ortaya çıkarmıştır.

3. OPC (OLE for PROCESS CONTROL)

Bir süreç denetiminde haberleşme ağı Şekil 3.1'de gösterildiği gibi 3 katmana ayırılabilir. Bunlar;

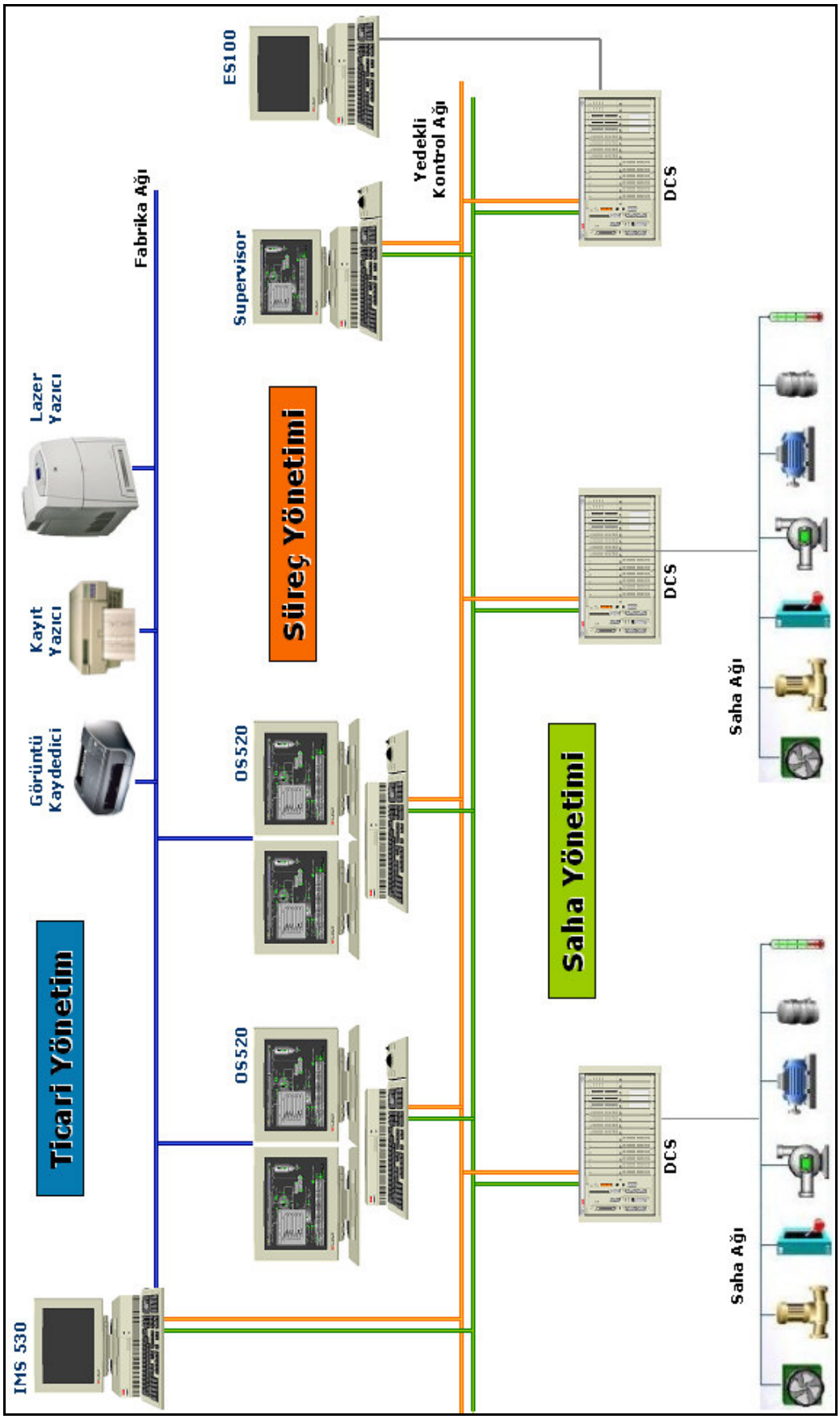
- Saha yönetimi
- Süreç yönetimi
- Ticari yönetim

katmanlarından oluşmaktadır [3].

Bu mimaride herbir katman, gelişen teknolojik imkanlar sayesinde, sürecin kontrol ve gözlemlenmesinde çeşitli katkılar sağlamaktadır. Günümüz kontrol ve otomasyon sistemleri, artık, insan ile makine arasında etkileşimi sağlamakla kalmayıp aynı zamanda üretimin karar verme noktalarında da yoğun biçimde kullanılmaktadırlar. Bu da üretimdeki verimliliğin artması yani, daha az kaynakla daha çok üretimin gerçekleşmesini sağlamaktadır. Bu durum, işletmelerin maliyetlerinde önemli ölçülerde azalma sağlamaktadır. Bu sayede işletmeler, yeni pazar kaynakları aramak amacıyla, satış bölümlerine veya Ar-Ge (Araştırma ve Geliştirme) faaliyetlerine daha fazla kaynak aktarma olanağı kazanmaktadırlar.

Olayın teknik yönüne bakılırsa, haberleşme alt yapısının sağlam, güvenilir ve kesintisiz olmasındaki önem ortaya çıkacaktır. Zira, katmanlar arası bilgi alış verişinin kesilmesi veya güvenilir olmaması, doğrudan, üretim verimliliğini etkilemektedir.

Saha Yönetimi: Akıllı cihazların gelişmesi ile birlikte, artık sahadan sadece cihazın gösterdiği değere değil, aynı zamanda, ayar değerlerinden, malzeme bilgilerine kadar birçok bilgiye erişmek mümkün olmuştur. Bütün bu bilgilerin operatöre ve bunu kullanabilecek uygulamalara düzgün ve sürekli bir biçimde aktarılması gerekliliği saha yönetiminde çeşitli özel ağların (ProfiBus, ModBus, CANopen vb.) ortaya çıkmasını sağlamıştır. Bu özel ağlar sayesinde saha yönetimi oldukça gelişmiş ve süreç denetleyicisine daha fazla bilgi gönderilmiştir.



Şekil 3.1 : Süreç denetimi klasik ağ mimarisi

Süreç Yönetimi: Sahadan alınan bilgiler kullanılarak, DCS ve PLC gibi kontrol cihazlarıyla, sürecin otomatik bir şekilde kontrolü sağlanmaktadır. Bu katmanda, operator tarafından sisteme yapılan ayar ve düzeltmeler, SCADA ve diğer yardımcı programlar sayesinde, DCS ve PLC'ler aracılığıyla, sahadaki cihaz ve ekipmanlara gönderilmekte ve sürecin istenen biçimde davranması sağlanmaktadır.

Ticari Yönetim: Süreçten alınan bilgilerin ticari uygulamaların kullanımına sunulması, üretim sisteminin finansal konularla bütünleşmesini sağlar. Ticari yönetim programları ERP ve MES sistemleri sayesinde, değişen ticari koşullara göre, üretim planlanmakta ve önemli ölçüde avantajlar sağlanmaktadır.

Bütün bu anlatılanları verimli bir şekilde yapabilmek için, veri elde etmeye uygun, açık bir haberleşme yapısına sahip olmak çok önemlidir.

3.1 OPC'den Önce

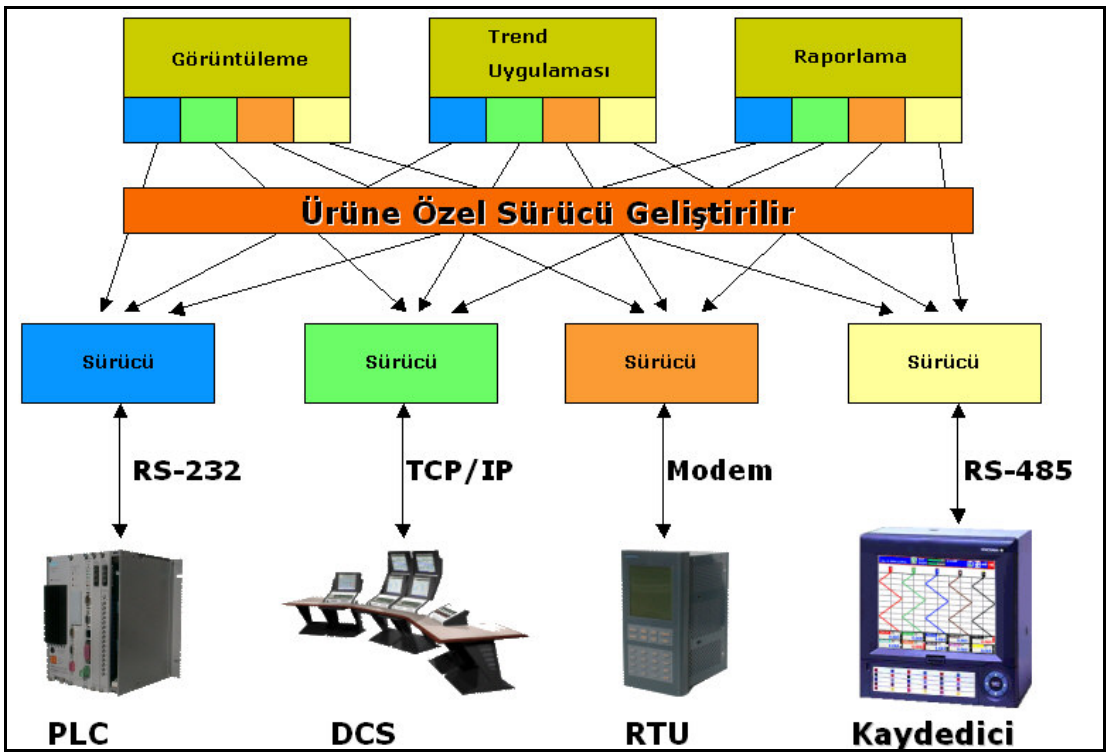
Otomasyon sistemleri yazılımları arasında OPC kullanımından önceki genel yöntem, sürücülerle haberleşme üzerine dayanmaktaydı [4]. Bunun için örneğin Şekil 3.2'de gösterilen görüntüleme programının bir PLC'den veri okuyup yazabilmesi için o PLC'nin Windows için yazılmış olan sürücüsü ile haberleşmek zorundadır. Bu yüzden görüntüleme programında ilave bir arayüz hazırlanır ve bu arayüzün tek amacı vardır. O da, belirtilen sürücü ile bağlantıyı sağlayarak, oradan aldığı bilgileri kendi uygulamasına aktarmak ve/veya kendi uygulamasındaki verileri PLC'ye gönderilmesi için sürücüye iletmektir. Sürücü dosyası da, yine büyük bir çoğunlukla, uygulama yazılımını yapan firma tarafından geliştirilir.

Aynı programın başka bir cihazla haberleşebilmesi için, örneğin, bu kez DCS sistemine ait verileri ekranda görüntüleyebilmesi için, o DCS sistemine ait sürücü dosyası ile haberleşmesi gerekmektedir. Yani, DCS sistemi içinde görüntüleme programında başka bir arayüz oluşturulmalıdır. Benzer şekilde, diğer kontrol cihazlarının bilgilerini kendi ekranlarına taşıyabilmesi için de bu adımlar gerçekleştirilmelidir.

Kontrol ve otomasyon dünyasında çok sayıda değişik marka ve ürünlerin olduğu göz önüne alındığında, programların ya belli marka ve ürünlere hizmet verecek şekilde hazırlanmış olması gerekmekte ya da piyasada olan tüm cihazlar için özel sürücü arayüzlerinin hazırlanması gerekmektedir. Endüstriyel kullanıcılar açısındansa bu

durum, ya donanım üreticisi tarafından sunulan yazılımı kabul etmek ya da elindeki yazılımın desteklediği donanımı seçmek yönünde bir kısıtlama demektir.

Ancak durum, mevcut kontrol sistemine yeni yazılım bileşenlerinin eklenmesi ile, daha da karmaşık bir hale gelmektedir. Zira, örneğin, bir raporlama programının aynı donanımlardan verileri okuması için tıpkı görüntüleme programında olduğu gibi kendi uygulaması için sürücü arayüzleri oluşturulması gerekmektedir. Bu da ciddi anlamda zaman ve kaynağın sadece bu iş için ayrılması anlamına gelir.



Şekil 3.2 : Her yazılım kendi sürücüsü ile haberleşir

Otomasyon dünyasında çok sayıda artan yazılım uygulamalarının avantajlarından faydalanmak ve kontrol edilen sistemler hakkında daha fazla bilgi sahibi olabilmek istenir.Bunun için uygulanan bu tip çözümlerde, yazılım yönünden bakıldığında, aslında aynı bilgiyi değişik şekillerde kullanan farklı uygulamaların her biri için ayrı ayrı sürücü arayüzü geliştirilme zorunluluğu ciddi bir engel olarak karşımıza çıkmaktadır. Öte yandan donanım üreticileri tarafından baktığımız zaman, ürünlerinin sadece bazı yazılımlarla uyumlu olması, olumsuz bir yön olarak karşımıza çıkmaktadır.

Aslında bu durum, Windows'un ilk işletim sistemlerinde ki durumla aynıdır. O zamanlar, örneğin, bilgisayara bağlı olan bir yazıcı, ürün bazında sürücülere sahipti.

Yani satın alınan yazıcının Word programı için sürücü dosyası var iken, bilgisayarda yüklü başka bir uygulama programı için sürücüsü bulunmamaktaydı. Bu durumda, bilgisayardan bir çıktı alınması gerektiğinde mutlaka Word programı kullanılmak zorundaydı. Bu yüzden, yazıcı satın alınmadan önce kullanmak istenilen uygulamalara uygun sürücüsünün olup olmadığı bilinmeli idi [5].

Ürüne özel sürücü geliştirmeye dayanan bu tür çözümler şu sorunları ortaya çıkarmıştır:

- **Aynı çalışmanın defalarca tekrarı:** Her yazılım geliştirici kendi yazılımı için aynı cihazla haberleşmek için bir sürücü yazmak zorundadır.
- **Sürücüler arası uyumsuzluk:** Son kullanıcı, bir programda sahip olduğu bir özelliği, aynı donanımla haberleşen diğer bir programda bulamayabilir. Zira, sürücü geliştirenler, kendi programları ile direk olarak ilgili olmayan bazı donanım özelliklerini desteklemeyebilirler.
- **Gelecekte donanımda olacak değişikliklere destek:** Geliştirilen sürücüler donanım firmalarından bağımsız olarak üretildiklerinden, aynı markanın yeni nesil bir ürünü önceki donanım için yazılmış sürücü ile haberleşmeyebilir.
- **Erişim problemi:** Farklı iki uygulama için farklı sürücüler kullanmak gerekeceğinden dolayı çoğu zaman aynı anda aynı cihaza erişilemeyebilir.

Kontrol ve otomasyon sistemleri için üretilen programların hızla artması bu problemlerin çözümüne yönelik olan ortak isteği daha da güçlendirmiştir. Artık tüm endüstrinin beklentisi, donanım ve yazılım üreticilerinin arasında güvenli ve standart bir arayüz sağlanması olmuştur. Eğer böyle bir arayüz sağlanmış olursa herhangi bir donanımla herhangi bir yazılımın haberleşmesinde çok fazla bir engel kalmamış olacaktır.

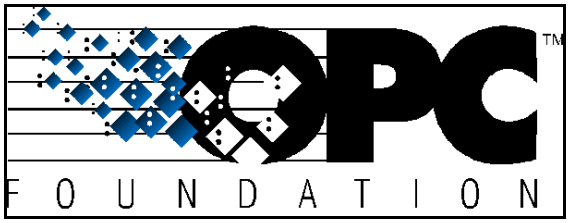
3.2 OPC Derneği (OPC Foundation)

Endüstriden gelen bu istekler doğrultusunda, donanım ve yazılım firmalarından beşi (Intellution, Opto-22, Fisher-Rosemount, Rockwell Software ve Intuitiv Software) kontrol sistemleri yazılımları arasında standart bir veri alış veriş yöntemi kurmak üzere Microsoft firması ile işbirliği içinde OPC Görev Birliği'ni (OPC Task Force) 1995 yılının Mayıs ayında kurdular. OPC komitesinin amacı, Windows tabanında

oluşturulan kontrol sistemleri cihaz sürücülerinin gelişimini kolaylaştırmak ve performansını arttırarak standart bir uygulama arayüzü ile diğer programlar arası veri alış verişini sağlamaktır [6].

Fabrika seviyesindeki cihazlardan başlayarak daha üst seviyedeki bir veri tabanına kadar çok sayıda veri kaynağı ile standart bir yöntem vasıtası ile haberleşmenin sağlanması OPC'nin oluşumunda önemli bir motivasyon kaynağıdır.

OPC, Microsoft'un OLE / COM (Object Linking & Embedding / Component Object Model) standardına dayanır. OLE / COM, Microsoft'un farklı uygulamalar arası bütünleşmeyi hedefleyen nesne tabanlı bir teknolojisidir. OPC ise OLE tabanlı bir haberleşme standardıdır ve farklı otomasyon seviyeleri arasında hızlı ve güvenilir bir bağlantı sağlar. OLE teknolojisinin kullanımıyla OPC, ister yönetim katındaki bir uygulama olsun isterse süreç denetiminde kullanılan bir uygulama olsun, farklı uygulamalar arası veri alış verişinin, tanımladığı yöntem ve nesneler ile, standart bir şekilde gerçekleşmesini sağlar.

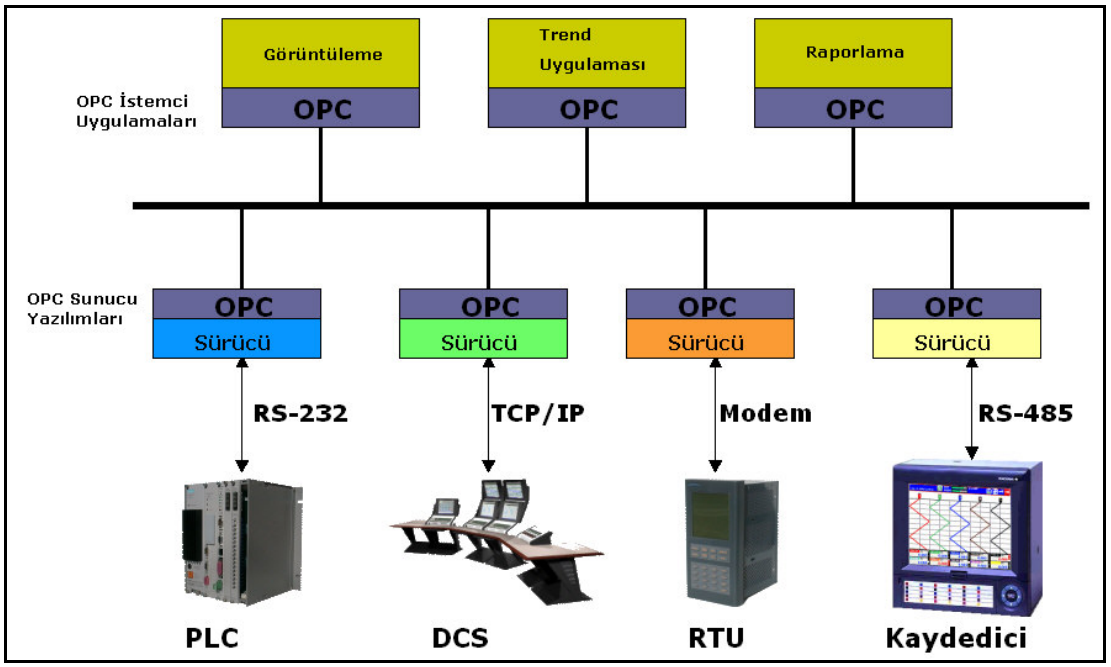


Şekil 3.3 : OPC Derneğinin logosu

OPC Görev Birliği, çalışmalarını yoğun bir şekilde yaparak ilk standardını 1996 yılının Ağustos ayında OPC Standardı Sürüm 1.0 olarak piyasaya çıkarmıştır. Hemen peşinden de Eylül 1996'da Chicago'daki ISA (Instrument Society of America) gösterisinde, kar amacı gütmeyen OPC Derneği kurulmuştur. Şekil 3.3'te OPC Derneği'nin logosu görülmektedir. O tarihten sonra OPC Görev Birliği'nin tüm çalışmaları dernek çatısı altında devam etmiş ve günümüze kadar etkinliğini sürdürmüştür [7] .

3.3 OPC ile Problemlerin Çözümü

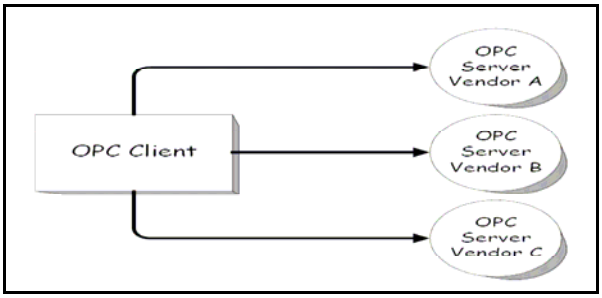
OPC tanımladığı standart arayüzle donanım üreticileri ve yazılım sağlayıcıları arasına belirgin bir çizgi çizmiştir. Artık donanım üreticileri, sadece tek bir sürücü geliştirerek –ki bu OPC Sunucu yazılımıdır, piyasada bulunan tüm OPC İstemci özelliğine sahip yazılımlarla haberleşebilme özelliğine kavuşmuş olmaktadır.



Şekil 3.4 : OPC arayüzü ile her yazılım standart bir arayüzle haberleşir

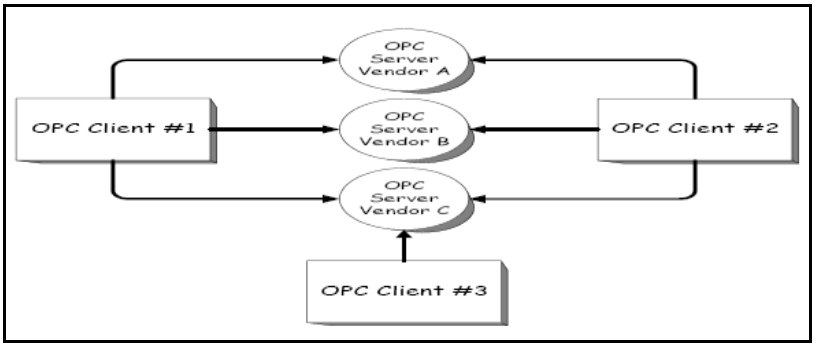
Şekil 3.2'de gösterilen örnek bir otomasyon sistemi uygulamasında, OPC'den önce, her uygulama yazılımı ürününe özel sürücü gelişimi ve bunun yol açtığı sorunlar ele alınmıştı. Şekil 3.4'te ise OPC kullanımı ile, sistemde kullanılan tüm yazılım ve donanımlar için tek ve standart bir arayüz olan OPC Arayüzü kullanılmış ve ürüne özel sürücülerin tüm dezavantajlarından kurtulunmuş olunmaktadır.

OPC'nin sağladığı standart arayüz dışında, İstemci – Sunucu (Client - Server) yapısındaki çalışma prensibi donanıma olan erişim problemlerinin önüne geçmiştir. OPC istemcisi olan bir yazılım, sahip olduğu standart arayüzle, Şekil 3.5'te gösterildiği gibi, aynı anda aynı arayüzü kullanarak farklı markalara ait OPC Sunucu programlarına bağlanabilir ve o programlar aracılığıyla fiziksel donanımla veri haberleşmesinde bulunabilir.



Şekil 3.5 : Bir istemci birden fazla sunucuya aynı anda bağlanabilir.

Aynı şekilde bir OPC sunucu programı Şekil 3.6'ta gösterildiği gibi, birden fazla istemci programa aynı anda hizmet verebilir.



Şekil 3.6 : Bir sunucu birden fazla istemciye hizmet verebilir.

3.4 OPC'nin Faydaları

OPC'nin endüstride yaygın bir standart olarak kabul edilmesi ile birlikte tüm donanım ve yazılım üreticileri artık sadece tek bir arayüzü desteklemekle yetinebilmektedirler. Artık bir donanım için, farklı üreticiler tarafından üretilmiş sürücülere gerek kalmamıştır. OPC standardını destekleyen bir ürün, OPC uyumlu herhangi bir ürünle doğrudan haberleşebilmektedir.

OPC, OLE standardını yapı tanımlamaları, arayüzler ve bazı teknikler geliştirerek daha verimli veri transferi yapılmasını sağlamıştır. OPC, OLE'nin tüm avantajlarını kullanırken onu süreç kontrolu endüstrisinin ihtiyaçlarına uygun hale getirecek düzenlemeler de yapmıştır.

OPC'nin faydaları endüstriyel otomasyonun farklı kesimlerinde yer alan gruplar açısından şöyle özetlenebilir.

Donanım Üreticileri açısından,

- Cihazları piyasadaki tüm OPC uyumlu yazılımlar tarafından kullanılabilir.
- Ürün güncellemelerinde yenilenmesi (update) gereken tek bir ürün kalmıştır.
- Destek verilecek tek bir ürün vardır.

Yazılım Geliştiricileri açısından,

- Haberleşilecek cihaza ait özelliklerin bilinmesi gerekmez.
- Tek bir arayüz gelişimi için zaman ve kaynak ayrılır.

- Tek bir arayüzle piyasada OPC desteği olan tüm cihazlarla haberleşilir.
- Standart arayüz kullanıldığından sorun ihtimali azalır.

Sistem Entegrator firmalar açısından,

- Tek bir ürüne veya yazılıma bağımlılık ortadan kalkar.
- Proje geliştirme zamanı, standart arayüz sayesinde kısalır.

Son Kullanıcı açısından,

- Seçim şansını arttırır.
- En uygun olan donanım/yazılım çiftini seçebilir.

3.5 OPC Standartları

OPC derneği ilk olarak 1996 yılında yayınladığı ilk standardın endüstride yaygın olarak kabul edilmesinin ardından, piyasadan gelen istekler üzerine, canlı veri erişiminin yanı sıra, otomasyon yazılımlarında ihtiyaç olan diğer haberleşme kaynakları için de standart sunucu ve istemci arayüzü yayınlamaya başlamıştır. Şu an kullanımda olan OPC standartları Tablo 3.1'de listelenmiştir.

Tablo 3.1: Şu an kullanımda olan OPC Standartları

Standart	**İçerik**	**Sürüm**
OPC Data Access	Gerçek zamanlı veri okuma ve yazma	3.00
OPC Alarm & Events	Sistemde tanımlanan olayların görüntülenmesi	1.10
OPC Historical Data Access	Geçmişe yönelik verilerin okunması	1.20
OPC Batch	OPC-DA standardının özel bazı sektörler için genişletilmiş hali	2.00
OPC Security	Arayüzlerde güvenli bağlantılar için	1.00
OPC XML-DA	OPC ile XML'in birleşmesi	1.00
OPC Data Exchange	OPC Sunucular arası direk bağlantı	1.00
OPC Commands	OPC Sunucuların işleyişleri hakkında çalışma anında bilgi alımı	1.00
OPC Complex Data	Cihazlarda oluşturulan karmaşık veri tiplerine erişim	1.00
OPC Unified Architecture	Tüm OPC standartlarını bir araya getiren ve Web Servislerini kullanan yeni standart	1.00

3.6 OPC Uyumlu Yazılımlar

Günümüzde OPC uyumlu yazılımlar, bir başka deyişle, OPC arayüzüne sahip olan yazılımların sayısı hızla artmaktadır. Bu konuda derneğin ağ sitesi olan www.opcfoundation.org adresinde arama yapılabilir ve OPC İstemci veya OPC Sunucu olan yazılımlar görüntülenebilir. OPC'nin oluşturulma amaçlarından biri, PLC/DCS gibi otomasyon cihazlarından veri okunmasıdır. Ama günümüzde, sadece kontrol sistemleri yazılımları arasında değil, diğer gerçek zamanlı veri haberleşmesinde bulunacak yazılımlarda da OPC arayüzüne rastlamak mümkündür.

3.6.1 OPC Sunucusu Olan Donanımlar

OPC'nin artan önemi ve kendini ispatlamış güvenli alt yapısı nedeniyle, bugün, endüstriyel otomasyon donanımı üreticileri arasında OPC Sunucu yazılımı olmayan marka kalmamıştır. Öyle ki, donanım üreticileri haricinde bir çok yazılım firması da piyasada bulunan donanımlar için OPC sunucusu üretmektedir. Bu firmalar OPC sunucu geliştirmeden önceleri ürüne özel sürücü geliştiren firmalardır. OPC'nin esnek çalışma mantığı sayesinde, geçmişte geliştirdikleri sürücülerini OPC arayüzü ile donatmışlar ve ürüne özel olan sürücülerini genel bir sürücü haline getirmişlerdir. Tablo 3.2'de OPC Sunucusu bulunabilecek bazı donanımlar listelenmiştir. Dikkat edilirse hemen hemen tüm üreticiler OPC Sunucu yazılımı çıkarmışlardır.

Tablo 3.2: OPC Sürücüsü olan donanımlar

ABB	FGH Controls	Matsushita	Scantronic
Allen-Bradley	Fike	Mettler Toledo	SDS network
APEX	Frick	Mintz Systems	Sharp
Applicom Int.	Gavish	Mitsubishi	Shinko Technos
ASI network	General Electric	Modicon	Siemens
B&R Industrie	Grinnel	Moeller	Sofrel Telecontrol
Bristol Babcock	Harris	Motorola	Square D
CANopen network	Hartmann Braun	Notifier	Techno Trade
Cerberus (Siemens)	Hilscher	Observator	Telemecanique
Cincinnati Milacron	Hima	Omni	Texas Instruments
ControlNet network	Hitachi	Omron	TOA Corp.
Cristplant	Idec (Izumi)	Opto-22	Trane
Cutler Hammer	IMS	Perax	Unitronics
Dixell	Interbus network	Philips	VG Gas Systems
DeviceNet network	Introl	Profibus network	Vision Systems Ltd
DuTeC	ITT Flygt	Red Lion Controls	Westinghouse
Echelon Lonworks	Johnson Controls	RKC	WIT Concept
Elco Corporation	Kenelec	Saia	Yokogawa
Electromatic	Keyence	Samsung	York International

Ancak şunu da belirtmek gerekir ki OPC Sunucu programı olanlar sadece donanımlar değildir. Günümüzde bazı yazılımların da OPC Sunucusu geliştirilmektedir. Örneğin, bir çok kontrol sisteminden çeşitli yollarla bilgi okuyan bir SCADA sisteminin de OPC Sunucu yazılımı olabilmektedir. Bu sayede o SCADA programındaki canlı veriler, tek bir arayüzden diğer uygulamalara aktarılabilmektedir.

3.6.2 OPC İstemcisi Olan Yazılımlar

OPC istemcisi programlar arka planda kullanılan donanım ne olursa olsun OPC sunuculardan bilgi alabilir ve veri yazabilir. Bu açıdan bakıldığında, herhangi bir program OPC istemci arayüzünü barındırarak canlı sistemlerle etkileşime geçebilir. Bu yüzden, şu an kayıtlı sayısı, on bini aşan istemci yazılımları endüstrinin bir çok alanında kullanılmaktadır. Bunların çoğu SCADA, IMS, MES programları gibi doğrudan süreci etkileyen uygulamalar olduğu gibi, ofis programlarından, veri tabanı programlarına ve hatta akademik alanda çok sık kullanılan MATLAB programına kadar uzanmaktadır. Bu sayede MATLAB ile fiziksel donanımlar arasında, canlı veri alış verişi sağlanmış olmaktadır.

Tezin uygulama kısmı da MATLAB üzerinde gerçekleştirilecektir.

3.7 OPC'nin Yaygınlığı

OPC Derneğinin, 2003 yılında ARC şirketine yaptırdığı kamuoyu araştırmasında çıkan sonuca göre, endüstriyel kullanıcılar arasında OPC standardının bilinirliği %84 olmuştur. Ayrıca OPC arayüzüne sahip olma oranı, sektörde kullanılan yazılımlar arasında MES sistemlerinde %78, SCADA sistemlerinde %75, PLC ve DCS sistemlerinde %68 ve ERP sistemlerinde %53 oranında kullanıma sahiptir [7].

3.8 Örnek Bir Otomasyon Sistemi

OPC'nin otomasyon sistemlerine yaptığı katkıyı daha rahat görebilmek maksadıyla, OPC'den önceki ürüne özel sürücü yöntemi ve OPC'den sonra OPC Sunucu ve İstemci yapısına dayanan yöntemlerin aynı sistem üzerindeki uygulamalarına bakmak faydalı olacaktır.

3.8.1 Ürüne Özel Sürücü Kullanılarak

Şekil 3.7'de gösterilen klasik anlamda bir otomasyon sisteminde iki PLC ünitesi ve bunlara bağlı dört SCADA istasyonu görülmektedir. Bu sistemde PLC-SCADA haberleşme alt yapısı RS-232 ve RS-485'tir. Bilgisayarların seri portlarından RS-485 ortamına dönüşüm yapılmış ve PLC'lerin RS-485 uyumlu haberleşme kartlarına giriş yapılmıştır. Bu yapı yedekli bir yapıdır. Bu yapıda, her bir operatör istasyonu, her bir PLC'ye yeni bir bağlantı noktası ekleyerek, PLC'ler ile haberleşmelerini sağlamaktadırlar.

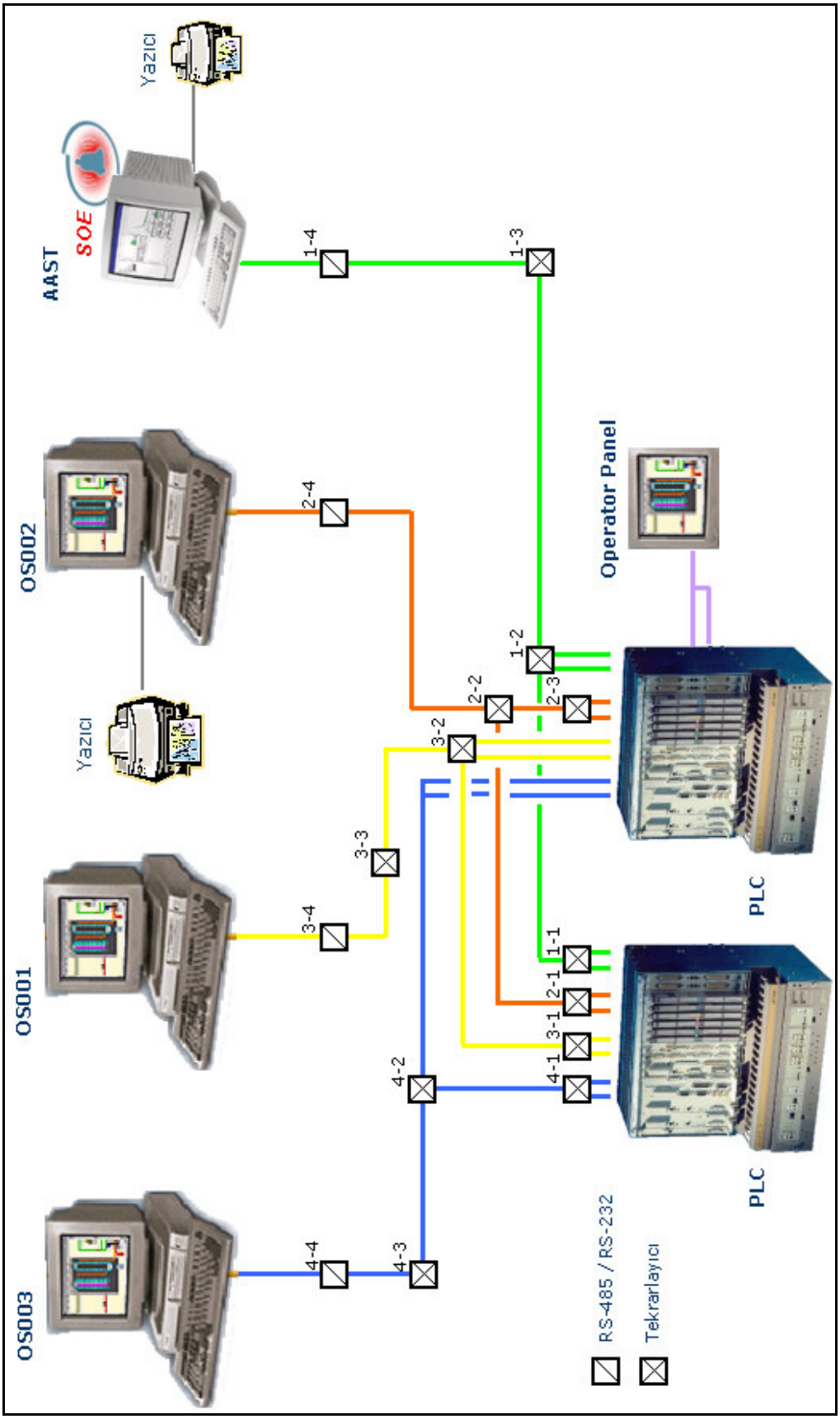
Yapı olarak oldukça güvenli gözüken bu mimaride, ilk olarak, haberleşme alt yapısının karmaşıklığı ve genişlemeye uygun olmadığı göze çarpmaktadır. Her bilgisayardan ayrı bir haberleşme kablosu PLC'lere çekilmiş ve PLC'lere, haberleşeceği her bilgisayar için, bir haberleşme kartı ilave edilmiştir. Bu da PLC'ler için ilave bir CPU yükü demektir.

Uygulamalar açısından baktığımız zaman, her uygulamanın kendine has sürücüsü ile haberleştiğini görmekteyiz. Bu da ekranlar arası bilgilerin tutarlılığını azaltmaktadır. Öyle ki, bir ekranda bir değer başka bir formatta gösterilirken diğer bir ekranda aynı değer başka bir biçimde gösterilebilmekte ve bu da ekranlar arası veri takibinde sorunlar çıkarabilmektedir.

3.8.2 OPC Kullanılarak

Şekil 3.8'te ise aynı sistemin OPC kullanımı ile nasıl gerçeklenebileceği gösterilmektedir. İlk olarak göze çarpan husus, kuşkusuz, haberleşme alt yapısındaki sadeliktir. Bu kez, PLC'lerle sadece OPC Sunucu programı kurulu olan bilgisayarlar haberleşmekte ve diğer uygulama yazılımları ise bu OPC Sunucu bilgisayarlara bağlanmaktadırlar. Burada diğer sistemde olduğu gibi yedekliliği sağlamak açısından iki bilgisayara OPC Sunucusu kurulmuş ve bunlar birbirleri ile yedekli çalışacak şekilde ayarlanmışlardır.

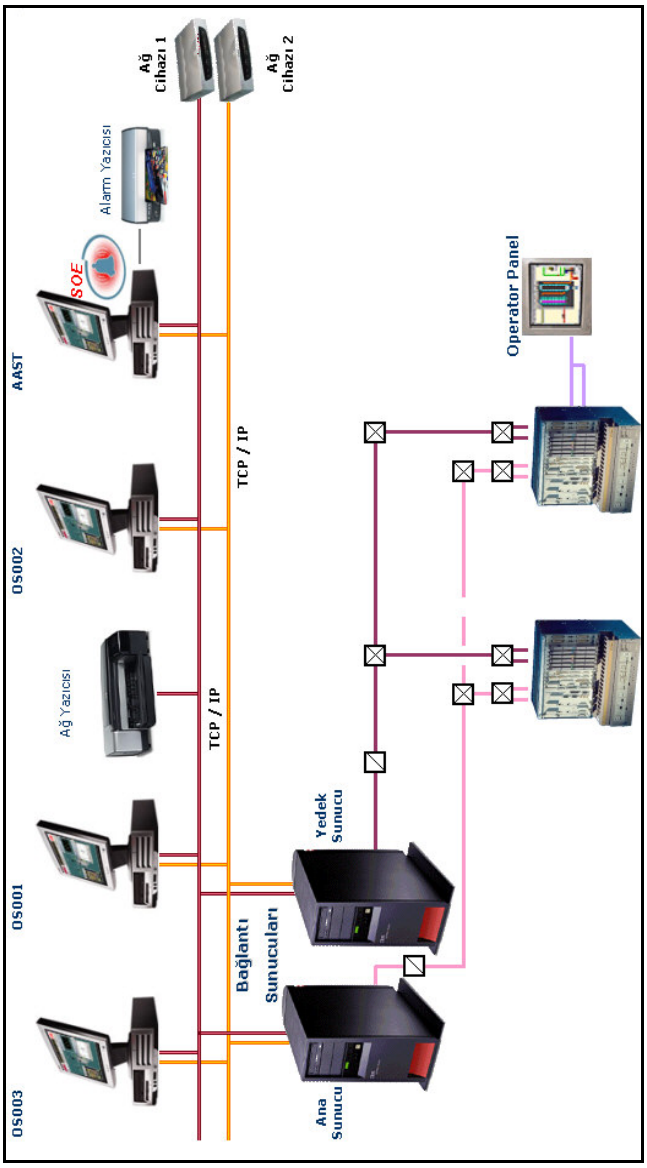
PLC tarafından bakıldığı zaman bu yapının en büyük avantajı haberleşme kartlarındaki azalmadır. Üstelik bu azalma uygulama yazılımları açısından bir kayıp olmadan sağlanmıştır. Önceki bağlantı şeklinde de her uygulama PLC'lere iki yoldan bağlantı kurarken bu yerleşimde de iki farklı yoldan bağlantı kurabilmektedirler. Yani yedeklilikten ödün vermeden alt yapı rahatlatılmıştır. Bu rahatlamanın PLC'lerin CPU yüklerinde de olumlu etkisi mutlaka görülecektir.



Şekil 3.7 : OPC kullanımından önce örnek bir haberleşme altyapısı

Uygulama yazılımları açısından baktığımız zaman, tüm uygulamaların aynı arayüzü kullandıklarını görmekteyiz. Bu da haberleşilen veriler arasında tutarlılık anlamına gelmektedir. Aynı zamanda, uygulama yazılımları ile OPC Sunucu veri kaynakları arasında standart bir TCP/IP ethernet ağı kullanılmış olması, bakım konularında önemli avantajlar sağlamaktadır. Günümüzde UTP kablo ve sonlandırma birimleri RJ-45 modüllerini standart olarak kolaylıkla elde etmek mümkündür. Ayrıca bu konulardaki genel bilgi birikimi özel sistemlere göre oldukça yüksektir. Bu da bir sorun anında daha hızlı çözüm sağlanması anlamına gelmektedir.

Kullanılan haberleşme alt yapısı genişleyebilir niteliktedir ve gelecekte yapılabilecek ilavelere açık bir yapıdadır.



Şekil 3.8 : OPC kullanımı ile örnek bir haberleşme alt yapısı

4. DAĞITILMIŞ BİLEŞEN NESNE MODELİ - DISTRIBUTED COMPONENT OBJECT MODEL (DCOM)

OPC teknolojisinin temelini oluşturan DCOM mimarisinin ilk örneği, Windows'un 1987 yılında uygulamalar arası veri transferi için geliştirdiği "Pano" (Clipboard) yöntemi idi. Bu yöntem çok basit bir şekilde Kopyala-Yapıştır şeklinde çalışır ve kullanımı da gayet kolaydır. Ancak bu mekanizma dinamik bir yapıya sahip değildir. Bunun yerine kısa bir süre sonra DDE (Dynamic Data Exchange – Dinamik Veri Alışverişi) yapısı geliştirildi. Bu sayede uygulamalar arası veri transferi dinamik olarak yapılabiliyordu. Ancak DDE yönteminin kullanımı Pano yöntemine göre daha zahmetlidir.

1992 yılına gelindiğinde, Microsoft ,metin tabanlı programlar arasında dinamik veri alış verişini sağlayan OLE (Object Linking & Embedding) standardını geliştirdi. Bu teknikle, örneğin, bir Excel tablosundaki değişikliğin otomatik olarak bir Word dökümanında görülmesi (Linking) ve dahası direk olarak Word dökümanı içerisinden değişikliğin (Embedding) yapılması sağlanmıştır. Bu aşama birinci OLE (OLE1) safhası olarak adlandırılır. 1995 yılında, OLE1 bir adım daha öteye götürülerek, sadece metin tabanlı programlar arasında değil herhangi iki uygulama arasında da kullanılabilecek şekilde yeniden geliştirilmiş ve sonucunda OLE2 standardı ortaya çıkmıştır. Bu standart aynı yılın sonlarına doğru Bileşen Nesne Modeli - Component Object Model (COM) adını almış ve bundan sonra, Microsoft'un nesneye dayalı dağıtılmış sistemlerinin temelindeki teknoloji olmuştur.

DCOM ise COM özelliklerinin farklı bilgisayarlar üzerinde de kullanılabilmesiyle 1996 yılında ortaya çıkmıştır [7].

4.1 COM'un TEMELLERİ

COM programlama C programlama değildir. Üstelik COM bir programlama dili değil bir programlama yöntemidir. Bu sebeple COM'un nasıl çalıştığını anlamak ilk başlarda biraz karmaşık gözükebilir. Ancak özellikle COM terminolojisi ile C++ terimlerini karşılaştırarak işe başlamak bu karmaşıklığı azaltacaktır.[8]

Örneğin xxx adında bir C++ sınıf'a (class) sahip olunsun. Bu class'a ait MethodA, MethodB ve MethodC işlevleri (fonksiyonları) tanımlanmış olsun. Her işlev belli sayıda parametreleri kabul eder ve bir sonuç (result) üretir. Class tanımı şu şekilde yapılır.

```
class xxx {
public:
        int MethodA(int a);
        int MethodB(int b);
        float MethodC(float c);
};
```

Sınıf tanımı sadece bir tanımlamadır. Sınıf tanımı kullanılarak bu işlevlerin bir örneği oluşturulur. İşlevi yerine getiren bu örneklerdir. Class ise sadece işlevin tanımını verir. Üye bir işlevi çağırma işlemi bir işaretçi (pointer) ile gerçeklenir. Örneğin;

```
xxx *px;   // pointer to class xxx
px = new xxx;   //create object on heap
px -> MethodA (1)   // call method
delete px;   // free object
```

COM kodu yazılırken ise new ve delete komutlarının yerine daha değişik komutlar kullanılır.

```
ixx *pi   // pointer to ixx COM interface
CoCreateInstance (,,,,&pi)   // create interface
pi -> MethodA();   // call method
pi -> Release();   // free interface
```

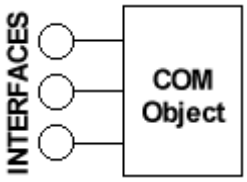
Yukarıda yazılan koda bakıldığında sanki C++'taki sınıf tanımı ile COM'un Arayüz (interface) tanımının örtüştüğü görülür. Ancak burada C++'taki gibi new ve delete komutlarının yerine CoCreateInstance ve Release komutları kullanılmaktadır. İşte bu noktada COM'un en önemli kavramlarından biri olan arayüz tanımı üzerinde biraz durmak gerekir.

4.1.1 Arayüz (Interface)

Sözlük anlamı olarak bakıldığında arayüz iki cisim arasındaki ortak yüzey olarak karşımıza çıkmaktadır. Bu tanım COM terminolojisinde de aslında aynı anlamda geçerlidir. Arayüz, sunucu ve istemci arasında ortak bir platform sunmaktadır.

Sunucuya ait bir COM nesnesi tüm işlevlerini bir arayüz aracılığıyla sağlar. O nesneden sağlanılacak olan tüm veriler bir arayüz üzerinden gerçekleştirilir. Arayüzlerde de işlev tanımlamaları bulunmaktadır. İstemci bu arayüzden istediği işlevi çağırarak sunucudan bu işlevin sonucunu bekler.

COM'un güçlü istemci-sunucu mimarisi de burada yatmaktadır. Bir istemci sunucuya ancak arayüzler üzerinden ulaşabilir. Bu sayede arka plandaki sunucu ne kadar farklı veya karmaşık olursa olsun istemci sadece ve sadece bir arayüz üzerinden isteklerini gerçekleştirecektir. Bu da haberleşmede tam bir saydamlık (transparency) sağlamaktadır.



Şekil 4.1 : Bir COM nesnesi ve buna ait arayüzlerin gösterimi

Şekil 4.1'de bir COM nesnesinin arayüzleri ile birlikte göterimi bulunmaktadır. İstemci tarafından bakıldığı zaman COM nesnesi küçük yuvarlaklarla gösterilen çeşitli arayüzlerden ibaretttir. Her bir arayüzde o arayüze ait işlevler bulunmaktadır. İstemci açısından işlevin nasıl gerçeklendiği, hangi ortamda yazıldığı gibi konular önemli değildir. Zira istemci sadece ve sadece o işlevin yerine getirilmesi için gerekli parametreleri sağlamaktadır.

COM'un arayüz yapısını anlamak için gerçek hayattan şu örneği verebiliriz. Kullandığımız arabayı bir COM nesnesi olarak görelim. Bu nesneye ait çeşitli arayüzleri Sürüş ve Radyo olarak ele alalım. Sürüş arayüzünde de şu işlevlerin tanımlı olduğunu düşünelim. İleri, Geri, Sağa, Sola, Hızlanma, Yavaşlama vb. Radyo arayüzünde ise Açma, Kapama, Ses arttırma, Ses azaltma vb.

İstemci olarak ise aracı kullanan şoförü düşünebiliriz. Şoför araba nesnesinin sadece ve sadece arayüzleri ile etkileşim halindedir. Yani bir sürücünün aracı kullanması için motorun nasıl çalıştığını veya fren sisteminin aracı nasıl durdurduğunu veya radyosunu kullanırken bir sonraki istasyonun nasıl bulunduğuna dair bir bilgisi olması gerekmez. Örneğin Sürüş arayüzünde Sağa işlevi için direksiyonu kullanması veya Hızlanma işlevinin yerine getirilmesi için gaz pedalına basması yeterli olacaktır. Burada direksiyonun ne kadar çevrildiği veya gaz pedalına ne kadar basıldığı sunucuya o işleve ait parametreler olarak gönderilmektedir. Sunucuda bu isteklere göre kendi içinde belirlenen mekanizmaları çalıştırarak, yani bir anlamda bu iş için yazılmış programını çalıştırarak, istenen işlevi yerine getirir.

İstemci, yani sürücü, bu işlevin nasıl yerine getirildiğinden tamamen bağımsızdır. Sürücü araca sadece yavaşlamasını söyler ancak nasıl yavaşlayacağını (hidrolik bir sistemle mi, hava ile çalışan bir sistemle mi) nesnenin, yani aracın, iç tasarımı belirler. Sürücü araca Yavaşla isteğini iletip araçta yavaşladığı sürece arka plandaki işleyiş önemli değildir. Bu açıdan baktığımızda ilk tasarlanan araçlar ile günümüz son model araçlarının arayüzlerinin çok fazla değişime uğramamasına rağmen, araç içerisinde kullanılan tekniklerin son derece değiştiğini ve geliştiğini görebiliriz. Ancak sürücü olarak bizlerin bu gelişen teknikler hakkında çok az veya hiçbir bilgimiz olmadan da aynı arayüzleri kullanarak yeni sistem araçları kullanabilir olduğumuzu fark edebiliriz.

4.1.2 IUnknown – Tüm Arayüzlerin Kaynağı

Bu özel arayüz tüm COM arayüzlerinin kaynağıdır. Tanımlanan tüm nesneler bu özel arayüzü barındırır. Bu arayüzün 3 işlevi vardır bunlar;

- QueryInterface
- AddRef
- Release

Bu üç işlevden ilki olan QueryInterface belli bir arayüzün COM nesnesinde var olup olmadığını sorgulamak için kullanılır. AddRed ve Release işlevleri ise COM nesnesinin hafızada ne kadar süre ile kalacağını belirlerler. Bu özel arayüz üzerinde daha derin bilgiler verilebilir ancak bu biraz ileri seviye bir konu olacağından IUnknown'un temel işlevlerinin bilinmesi şu aşamada yeterlidir.

4.1.3 COM Nesnesi (COM Object)

COM için nesne arayüzlerin bir araya gelmesi ile oluşturulur. Bir COM nesnesinde çeşitli alanlar için ayrılmış arayüz tanımlamaları ve bu arayüzlere ait çeşitli işevler bulunur. Bir COM nesnesi oluşturmak için şu adımların yapılması gereklidir.

- Bir COM nesnesi oluşturulur ve buna bir isim verilir.
- Bu nesneye ait arayüz(ler) tanımlanır ve bun(lar)a isim verilir.
- Tanımlanan arayüz(ler)e ait işlevler tanımlanır ve bunlara isim verilir.
- Bu COM Nesnesini barındıran COM Sunucusu kurulur.

Burada dikkat edilmesi gereken en önemli husus isimlendirmedir. COM herhangi iki uygulama arasında kullanılbileceğinden dolayı dünya üzerindeki bir çok programcının benzer amaçlar için yazdıkları COM nesnelerinin ve arayüzlerinin aynı isimleri taşıması muhtemeldir. Bu durumda aynı isimli farklı işleri yapan COM nesne ve arayüzlerinin bir arada kullanılması ciddi karışıklıklara yol açabilir. COM'un tanımı gereği tüm makinalar üzerinde çalışabilirliği garanti edilmelidir. Bu durumda isimlendirme de karışıklığın önüne geçilmesi için bazı önlemler alınmalıdır. İşte bu noktada GUID (Globally Unique Identifier) kavramı bu isimlendirmelerin herbir tanım için tek olmasını sağlamaktadır. Bu konu ile ilgili açıklama Bölüm 4.2'te verilmiştir.

4.1.4 COM Sunucusu (COM Server)

COM sunucusu COM nesne ve arayüzlerini barındıran programdır. Sunucular genel olarak üç şekilde çalışırlar

- Sürceç içi veya DLL sunucuları şeklinde
- Bağımsız EXE sunucuları şeklinde
- Windows NT servisi şeklinde

COM nesne ve arayüz tanımlamaları kullanılacak olan sunucu çeşidinden bağımsızdır ve her üç çeşit için aynıdır. İstemci açısından da hiçbir fark yoktur. Ancak COM sunucu programının yazımı (kaynak kodu) kullanılacak olan şekile göre önemli değişiklikler gösterir.

Süreç içi sunucuları DLL (Dynamic Link Libraries) ile çevrim içi çalışmada sürece yüklenirler. COM işlemleri uygulama ile aynı süreçte yürütülür.

Süreç dışı sunucular ise istemci sunucu arasında daha kesin bir çizgi çizer. Bu çeşit sunucular ayrı bir EXE programı olarak kendi süreçlerinde çalışırlar. Bu EXE programının çalıştırılması ve durdurulması SCM, Windows Service Control Manager, tarafından yapılır.COM arayüzlerine yapılacak olan istekler süreçler arası haberleşme mekanizmaları tarafından sağlanır. Bu mekanizma hakkında ki bilgiler Bölüm 4.3'te verilmiştir. Eğer bağımsız çalışan bu sunucu EXE başka bir bilgisayar üzerinde ise buna Distributed COM (Dağıtılmış COM) denilmektedir.

Windows NT servislerine dayanan sunucu çeşidinde ise sunucu Windows NT tarafından idare edilmektedir ve herhangi bir kullanıcı Windows ortamına giriş yapmasa bile Windows açılışı ile birlikte çalışır duruma getirilebilir.

4.2 GUID (Globally Unique Identifier)

Bir ismin tüm dünya üzerinde tek olmasını sağlayabilecek iki yol vardır.

- Yarı Resmi veya Resmi bir kurum tarafından isimlerin onaylanması
- Her seferinde değişik bir isim üreten özel bir algoritma kullanılması

İlk yol olarak bahsedilen yol bugün internet ortamındaki isimlerin kayıt edilmesinde kullanılan yöntem olduğu bilinmektedir. Bu yöntem ile her ülke kendi alan adlarını tescil ederken çeşitli kurallar uygulamaktadırlar. Örneğin ülkemizde ODTÜ bünyesinde faaliyet gösteren internet alan adları tescilini yapan birim bir işyerine ait .tr uzantılı ismin internet üzerinde yer alabilmesi için çeşitli belgeler istemekte ve belli bir ücret ödenmesini istemektedir. Üstelik bu işlemlerin tamamlanması bir kaç hafta sürmektedir.

Aynı yöntemin COM programcıları içinde uygulanması ve COM nesne ve arayüz tanımlamalarının isimlendirilmelerinde kullanılabilirliğini düşündüğümüzde dünya üzerindeki program ve programcı sayısı göz önüne alındığında pekte mümkün olmadığı anlaşılacaktır.

İkinci yöntem ise yazılımcılar açısından daha rahat bir yöntemdir. Ancak bunun için bir algoritma bulunması ve bu algoritmanın her seferinde değişik sonuç üretmesi beklenmektedir. Bu yönde ilk çalışmalar Open Software Foundation (OSF)

tarafından yapılmış ve ağ adresi, 100 nano saniye hassasiyetinde zaman ve bir sayaç bilgisinin biraraya getirildiği algoritmayı geliştirmiştir. OSF bu algoritmaya UUID, Universally Unique Identifier, demiştir. Ancak daha sonraları Microsoft aynı algoritma için GUID, Globally Unique Identifier adını kullanmıştır.

Basit ama oldukça güçlü olan bu algoritma ile her seferinde değişik bir isim üretmek mümkün olmuştur. Algoritma sonucu 128-bitlik bir rakamdır. Ancak kolay okunabilmesi için 16'lık sayı düzeninde gruplandırılarak ifade edilir. Örnek bir GUID numarası şu şekildedir.

65904879-C3D4-4567- AEF34FD345A2

128-bitlik bir sayı (2 üzeri 128) oldukça büyük bir rakamdır. Öyle ki dünyanın varolduğu zamandan bu yana her 100 nano saniyelik dilimleri saysa idik şu an hala 39-bitlik boş alanımız olurdu. İnternette olan her makinanın IP adresi farklı olduğundan ve her 100 nano saniyelik dilim göz önüne alındığından bu algoritmanın dünya üzerinde kaç kişi çalıştırırsa çalıştırsın herbir kullanıcıya farklı numara verebilme ihtimali çok çok yüksektir.

Microsoft Visual Studio programının bir parçası olarak sunduğu GUIDGEN programı ile kullanıcısına bu GUID numarasını sağlamaktadır.

COM ortamında da herbir nesne ve arayüz için bu numaralar kullanılmaktadır. Yani aslında bir COM nesne ve arayüzü için iki isim mevcuttur. Bunlardan biri dökümentasyonlarda ve tanımlamalarda kullanılması için anlamlı ve okunaklı bir isim ve diğeri ise makina ortamında tek bir referans sağlaması için GUID numarasıdır.

4.3 Proxy ve Stub

COM mimarisinde istemci ve sunucu arasında tam bir saydamlık sağlamak için her iki birim arasındaki haberleşmeler Proxy ve Stub üzerinden gerçekleştirilir. İstemci sunucudan bir istekte bulunacağı zaman bu isteğini Proxy'ye iletir ve Proxy'deki bu istek sunucuya iletilmek üzere Stub'a gönderilir. Bu işleme Marshalling denilmektedir. Daha sonra Stub'a gelen bu istek sunucuda istenilen işleve dönüştürülür ki bu işleme de Demarshalling denilmektedir. Proxy ve Stub dosyaları genellikle tek bir DLL dosyasında olmaktadır. Microsoft'un IDL (Interface Definition Language) derleyicisi COM nesne ve arayüzlerine ait tüm işlev ve

parametreleri C kodundaki dosyalara çevirir. Daha sonra bu dosyalar tek bir DLL dosyasında birleştirilir. Sunucu ve istemci çevrimiçi çalışmada bu dosyadan faydalanarak birbirleri ile haberleşirler.

4.4 Belirteçler (Identifier)

Önceki bölümlerde program açısından isimlerin aslında GUID isimleri olduğundan bahsedilmişti. GUID numaraları COM ortamında şuralarda kullanılır;

- CLSID: Class ID – COM Nesnesinin eşsiz belirtecidir. Çevrimiçi ortamda bir COM Nesnesinin çağırılabilmesi için o nesneye ait CLSID'nin parametre olarak belirtilmesi gerekmektedir.
- IID: Interface ID – Nesneye ait arayüzün eşsiz belirtecidir. Yine çevrimiçi çalışmada belirli bir arayüze erişebilmek için QueryInterface işlevine istenilen arayüzün IID'si gönderilir.
- CATID: Category ID – Nesnelerin bağlı oldukları kategoriyi göstermeleri için kullanılan belirteçdir. Bu sayede belli bir kategoriye ait olan tüm COM nesne ve bileşenleri ortaya çıkarılabilir.

COM sunucu yazılımı oluşturulurken belirlenen bu belirteçlerin istemciler tarafından bilinebilmesi ve kullanılabilmesi için kayıt defterine işlenmesi gerekmektedir. Bunu yapmak için iki yöntem bulunmaktadır. Birinci yöntem bir .rgs dosyası ile tüm bu bilgilerin bilgisayarın kayıt defterine eklenmesidir –ki bu yöntem pek pratik ve güvenilir bir yöntem değildir ya da ikinci bir yol olarak sunucu bileşenlerinin kurulumu sırasında bu bilgilerin kayıt defterine otomatik olarak işlenmesidir. Bu sayede istemcinin sunucu ile iletişime geçmesi için gereken tüm bilgiler kayıt defterinde bulunmuş olacaktır.

5. OPC OVERVIEW ve OPC-DA STANDARTLARI

OPC Standardı ilk başta sadece anlık veri erişimine yönelik düşünülen bir standarttı. Ancak daha sonraları bu standardın başarılı sonuç vermesiyle otomasyon yazılımı dünyasından diğer alanlarda da bu tip standartların olması isteği giderek artmış ve neticede önceki bölümlerde zikredilen (Alarm&Event, HDA, Batch, Command, Security vd.) standart yapılar oluşturulmuştur.

Her yeni OPC standardı yayınlandıkça görülmüştür ki her kısımda ortak olan tanımlamalar ve teknikler vardır. İşte bu kısımların bir araya geldiği OPC Overview (OPC Genel Bakış) ve OPC Common Definitions and Interfaces (OPC Ortak Tanımlamalar ve Arayüzler) standartları bu ortak noktaları ortaya koymuştur. Bu standartta OPC'nin uygulama alanları, OPC'nin gerisindeki teknoloji ve mevcut OPC Standartları tanıtılmıştır. Ayrıca Ortak Tanımlamalar ve Arayüzler'de tüm OPC Standartlarında kullanılan tanımlamalar ve arayüzler belirtilmiştir.

Bu teze konu olan kısım gereği bu standartlardan sadece Data Access (Anlık Veri Erişimi) ile ilgili olan OPC-DA Standardı ile detaylı bilgi verilecektir. Bunun için önce OPC Overview & Interfaces hakkında bilgi de sunulacaktır.

5.1 OPC Overiview

5.1.1 Standart Nesneler ve Kullanımı

OPC sunucu ve istemcileri bazı bileşenleri paylaşırlar. Bunlar,

- Proxy/Stub dosyaları
- OPCServerBrowser bileşeni
- Automation Wrapper bileşeni

OPC haberleşmesinin olacağı her ortamda bu bileşenlerin mutlaka olması gerekmektedir. Özellikle OPCServerBrowser bileşeni istemci programların sunucu programları bulmalarına olanak sağlaması bakımından oldukça önemli bir bileşendir. 2003 yılından itibaren OPC Derneği bu üç kalemin bir arada olduğu OPC Core

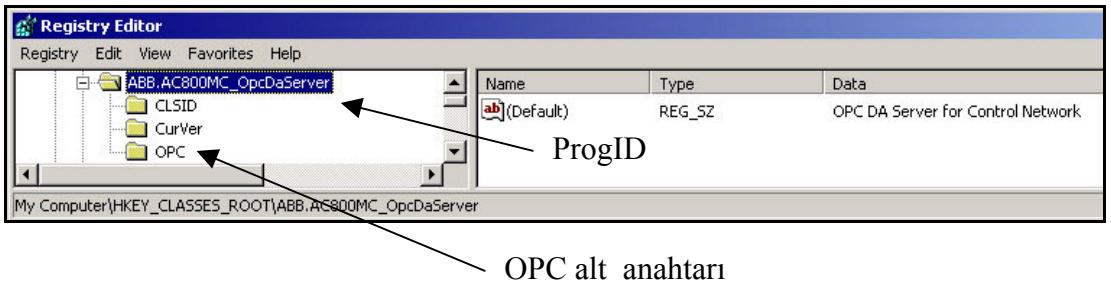
Components (OPC Çekirdek Bileşenleri) dosyalarını internet sitesi üzerinden ücretsiz olarak sunmaktadır. Ancak bu bileşenlerin sağlanması yükümlülüğü OPC Sunucu üreten tarafa aittir.

5.1.2 Kurulum ve Kayıt Gereksinimleri

OPC istemcisinin sunucuyu çalıştırabilmesi için o sunucuya ait CLSID'sini bilmesi gerekmektedir. Dolayısıyla çevrimiçi durumda istemci yazılıma bu bilginin girilmesi gerekmektedir. Bu işlem ya el ile bir kereye mahsus olmak üzere istemci yazılım kodlarına girilecektir –ki bu yöntem hiç pratik olmaz ya da istemci yazılım bu bilgiyi bir yerden parametre bilgisi olarak almalıdır. Bu amaçla her OPC Sunucusu Windows Kayıt Defterine şu 3 kaydı mutlaka yapar.

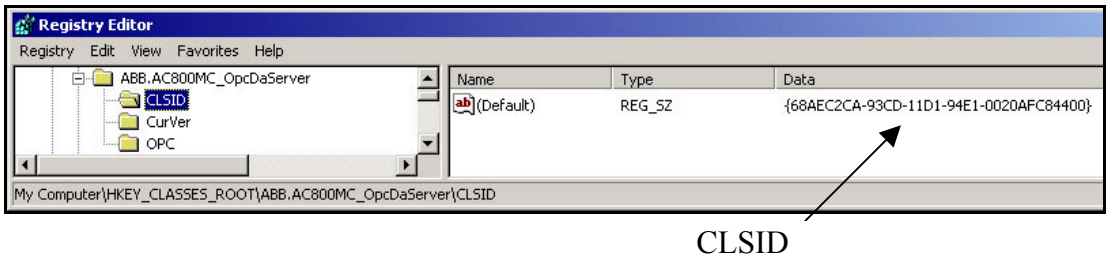
- ProgID: ProgramIdentifer – Sunucu bileşeni tanıtan okunaklı bir metin yazısı. Bu yazının standardı <ÜreticiAdı>.<SunucuAdı> şeklindedir. Örn. WIZCON.WIZOPCDA şeklinde.
- CLSID: ClassIdentifier – 128-bitlik GUID numarasıdır. Bu numara tek ve eşsizdir. Dolayısıyla bu numara ile belirtilen bileşen sadece ve sadece istenilen OPC sunucuya aittir. COM çevrimiçi ortamında sunucunun istemci tarafından başlatılabilmesi için bu CLSID'ye ihtiyacı vardır.
- AppID: Application ID – Sunucunun diğer bir okunaklı belirtecidir. Standartta bunun içeriği serbest bırakılmıştır.

ProgID mutlaka OPC adında bir alt anahtara (sub key) sahip olmalıdır. Bu sayede kayıt defterinde OPC sunucuları aramak mümkün olmaktadır. Şekil 5.1'de OPC alt anahtarının bulunduğu bir kayıt gösterilmektedir.



Şekil 5.1 : Kayıt defterinde OPC alt anahtarını barındıran bir kayıt örneği

Ayrıca her ProgID'nin bir de CLSID adlı alt anahtarı vardır. Bu alt anahtarda o sunucunun 128-bitlik GUID numarası bulunur.



Şekil 5.2 : ProgID anahtarındaki CLSID bilgisi

Bulunan CLSID de kayıt defterinde bir anahtar adıdır. Dolayısıyla bu CLSID ismiyle kayıt defteri bir kez daha arandığında sunucu ile ilgili daha detaylı bilgilere ulaşılabilmektedir.

Zira birden fazla OPC standardı olduğu göz önüne alındığında bilgisayardaki OPC Sunucuları bulmak için sadece OPC alt anahtarının ve CLSID bilgisinin aranması yeterli gelmemektedir. Zira bir DA istemcisi yine bir DA sunucusu ile haberleşmek isteyecektir. Benzer şekilde bir A&E istemcisi karşısında A&E sunucusu isteyecektir. Dolayısıyla her farklı standart için yine 128-bitlik bir CATID (Category ID) kullanılmıştır.

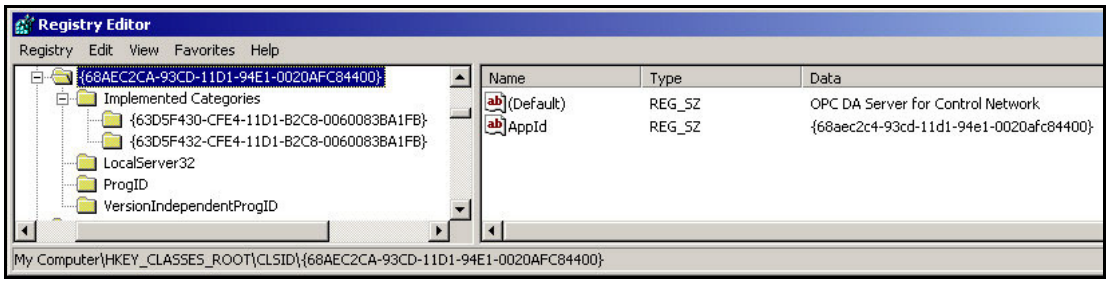
Tablo 5.1'de bazı OPC Standartlarına ait CATID bilgileri verilmiştir. CLSID bilgilerinin aksine bu numaralar istemci programlarda sabit olarak tanımlanmıştır.

Tablo 5.1: OPC Standartları ve CATID bilgileri

Standart	**CATID**
OPC Data Access 1.0A	{63D5F430-CFE4-11D1-B2C8-0060083BA1FB}
OPC Data Access 2.0	{63D5F432-CFE4-11D1-B2C8-0060083BA1FB}
Alarm & Event 1.0	{58E13251-AC87-11D1-84D5-00608CB8A7E9}
OPC Batch 1.0	{A8080DA0-E23E-11D2-AFA7-00C04F539421}
OPC Historical DA 1.0	{7DE5B060-E08911D2-A5E6-000086339399}
OPC Batch 2.0	{843DE67B-B0C9-11D4-A0B7-000102A980B1}

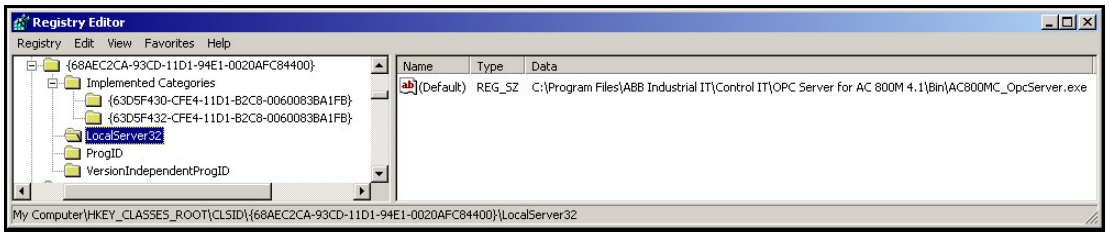
CATID bilgileri sayesinde sunucunun hangi kategoride bir sunucu olduğu belirlenmektedir. Bu bilgiler CLSID olarak bulunan GUID numarasının kayıt defterinde aranması ve bulunan anahtarın altındaki ImplementedCategories adlı alt anahtarlarında bulunur. Şekil 5.3'te CLSID'ye ait bilgilerin kayıt defterindeki alt anahtarları ve bunlara ait değerler gösterilmektedir.

Ayrıca yine CLSID anahtarında sunucuya ait diğer bilgilerde bulunmaktadır. Örneğin sunucu yazılımının EXE dosyasının nerede olduğu bilgisi burada bulunabilir. Şekil 5.4'te sunucu yazılımın bilgisayardaki konumu LocalServer32 alt anahtarında yazdığı görülmektedir.



Şekil 5.3 : CLSID anahtarına ait alt anahtarlar

İstemci tarafından sunucu işlemlerinin başlatılabilmesi için gerekli olan bilgiler yukarıda açıklandığı gibi kayıt defterinde bulunmaktadır. Peki bu bilgilerin program ortamında istemci yazılım tarafından elde edilmesi nasıl olacaktır? Bu sorunun cevabı OPCServerBrowser'dır. OPCServerBrowser sayesinde yerel veya uzak bilgisayarlardaki OPC sunucuları aramak ve bunlar ile iletişime geçmek mümkündür.



Şekil 5.4 : Sunucu yazılımın bilgisayarda kurulu olduğu yer bilgisi

5.1.3 OPCServerBrowser

OPC Derneği tarafından sağlanan OPCServerBrowser bir DCOM Sunucudur. Diğer tüm DCOM sunucuları gibi bu sunucununda istemci tarafından çalıştırılabilmesi için CLSID bilgisine ihtiyaç duyulmaktadır. Ancak bu bilgi sunucu yazılımın oluşturulması sırasında programın içine gömülür ve her istemci yazılım OPCServerBrowser DCOM sunucusunu çalıştırabilir.

OPCServerBrowser IOPCServerList arayüzünde şu işlevleri sunmaktadır.

- EnumClassesOfCategory: İstemci yazılım bu işlevi çağırıp haberleşmek istediği OPC sunucunun kategori bilgisini, CATID, parametre olarak gönderir. Örneğin OPC-DA v2.0 bir sunucu ile haberleşmek için {63D5F432-CFE4-11D1-B2C8-0060083BA1FB}GUID numarası sunucuya iletilir. OPCServerBrowser ise aldığı bu bilgi ile kayıt defterini tarar ve belirtilen kategoriye ait CLSID numaralarını istemciye gönderir.

- GetClassDetails: İstemci yazılım aldığı CLSID bilgilerinden birini bu işlevin bir parametresi olarak sunucuya gönderir. Sunucu sonuç olarak belirtilen CLSID'ye ait ProgID bilgisini alır. İstemci programının kullanıcısı açısından CLSID çok anlamlı bir bilgi değildir. Dolayısıyla bu CLSID'nin okunaklı kısmını barındıran ProgID bilgisinin elde edilmesi istemci kullanıcısı açısından daha önemlidir.
- CLSIDfromProgID: Bu işlev ise bağlanılacak olan sunucuya ait ProgID'nin bilinmesi durumunda kullanılır. Bu kez OPCServerBrowser sunucusuna ProgID bilgisi gönderilir ve buna ait CLSID bilgisi alınır.

Tekrar belirtmekte fayda var ki bu işlemler hem yerel makina üzerinde hem de uzak makina üzerinde gerçekleştirilebilir. Yani OPCServerBrowser yerel makinanın kayıt defterini tarayabildiği gibi ağ üzerindeki başka bir makinanın kayıt defterini inceleyebilir ve o makina da kurulu olan OPC sunucuların listesini istemciye iletebilir.

5.2 OPC Data Access (DA)

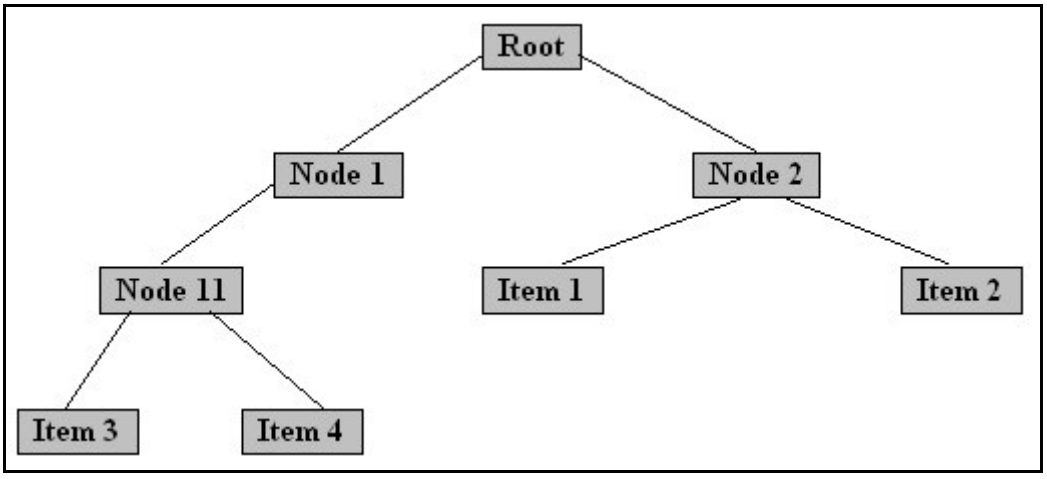
OPC-DA Standardı süreçteki canlı verilere erişimde istemci ve sunucu programları arasında bir arayüz sağlar. OPC-DA sunucuları bir veya birden fazla OPC-DA istemcilerine çeşitli veri kaynaklarına erişimlerini sağlarlar. Bu veri kaynakları basit bir transmiter bilgisi veya bütün bir denetleyici bilgisi olabilir. OPC istemci programı ise basit bir Excel programı olabileceği büyük Visual Basic görüntüleme programı veya bütün bir ERP programının bir parçası olabilir.

OPC sunucuları birden fazla istemciye aynı anda hizmet edebildikleri gibi, OPC istemcileri de aynı anda birden fazla OPC sunucusuna bağlanıp bilgi alabilirler.

5.2.1 İsim Alanı (NameSpace) ve Nesne Düzeni

OPC-DA Standardında sunucu erişebildiği tüm veri okuma ve yazma noktalarını bir İsim Alan'ında (NameSpace) toplar. Bu alan sunucu yazılımı sırasında istenildiği şekilde düz veya hireyaşik bir düzende oluşturulabilir. Hireyaşik düzende her seviye bir düğüm (node) noktası olarak adlandırılır ve her düğüm noktasının altında ona ait yapraklar (leaf) bulunur. Yapraklara aynı zamanda Parça (item) denilmektedir. Bu

şekilde bir hiyeraşik yapı aslında bilgisayar dünyasında çok kullanılan ağaç yapısı olarakta bilinir. Örnek bir isim alanı Şekil 5.5'te gösterilmiştir.



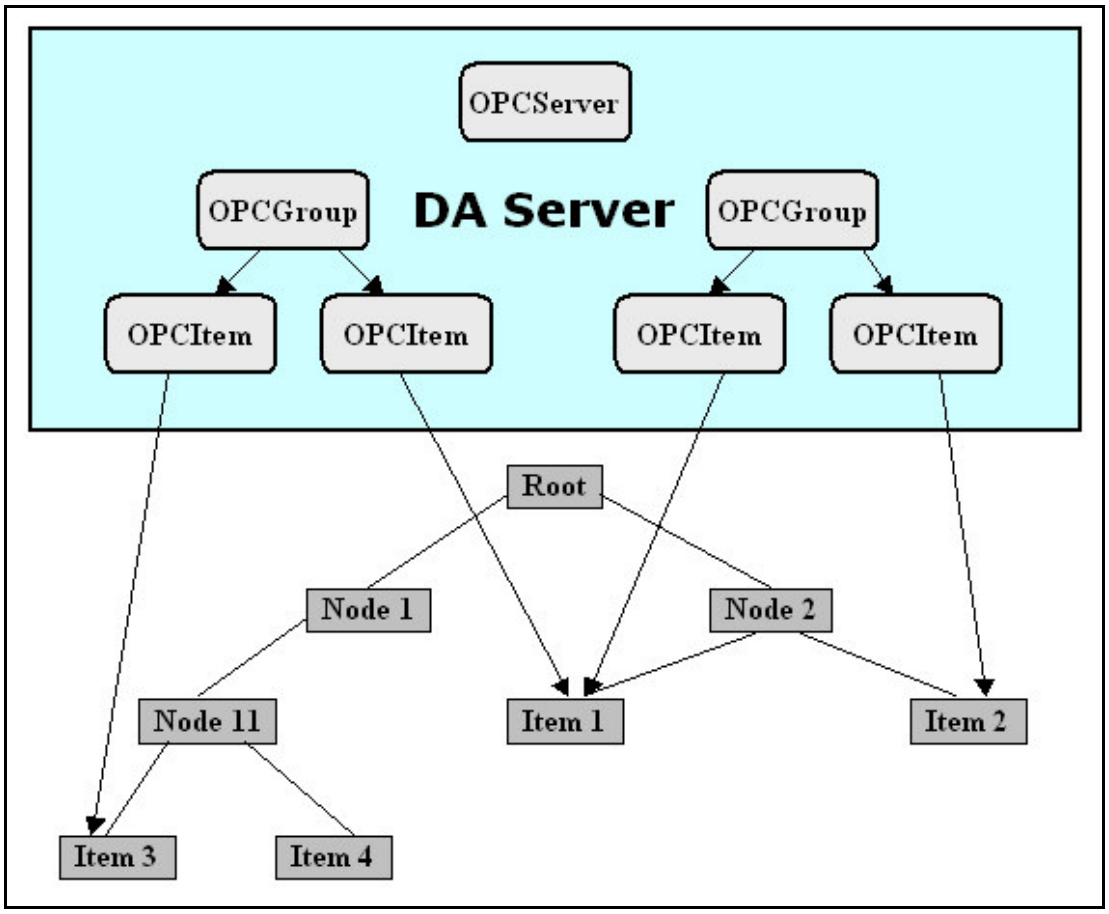
Şekil 5.5 : Örnek bir İsimAlanı yerleşimi

OPC sunucu içerisinde tanımlanmış olan parçalara (yapraklara) sadece o değişkenlerin değeri değil başka diğer özelliklerde eklenebilir. Örneğin Item3 olarak yukarıda belirtilen değişkenin açıklaması, program içindeki yeri, eğer bu bir ölçüm cihazı ise ayar bilgisi gibi statik zamanla değişmeyen ancak süreç için faydalı olarak olan bilgileri aynı parçaya (item) birer Property (Özellik) olarak eklenebilir.

Bu yapı tamamiyle OPC sunucuya ait olan bir yapıdır. İstemci program kendi yapısını oluşturabilir. Örneğin bu gruplandırma sürecin çeşitli alanlarına ayırarak yapabilir. A kısmı verilerini bir grupta B kısmı verilerini bir grupta toplayabilir. Ya da süreçte hızlı değişen değerleri bir grupta toplayabilir. Bir sonraki kısımda detaylıca ele alınacağı gibi istemci program OPCServer, OPCGroup ve OPCItem nesnelerini oluşturarak süreç üzerinde kendi yapılandırmasını yapabilir.

5.2.2 Standart Nesneler ve Kullanımı

OPC istemci programı ilk olarak OPCServer nesnesini oluşturur. Bu sayede sunucunun isim alanı ve yapısı hakkında bilgi sahibi olur. İsim alanı hakkında bilgi edinildikten sonra OPCGroup nesnesi oluşturularak bu nesneye eklenecek olan OPCItem'lar için bir gruplandırma yapılır.



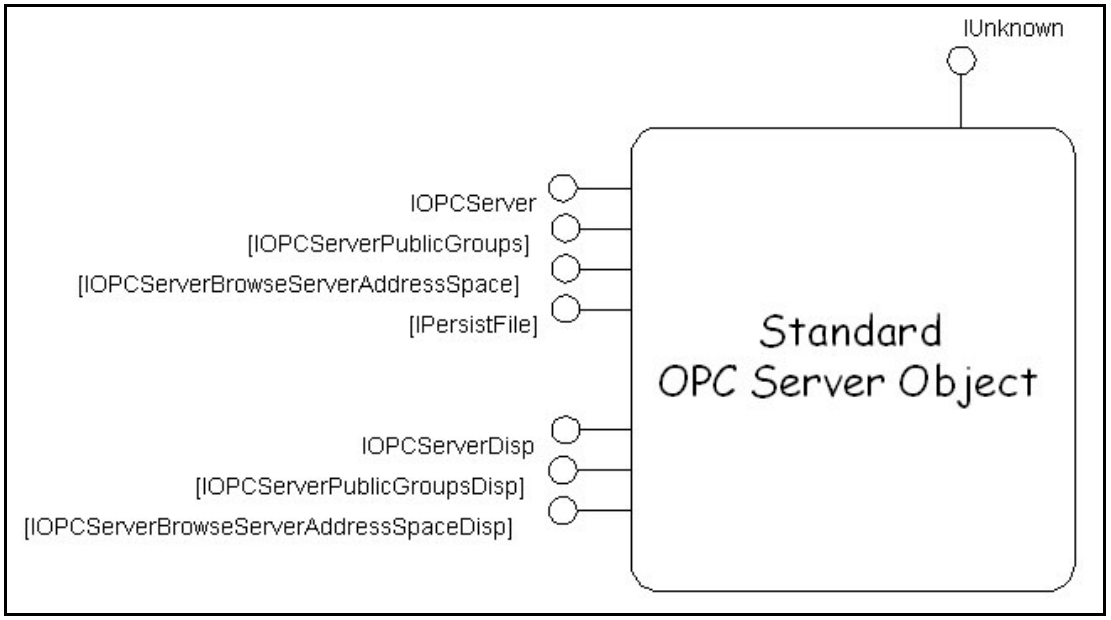
Şekil 5.6 : OPC istemci tarafından oluşturulan OPC nesneleri

Şekil 5.6 bir OPC istemcisinin OPC sunucuda oluşturduğu düzeni göstermektedir. Burada dikkat edilirse OPC sunucunun yazılması sırasında belirlenen isim alanı sabittir. Ancak üst kısımda ayrı bir çerçeve içinde gösterilen ve tamamiyle OPC istemcisi tarafından oluşturulan bölge, bu sunucuya bağlanacak olan her istemci tarafından ayrı ayrı oluşturulur. Dolayısıyla her istemci ilgilendiği değerler ve özellikler için bir düzen kurar ve bunların sonuçlarını sunucudan bekler.

5.2.2.1 OPCServer Nesnesi

OPC istemcisi tarafından ilk oluşturulan nesne OPCServer nesnesidir. Bu nesne sunucu ile olan bağlantının sağlanması ve isim alanının görüntülenmesi ve OPCGroup nesnelerinin yönetimi gibi çeşitli görevleri vardır.

Şekil 5.7'de gösterilen OPCServer nesnesinde tanımlı olan arayüzlerde şu noktalar göze çarpmaktadır.



Şekil 5.7 : OPCServer nesnesinin arayüzleri

- Her arayüz adı I harfi ile başlar
- OPC'ye özel olan arayüzler IOPC ile başlar
- Zorunlu olmayan arayüz tanımlamaları [ ] içine alınmıştır.
- Sonu Disp ile biten arayüzler VB gibi işaretçi kullanımını desteklemeyen uygulamalar için geliştirilmiştir.

Her arayüzde tanımlı çeşitli işlevler vardır. Bu işlevlerin listesi aşağıdaki gibidir.

IOPCServer

- AddGroup
- GetErrorString
- GetGroupByName
- GetStatus
- RemoveGroup
- CreateGroupEnumerator

IOPCServerPublicGroups (zorunlu değil)

- GetPublicGroupByName
- RemovePublicGroup

IOPCBrowseServerAddressSpace (zorunlu değil)

- QueryOrganization
- ChangeBrowsePosition
- BrowseOPCItemIDs
- GetItemID
- BrowseAccessPaths

IPersistFile (zorunlu değil)

- IsDirty
- Load
- Save
- SaveCompleted
- GetCurFileName

IOPCServerDisp

- Item
- AddGroup
- GetErrorString
- GetGroupByName
- RemoveGroup
- SaveConfig
- LoadConfig
- SetEnumeratorType

IOPCServerPublicGroupsDisp (zorunlu değil)

- GetPublicGroupByName
- RemovePublicGroup

IOPCBrowseServerAddressSpaceDisp (zorunlu değil)

- ChangeBrowsePosition
- SetItemIDEnumerator
- GetItemIDString
- SetAccessPathEnumerator

OPC Standardında tanımlı olan tüm arayüz ve işlevler hakkında daha detaylı bilgi için bkz. [9].

5.2.2.2 OPCGroup Nesnesi

İsim alanının taranmasından sonra OPCItem'ların gruplandırılması için OPCGroup Nesnesi oluşturulur. Bu gruplandırmanın neye göre yapılacağı hakkında bir standart belirlenmemiştir. İstenilirse sürecin belli bir kısmına ait olan değişkenler bir gruba toplanabilir veyahut örnekleme zamanı aynı olan değişkenler bir grupta toplanabilir.

OPCGroup nesnesi oluşturulurken istemci program şu değerleri parametre olarak gönderirir.

- Grup ismi
- RequestedUpdateRate (talep edilen veri tipi)
- PercentDeadband (ölü bölge alanı % olarak)
- ActiveState (etkinlik durumu)

Grup isminin gönderilmesi zorunlu değildir. Eğer istemci bir isim göndermezse sunucu bir isim oluşturacaktır. Ancak diğer parametreler istemci tarafından gönderilmek zorundadır. Zira bu parametreler sunucu ile istemci arasında yapılacak olan veri haberleşmesinin sıklığını belirler. Öyle ki RequestedUpdateRate parametresi grup içerisindeki verilerin ne sıklıkla taranacağını belirlerken PercentDeadband parametresi istemciye yeni bir değer gönderilmesi için istemciye son gönderilen değer ile yeni okunan değer arasında ne kadar fark olması gerektiğini belirler. ActiveState parametresi ise gruba ait değişkenlerin okunup okunmayacağını belirler.

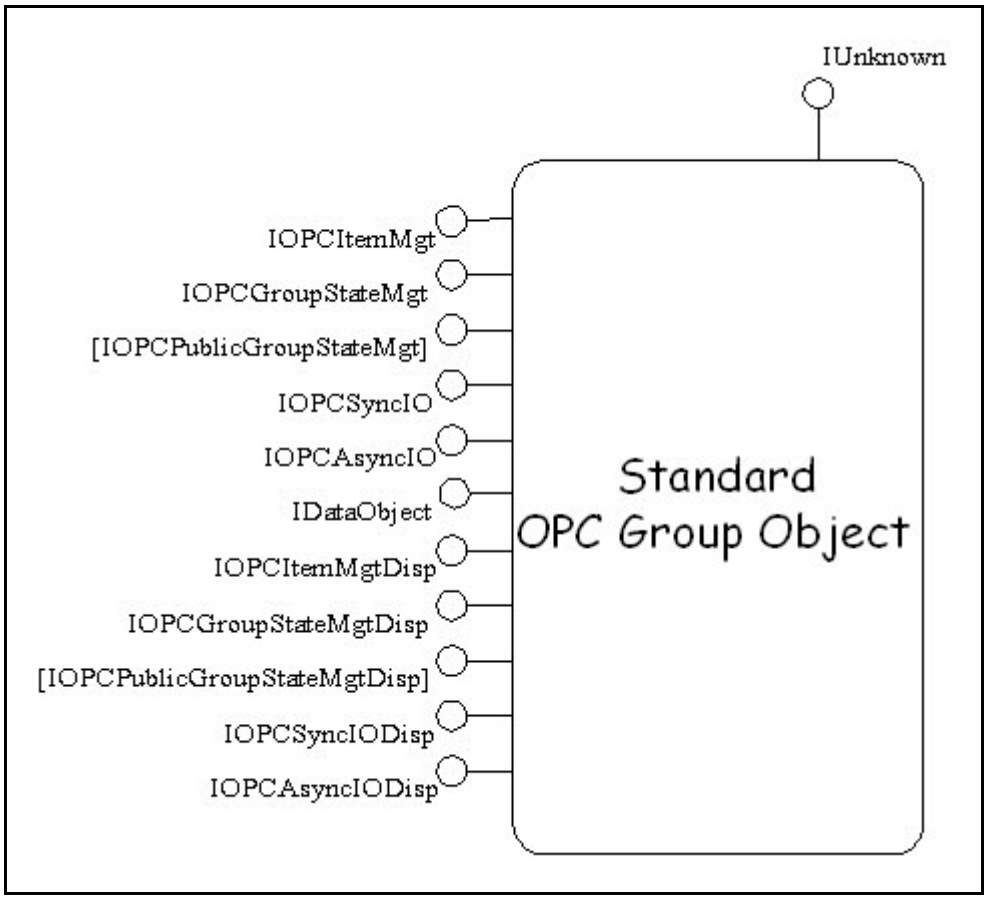
Şekil 5.8'de gösterilen OPCGroup nesnesine ait arayüzler ve bu arayüzlerde tanımlı olan işlevler şunlardır.

IOPCGroupStateMgt

- GetState
- SetState
- SetName
- CloneGroup

IOPCPublicGroupStateMgt (zorunlu değil)

- GetState
- MoveToPublic



Şekil 5.8 : OPCGroup nesnesinin arayüzleri

IOPCSyncIO

- Read
- Write

IOPCAsyncIO

- Read
- Write
- Cancel
- Refresh

IOPCItemMgt

- AddItems
- RemoveItems
- SetActiveState
- SetClientHandles
- SetDatatypes
- CreateEnumerator

IDataObject

- DAdvise
- DUnadvise

IOPCItemMgtDisp

- Item
- AddItems
- ValidateItems
- RemoveItems
- SetActiveState
- SetClientHandles
- SetDatatypes

IOPCGroupStateMgtDisp

- CloneGroup

IOPCSyncIODisp

- OPCRead
- OPCWrite

IOPCASyncIODisp

- AddCallbackReference
- DropCallbackReference
- OPCRead
- OPCWrite
- Cancel
- Refresh

IOPCPublicGroupStateMgtDisp

- MoveToPublic

Veri okuma ve yazma ile ilgili olan IOPCSyncIO ve IOPCAsyncIO arayüzleri ile bilgi için bkz. Bölüm 5.2.2.4.

5.2.2.3 OPCItem Nesnesi

OPCGroup nesnesi oluşturulduktan sonra IOPCItemMgt arayüzündeki işlevlerle OPCItem'lar oluşturulur. Bunlar oluşturulurken şu parametreler istemci tarafından sunucuya gönderilmesi gerekir.

- Fully qualified ItemID (isim alanında değişkeni tanımlayan yolun tam ismi)
- ActiveState
- RequestedDataType
- AccessPath
- ClientHandle

Sunucu bu parametreleri aldığında istemciye şu sonuçları gönderir;

- CanonicalDataType
- ServerHandle

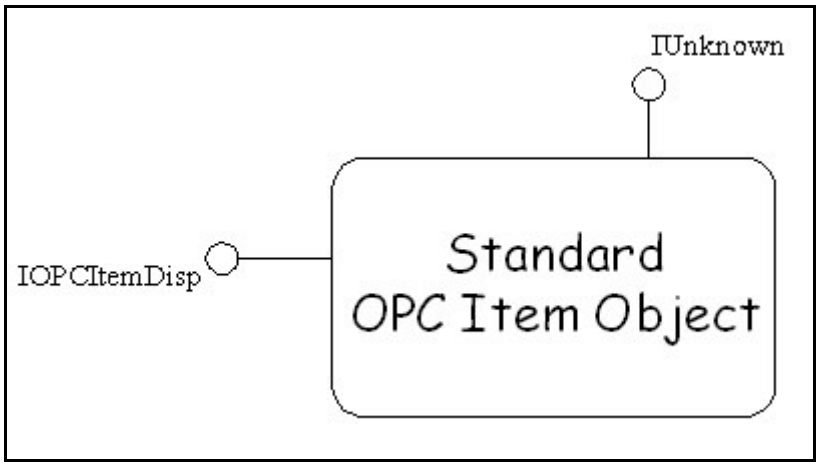
İsim alanında değişkeni tanımlayan yolun tam ismi sunucu içerisinde o değişkeni diğerlerinde ayrı kılan bir isimdir. Dolayısıyla bir istemci tarafından erişilmek istendiğinde o değişkenin öncelikle Fully Qualified ItemID ile tam yolu belirtilmelidir.

OPCItem'in durumu ActiveState işlevi ile active (etkin) veya Inactive (pasif) olarak ayarlanabilir. Bu sayede değişkenin otomatik olarak okunup okunmayacağı belirlenmiş olur.

İsim alanındaki tüm değişkenler DCOM'un standart veri tipi olan ve hernagi bir veri tipinde olabilen VARIANT tipindedir. Ancak istemci tarafı tüm veri tiplerini tanımlayabilir. Örneğin eğer istemci programda ondalık haneleri göstermek için gerekli veri tipi tanımlanmamışsa sunucudan okuyacağı değeri tam sayı olarak isteyebilir. RequestedDataType işlevi bu gibi durumlar için kullanılır. Sunucu isim alanındaki değişkenin istemcinin talep ettiği veri tipine dönüşümünü yapmak durumundadır. Eğer talep edilen veri tipi sunucunun desteklediği bir yapı değilse ve sunucu dönüşümü yapamayacaksa bu durumu istemciye bir geri dönüş parametresi ile bildirir.

Fully Qualified ItemID ile istemci tarafından sunucu içerisindeki tek bir değişkenin çağırılması mümkündür. Ancak öte yandan aynı Fully Qualified ItemID bilgisi ile diğer istemcilerinde bir OPCItem oluşturmaları olasıdır. Dolayısıyla sunucu açısından bakıldığında bir OPCItem açmak için sadece Fully Qualified ItemID yeterli değildir. Zira aynı değişkene bir okuma vyae yazma isteğinin nereden geldiğinin sunucu tarafından bilinmesi gerekir. Bunun için ClientHandle ve

ServerHandle bilgileri kullanılır. İstemci talep ettiği her değişken için 32-bitlik bir numara yayınlar (ClientHandle) buna karşılık sunucuda yine 32-bitlik bir numarayı (ServerHandle) atar. Farklı iki istemci aynı ClientHandle numarasını göndersede sunucu tarafında bunlara farklı ServerHandle numaraları atanarak her OPCItem için farklı bir ClientHandle-ServerHandle ikili bilgisi oluşturulur. Bu bilgi aynı zamanda sunucu içerisinde istenilen değişkene ait direk referans bilgisi olarak kullanılır. Zira sabit uzunluktaki bir değer ile çalışmak uzunluğu değişen bir değer ile çalışmaktan daha kolay ve etkin olacaktır.



Şekil 5.9: OPCItem nesnesinin arayüzleri

Şekil 5.9'da gösterilen OPCItem nesnesinde şu işlevler tanımlıdır.

IOPCItemDisp

- OPCRead
- OPCWrite

Süreçteki canlı değerlerin dışında uygulamarın başka öenmli bilgileri de mevcuttur. Örneğin cihazın adı, kalibrasyon bilgileri, bakım bilgileri vb. bilgilerde isim alanına kaydedilebilir. Ancak her bir özellik için ayrı bir parça (Item) oluşturmak yerine aynı parçaya ait alt özellikler olarak koymak isim alanının gereksiz yere büyük olmasını engeller. Bu amaçla DA 2.0 standardında Properties kavramı geliştirilmiştir. Her Property; PropertyID (özellik no), DataType (veri tipi) ve Description (açıklama) bilgilerini barındırır. PropertyID kısmında numaralandırma şu şekilde yapılır.

- 1-99 arası OPC standardında olan ve isim alanındaki her değişkene konulması mecbur olan özellikler için ayrılmıştır.

- 100-4999 arası yine standart dahilinde tanımlanan ancak sunucu üreticileri tarafından desteklenmesi mecbur olmayan özellikler için ayrımıştır.
- 5000-65535 arası sunucu üretici tarafından geliştirilen özellikler için ayrılmıştır.

Özelliklerin kullanılmasındaki asıl amaç süreçte değişmeyen statik bilgilerinde dinamik bilgilerin sağlandığı aynı arayüzle istemci programa iletilebilmesidir.

5.2.2.4 Değer Yazma ve Okuma

Şekil 5.6'da gösterilen düzeni sağladıktan sonra artık istemci program canlı sistemden veri okuyabilir ve yazabilir. Bu veri haberleşmesi Tablo 5.2'de özetlenmiştir.

Tablo 5.2: İstemci ve sunucu arsındaki veri haberleşme teknikleri

	Bellek (cahce)	**Cihaz (device)**
Senkron okuma (synchronous read)	x	x
Asenkron okuma (asynchronous read)	x	x
Yenileme (refresh)	x	x

Senkron ve Asenkron okumalarda istemci isteğini sunucuya iletip sonucu bekler. Bellek veya cihazdan erişme ise sunucunun cihaz ile olan haberleşmesine göre karar verilir. Zira eğer erişim noktasında cihaz istenilirse ve sunucunun cihaza asenkron olarak erişiminde gecikmeler oluşuyorsa istemcinin veri elde etmesinde sıkıntılar oluşabilir. Yenileme şeklinde iletişimde ise istemci tüm aktif OPCGroup'lardaki aktif OPCItem'ları okuyacaktır.

İstemciden veri yazmada ise yine senkron veya asenkron olarak işlem gerçekleştirilebilir. Ancak yazma işlemleri her zaman direk olarak cihaza yöneliktir. Yani önce belleğe atma ve daha sonra bu değeri yazma mümkün değildir.

Ayrıca istemci ile sunucu arasında yapılacak olan veri haberleşmesinin biçimi de şu şekilde tanımlanmıştır.

123.45 12.10.2007 10:34:23.100 00000000 11000000

İlk kısım süreçteki değeri taşımaktadır. Bu değer VARIANT tipindedir. Sunucu ve istemci programlar birbirleri arasında haberleşirken bu veri tipini kullanırlar. Ancak daha öncede bahsedildiği gibi VARIANT veri tipini kendi uygulamaları içinde istedikleri tipe dönüştürmekle yükümlüdürler.

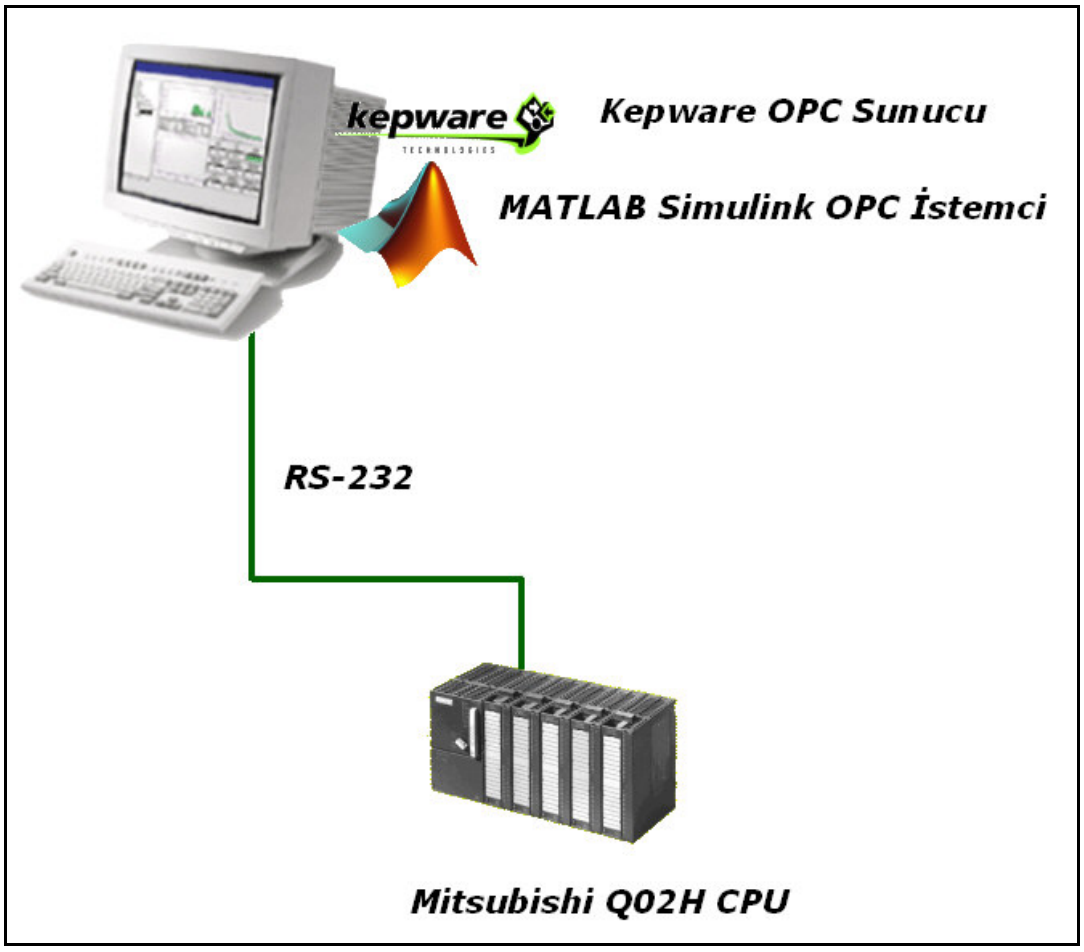
İkinci kısımda ise verinin zaman damgası, UniversalTimeCoordinated (UTC), bulunmaktadır. Bu değer okunan cihazdan alınabilinir. Eğer cihaz bunu desteklemiyorsa bu değer sunucuda üretilir.

Son kısımda ise durum bilgisi 16-bitlik bilgi halinde sunulmaktadır. Şu an için ilk sekiz bit kullanılmaktadır. Son sekiz bitte ise ilk iki bit değerin ne kadar sağlıklı olduğunun belirtisidir. Burada “Good” , “Bad” , “Uncertain” olarak değerlendirilir. Örneğin cihaz ile bağlantı sağlanamıyorsa “Bad” olarak işaretlenir. Eğer cihaz ile bağlantı var ama okunan değer anlamlı değilse “Uncertain” olarak işaretlendir. Daha sonraki dört bit ilk iki bitteki bilginin açıklaması için kullanılırken son iki bit daha detaylı bilgi verme amacı ile kullanılabilir.

6. UYGULAMA

OPC uygulaması olarak Mitsubishi Q serisi bir PLC'den MATLAB Simulink programına veri aktarımı ve PLC'ye veri yazma işlemi gerçekleştirilecektir.

Sistem diagramı Şekil 6.1'de gösterildiği gibi olacaktır.

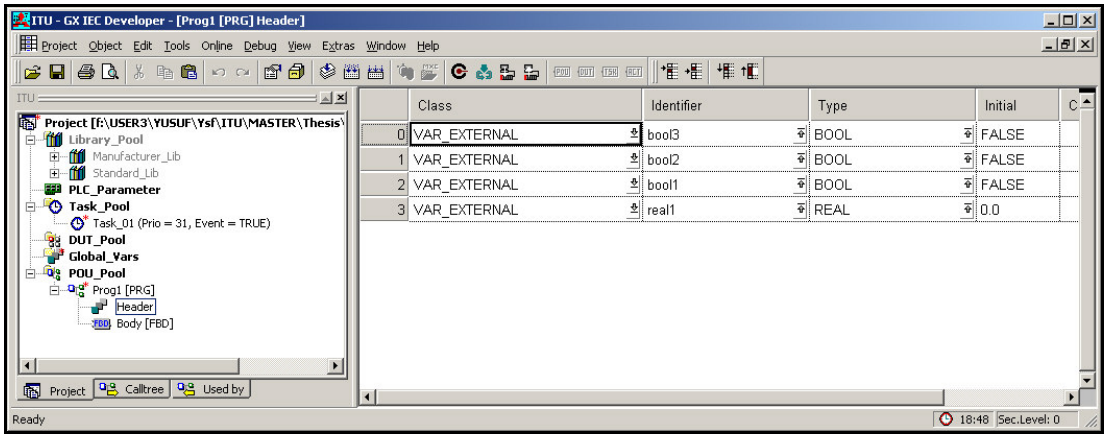


Şekil 6.1 : Uygulama sistem alt yapısı

Öncelikle Mitsubishi PLC'de okunacak ve yazılacak olan değişkenlerin adresleri not alınır. Eğer yoksa bunlar PLC editörü aracılığıyla Şekil 6.2'teki gösterildiği gibi oluşturulur.

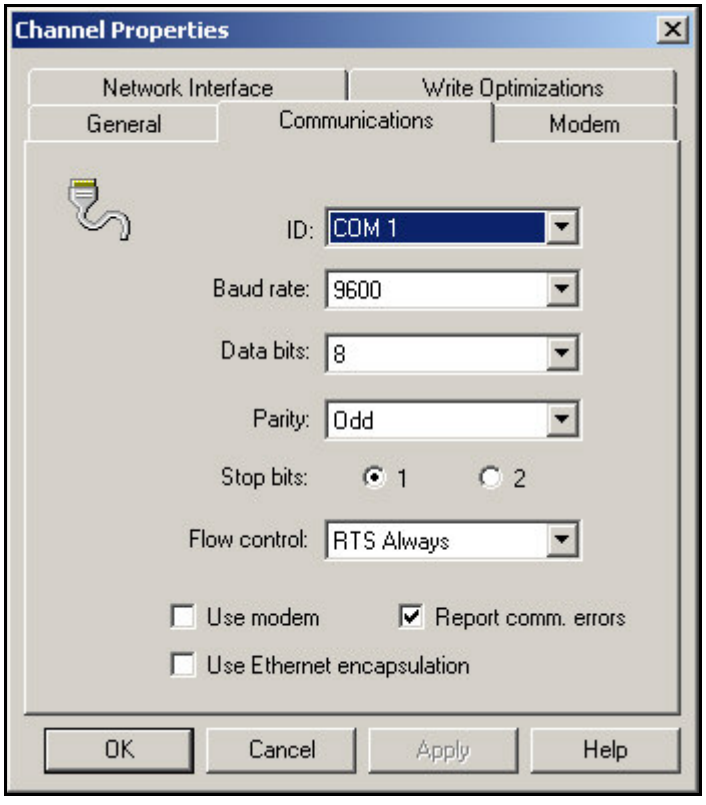
Daha sonra PLC'nin değişkenlerine ulaşacak olan Kepware OPC Sunucu programında bu değişkenler OPC değişkenleri olarak tanıtılır. Bunun için öncelikle

PLC ile Kepware OPC Sunucu arasındaki fiziksel bağlantı ayarları Şekil 6.3'te gösterildiği gibi girilir.

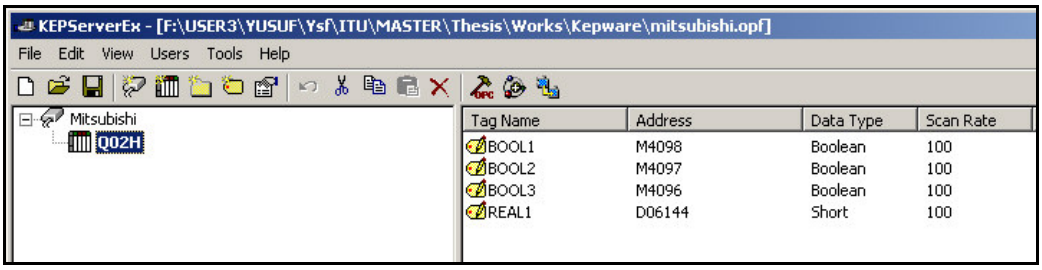


Şekil 6.2 : PLC editöründe değişken tanımlama

Bir sonraki adım OPC Sunucuda kullanılacak olan PLC değişkenlerinin oluşturulması işlemi vardır. Bunun için PLC'deki hafıza adresleri kullanılarak OPC Sunucuda uygun isimler verilecektir. Bu verilecek olan isimler OPC istemciler tarafından görülecek olan isimlerdir. Dolayısıyla anlamlı bir isim vermek bu noktada önemlidir. Şekil 6.4'te tanımlanan OPC isimleri gözükmektedir.



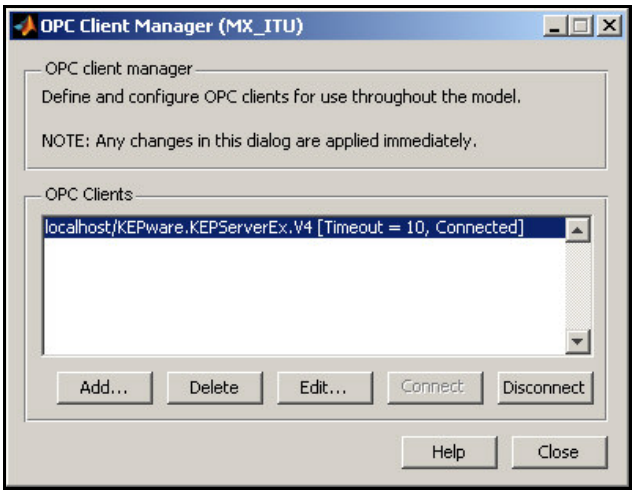
Şekil 6.3 : OPC Sunucu ve PLC arasındaki bağlantı ayarları



Şekil 6.4 : OPC sunucuda tanımlanan değişkenler

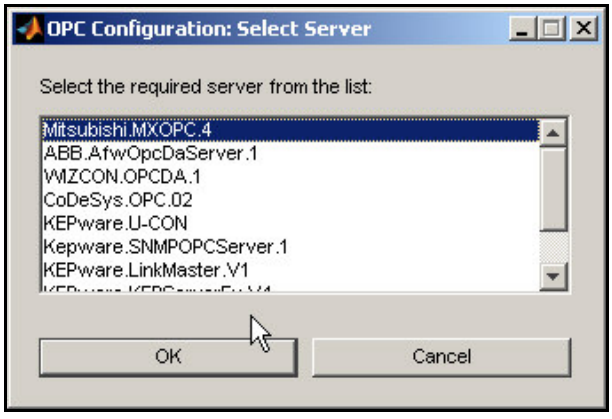
Artık OPC sunucusu istemcilere hizmet verebilecek duruma gelmiştir. Bundan sonra OPC istemci yani MATLAB Simulink ayarları yapılacaktır.[10]

Simulink programında OPC Toolbox kütüphanesinde bu amaçla elemanlar bulunmaktadır. İlk eleman "OPC Configuration" elemanıdır. Bu elemanla Şekil 6.5'teki gibi Simulink ile OPC Sunucu arasında bağlantı kurulur.

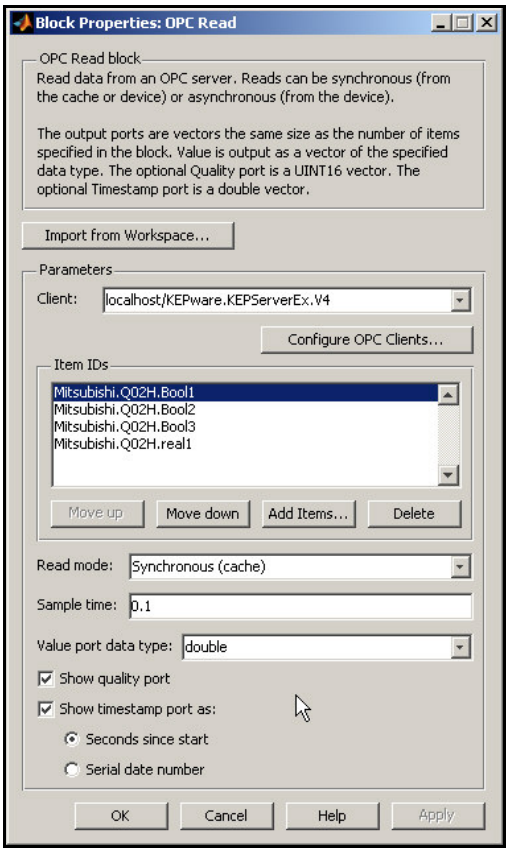


Şekil 6.5 : OPC Configuration elemanı ayar penceresi

Buradan "Add" butonuna basılarak istenilen OPC Sunucu seçilir. (Şekil 6.6)

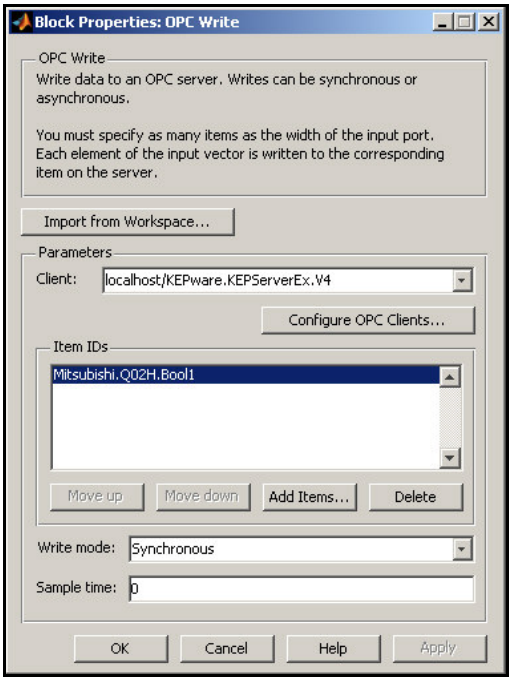


Şekil 6.6 : OPC Sunucu seçimi

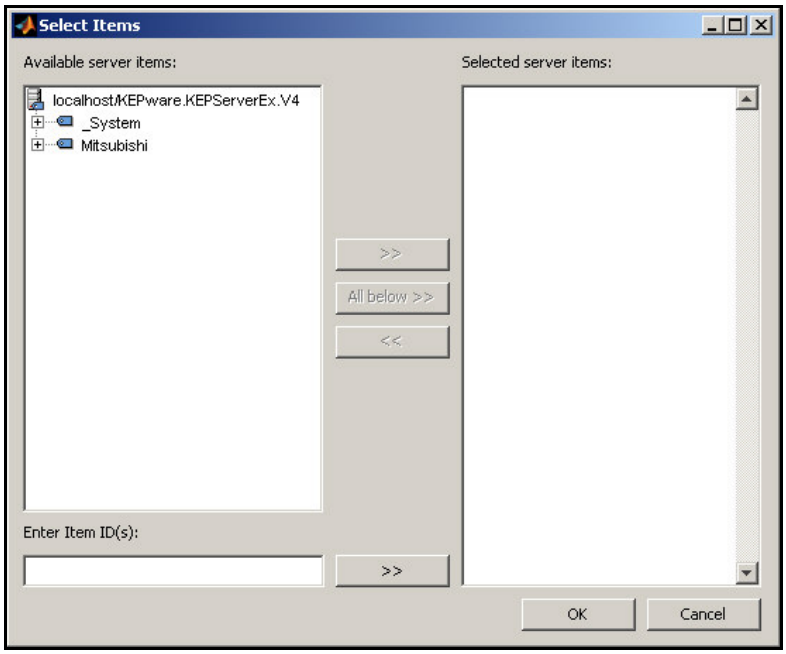


Şekil 6.7 : OPC Read bloğu ayar penceresi

Her iki ayar penceresinden "Add Items"butonu ile okunmak ve yazılmak istenen OPC değişkenleri Şekil 6.9'te gösterilen pencereden seçilir.

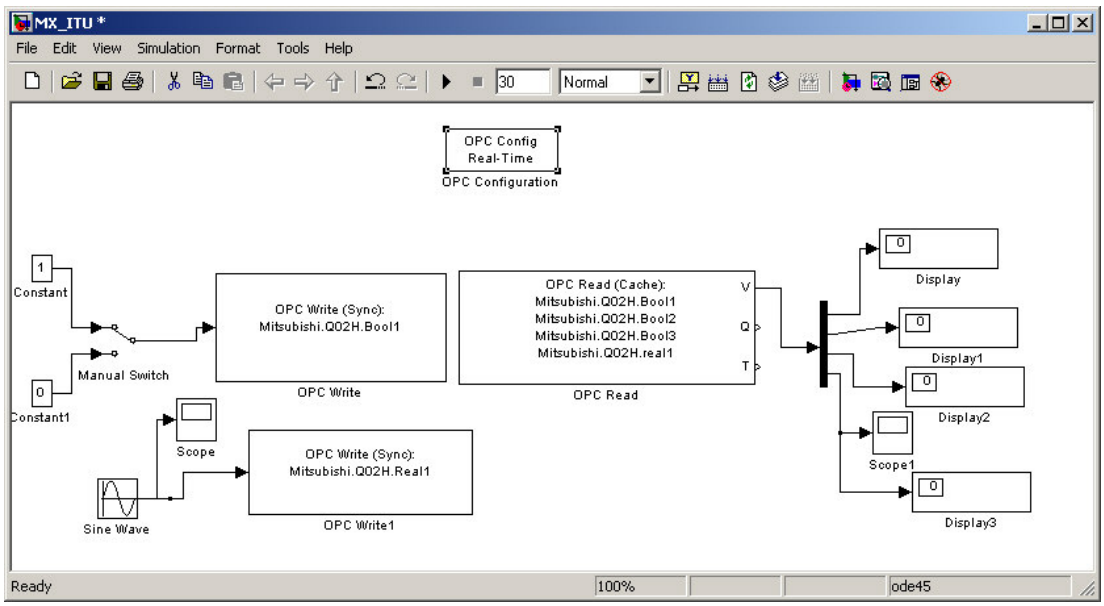


Şekil 6.8 : OPC Write bloğu ayar penceresi



Şekil 6.9 : OPC değişken ekleme penceresi

Artık Simulink üzerinden OPC Sunucu ile veri alış verişi yapmak için tüm ayarlar tamamlanmış olmaktadır. Bu işlemlere ilave olarak standard Simulink bloklarından istenilenleri kullanılarak bir demo modeli hazırlanabilir. Bu çalışma için hazırlanan örnek Simulink modeli Şekil 6.10'da gösterilmiştir.



Şekil 6.10 : Örnek Simulink OPC modeli

Bu model Bool1 adlı değişkene manuel bir anahtar aracılığıyla yazarken Bool1, Bool2, Bool3 ve Real1 adlı OPC değişkenlerini sunucudan okumaktadır. Ayrıca Simulink ortamında üretilen bir Sinüs dalgası Real1 adlı değişkene yazılmaktadır.

7. SONUÇ VE ÖNERİLER

Bu çalışmada endüstriyel otomasyon sistemleri arası haberleşmeler tanıtılmıştır. Özel olarak endüstriyel otomasyon yazılımları arasında kullanılan haberleşme teknikleri detaylıca incelenmiş ve OPC'nin bu alana getirdiği fayda ve katkılar sıralanmıştır.

Uygulama olarak MATLAB Simulink ile Mitsubishi PLC arasında veri alış verişi gerçekleştirilmiştir. Uygulama kısa bir tanıtım olarak tasarlanmış ve basit anlamda Simulink programına canlı bir sistemden nasıl veri alınabileceği ve Simulink'ten canlı sisteme nasıl veri gönderileceği gösterilmiştir.

Bu sayede Simulink üzerinde geliştirilen modellerin simulasyon sonuçları ile gerçek dünyada olan sonuçlarını karşılaştırma imkanı verilmiştir. Buradaki uygulama temel alınarak daha önceden geliştirilen algoritmaların fiziksel ekipmanlar üzerinde denenmesinin ve test edilmesinin önü açılmıştır. OPC'nin standart arayüzü sayesinde haberleşilecek olan donanım ne olursa olsun Simulink üzerinden aynı birim çalıştırılarak istenilen işlevler yerine getirelebilir.

KAYNAKLAR

[1] **Ibn Al-Razzaz Al-Jazari,** 1989. The Book of Knowledge of Ingenious Mechanical Devices Reidel, Pakistan

[2] **ABB Press,** 2001. IEC 61131 Control Languages, ABB Automation Products AB, Sweden

[3] **OPC Overview v1.0,** 1997. Industry Standard Type, OPC Task Force, USA

[4] **Li Zheng, Hiroyuki Nakagawa**, 2002. OPC (OLE for Process Control) Specification and its Developments, SICE 2002 Aug. 5-7 2002. s. 917-920.

[5] **Wu Sitao, Qian Qingquan,** 2000. Using Device Driver Software in SCADA Systems, *IEEE* 0-7803-5935-6/00, **2046**, 2046-2049

[6] **Renee Pattle, Jürgen Ramisch,** 1997. OPC the defacto Standard for Real Time Communicaiton, *IEEE* 0-8186-8096-2/97, **289**, 289-294

[7] **Frank Iwanitz, Jürgen Lange,** 2006. OPC Fundementals, Implementation and Application, Hüthig GmbH & Co., Heidelberg, Germany

[8] **William Robins and Marshall Brain,** 1998. Understanding DCOM, Prentice Hall, 0130959669, USA

[9] **OPC DA v1.0a,** 1997. OPC Specification, OPC Foundation, USA

[10] **The MathWorks Inc.,** 2006. OPC Toolbox For Use with MATLAB and Simulink, The Mathworks Inc., USA

ÖZGEÇMİŞ

Yusuf Ünlü 1980 yılında İstanbul'da doğdu. İlk ve orta eğitimini yine aynı şehirde tamamladıktan sonra 1998 yılında Kocaeli Üniversitesi Elektronik ve Haberleşme Mühendisliği bölümünde lisans eğitimini almaya başladı. Bitirme tezinde "Hibrit Kontrol Sisteminde Çimento Silosu Sevk Sistemi Simulasyon Programı" uygulamasını başarıyla vermesiyle 2003 yılında bu okuldan mezun oldu. 2004 yılında İstanbul Teknik Üniversitesi Kontrol ve Otomasyon Mühendisliği yüksek lisans programına başlayarak eğitimine devam etti. 2002 yılından bu yana ASP Otomasyon Ltd. şirketinde Kontrol Mühendisi olarak çalışmakta olan Yusuf Ünlü, kontrol sistemleri arasında standart haberleşme ve programlama yöntemleri ile ilgilenmekte ve bu yönde çözümler üretmektedir.