İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

RAYLI ULAŞIM SİSTEMLERİNDE ANKLAŞMAN ALGORİTMASI TASARIMI VE OTOMAT YAKLAŞIMI İLE OTOMATİK KOD ÜRETME

YÜKSEK LİSANS TEZİ

Arcan SONAT

Anabilim Dalı : Kontrol ve Otomasyon Mühendisliği

Programı : Kontrol ve Otomasyon Mühendisliği

HAZİRAN 2010

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

RAYLI ULAŞIM SİSTEMLERİNDE ANKLAŞMAN ALGORİTMASI TASARIMI VE OTOMAT YAKLAŞIMI İLE OTOMATİK KOD ÜRETME

YÜKSEK LİSANS TEZİ
Arcan SONAT
(504081103)

Tezin Enstitüye Verildiği Tarih : 7 Mayıs 2010
Tezin Savunulduğu Tarih : 10 Haziran 2010

Tez Danışmanı : Doç. Dr. M. Turan SÖYLEMEZ (İTÜ)
Diğer Jüri Üyeleri : Doç. Dr. Salman KURTULAN (İTÜ)
Yrd. Doç. Dr. D. Turgay ALTILAR (İTÜ)

HAZİRAN 2010

ÖNSÖZ

Çalışmalarım boyunca değerli yardım ve katkılarıyla beni yönlendiren, her aşamada tecrübelerinden faydalandığım değerli hocam ve danışmanım olan Doç. Dr. Mehmet Turan SÖYLEMEZ'e, tez boyunca birlikte çalıştığım arkadaşım Serhat TÜRK'e ve mesai arkadaşım Ahmet Kuzu'ya katkılarından dolayı sonsuz teşekkürü bir borç bilirim.

Yaklaşık 8 aydır Ulusal Demiryolu Sinyalizasyon Projesi (UDSP) kapsamında çalışmakta olduğum TÜBİTAK UEKAE'de anlayışları ve destekleri için başta UDSP müdürü Ayşen Daloğlu Petan olmak üzere tüm mesai arkadaşlarıma teşekkürlerimi sunarım.

Son olarak bütün hayatım boyunca hem maddi hem manevi tüm desteklerinden dolayı ailem sonsuz teşekkürlerimi sunarım.

Haziran 2010

Arcan SONAT

(Elektrik Mühendisi)

İÇİNDEKİLER

Sayfa

ÖNSÖZ ..... **iii**
İÇİNDEKİLER ..... **v**
KISALTMALAR ..... **vii**
ÇİZELGE LİSTESİ ..... **ix**
ŞEKİL LİSTESİ ..... **xi**
ÖZET ..... **xiii**
SUMMARY ..... **xv**
1. GİRİŞ **1**
2. RAYLI ULAŞIM SİSTEMLERİNDE SİNYALİZASYON **3**
2.1 Sinyalizasyon Sistemlerinin Tarihçesi 3
2.2 Sinyalizasyon Sisteminin Temel Öğeleri 5
2.2.1 Saha elemanları 5
2.2.1.1 Ray devreleri 5
2.2.1.2 Makaslar 8
2.2.1.3 Sinyaller 9
2.2.1.4 Hemzemin geçit 9
2.2.2 Trafik Kontrol Merkezi 10
2.2.3 Anklaşman Sistemi 11
2.2.3.1 AS'nin görevi 11
2.2.3.2 Hatada güvenli kavramı 12
2.2.3.3 Güzergahın tanzim edilmesi 13
3. AYRIK OLAY SİSTEMLERİ VE OTOMATLAR **15**
3.1 Ayrık Olay Sistemleri 15
3.2 Model 16
3.3 Durum 17
3.4 Olay 17
3.5 Ayrık Olay Sistemleri Dil Modelleri 17
3.6 Dil 17
3.7 Otomatlar 18
3.7.1 Otomatlarla temsil edilen diller 20
3.7.2 Kilitlenme 20
3.8 Gerçekleme 22
3.8.1 Standart gerçekleme yöntemi 22
3.8.2 Çığ etkisi 23
3.8.3 Durum geçişlerinin matematiksel denklem ile gerçeklenmesi 24
3.8.4 İlk durumun gerçeklenmesi 28
3.8.5 Başlangıç durumuna kurmada ortaya çıkan sorunlar 29
3.8.6 İlk durumun sorunsuz gerçeklenmesi için önerilen yöntem 32
4. OTOMATİK OTOMAT KODU ÜRETME **37**
4.1 Program Açıklamaları 37
5. ANKLAŞMAN ALGORİTMALARI İÇİN OTOMATİK DEĞİŞKEN İSİMLERİ OLUŞTURMA **41**

5.1 Program Açıklamaları 41
6. ÖRNEK BİR İSTASYON HATTININ OTOMATLAR KULLANILARAK ANKLAŞMAN ALGORİTMASI TASARIMI 45
6.1 Giriş 45
6.2 Sinyalizasyonu Yapılan Örnek İstasyonun Hattının Tanıtılması 45
6.3 Sinyalizasyonu Yapılan Hat İçin Anklaşman Tablosunun Oluşturulması 48
6.4 Algoritma Tasarımının Modüler Bir Tasarım Metoduyla Gerçekleştirilmesi.. 53
6.4.1 Jenerik tanzim algoritması 55
6.4.2 Tanzim giriş anahtarlama algoritması 59
6.4.3 Tanzim çıkış anahtarlama algoritması 62
6.4.4 Seçici algoritması 64
6.4.5 Makas algoritması 67
6.4.6 Sinyal algoritması 69
7. SİMULATÖR TASARIMI 73
7.1 Modbus Haberleşmesi 73
7.2 UDP 74
7.3 Programın Tanıtılması 74
8. SONUÇ VE ÖNERİLER 79
KAYNAKLAR 81
ÖZGEÇMİŞ 85

KISALTMALAR

AOS : Ayrık Olay Sistemi
AS : Anklaşman Sistemi
DES : Discrete Event Systems (Ayrık Olay Sistemleri)
DNS : Domain Name System
FBD : Function Block Diagram (Fonksiyon Blok Diyagramı)
IEEE : Institute of Electrical and Electronics Engineers (Elektrik ve Elektronik Mühendisleri Enstitüsü)
KM : Kontrol Merkezi
PC : Personal Computer (Kişisel Bilgisayar)
PLC : Programmable Logic Controller (Programlanabilir Mantıksal Denetleyici)
SFC : Sequential Function Chart (Ardışık Fonksiyon Şeması)
SIL : Safety Integrity Level
SNMP : Simple Network Management Protocol
TCDD : Türkiye Cumhuriyeti Devlet Demiryolları
TCP : Transmission Control Protocol (İletim Kontrol Protokolü)
TÇAO : Tanzim Çıkış Anahtarlama Otomatı
TFTP : Trivial File Transfer Protocol
TGAO : Tanzim Giriş Anahtarlama Otomatı
TKM : Trafik Kumanda Merkezi
UDP : User Datagram Protocol (Kullanıcı Veri Bloğu İletişim Kuralları)
UDSP : Ulusal Demiryolu Sinyalizasyon Projesi

ÇİZELGE LİSTESİ

Sayfa

Çizelge 6.1 : İstasyon hattında bulunan ray bölgeleri..... **47**
Çizelge 6.2 : İstasyon hattında bulunan makaslar. **47**
Çizelge 6.3 : İstasyon hattında bulunan sinyaller. **48**
Çizelge 6.4 : Örnek istasyon için oluşturulmuş anklaşman tablosu (rota 1-rota 14). **51**
Çizelge 6.5 : Örnek istasyon için oluşturulmuş anklaşman tablosu (rota 15-rota 20). **52**
Çizelge 6.6 : Tanzim otomatına ait PLC kodu. **58**
Çizelge 6.7 : Tanzim otomatına ait durumlarda verilecek çıkışlar. **59**
Çizelge 6.8 : Seçici otomatına ait PLC kodu. **66**
Çizelge 6.9 : Seçici otomatına ait durumlarda verilecek çıkışlar. **66**
Çizelge 6.10 : Makas otomatına ait PLC kodu. **69**
Çizelge 6.11 : Makas otomatına ait durumlarda verilecek çıkışlar. **69**
Çizelge 6.12 : 2D sinyal tablosu. **70**

ŞEKİL LİSTESİ

Sayfa

Şekil 2.1 : Basit bir DC ray devresi. **6**
Şekil 2.2 : Kodlu ray devresi. **7**
Şekil 2.3 : Aks sayıcılı ray devresi (Söyler ve Açıkbaş, 2005). **7**
Şekil 2.4 : Hareketli blok sinyal şeması. **8**
Şekil 2.5 : 3'lü cüce, 3'lü yüksek, 4'lü yüksek sinyaller. **9**
Şekil 2.6 : Trafik kontrol merkezi (Durmuş ve Söylemez, 2009). **10**
Şekil 2.7 : KM, AS ve saha arasındaki bilgi alışverişi. **11**
Şekil 2.8 : Hatada güvenli kavramına örnek (Goddard, 2008). **12**
Şekil 2.9 : Güzergahın tanzim edilmesi. **14**
Şekil 3.1 : Zamana bağlı sürekli sistemlerle (a), ayrık olay sistemlerinin (b) durum uzayındaki davranışlarının karşılaştırılması. **16**
Şekil 3.2 : Durum geçiş diyagramı. **18**
Şekil 3.3 : Kilitlenmeli otomat. **22**
Şekil 3.4 : (a) Örnek bir otomat, (b) FBD ile yazılmış PLC programı. **23**
Şekil 3.5 : (a) Çığ etkisi olan otomat, (b) FBD'de yazılmış PLC kodu. **24**
Şekil 3.6 : Örnek 3.2'ye ait FBD ile yazılmış PLC programı. **26**
Şekil 3.7 : Şekil 3.4'teki otomata ait çığ etkisi olmayan PLC programı. **27**
Şekil 3.8 : Örnek 3.2'ye ait FBD ile yazılmış PLC programı. **29**
Şekil 3.9 : Örnek bir otomat. **30**
Şekil 3.10 : Örnek 3.3'te verilen otomatın FBD ile yazılmış PLC programı. **30**
Şekil 3.11 : Başlangıç durumundan çıkartan geçiş (e1) sağlandığında. **31**
Şekil 3.12 : Bir sonraki çevrimde e2 geçişinin sağlanması. **31**
Şekil 3.13 : Başlangıç durumuna geçerken hiçbir durumda bulunmama. **32**
Şekil 3.14 : Örnek 3.3'te verilen otomata ait sorunsuz PLC programı. **33**
Şekil 3.15 : Örnek 3.3'te verilen otomatın sorunsuz PLC programı. **34**
Şekil 4.1 : Geliştirilen programın iş akış şeması. **38**
Şekil 4.2 : Geliştirilen programa otomatın durum ve geçişlerinin girilmesi. **39**
Şekil 4.3 : Geliştirilen program ile otomatik olarak elde edilen PLC kodu. **40**
Şekil 5.1 : İlgili alanlara sayıların yazılması. **41**
Şekil 5.2 : İlgili alanlara saha elemanları isimlerinin girilmesi. **42**
Şekil 5.3 : İstenilen yere dosyanın oluşturulması. **43**
Şekil 6.1 : İstasyon bölgesi. **46**
Şekil 6.2 : Jenerik tanzim otomat modeli. **55**
Şekil 6.3 : TGAO ile tanzim fonksiyon blokların bağlanması. **60**
Şekil 6.4 : Tanzim giriş anahtarlama otomatı. **60**
Şekil 6.5 : TÇAO ile tanzim fonksiyon bloklarının bağlanması. **63**
Şekil 6.6 : TGAO, tanzim ve TÇAO bloklarının bağlanması. **64**
Şekil 6.7 : Seçici otomatı. **65**
Şekil 6.8 : Makas otomat modeli. **67**
Şekil 7.1 : Simülatör. **75**
Şekil 7.2 : İşletilen rotada sinyalin açılması. **76**

Şekil 7.3 : Trenin kırmızı bildirim veren sinyale kadar ilerlemesi. **..... 77**

RAYLI ULAŞIM SİSTEMLERİNDE ANKLAŞMAN ALGORİTMASI TASARIMI VE OTOMAT YAKLAŞIMI İLE OTOMATİK KOD ÜRETME

ÖZET

Raylı ulaşım sistemlerinde, sistemin emniyetli bir şekilde ve ulaşımın planlanan zamanda gerçekleştirilmesinde başrolü oynayan sinyalizasyon sistemleri büyük önem teşkil etmektedir. Öncelikli amacı kazaların önlenmesi olan sinyalizasyon sistemleri, tramvay ve metro gibi demiryolu taşımacılığında hız, dakiklik ve ekonomik olma bakımından da ön plana çıkmaktadır. Gerek dünyada gerekse ülkemizde meydana gelen kazalar sinyalizasyon sisteminin önemini açıkça göstermektedir. Sinyalizasyon sistemlerinin güvenliğinden sorumlu bileşeni ise anklaşman sistemidir.

Bu tezin temel amacı, anklaşman sistemi için formal yöntemlerden otomatlar kullanılarak modüler bir algoritma tasarımı yapılmasıdır. Bu çalışmada anklaşman algoritması tasarımının adımları detaylı bir biçimde anlatılmaktadır. Ele alınan örnek bir istasyon için oluşturulan anklaşman tablosunu gerçekleyecek olan anklaşman algoritması, bu istasyon için geliştirilen bir yazılım simülatörü vasıtası ile test edilmiştir. Tezde, otomat modelinden Programlanabilir Lojik Kontrolör (PLC) kodu elde edilirken, literatürde "çığ etkisi" olarak bilinen sorun da engellenmektedir. Bu sorunu engellemek için bir matematiksel denklem tanıtılmaktadır. Ayrıca başlangıç durumuna kurmada, otomatların tek bir çevrim için birden fazla durumda olması veya hiçbir durumda olmaması tespit edilmektedir. Bu sorunu önlemek için yeni bir yöntem önerilmektedir. Bu çalışmada, otomatik otomat kodu oluşturulması ve PLC programı için otomatik değişken isimleri üretilmesi için Microsoft Visual Basic ortamında yazılımlar geliştirilmiştir. Bu yazılımlar sayesinde, farklı istasyonların anklaşman algoritması tasarımlarının daha kolay ve hızlı bir şekilde gerçekleştirilmesi sağlanmaktadır.

INTERLOCKING ALGORITHM DESIGN IN RAILWAY TRANSPORTATION SYSTEMS AND AUTOMATED CODE GENERATION WITH AUTOMATA APPROACH

SUMMARY

Signalization systems playing vital role in realizing safe railway systems and punctual transportation have significant importance in railway transportation systems. Signalization systems primarily aiming to prevent the accidents come to prominence in railway transportation such as tramway and subway with regard to speed, punctuality and economy. Accidents occurred in both our country and world obviously indicate the importance of signalization systems. Interlocking system is the component of signalization systems responsible from safety.

The fundamental objective of this thesis is to design a modular algorithm for interlocking systems using automata. The steps of designing interlocking algorithm are explained in detail. Interlocking algorithm that realizes the control table generated for a sample station layout is tested by the virtue of a software simulator developed specially for this station. In this work, while Programmable Logic Controller (PLC) code is obtained from automaton model, the issue known as avalanche effect in literature is prevented. A mathematical equation is introduced in order to avoid this problem. Furthermore, as setting automata to initial state, being in more than one state or not even one state for a single cycle is detected. A new method is suggested to eliminate this matter. In this study, programs are developed on Microsoft Visual Basic in order to form PLC code from automaton model and generate variable names for Silworx PLC program automatically. Interlocking algorithm of different station layouts can be achieved fast and easy via the proposed programs.

1. GİRİŞ

Raylı ulaşım sistemleri teknolojilerinin gelişiminin başlangıç hedefi kazaları önlemektir. Bu amaca bağlı olarak sinyalizasyon sistemleri gelişim göstermiştir. Raylı ulaşım sistemlerinde güvenlik; yolcu ve işletme açısından son derece önemli bir yere sahiptir. Ülkemizde ve yurtdışında gerçekleşen demiryolu kazaları ile bunu çok net bir şekilde görmekteyiz. Trafik ve ulaşım sorunlarını engellemek amacıyla dünyanın birçok ülkesinde raylı ulaşım sistemlerine önem verilmiş ve yatırımlar yapılmıştır.

Anklaşman sistemleri, raylı ulaşım sinyalizasyon sistemlerinin güvenlik katmanını oluşturan bileşenidir (Calame, 2006). Bu sistemlerin davranışları, zamanda anlık olarak gerçekleşen olaylara bağlı olarak ifade edilmektedir. Bu anlık olaylara; trenin ray devresine girmesi, sinyallerin renk değiştirmesi, veri gönderilmesi, makas hareketinin tamamlanması, makinenin hataya düşmesi örnek olarak verilebilir. Bu anlık olayların gerçekleşmesi sisteme ait durumların değişmesine yol açar. Dinamik sistemlerin bu sınıfı *Ayrık Olay Sistemleri* olarak adlandırılmaktadır.

Son yıllarda ayrık olay sistemleri oldukça dikkat çekmiş olup, karmaşık yapılı sistemlerin doğru ve güvenli çalışmalarını sağlamaları sebebiyle çeşitli araştırmalara konu olmuştur (Lai, 2008). En çok bilinen ayrık olay sistemlerini modelleme biçimlerinden biri de otomatlardır (Türk S. ve Sonat A., 2010). Otomatları kullanarak çeşitli teorik yaklaşımlar Boel ve van Schuppen (Boel ve Schuppen, 2002), Debouk (Debouk, 2000), Hashtrudi Zad (Zad, 2003), Jiang ve Kumar (Jiang ve Kumar, 2004), Lunze ve Schroder (Lunze ve Schroder, 2004), Sampath (Sampath, 1998) tarafından ortaya konulmuştur. Raylı sistemlerde sinyalizasyonda otomatlardan sıklıkla faydalanılmıştır (Wang, 2003), (Ingleby ve Mee, 1995), (Yong, 2003).

Bu tez çalışmasında Bölüm 2'de sinyalizasyon sistemlerinin tarihi gelişimi kısaca verilecek, ardından sinyalizasyon sistemlerini oluşturan temel bileşenler tanıtılacaktır. Bölüm 3'te ise ayrık olay sistemlerine değinilecek ve otomat modellerinden PLC kodu elde etmek için bir gerçekleme yöntemi gösterilecek olup, otomatı sorunsuz bir şekilde başlangıç durumuna kurmak için bir yaklaşım

önerilecektir. Bölüm 4'te bu gerçekleme yöntemi sayesinde, Microsoft Visual Basic uygulaması ile PLC'lere otomatik kod üreten bir yazılım tanıtılacaktır. Bölüm 5'te ise yine Visual Basic uygulaması ile anklaşman algoritmaları için geliştirilen Silworx PLC programı için otomatik değişken isimleri üreten bir yazılım tanıtılacaktır. Bölüm 6'da anklaşman algoritması tasarımı için gerekli adımlar gösterilecek ve ele alınan örnek bir istasyon hattı için bu tasarım modüler bir şekilde gerçeklenecektir. Bölüm 7'de bu istasyon hattı için Microsoft Visual C++ programında geliştirilen yazılım simülatörü tanıtılacaktır.

2. RAYLI ULAŞIM SİSTEMLERİNDE SİNYALİZASYON

Demiryolu ulaşım sistemlerinde hız, dakiklik ve emniyet ön planda tutulan kriterlerdir. Ulaşımın planlanan zamanda ve emniyetli sürdürülebilmesini sağlayan sinyalizasyon sistemleri, demiryolu taşımacılığının en önemli parçasıdır. Sinyalizasyon sistemlerinin amaçlarını

- Demiryolu üzerindeki araçlar arasındaki kazaların önlenmesi
- Güzergâhın gerektirdiği şekilde makasların tanzimini ve kilitlemesini yaparak oluşabilecek raydan çıkmaya bağlı kazaların engellenmesi
- Tanzim edilen bir rotanın başka bir tren için de tanzim edilmesinin önlenmesi
- Hemzemin geçitlerinin güvenliğinin sağlanması

olarak sıralayabiliriz (Gündoğdu ve Söyler, 2005).

2.1 Sinyalizasyon Sistemlerinin Tarihçesi

1814 yılında George Stephenson tarafından bulunan ilk buharlı lokomotif işletme için sahaya konmasıyla demiryolu taşımacılığı başlamış oldu. Bununla birlikte bütün çalışmalar demiryolu güzergâhları, lokomotif ve vagon üretimi üzerinde yoğunlaşmıştır (Özdemir, 2000).

Demiryolu ilk gelişmeye başladığı yıllarda ulaşımın hızlı bir şekilde gerçekleştirilmesi için kaza gibi güvenlikle ilgili problemler arka plana atılıyordu. Fazla sayıda hat ve kavşak olmaması, trenlerin az sayıda vagonlardan oluşması ve bundan dolayı trenlerin ağırlıklarının fazla olmaması, tren hızının günümüze oranla yavaş olması gibi sebeplerden ötürü trenlerin kontrolü kolay yapılmakta ve güvenlik problemleri için herhangi bir önlem almak gereksiz görülüyordu. Az sayıda tren sefere çıkartıldığı için yolun açık olduğu varsayılıp, önünde tren olmadığı hesap ediliyordu (Söyler ve Açıkbaş, 2005).

İlerleyen zamanlarda kazalar yaşanmış ve bunlar artık bir problem teşkil etmeye başlamıştır. İlk olarak trenlerin seferleri sırasında önündeki yol hakkında bilgi vermek amacıyla, hat üzerine işaretçi görevliler konmaya başlandı. El veya bayrakla

işaret veren bu görevliler ilk başlarda kazaları önlemekte etkili olsalar da, daha sonraki süreçlerde trenlerin hızlarının artması, trenlere bağlanan vagon sayılarının artması ve bununla birlikte trenlerin ağırlıklarının da artmasıyla, makinistlerin sadece bu işaretçilere uyarak treni durdurmaya çalışması kazaları önlemek için yeterli olmamıştır. Buna çözüm olarak üretilen tehlikeli kesimlerden önce işaretlerin arttırılması fikri de kazaları önlemeye yetmemiştir. Bu işaretlerin veya bayrakların uzak mesafelerden görülememesi tren hareketlerine kısıtlama getirmiş ve hat üzerinde daha fazla sayıda işaretçilerin kullanılmasını gerektirmiştir (Söyler ve Açıkbaş, 2005).

1840'lı yıllara gelindiğinde, zaman aralığı yöntemine geçilmiştir. Bu yönteme göre trenler için belli süreler hesaplanmış ve bu süreler dâhilinde karşılaşma noktalarına varmaları beklenmişti. İlk zamanlarda hız, emniyet, ekonomi bakımından etkili bir yöntem gibi gözükse de bu yöntem de kazaları önlemeye çare olamamıştır. Bu yöntemin dezavantajı; trenlerin diğer trenlerin konumları hakkında haberlerinin olmamasıydı. Sefere çıkan trenin herhangi bir şekilde yolda kalmasından arkadaki trenin haberi olmamakta ve kazalar kaçınılmaz olmaktaydı (Söyler ve Açıkbaş, 2005).

Bu dezavantajından dolayı bu yöntem yerini mesafe aralık yöntemine bırakmıştır. Bu yöntemle demiryolu hattı bloklara ayrılmış ve bu blokların başlarına işaretler konmuştur. Makinistler bu işaretlere bakarak önlerindeki blokların meşgul olup olmadığını anlamaktaydılar (Söyler ve Açıkbaş, 2005).

Telgrafın bulunmasıyla birlikte demiryolu sistemlerine çözüm üretmek amacıyla zil ile telgraf kullanılmaya başlamıştır. Önceki istasyonun sinyal operatörü bir sonraki istasyondaki sinyal operatöründen izin istiyor ve devreye yol veriyordu, öteki memurun izin vermesiyle kendi istasyonundaki sinyal devresinin çalıştırılması gerçekleşiyordu. Fakat bu yöntem de, sonraki istasyondan izin verilmesine rağmen arada bir trenin unutulmasıyla etkisini yitirmiştir (Goddard, 2008).

Mesafe aralık yöntemine geçilmesi sabit hat sinyallerinin keşfinde önemli rol oynamıştır. Bununla birlikte 1900'lü yıllarda, elle çalıştırılan blok sistemi, kontrollü elle çalıştırılan blok sistemi, yarı otomatik blok sistemi, otomatik blok sistemi, mekanik blok sistemi gibi sistemlerin gelişmesine öncülük etmiştir. Böylelikle sinyalizasyon sistemleri hızlı bir şekilde gelişmeye devam etmiştir (Söyler ve Açıkbaş, 2005).

Günümüzde sinyalizasyon sistemlerini ele aldığımızda, trenlerin makinistlere bile ihtiyaç duymadan otomatik olarak sürülebilir dereceye gelerek çok fazla gelişme kaydettiğini görmekteyiz.

2.2 Sinyalizasyon Sisteminin Temel Öğeleri

Sinyalizasyon sistemleri, ulaşımın güvenli bir biçimde olmasını ve önceden planlanan zamanda gerçekleştirilmesini amaçlayan demiryolu sistemlerinin en önemli parçasını teşkil etmektedir. Sinyalizasyon sistemlerinin asıl amacı güvenliği en üst seviyede tutarak demiryolu üzerindeki araçlar arasındaki kazaları önlemektir.

Sinyalizasyon sistemleri genel olarak

1- Saha Elemanları
2- Trafik Kontrol Merkezi
3- Anklaşman Sistemi

olmak üzere 3 temel başlık altında incelenebilir. Saha elemanları; ray devrelerinden, makaslardan, sinyallerden ve hemzemin geçitlerden oluşmaktadır.

2.2.1 Saha elemanları

Saha elemanları genel olarak

- Ray devreleri
- Makaslar
- Sinyaller
- Hemzemin geçitler

olarak gösterilebilinir.

2.2.1.1 Ray devreleri

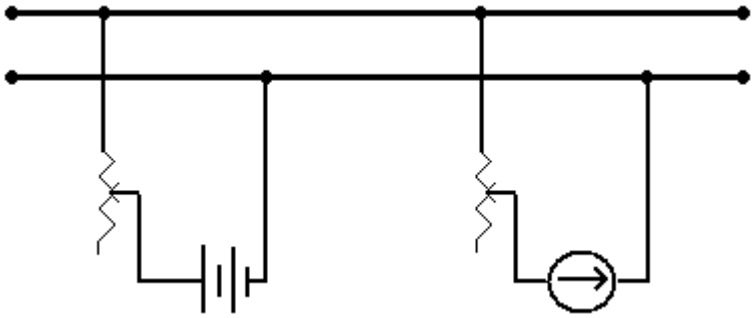
Trenlerin konumlarının tespit edilmesinde kullanılan saha elemanlarıdır. Trenlerin konumları değişik tiplerdeki ray devreleriyle anlaşılabilir. Bunlar

1- İzole cebireli ray devreleri
2- Kodlu ray devreleri
3- Aks sayıcılı ray devreleri
4- Hareketli ray devreleri

şeklinde sıralanabilir.

İzole cebireli ray devreleri:

Ray hattı belli bölgelere ayrılır ve bu bölgelerin izole cebireler ile birbirinden elektriki olarak yalıtılması sağlanır. Bu ray bölgelerinin bir ucundan besleme gerilimi verilerek diğer uçtan bu gerilim kontrol edilir. Bu uygulanan gerilim diğer uçtan alınıyorsa bu ray bölgesinde tren yok, alınamıyorsa bölgede tren var anlamına gelmektedir. Tren bölgeye girdiğinde aksları sayesinde iki ray arasını kısa devre yapar ve bir uçtan uygulanan gerilim diğer uçtan ölçülemez. Burada trenin konumunun algılanması hatada güvenli olarak gerçekleştirilir. Kısa devre, ekipman bozulması, kablo kopması, elektrik kesintisi gibi durumlarda yine aynı şekilde gerilim diğer taraftan alınamayacağı için bölgede tren var kabul edilip, bölge kilitlenecektir. Sistemde herhangi bir arıza olsa dahi en güvenli duruma geçer ve kazaların önlenmesi sağlanır (Söyler ve Açıkbaş, 2005).

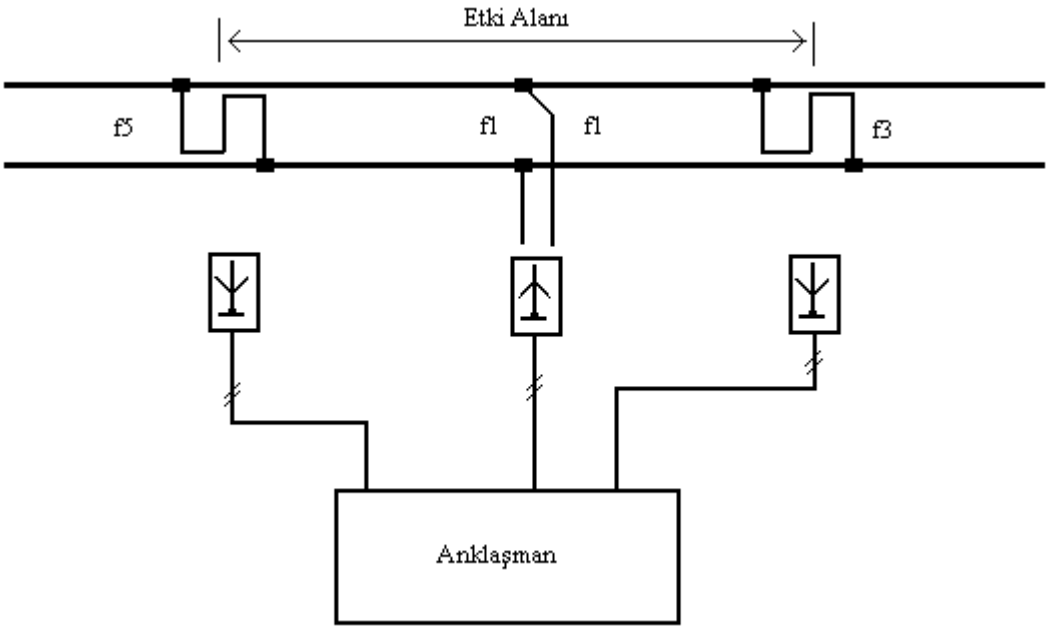


Şekil 2.1 : Basit bir DC ray devresi.

İzole cebireli ray devreleri; İstanbul LRT hattı, İzmir Metrosu, TCDD banliyö ve şehirlerarası hatlarda kullanılmaktadır.

Kodlu ray devreleri:

Kodlu ray devrelerinde raylar izole cebireler yerine kapasitif ayırıcılarla ayrılır. Ray bölgesinin bir tarafında verici diğer tarafında ise alıcı bulunmaktadır. Verici ile bir uçtan raya uygulanan ses frekansı diğer taraftan ölçülür (Şekil 2.2). Frekansta sapma yoksa bölgede tren yok, sapmanın olması durumunda ise bölgede tren var kabul edilir. Burada da hatada güvenli mantığına göre bölgedeki trenin varlığının kontrolü yapılmaktadır.



Şekil 2.2 : Kodlu ray devresi.

Bu ray devresinde izole cebireler olmadığından dolayı yolculuk konforu artar ve bakım maliyeti düşer. Kodlu ray devreleri, Ankaray raylı sistemlerde ve Taksim 4 Levent İstanbul Metrosu'nda kullanılmaktadır (Söyler ve Açıkbaş, 2005).

Aks sayıcılı ray devreleri:

Bu ray devresinde trenin varlığı trende bulunan akslar sayılarak anlaşılmaktadır. Ray bölgesine giren aks sayısı ray bölgesinden çıkan aks sayısına eşitse bölgede tren yok, eşit değilse bölgede tren var olarak kabul edilir.

Bu yöntemin dezavantajı ise ray devresi kırığını tespit edememektir. Sadece bölgeye girip çıkan akslar sayıldığından dolayı, bölgede oluşacak herhangi bir ray kırığı bu yöntemle anlaşılamamaktadır.

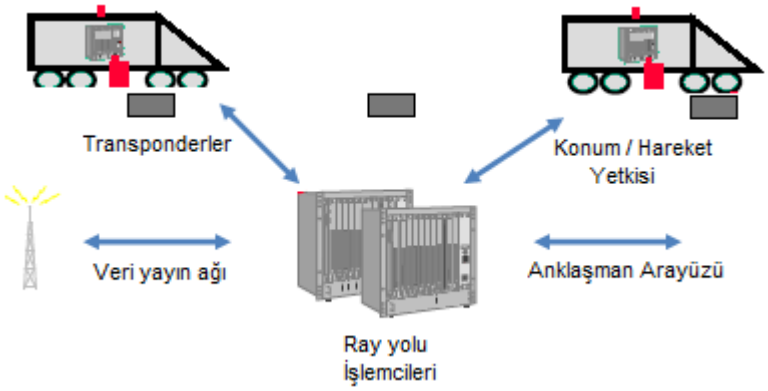


Şekil 2.3 : Aks sayıcılı ray devresi (Söyler ve Açıkbaş, 2005).

Bu ray devresinde de izole cebireler olmadığından dolayı yolculuk konforludur. Aks sayıcılı ray devresi Bursaray hattında görülmekte ve ayrıca dünyada şehirlerarası hatlarda bilhassa hızlı bir şekilde yaygınlaşmaktadır (Söyler ve Açıkbaş, 2005).

Hareketli blok ray devreleri:

Bu yöntemde ray devreleri sabit olmayıp, trenin hızına, frenleme gücüne, bölgenin eğikliğine ve eğimine göre sürekli değişmektedir. Trenlerin bulundukları ray devreleri bu bilgilere göre kontrol merkezindeki program sayesinde hesaplanmakta ve buna göre ray devresi gereksiz yere uzun tutulmadığından dolayı hattın kapasitesi artmaktadır. Ekonomik olabilmesi için daha çok 1,5 dakikanın altındaki trenler arasındaki mesafelerde (headway) kullanılması tercih edilir.



Şekil 2.4 : Hareketli blok sinyal şeması.

Hareketli ray devreleri Ankara Metrosunda kullanılmıştır (Söyler ve Açıkbaş, 2005).

2.2.1.2 Makaslar

İstasyon ve saydinglerde bir yoldan diğer bir yola geçişi sağlayan yol tesisidir. Trenlerin manevra yapabilmesi makaslar yardımı ile gerçekleşir. Makasın bulunduğu ray devresinde araç olduğunda, hatada güvenli mantığına göre makas araç bölgeyi boşaltana kadar komut almaz (Söyler ve Açıkbaş, 2005).

Makas motorları kalkış anında yüksek akım çektiklerinden dolayı aynı güç kaynağına bağlı bulunan makasların aynı anda çalıştırılmaya başlanması tercih edilmeyen bir durumdur. Bundan dolayı aynı güç kaynağına bağlı makaslar belli bir sırada çalıştırılır ve böylece motorun aşırı akım çekerek yanması engellenebilir.

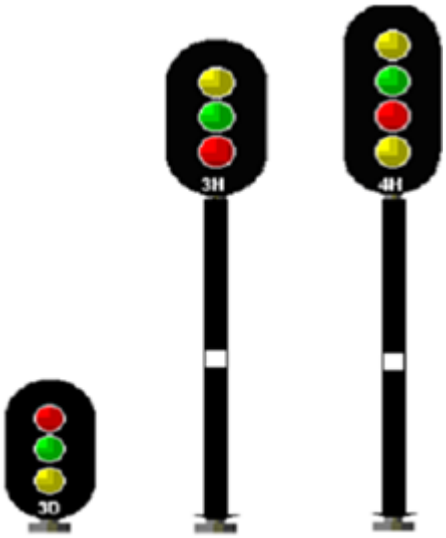
2.2.1.3 Sinyaller

Üzerlerinde iki, üç veya dört renkli lambası bulunan çelik boru, konsol veya köprüler üzerine yerleştirilmiş, çeşitli renk bildirileri vererek demiryolu trafiğini düzenleyen saha elemanlarıdır. Blokların başlangıcında, demiryolu trafiği hakkında makinisti bilgilendiren ve trenlerin güvenli bir şekilde ilerlemelerini sağlayan ışıklardır. Kırmızı dur anlamına gelmektedir. Sarı, önündeki bir blok boş izin verilen hızda devam et, yeşil ise önündeki en az iki blok boş devam et anlamına gelmektedir.

Sinyaller görünüşleri bakımından

1- Yüksek sinyaller
2- Cüce sinyaller

olmak üzere ikiye ayrılır. Yüksek sinyaller 3 veya 4 lambalı olabilmekte ve anayol üzerine yerleştirilmektedirler. Cüce sinyaller ise yan yol üzerlerine yerleştirilirler. Aşağıdaki Şekil 2.5'de sırasıyla 3'lü cüce, 3'lü yüksek ve 4'lü yüksek sinyaller gösterilmiştir. 3'lü cüce sinyal makaslardan sapılarak manevra yapılacağını bildirir. 3'lü yüksek sinyal genellikle manevranın olmadığı anayol çıkışlarında kullanılır. 4'lü yüksek sinyal ise genellikle istasyon girişlerinde veya manevranın yapılacağı yerlerde tercih edilir.



Şekil 2.5 : 3'lü cüce, 3'lü yüksek, 4'lü yüksek sinyaller.

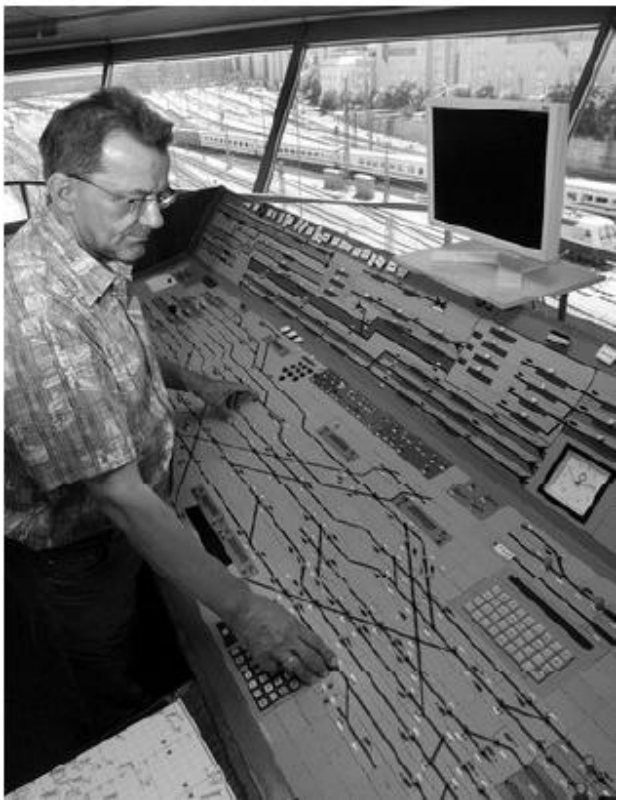
2.2.1.4 Hemzemin geçit

Karayolu ile demiryolunun kesiştiği, alt veya üst geçidi olmayan yerler hemzemin geçit olarak adlandırılmaktadır. Hemzemin geçitlerin kontrolü ile trenin bu

bölgelerden güvenli bir şekilde geçmesi sağlanmaktadır. Hemzemin geçitlerin kendilerine ait bir anklaşman sistemleri olabildiği gibi genel anklaşman içinde de değerlendirilebilirler. Kendilerine ait anklaşman sistemleri olduğunda, genel anklaşman sisteminden deaktif sinyali almalarıyla beraber bariyer kolu inmeye başlar ve trafik karayoluna kapanır. Deaktif sinyal kesilene kadar hemzemin geçit karayolu trafiği için kapalı konumda kalır.

2.2.2 Trafik Kontrol Merkezi

KM, sorumluluk alanındaki demiryolu bölgesinde gerçekleşen tüm demiryolu trafiğinin izlenmesi ve yönetilmesi ile görevli olan donanım ve yazılımlar bütünüdür. KM'nin doğrudan sahayla bağlantısı yoktur. Sahayla ilgili taleplerini anklaşmana iletir, sahaya ait durum bilgilerini ise yine anklaşman üzerinden alır.



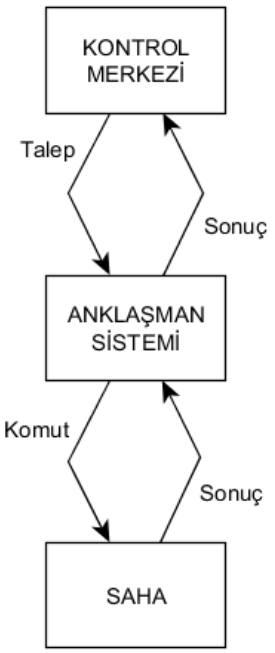
Şekil 2.6 : Trafik kontrol merkezi (Durmuş ve Söylemez, 2009).

KM, periyodik olarak saha bilgilerini AS üzerinden okur. Sistemin kullanıcısı olan dispeçerlere sorumluluk alanındaki sahaya ait durum bilgisi görme ve sistem üzerinde işlem yaparak talep gönderme olanağı sağlar. Dispeçerler taleplerini bu yazılım üzerinden yapar ve talebin AS'ye ulaştırılması sağlanır.

2.2.3 Anklaşman Sistemi

Anklaşman kavramı, sinyalizasyon sistemiyle beraber sistemde bulunan anklaşman ekipmanlarının kilitlenmesi, hareketlerinin birbirlerine bağlı olması, birinin hareketinin diğerlerinin hareketlerini etkilemesi, biri hareketini tamamlamadan diğerinin hareketine izin verilmemesi olarak tanımlanabilir (Gündoğdu ve Söyler, 2008).

AS, yazılım ve donanım olmak üzere iki temel bileşenden oluşur. AS yazılımı demiryolu sinyalizasyon sisteminin kontrolünü sağlayan yazılımdır. Bu yazılımda, KM'den gelen talepleri sahadan aldığı verilerle birlikte değerlendirerek KM'ye taleplerle ilgili cevaplar gönderilir. AS donanımı ise algoritmaların koştuğu hatada güvenli PLC'ler, haberleşme için kullanılan PLC'ler ve dağıtık giriş-çıkış bileşenlerinden oluşur. KM, AS ve saha arasındaki bilgi alışverişi Şekil 2.7'de gösterilmiştir.



Şekil 2.7 : KM, AS ve saha arasındaki bilgi alışverişi.

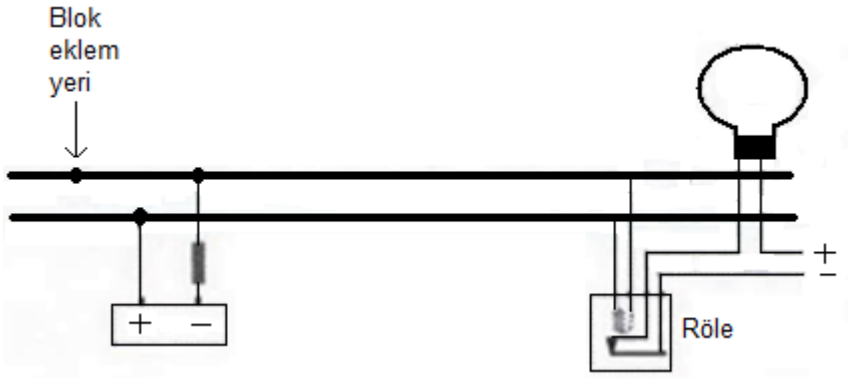
2.2.3.1 AS'nin görevi

AS, demiryolu sinyalizasyon sisteminin kontrolünü sağlayan sistemdir. Güzergâhın tanzimiyle ilgili koşulların güvenli ve uygun olduğu durumda merkezden gelen tanzim talebiyle birlikte AS güzergâhın tanzimine izin verir. Güzergâh tanziminin gerçekleşmesiyle güzergâhın güvenliği ve bütünlüğü AS'nin sorumluluğu altındadır.

İlgili tren için tanzim edilen güzergâhın başka bir tren için de tanzim edilmesi AS sayesinde engellenmektedir. Sinyal ve makaslara birbirleriyle çakışacak şekilde komut gönderilmemesi AS tarafından garanti altına alınır. Güzergâhın tanzim edilmesiyle birlikte AS tarafından ilgili makas ve sinyal kilitlemeleri yapılır ve aynı güzergâh için başka bir tanzim talebinin olması durumunda bu kilitlemeler kontrol edilerek, tanzim talebi AS tarafından reddedilir. Böylelikle iki farklı trenin aynı anda aynı ray bölgesine girmelerine izin verilecek şekilde sinyallerin açılması ve kafa kafaya gelecek biçimde makasların ayarlanması kesinlikle mümkün değildir.

2.2.3.2 Hatada güvenli kavramı

Sistemin çalışması esnasında sistemin tümünde veya tek bir bileşende hata oluşması durumunda, sistemin güvenli duruma ulaşması olarak açıklanabilir. Sistem o durum için en güvenli duruma geçerek sistemin bütünlüğü korunmuş olur ve hatanın etkisi en aza indirgenmiş olur. Sistemlerin tasarım aşamasında sistemin güvenliğiyle ilgili tüm fonksiyonlar çıkarılarak, her fonksiyon için hatada güvenli durumlar belirlenmelidir (Gündoğdu ve Söyler, 2008).



Şekil 2.8 : Hatada güvenli kavramına örnek (Goddard, 2008).

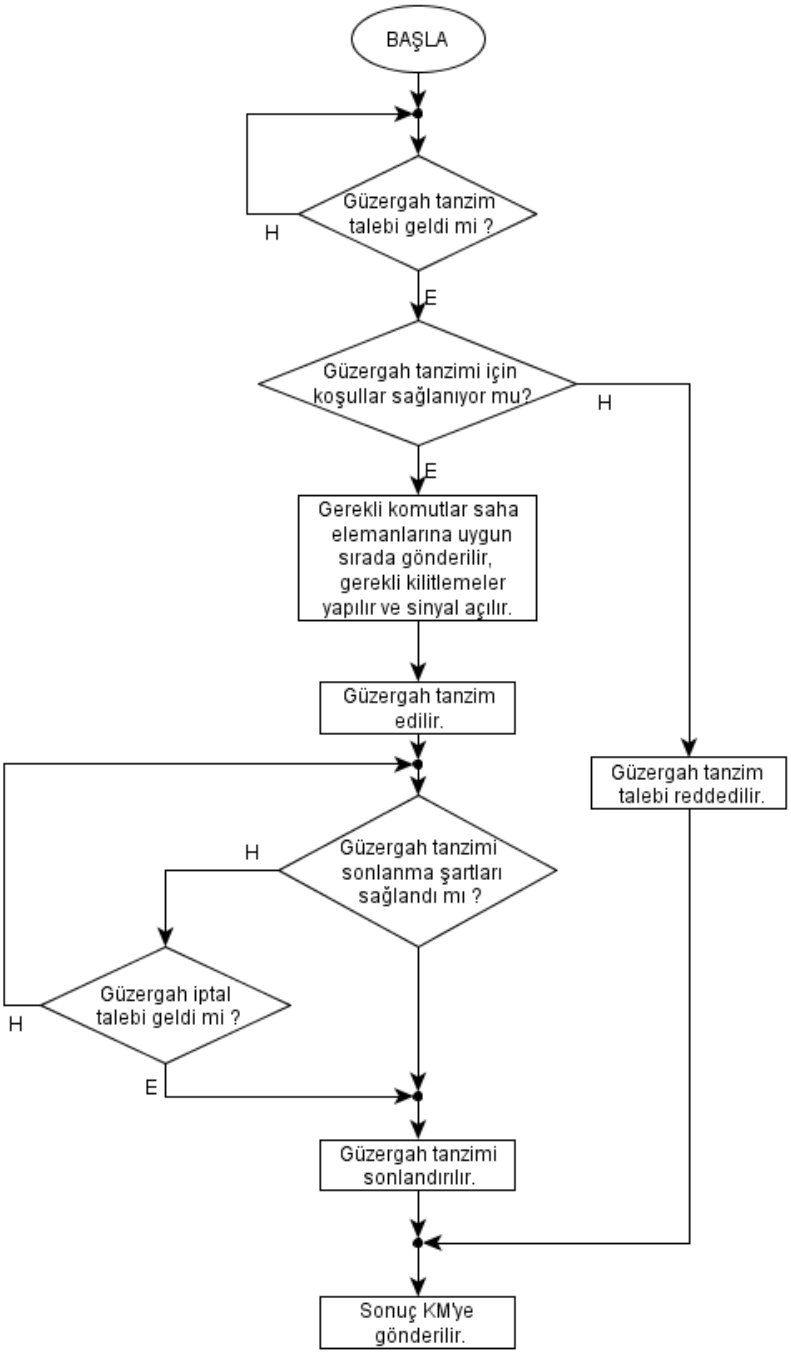
Şekil 2.8'de gösterilen devre aynı zamanda sinyalizasyonda temel tasarım prensibi olan hatada güvenli kavramını temsil etmektedir. Şekilde görülen lamba, güç kaynağından gerilim uygulandığı sürece yeşil yanar. Bunun anlamı güç kaynağından röleye kadar olan blokta uygulanan gerilimi engelleyici herhangi bir sorunun olmamasıdır. Bloğa herhangi bir demiryolu taşıtı girdiğinde ise bu bölgede kısa devre meydana gelir ve röle kapanarak lambanın kırmızı yanması sağlanır. Güç kaynağı arızası, kablo kopması, lamba patlaması gibi durumlarda da yeşil ışık yanmaz ve bölgede tren olduğu kabul edilerek bölge kilitlenir.

2.2.3.3 Güzergahın tanzim edilmesi

AS'nin bir güzergâhın tanzimine izin vermesi, bir takım durumları kontrol etmesi ve bunların uygun olduğunu tespit etmesi sonucu gerçekleşir. Bu süreci genelleştirilmiş hatlarıyla biraz daha detaylı ve sıralı bir şekilde açıklamak istersek,

1. Dispeçer güzergâh tanzim talebinde bulunur.
2. İlgili güzergâhta bulunan makaslar kilitli değildir, makas huzmelerine ait ray devreleri serbesttir ve bu ray devreleri üzerinde kurulu veya kurulmakta olan herhangi bir güzergâh yoktur.
3. İlgili güzergâh kilitlemesi yapılır.
4. Makaslar güzergâhın gerektirdiği konuma geçerler ve mekanik olarak kilitlenirler.
5. Güzergâhta bulunan ray devrelerinin son kez serbest olduğu kontrol edilir ve sinyalin de ilgili güzergâh için açılmasıyla birlikte, güzergâh tanzim edilmiş olur.

Bir güzergâhın tanzimi için gereken şartlar ve yapılan kontroller temel olarak bu şekildedir. Tanzim edilen güzergâhta trenin sırasıyla ray devrelerinde geçişiyle birlikte, ilgili kilitler çözülür ve diğer trenler için tanzim edilebilir duruma gelir. Aşağıdaki Şekil 2.9'da bu sürecin akış şeması verilmiştir.



Şekil 2.9 : Güzergahın tanzim edilmesi.

3. AYRIK OLAY SİSTEMLERİ VE OTOMATLAR

Bilgisayar ve haberleşme uygulama alanlarındaki önemli derecedeki gelişmeler yeni bir dinamik sistem türünü tetiklemiştir. Dijital bilgisayarların çalışma prensiplerinden diferansiyel denklemlerin nümerik çözümlerine, hava trafik kontrol sistemlerinden demiryolu sinyalizasyon ve anklaşman sistemlerine kadar pek çok alanda bu tür sistemlerin varlığından söz edebiliriz. Zamanda anlık olarak oluşan olaylara bağlı olarak bu tür sistemlerin davranışları ifade edilebilir. Bu olayların gerçekleşmesiyle birlikte sistem bir durumdan diğerine geçmektedir. Bu özelliklere sahip dinamik sistem sınıfı Ayrık Olay Sistemleri (AOS) olarak isimlendirilmektedir (Hasdemir, 2008).

Ayrık olay sistemleriyle ilgili yapılmış akademik çalışmalar literatürde oldukça fazladır. Özellikle bu sistemlerin kontrolüyle ilgili olan Üstdenetim Kuramı son zamanlarda oldukça ilgi görmüş ve bu teoriyle ilgili birçok çalışmalar yayınlanmıştır (Ramadge ve Wonham, 1987), (Chen ve Lafortune, 1991).

3.1 Ayrık Olay Sistemleri

Diferansiyel denklemler, analiz ve tasarımda sistem ve kontrol teorileri için çıkış noktalarını oluşturmaktadırlar. Sistemin dinamiği serbest değişken olan zamana bağlı olarak ifade edilebilmektedir. Bir sistemin durum uzayı, durumların olabilecek tüm değerlerinin oluşturduğu küme olup, Q ile gösterilmektedir. Sisteme uygulanan giriş u olmak üzere durum denklemi,

$$\dot{q}(t) = f(q(t), u(t), t) \quad \textbf{(3.1)}$$

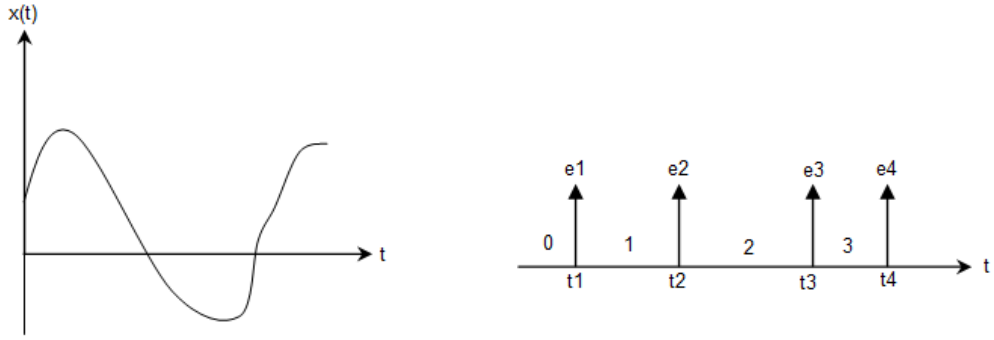
şeklinde verilebilir. Bu denklem ayrık zamanda

$$q(k+1) = f(q(k), u(k), t) \quad \textbf{(3.2)}$$

Şeklindeki bir diferansiyel denkleme denk gelmektedir.

Ayrık olay sistemlerinde ise durum uzayı Q={0,1,2, .. } gibi ayrık bir kümeden oluşur ve durum geçişleri zamana bağlı olarak değil, zamanın ayrık noktalarında gözlemlenen olaylarla gerçekleşmektedir. Şekil 3.1'de durum uzayı Q={0,1,2,3}

olan ayrık olay sistemi ile zamana bağlı olarak değişen sürekli zamanlı bir sistemin olaylara ve zamana bağlı durumlarının davranışlarını karşılaştıran örnek gösterilmektedir.



(a) (b)

Şekil 3.1 : Zamana bağlı sürekli sistemlerle (a), ayrık olay sistemlerinin (b) durum uzayındaki davranışlarının karşılaştırılması.

Şekil 3.1b'de görüldüğü üzere ayrık olay sistemlerinde durumların değişimleri olaylarla gerçekleşmektedir. Sistemin durum uzayındaki yeri sadece olaylarla belirlendiğinden dolayı bu sistemlerin davranışı için $\dot{q}(t) = f(q(t), u(t), t)$ gibi zamana bağlı ifade verilemez (Cassandras ve Lafortune, 1999).

3.2 Model

Sistem davranışını göz önünde bulundurarak buna karşılık gelen bir model sezgisel olarak düşünülerek, var olan sistemin bir modeli bulunmaya çalışılır. Sistemin davranışını anlayabilmek için geliştirilen matematiksel yöntemlerle o sisteme ait model elde edilmeye çalışılır. Gerçek olan sistemin modeli, matematiksel denklemlerden oluşan bir soyutlamadır.

Verilen bir sistemin her zaman bir modelini elde etmek mümkünken, matematiksel denklemleri verilen bir modelden her zaman gerçek bir sistem elde edemeyiz. Bunun nedeni olarak matematiksel denklemlerin her zaman gerçek sonuçlar vermemesini söyleyebiliriz. Bundan dolayı oluşturulan model genellikle sistemin gerçek davranışına sadece yaklaşık bir davranış gösterir (Cassandras ve Lafortune, 1999).

3.3 Durum

Bir sistemin t_0 anındaki durumu, tek başına u(t) ($t \geq t_0$) bilgisinden, y(t) ($t \geq t_0$) anındaki değerlerini belirlemeye yeten ve gereken bilgidir. Durum değişkenleri,

$$q(t) = [\, q_1, q_2, .. , q_n]T \quad \textbf{(3.3)}$$

olarak tanımlanır (Cassandras ve Lafortune, 1999).

3.4 Olay

Bir durumdan diğer bir duruma geçişi sağlayıp, sadece anlık olarak oluşur. Bir olay, butona basılması, bir veri paketinin gönderilmesi ya da makinenin arızaya geçmesi gibi ortaya çıkan belirli etkiler üzerinden ifade edilebilir. Eğer sistem birçok olaydan etkileniyorsa ayrık olaylar kümesi tanımlanabilir (Cassandras ve Lafortune, 1999).

3.5 Ayrık Olay Sistemleri Dil Modelleri

Ayrık olay sistemlerinin mantıksal davranışları üzerinde çalışmanın biçimsel bir yolu da diller ve otomat teorilerine dayanmaktadır. Her ayrık olay sistemleri ile ilişkilendirebileceğimiz bir olay kümesi vardır. Bu kümenin sonlu olduğu kabul edilir. Olay kümesi dilin "alfabe"si, olay dizileri ise dildeki "kelime"ler olarak düşünülebilir. Herhangi bir olay içermeyen kelime "boş kelime" olarak adlandırılmaktadır ve "ε" ile gösterilir. Bir kelimenin uzunluğu ise o kelime içindeki harf yani olay sayısına eşittir, aynı olayın tekrarları da bu sayıya dahil edilir ve $|\delta|$ ile ifade edilir. Boş kelimenin uzunluğu sıfıra eşittir (Cassandras ve Lafortune, 1999).

3.6 Dil

E olay kümesi üzerinde tanımlıdır ve bu olay kümesine ait sonlu uzunluktaki kelimelerden oluşur, L ile ifade edilir.

$E = \{e_1, e_2, e_3\}$ olay kümesine ait birden fazla dil tanımlanabilir.

$$L_1 = \{\varepsilon, e_1, e_1e_3e_3\} \quad \textbf{(3.4)}$$

$$L_2 = \{e_1 \text{ ile başlayan ve uzunluğu üç olan tüm kelimeler}\} \quad \textbf{(3.5)}$$

Kelime ve dolayısıyla dil oluşturmadaki en önemli işlem "türetme" işlemidir. L1 dilindeki $e_1e_3e_3$ kelimesi e_1e_3 kelimesi ve e_3 olayından türetilmiştir, e_1e_3 kelimesi ise e_1 ve e_3 olaylarından türetilmiştir (Cassandras ve Lafortune, 1999).

3.7 Otomatlar

Bir otomat, iyi tanımlanmış kurallara göre bir dili temsil etmekten sorumlu araçtır. Otomat kavramını ifade etmenin en kolay yolu durum geçiş diyagramlarını göz önünde bulundurmaktır (Cassandras ve Lafortune, 1999).

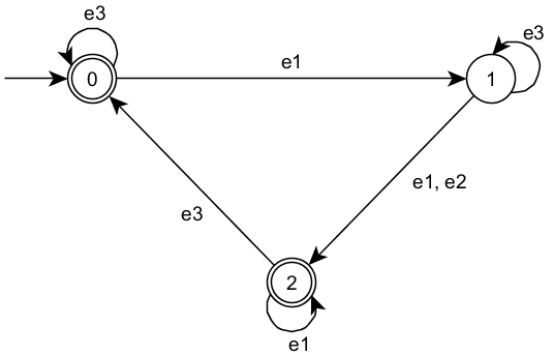
Şekil 3.2'deki daireler durumları, oklar ise bu durumlar arasındaki geçişleri ifade eder. Durumlar ve geçişlerden oluşan bu şekil bir otomata ait dinamiklerin tanımlanmasını sağlamaktadır. Bir otomatın durum kümesi dairelerin kümesine eşittir, Q={0,1,2}. Geçiş etiketleri, otomatın E geçiş kümesinin (alfabe) elemanlarıdır, E={e1,e2,e3}. Modeldeki oklar ise otomatın f:Q×E→Q şeklinde ifade edilen geçiş fonksiyonunun grafiksel gösterimini sağlar (Cassandras ve Lafortune, 1999).

$f(0,e1)=1$ $\quad$ $f(0,e3)=0$

$f(1,e1)= f(1,e2)=2$ $\quad$ $f(1,e3)=1$

$f(2,e1)=2$ $\quad$ $f(2,e3)=0$ **(3.6)**

$f(2,e3)=0$ notasyonu, eğer otomat 2 durumunda ise a olayının oluşmasıyla otomat 0 durumuna geçiş yapacağını gösterir. Bu e3 olayının oluşumunun sebebi önemsizdir. Bu olay, otomat ile modellenen sisteme dışarıdan gelen bir giriş olabileceği gibi, sistem tarafından kendiliğinden üretilen bir olay da olabilir



Şekil 3.2 : Durum geçiş diyagramı.

Şekil 3.2'den görüldüğü üzere $f(0,e3)=0$ gibi durum değişikliğine neden olmayan olayların meydana gelebildiği görülebilmektedir. Ayrıca belirli bir durumda farklı olayların oluşmasıyla aynı duruma geçiş yapıldığı şekilden anlaşılmaktadır, $f(1,e1)=f(1,e2)=2$. Otomat, 1 durumundan 2 durumuna geçiş yaptığında bunun hangi geçiş sayesinde olduğu fark edilemez. Geçiş fonksiyonu olan f, Q×E tanım kümesinde kısmi fonksiyondur ve bundan dolayı E geçiş kümesindeki her olay için bir geçiş fonksiyonu tanımlamaya gerek yoktur. Şekil 3.2'de $f(0,e2)$ ve $f(2,e2)$ buna örnek olarak gösterilebilir.

0 durumuna gelen ok ile başlangıç durumu x_0, durum kümesinin işaretli durum alt kümesi olan Q_m ise iç içe geçirilmiş bir çift çemberle gösterilmiştir. Durumlara özel bir anlam yüklenmesi istendiği zamanlarda durumlar işaretlenmektedir. İşaretli durumlar "kabul gören" ya da "son durumlar" olarak adlandırılmaktadırlar (Cassandras ve Lafortune, 1999).

Bir otomat

$$G = \{Q, E, f, \Gamma, q_0, Q_m\} \quad \textbf{(3.7)}$$

şeklinde ifade edilir. Burada;

- Q durum kümesi,
- E sonlu olay kümesi,
- f: Q × E → Q durum geçiş fonksiyonudur. $f(q,e)=q'$, q durumundan q' durumuna e olayı ile bir geçiş etiketi olduğu anlamına gelmektedir.
- Γ: Q→ 2^E aktif olay fonksiyonudur; $\Gamma(q)$, otomat q durumundayken $f(q,e)$'nin tanımlı olduğu bütün e olaylarını içerir ve G'nin q durumundaki aktif olay kümesi olarak adlandırılır.
- q_0 başlangıç durumu,
- $Q_m \subseteq Q$ işaretli durumlar kümesidir.

Yukarıda açıklanan G notasyonu *durum makinesi* olarak da adlandırılır. Q durum kümesi sonlu ise G *sonlu otomat* olarak adlandırılır (Hasdemir, 2008).

Hangi durumun işaretleneceğinin uygun seçimi probleme bağlı modellemeye ilgilidir. İşaretli durumlara gelinmesiyle sistemin belirli görevleri tamamladığı anlamına gelmektedir.

G otomatı şu şekilde çalışır: q_0 ilk durumundan başlar ve e Є $\Gamma(q_0) \subseteq$ E olayının oluşmasıyla $f(x_0,e)$ Є Q durumuna geçiş yapar. Bu süreç f'nin tanımlı olduğu geçişlere bağlı olarak devam eder.

$f(q,\varepsilon) = q$

$f(q,se) = f(f(q,s),e)$ s Є E* , e Є E **(3.8)**

3.7.1 Otomatlarla temsil edilen diller

Otomat ile dil arasındaki bağlantı otomatının durum geçiş diyagramı incelenerek kolayca anlaşılabilir. Durum geçiş diyagramındaki olabilecek tüm geçişlerle elde edilen olay dizileri *üretilen dil*i, işaretli durumlarda sonlananlar ise *işaretlenen dil*i ifade eder (Tiryaki, 2007).

Tanım 3.1 (Bir otomatın ürettiği dil) G = {Q, E, f, Γ, q_0, Q_m } tarafından üretilen dil

$L(G) = \{s \in E^* : f(q_0,s) \text{ tanımlı}\}$ **(3.9)**

şeklinde ifade edilir (Tiryaki Ö., 2007).

Tanım 3.2 (Bir otomatın işaretlediği dil) G = {Q, E, f, Γ, q_0, Q_m } tarafından işaretlenen dil

$L_m(G) = \{s \in L(G) : f(q_0,s) \in Q_m\}$ **(3.10)**

şeklinde ifade edilir (Tiryaki, 2007).

Şekil 3.2'de verilen G otomatındaki üretilen ve işaretlenen diller

L(G) = {ε,e1,e1e3,e1e3e3,…,e1e1,e1e2,e1e3e1,e1e3e2,e1e3e1e3,e3e1e3e2e1e3,…}

L_m(G) = {e3, e3e1e1,e1e1,e1e2,e1e1e1,e1e2e1,e1e1e3,…}

şeklinde olur.

Üretilen dil kümesi, bir otomatın ürettiği dil içindeki bir s kelimesinin tüm ön eklerini kapsadığından dolayı tanım gereği üretilen dil önek kapalı bir dildir, $L(G) = \overline{L(G)}$ (Tiryaki, 2007).

3.7.2 Kilitlenme

Genellikle G, L(G) ve L_m(G) tanımlarından

$L_m(G) \subseteq \overline{L_m(G)} \subseteq$ L(G) **(3.11)**

elde ederiz. İlk kısım Q_m'nin Q'nun alt kümesi olmasından yani tüm durumların işaretli olmadığından kaynaklanır. İkinci kısım ise $L_m(G)$'nin tanımının ve L(G)'nin tanımdan dolayı ön ek kapalı olmasının bir sonucudur (Cassandras ve Lafortune, 1999).

Tanım 3.3 (Ölü Kilit) Bir G otomatı, $\Gamma(q) = \phi$ fakat $q \notin Q_m$ olan bir x durumuna erişilebilir. Buna *ölü kilit* (Deadlock) denir, çünkü başka bir olayla bu durumdan çıkamayarak sistem bu durumda kilitlenir. Ölü kilit gerçekleşirse $\overline{L_m(G)}$, L(G)'nin alt kümesi olacaktır. q durumunda sonlanan herhangi bir L(G) kelimesi, $L_m(G)$'deki kelimelerin ön eki olamaz (Cassandras ve Lafortune, 1999).

Tanım 3.4 (Canlı Kilit) Bir G otomatında işaretli olmayan durumlar kendi aralarında birbirlerine erişebiliyor fakat hiçbir geçişle bu kümenin dışına çıkamayabilirler. Eğer sistem bu durumlardan oluşan kümeye girerse, bunu *canlı kilit* (Livelock) olarak adlandırmaktayız. Otomat canlıdır, oluşan geçişlerle durum değiştirmeye devam eder ama bu küme içinde işaretli durum bulunmadığından ve hiçbir geçişle bu kümenin dışına da çıkamayacağından dolayı otomat başladığı görevi tamamlayamaz. Canlı kilit gerçekleşirse $\overline{L_m(G)}$, L(G)'nin alt kümesi olacaktır (Cassandras ve Lafortune, 1999).

Tanım 3.5 (Kilitlenme) Bir G otomatı için

$$\overline{L_m(G)} \subset \mathrm{L(G)} \qquad \textbf{(3.12)}$$

denklemi sağlanıyorsa kilitlenmeli,

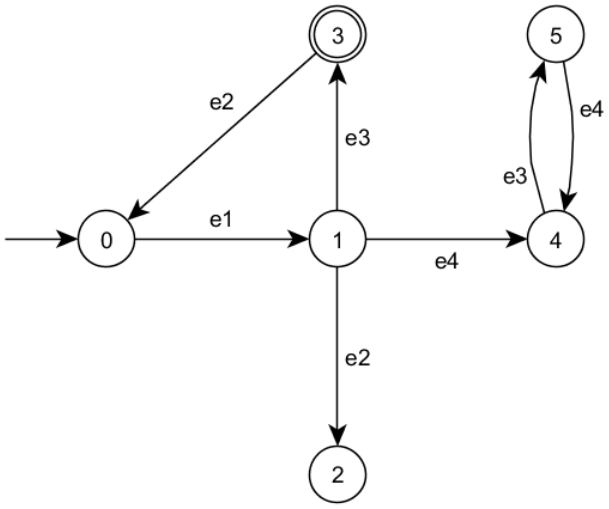
$$\overline{L_m(G)} = \mathrm{L(G)} \qquad \textbf{(3.13)}$$

denklemi sağlanıyorsa kilitlenmesiz otomat denir. Eğer otomat kilitlenmeli ise bu otomatta ölü kilit ya da canlı kilit oluşabileceği anlamı taşımaktadır (Cassandras ve Lafortune, 1999).

Örnek 3.1.

Aşağıdaki Şekil 3.3'te içinde ölü kilit ve canlı kilit bulunan bir sistem gösterilmiştir. Sistemdeki 2. durum ölü kilit durumudur. Otomat bu işaretli olmayan duruma geldiğinde olabilecek hiçbir geçişle bu durumdan çıkamayacağı için bu durumda kilitlenip kalır. 4 ve 5 numaralı durumlar ise canlı kilide sebep olan durumlardır. Otomat 4. duruma gelmişse bundan sonra sadece 4. ve 5. durumlar arasında sınırlı

hareket eder. Yukarıda açıklandığı üzere bu durumlar işaretli olmadıklarından dolayı canlı kilit oluşmuş olur.



Şekil 3.3 : Kilitlenmeli otomat.

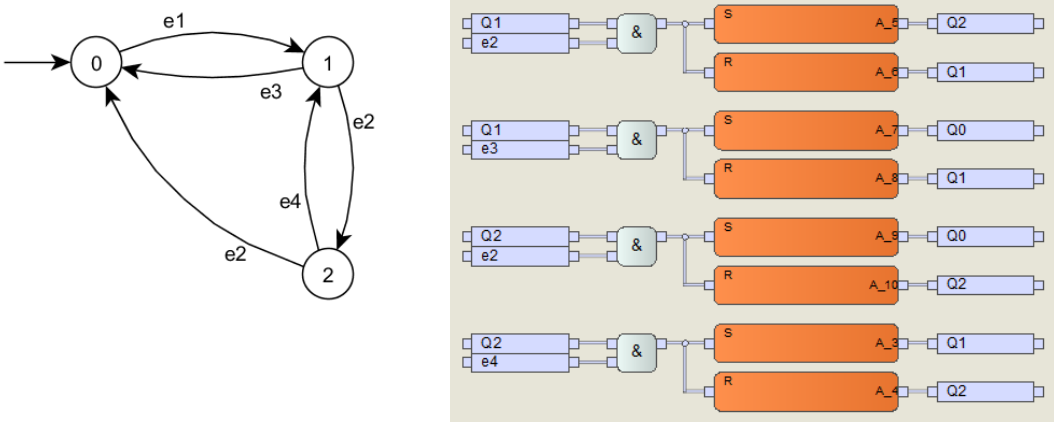
3.8 Gerçekleme

Mikrokontrolör tabanlı teknolojik araçlar olan PLC'lerin çalışma mantığı hafıza alanlarındaki sayısal verilerin çevrimsel olarak işlenmesini temel almaktadır. Bu kısımda "Standart gerçekleme yöntemi" ile kurallar verilecek ve buna göre PLC kodunun oluşturulması gösterilecektir.

3.8.1 Standart gerçekleme yöntemi

Bellek bitleri sayesinde durum ve olaylar temsil edilebilir. Bu bellek bitinin mantıksal 1 seviyesinde olması o durumda olduğu anlamı taşıyacaktır. Ayrık olay sistemlerinde olaylar anlık olarak gerçekleştiğinden dolayı PLC'de bunu karşılık olması bakımından olay işaret seviyelerinin çıkan ya da düşen kenarları kullanılabilir. PLC'nin her tarama çevriminde bu işaret seviyeleri kontrol edilir. Eğer o anki işaret seviyesi 1, bir önceki işaret seviyesi 0 ise bu çıkan kenara denk gelmektedir. Sadece o çevrim için çıkan kenar fonksiyonu çıkış verir. Tam tersi durumda ise yani mantıksal 1'den 0'a geçiş olması durumunda düşen kenar fonksiyonu tek bir çevrim süresinde çıkış verir. Durum geçiş fonksiyonlarında en basit yöntemle mantıksal VE, VEYA kullanılmaktadır. Belirli bir durumu temsil eden bit ile o durumdan çıkartan geçişe mantıksal VE, eğer çıkaran geçişler birden fazlaysa bunlara da kendi aralarında mantıksal VEYA işlemi uygulanır. Ayrıca yeni

durumu temsil eden bitin kurulması için PLC'deki kurma komutu, önceki durumu sıfırlamak için ise PLC'deki sil komutu kullanılır. Şekil 3.4'te örnek bir otomat ve buna karşılık gelen FBD ile yazılmış PLC programı verilmiştir. Şekil 3.4b'de Q1, Q2, Q3 otomat durumlarını temsil eden bitlerdir. e1, e2, e3, e4 ise çıkan kenarlarıyla birlikte olaylara karşılık gelen işaretlerdir. "&" simgesi olan bloklar mantıksal VE, "S" harfi olan bloklar SET, "R" harfi olan bloklar ise RESET işlemine karşılık gelmektedir.



(a) (b)

Şekil 3.4 : (a) Örnek bir otomat, (b) FBD ile yazılmış PLC programı.

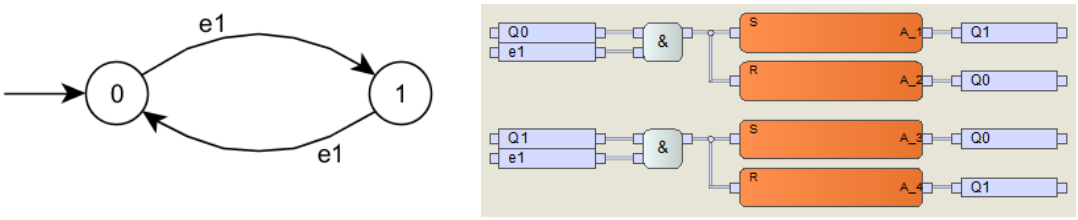
3.8.2 Çığ etkisi

Yukarıda verilen yöntem ile otomattan PLC programına geçişte bazı beklenmedik durumlarla karşılaşabiliriz. Bu istenmeyen durum geçişleri çığ etkisi (avalanche effect) olarak bilinmektedir (Fabian ve Hellgren, 1998). Şekil 3.4a'da verilen otomattan elde edilen PLC gerçeklemesinde çığ etkisi sorunuyla karşılaşmış bulunmaktayız.

Şekil 3.4a'daki otomat e1 geçişiyle birlikte 0 durumundan 1 durumuna geçer. Otomat bu durumdayken e2 geçişiyle beraber 1 durumundan çıkıp 2 durumuna geçmesi ve ardından başka bir e2 geçişiyle 2 durumundan çıkıp 3 durumuna geçmesi beklenir. Fakat Şekil 3.4b'deki merdiven diyagramından görüldüğü üzere Q1 durumunda iken e2 olayının gerçekleşmesiyle Q2 biti mantıksal 1 seviyesine geçecek ve Q1 biti ise mantıksal 0 seviyesine düşecektir. Aynı tarama çevrimi içinde bir alt basamağa geçildiğinde ise Q2 biti mantıksal 1 seviyesinde olduğundan ve ayrıca e2 olayı da

çevrim boyunca çıkış vereceğinden dolayı bu sefer Q0 biti kurulacak ve aynı zamanda Q2 biti de silinecektir. Otomat 1 durumundan e2 geçişiyle birlikte 2 durumunu atlar ve 3 durumuna geçiş yapar. Bundan dolayı Şekil 3.4a'daki otomata karşılık gelen PLC programı çığ etkisi sorunundan ötürü hatalı olarak çalışmaktadır.

Yukarıda anlatılan gerçekleme yöntemi kullanıldığında, bir duruma hem giren hem de çıkan geçişlerin aynı olması her otomatta karşılaşılabilecek çığ etkisi sorununa neden olmaktadır. Literatürde bu soruna çözüm olarak programı yazarken basamak sırasını tersten başlatmaya dayalı öneriler getirilmiştir (Fabian ve Hellgren, 1998). Fakat bu çözüm sistematik olmamakla birlikte programcının deneyimine göre sonuçlar vermektedir. Ayrıca bazı durumlarda ise bu soruna çözüm olmamaktadır. Aşağıda gösterilen Şekil 3.5'te verilen basit bir otomat için PLC programı, basamak sırası ter çevrilerek de yazılsa çığ etkisi sorunu engellenememektedir (Hasdemir, 2008).



(a) (b)

Şekil 3.5 : (a) Çığ etkisi olan otomat, (b) FBD'de yazılmış PLC kodu.

3.8.3 Durum geçişlerinin matematiksel denklem ile gerçeklenmesi

Çığ etkisi sorununu yukarıda anlatılan standart gerçekleme yöntemi adını verdiğimiz yöntemle sistematik bir şekilde engellemek imkânsızdır (Hasdemir, 2008). Ayrıca kurma ve silme komutları normal bit işlemlerine göre PLC hafızasında daha fazla yer kaplamaktadır. Bunların yanında durumlara ilişkin geçiş PLC programındaki takip de zorlaşmaktadır. Şekil 3.4b'de sadece 1 durumuna ait 2 adet kurma ve 2 adet de silme komutları kullanılmıştır. Bu 4 komutu ilgilendiren şartlar birlikte değerlendirilerek bu duruma ait geçiş fonksiyonları saptanır. Bu ise hata bulma veya programda değişiklik yapılması gereken zamanlarda bir hayli güçlük çekmemize sebep olur.

Tezin ilerleyen bölümlerinde; PLC programında kurma ve silme komutlarının yerine bunlara oranla çok daha az yer kaplayan bit işlem komutları kullanılarak, çığ etkisi sorununu kesin olarak engelleyen, değişikliklere daha rahat izin veren, kolay takibi yapılarak hata bulmayı kolaylaştıran matematiksel bir yöntem tanıtılacaktır.

Durumları temsil eden q değişkeninin kurma koşulu S, silme koşulu ise R ile gösterelim. Bu q değişkeninin şimdiki değeri q(m), bir sonraki değeri ise q(m+1) ve $m \in N$ olmak üzere,

$$q(m+1) = S \vee q(m) \wedge \overline{R} \tag{3.14}$$

mantıksal fonksiyonunda q durumu S ile kurulurken, R ile silinmektedir. Bu denklemde "∨" mantıksal VEYA, "∧" ise mantıksal VE işlemlerini temsil etmektedir. R'nin üzerinde çizgi olması mantıksal DEĞİL anlamına gelmektedir. Durum geçiş fonksiyonlarının gerçeklenmesinde bu verilen matematiksel denklem kullanılabilir (Hasdemir, 2008).

Tanım 3.6. q_m durumuna ait kurma koşulunun k adımındaki mantıksal ifadesi

$$S_m(k) = \sum_{n \in I_{SQ}(m)} q_n(k) \wedge e(m,n) \tag{3.15}$$

ile verilir. Burada $I_{SQ}(m)$; q_m durumuna geçişe ait ilişkin durum indis kümesini temsil eder.

Tanım 3.7. q_m durumuna ait silme koşulunun mantıksal ifadesi

$$R_m(k) = \sum_{n \in I_{SE}(m)} e_n(k) \tag{3.16}$$

ile verilir. Burada $I_{SE}(m)$; q_m durumuna geçişe neden olan olay indis kümesini göstermektedir.

Bu iki tanım ile verilen kurma ve silme koşulları birleştirildiğinde denklem 3.11'den yola çıkılarak

$$q_m(k+1) = S_m(k) \vee q_m(k) \wedge \overline{R_m(k)} \tag{3.17}$$

Bu denklemde verilen $q_m(k)$; m numaralı durumun şimdiki değerine karşılık gelmektedir ve bunu q_m ile gösterelim. $q_m(k+1)$; m numaralı durumun bir sonraki değeridir ve bunu da Q_m ile gösterelim.

$$Q_m = S_m \vee q_m \wedge \overline{R_m} \tag{3.18}$$

Programın son basamaklarında m numaralı durumun şimdiki değeri olan q_m mantıksal değişkenine, m numaralı durumun bir sonraki değeri olan Q_m mantıksal değişkeni ataması yapılır (Hasdemir, 2008).

Örnek 3.2.

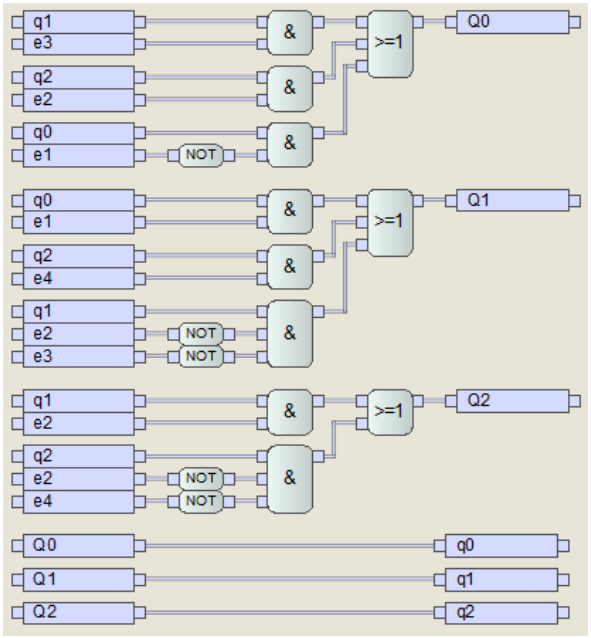
Şekil 3.4a'da verilen otomatın denklem 3.14'te gösterilen yöntemle PLC programını yazalım.

$$Q_0 = (q_1 \wedge e_3) \vee (q_2 \wedge e_2) \vee (q_0 \wedge \overline{e_1}) \quad \textbf{(3.19)}$$

$$Q_1 = (q_0 \wedge e_1) \vee (q_2 \wedge e_4) \vee (q_1 \wedge \overline{e_2} \wedge \overline{e_3}) \quad \textbf{(3.20)}$$

$$Q_2 = (q_1 \wedge e_2) \vee (q_2 \wedge \overline{e_2} \wedge \overline{e_4}) \quad \textbf{(3.21)}$$

Şekil 3.4a'da verilen otomatın durumlarına ait mantıksal ifadeler yukarıda gösterildiği üzere elde edilir. Aşağıda Şekil 3.6'da bu otomata ilişkin FBD ile yazılmış PLC programı gösterilmektedir.



Şekil 3.6 : Örnek 3.2'ye ait FBD ile yazılmış PLC programı.

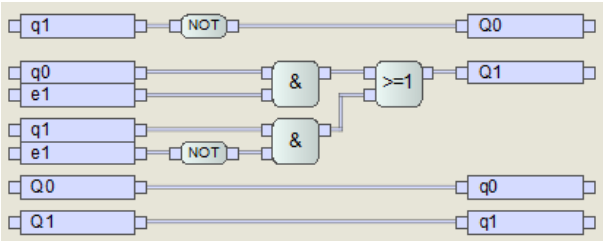
Önceki gerçekleme yönteminde 2 numaralı duruma gelen ve çıkan olayların aynı olmasından kaynaklanan durum atlama sorunu olmaktaydı. 1 durumundayken e2 geçişi sağlandığında otomat 2 numaralı durumu atlayıp 0 numaralı başlangıç durumuna gitmekteydi.

Daha önce anlatılan standart gerçekleme yöntemi ile programlamada oluşan çığ etkisi sorunu bu bölümdeki programlama yöntemi sayesinde ortadan

kaldırılmaktadır. Şekil 3.6'yı dikkatli bir biçimde incelersek, Q_1 ve q_1 bitleri mantıksal 1 seviyesinde iken e2 geçişinin oluşmasıyla birlikte Q_2 bitinin mantıksal 1 seviyesine gelmesini sağlar. Bir alt basamakta Q_0 değişeninin kurulabilmesi için q_2 değişeninin ve e2 geçişinin bu çevrim için mantıksal 1 seviyesinde olmaları gerekmektedir. e2 geçişi sağlanmasına rağmen Q_2 değişeni daha q_2 değişkenine atanmadığı için Q0 durumu bu çevrim süresinde kurulamaz. İki alt basamakta Q_2 değişkeni q_2 değişkenine atanarak q_2 değişkenini de mantıksal 1 seviyesine çeker ve 2 numaralı duruma geçilmiş olur. Böylece 1 numaralı durumda iken e2 geçişinin sağlanmasıyla birlikte 2 numaralı durumu atlamamış oluruz. Ayrıca burada verilen e1, e2, e3 ve e4 geçişleri anlık olarak gerçekleştiği kabul edilerek çıkan kenarlarla ifade edilir. Bir sonraki çevrimde, 0 numaralı duruma geçebilmek için gerekli olan şartlardan q_2 biti mantıksal 1 seviyesinde olmasına rağmen e2 geçişi anlık olduğundan bu çevrimde mantıksal 0 seviyesindedir. Bundan dolayı otomat e2 geçişinin bir sonraki çıkan kenarı veya e4 geçişi gelene kadar 2 numaralı durumda kalır. Böylece çığ etkisinin engellendiği örnek üzerinde de gösterilerek kanıtlanmış olur.

Burada görüldüğü üzere her bir durum için iki farklı değişken kullanılmıştır. Q mevcut PLC tarama çevrimi, q ise bir sonraki tarama çevrimi için kullanılmaktadır. Bu durum bellekte daha fazla yer kaplamasına sebep olabilir. Bunu engellemek için ise sadece giren ve çıkan geçişlerin aynı olduğu durumlar için iki farklı bit kullanmaktır (Hasdemir, 2008).

Şekil 3.4a'da verilen otomat için standart gerçekleme yönteminde basamak sırası ters çevrilerek de programlansa çığ etkisi oluşmaktaydı. Bu bölümde verilen yöntemle gerçeklenen PLC programı Şekil 3.7'de gösterilmiştir.



Şekil 3.7 : Şekil 3.4'teki otomata ait çığ etkisi olmayan PLC programı.

Şekil 3.7'den görüldüğü üzere e1 geçişinin sağlanmasıyla Q_1 değişkeni mantıksal 1 seviyesine gelecek. Bir alt basamakta Q_0 değişkeninin kurulması, 1 numaralı

durumun şimdiki tarama çevrimindeki değeri ile değil bir sonraki ile olduğundan dolayı çevrim sonunda otomat 1 numaralı durumda olur. Böylece çığ etkisi sorunu bu otomat modeli için de geçerliliğini yitirmiş olmaktadır.

3.8.4 İlk durumun gerçeklenmesi

Yukarıdaki yöntemlerde PLC'nin ilk başta herhangi bir duruma kurulmadığı açıkça görülmektedir. Herhangi bir duruma kurulması için gerekli koşullardan biri de, bu duruma getiren diğer durumların kurulu olmasıdır. Hiçbir durumda olmayan otomat geçişlerin tetiklenmesine rağmen durum değiştiremeyecektir (Hasdemir, 2008).

Bundan dolayı tam bir gerçekleme için otomatın başlangıç durumuna programlanma şartı gereklidir. Bunu gerçeklemek için PLC programları tarafından sağlanmış olan sadece ilk tarama çevriminde mantıksal 1 seviyesinde olan özel bellek yerleri mevcuttur. Bu özel biti kullanarak program çalışmaya başladığı anda otomat istenen başlangıç durumuna kurulur (Hasdemir, 2008).

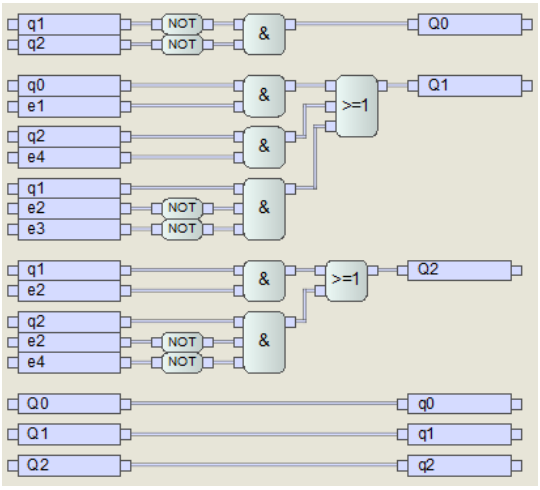
Bundan başka yöntemler de mevcuttur. Otomatı başlangıç durumuna programlamak için "Otomat hiçbir durumda değilse ilk durumdadır" yöntemini kullanabiliriz.

$$q_0(m+1) = \prod_{n \in I_Q \setminus n_0} \overline{q_n(m)} \qquad \textbf{(3.22)}$$

Burada n_0 ilk durumun indis numarasını belirtir. Buradan yola çıkılarak Şekil 3.4a'da verilen otomat için başlangıç durumu

$$Q_0 = \overline{q_1} \wedge \overline{q_2} \qquad \textbf{(3.23)}$$

şeklinde programlanabilir (Hasdemir, 2008). Denklem 3.19 yerine yukarıda verilen denklem 3.23'ü yazarız. 1 ve 2 numaralı durumları gösteren mantıksal fonksiyonlara ilişkin denklem 3.20 ve 3.21 ise aynı şekilde kalır. Buna ait FBD ile yazılmış PLC programı ise Şekil 3.8'de gösterilmiştir.



Şekil 3.8 : Örnek 3.2'ye ait FBD ile yazılmış PLC programı.

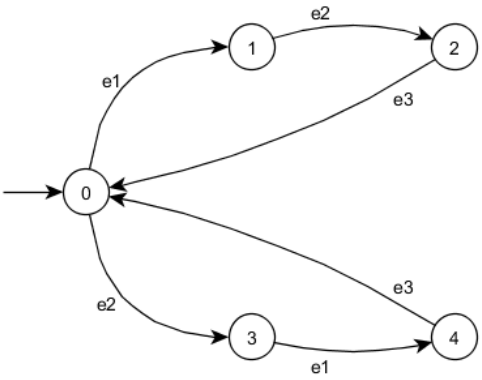
Bu şekilde bir gerçekleme yaparak PLC'nin ilk tarama çevriminde diğer durumlar mantıksal 0 seviyesinde olacağından dolayı 0 numaralı durum kurulacaktır. Denklem 3.19'da verilen mantıksal fonksiyon yerine, denklem 3.23'de gösterilen mantıksal fonksiyonu yazarak hem program hafızasında daha az yer kaplamış, hem de başlangıç durumuna kurma problemini çözmüş olmaktayız.

3.8.5 Başlangıç durumuna kurmada ortaya çıkan sorunlar

Yukarıda matematiksel denklemlerle verilen yöntemle birlikte gösterilen başlangıç durumuna kurma mantığı otomatta bizi bazen istemediğimiz durumlara götürebilir. Şekil 3.8'de verilen programı dikkatli bir biçimde inceleyecek olursak, otomat ilk tarama çevriminde 0 numaralı başlangıç durumuna kurulacaktır. Ardından e1 geçişinin aktif olmasıyla bir çevrim boyunca hem 0 numaralı durumda hem de 1 numaralı durumda kalacaktır. Ancak bir sonraki çevrimde 0 numaralı durum silinecek ve otomat sadece 1 numaralı durumda olacaktır. Fakat otomatın sadece bir çevrim süresi de olsa aynı anda birden fazla durumda birden olması istenmez. Ayrıca otomat 1 numaralı durumda iken e3 geçişinin sağlanması veya 2 numaralı durumda iken e2 geçişinin aktif olmasıyla birlikte 0 numaralı başlangıç durumuna geçişlerde ise bir çevrim boyunca otomat hiçbir durumda olmayacaktır.

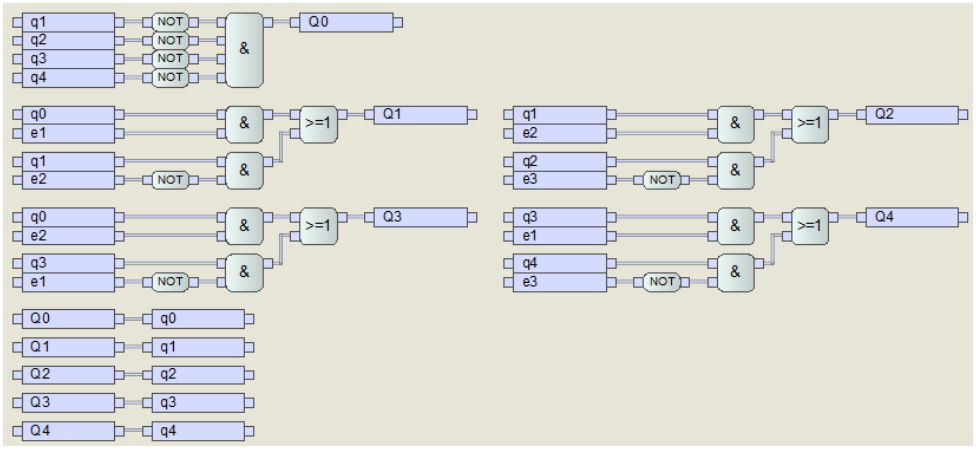
Örnek 3.3

Şekil 3.9'da verilen otomatı aynı çevrimde birden fazla durumun kurulması ve hiçbir durumun kurulmaması yönünden inceleyelim.



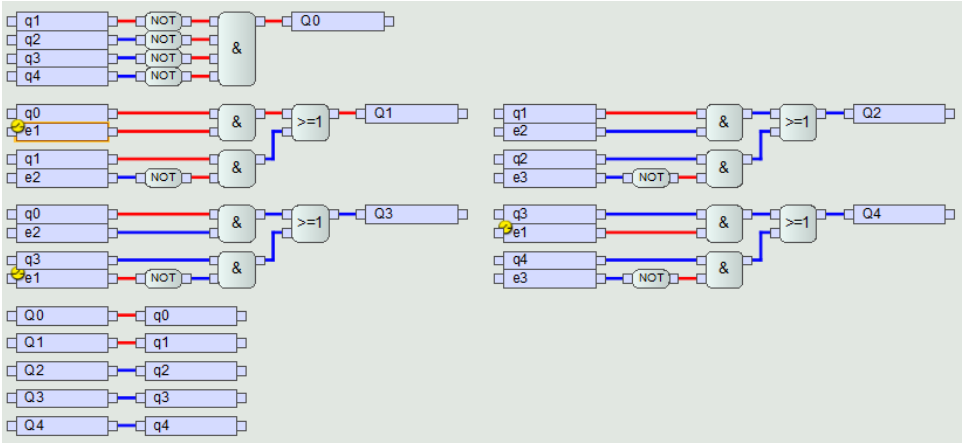
Şekil 3.9 : Örnek bir otomat.

Bu otomatın gerçeklemesi son gösterilen yöntemle yapıldığında Şekil 3.10'da gösterilen PLC programını elde ederiz. Bu şekildeki içinde "NOT" yazan bloklar mantıksal DEĞİL, "&" etiketli bloklar mantıksal VE işlemi, ">=1" etiketi olan bloklar ise mantıksal VEYA işlemine karşılık gelmektedir.



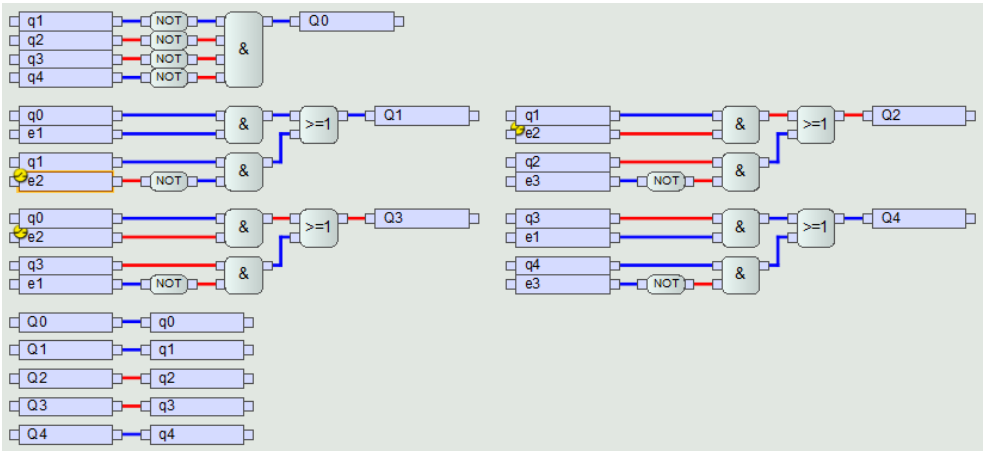
Şekil 3.10 : Örnek 3.3'te verilen otomatın FBD ile yazılmış PLC programı.

Kullandığımız Silworx PLC programında tarama yönü sol yukarıdan sağ aşağıya doğrudur. Buna göre yukarıdaki şekilde verilen programın tarama sırası Q0, Q1, Q2, Q3, Q4 şeklindedir. Bu programı PLC'de çevrim çevrim işlettiğimizde ise aşağıda verilen şekilleri elde ederiz.



Şekil 3.11 : Başlangıç durumundan çıkartan geçiş (e1) sağlandığında.

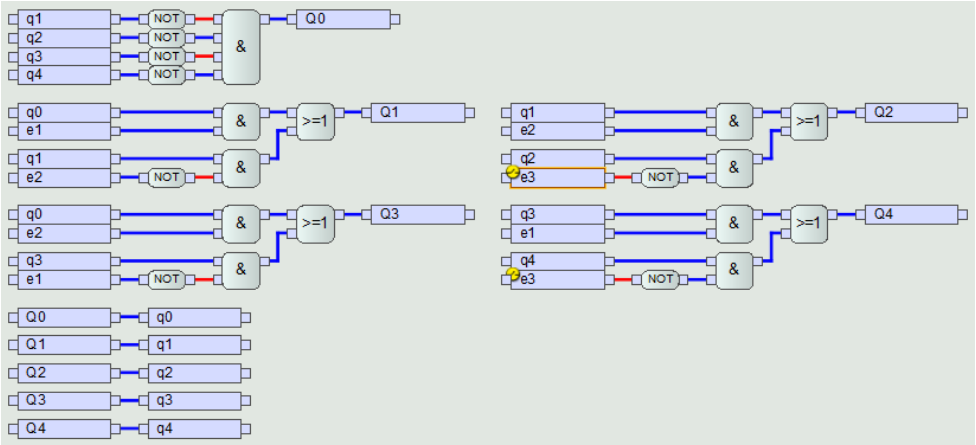
e1 geçişi sağlandığında yukarıdaki şekilden de görüldüğü üzere otomat bir çevrim boyunca hem 0 hem de 1 numaralı durumlarda kalır. Bunun sebebi ise Q_0 bitinin mantıksal 1 seviyesinde olması için gerekli olan diğer durumlarda olmaması koşulu sağlanmaktadır. Ardından hemen alt tarafında q_0 ve e1 bitlerinin mantıksal 1 seviyesinde olmasından dolayı Q_1 durumu da kurulur. Aynı çevrim süresinde alt taraflarda ise Şekil 3.10'dan da görüldüğü üzere q_0 ve q_1 bitleri de mantıksal 1 seviyesine çekilir.



Şekil 3.12 : Bir sonraki çevrimde e2 geçişinin sağlanması.

Otomat hem 0 hem de 1 numaralı durumlarda iken bir sonraki çevrimde e2 geçişinin sağlanmasıyla Şekil 3.10'dan da görüldüğü gibi 2 ve 3 numaralı durumlara geçiş yapılır. Normalde istenen otomatın sadece 2 numaralı durumda olması iken, 0 numaralı durumdan çıkmadığı için 3 numaralı duruma geçiş için de şartlar oluşmuş olur. Böylece bu ilk duruma kurma yöntemi ile istenmeyen durumlara gitmiş oluruz. Ayrıca otomat 2 numaralı durumda iken e3 geçişinin sağlanmasıyla Şekil 3.11'de

görüldüğü gibi bir çevrim boyunca hiçbir durumda olmaz. Bundan dolayı bu çevrimde oluşan bir geçişi de kaçırmış oluruz.



Şekil 3.13 : Başlangıç durumuna geçerken hiçbir durumda bulunmama.

3.8.6 İlk durumun sorunsuz gerçeklenmesi için önerilen yöntem

Bu tezde bu sorunları çözmek için bir yöntem önerilecektir. Bu yöntem sayesinde otomat başlangıç durumundan çıkarken aynı çevrimde birden fazla durumda kalmanın önüne geçilmiş ve başlangıç durumuna dönüşte de hiçbir durumda bulunmama ortadan kaldırılmış olunacaktır.

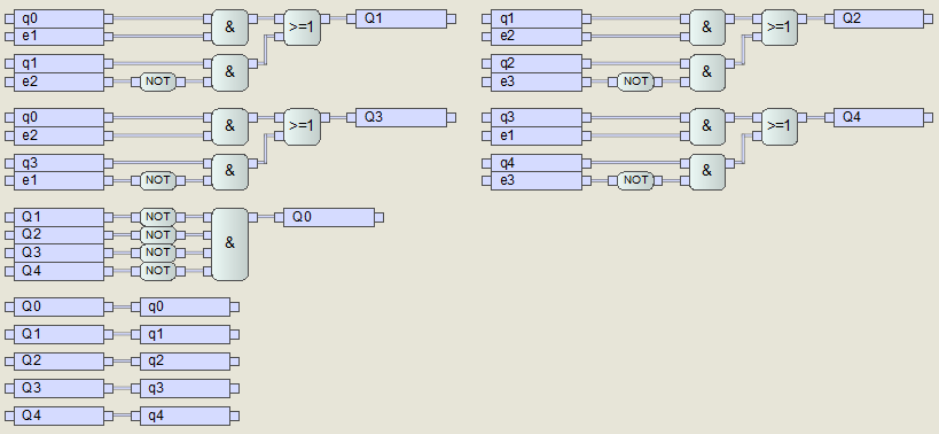
Bu yönteme göre başlangıç durumuna kurmak için hiçbir durumda değilse başlangıç durumuna kur mantığı aşağıda şekilde düzenlenir.

$$q_0(m+1) = \prod_{n \in I_Q \setminus n_0} \overline{q_n(m+1)} \qquad \textbf{(3.24)}$$

Bu denklemde, I_Q; durum indis kümesini, n_0; sayma vektörünün 0. adımdaki değerini temsil etmektedir. Denklem 3.24'te verilen mantıksal fonksiyon, diğer durumların mantıksal fonksiyonları yazıldıktan sonra yazılmalıdır. Ardından durumların mevcut çevrim değerlerinin bir sonraki çevrim değerlerine atanması yapılmalıdır. Şekil 3.9'da verilen otomata ilişkin yukarıda belirtilen sorunları ortadan kaldıracak şekilde oluşturulan PLC programı Şekil 3.14'te gösterilmiştir. Bu programa göre 0 numaralı durumda iken e1 geçişi tetiklendiğinde Q_1 biti mantıksal 1 seviyesine gelir. Ardından Q_0 bu yeni yönteme göre daha alta yazıldığı için Q_1 biti kurulduğundan dolayı 0 numaralı durum silinir. Bir sonraki çevrimde e2 geçişi sağlandığında otomat sadece 1

durumunda olacağından 3 numaralı duruma geçiş engellenerek sadece 2 numaralı duruma geçilir.

Ayrıca 2 numaralı durumda iken e3 geçişinin gelmesiyle Q_2 biti mantıksal 0 seviyesine geçer. Hiçbir durumda olunmadığından dolayı da alt tarafta 0 numaralı durum kurulur. Böylece daha önce bahsedilen iki sorunun da önüne geçildiği görülmüş olur.



Şekil 3.14 : Örnek 3.3'te verilen otomata ait sorunsuz PLC programı.

Yukarıda anlatılan yöntemde bir durumdan çıkışı sağlayan geçişlerin aynı anda mantıksal 1 olamayacağı kabul edilerek otomat tasarlanmıştır. Aksi halde böyle bir durumda istenmedik sonuçlar doğabilir. Örneğin Şekil 3.9'da verilen otomat 0 numaralı durumda iken e1 ve e2 geçişlerinin aynı çevrim içinde mantıksal 1 seviyesine geçmeleriyle otomat hem 1 hem de 3 durumlarına dallanır. Otomatın tasarımı aşamasında bu durumlara dikkat edilmelidir. Bu tarz durumların önüne geçmek için iki yöntem önerilebilir.

Bu yöntemlerden ilki otomatın tasarımı aşamasında aynı anda tetiklenerek otomatı birden fazla duruma geçirebilecek geçişler belirlenerek bu geçişler öncelik sırasına göre programlaması yapılabilir. Örneğin, şekil 3.9'da verilen otomatın başlangıç durumundan çıkmasını sağlayan e1 ve e2 geçişlerinden e1 geçişinin e2 geçişine önceliği olsun. Bu durumda, 0 numaralı durumdan 1 numaralı duruma geçiş aynı şekilde e1 olarak kalır, fakat 0 numaralı durumdan 2 numaralı duruma geçiş için $e2 \wedge \overline{e1}$ yazılması gerekir. e1 ve e2 geçişleri aynı çevrimde mantıksal 1 olduğunda 0 numaralı durumdan 1 numaralı duruma geçiş sadece e1 olduğundan dolayı bu geçiş sağlanır, 0 numaralı durumdan 2 numaralı duruma geçişte ise e1 geçişinin mantıksal DEĞİL'i olduğundan bu geçiş sağlanmaz. Böylelikle otomat 0 numaralı

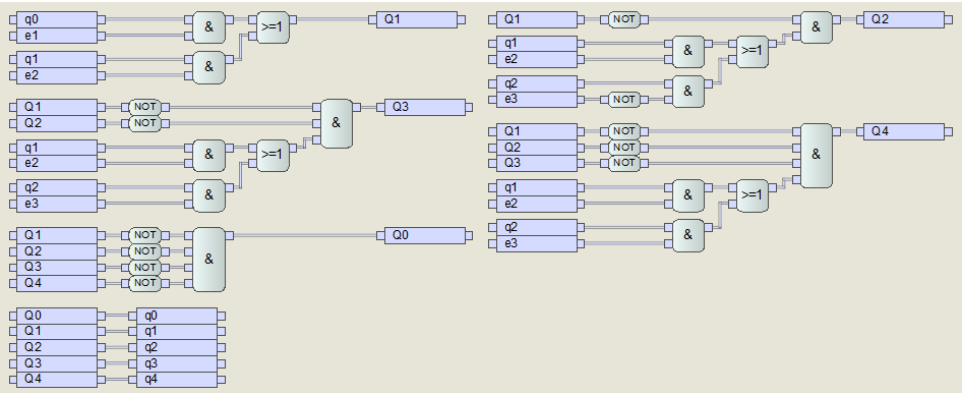
durumdayken e1 ve e2 geçişleri aynı anda tetiklense dahi e1 geçişine öncelik verilmiş olduğundan otomat 1 numaralı duruma geçecektir.

İkinci yöntem ise, böyle bir durumu tasarım aşamasında göz önünde bulundurup gerekli geçişlere öncelik vererek değil, kodlama aşamasında engellemektir. Ayrıca bu yöntemde dış etkilerden kaynaklanabilecek beklenmedik bit değişimlerinin, otomatı birden fazla duruma dallandırması da engellenmektedir. Otomatın her zaman için sadece bir durumda kalmasını garanti edeceği düşünülen yeni bir yaklaşım genel haliyle aşağıdaki gibi önerilebilir.

$$q_m(k+1) = \prod_{p=i}^{m-1} \overline{q_p(k+1)} \wedge \left(S_m(k) \vee q_m(k) \wedge \overline{R_m(k)}\right) \qquad \textbf{(3.25)}$$

Bu denklemde i, ilk durumun indis numarasını göstermektedir. Kodladığımız her yeni durumda, önceki durumların mantıksal DEĞİL'lerinin çarpımları alınmaktadır. Bu şekilde, otomatın bir önceki durumdan çıkmadan yeni bir duruma geçmemesi, önerilen bu mantıksal ifadeyle sağlanmaktadır. Önerilen bu mantıksal ifadeyle otomatın aynı anda biden fazla durumda olmaması garanti altına alınırken olabilecek tüm çığ etkileri de engellenmiş olur. Ancak mantıksal ifadenin uzunluğu durum sayısına bağlı olarak artacağından dolayı, durum sayısının fazla olduğu otomatlarda kod boyutunun artması da kaçınılmaz olmaktadır. Güvenliğin ön planda olduğu ve bellek problemi yaşanmayacak programlarda bu gerçekleme yönteminin kullanılması önerilebilir.

Bu önerilen gerçekleme yöntemi kullanılarak aşağıda Örnek 3.3'de verilen otomata ait FBD ile yazılmış PLC programı verilmiştir.



Şekil 3.15 : Örnek 3.3'te verilen otomatın sorunsuz PLC programı.

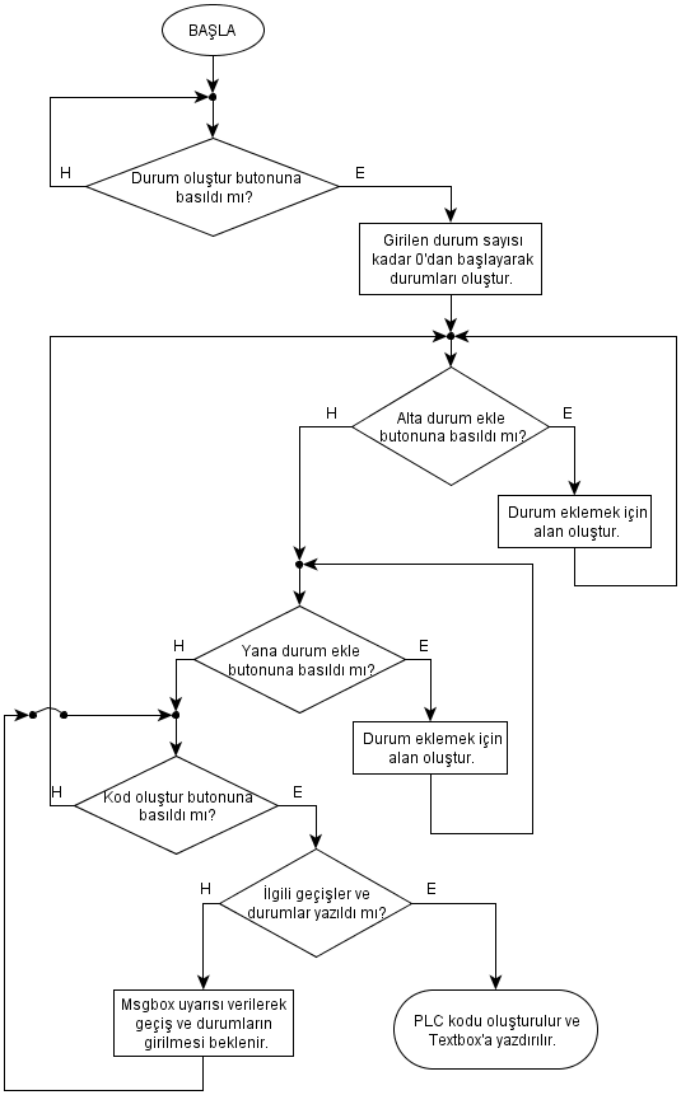
Şekil 3.15'ten görüldüğü üzere otomat 0 numaralı durumdayken aynı çevrim içinde e1 ve e2 geçişleri tetiklendiğinde, otomatta Q1 biti kurulur. Ardından aynı çevrimde Q2 bitinin kurulabilmesi için Q1 bitinin mantıksal DEĞİL'i olduğundan, Q2 biti kurulamaz. Böylelikle bir durumdan çıkartan iki farklı geçiş aynı çevrimde mantıksal 1 olsa dahi bu yeni önerilen yöntem sayesinde otomatın iki farklı duruma dallanması engellenmektedir.

4. OTOMATİK OTOMAT KODU ÜRETME

Bir önceki bölümde anlatılan matematiksel denklemlerden PLC programı elde etme yöntemi, gerçeklemenin otomatik bir şekilde gerçekleştirilmesi için bize yol göstermektedir. Burada gerçeklemenin otomatik olarak yapılmasından kasıt, tasarımın otomat modelini giriş olarak alıp, buna karşılık gelen PLC kodunu bir bilgisayar programı sayesinde otomatik olarak üretmektir. Bu bölümde otomat modellerinden sistematik bir şekilde PLC programını oluşturma özelliğinden faydalanarak, önceki bölümde anlatılan yöntemle PLC kodu üreten bir Microsoft Visual Basic uygulaması tanıtılacaktır.

4.1 Program Açıklamaları

Bu çalışmada, otomat modelinden bir endüstriyel işlemciye otomatik kod üreten bir program geliştirme amaçlanmıştır. Bu amaç için, otomat modeli oluşturulan sistemin PLC kodunu üretmek için gerçeklemesinin yapılması amacıyla tasarlanan programda, ilk önce otomatın durum sayısı programa girdi olarak alınmaktadır. Girilen sayı kadar durum, 0'dan başlanarak sırayla oluşturulur. Bu durumların her birinin yanında ise ilgili geçiş için ve ilgili geçişle gidebileceği bir sonraki durum için ayrı bir alan programda dinamik olarak oluşturulur. Eğer bir durumdan birden fazla geçiş varsa bunun için durum ekleme butonları kullanılarak istenildiği kadar geçiş durum ikilisi eklenebilir. Modellenen otomatın durum geçişleri bu alanlara girilerek, her bir durumdan ilgili geçişle gittiği durumlar belirlenmiş olur. Ardından kod oluştur butonu vasıtasıyla durum geçişleri girilen otomat modeline ait PLC kodu otomatik olarak oluşturulur. Programa ait iş akış şeması Şekil 4.1'de verilmiştir. Bu oluşturulan PLC kodu, bir önceki bölümde anlatılan çığ etkisinden arındırılmış, ilk durumun sorunsuz gerçeklendiği, aynı çevrimde birden fazla durumun kurulması veya hiçbir durumda bulunmaması gibi sorunları engelleyen matematiksel denklemlerle oluşturulmuş mantık fonksiyonlarından türetilmiştir.



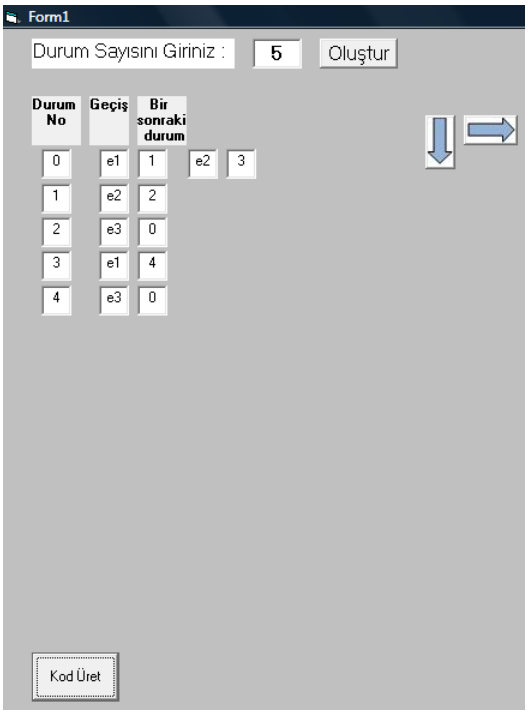
Şekil 4.1 : Geliştirilen programın iş akış şeması.

Bu programın kullanımı ile yukarıdaki sorunlardan arındırılmış bir gerçekleme yöntemi kullanıldığından, otomatın davranışına uygun bir PLC kodu elde etmiş oluruz. Ayrıca otomat modelinden PLC kodu elde ederken ortaya çıkabilecek insan hataları da bu program sayesinde yok edilmiş olmaktadır. Böylece uygun gerçekleme yöntemini bilmemize rağmen otomattan koda dönüşüm sırasında yapılabilecek ufak hatalar da engellenerek, hata arama süre kaybını da geliştirilen bu program sayesinde sıfırlamış olmaktayız.

Örnek 4.1

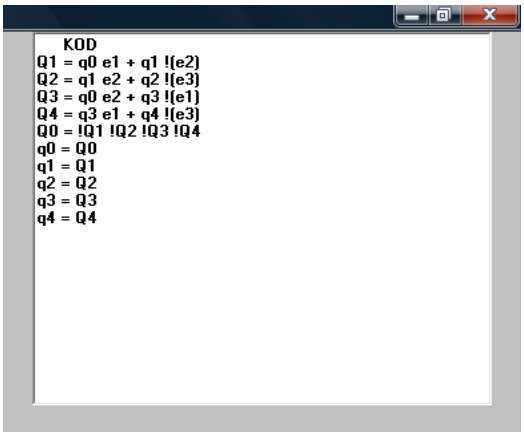
Geliştirilen bu program ile bir önceki bölümde verilen Şekil 3.9'daki otomata ilişkin PLC kodunu elde edelim.

Öncelikle otomat 5 durumlu olduğundan dolayı ilgili alana 5 sayısı girilerek “Oluştur" butonuna basılır. Butona basılmasıyla birlikte “Durum No” yazan yerin altına 0’dan 4’e kadar durumlar otomatik olarak oluşturulur. Ardından durumlardan çıkan her bir geçiş ve bu geçişle gideceği durumu belirtmek için altı ve yanı gösteren oklar ile geçiş-durum ikilileri eklenir. Bir durumdan birden fazla duruma ilgili geçişlerle gidiliyorsa yanı gösteren ok ile istenildiği kadar geçiş-durum ikilileri eklenebilir. İlgili durumların yanına sadece o durumdan çıkartan geçiş ile gittiği durum giriş olarak beklenmektedir. Şekil 4.2’de görüldüğü üzere 0 numaralı durumdan e1 geçişi ile 1 numaralı numaralı, e2 geçişi ile ise 3 numaralı duruma gittiği belirtilmiştir, 0 numaralı duruma ilgili geçişlerin tetiklenmesiyle gelen durumlar belirtilmemiştir.



Şekil 4.2 : Geliştirilen programa otomatın durum ve geçişlerinin girilmesi.

Oluşturulan alanlara ilgili geçiş ve durum numaraları yazılarak “Kod Üret” butonuna basılır. Şekil 4.3’te görüldüğü üzere modellenen otomata ilişkin PLC kodu elde edilmiş olur.



Şekil 4.3 : Geliştirilen program ile otomatik olarak elde edilen PLC kodu.

Yukarıdaki şekilden de görüldüğü üzere daha önce elde edilen kodun aynısını otomatik bir şekilde geliştirilen programı kullanarak elde ettik. Burada "!" işareti kendisinden sonra gelen ifadenin mantıksal DEĞİL'ini temsil etmek üzere kullanılmıştır.

Bu program sayesinde daha karışık otomatları da aynı şekilde elde edebiliriz. Programı kullanırken dikkat edilmesi gereken noktalardan biri eğer bir ifadenin mantıksal DEĞİL'ini yazmak istersek bunun için "!" işareti kullanmamız gerekir. Ayrıca otomat modelini oluştururken durum isimleri olarak 0'dan başlayan sırayla devam eden sayıları kullanmak gerekir. Programda durum sayısını girdikten sonra oluşan durumlar 0'dan başlayarak sırayla devam ettiğinden dolayı buna denk gelmesi için modellemede de durum isimleri bu şekilde olmalıdır.

5. ANKLAŞMAN ALGORİTMALARI İÇİN OTOMATİK DEĞİŞKEN İSİMLERİ OLUŞTURMA

Algoritma tasarımı aşamasında ilk olarak kullanacağımız değişken isimlerinin kararlaştırılıp bu değişkenleri PLC'ye tanıtmamız gerekir. Fazla güzergâh sayısına sahip istasyonlarda saha elemanlarının sayısı da fazla olacağından dolayı PLC'de kullanılacak değişken sayısı da fazla olmaktadır. Bunları her seferinde elle tek tek girmek üzerimize fazladan iş yükü bindirmektedir. Ayrıca tezde gösterilen istasyondan başka bir istasyon ile karşılaşmamız durumunda, tekrar değişken isimlerini düşünmemiz ve bu değişken isimlerini PLC'ye tek tek girmemiz gerekmektedir. Kullanmamız gereken değişken sayısı ne kadar fazla olursa her birini tek tek yazmak tasarım süresini o kadar uzatmaktadır.

Bu işi daha kısa bir sürede yapabilmek ve herkes tarafından anlaşılabilecek otomatik değişken isimleri üretmek için Microsoft Visual Basic programı kullanılarak bir yazılım geliştirilmiştir.

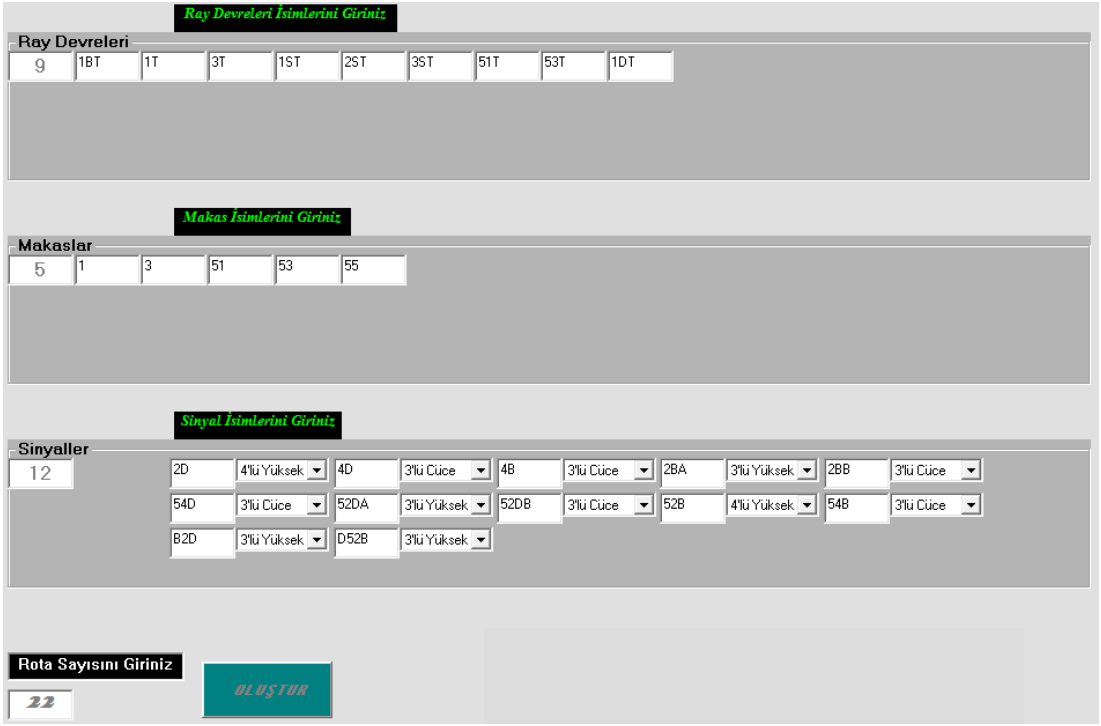
5.1 Program Açıklamaları

Aşağıdaki Şekil 5.1'de gösterildiği üzere yazılıma ilk başta istasyonda kaç tane ray devresi, makas, sinyal ve rota olduğunu kullanıcıdan girmesi beklenir.



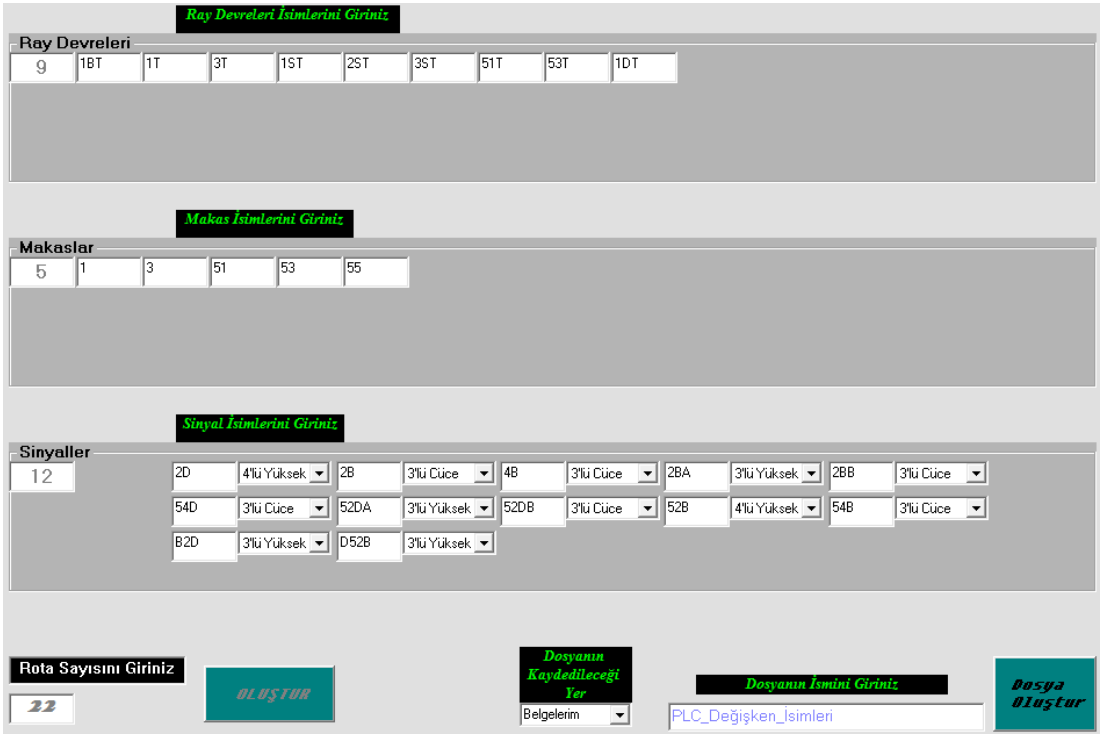
Şekil 5.1 : İlgili alanlara sayıların yazılması.

Ardından “OLUŞTUR” butonuna basılmasıyla girdiği sayı kadar kutucuk ilgili yerlerde Şekil 5.2’de görüldüğü üzere dinamik olarak oluşturulmaktadır. Kullanıcının bu kutuların içine ray devrelerinin, makasların, sinyallerin isimlerini girmesi gerekmektedir. Ayrıca sinyalin 3’lü cüce, 3’lü yüksek ve 4’lü yüksek olmasına göre farklı indikasyonları ve komutları olacağından dolayı, sinyallerin tipleri de seçilmesi gerekmektedir.



Şekil 5.2 : İlgili alanlara saha elemanları isimlerinin girilmesi.

Bunları yaptıktan sonra arayüzde bulunan “Dosyanın kaydedileceği yer” kısmından dosyanın bilgisayarda nereye kaydedileceğini seçmesi gerekmektedir. Ardından Şekil 5.3’de gösterildiği üzere hemen yanındaki “Dosyanın İsmini Giriniz” yazılı yerin altındaki ilgili yere ise dosyanın hangi adla kaydedileceğini yazması gerekir. Bu işlemleri yaptıktan sonra “Dosya Oluştur” butonuna bastığında ise istasyon için olabilecek bütün değişkenler “.csv” formatındaki dosyaya yazdırılacaktır.



Şekil 5.3 : İstenilen yere dosyanın oluşturulması.

Daha sonra oluşturulan bu değişken isimleri dosyası, SILWORX programında "Global Variables" menüsünden "import" edilerek bütün değişkenler açıklamalarıyla birlikte programa tanıtılmış olur.

Bu programla amacımız algoritma tasarım sürecinde algoritmada kullanılacak değişken isimlerinin tekrar tekrar düşünülerek zaman kaybının engellenmesi ve standart bir hal almasıdır. Ayrıca farklı bölgeler için saha elemanları isimleri ve rota sayıları değişebilir. Bu yazılımı kullanarak aynı standarttaki farklı değişken isimlerini oluşturup, algoritmanın yazılacağı PLC programına kısa bir sürede tanıtmış oluruz.

6. ÖRNEK BİR İSTASYON HATTININ OTOMATLAR KULLANILARAK ANKLAŞMAN ALGORİTMASI TASARIMI

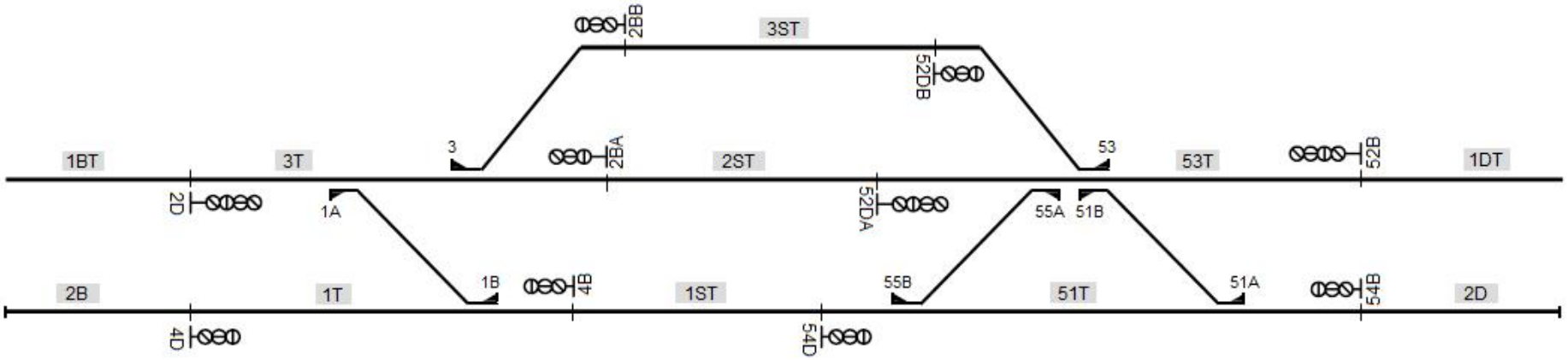
6.1 Giriş

Bu bölümde örnek bir istasyon hattının, otomat yaklaşımı kullanılarak anklaşman algoritması tasarımının gerçeklenmesi adım adım gösterilecektir. Öncelikle örnek istasyon hattı tanıtılacak, ardından bu istasyon hattı için anklaşman tablosu oluşturulacaktır. Oluşturulan anklaşman tablosu üzerinden modüler bir yaklaşım metoduyla otomatlar kullanılarak algoritma tasarımı gerçekleştirilecektir. Algoritma tasarımı; tanzim bloğu, makas blokları ve sinyal bloğu olmak üzere 3 temel fonksiyon bloğuna ayrılacaktır. Ayrıca bütün rotalar için kullanılabilecek tek bir tanzim bloğu için, önüne ve arkasına yerleştirilen "Tanzim Giriş Anahtarlama Bloğu" ve "Tanzim Çıkış Anahtarlama Bloğu" adı verilen fonksiyon blokları kullanılarak modüler bir yaklaşım önerilecektir.

6.2 Sinyalizasyonu Yapılan Örnek İstasyonun Hattının Tanıtılması

Bu tez çalışmasında sinyalizasyonu yapılan örnek istasyon hattı aşağıdaki Şekil 6.1'de gösterilmiştir.

Şekilden görüldüğü üzere örnek istasyon hattı 9 adet ray devresi, 5 adet makas, 10 adet sinyalden oluşmaktadır. Ray devrelerinin isimlerinin sonuna "T" harfi konulmuştur. İsminin sonuna "T" harfi konmayanlar ray devresi olmayan ray bölgeleridir. Buralarda trenin varlığını kontrol edemeyiz. Buna karşın bu bölgelerden tren çıkarmak için diğer rotalarda olduğu gibi tanzim talebi gerekmektedir. AS'nin bu rotalar için tanzim gerçekleştirmesiyle trenler ray devrelerine giriş yaparlar ve AS trenlerin konumlarını kontrol edebilir duruma geçer.



Şekil 6.1 : İstasyon bölgesi.

Aşağıda Çizelge 6.1'de bu örnek istasyonda bulunan ray bölgelerinin isimleri, ray devreli olup olmadıkları ve istasyona göre bulundukları bölgeler verilmiştir. 1ST, 2ST ve 3ST Şekil 6.1'den de görüldüğü üzere istasyon ray devreleridir. 2B ve 2D isimli bölgeler kendilerine ait sinyalleri olan fakat ray devreleri olmayan ray bölgeleridir.

Çizelge 6.1 : İstasyon hattında bulunan ray bölgeleri.

Ray Bölgesi İsimleri	**Ray Devresi**	**Bulunduğu Bölge**
1BT	Var	Batı
1T	Var	Batı
3T	Var	Batı
1ST	Var	İstasyon
2ST	Var	İstasyon
3ST	Var	İstasyon
51T	Var	Doğu
53T	Var	Doğu
1DT	Var	Doğu
2B	Yok	Batı
2D	Yok	Doğu

Örnek istasyon hattında bulunan makas isimleri, yapıları, üzerlerinde bulundukları ray devreleri ve istasyona göre bulundukları bölgeler aşağıdaki Çizelge 6.2'de verilmiştir. Bu çizelgeden görüldüğü üzere basit makaslar tek bir ray devresi üzerinde dallanmayı sağlarken, S makaslar iki ray devresini birbirine bağlamaktadır.

Çizelge 6.2 : İstasyon hattında bulunan makaslar.

Makas İsimleri	**Makas Yapısı**	**Bulunduğu Ray Devresi**	**Bulunduğu Bölge**
1	S Makas	3T, 1T	Batı
3	Basit Makas	3T	Batı
51	S Makas	51T, 53T	Doğu
53	Basit Makas	53T	Doğu
55	S Makas	51T, 53T	Doğu

Çizelge 6.3'de ise örnek istasyon hattında bulunan sinyallerin isimleri, görünüşleri ve görevleri bakımından tipleri, istasyona göre bulundukları bölgeler ve gösterdikleri yönler belirtilmiştir.

Çizelge 6.3 : İstasyon hattında bulunan sinyaller.

Sinyal İsimleri	**Sinyal Tipleri**		**Bulunduğu Bölge**
2BA	3'lü Yüksek	Çıkış	Batı
2BB	3'lü Cüce	Çıkış	Batı
4B	3'lü Cüce	Çıkış	Batı
2D	4'lü Yüksek	Giriş	Batı
4D	3'lü Cüce	Giriş	Batı
52DA	4'lü Yüksek	Çıkış	Doğu
52DB	3'lü Cüce	Çıkış	Doğu
54D	3'lü Cüce	Çıkış	Doğu
52B	4'lü Yüksek	Giriş	Doğu
54B	3'lü Cüce	Giriş	Doğu

6.3 Sinyalizasyonu Yapılan Hat İçin Anklaşman Tablosunun Oluşturulması

Yolların tanzimi ile ilgili olarak demiryolu trafiğinin emniyetini sağlamak maksadı ile tertiplenen anklaşman tablosu, sinyalizasyon tasarım sürecinde önemli bir role sahiptir. Anklaşman tabloları, anklaşman tasarımcıları tarafından tasarım spesifikasyonlarında ve testçiler tarafından test aşamalarında kullanılmak üzere oluşturulan tablolar olarak addedilirler ve anklaşman sistemleri için anahtar fonksiyonel güvenlik isterlerini içerirler. Bir anklaşman tablosu, güzergah tanzimi için gerekli olan düzenlemelerin ve kuralların yapılandırılmış tablosal gösterimidir (Mirabadi ve Yazdı, 2009).

Anklaşman tablolarının formatı ve içeriği standartlaşmamış olup, aynı demiryolu idaresi içinde bile değişkenlik gösterebilir. Yine de anklaşman tablo tasarımının genel prensipleri apaçık ortadadır (Tombs, 2002).

Anklaşman yazılımı tasarımı anklaşman tablosu hazırlama işlemi ile başlamış olur (Türk S. ve Sonat A., 2010). Üzerinde saha elemanlarının konumları belirtilen ve isimleri yazan istasyon hattı planı elimize geçtikten sonra bu istasyona ait anklaşman

tablosu oluşturma işlemine geçilir. Bu tablo, yol modeline göre tertip edilirken genellikle aşağıdaki bilgileri içerir:

- Bölgedeki tüm güzergâhlar
- Güzergâhların tanzimi için ilgili makasların olması gereken konumları
- Güzergâhların tanzimi için kontrol edilmesi gereken ray devrelerinin durumları
- Güzergâhların tanzimi için kilitlenmesi gereken sinyaller, güzergâhlar, makaslar
- Güzergâh tanzimi için gerekli koşullar sağlandığında, TCDD yönetmeliğine göre belirlenmiş şekilde bildirim verecek sinyaller

İstasyonlarda rotaların tanzimi, bu anklaşman tablosuna göre kontrol merkezinden, o güzergâhtaki sinyal, makas ve hat engellerinin tanzimi sureti ile yapılır.

Aşağıda gösterilen Çizelge 6.4'te ilk sütundaki "Tanım" kısmında istasyonda bulunan rotalar numaralarıyla birlikte verilmiştir.

2. sütunda ise güzergâhta bulunan sinyalin ilgili yön için ne yanması gerektiği belirtilmiştir. Burada önemli olan nokta ise bu sinyalin hangi renk bildirimini vermesi gerektiği, 3. sütunda verilen sinyalin durumuna göre değişmektedir. Örneğin, rota 1'i tanzim ettiğimizde 54D sinyali sarı renk bildirimini verirken 2D sinyali 1S yolu için sarı üzeri yeşil, 54D sinyali kırmızı veya sarı üzeri kırmızı renk bildirimlerinden birini verirken 2D sinyali 1S yolu için sarı üzeri sarı renk bildirimini verir.

4. sütunda, rotayı tanzim ettiğimizde rotada bulunan her bir makasın hangi yönde kilitlemelerinin yapılması gerektiği gösterilmektedir. Burada makas isimlerinin önündeki "t" harfi o makasın ters yönde kilitlenmesi gerektiğini belirtmektedir. "t" harfinin bulunmadığı durumlarda ise makasın normal konumda kilitlenmesi amaçlanmaktadır. Ayrıca bu sütunun altında bulunan ikinci kısımda hangi sinyallerin kilitlenmesi gerektiği gösterilmiştir. Örneğin rota 1'i tanzim etmemiz durumunda Çizelge 6.4'te gösterildiği üzere rt1 satırının 4. sütununda; t1 ile 1 makasının ters yönde kilitlenmesi, 4B ile 4B sinyalinin kilitlenmesi, 52B1 ve 54B1 ile ise 52B ve 54B sinyallerinin 1ST ray devresine doğru açılmaması için kilitlenmesi gerektiği kastedilmektedir.

Rota tanzim edilirken rotada bulunan ray devrelerinin durumları ise 5. sütunda gösterilmiştir. Rota tanzimi için ray devrelerinin boş olması gerekmektedir. Eğer tanzim talebi gelen rota istasyon ray devresi içeriyorsa, istasyon ray devresi dolu olsa dahi rota tanzim edilir. Bu durumda ilgili sinyal, Çizelge 6.4'ten de görüldüğü gibi 3. sütunda gösterilen kendisinden bir sonraki sinyalin verdiği renk bildirimine bakılmaksızın sarı üzeri kırmızı renk bildirimi vermesi sağlanır.

Son sütunda ise rota tanzim edildiğinde kilitlenmesi gereken ray devreleri isimleri verilmiştir. Tren tanzim ettiği rotada bulunan ray devrelerinin üzerinden geçip bu ray devrelerinden çıkmadığı sürece bu kilitler kaldırılmaz. Bu kilitler süresince makas kilitleri de kalkmaz.

Aşağıdaki Çizelge 6.4'te ele alınan istasyon bölgesine ait anklaşman tablosunun rota 14'e kadar olan kısmı, Çizelge 6.5'te ise rota 15'ten rota 20'ye kadar olan kısmı gösterilmektedir.

Çizelge 6.4 : Örnek istasyon için oluşturulmuş anklaşman tablosu (rota 1-rota 14).

1			2		3		4		5	6
TANIM			Sinyal No		Sinyal Durumu		Kilit		Sinyal Kontrol veya Dedektör Kontrol	Rota Kilit
İsim	Rota No	Rota								
GİRİŞ SİNYALİ	rt1	1BT – 1ST	2D	1S	SY	54D S	t1	4B 52B1 54B1	3T 1T 1ST	(3T)(1T)
					SS	54D K, SK				
					SK				3T 1T t1ST	
	rt2	1BT – 2ST		2S	Y	52DA S	1 3	2BA 52B2 54B2	3T 2ST	(3T)
					S	52DA K, SK				
					SK				3T t2ST	
	rt3	1BT – 3ST		3S	SY	52DB S	1 t3	2BB 52B3 54B3	3T 3ST	(3T)
					SS	52DB K, SK				
					SK				3T t3ST	
	rt4	(2B) – 1ST	4D		Y	54D S	1	4B 52B1 54B1	1T 1ST	(1T)
					S	54D K, SK				
					SK				1T t1ST	
ÇIKIŞ SİNYALİ	rt5	1ST – 1BT	4B	1B	S		t1	2D	1T 3T 1BT	(1T)(3T)
	rt6	1ST – (2B)		2B	SK		1	4D	1T	(1T)
	rt7	2ST – 1BT	2BA		S		3 1	2D	3T 1BT	(3T)
	rt8	3ST – 1BT	2BB		S		t3 1	2D	3T 1BT	(3T)
	rt9	1ST – 1DT	54D	1D	S		t55 53 51	52B	51T 53T 1DT	(51T)(53T)
	rt10	1ST – (2D)		2D	SK		55 51	54B	51T	(51T)
	rt11	2ST – 1DT	52DA	1D	S		55 53 51	52B	53T 1DT	(53T)
	rt12	2ST – (2D)		2D	SK		55 53 t51	54B	53T 51T	(53T)(51T)
	rt13	3ST – 1DT	52DB	1D	S		t53 51	52B	53T 1DT	(53T)
	rt14	3ST – (2D)		2D	SK		t53 t51	54B	53T 51T	(53T)(51T)

Çizelge 6.5 : Örnek istasyon için oluşturulmuş anklaşman tablosu (rota 15-rota 20).

1			2		3		4		5	6
TANIM			Sinyal No		Sinyal Durumu		Kilit		Sinyal Kontrol veya Dedektör Kontrol	Rota Kilit
İsim	Rota No	Rota								
GİRİŞ SİNYALİ	rt15	1DT – 1ST	52B	1S	SY	4B S	51 53 t55	54D 2D1 4D1	53T 51T 1ST	(53T)(51T)
					SS	4B K, SK				
					SK				53T 51T t1ST	
	rt16	1DT – 2ST		2S	Y	2BA S	51 53 55	52DA 2D2	53T 2ST	(53T)
					S	2BA K				
					SK				53T t2ST	
	rt17	1DT – 3ST		3S	SY	2BB S	51 t53	52DB 2D3	53T 3ST	(53T)
					SS	2BB K				
					SK				53T t3ST	
	rt18	(2D) – 1ST	54B	1S	Y	4B S	51 55	54D 2D1 4D1	51T 1ST	(51T)
					S	4B K, SK				
					SK				51T t1ST	
	rt19	(2D)– 2ST		2S	Y	2BA S	t51 53 55	52DA 2D2	51T 53T 2ST	(51T)(53T)
					S	2BA K				
					SK				51T 53T t2ST	
	rt20	(2D)– 3ST		3S	Y	2BB S	t51 t53	52DB 2D3	51T 53T 3ST	(51T)(53T)
					S	2BB K				
					SK				51T 53T t3ST	

6.4 Algoritma Tasarımının Modüler Bir Tasarım Metoduyla Gerçekleştirilmesi

Algoritma tasarımı yukarıda gösterilen anklaşman tablosu temel alınarak gerçekleştirilecektir. Bu tabloda rota sayısı kadar satır bulunmaktadır ve her satırda ilgili rotanın tanzimi durumunda yapılması gereken düzenlemeler ve uyulması gereken kurallar belirtilmiştir. Bu rotaların her biri için fonksiyon bloğu oluşturup ilgili girişleri içine almak suretiyle algoritma tasarlanabilir. Burada bulunan makaslar ve sinyaller genel rota tanzimi fonksiyon bloğundan ayrı düşünülebilir. Rota tanzimi fonksiyon bloğunda saha ile ilgili kontroller yapıldıktan sonra rotada bulunan makaslar ile ilgili kontrollerin yapılması ve bu kontrollerin uygun bulunması durumunda makaslara komut gönderilmesi gerekir. Bu makaslarla ilgili olan kod parçaları farklı fonksiyon bloklarda yapılabilir. Makas fonksiyon bloklarında yapılan kontrollerin sonucunda makas ilgili konuma alınır ve bu bilgi ilgili rota tanzimi fonksiyon bloğuna gönderilir. Bu bilgiyle birlikte tanzim fonksiyon bloğunda durum geçişi sağlanır. Aynı durum sinyaller için de geçerlidir. Rota tanzimi fonksiyon bloğunda sinyalin açılması gereken duruma gelindiğinde sinyal fonksiyon bloğuna komut gönderilir. Bu fonksiyon bloğunda yapılan kontroller sonucu ilgili sinyale uygun renk bildirimini vermesi için komut yollanır.

Algoritmayı tasarlarken yapacağımız bütün işi tek bir blokta yapmak yerine, bunları fonksiyon bloklara bölerek programlamak bize birçok konuda avantaj sağlamaktadır. Bize sağladığı faydaları şöyle sıralayabiliriz:

- Otomatlardaki en önemli sorun karmaşık sistemler için durum patlaması olayıdır. Bu sorunu elemine etmek için çözüm sistemi modüllere ayırarak tasarım yapmaktır (Türk S. ve Sonat A., 2010).
- Programın kaynak kodu küçülür.
- Kaynak dosyanın okunabilirliği artar. Okunabilirliğin artması, kodu yazanın ve okuyanın işini kolaylaştırır. Böylece proje geliştirme sürecinin azalması yönünde kazanım sağlanmış olur.
- Belirli kod parçalarının programın farklı yerlerinde yinelenmesi, programda yapılacak olası bir değişikliğin maliyetini arttırır. Programın farklı yerlerinde, kodun kullanıldığı yere bağlı olarak değişiklikler yapmak gerekir. Kaynak dosyalarda böyle değişiklikler yapmak hem zaman alıcıdır hem de risklidir. Çünkü bir değişikliğin yapılmasının unutulması durumunda ya da değişiklik

yapılmaması gereken bir yerde kodun değiştirilmesi durumunda program yanlış çalışabilir. Oysa ortak kod parçaları fonksiyonlar şeklinde paketlendiğinde, yalnızca fonksiyonlarda değişiklik yapılmasıyla, istenen değişiklik gerçekleştirilmiş olur.

- Programda hata arama daha kolay gerçekleştirilir. Projelerdeki hata arama maliyeti azalır.
- Yazılan fonksiyonlar başka projelerde de kullanılabilir.

Fonksiyon blokları PLC programlamanın yapı taşlarıdır. Projede kullandığımız HIMA PLC'lerin geliştirme ortamı olan Silworx 3.30 versiyonlarında herhangi fonksiyon bloğu kısıtlaması yoktur.

Anklaşman algoritması; tanzim, makas ve sinyal olmak üzere fonksiyon bloklara ayrılarak tasarlanması bize yukarıda saydığımız avantajları sağlamaktadır. İstasyonda bulunan olası güzergâh sayısı kadar tanzim fonksiyon bloğu, makas sayısı kadar makas fonksiyon bloğu ve sinyal sayısı kadar sinyal fonksiyon blokları tasarlanabilir. Ele aldığımız örnek istasyonda 20 rota, 5 adet makas ve 10 adet de sinyal bulunmaktadır.

Her bir rotada aynı sayıda saha elemanları bulunmadığından dolayı her bir rota için farklı tanzim fonksiyon bloğu tasarlamak gerekmektedir. Örneğin rota 1'de kontrol edilmesi gereken ray devreleri (3T, 1T, 1ST) 3 tane, makas (1) 1 tane iken rota 10'da sadece 1 tane ray devresi (51T) bulunurken 2 adet makas (51, 55) bulunmaktadır. Makaslarda yapılan kontroller aynı olduğundan dolayı tek bir makas fonksiyon bloğu oluşturup, 5 adet kopyalayarak her bir makas ile ilgili değişkenler bu fonksiyon bloklarının girişlerine bağlanır.

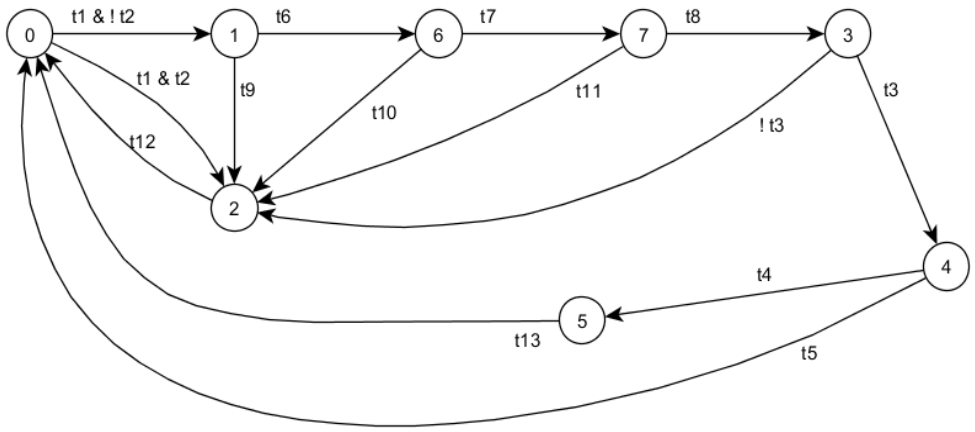
Tasarlanabilecek genel bir tanzim fonksiyon bloğu sayesinde ise 20 adet farklı fonksiyon bloğu tasarlamanın önüne geçilmiş olunur. Her bir rotada bulunan maksimum sayıdaki saha elemanlarını göz önünde bulundurarak genel bir tanzim fonksiyon bloğu oluşturulur. Bundan 20 tane kopyalayarak her bir rota ile ilişkili değişkenler tanzim fonksiyon bloğunun girişlerine bağlanır. İlgili rotada kullanılmayan girişlere ise duruma göre 0 veya 1 ataması yapılarak ilgili geçişlerin yapılması sağlanır.

Bu tezde tasarlayacağımız tek bir tanzim fonksiyon bloğu sayesinde, 20 tane farklı tanzim fonksiyon bloğu tasarlamaktan kurtulmuş olacağız. Ayrıca tasarlayacağımız

tanzim fonksiyon bloğundan 20 tane kopyalayıp, her birinin girişlerine ilgili bitleri atamak yerine, “Tanzim Giriş Anahtarlama Algoritması” adını verdiğimiz tek bir fonksiyon bloğu tasarlayarak, bu fonksiyon bloğun çıkışlarını tanzim fonksiyon bloğunun girişlerine atama yapmak üzere bağlayacağız. “Tanzim Çıkış Anahtarlama Algoritması” adını verdiğimiz bir fonksiyon bloğunun girişlerine de tanzim fonksiyon bloğunun çıkışlarını bağlayacağız. Böylece global değişkenler TGAO’da jenerik değişkenlere dönüştürülüp tanzim algoritmasında kullanılacak olan girişlere atanacak. Bu algoritma sırasında aktif olan jenerik değişkenler TÇAO’na gönderilecek ve burada tasarlanan algoritma sayesinde gerçek değişkenlere dönüştürülecek. Tezin ileriki bölümlerinde bu fonksiyon bloklar daha detaylı bir biçimde açıklanacaktır.

6.4.1 Jenerik tanzim algoritması

Tanzim otomatı, ilgili güzergâhta bulunan ray devrelerinin ve rota kilitlerinin kontrolünü, bu güzergâhla çakışan başka bir güzergâh için tanzim talebi olması halinde bu güzergâhın işletilip işletilmediğinin kontrolünü yapar. Koşulların uygun bulunması durumunda ilgili güzergâhta bulunan makasların güzergâhın gerektirdiği konumlara geçmesi için makas otomatlarına komut gönderip makasların talep edilen konuma geçtiği bilgisini bekler. Makas ve sinyal otomatlarından hata bilgisi gelip gelmediğini kontrol eder. Güzergâhın tanzimine aykırı bir durum tespit etmemesi halinde sinyal otomatına gerekli bilgiyi göndererek güzergâh sinyalinin yakılmasını sağlar (Türk S. ve Sonat A., 2010). Tanzim otomatına ilişkin gösterim Şekil 6.2’de verilmiştir.



Şekil 6.2 : Jenerik tanzim otomat modeli.

Tanzim otomatındaki geçişler,

$$t1 = \text{jrt1} \tag{6.1}$$

jrt1 ilgili rota tanzim talebi iken

$$t2 = \left(\bigvee_{i=1}^{3} \overline{\text{jrd(ı)}}\right) \vee \left(\bigvee_{j=1}^{2} \text{jrk(j)}\right) \vee \left(\bigvee_{k=2}^{3} \text{jrt(k)_ist}\right) \tag{6.2}$$

2 numaralı denklemde; jrd(i) güzergâhta bulunan ray devrelerinin meşgul olmaması durumunu, üstü çizili olduğunda ise meşgul olduğunu, jrk(j) rota kilitlerini ve jrt(k)_ist ise çakışan güzergâh isteklerini temsil eder. Çakışan güzergâhtan kasıt, son ray devreleri ortak olan iki güzergâhtır. Örneğin, Şekil 6.1'de verilen istasyon bölgesinde 1BT-2ST rotası ile 1DT-2ST ve 2D-2ST rotaları çakışan güzergâhlardır. Bu çakışan güzergâhlarda kurulu veya kurulmakta olan bir tanzim talebi olup olmadığı, tanzim otomatına tanzim talebinin gelmesiyle birlikte kontrol edilir. Olması durumunda tanzim talebi reddedilir. Çakışan güzergâhların özellikle burada belirtilmesinin nedeni; diğer güzergâhların rota kilidine bakılırken, çakışan güzergâhların sadece son ray devreleri ortak olduğundan dolayı rota kilidine bakılamamasıdır.

$$t3 = \bigwedge_{i=1}^{3} \text{jrd(i)} \tag{6.3}$$

$$t4 = (t3 \wedge \text{jrt1_ipt} \uparrow) \vee js_hata \tag{6.4}$$

4 numaralı denklemde jrt1_ipt rota tanzim iptal talebini gösterir. Yanındaki ok ise bu bitin çıkan kenarını temsil etmektedir. js_hata ise güzergâh sinyali hatasını göstermektedir.

$$t5 = \left(\overline{jrd3} \vee jrd2 \uparrow\right) \tag{6.5}$$

$$t6 = jm_1_klt \tag{6.6}$$

$$t7 = jm_2_klt \tag{6.7}$$

$$t8 = jm_3_klt \tag{6.8}$$

jm_1_klt, jm_2_klt, jm_3_klt güzergâhtaki makasların kilitlendiğine dair değişkenlerdir.

$$t9 = jm_1_hata \tag{6.9}$$

$t10 = jm_2_hata$ **(6.10)**

$t11 = jm_3_hata$ **(6.11)**

jm_1_hata, jm_2_ hata, jm_3_ hata güzergâhtaki makasların hataya düştüğünü temsil eden değişkenlerdir.

$t12 = 5\ saniye\ (2.durumda)$ **(6.12)**

$t13 = 3\ dakika\ (5.durumda)$ **(6.13)**

6.12 numaralı denklem otomat 2. durumdayken 5 saniyenin dolmasıyla aktif olacak geçişi, 6.13 numaralı denklem ise otomat 5. duruma geldiğinde 3 dakikanın geçmesiyle aktif olacak geçişi göstermektedirler.

Tanzim otomatını açıklayacak olursak, herhangi bir tanzim talebi olmaması durumunda başlangıç durumu olan durum 0'da bekler. Güzergâh tanzim talebiyle birlikte ilgili güzergâhın tanzim edilebilmesi için gerekli olan şartların sağlanıp sağlanmadığının kontrolünü yapar. Bu şartlar; ray devrelerinin meşgul olmaması, bu rota için daha önceden rota kilidinin olmaması ve bu rotayla çakışan kurulu veya kurulmakta olan bir rotanın olmamasıdır. Bu şartların sağlanması durumunda tanzim otomatı 1 numaralı duruma geçer ve ilgili rota kilitlerini çeker. Sağlanmaması durumunda ise 2. duruma geçerek rota tanzim talebini reddeder. 1, 6, 7 durumlarında güzergâhta bulunan ilgili makaslara güzergâhın gerektirdiği konumlara çevrilmeleri için, makasların bölgedeki konumlarına uygun bir sırada gerekli komutları yollar. Her bir makas için yolladığı komut sonrası ilgili makasın kilitlendiği kontrolünü yapar. Makasın kilitlenmesiyle beraber durum atlar. Eğer komut gönderdiği makaslardan en az biri hataya düşer ise otomat yine 2. duruma giderek tanzim talebi reddedilir. İlgili güzergâhtaki bütün makaslar uygun konuma gelip kilitlendikten sonra otomat 3 numaralı duruma geçer ve güzergâhtaki son kontroller yapılır. Bu kontrollerin uygun bulunması durumunda ilgili güzergâh tanzim edilmiş olur (Durum 4). 4 numaralı durumda güzergâh tanzimliyken tren daha güzergâha girmeden rota iptal talebi gelmesi durumunda veya güzergâh sinyalinin hataya düşmesiyle birlikte 5. duruma geçilir, sinyalin kırmızı bildirim vermesi sağlanır ve burada kilitlerin çözülmesi için güvenlik amaçlı olarak 3 dakika beklenir. Trenin güzergâhın son ray devresinden bir önceki ray devresinden çıkmasıyla birlikte güzergâh tanzimi sonlanır ve tanzim otomatı başlangıç durumuna döner. Böylece yeni bir güzergâh tanzim talebini değerlendirmek için hazır duruma gelir.

Her bir rota için yapılan kontroller aynı olduğundan dolayı bu tanzim otomatı, örnek istasyondaki herhangi bir rotada bulunabilecek en fazla sayıdaki makas ve ray devreleri temel alınarak tasarlanmıştır. Boşta kalacak makas ve ray devreleri için TGAO'da ilgili değişkenlere 1 veya 0 atamaları yapılarak geçişlerin aktif hale gelmesi sağlanmıştır. Bu konu TGAO tasarımında daha detaylı anlatılacaktır.

Güzergâhlarda bulunan makas sayıları tanzim otomatında durum artışına sebep olabilecek bir durumdur. Makaslar motorlarının güç kaynağından fazla akım çektikleri için aynı güç kaynağına bağlı makasların aynı anda çalışmaları istenmez. Yukarıda anlattığımız tanzim otomatı güzergâhta en fazla 3 makas olması durumunda geçerlidir. Bizim ele aldığımız örnek istasyon hattında bulunan olası rotalarda en fazla 3 makas olabilmektedir. Bizim incelediğimiz hattan farklı bölgelerde bir güzergâhta 3'ten fazla makas olması durumunda ise Şekil 6.2'de gösterilen otomat modelindeki 1 ve 3 numaralı durumlar arasında kalan durumların arttırılması yeterli olacaktır. Buradaki durumlar sadece makas fonksiyon bloklarına komut yollamak için eklenmiş durumlardır. İlgili makas bloğuna komut yolladıktan sonra makasın hazır olduğuna dair bilginin gelmesiyle güzergâhta bulunan diğer makasa komut yollamak için bir sonraki duruma geçilir. Makas bloğundan hata bilgisi gelmesi durumunda ise 2 numaralı duruma geçilir ve tanzim talebi reddedilir.

Çizelge 6.6 : Tanzim otomatına ait PLC kodu.

$Q_1 = (q_0 \wedge t_1 \wedge \overline{t_2}) \vee (q_1 \wedge \overline{t_6} \wedge \overline{t_9})$
$Q_2 = (q_0 \wedge t_1 \wedge t_2) \vee (q_1 \wedge t_9) \vee (q_6 \wedge t_{10}) \vee (q_7 \wedge t_{11}) \vee (q_3 \wedge \overline{t_3}) \vee (q_2 \wedge \overline{t_{12}})$
$Q_3 = q_7 \wedge t_8$
$Q_4 = (q_3 \wedge t_3) \vee (q_4 \wedge \overline{t_4} \wedge \overline{t_5})$
$Q_5 = (q_4 \wedge t_4) \vee (q_5 \wedge \overline{t_{13}})$
$Q_6 = (q_1 \wedge t_6) \vee (q_6 \wedge \overline{t_7} \wedge \overline{t_{10}})$
$Q_7 = (q_6 \wedge t_7) \vee (q_7 \wedge \overline{t_8} \wedge \overline{t_{11}})$
$Q_0 = \overline{Q_1} \wedge \overline{Q_2} \wedge \overline{Q_3} \wedge \overline{Q_4} \wedge \overline{Q_5} \wedge \overline{Q_6} \wedge \overline{Q_7}$
$q_i = Q_i \quad (i = 0, 1, \dots, 7)$

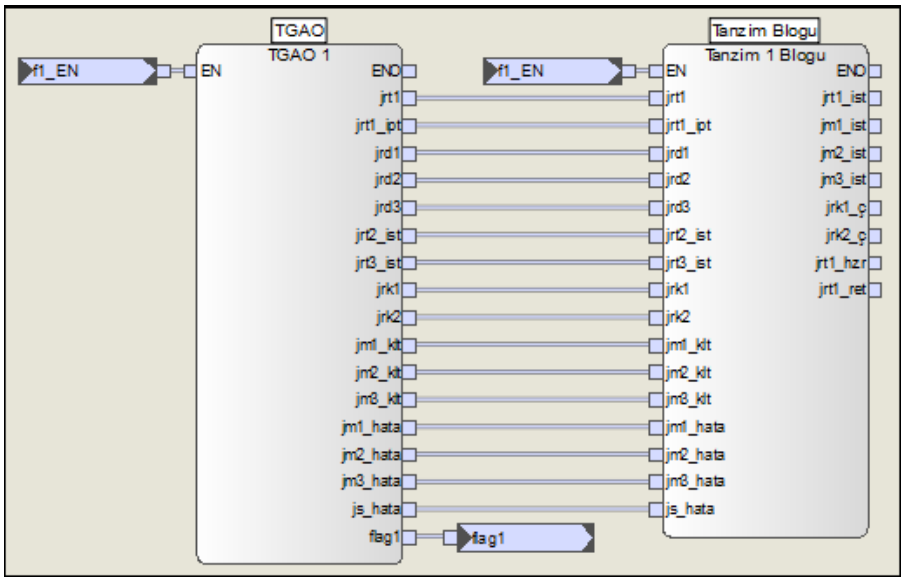
Çizelge 6.6'da tanzim otomatına ait PLC kodu Bölüm 3'te önerildiği şekilde oluşturulmuştur. Çizelge 6.7'de ise tanzim otomatına ait durumlarda aktif hale gelecek çıkışlar gösterilmektedir.

Çizelge 6.7 : Tanzim otomatına ait durumlarda verilecek çıkışlar.

$jrt1_ist = q_1 \vee q_3 \vee q_4 \vee q_5 \vee q_6 \vee q_7$
$jrk1_c = q_1 \vee q_3 \vee q_4 \vee q_5 \vee q_6 \vee q_7$
$jrk2_c = q_1 \vee q_3 \vee q_4 \vee q_5 \vee q_6 \vee q_7$
$jm1_ist = q_1$
$jm2_ist = q_6$
$jm3_ist = q_7$
$jrt1_ret = q_2$
$jrt1_hzr = q_4 \vee q_5$

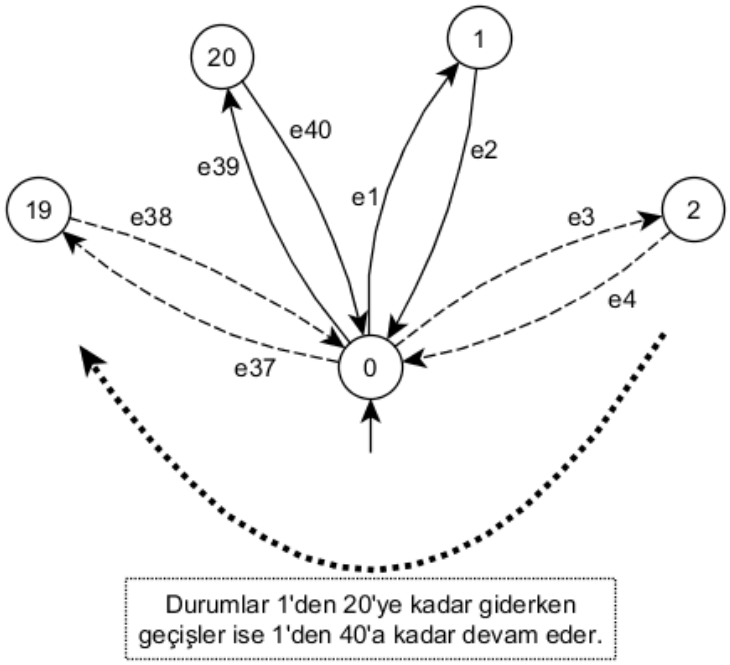
6.4.2 Tanzim giriş anahtarlama algoritması

Tasarlanan jenerik tanzim algoritması örnek istasyonda bulunan bütün olası rotaların tanzimini gerçekleştirebilecek yapıdadır. Bunu gerçekleştirebilmek için gerekli atamaların yapılması gerekmektedir. Tanzim otomatı fonksiyon bloğunu rota sayısı kadar arttırıp, girişlerine her bir rotaya özel olan bitleri atanarak da algoritma tasarımı yapılabilir. Fakat bunun yerine daha az sayıda tanzim otomatı fonksiyon bloğu kullanabilmek için bu atamaların tek bir fonksiyon bloğu içinde yapılması gerekmektedir. Bu fonksiyon bloğunun içinde de bütün rotalara ait atamalar yapılmalıdır. Şekil 6.3'de gösterildiği üzere tanzim fonksiyon bloğuna uygun biçimde bağlanır. Tanzim otomatı rotayla ilgili atamalar sonucunda GAO'daki ilgili bitin mantıksal seviyesine göre durum geçişlerini kontrol ederek, rotayı tanzim etme ya da tanzim talebini reddetme kararını verir.



Şekil 6.3 : TGAO ile tanzim fonksiyon blokların bağlanması.

Öncelikle gelen tanzim talebine göre hangi rotayla ilgili atamaların yapılması gerektiğine karar vermek için bir otomat tasarımı yapmak gerekmektedir. Şekil 6.4'te bu otomat gösterilmektedir.



Şekil 6.4 : Tanzim giriş anahtarlama otomatı.

Bu otomata göre

$$e1 = rt1 \wedge \overline{rt1_ıst} \wedge \overline{rt1_hzr}$$

$$e2 = \overline{rt1_hzr\downarrow \vee\ rt1_ret\downarrow}$$

. .

. .

. .

$$e39 = rt20 \wedge \overline{rt20_ıst} \wedge \overline{rt20_hzr}$$

$$e40 = \overline{rt20_hzr\downarrow \vee\ rt20_ret\downarrow} \qquad \textbf{(6.14)}$$

ile gösterilebilir. 0 numaralı durum başlangıç durumu olmak üzere, bu durumdan başka rota sayısı kadar durum mevcuttur. Gelen tanzim talebine göre bu rotayla ilgili kurulu veya kurulmakta olan bir tanzim talebinin işletilip işletilmediğini kontrol eder. İşletilmemesi halinde ilgili duruma geçerek o rotayla ilgili atamaların yapılabilmesine olanak sağlar. Örneğin, 1 numaralı rota için tanzim talebi geldiğinde (rt1), bu rotaya ait daha önceden bir istek yoksa ($\overline{rt1_ıst}$) ve bu rota daha önceden tanzim edilmemişse ($\overline{rt1_hzr}$) ilgili atamaların yapılabilmesi için otomat ilgili duruma (1) geçmektedir. Bu durumda rota 1 için gereken atamalar yapılmaktadır. İlgili rota için yapılan tanzim talebinin sonucunda ya tanzim edildi (rt1_hzr) ya da tanzim koşullarına aykırı bir durumdan dolayı rota tanzim talebi reddedildi (rt1_ret) bitleri kurulmaktadır. Bu bitlerin düşen kenarları gelinceye kadar GAO ilgili durumda kalır. Bu bitler mantıksal 0 seviyesine düştüğü anda GAO başlangıç durumuna geri döner. Başka bir tanzim talebiyle birlikte aynı prosedür tekrar işletilir.

Tanzim algoritmasında kullanılan değişkenlerin farklı rotalarda farklı bitlerle temsil edilmesi gerekmektedir. Örneğin, tanzim otomatında rd1, rd2 ve rd3 değişkenlerine; rota 1 için sırasıyla 1, 3T, 1T atamaları, rota 5 için sırasıyla 1T, 3T, 1BT atamaları yapılmaktadır. Rota 1 için rd1'e mantıksal 1 atama yapılmasının sebebi rota1'de 3T ve 1T olmak üzere iki ray devresinin kontrol edilmesi gerektiğidir. Rota 1 için rd1 değişkeni fazla olmuş olur. Tasarladığımız tanzim otomatının genel yapısından dolayı bu değişken de tanzim otomatında kontrol edilmektedir. Bunun için rd1'in rota 1 için hep meşgul değilmiş gibi düşünülmesi gerekir ve bu değişkene mantıksal 1 ataması yapılır. Böylece tanzim otomatında bu ray devresi hep boş gözükeceğinden dolayı tanzimi engelleyen bir durum söz konusu olmayacaktır. Aslında rota 1'de 3

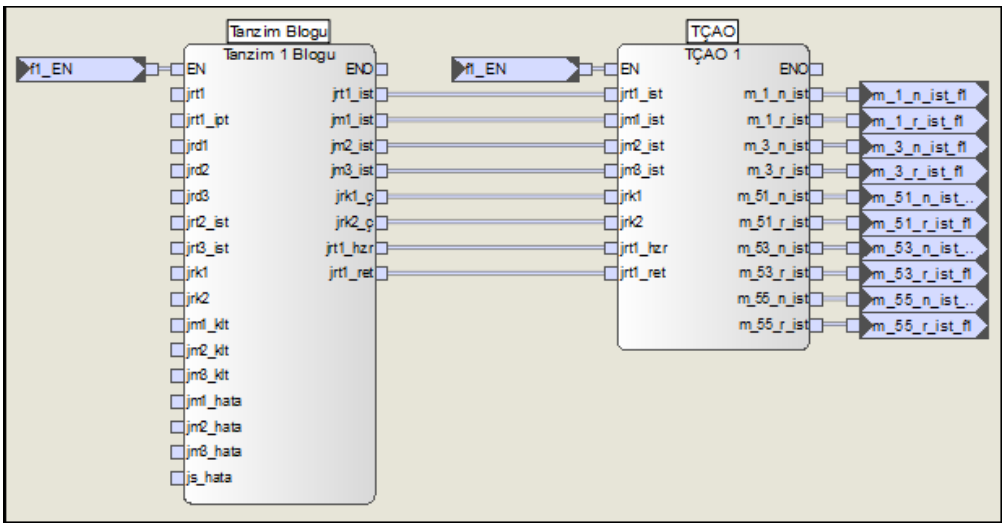
tane ray devresi olmasına rağmen 1ST istasyon ray devresidir. Bu ray devresi dolu olmasına karşın bu rotayı tanzim edebileceğimizden dolayı, tanzim otomatında bu ray devresinin kontrolü yapılmaz. Bu ray devresinin meşgul olması durumu sinyal otomatında kontrol edileceğinden dolayı, tanzim otomatı rotanın hazır (rt_hzr) olduğunu bildirmesiyle sinyal otomatı da bu rota için sarı üzeri kırmızı bildirim verilmesini sağlayacaktır. Sinyal otomatı ileriki bölümlerde detaylı bir biçimde anlatılacaktır.

Bu şekilde bütün rotalar için gerekli bitlerin ataması TGAO'da yapılarak tanzim fonksiyon bloğundan rota sayısı kadar kullanılmasının önüne geçilmiş olunur. Tek bir TGAO fonksiyon bloğuyla bütün rotalara ilişkin atamalar yapılmış olur.

6.4.3 Tanzim çıkış anahtarlama algoritması

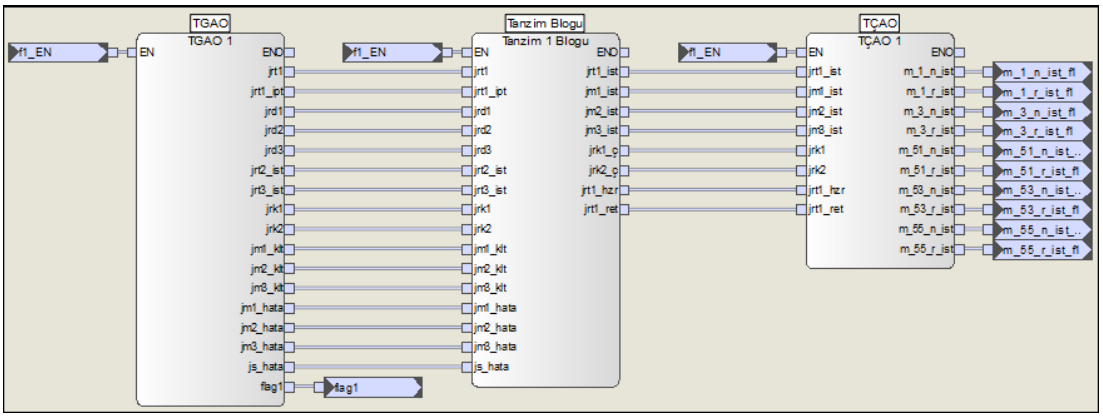
TGAO tarafından yapılan atamalara göre tanzim otomatında tetiklenen geçişler ile durum değiştirilir ve durumlardaki ilgili çıkışlar fonksiyon bloktan çıkış olarak verilmesi gerekmektedir. Bu otomatta gerçekleşen çıkışlar jenerik çıkışlardır ve bu çıkışların da gerçek değerlerine atanması gerekmektedir. Bunu sağlayacak yapı ise bu tezde "Tanzim Çıkış Anahtarlama Otomatı" olarak adlandırılmaktadır. Otomat modeli TGAO'ya ait Şekil 6.4'te gösterilen otomat modeliyle aynıdır, sadece bu durumlardaki yapılacak işlemler farklıdır. Bu otomat sayesinde örneğin, tanzim otomatında ilgili rota için jenerik 1 makasına komut göndermek gerekiyor, ilgili rotanın hangi rota olduğu tespit edilerek hangi makasa hangi yönde komut verilmesi gerektiğinin ayarlaması yapılmaktadır. Diyelim ki rota 1 için tanzim talebi tanzim otomatında işletiliyor olsun. Tanzim için şartlar sağlandı ve makasların güzergâhın gerektirdiği konumlara alınması gerekiyor. Tanzim otomatından TÇAO'ya jm1_ist komutu gönderiliyor. TÇAO sadece bu komutu alarak, 1 numaralı rotanın da işletildiğini bildiği için makas 1'i sapana alması gerektiğine karar verir ve makas 1 otomatına bu yönde komut yollar. Makas 1 otomatından, 1 numaralı makasın istenilen konuma geçerek elektronik kilitlendiğine dair bilgi TGAO sayesinde tanzim otomatına iletilir. Tanzim otomatında durum değişir ve bu sefer jm2_ist komutu TÇAO'ya gönderilir. Rota 1'de tek makas bulunduğundan dolayı TGAO tarafından rota 1 için jenerik 2 makası kilidine mantıksal 1 ataması yapıldığından dolayı tanzim otomatında ilgili geçiş sağlanır. Jenerik 3 makası için de aynı olay söz konusudur. Rota 1 için jenerik 3 makasının da kilit bitine TGAO'da mantıksal 1 ataması yapılır.

Aşağıdaki Şekil 6.5'te tanzim otomatı fonksiyon bloğu ile TÇAO fonksiyon bloğu arasındaki bağlantı gösterilmiştir.



Şekil 6.5 : TÇAO ile tanzim fonksiyon bloklarının bağlanması.

Sonuç olarak KM'den gelen talepler, makas ve sinyal fonksiyon bloklarından gelen bilgiler önce TGAO'ya alınmaktadır. Buradaki algoritma ile yapılan gerekli atamalar tanzim fonksiyon bloğuna gönderilmektedir. Burada ele aldığımız istasyondaki bütün rotaları tanzim edebilecek en fazla eleman sayısına göre hazırlanmış algoritma bulunmaktadır. TGAO fonksiyon bloğundan alınan bilgilere göre tanzim bloğundan jenerik çıkışlar üretilir ve bunlar TÇAO'na yollanır. Burada hangi rotanın işletildiği bilgisi ile birlikte tanzim bloğundan alınan jenerik değerler ilgili gerçek değerlerine atanır. Örneğin rota 2 tanzim talebi gelmiş ve bu fonksiyon bloklarında işletiliyor olsun. Tanzim fonksiyon bloğundan jm1_ist çıkışı verildiğinde TÇAO'da yapılan atamalar sayesinde, bunun 1 isimli makasın normal konumuna getirilmesini sağlayan m_1_n_ist çıkışı kurulur. 1 makası normal konuma alınınca bu bilgi TGAO'ya gelir. Buradaki atamalar ile jm1_klt çıkışı aktif olur ve tanzim fonksiyon bloğuna gönderilir. Burada durum geçişi sağlanır ve bir sonraki durumda diğer makasın komutu (jm2_ist) TÇAO'ya gönderilir. Bu da rota 2 için 3 isimli makasın normal konumuna denk gelmektedir. TÇAO'da bu atama yapılır. Şekil 6.6'da bu üç fonksiyon bloğunun arasındaki bağlantı gösterilmiştir.



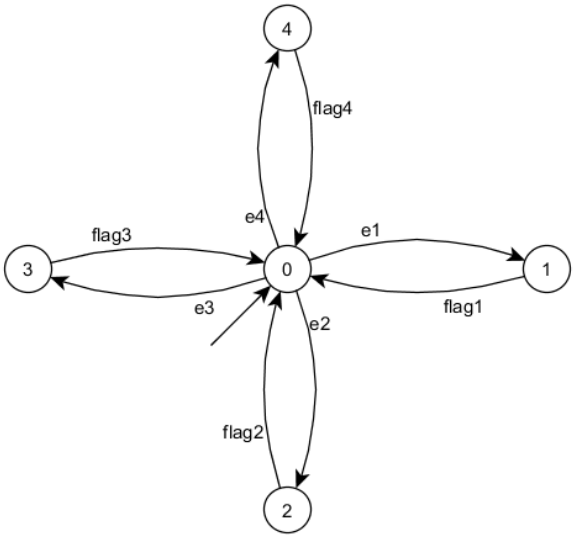
Şekil 6.6 : TGAO, tanzim ve TÇAO bloklarının bağlanması.

6.4.4 Seçici algoritması

Örnek istasyonda bulunan olası 20 rotayı tanzim etmek için jenerik bir yapı kazandırılarak oluşturulan tanzim otomatı sayesinde 20 farklı tanzim otomatı oluşturmaktan kurtulmuş olduk. Ayrıca oluşturduğumuz tanzim fonksiyon bloğundan 20 tane kopyalayıp, giriş ve çıkışlarına her bir rotayla ilgili değişkenleri bağlamak yerine, yine otomatlardan faydalanılarak oluşturulmuş TGAO ve TÇAO sayesinde tek bir tanzim bloğuyla örnek istasyonda bulunan her bir rota tanzim edilebilir özelliği kazandırılmıştır.

Örnek istasyonda aynı anda birden fazla rotanın tanzimi gerçekleşebilir durumdadır. Bunun için biz de tasarladığımız TGAO, TÇAO ve tanzim otomatlarını istasyonda olabilecek maksimum sayıda rota tanzimine göre arttırmamız gerekmektedir.

İstasyonu göz önünde bulundurarak aynı anda maksimum rota tanzimi sayısının 4 (1BT-2ST, 2ST-1DT, 2B-1ST, 1ST-2D) olduğunu görmekteyiz. Bunun için TGAO, TÇAO ve tanzim otomatlarından 4 tane olması gerekir. PLC programında oluşturulan bu 3 fonksiyon bloğunu kopyalayarak, bu üçlü fonksiyon bloğundan 4 adet elde etmiş oluruz. Fakat tanzim talebi geldiğinde hangi bloğun içinde işletileceğini de bir "Seçici" adını verdiğimiz fonksiyon blokla yapmamız gerekmektedir. Bu seçiciye ait otomat modeli aşağıda Şekil 6.7'de gösterilmiştir.



Şekil 6.7 : Seçici otomatı.

Seçici otomatında kullanılan geçişler,

$$e1 = rtx \wedge \overline{\text{f1_EN}} \quad \textbf{(6.15)}$$

$$e2 = rtx \wedge \text{f1_EN} \wedge \overline{\text{f2_EN}} \quad \textbf{(6.16)}$$

$$e3 = rtx \wedge \text{f1_EN} \wedge \text{f2_EN} \wedge \overline{\text{f3_EN}} \quad \textbf{(6.17)}$$

$$e4 = rtx \wedge \text{f1_EN} \wedge \text{f2_EN} \wedge \text{f3_EN} \wedge \overline{\text{f4_EN}} \quad \textbf{(6.18)}$$

Yukarıdaki denklemlerdeki rtx, herhangi bir rota tanzim talebinin yükselen kenar fonksiyonuyla tetiklenen geçişine denk gelmektedir. f1_EN, f2_EN, f3_EN ve f4_EN ise sırasıyla 1, 2, 3 ve 4 numaralı tanzim otomatı fonksiyon bloklarının ENABLE girişlerine bağlanan, seçici fonksiyon bloğunun çıkışından alınan değişkenlerdir. Şekil 6.6'da f1_EN değişkeninin bağlantısı görülmektedir. Ayrıca Şekil 6.7'de görülen flag1, flag2, flag3 ve flag4 ise sırasıyla 1, 2, 3 ve 4 numaralı tanzim otomatı fonksiyon bloklarının 0 numaralı durumdan başka herhangi bir durumundayken mantıksal 1 seviyesinde olan değişkenlerdir. Bu değişkenler ise Şekil 6.6'da görüldüğü gibi TGAO fonksiyon bloklarının çıkışından alınır, seçici fonksiyon bloğunun girişine bağlanır.

Çizelge 6.8 : Seçici otomatına ait PLC kodu.

$Q_1 = (q_0 \wedge e_1) \vee (q_1 \wedge \overline{flag1})$
$Q_2 = (q_0 \wedge e_2) \vee (q_2 \wedge \overline{flag2})$
$Q_3 = (q_0 \wedge e_3) \vee (q_3 \wedge \overline{flag3})$
$Q_4 = (q_0 \wedge e_4) \vee (q_4 \wedge \overline{flag4})$
$Q_0 = \overline{Q_1} \wedge \overline{Q_2} \wedge \overline{Q_3} \wedge \overline{Q_4}$
$q_i = Q_i \quad (i = 0, 1, \dots, 4)$

Çizelge 6.8'de seçici otomatına ait PLC kodu görülmektedir. Çizelge 6.9'da ise bu otomatın ilgili durumlarında kurulması gereken çıkışlar gösterilmektedir.

Çizelge 6.9 : Seçici otomatına ait durumlarda verilecek çıkışlar.

$f1_EN = q_1 \vee flag1$
$f2_EN = q_2 \vee flag2$
$f3_EN = q_3 \vee flag3$
$f4_EN = q_4 \vee flag4$

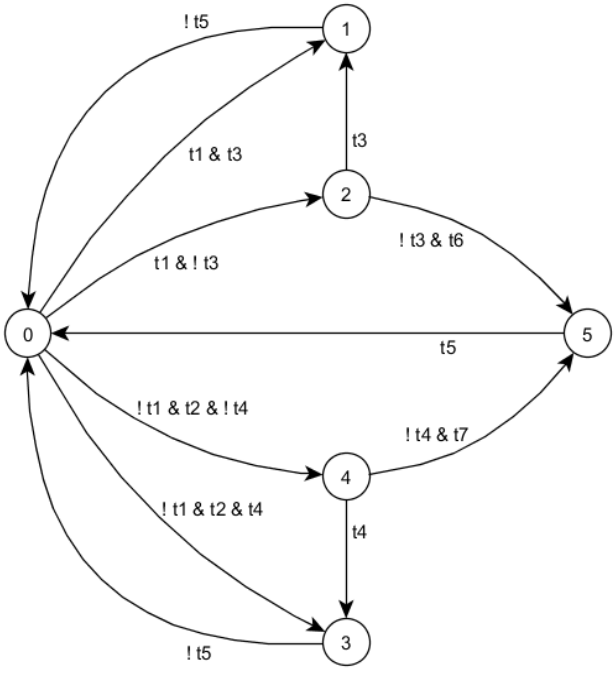
Bu otomatın amacı gelen tanzim talebini daha önce bahsettiğimiz üzere arttırmış olduğumuz 4 adet TGAO-tanzim otomatı-TÇAO üçlülerinden mevcut durumda işlem yapılmayana iletmektir. Başlangıç durumunda bekleyen seçici otomatı tanzim talebi geldiği anda rtx biti mantıksal 1 seviyesine geçer ve e1, e2, e3, e4 geçişlerinden hangisi sağlanıyorsa o duruma geçer. Örneğin herhangi bir tanzim talebi şu anda işletilmiyor olsun. Bu durumda f1_EN, f2_EN, f3_EN ve f4_EN bitleri mantıksal 0 seviyesindedir. Gelen tanzim talebiyle rtx biti mantıksal 1 seviyesine geçecek ve f1_EN biti de mantıksal 0 seviyesinde olduğundan, e1 geçişi sağlanmış olur. Otomat 1 numaralı duruma kurulur. Bu durumda f1_EN çıkışı mantıksal 1 seviyesine geçer ve gelen tanzim talebi ilk tanzim otomatında işletilmeye başlanır. TGAO başlangıç durumundan çıktığı anda flag1 biti kurulur ve seçici otomatı 1 numaralı durumdan 0 numaralı duruma geri döner. Fakat f1_EN biti silinmez çünkü bu bit, seçici otomatı 1 numaralı durumdayken veya TGAO başlangıç durumundan başka bir durumdaysa devamlı kurulu vaziyettedir. Tam bu tanzim talebi işletildiği

sırada başka bir tanzim talebi gelirse, f1_EN biti kurulu, f2_EN biti mantıksal 0 seviyesinde olduğundan dolayı e2 geçişi sağlanır ve bu sefer seçici otomatı 2 numaralı duruma geçer. Bu durumda ise f2_EN biti kurularak, gelen tanzim talebi bu sefer ikinci TGAO-tanzim otomatı-TÇAO üçlüsüne alınır.

4 adet TGAO-tanzim otomatı-TÇAO üçlülerinden hepsi işletiliyorken tanzim talebi geldiğinde ise istasyonda en fazla 4 tanzim olabileceğinden dolayı bu talep dikkate alınmaz.

6.4.5 Makas algoritması

Makas otomatı başlangıç durumundayken ÇAO'dan normale veya sapana geçmesiyle ilgili talebin gelmesiyle birlikte makasın konumunu kontrol eder. Makasın talep edilen konumda olması durumunda makası elektronik olarak kilitler ve meşgul duruma düşürür. Eğer makas talep edilen konumda değilse makas motoruna komut yollar ve makas meşgul duruma düşer. Makasa ait sensörlerden aldığı bilgilerle makasın istenen konuma geçip geçmediğini kontrol eder. Geçtiği anda komut yollamayı keserek makası elektronik olarak kilitler. İlgili rota kilitlerinin düşmesiyle birlikte makas da kilitli durumdan çıkar. Belli bir süre içinde talep edilen konuma makas geçirilemezse, makas otomatı ilgili makas için hata bildirimi verir. Makas ait otomat modeli aşağıdaki Şekil 6.8'de gösterilmiştir.



Şekil 6.8 : Makas otomat modeli.

Makas otomatında kullandığımız geçişler,

$$t1 = m_n_ist \quad \textbf{(6.19)}$$

$$t2 = m_i_ist \quad \textbf{(6.20)}$$

6.19 numaralı denklemdeki m_1_n_ist makası normal çevirme talebini, 6.20 numaralı denklemdeki m_1_i_ist ise makası sapana çevirme talebini belirtir.

$$t3 = i_m_n \wedge \overline{i_m_i} \quad \textbf{(6.21)}$$

$$t4 = \overline{i_m_n} \wedge i_m_i \quad \textbf{(6.22)}$$

Yukarıdaki denklemlerde bulunan i_m_n değişkeni makas normalde sensörü görüyor, i_m_i değişkeni ise makas sapanda sensörü görüyor anlamlarına gelmektedir. Üstü çizili olduklarında ise görmüyor anlamlarına gelmektedir.

$$t5 = \bigvee_{i=1}^{2} rk_(i) \quad (\mathbf{6.23})$$

rk_(i) rota kilidini temsil etmektedir.

$$t6 = 7\ saniye\ (2.durumda) \quad \textbf{(6.24)}$$

$$t7 = 7\ saniye\ (4.durumda) \quad \textbf{(6.25)}$$

6.24 numaralı denklemde makas otomatı 2. durumdayken 7 saniye geçmesi durumunda aktif olacak geçişi, 6.25 numaralı denklem ise 4. durumdayken 7 saniye geçmesi durumunda aktif olacak geçişleri belirtmektedir.

Makas otomatına makas normal veya sapan taleplerinin iletilmesiyle birlikte makasın konumu kontrol edilir. Makas normal talebi gelmiş ve makas normal konumda ise 0 başlangıç durumundan 1 numaralı duruma geçilir. Burada makas elektronik olarak kilitlenir ve meşgul duruma geçer. Rota kilidi kalkana kadar bu durumda kalır. Başlangıç durumunda makas normal talebi gelmiş ve makas normal konumda değil ise 2 numaralı duruma geçilir. Burada makas motoru çalıştırma komutu yollanır ve meşgul duruma geçer. Makasın normal konuma geçmesiyle birlikte makas motoruna yollanan komut kesilir ve otomat 1 numaralı duruma geçer. 2 numaralı durumda iken makas 7 saniye boyunca normal konuma geçemezse, otomat 5 numaralı duruma gider. Burada makas hata biti çektirilir ve bu bit tanzim otomatında kontrol edilerek güzergâh tanzim talebi reddedilir. Bu makası içeren başka bir güzergâh tanzim talebinin alınmasıyla birlikte makas hatadan kurtularak başlangıç durumuna geçer.

Makas otomatı tasarımı da her makas için aynıdır, sadece girişlerine bağlanan değişkenler makasa göre farklılık gösterir.

Çizelge 6.10 : Makas otomatına ait PLC kodu.

$Q_1 = (q_0 \wedge t_1 \wedge t_3) \vee (q_2 \wedge t_3) \vee (q_1 \wedge t_5)$
$Q_2 = (q_0 \wedge t_1 \wedge \overline{t_3}) \vee \left(q_2 \wedge \overline{t_3} \wedge \left(\overline{\overline{t_3} \wedge t_6}\right)\right)$
$Q_3 = (q_0 \wedge \overline{t_1} \wedge t_2 \wedge t_4) \vee (q_4 \wedge t_4) \vee (q_3 \wedge t_5)$
$Q_4 = (q_0 \wedge \overline{t_1} \wedge t_2 \wedge \overline{t_4}) \vee \left(q_4 \wedge \overline{t_4} \wedge \left(\overline{\overline{t_4} \wedge t_7}\right)\right)$
$Q_5 = (q_2 \wedge \overline{t_3} \wedge t_6) \vee (q_4 \wedge \overline{t_4} \wedge t_7) \vee (q_5 \wedge \overline{t_5})$
$Q_0 = \overline{Q_1} \wedge \overline{Q_2} \wedge \overline{Q_3} \wedge \overline{Q_4} \wedge \overline{Q_5}$
$q_i = Q_i \quad (i = 0, 1, \dots, 5)$

Çizelge 6.10'da makas otomatına ait PLC kodu görülmektedir. Çizelge 6.11'de ise bu otomatın ilgili durumlarında kurulması gereken çıkışlar gösterilmektedir.

Çizelge 6.11 : Makas otomatına ait durumlarda verilecek çıkışlar.

$q_m_n = q_2$
$q_m_i = q_4$
$m_mesgul = q_1 \vee q_2 \vee q_3 \vee q_4$
$m_kilit = q_1 \vee q_3$
$m_hata = q_5$

6.4.6 Sinyal algoritması

Tanzim otomatından rotanın hazır olduğuna dair bilgi sinyal otomatına gelmesiyle birlikte güzergâh ile ilgili kontroller yapılarak, ilgili renk bildirimi verilmesi için sinyale komut yollanır. Sinyale komut yollandıktan sonra ilgili renk bildirimini verdiğine dair indikasyon beklenir. Belirlenen bir süre içinde bu indikasyon alınmaz ise sinyal hataya düşer. Sinyalin hataya düşmesiyle tanzim otomatında devam etmekte olan tanzim işlemi iptal edilir.

Sinyale ait fonksiyon bloğunun oluşturulmasında otomat modeli kullanılmamıştır. Her bir sinyal için tablolar oluşturulmuştur. Bu tablolarda sinyalin uygun renk bildirimini vermesi için gerekli şartlar bulunmakta ve bu şartlara göre hangi renk bildirimini vermesi gerektiği belirtilmektedir. Aşağıdaki Çizelge 6.12'de örnek istasyonda bulunan 2D sinyali için hazırlanan tablo gösterilmektedir. Bu tablodan her bir sinyal için hazırlanır ve sinyal algoritması bu tablolar vasıtasıyla oluşturulur.

Çizelge 6.12 : 2D sinyal tablosu.

rt1_hzr	1	1	1	0	0	0	0	0	0
rt2_hzr	0	0	0	1	1	1	0	0	0
rt3_hzr	0	0	0	0	0	0	1	1	1
I_R_3T	1	1	1	1	1	1	1	1	1
I_R_1T	1	1	1	-	-	-	-	-	-
I_R_1ST	1	1	0	-	-	-	-	-	-
I_R_2ST	-	-	-	1	1	0	-	-	-
I_R_3ST	-	-	-	-	-	-	1	1	0
I_S_52DA	-	-	-	S veya Y	K veya SS veya SY	-	-	-	-
I_S_54D	S veya Y	K	-	-	-	-	-	-	-
I_S_52DB	-	-	-	-	-	-	S veya Y	K	-
Q_S_2D	**SY**	**SS**	**SK**	**Y**	**S**	**SK**	**SY**	**SS**	**SK**

Çizelge 6.12'de, "1"; sol sütunda yer alan değişkenlerin kurulu olmasını, "0"; bu değişkenlerin mantıksal 0 seviyesinde olduğunu temsil etmektedir, "-" ise fark etmez anlamına gelmektedir. I_R_3T, I_R_1T, I_R_1ST, I_R_2ST, I_R_3ST sırasıyla 3T, 1T, 1ST, 2ST ve 3ST ray devrelerinin indikasyonlarıdır. Bu ray devrelerinden mantıksal olarak 1 indikasyonu alınması meşgul olmadıkları anlamına gelmektedir.

I_S_52DA, I_S_54D, I_S_52DB sırasıyla 52DA, 54D ve 52DB sinyallerine ait indikasyonlardır. Bu sinyallerin verdikleri bildirimlere göre renk indikasyonları alınır. Bundan dolayı yanlarına Y (yeşil), S (sarı), K (kırmızı), SY (sarı üzeri yeşil), SS (sarı üzeri sarı), SK (sarı üzeri kırmızı) gibi harfler getirilir. Q_S_2D ise 2D sinyalinin komutunu temsil eden değişkendir.

Sinyale uygun renk bildirim komutunu yollayabilmek için sinyalin açılabileceği rotaların tanziminin kontrolü, tanzimli rotalarda bulunan ray devrelerinin meşguliyetinin kontrolü ve tanzimli rotanın sonunda bulunan bir sonraki rotanın sinyalinin indikasyonunun kontrolü yapılması gerekmektedir. Bu kontrollerin sonucunda ilgili sinyale gönderilmesi gereken komut tablonun en alt satırında belirtilir.

Tablonun her bir sütunu bir geçişe denk gelmektedir. Örneğin, ilk sütun için rt1_hzr=1, I_R_3T, I_R_1T, I_R_1ST ray devrelerinden 1 indikasyonu alınıyor yani bu ray devreleri meşgul değil ve aynı istikametteki bir sonraki sinyal olan 54D sinyalinden sarı veya yeşil indikasyon alınıyor. Bu durumda 2D sinyaline sarı üzeri yeşil komutu gönderilir. Bu şartların hepsi fonksiyon blok içinde bir "VE" mantıksal kapısına dahil edilerek bu kapının çıkışına Q_S_2D_SY değişkeni bağlanır. Fakat Q_S_2D_SY tablodan görüldüğü üzere hem ilk sütunda hem de 7. sütunda olmak üzere iki yerde verildiğinden dolayı, iki tane "VE" mantıksal kapılarının çıkışı bir adet "VEYA" mantıksal kapısının girişlerine bağlanır. Bu kapının çıkışı da Q_S_2D_SY değişkenine bağlanır.

İlgili sinyale komut gönderilmesiyle birlikte belli bir süre içinde indikasyonunun gelmesi beklenir. Sinyalin gönderilen renk bildirimine ait indikasyonun alınamaması durumunda sinyal hatasına ait s_hata biti çektirilerek, sinyalde hata olduğunun bilgisi tanzim otomatına gönderilir ve tanzim sonlandırılır.

Böylece örnek istasyonda bulunan her bir sinyal için tablo oluşturup, bunlar uygun bir biçimde kodlanarak sinyal fonksiyon bloğunun içine yazılır.

7. SİMULATÖR TASARIMI

Tasarlanan algoritmanın testinin yapılabilmesi için bir yazılım simülatörüne ihtiyaç duyulmaktadır. PLC programlarındaki simülatörlerde bitlerin takibi kolay bir şekilde gerçekleştirilememektedir. Bu simülatör sayesinde algoritmanın test aşaması daha hızlı bir şekilde yapılabilmektedir.

7.1 Modbus Haberleşmesi

Modbus, PLC'lerde kullanılmak üzere 1979'da Modicon tarafından ileri sürülmüş seri iletişim protokolüdür (URL-1).

Modbus haberleşme sistemi, endüstriyel ortamlarda çalışan sistemlerin aralarındaki iletişimi sağlamak için geliştirilmiştir. Modbus sistemi, daha basit yapıdaki donanımlarla gerçekleştirilebilir. Çok hızlı olmayan bir haberleşme kullanıldığından dolayı birçok cihaz Modbus sistemine uyumlu olarak geliştirilebilir. RS232 olarak bilinen standart PC seri portu üzerinden ya da RS485 arabirimi üzerinden ya da Ethernet ağı üzerinden gerçekleştirilebilir (URL-2).

Modbus, temelde veri alışverişi yapısını tanımlar. Bu alışverişi gerçekleştirebilecek herhangi fiziksel hat üzerinden de Modbus haberleşme sağlanabilir. Veri alışveriş yapısı halka açık olarak sunulmuştur ve internet vb. kaynaklardan kolaylıkla bulunabilir. Bu yapıya uygun olarak programlanmış haberleşme yeteneği olan her türlü cihaza uygulanabilir (URL-2).

Bu tez çalışmasında HIMA HIMATRIX PLC'ler kullanılmıştır. PLC'ye kodu yüklemeden önce değişkenlerin adreslemesini yapmamız gerekmektedir. Bunun için Silworx programında "Protocols" menüsünden "Modbus Slave Set" eklememiz gerekmektedir. Buradaki "Bit Variables" sekmesinin altındaki ilgili yerlere, PLC'ye giriş olarak aldığımız ve PLC'den çıkış olarak verdiğimiz değişkenlerin adreslemesi yapılmaktadır. Simülatör programında da aynı adresleri kullanacağımızdan dolayı bu adresleri saklamamız gerekmektedir.

7.2 UDP

TCP/IP protokol takımının iki aktarım katmanı protokolünden birisidir. Bu protokolde veriler bağlantı kurulmadan yollanır.

Gelişmiş bilgisayar ağlarında paket anahtarlamalı bilgisayar iletişiminde bir veri bloğu modu oluşturabilmek için UDP protokolü yazılmıştır. Bu protokol, minimum protokol mekanizmasıyla bir uygulama programından diğerine mesaj göndermek için bir prosedür içerir. Bu protokol 'transaction' yönlendirmelidir. Paketin teslim garantisini isteyen uygulamalar TCP protokolünü kullanır.

Geniş alan ağlarında (WAN) ses ve görüntü aktarımı gibi gerçek zamanlı veri aktarımlarında UDP kullanılır.

UDP bağlantı kurulum işlemlerini, akış kontrolü ve tekrar iletim işlemlerini yapmayarak veri iletim süresini en aza indirir.

UDP ve TCP aynı iletişim yolunu kullandıklarında UDP ile yapılan geçek zamanlı veri transferinin servis kalitesi TCP'nin oluşturduğu yüksek veri trafiği nedeniyle azalır.

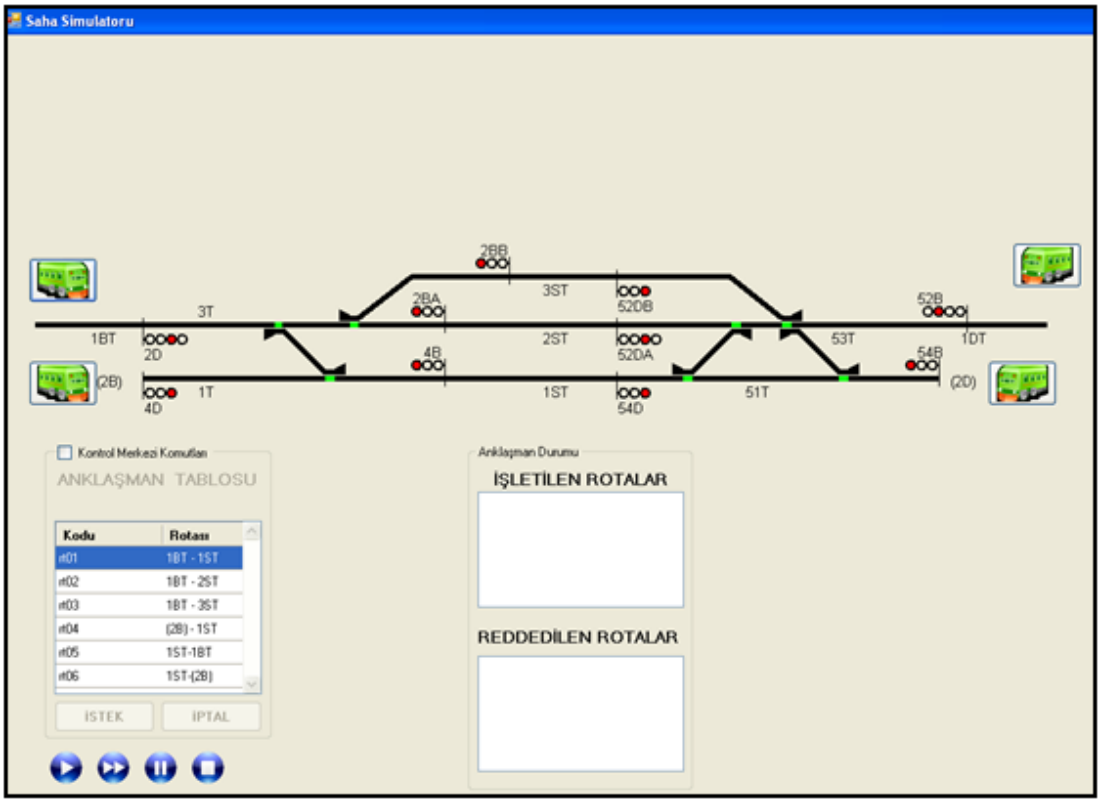
DNS, TFTP, ve SNMP protokolleri UDP'yi kullanan protokollerden bazılarıdır. UDP, ağ üzerinde fazla bant genişliği kaplamadığından dolayı uygulama programcıları tarafından TCP'ye göre daha fazla tercih edilir.

UDP güvenilir olmayan bir aktarım protokolüdür. Bu protokolde, ağ üzerinden veri paketi gönderilir fakat bu paketin istenilen yere gidip gitmediği takip edilmez. UDP, paketin yerine ulaşıp ulaşmayacağına onay verme yetkisi yoktur. UDP üzerinden güvenilir şekilde veri göndermek isteyen bir uygulama bunu kendi yöntemleriyle yapmak zorundadır.

Bu bölümde URL-3'ten yararlanılmıştır.

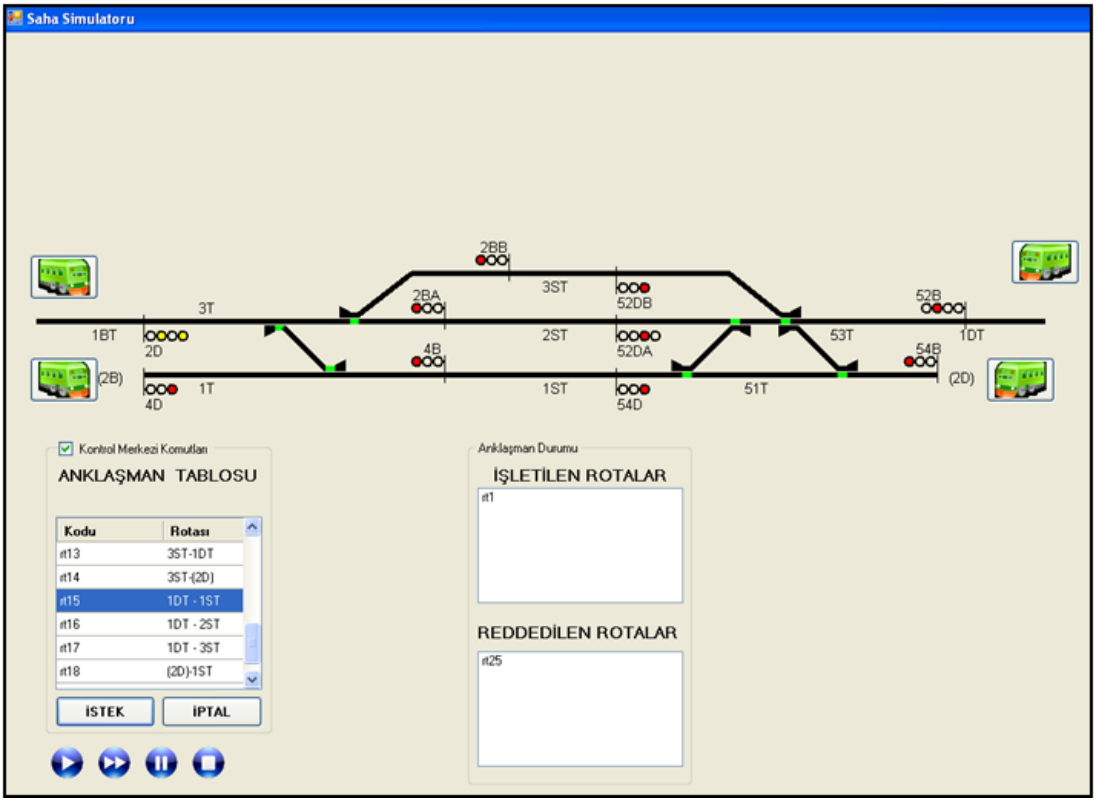
7.3 Programın Tanıtılması

Bölüm 6'da ele alınan sahaya ait simülatör resmi Şekil 7.1'de gösterilmektedir.



Şekil 7.1 : Simülatör.

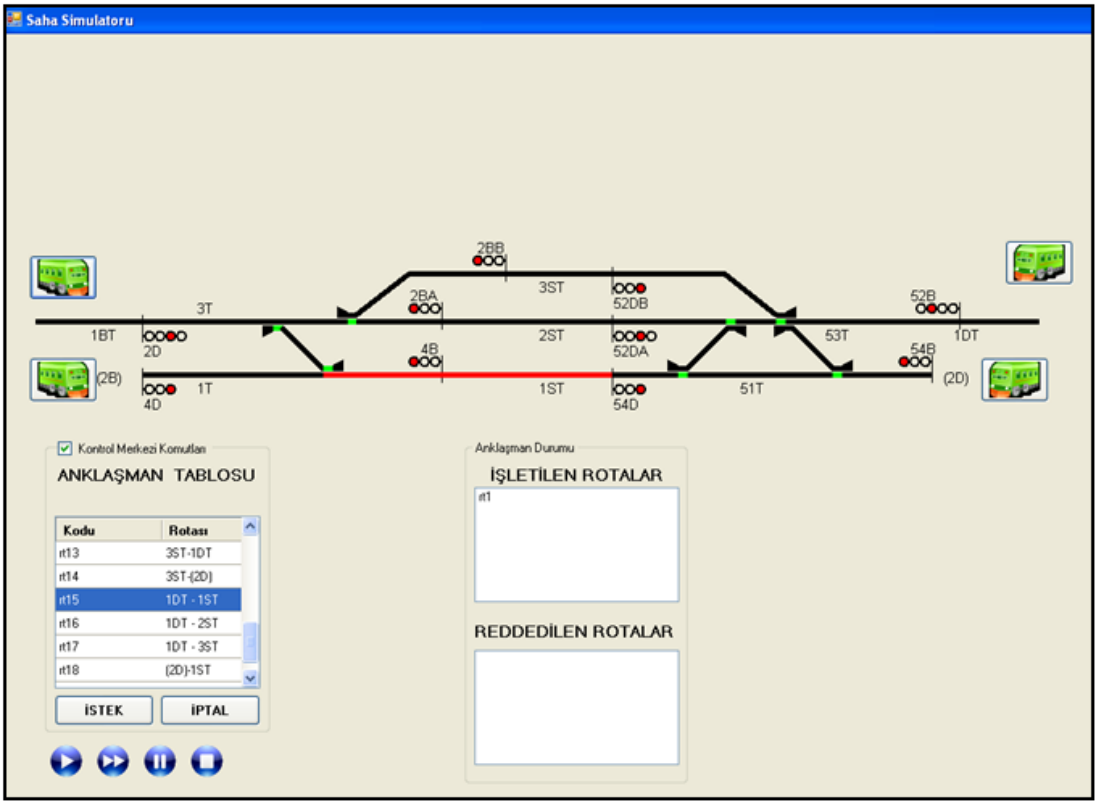
Bu simülatörde, kontrol merkezi tarafından gönderilecek olan tanzim taleplerini karşılamak üzere rt1'den başlayıp rt20'ye kadar giden seçeneklerden biri işaretlenip alt tarafındaki "İSTEK" butonuna basılır. Böylece ilgili rota için tanzim talebi yapılmış olur ve anklaşman algoritmalarının yüklü olduğu PLC'ye bu talep ulaştırılır. AS tarafından değerlendirilen talebin uygun bulunması durumunda Şekil 7.2'de gösterildiği gibi "İŞLETİLEN ROTALAR" kısmında bu rotanın ismi yazar. İlgili rotada bulunan makaslar uygun konuma alındıktan sonra güzergâha ait sinyal açılır. Aşağıdaki Şekil 7.2'de rota 1 için tanzim talebi yapılmış, bu talep anklaşman tarafından uygun bulunmuştur. Anklaşman tarafından uygun bulunduğu bilgisi simülatöre yollanarak "İŞLETİLEN ROTALAR" kısmında 1 nolu rotanın işletildiği bilgisinin görünmesi sağlanmıştır. Anklaşman tarafından 2D sinyaline ilgili kontroller sonucu sarı üzeri sarı komut yollanmış ve rota 1 tanzim edilmiştir.



Şekil 7.2 : İşletilen rotada sinyalin açılması.

Rota 1 tanzim edildikten sonra rota 15 tanzim talebi simülatörden seçilerek anklaşmana gönderilmiştir. Rota 1 ile rota 15 çakışan rotalar olduklarından bu talep anklaşman tarafından kabul edilmez ve simülatöre bu bilginin iletilmesi sağlanır. Şekil 7.2'de görüldüğü gibi "REDDEDİLEN ROTALAR" kısmında bu rotanın ismi yazar.

Sinyalin açılmasının ardından simülatörde üzerinde tren simgesi bulunan butonlar yardımıyla ilgili güzergâh için tren yollanır. Trenin ilgili güzergâhta ilerlemesi, ray devrelerinin renginin kırmızıya dönmesi sayesinde izlenebilir. Trenin ray devrelerine girmesiyle birlikte ray devreleri kırmızı rengini alır. Sinyaller vasıtasıyla trenin ilerletilmesi sağlanır. Önündeki sinyalin kırmızı olması durumunda sinyalin önünde bekler ve sinyalin trenin ilerlemesine izin verecek uygun bir bildirim vermesi ile hareketine devam eder. Şekil 7.3'te trenin 3T ve 1ST ray devrelerinde bulunduğu ray devrelerinin kırmızı renk almasından anlaşılmaktadır. 54D sinyali kırmızı bildirim verdiğinden trenin ilerlemesi engellenir.



Şekil 7.3 : Trenin kırmızı bildirim veren sinyale kadar ilerlemesi.

"Kontrol Merkezi Momutlarının" yer aldığı kısımdan rota 9'u seçip "İSTEK" butonuna basılmasıyla, anklaşmanda yapılan kontroller sonucu 9 nolu rotanın tanzimi gerçekleştirildiği takdirde 1ST ray devresinde bulunan trenin tanzim edilen rotada ilerlemesi sağlanır.

Bu bölümde gösterilen simülatör uygulamasının kodları, Ahmet Kuzu'nun UDSP projesi kapsamında Mithatpaşa istasyonu için geliştirdiği simülatörden alınmış ve tezde ele alınan istasyon hattına uyarlanmıştır.

8. SONUÇ VE ÖNERİLER

Bu tez çalışmasında, raylı ulaşım sistemlerinde kullanılan sinyalizasyon sistemleri temel bileşenleri tanıtılmış, anklaşman algoritması tasarımı gerçekleştirilmiş ve bu tasarımın kolaylaştırılması yönünde çalışmalar gösterilmiştir. Bu algoritma tasarımında biçimsel yöntemlerden otomatlar kullanılmıştır. Otomat modelinden PLC kodu elde etmek için matematiksel denkleme dayanan bir gerçekleme yöntemi tanıtılmıştır. Bu yöntem sayesinde uygulamada sıklıkla karşılaşılan literatürde çığ etkisi olarak adlandırılan sorun çözülmektedir. Literatürde bu sorun standart yöntemlerle sistematik olmaktan uzak bir şekilde gerçeklenmeye çalışılmıştır. Bu tezde ise bazı durumlarda bu gerçeklemenin de problem çıkarabileceği bir örnek verilmiştir. Ayrıca otomatlarda başlangıç durumuna kurma analizi yapılmıştır. Bunun sonucunda istenilen davranışın elde edilmesinde sakıncalı durumlar tespit edilmiş, bunlar örnek üzerinden gösterilmiş ve bu sorunun çözümü için yeni bir yaklaşım önerilmiştir.

Anklaşman algoritması tasarımında, üzerinde saha elemanları isimleri olan, konumları ve özellikleri belirtilen örnek bir istasyon hattı resmi çizilmiştir. Bu saha resmine göre anklaşman tablosu oluşturma işlemi detaylı bir biçimde anlatılmıştır. Oluşturulan bu anklaşman tablosunu gerçekleyecek modüler bir algoritma tasarımı gerçekleştirilmiş olup, bu algoritmanın geliştirilen yazılım simülatörüyle test edilmesi imkânı sunulmuştur.

Bu tezde, otomatlarla gerçekleştirilen anklaşman algoritması tasarımının daha kolay ve hızlı bir şekilde yapılabilmesi yönünde de çalışmalar yapılmıştır. Otomatlar için tanımlanan gerçekleme yöntemi programlanabilir olma özelliğine de sahiptir. Bu özellik sayesinde tasarlanan otomat modeline karşılık gelen PLC kodunun otomatik üretilmesi için Microsoft Visual Basic ile bir yazılım geliştirilmiştir. Bu uygulama, otomat modelinde ilgili geçişlerle gidilecek durumları giriş olarak almaktadır ve buna göre PLC kodu oluşturulmaktadır. Gelecekte yapılabilecek bir çalışmada, otomat modeli çizimi de bu uygulamanın bir parçası haline getirilmesiyle daha kullanışlı bir uygulama mümkün kılınabilir. Ayrıca anklaşman algoritmalarında kullanılacak değişken isimlerinin standartlaşması ve hızlı bir şekilde PLC

programında tanımlanabilmeleri için de bir yazılım geliştirilmiştir. Bu yazılım ile PLC programında kullanılacak anklaşman algoritmasına ait değişken isimlerinin yer aldığı belge oluşturulmakta ve PLC programına bu belgenin içeriği alınmaktadır. Bu değişken isimleri belgesi HIMA Silworx programı için ".csv" formatında oluşturulmaktadır. Program çıktısının başka PLC modellerinde kullanılabilmesi, ileride yapılacak çalışmalarla mümkün hale gelebilecektir.

KAYNAKLAR

Boel R.K. ve Schuppen J.H.V., 2002, "Decentralized failure diagnosis for discrete-event systems with costly communication between diagnosers." In Proc. WODES'02: 6th Work. on Discrete Event Systems, Zaragoza, İspanya, s: 175–181, Ocak.

Calame J.R., Goga N., Ioustinova N. ve Jaco van de Pol, 2006, "TTCN-3 Testing of Hoorn-Kersenboogerd Railway Interlocking," *Electrical and Computer Engineering Canadian Conference on*, s: 620-623.

Cassandras G.C. and Lafortune S., 1999, Introduction to discrete event systems. Kluwer Academic Publishers, Massachusetts, USA.

Chen E. ve Lafortune S., 1991. Dealing with blocking in supervisory control of discrete event systems, *IEEE Transactions on Automatic Control*, 25(1), 206-230.

Debouk R., Lafortune S., ve Teneketzis D., 2000, "Coordinated decentralized protocols for failure diagnosis of discrete-event systems." Discrete Events Dynamical Systems, 20:33–79.

Durmuş M. S., Söylemez M. T., 2009, "Railway Signalization and Interlocking Design via Automation Petri Nets," Proceedings of the 7th Asian Control Conference, Hong Kong, China, August 27-29.

Fabian M. and Hellgren A., 1998. PLC-based implementation of supervisory control for discrete event systems, *37th IEEE conference on Decision & Control*, Tampa, Florida, USA.

Goddard E., 2008, "Electric traction systems overview of signaling and train control systems", Electric Traction Systems, *IET Professional Development Course on*, s: 314-322.

Gündoğdu F. ve Söyler H., 2005: Raylı sistemlerde emniyet standartları ve makas otomasyon sistemine uygulanması, *Elektrik – Elektronik – Bilgisayar Mühendisliği 11. Ulusal Kongresi ve Fuarı*, İstanbul, 22– 25 Eylül.

Gündoğdu F. ve Söyler H., 2008: Demiryolu sinyalizasyon sistemlerinde tasarım kriterleri ve fail-safe kavramı , *Kentiçi Raylı Sistemler Bülteni*, sayı 8.

Hasdemir T., 2008, Ayrık olay sistemlerinin tasarımı ve kontrolü için yeni bir gerçekleme ve otomatik kod üretme yöntemi, İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Doktora Tezi, Eylül, İstanbul.

Ingleby M. ve Mee D., 1995, "A calculus of hazard for railway signaling", Industrial-Strength Formal Specification Techniques, Proceedings Workshop on, s: 146-158.

Jiang S. ve Kumar R., 2004, "Failure diagnosis of discrete-event systems with linear-time temporal logic specifications," *Automatic Control IEEE*

Transactions on, vol. 49, s: 934-945.

Lai S., Nessi D., Cabasino M.P., Giua A., Seatzu C., 2008, "A Comparison Between Two Diagnostic Tools Based on Automata and Petri Nets", *Proceedings of the 9th International Workshop on Discrete Event Systems*, Göteborg, Sweden, Mayıs 28-30.

Lunze J. ve Schroder J., 2004, "Sensor and actuator fault diagnosis of systems with discrete inputs and outputs," Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, vol. 34, s: 1096-1107.

Mirabadi A., Yazdı M.B., 2009, "Automatic generation and verification of railway interlocking control tables using FSM and NUSMV," Iran University of Science and Technology, School of Railway Engineering (S.R.E), Iran, Tehran, Narmak.

Özdemir S., 2000, Demiryollarında sinyalizasyon, Kocaeli Üniversitesi, Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, Ocak, Kocaeli.

Ramadge P.J. ve Wonham W.M., 1987. Supervisory control of a class of discrete event processes, *SIAM Journal of Control and Optimization*, 25(1), 206-230.

Ramadge P.J. ve Wonham W.M., 1987. On the supremal controllable sublanguage of a given language, *SIAM Journal of Control and Optimization*, 25(3), 637-659.

Sampath M., Lafortune S., ve Teneketzis D., 1998, "Active diagnosis of discrete-event systems," Automatic Control, IEEE Transactions on, vol. 43, s: 908-929.

Sampath M., Sengupta R., Lafortune S., Sinnamohideen K., ve Teneketzis D., 1995, "Diagnosability of discrete-event systems," *Automatic Control IEEE Transactions on*, vol. 40, s: 1555-1575.

Söyler H., Açıkbaş S., 2005, Raylı toplu taşımada sinyalizasyon sistemleri, *Elektrik – Elektronik – Bilgisayar Mühendisliği 11. Ulusal Kongresi ve Fuarı*, İstanbul, 22– 25 Eylül.

Tiryaki Ö., 2007, Biçimsel dillerden endüstriyel işlemcilere otomatik kod üretme: otomat yaklaşımı, İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, Ocak, İstanbul.

Tombs D., Robinson N., Nikandros G., 2002, "Signaling control table generation and verification", *Conference on Railway Engineering Wollongong*, 10-13 November.

Türk S., Sonat A., Kuzu A., Söylemez M.T., Songüler Ö., 2010, Raylı ulaşım sinyalizasyon sistemleri için anklaşman tablosundan otomatik anklaşman algoritması üreteci, *Otomatik Kontrol Ulusal Toplantısı (gönderildi)*, Gebze, İstanbul

Url-1 <*http://en.wikipedia.org/wiki/Modbus*>, alındığı tarih 26.04.2010.

Url-2 <*http://www.mrs.com.tr/teknikbilgi/modbus.htm*>, alındığı tarih 26.04.2010.

Url-3 <*http://tr.wikipedia.org/wiki/UDP*>, alındığı tarih 26. 04.2010.

Wang Y., Jia L. ve Qin Y., 2003, "Application of entity automata to railway

transportation system modeling and simulation," *Intelligent Transportation Systems Proceedings. IEEE*, s: 1463-1466 vol.2.

Yong Q., Ying-Jie W. ve Li-Min Jia, 2003, "Hybrid cellular automata model for railway transportation system and its implementation on GIS," *Intelligent Vehicles Symposium Proceedings. IEEE*, s: 543-546.

Zad S.H., Kwong R., ve Wonham W., 2003, "Fault diagnosis in discrete-event systems: framework and model reduction," *Automatic Control IEEE Transactions on*, vol. 48, s: 1199-1212.

ÖZGEÇMİŞ

Ad Soyad: Arcan Sonat

Doğum Yeri ve Tarihi: İstanbul - 24.08.1985

Lisans Üniversite: Yıldız Teknik Üniversitesi

Yayınlar:

- Türk S., **Sonat A.,** Kuzu A., Söylemez M.T., Songüler Ö., 2010, Raylı ulaşım sinyalizasyon sistemleri için anklaşman tablosundan otomatik anklaşman algoritması üreteci, *Otomatik Kontrol Ulusal Toplantısı (gönderildi)*, Gebze, İstanbul