T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

MİKRODENETLEYİCİ TABANLI MODÜLER KAVŞAK KONTROL CİHAZI TASARIMI

YÜKSEK LİSANS TEZİ

Elektrik ve Elektronik Müh. Önder BERBEROĞLU

Enstitü Anabilim Dalı	**:**	**ELEKTRİK-ELEKTRONİK MÜH.**
Enstitü Bilim Dalı	**:**	**ELEKTRONİK**
Tez Danışmanı	**:**	**Yrd. Doç. Dr. Ayhan ÖZDEMİR**

Şubat 2008

T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

MİKRODENETLEYİCİ TABANLI MODÜLER KAVŞAK KONTROL CİHAZI TASARIMI

YÜKSEK LİSANS TEZİ

Elektrik ve Elektronik Müh. Önder BERBEROĞLU

Enstitü Anabilim Dalı : ELEKTRİK-ELEKTRONİK MÜH.

Enstitü Bilim Dalı : ELEKTRONİK

Bu tez 08 / 02 /2008 tarihinde aşağıdaki jüri tarafından Oybirliği ile kabul edilmiştir.

Yrd. Doç. Dr. Ayhan ÖZDEMİR	**Prof. Dr. Ertan YANIKOĞLU**	**Yrd. Doç. Dr. Ali GÜLBAĞ**
Jüri Başkanı	**Üye**	**Üye**

TEŞEKKÜR

Bu proje kapsamında desteğini ve yardımını hiç esirgemeyen, programın hazırlanma safhasında benimle sabahlara kadar uykusuz kalan arkadaşım Elektrik ve Elektronik Yüksek Mühendisi Muhammed Bıyıklı'ya (Belbim A.Ş.), yardımlarından ve yönlendirmelerinden dolayı danışmanım Yrd. Doç. Dr. Ayhan Özdemir'e (Sakarya Üniversitesi), tecrübelerini benimle paylaşan Elektronik Mühendisi Orhan Nergis'e (Adapazarı Büyükşehir Belediyesi), Adapazarı Büyükşehir Belediyesi Trafik ve Sinyalizasyon Birimi elemanlarına, manevi desteklerinden dolayı aileme ve Nurdan Türkseven'e teşekkür ederim.

İÇİNDEKİLER

SİMGELER VE KISALTMALAR LİSTESİ

PLC : Programlanabilir Lojik Kontrolör

MUTCD : DüzenliTrafik Kontrol Devreleri Klavuzu

C : Periyot Süresi

Y_i : Boşaltma Fasılası

g_i : Yeşil Süre(Fasıla)

r_i : Kırmızı Süre(Fasıla)

t_L : Kayıp Süre

g : Efektif Yeşil

r : Efektif Kırmızı

s : Doyum Akımı

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

ÖZET

Anahtar kelimeler: Modüler, özel durumlara göre ayarlanabilen

Şehirlerin büyümesi ile birlikte trafik çilesi de içinden çıkılmaz bir hale gelmektedir. Günümüzde eğer tek merkezden yönetilen büyük bir sistem kurmamışsanız trafik sinyalizasyonu her kavşak için ayrı bir sistem kurularak yapılmaktadır. Sistem kavşağa kurulduktan sonra da herhangi bir değişiklik yapabilmek zor olmakta veya bunun için sistemi kuran firmaya ihtiyaç duyulmaktadır.

Bu gibi durumlarda daha verimli çalışabilmek için her kavşağa uyum sağlayabilecek ve sistem teknikerleri tarafından gerektiğinde değişiklikler yapılabilecek bir sistem çok daha kullanışlı olmaktadır. Kurulan sistemlerin az yer kaplaması ve fazla maliyet gerektirmemesi de tercih sebebidir.

THE DESİGN OF MODULER CROSSROADS CONTROL SYSTEM BASED ON MICROCONTROLLER

SUMMARY

Key Words: Moduler, Adjustable According To Special States

After development of cities, traffic became problem for people. If great system managed from aone centre wasn't established traffic signalization is made with different system for every crossroads. After the system was installed it's very difficult to change anything. The company establishing the system is needed for changing.

At this situations, the system that is compatible with every crossroads and can be changed by system technicians if it is necessary is more efficient to work. Low cost price and being smaller are the reasons to prefer the system.

BÖLÜM 1. GİRİŞ

Gün geçtikçe daha kullanışlı, daha az yer kaplayan ve daha ucuza üretilebilen cihazlar daha da tercih edilir hale gelmektedir. Şehirlerin büyümesi ile birlikte artan kavşak sayıları ve beraberinde artan araç sayıları da trafik sinyalizasyonunun da uygulanmasını zorunlu kılmaktadır. Literatüre (MUTCD) göre günün ortalama bir 8 saatinde 2 veya daha fazla şeritli bir yolda kavşaktan 600 veya daha fazla araç geçtiğinde (bkz. Tablo 2.1.) kavşağın trafik sinyalizasyonu ile kontrol altına alınması gerekir [1].

Yaptığımız gözlem ve incelemeler sonucu genel olarak kavşak kontrol cihazları her kavşağa göre sabit programlanıp kullanılmaktadır. Kavşak kontrol cihazları iki ana bölümden oluşur. Kontrol için kullanılan yöntem ve kontrol için kullanılan cihaz. Genellikle donanımsal olarak mikro işlemci ve ya PLC tabanlı cihazlar kullanılıp ayrıca yazılımsal olarak ta tek bir kavşağa göre programlanırlar. Biz burada yöntem olarak her kavşağa, program değiştirilmeden uyum sağlayabilecek ve parametreler ayarlanabilecek bir sistem geliştirdik ve gerçek zaman uygulamasını da yaptık. Donanımsal olarak PLC de kullanılabilirdi; fakat bu pahalı bir çözüm olurdu. Teknolojinin gelişmesiyle birlikte mikroişlemcilerde gelişmiş aynı zamanda gerçek zaman saati, ram gibi çevre birimleri de gelişmiştir. Dolayısıyla küçük bir kart tasarımıyla istediğimiz her işi yapabilecek duruma geldik. Bütün bu sebeplerden dolayı gerçekleştirdiğimiz bu sistemde mikrodenetleyici kullandık.

BÖLÜM 2. GENEL HATLARIYLA TRAFİK SİNYALİZASYONU

2.1. Trafik Sinyalizasyon Gereksinimi

Trafik sinyalizasyonu, yoğun trafik hacmine sahip kavşaklarda trafik polisi ile yönlendirme haricinde en etkin trafik kontrol sistemidir. Zira farklı yönlere hareket eden trafik akımlarının yarattığı çakışma nokta (katılma-ayrılma-kesişme) sayısı trafik sinyalizasyonu ile azaltılabilmektedir. Çünkü bir yöndeki trafik akımına izin verilirken bunu kesen trafik akımı durdurularak çakışma nokta sayısı azaltılmaktadır. Ancak bir kavşağın sinyalizasyon ile kontrol altına alınabilmesi için FHWA (MUTCD-Manual on Uniform Traffic Control Devices)'e göre

a. Minimum araç hacmi (8-saat)
b. Sürekli trafik akımının kesintisi(8-saat)
c. Minimum yay hacmi(8-saat)
d. 4- saat hacmi
e. Zirve-saat hacmi ve zirve-saat gecikmesi
f. Diğer(okul geçişleri, kaza etütleri, sistem, vb.)

Şartlarından en az birisini sağlamalıdır [1].

2.1.1. Minimum araç hacmi

Kavşak trafik hacmi Tablo 2.1'de belirtilen değerden fazla olması halinde kavşağın trafik sinyalizasyonu ile kontrol altına alınması gerekir [1]. Tablo 2.1.'de belirtilen değerler ortalama bir günün herhangi bir 8 saatinin her biri için alınan trafik hacmidir. "ortalama gün" hafta içindeki bir gün olup trafik hacminin normal yani temsili değerini ifade eden veya göz önüne alınan kavşakta sıklıkla tekrarlanan trafik hacmidir.

Tablo2.1. Minimum araç trafiği için hacim şartı (MUTCD) [1]

Her bir yaklaşım için şerit sayısı		Anayoldaki araç/saat trafiği (her iki yönün toplamı için)	Tali yol yaklaşımında en yüksek olan araç/saat trafiği (sadece bir yöndeki)
Anayol	Tali yol		
1	1	500	150
≥2	1	600	150
≥2	≥2	600	200
1	≥2	500	200

2.1.2. Sürekli trafik akımının kesintisi

Eğer anayoldaki araçlar tali yoldaki araçların geçmesi için aşırı beklemek zorunda kalırsa trafik sinyalizasyonu ile kavşak kontrol altına alınmalıdır [1]. Ayrıca tali yoldan kavşağa giren veya kavşağı geçen araçlar anayol trafiği tarafından kazaya maruz kalabilirler. Eğer ana ve tali yol trafik hacmi Tablo 2.2'de belirtilen değerlerden fazla ise trafik sinyalizasyonu ile kavşak kontrol altına alınmalıdır [1]. Tablo 2.2'de belirtilen değerler, ortalama günün herhangi bir 8 saatinin her biri için anılan trafik hacmidir.

Tablo2.2. Sürekli trafik akımının kesintisi için minimum trafik hacmi şartı (MUTCD)[1]

Her bir yaklaşım için şerit sayısı		Anayoldaki araç/saat trafiği (her iki yönün toplamı için)	Tali yol yaklaşımında en yüksek olan araç/saat trafiği (sadece bir yöndeki)
Anayol	Tali yol		
1	1	750	75
≥2	1	900	75
≥2	≥2	900	100
1	≥2	750	100

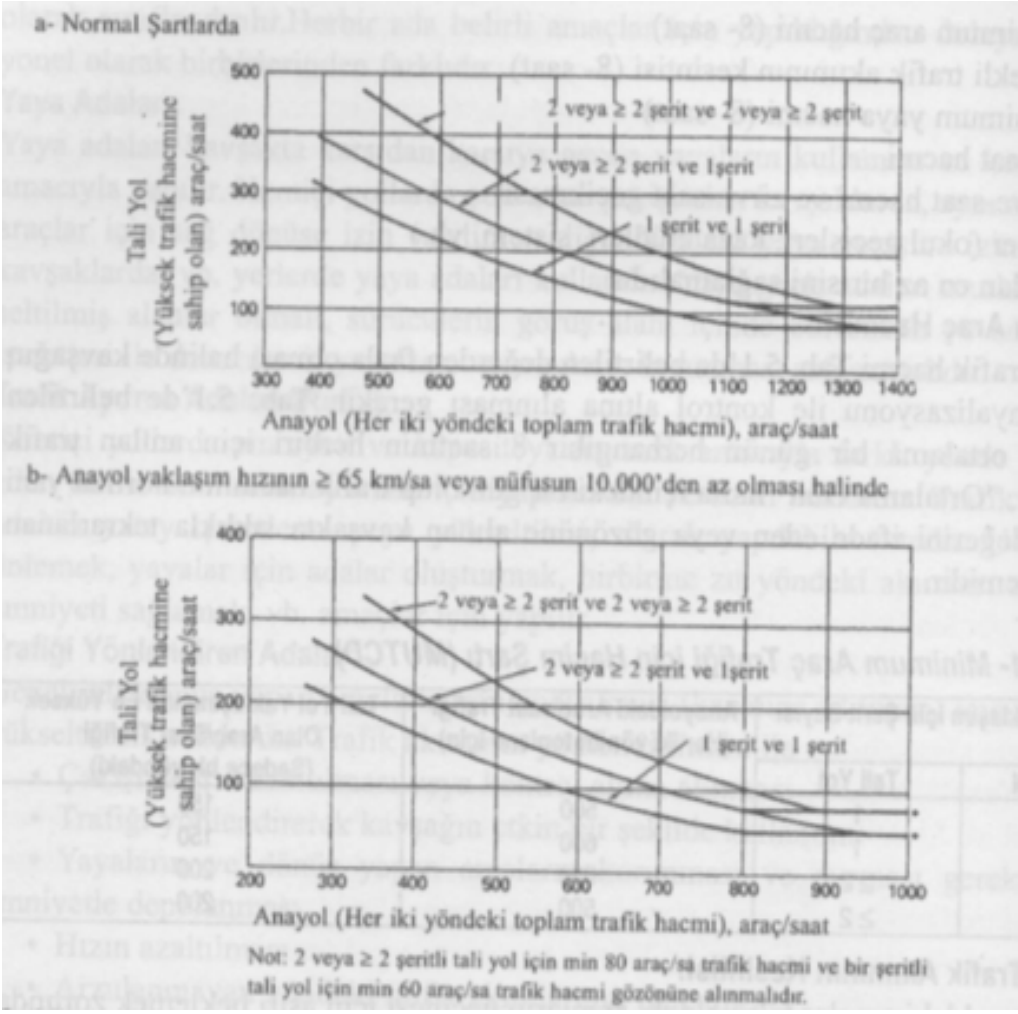
2.1.3. Minimum yaya hacmi

Ortalama bir günde anayolu geçen yayalar herhangi bir 4 saat için 100'den fazla veya herhangi bir saatte 190'dan fazla ise trafik sinyalizasyonu gereklidir [1].

2.1.4. 4-saat hacmi

MUTCD tarafından Şekil 2.1'de verilen standart grafik ile trafik sinyalizasyonunun gerekliliği saptanabilir. Eğer Şekil 2.1'de ortalama günün herhangi bir. 4 saatinin her

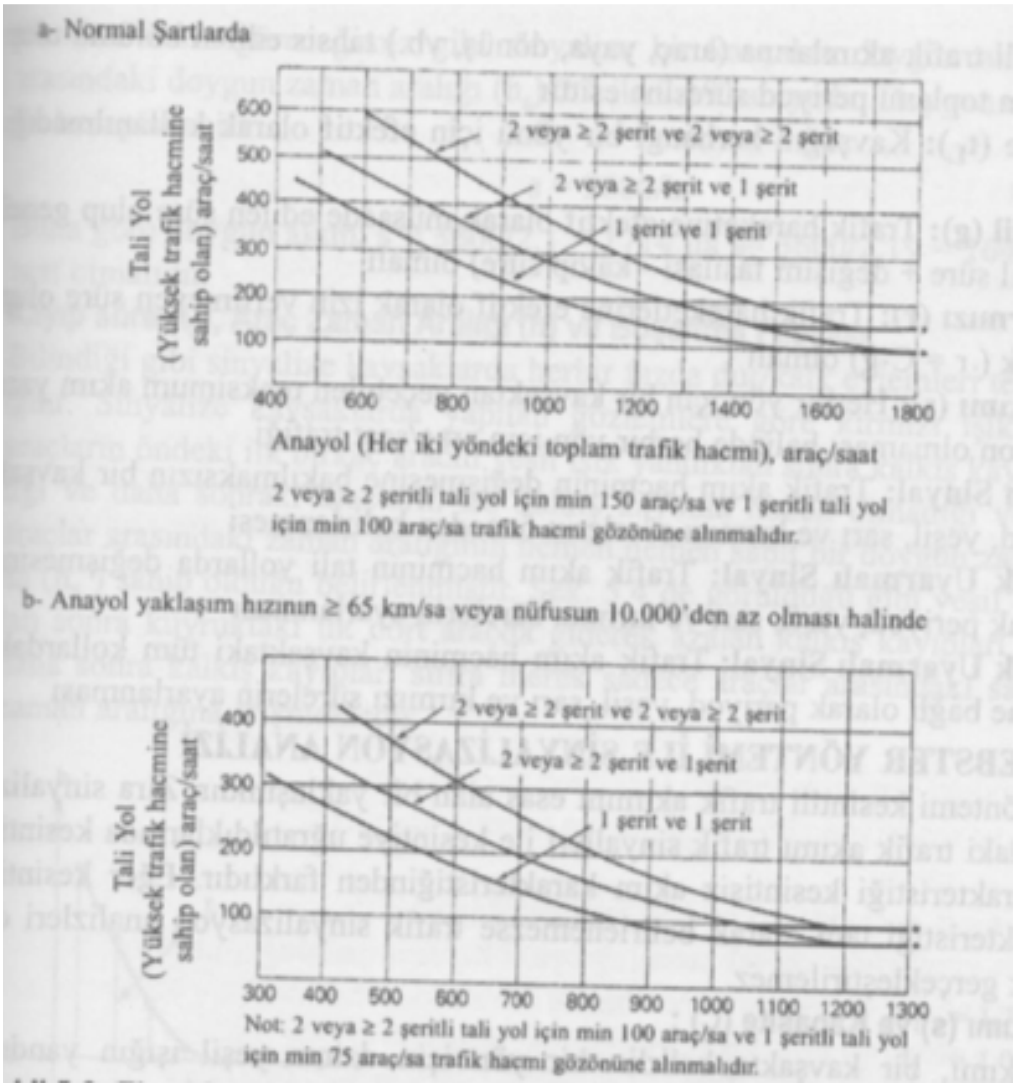
biri için anılan trafik hacmi göz önüne alınarak işaretlendiğinde eğrinin üzerinde kalıyorsa trafik sinyalizasyonu yapılmalıdır [1].



Şekil 2.1. 4-saat trafik hacmi için sinyalizasyon gereği (MUTCD)[1]

2.1.5. Zirve-saat hacmi

Günün bir saati boyunca tali yoldaki trafiğin belli bir değerden fazla olması halinde trafik sinyalizasyonu gereklidir. MUTCD tarafından Şekil 2.2'de verilen standart grafik ile trafik sinyalizasyonunun yapılmasının gerekliliği saptanabilir [1].



Şekil 2.2. Zirve-saat hacmi için sinyalizasyon gereği (MUTCD) [1]

2.1.6. Zirve-saat gecikme

Tali yoldaki gecikme tek şeritli yol için 4 araç-saat ve çift şeritli yol için 5 araç-saat'ten fazla olması halinde ve tek şeritli tali yolu tek yöndeki yaklaşımı 100 araç/saatten veya çift şeritli tali yolun tek yöndeki yaklaşımı 150 araç/saatten daha fazla olması halinde veya kavşağa giren toplam araç sayısının dört veya daha fazla kol için 800 araç/saatten veya üç kollu yaklaşım için 650 araç/saatten daha fazla olması halinde zirve-saat gecikmesi söz konusu olup trafik sinyalizasyonu gereklidir [1].

2.2. Sinyalizasyonun Temel Prensipleri ve Analiz Yöntemleri

Sinyalizasyon ile ilgili terimlerin tarifleri aşağıda verilmiştir [1],

Periyod: Sinyal ışıklarının (yeşil, sarı, kırmızı, vb.) sırayla bir tam dönüşü

Periyod süresi (C): Bir periyodun tamamlanması için geçen toplam süre, saniye

Fasıla (veya aralık): Sinyal ışıklarında hiçbir değişikliğin olmadığı süre, saniye

Değişim fasılası: Başlatılan bir hareket için "Sarı" gösterge olup yeşil ile kırmızı göstergelerin arasındaki fasıla

Boşaltma fasılası (Y_i): Başlatılan bir hareket için sarı ışıktan sonra tüm yönler için kırmızı (TÜM KIRMIZI) gösterge süresi olup "sarı artı tüm kırmızı" için geçen süre

Yeşil süre (fasıla) (g_i): Başlatılan bir hareket için yeşil gösterge süresi yani bir periyot için herhangi bir yöndeki yeşil süre

Kırmızı süre (fasıla) (r_i): Durdurulan bir hareket için kırmızı gösterge süresi

Faz: Bir periyot süresinin belirli kısımlarını (veya bir veya birkaç aralığını kapsayan) belirli trafik akımlarına (araç, yaya, dönüş, vb.) tahsis edilen bölümü olup tüm fazların toplamı periyot süresine eşittir

Kayıp süre (t_L): Kavşağın herhangi bir yönü için efektif olarak kullanılmadığı süre, saniye

Efektif yeşil (g): Trafik hareketine efektif olarak müsaade edilen süre olup genel olarak (yeşil süre + değişim fasılası - kayıp süre) olmalı

Efektif kırmızı (r): Trafik hareketlerine efektif olarak izin verilmeyen süre olup genel olarak (r = C-g) olmalı

Doyum akımı (s): Her bir yön için bir kavşaktan geçebilen maksimum akım yani sinyalizasyon olmaması halinde her bir yön için araç/saat trafiği

Önzamanlı sinyal: Trafik akım hacminin değişmesine bakılmaksızın bir kavşak için periyot, yeşil, sarı ve kırmızı sürelerin Önceden belirlenmesi

Yarı trafik uyarmalı sinyal: Trafik akım hacminin tali yollarda değişmesine bağlı olarak periyot, yeşil, sarı ve kırmızı sürelerin ayarlanması

Tam trafik uyarmalı sinyal: Trafik akım hacminin kavşaktaki tüm kollardaki değişmesine bağlı olarak periyot, yeşil, sarı ve kırmızı sürelerin ayarlanması

2.3. Sinyalizasyon Tasarımının Esasları

Sinyalize kavşakların sinyalizasyon tasarımı

a. Periyodun tayini ve yeşil sürelerin tahsisi (paylaşımı)
b. Faz planlarının ve sırasının oluşturulması
c. Sinyal koordinasyonu

için yapılır. Bolüm 2.2'de önzamanlı sinyalizasyon analizleri ile periyodun tayini ve yeşil sürelerin her bir faz için tahsisi ele alınmıştı. Burada kavşak sinyalizasyon uygulamaları ele alınacaktır.

Sinyal ışıklarının (göstergelerinin) anlamları aşağıdaki gibidir.

2.3.1. Sürekli ışıklar

Yeşil: Işıklı sinyal cephesine dönük araçların kavşağa girmesine izin verir. Sapmayan veya sağ/sol dönüş yapacak trafik akımı yeşil ışık, yer işaretlemesi, düşey trafik yasaklama işareti veya ışıklı yasaklama sinyalleri ile trafik akımı kontrol altına alınabilmektedir [2].

Sarı: Yeşil ışığın biteceğini veya kırmızı ışığın başlayacağını uyarır. Yeşil sonrası sarı ışık araçların kavşağa girmesine izin verirken kırmızı öncesi sarı ışık kavşağa girişi yasaklamaktadır [2].

Kırmızı: Tüm araçların kavşağa girmesini yasaklar ve tüm araçların durma çizgisinde veya yaya geçiş çizgisinde durmak zorunda olduğunu ifade eder [2].

2.3.2. Flâşörlü ışıklar

Sarı: Sürücünün kavşağı azami dikkatle geçmesi için uyarı göstergesidir.

Kırmızı: Sürekli kırmızı ışığı başlatılacağı için uyan niteliğinde olup tüm araçlar bu ışıkta durmak zorundadır.

2.3.3. Oklu ışıklar

Yeşil: Trafik akımının ok yönündeki hareketlerinin ihtiyatlı olarak yapılması için kullanılır. Örneğin; sağ veya sol dönüş yapan araçların yeşil ışıkta geçen yayaları kontrol etmesi ve yayalara öncelik tanıması için oklu yeşil ışık ile sürücüler uyarılmaktadır. Sol oklu yeşil ışık ise korunmuş sol dönüşe (yani sapmayan akımında araçlar arasında boşluk olduğunda sola dönüş yapılabilen) izin verildiğinde kullanılır.

Sarı: Ok yönündeki trafik akımının durdurulacağını ve/veya hemen arkasındaki kırmızı ışığın yanacağını belirtir. Yani sarı sürekli dairesel ışık ile aynı anlamı taşımakta olup trafik akımının belli yönlere yeşil ışıkla kontrol edileceğini ifade eder.

Kırmızı: Kırmızı dairesel ışık ile aynı anlamı ifade etmekte ve trafik akımının belli yönlere yeşil ışıkla kontrol edileceğini (yani yönlendirileceğini) uyarmaktadır.

2.3.4. Flâşörlü ve oklu ışıklar

Ok yönünde belirtilen trafik akımını kontrol altına almak amacıyla kullanılır. Flaşörlü sarı veya kırmızı oklu ışıklar sarı ve kırmızı dairesel ışıklar ile aynı anlamdadır.

MUTCD'ye göre sinyal cephesinin (trafik akımının sadece herhangi bir yaklaşımına kumanda eden sinyal lambaları) görünme mesafesi Tablo 2.3’de belirtilen değerlerden daha az olmamalıdır [1].

Tablo 2.3. Sinyal cephesi için Gerekli Görüş Mesafesi (MUTCD) [1]

%85 hız (km/sa)	Min. Görüş mesafesi (m)
35	50
40	65
50	80
55	100
65	120
70	140
80	165
90	190
95	220

Eğer trafik sinyal ışıkları üst üste konursa en üstte kırmızı, ortada sarı ve en altta yeşil olmalı ancak yan yana konulacaksa solda kırmızı, ortada sarı ve sağda yeşil olmalıdır. Oklu sinyal ışıkları kullanılacaksa ana sinyal ışıklarının en altında veya en sağında yer almalıdır. Ayrıca bir sinyal cephesinde en fazla 5 ışık yer almalıdır.

Kavşaklardaki trafik sinyalizasyonunun işletmesi aşağıdaki gibi farklı şekillerde yapılmaktadır.

2.3.5. Önzamanlı sinyalizasyon

Bu tip işletmede periyot süresi, fazlar ve tüm süreler önceden belirlendiği gibi yapılmaktadır. Ancak günün farklı saatlerinde önceden belirlenmiş farklı periyot ve ışık süreleri sinyal kontrol cihazı ile değiştirilebilmektedir.

2.3.6. Yarı-uyarmalı sinyalizasyon

Bu tip işletmede kavşağın tali yollarına yerleştirilen detektörler vasıtasıyla tali yolun yeşil ışıkları aktif hale getirilebilmektedir. Yani tali yolda bekleyen araçlar olduğunda anayola kırmızı ama tali yola yeşil ışık yanmaktadır.

2.3.7. Tam-uyarmalı sinyalizasyon

Bu tip işletmede tüm kavşak kollarında hatta gerektiğinde ilave dönüş şeritlerinde detektörler vasıtasıyla her bir yaklaşım yönü için ayrı ayrı yeşil süreler atanabilmektedir. Dolayısıyla her bir periyot süresi birbirinden farklı olup hatta yeşil ve kırmızı sürelerde değişebilmektedir. Ancak belirlenen minimum ve maksimum yeşil süre limitleri ile sinyalizasyon işlemi yapılmakladır [1].

2.3.8. Bilgisayar kontrollü sinyalizasyon

Bu tip işletmede şebekedeki tek bir kavşak için değil kavşakların tümü için sinyalizasyon süreleri ayarlanmaktadır. Burada birkaç ana arter esas alınarak bu ana arterleri kesen yollardaki kırmızı ışık süreleri, değişen trafik hacmi ile ayarlanmaktadır [1].

2.4. Sinyal Koordinasyonu

Önzamanlı sinyalizasyon analizi yapıldıktan sonra sinyalize edilen ana arterin üzerinde bulunan tüm kavşaklardaki sinyal ışıklarının birlikte çalışması için yapılan analize sinyal koordinasyonu denir. Bir başka deyişle, ana arter üzerinde bulunan sinyalizasyonların yeşil süreleri öyle ayarlanmalıdır ki ana arter üzerinde hareket eden araçların durmaksızın her kavşakta yeşil ışıkta geçebilmelidir [1].
Sinyal koordinasyonu

a. Senkronize (hemzaman-eşzaman,) sistem
b. Alternatif Sistem
c. Kesintisiz Sistem

ile farklı şekilde yapılmaktadır. Bu sistemlerin yegâne amacı, ana arter üzerinde sabit hızla hareket eden araçların her bir kavşakta yeşil ışıkla karşılaşmasını sağlamak yani durma ve/veya gecikmelerin elimine edilmesi için tüm kavşaklardaki sinyallerin koordine edilmesidir. Eğer

a. Kavşakların ara mesafesi mümkün olduğunca birbirine eşit
b. Tüm kavşakların periyotları birbirine eşit veya birbirinin 0.5, 1.0, 1.5, 2,0..... katı kadar

c. Trafik akımı sabit bir hiza sahip ise sinyal koordinasyonu en efektif şekilde gerçekleştirilebilmektedir [1].

BÖLÜM 3. TRAFİK SİNYALİZASYON CİHAZLARIYLA BİR KARŞILAŞTIRMA

Trafik sinyalizasyon sistemlerinin kontrol için kullanılan yöntem ve kontrol için kullanılan cihaz olarak iki bölümden oluşur. Donanımsal olarak PLC kullanılabileceği gibi mikrodenetleyiciler de kullanılabilir. Şekil 3.1, Şekil 3.2. ve Şekil 3.3'de PLC ile çalışan bir sinyalizasyon panosu örneği görülmektedir.

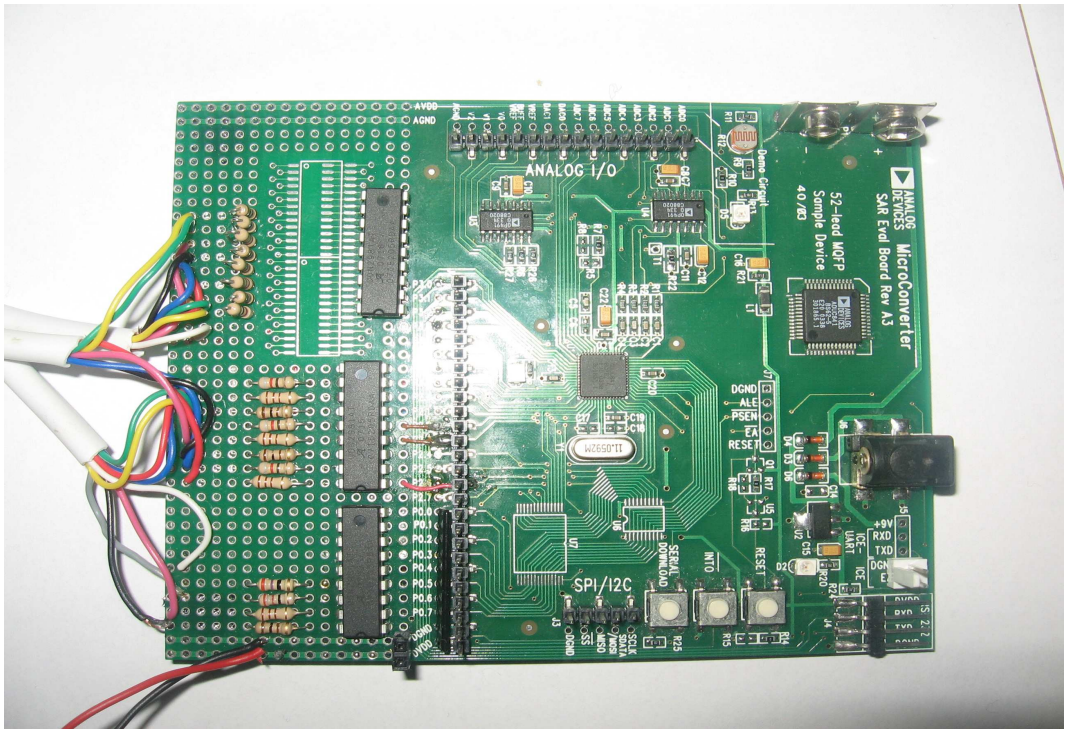
Şekil 3.1. PLC ile çalışan sinyalizasyon panosu

Şekil 3.2. PLC ile çalışan sinyalizasyon panosu

Şekil 3.3. PLC ile çalışan sinyalizasyon panosu

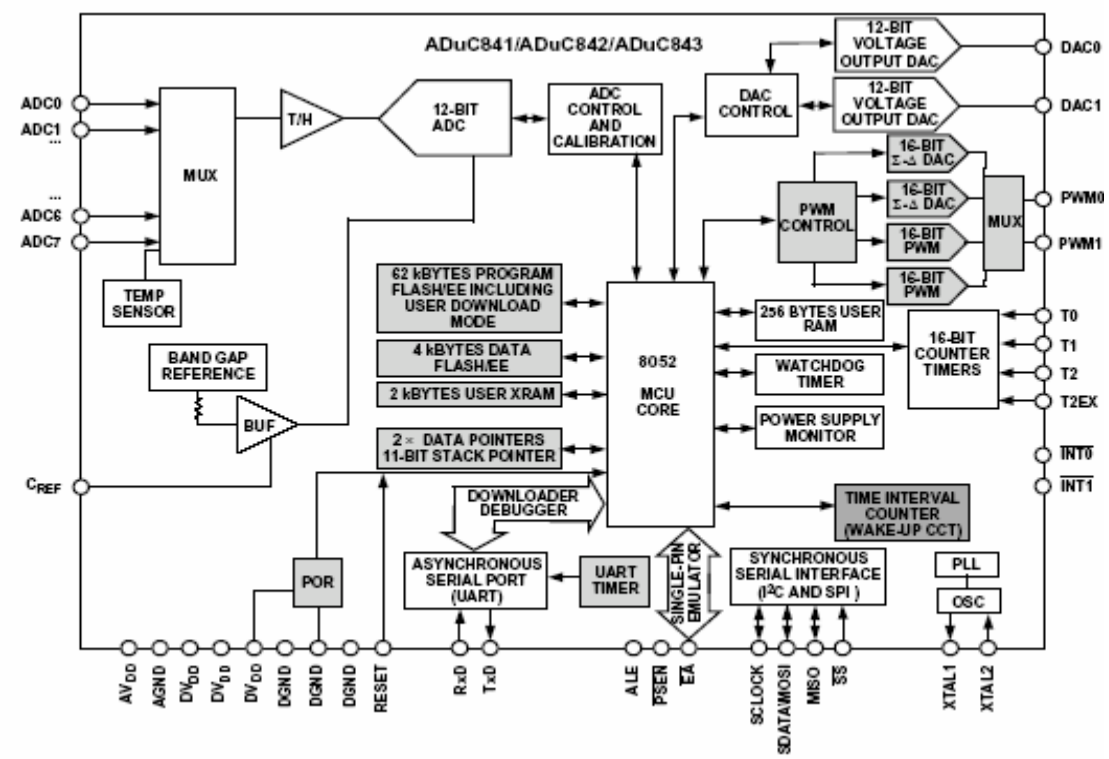
Sinyalizasyonda kullanılacak olan PLC cihazı yaklaşık 1000 YTL civarındadır. Şehirde bulunabilecek kavşakların çokluğu göz önüne alındığında meblağın ne kadar büyüyeceği aşikârdır. İçinde gerçek zaman saati bulunan, her türlü kavşağa uyum sağlayabilecek, dışarıdan sürelerin girilebileceği ve çok daha ucuz olan bir sistem sorunu çözmede daha da faydalı olacaktır. Bu çalışma sırasında kurduğumuz sistemin fiyatı ise 100 YTL'yi bile geçmemektedir.

Sinyalizasyon siteminde kullanılan donanım kontrol kısmı sürücü kısmı ve arayüz kısmı olarak üç bölüme ayrılır. Geliştirdiğimiz sistemde kontrol kısmında ana denetleyici olarak aduc841 mikrodenetleyiciyi kullandık. Aduc841'in mimarisi ana hatlarıyla Bölüm 3.1'de anlatılmıştır. Gelişen teknolojiyle birlikte gereken çevre birimleri mimarisi de geliştiğinden ekonomik ve az yer kaplayan tasarım geliştirmek mümkündür. Sürücü kısmı olarak statik anahtarlama elemanları triyaklar vasıtasıyla trafik işaret lambaları sürülmektedir. Bilindiği üzere yarı iletken elemanlar güvenlikleri sağlanırsa sonsuz ömürlüdür. Kontrol ve sürücü arasında arayüz devre vardır. Bu çalışmada geliştirilen yöntemin gerçek zaman uygulaması için kullandığımız mikrodenetleyici ve arayüz devresi Şekil 3.4.'de verilmiştir.



Şekil 3.4. Kullanılan mikrodenetleyici ve arayüz devresi

3.1. Genel Hatlarıyla Aduc841 Mimarisi [4]



Şekil 3.5 Aduc841'e ait mimari yapının blok diyagramı

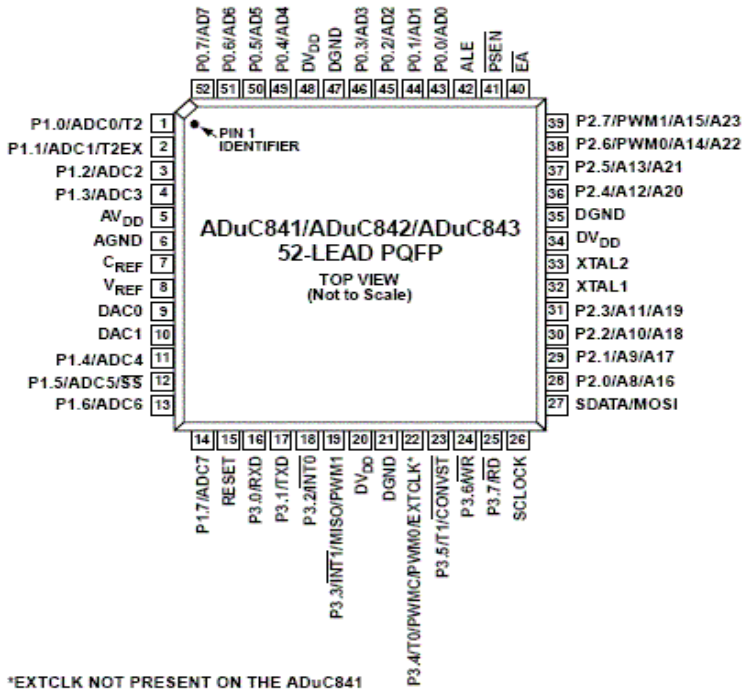
DUC 841, 8052 uyumlu 12-bit, yüksek performanslı ADC/DAC ve Flash/EE program/veri belleği bulunduran özel bir mikrodenetleyicidir.

Bu MCU 32 KHz kristalle çalışmasına rağmen üzerindeki PLL birimi sayesinde 16.78 MHz'e çıkabilmektedir. Saat darbeleri PLL biriminden mikrodenetleyicinin ihtiyaç duyduğu saat frekansı oluşturulmak üzere programlanabilir saat bölücü birimine yönlendirilir.

İçerisinde 8052 mikrodenetleyici bulunduğu için 8051 komut kümesine uyumludur. Ayrıca 62 Kbyte Flash/EE program hafıza, 4 Kbyte Flash/EE data hafıza, 256 byte RAM yonga üzerinde mevcuttur.

ADUC–841, 8 kanal, 12 bit çözünürlüğe sahip ADC, 2 kanal 12 bit çözünürlüğe sahip DAC, güç kaynağı göstergesi (power supply monitor) ve bandgap referans gibi analog özelliklere de sahiptir. Yonga üzerindeki dijital özellikler ise; TIC (time interval counter), WDT (watchdog timer), 3 adet timer/counter ve 2 seri I/O portudur (SPI, UART). Fabrika yazılımı; devre üzerinde seri yükleme, debug mod (UART ile) ve tek pin emulasyon modu (DLOAD pini ile) desteklemektedir.

3.1.1 Pin konfigürasyonu [4]

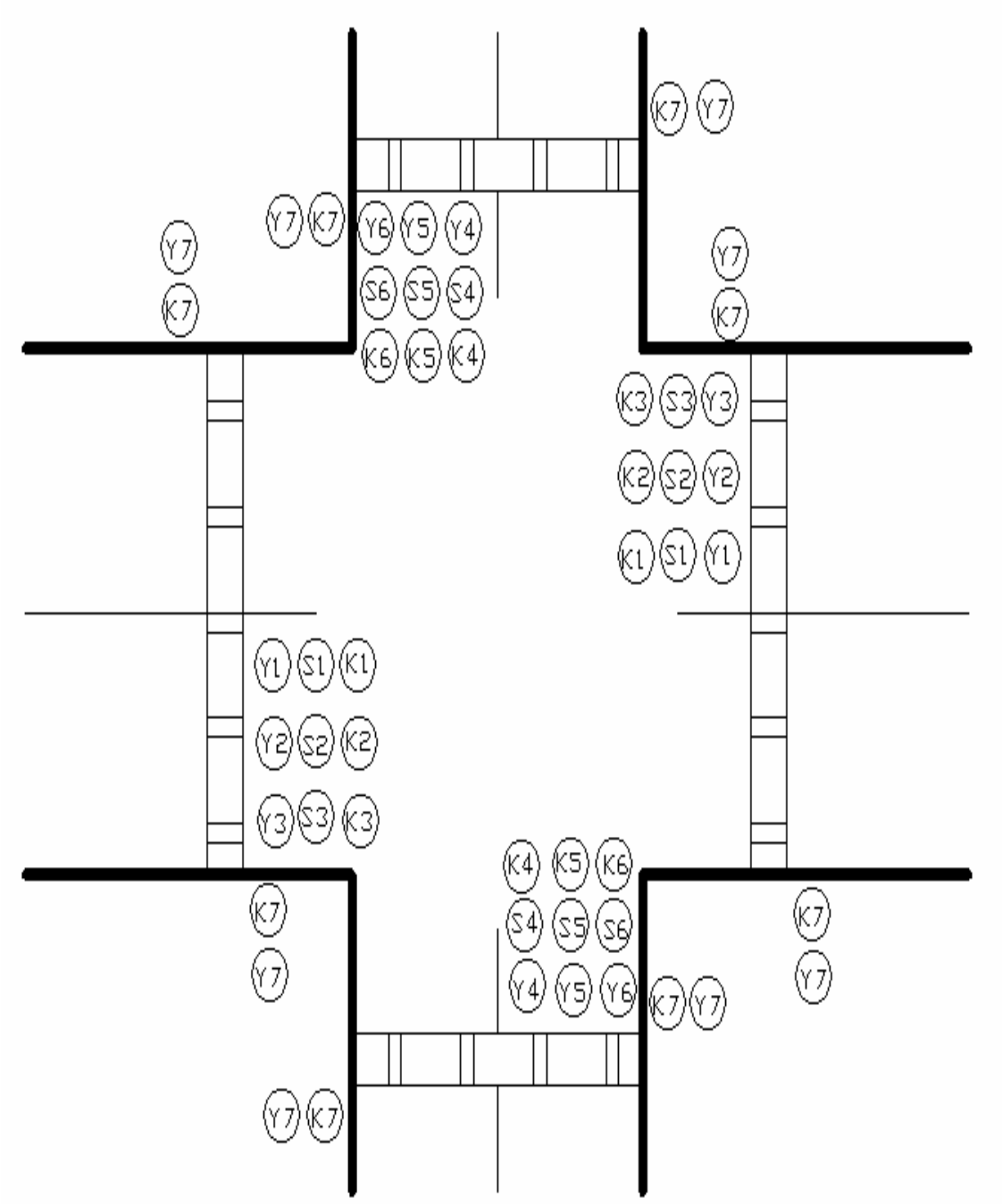


Şekil 3.6. 52 bacaklı ADUC-841'in pin konfigürasyonu

BÖLÜM 4. MİKRODENETLEYİCİ TABANLI MODÜLER KAVŞAK KONTROL CİHAZI TASARIMI

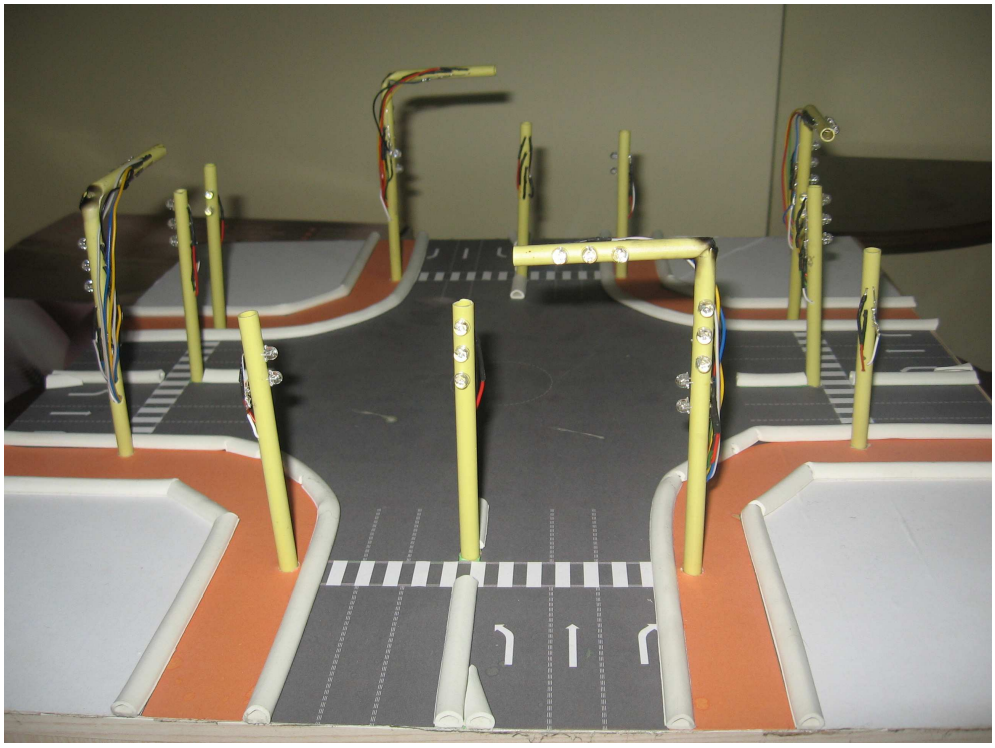
4.1. Model Kavşak Tasarımı

Sistem tek bir kavşağa değil her türlü kavşağa koordinasyonu sağlanabilecek şekilde düşünülmüştür. Sistemdeki bütün veriler dışarıdan girilebilecektir. Sisteme verilerin gelişigüzel girilmemesi için tasarımda yer alan ışıkların yerleri Şekil 4.1'de gösterilmektedir. Sistemde yeşil dalga uyumu bulunmaktadır. Yeşil dalga süresi sisteme önceden girilebilecektir. Trafik akışı hangi yola açık olursa olsun ana arter üzerindeki bir önceki kavşaktan veri geldiğinde girilen yeşil dalga süresi geçtiğinde sistem ana arter üzerindeki akışı sağlayacak şekilde kendini ayarlayacak ve bundan sonraki akışını buna göre düzenleyecektir. Işıklar arasında kavşağın güvenli bir şekilde boşalması için kavşağın durumuna uygun bir şekilde dışarıdan girilebilecek güvenlik süresi koyulmuştur. Geliştirilen sistemin bir önemli özelliği de gerçek zaman saati kullanılmasıdır. Bir gün istenilen aralıklarla 5 parçaya bölünebilecek, bu parçalardan her birinde hangi ışıkların ne kadar yanacağı belirlenebilecektir. Sistem verileri gerçek zaman saatinden okuduktan sonra o zaman dilimine denk gelen program parçasını çalıştıracak ve girilen sürelere uygun olarak trafik sinyalizasyonunu düzenleyecektir. Bu çalışmada geliştirilen yöntemin gerçek zamanda uygulanabilmesi için bir prototip örnek kavşak tasarlanmış ve gerçekleştirilmiştir. Tasarım için 3 şeritli yollardan oluşan bir kavşak oluşturulmuştur. Tasarlanan kavşak modeli Şekil 4.2, Şekil 4.3'de görülmektedir.



Şekil 4.1. Tasarımda kullanılan ışıkların yerleri

Burada Y harfi yeşil ışığı, K harfi kırmızı ışığı, S harfi sarı ışığı simgelemektedir. Harflerin yanında bulunan rakamlar ise sistem tasarımında kullanılan haliyle kaç numaralı ışık olduklarını belirtmektedir.



Şekil 4.2. Tasarlanan kavşak modeli

Şekil 4.3. Tasarlanan kavşak modeli

4.2. Kavşak Kontrol Cihazının Tasarımı

Sistemde kontrol ünitesi olarak Aduc841 mikrodenetleyici [4] kullanılmıştır. C programıyla sistemin programı yazılmıştır [EK]. Mikrodenetleyici vasıtasıyla udn2981 8 kanal voltaj sürücü entegreleri kontrol edilerek sisteme enerji verilmesi sağlanmıştır. Sistemi beslemek için 6 voltluk bir pil düzeneği kullanılmıştır. Bu düzenek sayesinde hem mikrodenetleyicinin beslenmesi hem de oluşturulan kavşak tasarımının beslenmesi sağlanmış böylece mikrodenetleyici için ayrı bir enerji kaynağına ihtiyaç kalmamıştır. Udn2981'in iç yapısı Şekil 4.4.'de görülmektedir [6]. Literatüre göre oluşturulan kavşakta kullanılan kırmızı ledler 1.5 V-1.6 V, sarı ledler 1.8 V, yeşil ledler 2.2 V-2.4 V voltaj değerlerinde ve 10-20 mA'lik iletim akımlarında çalışmaktadırlar [7]. Bu durum tasarlanan kavşaktaki her led için aynı gerilimi uygulayamacağımız anlamına gelir. Bunu çözebilmek için aşağıdaki formül yardımıyla hesaplanan dirençler kullanılmıştır. Kullanılan udn2981 Entegre ve direnç bağlantıları Şekil 4.5'de görülmektedir.

$$R = \frac{V - V_d}{A_d} \qquad [4.1]$$

Trafik akışını kontrol eden ledler karşılıklı olarak 2 ledin paralel bağlanmasından oluştuğu için direnç değerleri şu şekilde hesaplanır.

Kırmız ledler için; $$R = \frac{V - V_d}{A_d} = \frac{6 - 1{,}6}{0.04} = 110\Omega$$

Sarı ledler için; $$R = \frac{V - V_d}{A_d} = \frac{6 - 1.8}{0.04} = 105\Omega$$

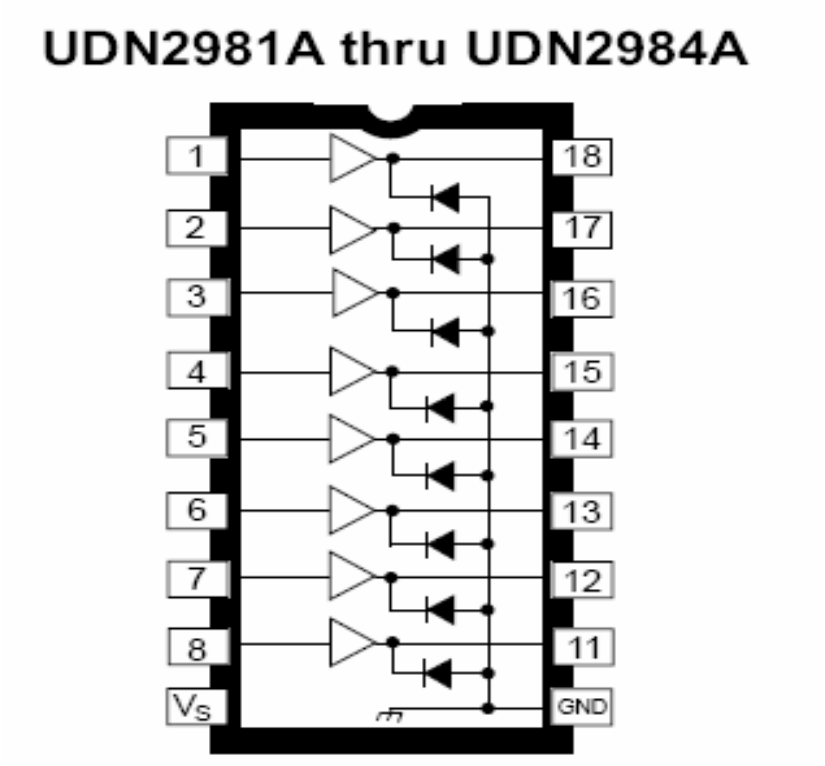
Yeşil ledler için; $$R = \frac{V - V_d}{A_d} = \frac{6 - 2.4}{0.04} = 90\Omega$$

Yaya akışını kontrol eden ledler karşılıklı olarak 8 ledin paralel bağlanmasından oluştuğu için direnç değerleri şu şekilde hesaplanır.

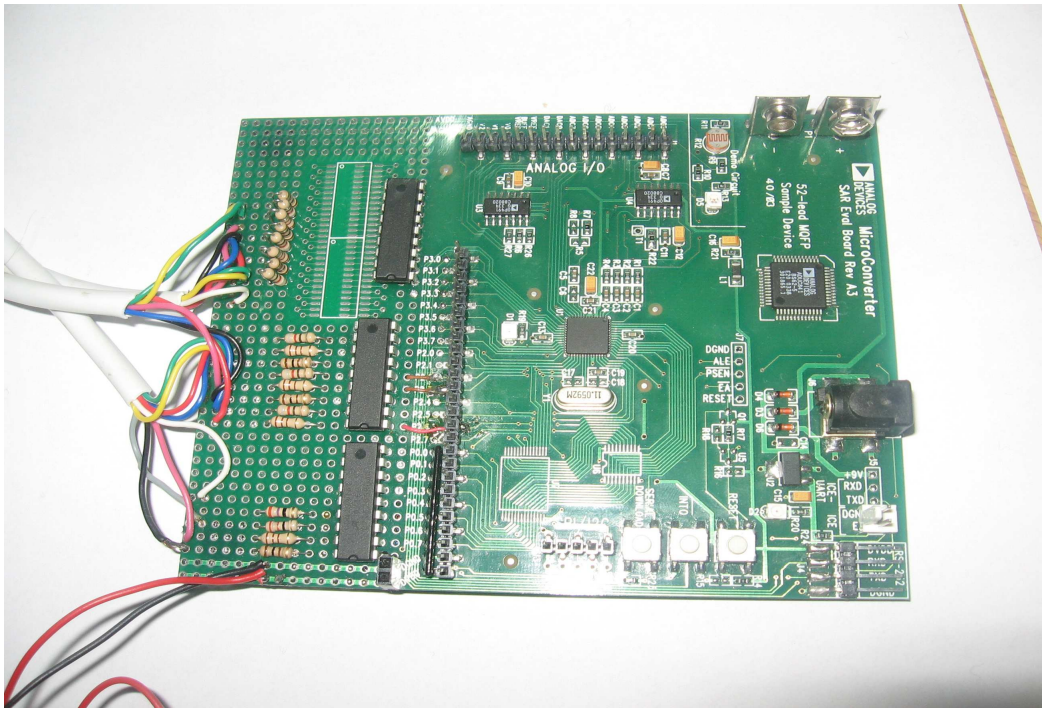
Kırmız ledler için; $$R = \frac{V - V_d}{A_d} = \frac{6 - 1{,}6}{0.16} = 27.5\Omega$$

Yeşil ledler için; $$R = \frac{V - V_d}{A_d} = \frac{6 - 2.4}{0.16} = 22.5\Omega$$

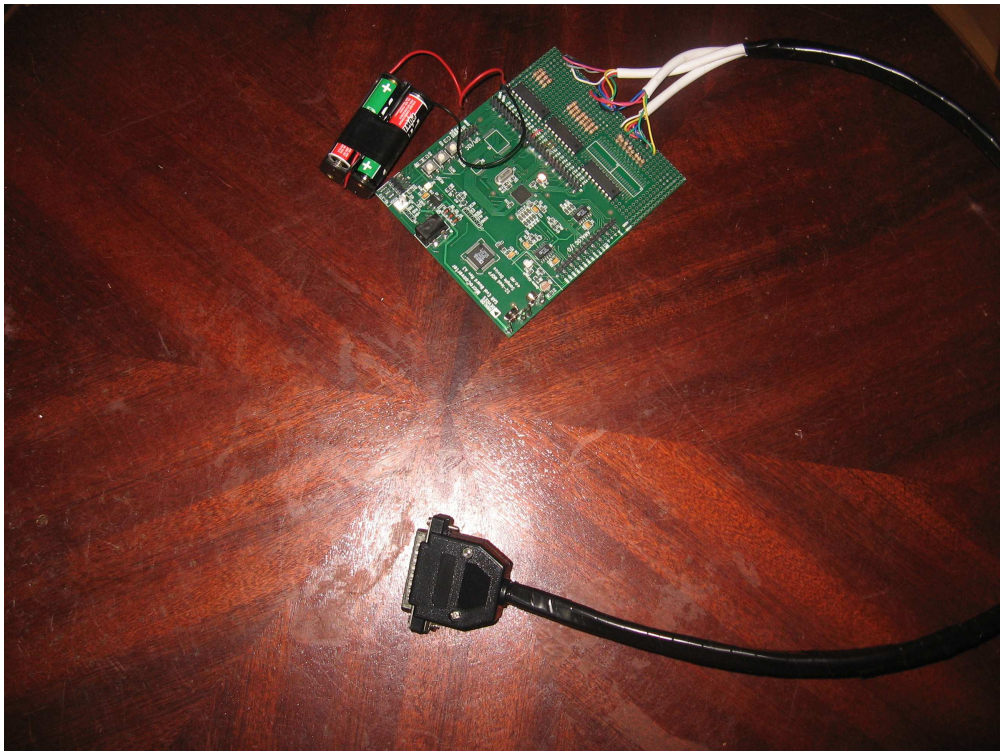
Sistemin kurulmuş ve çalışmaya hazır hali şekil 4.6'de görülmektedir.



Şekil 4.4. udn2981 iç yapısı [6]



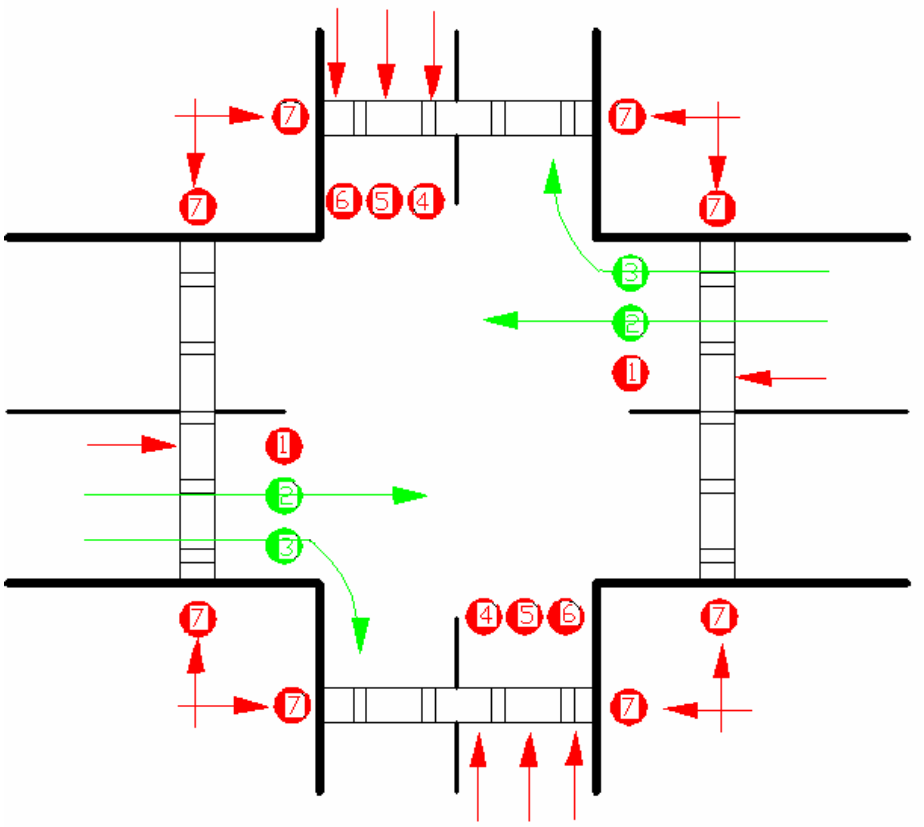
Şekil 4.5. Entegre ve direnç bağlantıları



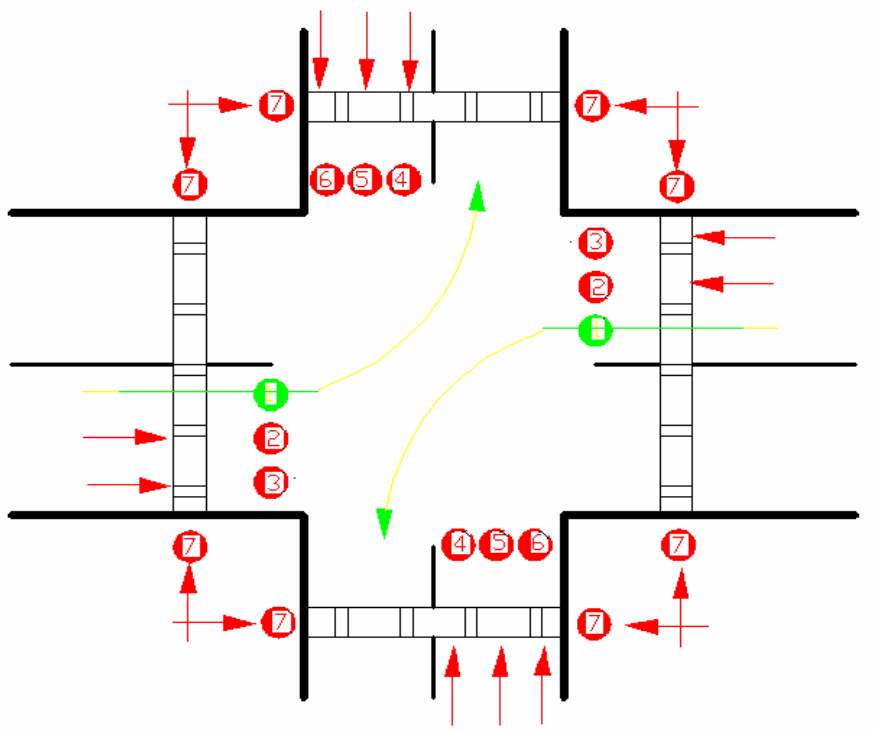
Şekil 4.6. Sistemin hazır hali

4.3. Sistemin Algoritması

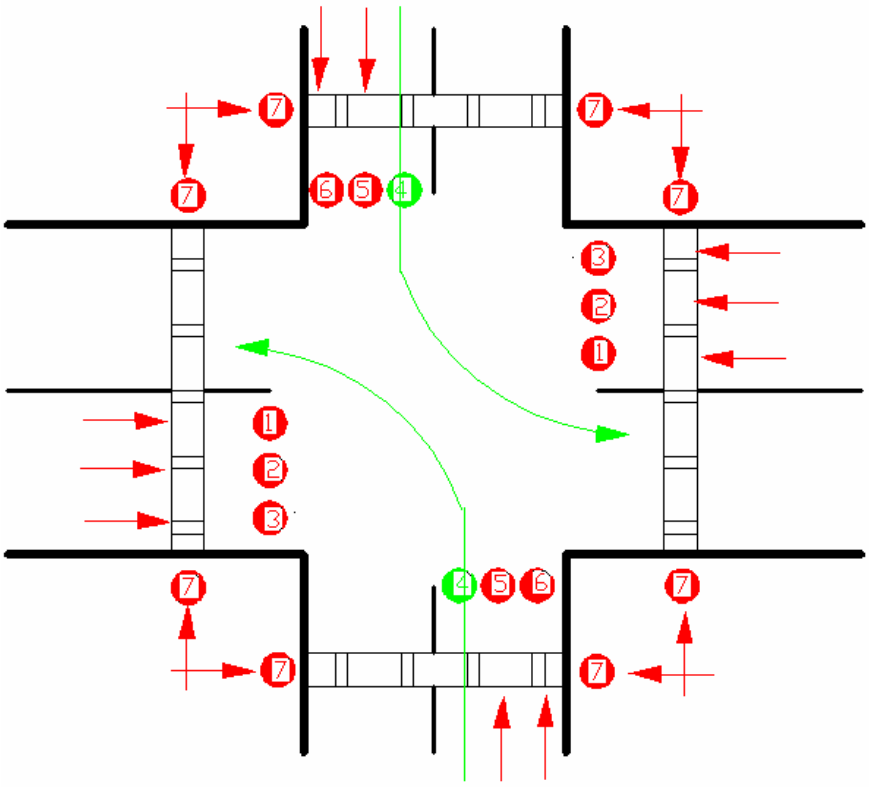
Algoritma oluşturulurken dikkate alınan 10 durum vardır. Bu 10 durumdan 2, 4, 6, 8, 10. durumlar güvenlik durumlarıdır. Bu durumlar ışıklar arasında kavşağın güvenli bir şekilde boşalmasını sağlamak için koyulmuş durumlar olup bütün ışıklar kırmızı yanmaktadır.Diğer 5 durum ve güvenlik durumu Şekil 4.7, Şekil 4.8, Şekil 4.9, Şekil 4.10, Şekil 4.11 ve Şekil 4.12'de daha net bir şekilde görülmektedir.



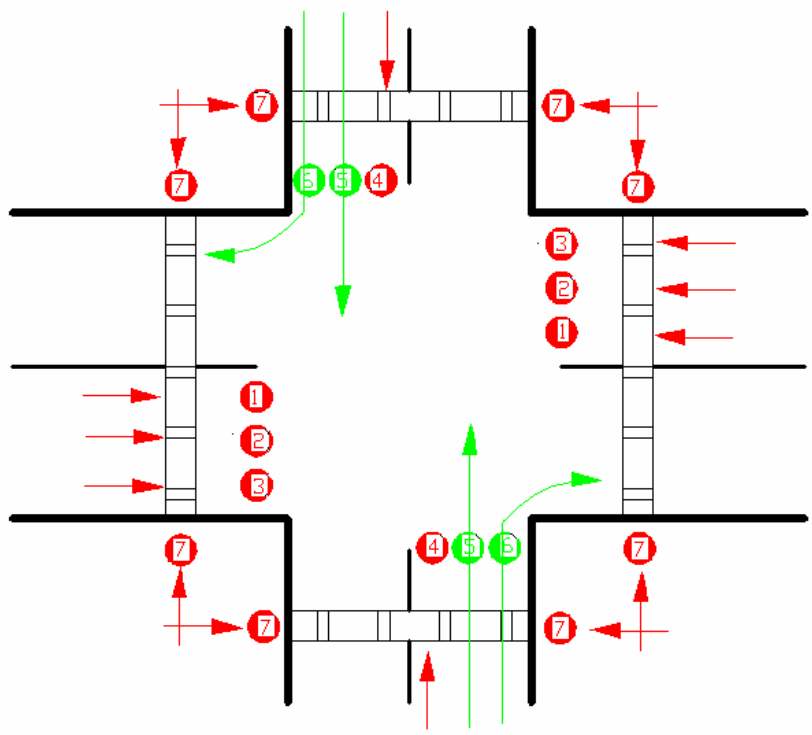
Şekil 4.7. Birinci durumda trafik akışı



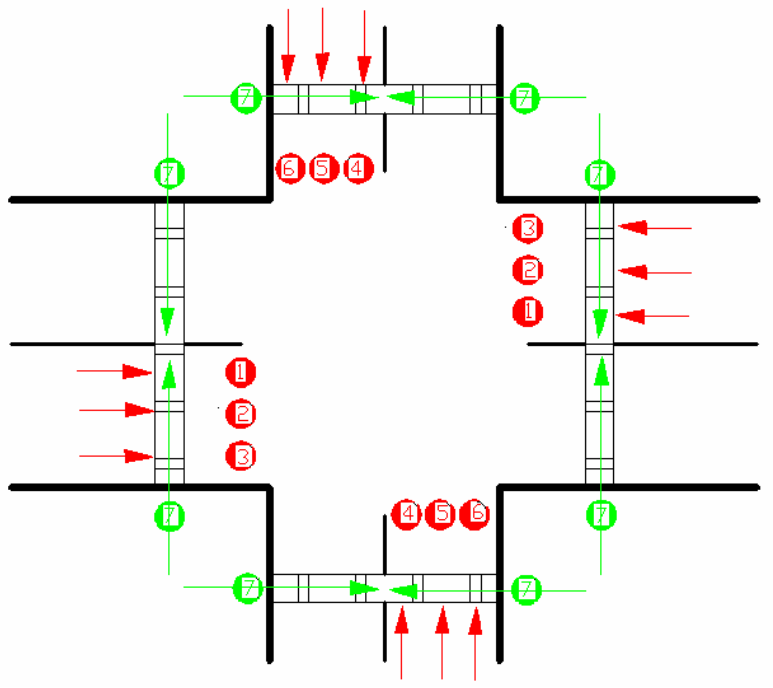
Şekil 4.8. Üçüncü durumda trafik akışı



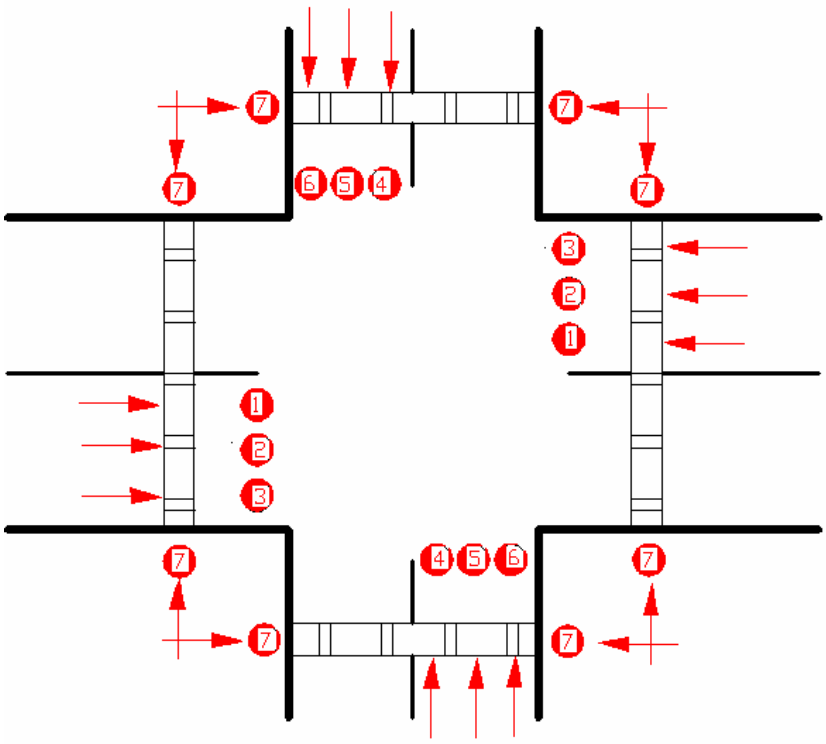
Şekil 4.9. Beşinci durumda trafik akışı



Şekil 4.10. Yedinci durumda trafik akışı



Şekil 4.11. Dokuzuncu durumda trafik akışı



Şekil 4.12. 2.4.6.8. ve 10. durumlarda trafik akışı

Yukarıdaki şekillerin özetlenmiş hali aşağıda görülmektedir.

1.durum	:	2,3 nolu ışıklar yeşil diğer ışıklar kırmızı
2.durum	:	güvenlik süresi
3.durum	:	1 nolu ışıklar yeşil diğer ışıklar kırmızı
4.durum	:	güvenlik süresi
5.durum	:	4 nolu ışıklar yeşil diğer ışıklar kırmızı
6.durum	:	güvenlik süresi
7.durum	:	5,6 nolu ışıklar yeşil diğer ışıklar kırmızı
8.durum	:	güvenlik süresi
9.durum	:	7 nolu ışıklar yeşil diğer ışıklar kırmızı
10.durum	:	güvenlik süresi

4.3.1. Sistem verilerinin girilmesi

Gerçek zaman saatini çalıştır.

i. 1. programın başlangıç saatini gir.
ii. 2. programın başlangıç saatini gir.
iii. 3. programın başlangıç saatini gir.
iv. 4. programın başlangıç saatini gir.
v. 5. programın başlangıç saatini gir.
vi. Yeşil dalga süre gir.
vii. Güvenlik süresini gir.

Yeşil dalga durumu:

Ana arter üzerindeki bir önceki kavşaktan veri gelince zamanlayıcıyı çalıştır
vi. sorgulamasında girilen süre kadar geçtikten sonra sistemi 1. program çalışacak şekilde ayarla.

4.3.2. Programlar

1. program

Zaman saatini kontrol et eğer i) durumunda girilen süre ile ii) durumunda girilen süre arasındaysa bu programda kal. Sonra bir sonraki programa geç.

Sorgulamalar ve durumlar:

a) 1.durumda 2 ve 3 nolu ışıkların yeşilleri kaç saniye yanacak?
b) 3. durumda 1 nolu ışığın yeşili kaç saniye yanacak?
c) 5. durumda 4 nolu ışığın yeşili kaç saniye yanacak?
d) 7. durumda 5 ve 6 nolu ışıkların yeşilleri kaç saniye yanacak?
e) 9. durumda 7 nolu ışığın yeşili kaç saniye yanacak?

1.durum:

Zamanlayıcıyı çalıştır.

1 sn süreyle 2 ve 3 nolu ışıkların sarılarını yak

a) sorgulamasında belirtilen süre kadar 2 ve 3 nolu ışıkların yeşillerini yak diğer lambaları kırmızı yak.

1 sn süreyle 2 ve 3 nolu ışıkların sarılarını yak

a) sorgulamasında geçen süre geçtiyse 2 ve 3 nolu ışıkların kırmızılarını yak ve sonraki duruma geç

2.durum:

Zamanlayıcıyı çalıştır.

vii) sorgulamasında girilen süre kadar sistemdeki bütün lambaların kırmızılarını yak.

3.durum:

Zamanlayıcıyı çalıştır.

1 sn süreyle 1 nolu ışıkların sarılarını yak

a) sorgulamasında belirtilen süre kadar 1 nolu ışıkların yeşillerini yak diğer lambaları kırmızı yak.

1 sn süreyle 1 nolu ışıkların sarılarını yak

a) sorgulamasında geçen süre geçtiyse 1 nolu ışıkların kırmızılarını yak ve sonraki duruma geç

4.durum:

Zamanlayıcıyı çalıştır.

vii) sorgulamasında girilen süre kadar sistemdeki bütün lambaların kırmızılarını yak.

5.durum:

Zamanlayıcıyı çalıştır.

1 sn süreyle 5 nolu ışıkların sarılarını yak

a) sorgulamasında belirtilen süre kadar 5 nolu ışıkların yeşillerini yak diğer lambaları kırmızı yak.

1 sn süreyle 5 nolu ışıkların sarılarını yak

a) sorgulamasında geçen süre geçtiyse 5 nolu ışıkların kırmızılarını yak ve sonraki duruma geç

6.durum:

Zamanlayıcıyı çalıştır.

vii) sorgulamasında girilen süre kadar sistemdeki bütün lambaların kırmızılarını yak.

7.durum:

Zamanlayıcıyı çalıştır.

1 sn süreyle 5 ve 6 nolu ışıkların sarılarını yak

a) sorgulamasında belirtilen süre kadar 5 ve 6 nolu ışıkların yeşillerini yak diğer lambaları kırmızı yak.

1 sn süreyle 5 ve 6 nolu ışıkların sarılarını yak

a) sorgulamasında geçen süre geçtiyse 5 ve 6 nolu ışıkların kırmızılarını yak ve sonraki duruma geç

8.durum:

Zamanlayıcıyı çalıştır.

vii) sorgulamasında girilen süre kadar sistemdeki bütün lambaların kırmızılarını yak.

9.durum:

Zamanlayıcıyı çalıştır.

1 sn süreyle 7 nolu ışıkların sarılarını yak

a) sorgulamasında belirtilen süre kadar 7 nolu ışıkların yeşillerini yak diğer lambaları kırmızı yak.

1 sn süreyle 7 nolu ışıkların sarılarını yak

a) sorgulamasında geçen süre geçtiyse 7 nolu ışıkların kırmızılarını yak ve sonraki duruma geç

10.durum:

Zamanlayıcıyı çalıştır.

vii) sorgulamasında girilen süre kadar sistemdeki bütün lambaların kırmızılarını yak.
Zamanlayıcıyı çalıştır.

b) sorgulamasında verilen süre kadar bekle. Sonra bir sonraki aşamaya geç.

2. 3. ve 4. programlardaki algoritmalar da aynı 1. program algoritması gibidir.

5. program

Zaman saatini kontrol et eğer v) durumunda girilen süre ile i) durumunda girilen süre arasındaysa bu programda kal. Sonra bir sonraki programa geç.
5. program flaşör durmudur.

Flaşör durumu

Zamanlayıcıyı çalıştır.

1 sn süreyle 1,2 ve 3 nolu ışıkların sarılarını yak.

1 sn süreyle 4,5,6 ve 7 nolu ışıkların kırmızılarını yak.

1 sn sonra bütün lambaları söndür.

BÖLÜM 5. ÖRNEK BİR KAVŞAK SİNYALİZASYONU

Önceki aşamalarda sırasıyla yöntem geliştirildi sonra yöntemin uygulanacağı kontrol cihazı ve uygulama alanı olarakta prototip kavşak modeli yapıldı. Bu bölümde ise örnek bir trafik sinyalizasyon akışı üzerinde durulmuştur. Örnek olarak 3 şeritli yollardan oluşan bir dörtyol kavşağı düşünülmüştür. Bu örneği gerçekleştirmek için geliştirdiğimiz sistem ve gerçek zaman uygulaması olan tasarlanan kavşak modeli kullanılmıştır. Bir gün trafik yoğunluğu hesaba katılarak 06:00-10:00, 10:00-16:00, 16:00-20:00, 20:00-24:00, 24:00-06:00 olmak üzere program parçalarına ayrılmıştır. Her bir aşama için yoğunluklar düşünülerek ayrı ayrı süreler girilmiş 5. aşama olan 24:00-06:00 arasında sitemin flaşör durumuna geçmesi hesaplanmıştır. Işıklar arasında kavşağın güvenli bir şekilde boşalması için 2 saniyelik güvenlik süresi koyulmuştur.

5.1. Birinci Programdaki Işıkların Yeşil Süreleri ve Devir Süresi

1.durum	:	2,3	yeşil	28 sn	
2.durum	:	güvenlik süresi		2 sn	(kavşak boşaltılıyor)
3.durum	:	1	yeşil	15 sn	
4.durum	:	güvenlik süresi		2 sn	(kavşak boşaltılıyor)
5.durum	:	4	yeşil	10 sn	
6.durum	:	güvenlik süresi		2 sn	(kavşak boşaltılıyor)
7.durum	:	5,6	yeşil	17 sn	
8.durum	:	güvenlik süresi		2 sn	(kavşak boşaltılıyor)
9.durum	:	7	yeşil	10 sn	
10.durum	:	güvenlik süresi		2 sn	(kavşak boşaltılıyor)

Devir : 90 sn

5.2. İkinci Programdaki Işıkların Yeşil Süreleri ve Devir Süresi

1.durum	:	2,3	yeşil	20 sn	
2.durum	:	güvenlik süresi		2 sn	(kavşak boşaltılıyor)
3.durum	:	1	yeşil	15 sn	
4.durum	:	güvenlik süresi		2 sn	(kavşak boşaltılıyor)
5.durum	:	4	yeşil	10 sn	
6.durum	:	güvenlik süresi		2 sn	(kavşak boşaltılıyor)
7.durum	:	5,6	yeşil	15 sn	
8.durum	:	güvenlik süresi		2 sn	(kavşak boşaltılıyor)
9.durum	:	7	yeşil	10 sn	
10.durum	:	güvenlik süresi		2 sn	(kavşak boşaltılıyor)

Devir : 80 sn

5.3. Üçüncü Programdaki Işıkların Yeşil Süreleri ve Devir Süresi

1.durum	:	2,3	yeşil	35 sn	
2.durum	:	güvenlik süresi		2 sn	(kavşak boşaltılıyor)
3.durum	:	1	yeşil	15 sn	
4.durum	:	güvenlik süresi		2 sn	(kavşak boşaltılıyor)
5.durum	:	4	yeşil	10 sn	
6.durum	:	güvenlik süresi		2 sn	(kavşak boşaltılıyor)
7.durum	:	5,6	yeşil	20 sn	
8.durum	:	güvenlik süresi		2 sn	(kavşak boşaltılıyor)
9.durum	:	7	yeşil	10 sn	
10.durum	:	güvenlik süresi		2 sn	(kavşak boşaltılıyor)

Devir : 100 sn

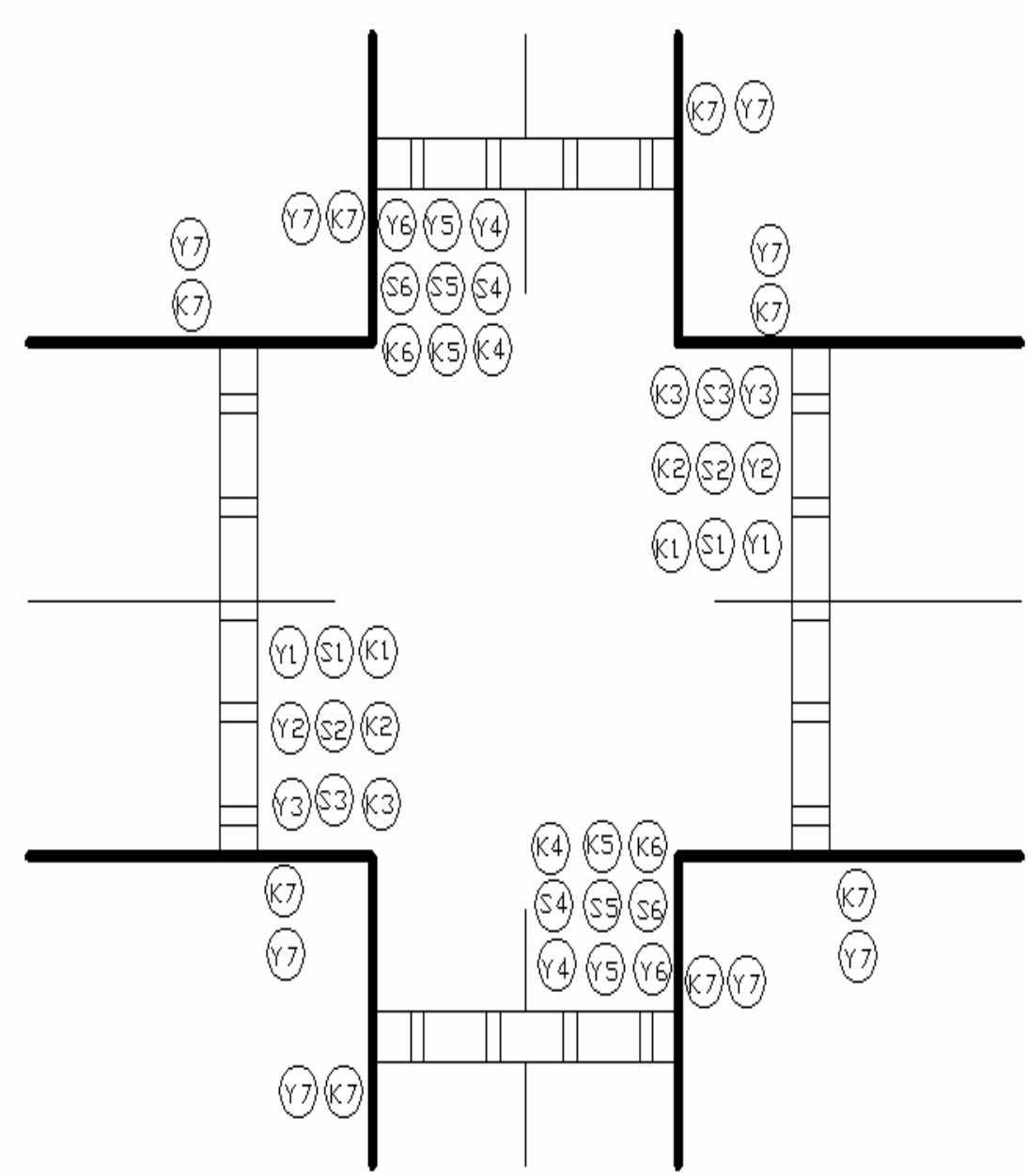
5.4. Dördüncü Programdaki Işıkların Yeşil Süreleri ve Devir Süresi

1.durum	:	2,3	yeşil	15 sn	
2.durum	:	güvenlik süresi		2 sn	(kavşak boşaltılıyor)
3.durum	:	1	yeşil	15 sn	
4.durum	:	güvenlik süresi		2 sn	(kavşak boşaltılıyor)
5.durum	:	4	yeşil	10 sn	
6.durum	:	güvenlik süresi		2 sn	(kavşak boşaltılıyor)
7.durum	:	5,6	yeşil	15 sn	
8.durum	:	güvenlik süresi		2 sn	(kavşak boşaltılıyor)
9.durum	:	7	yeşil	10 sn	
10.durum	:	güvenlik süresi		2 sn	(kavşak boşaltılıyor)

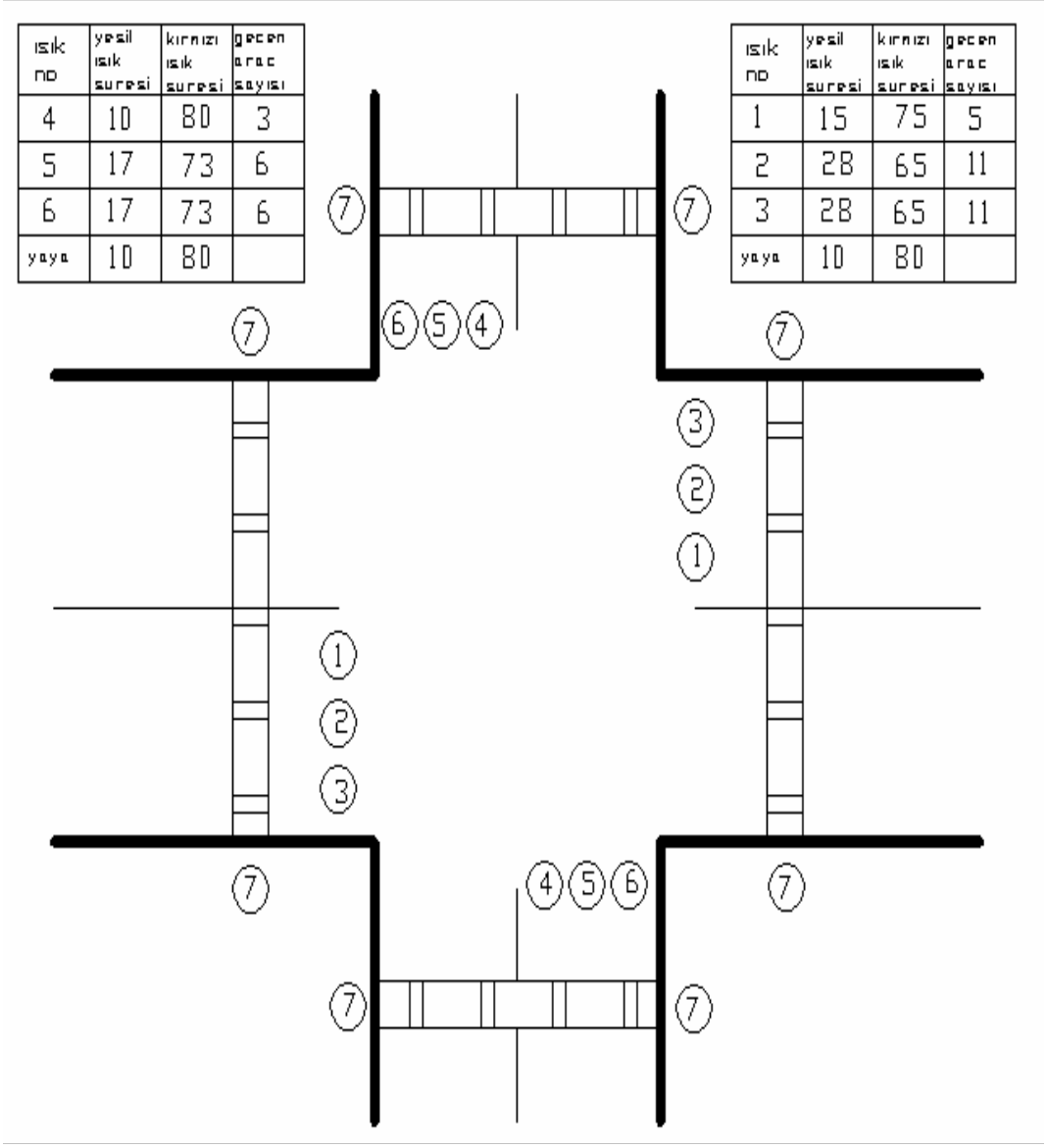
Devir : 75 sn

Şekil 5.1'de örnek uygulama için kullanılan kavşak modeli, Şekil 5.2, Şekil 5.3, Şekil 5.4 ve Şekil 5.5'de kavşak modeli üzerinde ışıkların ne kadar yanacağı ve bu süre içerisinde kavşaktan kaç araç geçebileceği gösterilmiştir. Belirtilen süreler ve geçebilecek araç sayıları aşağıdaki formülle hesaplanmıştır [3].

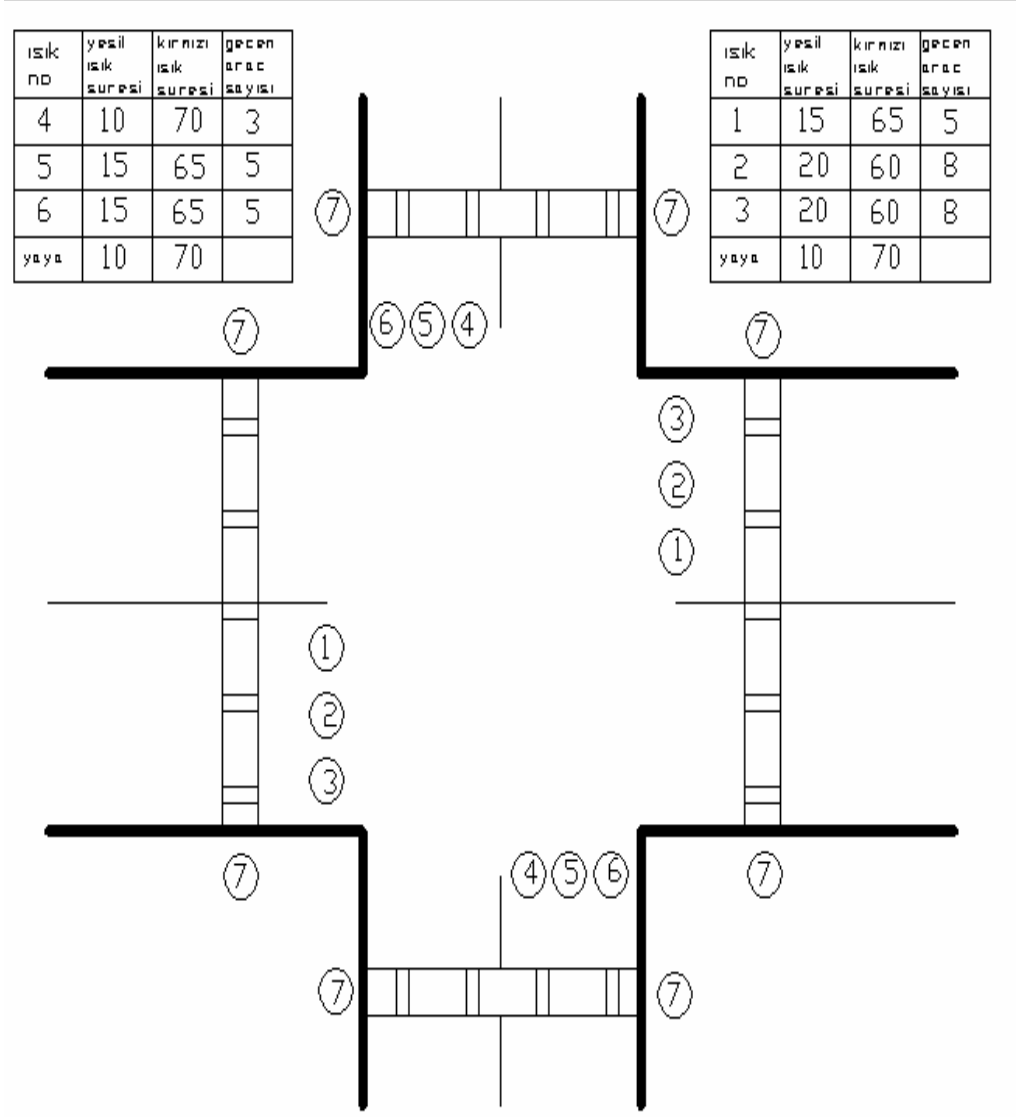
$$T = 3.6 + araçsayıra \times 2.19 \qquad [5.1]$$



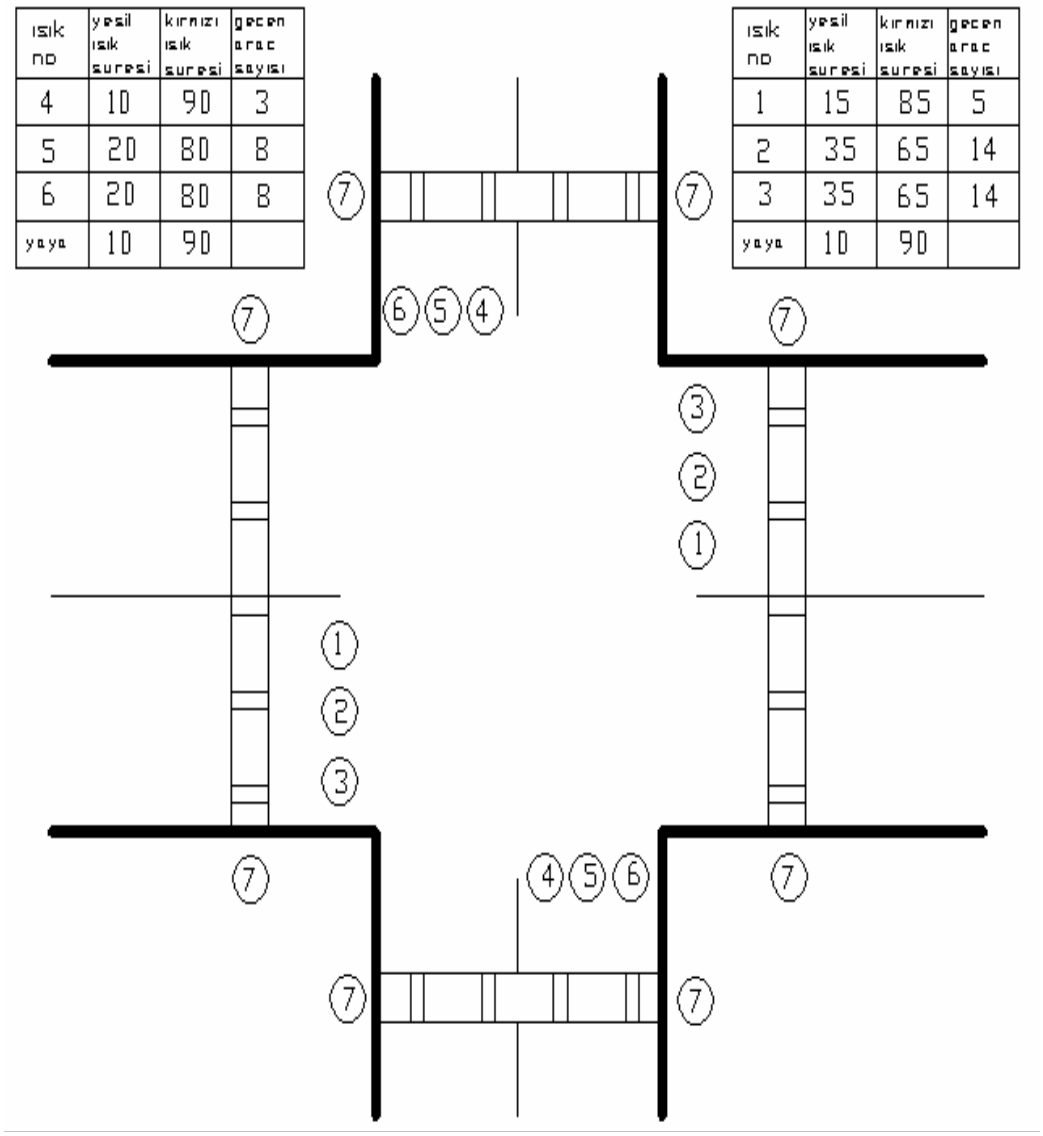
Şekil 5.1. Örnek olarak kullanılacak kavşağın sinyalizasyon modeli



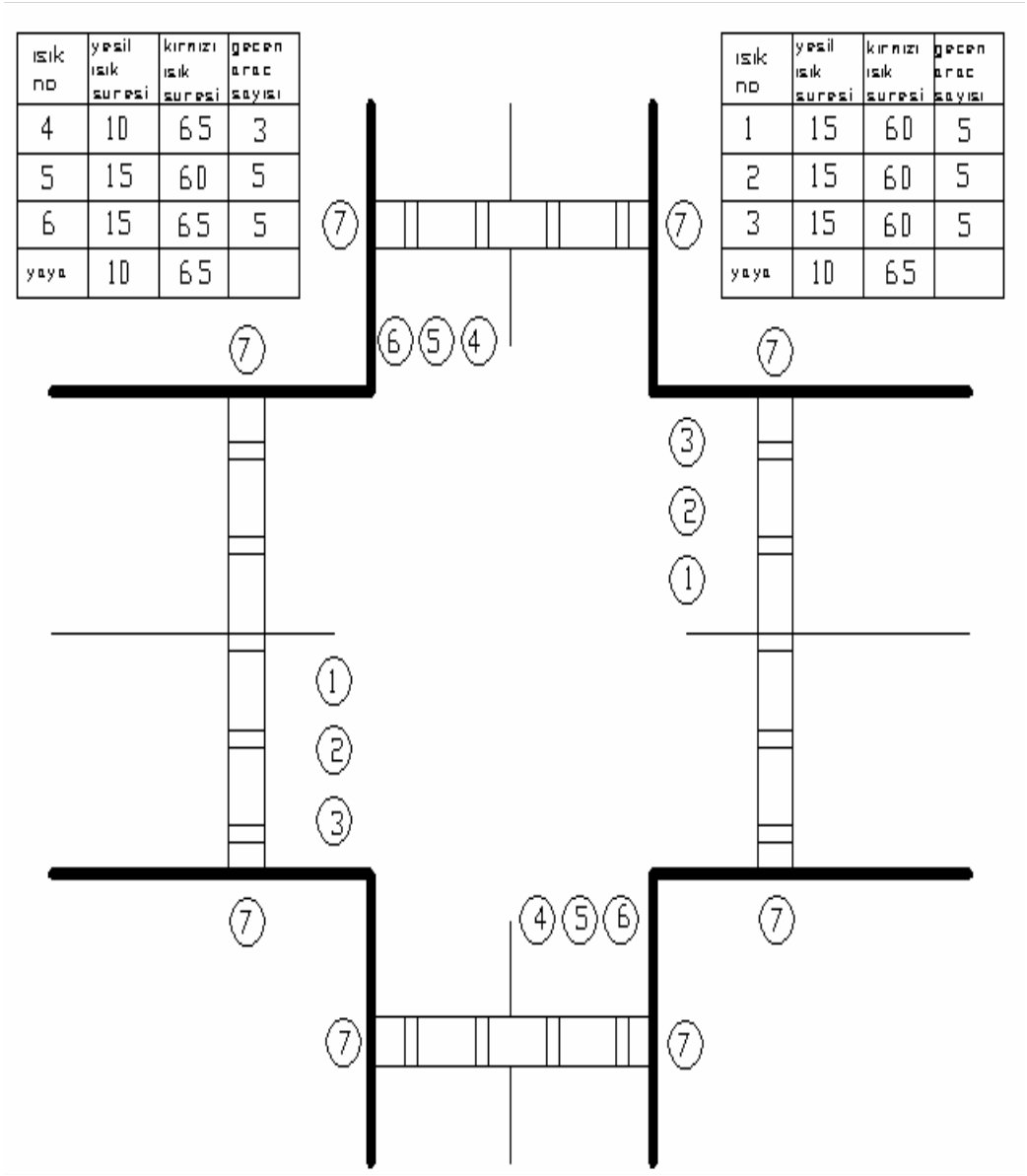
Şekil 5.2. Örnek modelde birici program süreleri ve geçebilecek araç sayıları



Şekil 5.3. Örnek modelde ikinci program süreleri ve geçebilecek araç sayıları

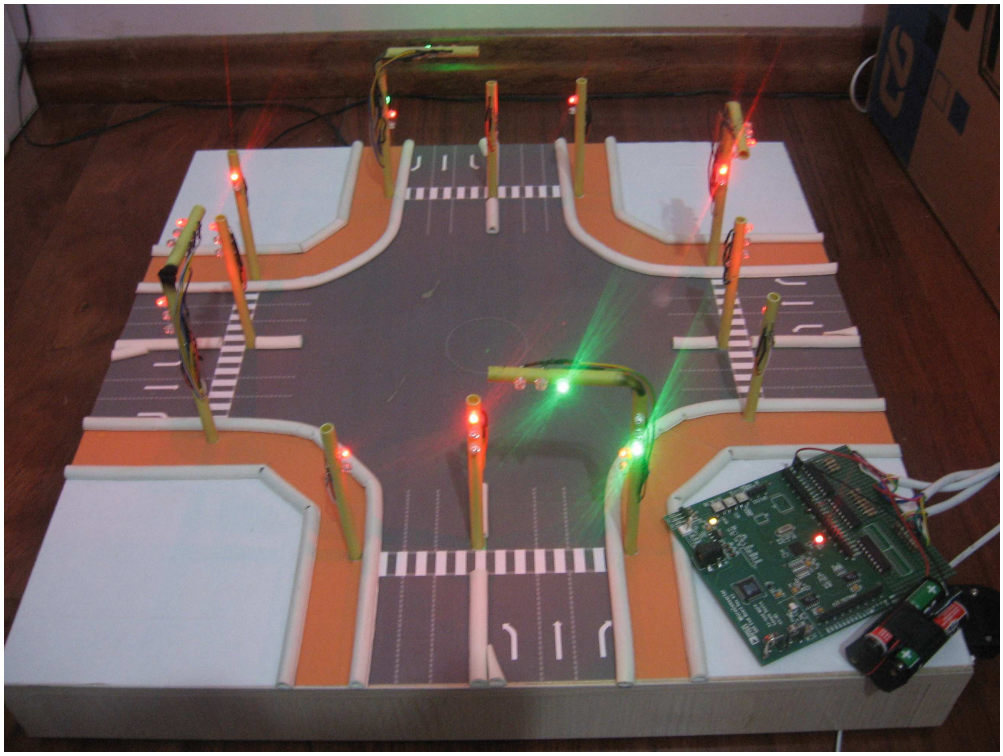


Şekil 5.4. Örnek modelde üçüncü program süreleri ve geçebilecek araç sayıları

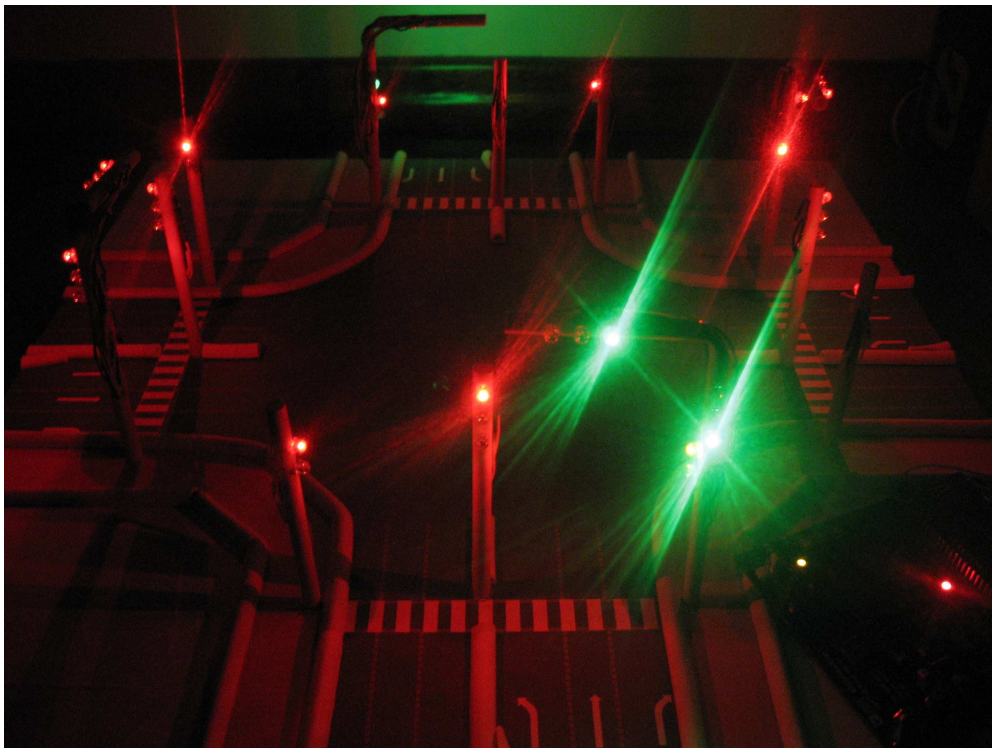


Şekil 5.5. Örnek modelde dördüncü program süreleri ve geçebilecek araç sayıları

Sistemin çalışır durumdaki hali Şekil 5.6 ve Şekil5.7’de görülmektedir.



Şekil 5.6. sistemin çalışması



Şekil 5.7. sistemin çalışması

BÖLÜM 6. SONUÇLAR VE ÖNERİLER

Bu çalışmada geliştirilen yöntem, tasarlanan kontrol cihazı ve prototip kavşak modeli kullanılarak başarılı bir şekilde uygulama yapılmış ve gerçekleştirilmiştir. Farklı trafik akışları için ve aynı zamanda yeşil dalga uyumunun istendiği ana arterler için uygulanabilirliği gözlemlenmiştir. Tasarlanan kavşak kontrol cihazı doğası gereği merkezi kontrol sistemlerine de uyarlanabilir.

Bundan sonraki çalışmalarda trafik yoğunluğuna ve araç sayısına göre geçiş sürelerini ayarlayabilen cihazlar veya kavşaklardaki cihazlar arası haberleşmenin kablosuz olarak yapıldığı cihazlar geliştirilebilir.

KAYNAKLAR

[1] TUNÇ, A. , Trafik Mühendsiliği ve Ugulamaları, 2003, 5:454-491

[2] ŞAKAR, M. , Karayolları Trafik Kanunu ve İlgili Mezuat 1992.

[3] DİRİM, M. , Trafik Mühendisliği 2, 7: 473-572

[4] MicroConverter® 12-Bit ADCs and DACs with Embedded High Speed 62-kB Flash MCU, www.analog.com

[5] GÜMÜŞKAYA, H. , Mikroişlemciler ve 80581 Ailesi, Alfa Yayınları, 2000

[6] 8-Channel Source Drivers, 2981 THRU 2984, www.datasheetcatalog.com

[7] Led 'lerin Elektriksel Özellikleri, www.silisyum.net/htm/optoelektronik/ledlerin_elektriksel_ozellikleri.htm

EK. Tasarımda Kullanılan Sistemin Mikrodenetleyici Programı

```
//====== main.c ======
#include "inc/define.h"
#include "inc/func.h"

//================================================
void delay(int length)
{
while (length >=0)
   length--;
}


//================================================
//=== hepsi kırmızı ===
//================================================
void hepKRMZ(){
        P0 = hepKRMZ_p0;
        P2 = hepKRMZ_p2;
//      P3 = hepKRMZ_p3;
        y2 = OFF;
        s2 = OFF;
        k2 = ON;
        y3 = OFF;
        s3 = OFF;
}
//================================================
//=== ilk yanan sarı ışık secimi ===
//================================================

void ilkSariSecim(char ilkSariSec){
        switch (ilkSariSec) {
           case 1:
                        k2 = OFF;
                        k3 = OFF;
                        s2 = ON;
                        s3 = ON;
                        break;

        case 2:
                        k1 = OFF;
```

```
                    s1 = ON;
                    break;

            case 3:
                    k4 = OFF;
                    s4 = ON;
                    break;

            case 4:
                    k5 = OFF;
                    k6 = OFF;
                    s5 = ON;
                    s6 = ON;
                    break;
        }
}

//=============================================
//=== yeşil ışık secimi ===
//=============================================

void yesilSecim(char yesilSec){
        switch (yesilSec) {
            case 1:
                    s2 = OFF;
                    s3 = OFF;
                    y2 = ON;
                    y3 = ON;
                    break;

        case 2:
                    s1 = OFF;
                    y1 = ON;
                    break;

            case 3:
                    s4 = OFF;
                    y4 = ON;
                    break;

            case 4:
                    s5 = OFF;
                    s6 = OFF;
                    y5 = ON;
                    y6 = ON;
                    break;
        }
}
```

```
//=============================================
//=== son yanan sarı ışık secimi ===
//=============================================

void sonSariSecim(sonSariSec){
        switch (sonSariSec) {
           case 1:
                        y2 = OFF;
                        y3 = OFF;
                        s2 = ON;
                        s3 = ON;
                        break;

        case 2:
                        y1 = OFF;
                        s1 = ON;
                        break;

           case 3:
                        y4 = OFF;
                        s4 = ON;
                        break;

           case 4:
                        y5 = OFF;
                        y6 = OFF;
                        s5 = ON;
                        s6 = ON;
                        break;
        }
}

//=============================================
//=== Program ===
//=============================================

void durum1a(){
        ilkSariSecim(durumSecim[0]);
        time = a1sarilar;
}
void durum1b(){
        yesilSecim(durumSecim[0]);
        time= asamaDurumSuresi[asama][0];
}
void durum1c(){
        sonSariSecim(durumSecim[0]);
        time = a1sarilar;
}
/*-*-*/
```

```
void durum2(){
	hepKRMZ();
	time= asamaDurumSuresi[asama][1];
}
/*-*-*/
void durum3a(){
	ilkSariSecim(durumSecim[1]);
	time = a1sarilar;
}
void durum3b(){
	yesilSecim(durumSecim[1]);
	time= asamaDurumSuresi[asama][2];
}
void durum3c(){
	sonSariSecim(durumSecim[1]);
	time = a1sarilar;
}
/*-*-*/
void durum4(){
	hepKRMZ();
	time= asamaDurumSuresi[asama][3];
}
/*-*-*/
void durum5a(){
	ilkSariSecim(durumSecim[2]);
	time = a1sarilar;
}
void durum5b(){
	yesilSecim(durumSecim[2]);
	time= asamaDurumSuresi[asama][4];
}
void durum5c(){
	sonSariSecim(durumSecim[2]);
	time = a1sarilar;
}
/*-*-*/
void durum6(){
	hepKRMZ();
	time= asamaDurumSuresi[asama][5];
}
/*-*-*/
void durum7a(){
	ilkSariSecim(durumSecim[3]);
	time = a1sarilar;
}
void durum7b(){
	yesilSecim(durumSecim[3]);
	time= asamaDurumSuresi[asama][6];
}
```

```
void durum7c(){
        sonSariSecim(durumSecim[3]);
        time = a1sarilar;
}
/*-*-*/
void durum8(){
        hepKRMZ();
        time= asamaDurumSuresi[asama][7];
}
/*-*-*/
void durum9(){
        k7 = OFF;
        y7 = ON;
        time= asamaDurumSuresi[asama][8];
}
/*-*-*/
void durum10(){
        y7 = OFF;
        k7 = ON;
        time= asamaDurumSuresi[asama][9];
}
//=============================================

void programlar(){
        if(time == 0){
                switch (kontrol) {

           case 0:
                        durum1a();break;
           case 1:
                        durum1b();break;
           case 2:
                        durum1c();break;
           case 3:
                        durum2();break;
           case 4:
                        durum3a();break;
           case 5:
                        durum3b();break;
           case 6:
                        durum3c();break;
           case 7:
                        durum4();break;
           case 8:
                        durum5a();break;
           case 9:
                        durum5b();break;
          case 10:
                        durum5c();break;
```

```
            case 11:
                        durum6();break;
            case 12:
                        durum7a();break;
            case 13:
                        durum7b();break;
            case 14:
                        durum7c();break;
           case 15:
                        durum8();break;
           case 16:
                        durum9();break;
           case 17:
                        durum10();break;
                  } //switch
           /**********************/
                  kontrol++;
                  if(kontrol == 18){
                  kontrol = 0;
                  }
           /**********************/
           } // if
}
//================================================

//================================================
//=== flasor ===
//================================================
void flasor(){
           if(P3==0){
                  P3= 0x90;
                  P2= 0x48;
                  P0= 0x2A;
           }else {P3=0; P2=0; P0=0;}
}
//================================================

char asamaKarsilastir(char asama1St, char asama1Dk, char asama2St, char
asama2Dk){
           if((asama1St<asama2St)                                                    ||
((asama1St==asama2St)&&(asama1Dk<asama2Dk))) return 1;
           return 0;
}
//================================================

void intTic () interrupt 10{
           int i,j,temp;
           TIMECON =   0x53;  // configure the Time Interval Counter to count seconds
           time--;
```

```
//------Yeşil Dalga----------------------------------------------------------
        if(yd==0){
                delay(100);
                if(yd==1)return;
                printf("\nYesil Dalga\n");
                yesilDalga=1;
                temp=yesilDalgaSure-1-asamaDurumSuresi[0][1];
        }

        if(yesilDalga==1){
                temp--;
                if(temp==0){
                        if(y2==1 || y3==1){
                                time = asamaDurumSuresi[asama][0];
                        }else{
                                time=0;
                                for(j=0;j<4;j++){
                                        if(durumSecim[j]==yesilDalgaIsik){
                                                switch(j){
                                                case 0: kontrol=17; break;
                                                case 1: kontrol=3; break;
                                                case 2: kontrol=7; break;
                                                case 3: kontrol=11; break;
                                                }//switch
                                        }//if
                                }//for
                                hepKRMZ();
                        }
                yesilDalga=0;
                }
        }
//------Aşama----------------------------------------------------------------

        for(i=0;i<5;i++){

                if(i==4)j=0;
                else j=i+1;

                // asama1<asama2

        if(asamaKarsilastir(asamaSureSaati[i][0],asamaSureSaati[i][1],asamaSureSaa
ti[j][0],asamaSureSaati[j][1]) ) {

                        if(                 (asamaSureSaati[i][0]<HOUR)                 ||
((asamaSureSaati[i][0]==HOUR) && (asamaSureSaati[i][1]<=MIN)) ) {

                                if(             (asamaSureSaati[j][0]>HOUR)             ||
((asamaSureSaati[j][0]==HOUR) && (asamaSureSaati[j][1]>MIN)) ) asama=i;
                        }
```

```
                }else if(((asamaSureSaati[i][0]<HOUR) ||
((asamaSureSaati[i][0]==HOUR) && (asamaSureSaati[i][1]<=MIN) )) ||
(asamaSureSaati[j][0]>HOUR) || ((asamaSureSaati[j][0]==HOUR) &&
(asamaSureSaati[j][1]>MIN)))asama=i;
        }

        switch (asama) {

        case 0:
        case 1:
        case 2:
        case 3:
                programlar();break;
        case 4:
                flasor();break;
        }
}
//=============================================
void ticInit (){
//Configure Time Interval Counter
        TIMECON = 0x53;   // configure the Time Interval Counter to count seconds
        INTVAL = 0x01;      // 1 saniye
//Configure External Interrupt
        IEIP2 = 0xA4;                   // enable TIC interrupt
/*
        EX0 = 1;
        IT0 = 0;
*/
   EA = 1;                              // enable interrupts

}
//=============================================
void setSureler(){
        int i,j,temp_sec,temp_min,temp_hour;
        int temp[2];
        char answer;

        printf("MIKRODENETLEYICI TABANLI MODULER KAVSAK
KONTROL CIHAZI\n");
        printf("Tasarlayan: Onder BERBEROGLU\n");

//------RTC Ayarlama-----------------------------------------------------------
        printf("\nSu an saat %02BD:%02BD:%02BD\n",HOUR,MIN,SEC);
        printf("Saati degistirmek istiyor musunuz? E/H\n");
        scanf("%c",&answer);

        if (answer=='E')
        {
                printf("\nSaati giriniz Saat:Dakika:Saniye\n");
```

```
                scanf("%02d:%02d:%02d",&temp_hour,&temp_min,&temp_sec);

                TIMECON = 0x01;
                SEC = temp_sec;
                MIN = temp_min;
                HOUR = temp_hour;
        }

//------Parametreler-------------------------------------------------------
        printf("\nParametre degisecek mi?E/H\n");
        scanf("%c",&answer);

        if (answer=='E'){
//------Yesil Dalga Süresi-------------------------------------------------
                printf("\nYesil Dalga Suresini giriniz (saniye): \n");
                scanf("%d",&temp[0]);
                yesilDalgaSure=temp[0];
//------Yesil Dalga Süresi-------------------------------------------------
                printf("\n--- Yesil Dalga Isiklari ---");
                printf("\n'1': 2 ve 3 numarali Isik");
                printf("\n'4': 5 ve 6 numarali Isik \n");
                i=1;
                while(i==1){
                        printf("\nYesil Dalga Isigini seciniz:");
                        scanf("%d",&temp[0]);
                        switch(temp[0]){
                        case 1:
                        case 4:
                                i=0;
                                yesilDalgaIsik=temp[0]; break;
                        default:
                                i=1;
        printf("\n*** Hata! Sadece 1 ve 4 degerlerini girebilirsiniz.***\n");break;
                                }//switch
                }//while

//------Asama Saatleri-----------------------------------------------------
                for(j=0;j<5;j++){
                        printf("\n%d.Program Baslama Saatini giriniz. ss:dd\n",(j+1));
                        scanf("%02d:%02d",&temp[0],&temp[1]);
                        asamaSureSaati[j][0]=temp[0];
                        asamaSureSaati[j][1]=temp[1];
                }

//------Durum Sıra Secimi--------------------------------------------------
                printf("\n\nIsik siralamalari asagidaki listeye gore yapiniz. \n");
                printf("\n '1' : 2 ve 3 numarali Isik");
                printf("\n '2' : 1 numarali Isik");
                printf("\n '3' : 4 numarali Isik");
```

```
                printf("\n '4' : 5 ve 6 numarali Isik\n");
                for(j=0;j<4;j++){
                        i=1;
                        while(i==1){
                        printf("\n%d. Sirada kac numarali isik yanacak \n",(j+1));
                                scanf("%d",&temp);

                                switch(temp[0]){
                                case 1:
                                case 2:
                                case 3:
                                case 4:
                                        i=0;
                                        durumSecim[j]= temp[0]; break;
                                default:
                                        i=1;
                printf("\n*** Hata! Sadece 1,2,3,4 degerleri giriniz.***\n");break;
                                        }//switch
                        }//while
                }//for

//------Güvenlik Süresi-----------------------------------------------------------
                printf("\nGuvenlik Suresini giriniz (saniye): \n");
                scanf("%d",&temp);
                for(j=0;j<4;j++){
                        for(i=0;i<5;i++){
                                asamaDurumSuresi[j][2*i+1]= temp[0];
                        }
                }
//------Durum Süreleri------------------------------------------------------------
                for(j=0;j<4;j++){
                        for(i=0;i<5;i++){
                printf("\n%d.Program %d.Durum Suresi (saniye): ",(j+1),(2*i+1));
                                scanf("%d",&temp);
                                asamaDurumSuresi[j][2*i]= temp[0];
                        }
                }
//------------------------------------------------------------------------------
        }
        printf("\nSure ayarlamalari tamamlandi.\n");
}
//=============================================

void initAll(){
        P1 = 0x00;

        hepKRMZ();
        kontrol = 0;
```

```
	PLLCON = 0x06;
	//Configure the baud rate 9600
	T3CON = 0x80;
	T3FD  = 0x08;
	SCON  = 0x52;
}
//===============================================
//===============================================

void main(){
	initAll();

	setSureler();
	ticInit();

	for(;;){}
	return;
}
```

Define

```
//====== define.h ======
#ifndef __DEFINE_H__
#define __DEFINE_H__

//======================

#include <stdio.h>
#include <ADuC841.h>
//------------------------------

unsigned int    asama           = 0;
unsigned char kontrol           = 0;
unsigned char time              = 1;    // saniye
unsigned char yesilDalgaSure        = 15;
unsigned char yesilDalgaIsik= 1;
unsigned char yesilDalga    = 0;
sbit    yd = P3^2;

//------------------------------
unsigned char idata durumSecim[4]=
{1,2,3,4};

unsigned char idata asamaDurumSuresi[4][10]=
{
{28,2,15,2,10,2,17,2,10,2},
{20,2,15,2,10,2,15,2,10,2},
{35,2,15,2,10,2,20,2,10,2},
{15,2,15,2,10,2,15,2,10,2},
```

```
};
unsigned char idata asamaSureSaati[5][2]=
{
{6,0} ,
{10,0},
{16,0},
{20,0},
{0,0} ,
};

//------------------------------

#define         ON              1;
#define         OFF             0;

//------------------------------
/*Lambalar*/
sbit     y1 = P0^4; //degisti
sbit     s1 = P0^5; //degisti
sbit     k1 = P0^6; //değişti
sbit     y2 = P3^3;
sbit     s2 = P3^4;
sbit     k2 = P3^5;
sbit     y3 = P3^6;
sbit     s3 = P3^7;
//------------------------------
sbit     k3 = P2^0;
sbit     y4 = P2^1;
sbit     s4 = P2^2;
sbit     k4 = P2^3;
sbit     y5 = P2^4;
sbit     s5 = P2^5;
sbit     k5 = P2^6;
sbit     y6 = P2^7;
//------------------------------
sbit     s6 = P0^0;
sbit     k6 = P0^1;
sbit     y7 = P0^2;
sbit     k7 = P0^3;
//------------------------------

#define a1sarilar       1; //
#define a1sureD1        26; //
#define a1sureD2        2; //
#define a1sureD3        13; //
#define a1sureD4        2; //
#define a1sureD5        8; //
#define a1sureD6        2; //
#define a1sureD7        15; //
```

```
#define a1sureD8      2; //
#define a1sureD9      8; //
#define a1sureD10     2; //
//------------------------------

#define a2sarilar     1; //
#define a2sureD1      18; //
#define a2sureD2      2; //
#define a2sureD3      13; //
#define a2sureD4      2; //
#define a2sureD5      8; //
#define a2sureD6      2; //
#define a2sureD7      13; //
#define a2sureD8      2; //
#define a2sureD9      8; //
#define a2sureD10     2; //
//------------------------------

#define a3sarilar     1; //
#define a3sureD1      33; //
#define a3sureD2      2; //
#define a3sureD3      13; //
#define a3sureD4      2; //
#define a3sureD5      8; //
#define a3sureD6      2; //
#define a3sureD7      18; //
#define a3sureD8      2; //
#define a3sureD9      8; //
#define a3sureD10     2; //
//------------------------------

#define a4sarilar     1; //
#define a4sureD1      13; //
#define a4sureD2      2; //
#define a4sureD3      13; //
#define a4sureD4      2; //
#define a4sureD5      8; //
#define a4sureD6      2; //
#define a4sureD7      13; //
#define a4sureD8      2; //
#define a4sureD9      8; //
#define a4sureD10     2; //
//------------------------------

#define hepKRMZ_p3        0x20; //
#define hepKRMZ_p2        0x49; //
#define hepKRMZ_p0        0x4A; //

#endif
```

Function

```
//====== func.h ======
#ifndef __FUNC_H__
#define __FUNC_H__
//======================

//***asama1*************
void durum1a();
void durum1b();
void durum1c();
void durum2();
void durum3a();
void durum3b();
void durum3c();
void durum4();
void durum5a();
void durum5b();
void durum5c();
void durum6();
void durum7a();
void durum7b();
void durum7c();
void durum8();
void durum9();
void durum10();

void programlar();

//************************
void delay(int length);
void flasor();
char asamaKarsilastir(char asama1St, char asama1Dk, char asama2St, char asama2Dk);
void ticInit ();
void setSureler();
void initAll();


#endif
```

ÖZGEÇMİŞ

Önder Berberoğlu, 14.06.1979 yılında Sakarya'nın Kocaali ilçesinde doğdu. İlkokulu Kocaali'de, ortaokulu Akçakoca'da tamamladı. 1997 yılında Sakarya Fatih Teknik Lisesi Elektrik Bölümünden mezun oldu. Aynı yıl Sakarya Üniversitesi Elektrik ve Elektronik Bölümünü kazandı. 2001 yılında mezun oldu. 2001 ve 2003 yılları arası Konya 3. Ana Jet Üssünde Asteğmen rütbesiyle mütercim olarak vatani hizmetini tamamladı. 2004 yılında halen çalışmakta olduğu Adapazarı Merkez Belediyesi İmar ve Şehircilik Müdürlüğünde işe başladı. Bu süre zarfında iskân bürosunda çalışmanın yanı sıra içlerinde 700 konutluk toplu konut projesi de dâhil olmak üzere birçok projede denetçi olarak görev yaptı. 2005 yılında da Sakarya Üniversitesi Fen Bilimleri Enstitüsü Elektrik ve Elektronik Mühendisliği EABD Elektronik EBD'da yüksek lisans eğitimine başlamıştır.