KOCAELİ ÜNİVERSİTESİ * FEN BİLİMLERİ ENSTİTÜSÜ

AKILLI TRAFİK SİNYALİZASYONU

YÜKSEK LİSANS

Elektronik Müh. Osman DEMİRCİ

Anabilim Dalı: Elektronik ve Haberleşme Mühendisliği

Danışmanı: Yrd. Doç. Dr. Cihan KARAKUZU

KOCAELİ, 2007

KOCAELİ ÜNİVERSİTESİ * FEN BİLİMLERİ ENSTİTÜSÜ

AKILLI TRAFİK SİNYALİZASYONU

YÜKSEK LİSANS

Elektronik Müh. Osman DEMİRCİ

Tezin Enstitüye Verildiği Tarih : 4 Haziran 2007

Tezin Savunulduğu Tarih : 26 Haziran 2007

Tez Danışmanı	Üye	Üye
Yrd. Doç. Dr. Cihan KARAKUZU	Prof. Dr.Tülay YILDIRIM	Yrd. Doç. Dr. Mehmet YAKUT

KOCAELİ, 2007

ÖNSÖZ VE TEŞEKKÜR

Özellikle nüfusu ve araç trafiği kalabalık olan şehirlerde trafikte araç sürmek her geçen gün daha da sıkıntı verici bir durum arz etmektedir. Birçok ülkede üniversiteler ve özel kuruluşlar bu trafik sıkışıklığını hafifletebilmek, kontrol edebilmek en azından trafikte insanların daha rahat araç sürebilmelerini temin edebilmek, gidilebilecek yere en kısa zamanda gidebilmek ve özellikle petrole dayalı yakıt tüketimini en aza indirebilmek, çevrenin daha az kirlenmesini sağlamak, birim zamanda daha çok aracın trafikte seyredebilmesini temin etmek yani trafiğe akıcılık kazandırabilmek için çalışmaktalar ve değişik çözüm şekillerini hayata geçirmeye çalışmaktadırlar. Şu an dünyada en çok kullanılan sabit saykıllı trafik sinyalizasyonu da bu açıdan bakıldığında artık yerini daha değişik sinyalizasyon tekniklerine bırakmaktadır. Bunlar içerisinde en yaygınlaşmakta olanı bulanık mantık ile yapılan sinyalizasyon kontrol uygulamaları gelmektedir.

Ülkemizde henüz trafikte uygulama imkanı bulamamış olmakla birlikte her gün daha çok uygulanmaya başlayan bulanık mantık uygulamaları sanayide de kullanım alanı gittikçe artmaktadır. Yakın gelecekte hayatın her alanına girecek gibi görünmektedir. Beni bu tez çalışmamda bulanık mantık uygulaması üzerine araştırma yapmaya yönlendiren Yrd. Doç. Dr. Cihan Karakuzu beye teşekkürü borç bilirim. Bunun yanında derslerinden istifade ettiğim KOÜ öğretim görevlilerine ve derslere devam edebilmek için imkan veren çalıştığım Karayolları teşkilatına da teşekkür ederim.

İÇİNDEKİLER

ŞEKİL LİSTESİ

TABLO LİSTESİ

SİMGELER

$\cap$:Kesişim
$\cup$:Birleşim
$\_$:Tümleyen
$\in$:Elemanıdır
$\notin$:Elemanı değildir
μCON :Yoğunlaşma simgesi
$\sum$:Kartezyen toplam
μ :Üyelik fonksiyonu değeri
u* :Durulanmış değer
' :Değil
O :Bileşim operatörü
$\leq$:Küçük eşittir
$\geq$:Büyük eşittir
$\wedge$:Ve
$\vee$:Veya
Z* :Keskin değer çıkışı
q :Kuyruk(query)

Alt indisler

ty :Yeşil bulanık kontrolörü uzama zamanı
tk : Kırmızı bulanık kontrolörü uzaman zamanı
Tymax : Maksimum yeşil uzama zamanı
Tymin : Minimum yeşil uzama zamanı
Tkmax : Maksimum kırmızı uzama zamanı
Tkmin : Minimum kırmızı uzama zamanı

Kısaltmalar

COG : Ağırlık Merkezi(Center Of Gravity)
WA : Ağırlıklı Ortalama (Weighted Of Average)
E : Hata oranı(Error)
B-D : Doğu-Batı yönü
K-G : Kuzey-Güney yönü
QBD : Batı-Doğu yönündeki kuyruk uzunluğu
QKG : Kuzey-Güney yönündeki kuyruk uzunluğu
LCD : Sıvı kristal gösterge(Liquid Crystal Display)
FLC : Bulanık Mantık Kontrolörü
MIN : Minimum
MAX : Maksimum
ÜF : Üyelik fonksiyonu

AKILLI TRAFİK SİNYALİZASYONU

Osman DEMİRCİ

Anahtar kelimeler : Bulanık mantık denetleyicisi, trafik sinyalizasyonu, trafik kavşağı, akıllı trafik ışıkları, araç kuyrukları.

Bu tezde akıllı trafik kavşak sinyalizasyonu sistemi donanımsal olarak dizayn edildi ve gerçekleştirildi. Trafik ışıklarının zaman periyodu, bulanık mantık temelli akıllı kontrolörler tarafından belirlenmektedir. Şehir içi uygulamalarında en çok karşılaşılan modellerden birisi dört yönlü trafik kavşağı tipidir. Bu projede dört yönlü bir trafik kavşağı sinyalizasyonu sistemi gerçekleştirildi. Batı-doğu doğrultusu trafik akışının en yoğun olduğu ana arter ve Kuzey-güney yönü ise trafik akışının daha az olduğu yön olarak kabul edilmiştir. Uygulamanın gerçek trafik altında yapılması riskli olduğundan ve böyle bir ortamı bulmak zor olacağından, gerçekleştirme ortamı elektronik donanım ile simüle edilmiştir. Her bir yöne ait araç üreten devrelerle, üretilen araç bilgilerini simüle eden devreler yapılmıştır. Her bir araç için trafik sinyalizasyonuna göre gecikme zamanları ve geçiş sayıları incelenmiştir. Bulanık mantık kontrolünde kırmızı ve yeşil ışık süreleri kulvarlardaki araç kuyrukları giriş parametresi olarak alınarak belirlenmiştir. Işıkların zaman peryodu araç kuyruk uzunluğuna göre değişmektedir. Normal trafik sinyalizasyonunda kırmızı ve yeşil ışık süreleri sabittir ve değişmez. Bu projede uygulama sonuçları her iki normal saykıllı trafik sinyalizasyonu ve bulanık mantık kontrolörü ile yapılan sinyalizasyon için ayrı ayrı yapılmıştır. Geçen araç sayısı ve toplam gecikme zamanını belirlemek için tablolar oluşturuldu ve her bir kontrolör tipi birbirleriyle karşılaştırıldılar ve aralarındaki farklılıklar açıkça görüldü.

INTELLIGENT TRAFFIC SIGNALISATION

Osman DEMİRCİ

Keywords : Fuzzy logic controller, traffic signalisation, traffic intersection, intelligent traffic lights, extending period, vehicle ques.

In this thesis, an intelligent traffic intersection signalisation system is designed and implemented by hardware. The time period of the lights are determined by intelligent controller based on fuzzy logic. The most famous traffic control form is four way traffic lights intersection systems. In this study a four way traffic intersection simulation system was realized. The main arter was thoght west-east way in which the traffic density flows and North-South way was second arter of traffic flowing. Because of practising the application under real traffic conditions was risky and finding such an envorement was diffucult, the Project was simulated by electronic hardware. It was made for every direction a circuit of vehicle yielding and simulating circuit. For every vehicles, the delay time and the number of vehicle has been examined . In fuzzy logic control, for the period of red light and the green light, the vehicle ques in the lines was designed as a input parameter . The timing period of the lights has been changed to the long of the vehicle ques . In normal traffic signallisation the period of red and green light is constant and doesn't changes. In this Project both the application for normal traffic signalisation and fuzzy logic controlled traffic signalisation was made separately . To determine the number of passing vehicles and total delay time of vehicles the tables was formed and was compared the two type of controller to each other and the differencies between them was seen clearly.

1. AKILLI TRAFİK SİNYALİZASYONU

Zadeh tarafından 1965 yılında ortaya atılan bulanık mantıkla ilgili kavramlardan sonra bulanık mantık tekniği giderek endüstri ve mühendislik uygulamaları için daha çekici ve kullanışlı hale geldi ve trafik açısından da birçok bulanık uygulama ve teorisi ortaya atılmıştır. Trafikle ilgili yapılan çalışmaların en popüleri Pappis ve Mamdani tarafından 1977 yılında önerilendir. Bu çalışma iki yollu-tek şeritli ve tek kavşak için tasarlanmıştı. Bu çalışma daha sonra 1984 yılında Nakatsuyama tarafından art arda iki kavşak için yapıldı.1993 yılında Favilla tarafından bu çalışma çok şeritli tekbir kavşak için yapıldı. Fakat halen çok şeritli ve çok kavşaklı durumlar için bu yetersiz kalmaktaydı. Bütün bu çalışmalara daha sonra sola ve sağa dönüş durumları da eklendi. Bugün ise yapılan çalışmalarda yaya geçitleri, araç uzunlukları, çoklu şeritler ve çoklu kulvarlar da eklenerek daha kullanışlı hale getirildi. [1]

Otomotiv sektörü sürekli olarak büyüyen ve büyüdükçe de daha çok araç üreten bir sektör haline gelmiştir. Üretilen araçlar yollara çıktıkça Dünya genelinde özellikle nüfusu kalabalık olan büyük şehirlerde trafik her geçen karmaşık hal almaktadır. Artan araç sayısı trafik seyrini zor hale getirdiği gibi trafiğin yoğun olduğu saatlerde sinyalizasyon noktalarında ve kavşaklarda trafik akışı neredeyse tamamen durmaktadır. Kısa mesafelerde dahi bir yerden bir yere gitmek insanların önemli derecede zamanlarını alabilmekte, fazladan yakıt sarfiyatı, gürültü kirliliği, hava kirliliği ve insanların sağlığını bozucu diğer etkiler ortaya çıkmaktadır.

Trafik akışını normal hale getirebilmek için başta üniversiteler olmak üzere değişik özel ve kamu kuruluşları bütün dünyada bu konu üzerine yoğun bir şekilde çalışmaktadırlar. Yollara çıkan araç sayısı azaltılamayacağına göre yapılacak tek şey mevcut trafik akışını çeşitli şekillerde kontrol etmek ve yönlendirmek tek çare gibi görünmektedir.

Trafik kontrolünde en önemli noktalardan biri kavşak noktalarıdır. Buralara birçok yönden araç geliş gidişi olduğu için herhangi bir yöndeki araç geçişi en optimum sayıya ulaştırılabilirse trafik akışından önemli oranda düzelme sağlanabilir.

Kavşak noktalarında trafik akışı genelde trafik sinyalizasyonu ile düzenlenmektedir. Trafik sinyalizasyonunun olmadığı noktalarda araç geçişleri trafik işaretleri ile yönlendirilmektedir. Bu noktalara kontrolsüz kavşak denilmektedir. Bu istenilmeyen bir trafik kontrol şekli olsa da ülkemizde de olmak üzere dünya şehirlerinin birçoğunda kullanılmaktadır. Diğer kontrol şekli sabit saykıl süreli Yeşil, kırmızı ve sarı trafik ışıklarını kullanmak ya da trafik görevlisi bulundurmaktır.

Sabit süreli trafik sinyalizasyon ışığı kullanmak bugün Dünya genelinde en fazla kullanılan yöntemdir. Her bir renk için kavşak noktalarına göre belirli bir süre atanmaktadır. Bu süre bitene kadar ışık yanmakta süre bitiminde de ışık fazı biterek diğer renge ait sinyalizasyon ışığı yanmaktadır. Bu süre dahilinde kulvarda hiç araç olmasa dahi ışık süresi bitene kadar yanmaya devam etmektedir. Bu zaman zarfında diğer yöndeki kulvarda gereksiz yere ciddi araç kuyrukları oluşabilmektedir. Bu gibi durumlarda genelde devreye trafik polisleri veya görevlileri girerek araçları yönlendirmektedir. Bu olmadığı durumda da trafik yoğunluğu baş göstermekte, araç kuyrukları oluşmakta, hava gereksiz yere fazladan kirlenmekte, trafik altındaki insanların ve sürücülerin stresi ve sinirleri bozulmakta, gürültü kirliliği artmakta ve fazladan yakıt tüketimi ortaya çıkmaktadır.

Eğer ülke petrol üreten konumunda değilse bu durum daha da önem kazanmaktadır. Fazladan çalışan motor dışarıya daha fazla döviz ödenmesi anlamına gelmektedir. Artan trafik yoğunluğuyla beraber insanların yaşadıkları şehirler gittikçe daha da yaşanılması zor yerler haline gelmektedir. Bunun zararını sadece insanlar çekmemekte diğer canlılar dahi bu sıkıntıdan paylarına düşeni almaktadırlar.

Trafik akışını düzene koyabilmek için sabit saykıl ve diğer uygulamaların yanında bulanık mantık ile trafik sinyalizasyon uygulamaları da giderek daha fazla artış göstermektedir. Bulanık mantık ile yapılan sinyalizasyon uygulamalarında kulvarlardaki araç kuyrukları, yaklaşan araç sayıları ve yaya geçitlerinin durumları göz önünde bulundurularak uygun şekilde düzenlenen kural tabanları ile bulanık mantık denetim sistemleri geliştirilmektedir.

Bu tezde ele alınan akıllı trafik sinyalizasyon uygulaması da dört yönlü bir kavşak için düşünülmüş bulanık mantık denetimli sinyalizasyon uygulamasıdır. Uygulama donanımsal olarak da gerçeklenmiş ve gerekli test sonuçları tablolar halinde alınmıştır.

Gerçek hayatta trafik altında bu işlemi gerçekleme imkanı riskli olduğu için uygulama donanımsal simülatör oluşturarak gerçeklenmiştir. Uygulama projesinde her kulvar için değişik dört sınıfta ve uzunlukta taksi, minibüs, kamyon ve tır şeklinde araçlar oluşturulmuş ve bu araçların kendi kulvarları üzerinde hareketleri simüle ettirilmişlerdir. Kulvar üzerinde yeşil ışık yanıyorsa araç kavşakta karşı şeride geçerek yoluna devam etmektedir. Eğer kırmızı ışık yanıyorsa araç gelip durmakta ve kuyruk oluşturmaya başlamaktadır.

Donanımsal gerçekleştirilen uygulama projesi üzerinde trafik sinyalizasyonu için sabit saykılda yeşilin yanma süresi 60 sn , kırmızının yanma süresi ise 40 sn şehir içi ortalaması olarak kabul edilmiştir. Trafik sinyalizasyonu, bulanık mantık denetimli çalışırken yeşil ışığın yanma süresi 20 sn den başlamakta ve şerit üzerindeki araç kuyruğunun durumuna göre bu süre artırılmaktadır. Yeşil ışığın maksimum süresi 20 sn+ 5*20 sn olmak üzere toplam 120 sn olabilmektedir. Aynı şekilde kırmızı ışık da 20 sn ilk başlangıç süresi ile başlamakta ve şeritteki kuyruk durumuna göre gerekiyorsa artırılabilmektedir. Kırmızı ışığın süresi de en çok 20 sn+ 5*20 sn olabilmektedir.

Kavşak üzerindeki yollardan bir yön ana trafik akışının ağırlıklı olarak aktığı arterdir. Diğer yöndeki yol ise trafiğin daha az olduğu tali arter diye tabi ettiğimiz ikincil yol güzergahıdır. Burada amaç ana arter üzerindeki birim zamanda geçen

araç sayısını artırmak ve ana arter üzerindeki ve tali arter üzerindeki araç sayılarının ışıkta ve kuyrukta bekleme sürelerini en asgari seviyede tutmaktır. Özellikle trafik yoğunluğunun olduğu saatlerde araç kuyruklarının uzun olması sebebiyle kuyrukta bekleyen araç duraklama sürelerini minimum seviyede tutmak önem taşımaktadır.

2. BULANIK MANTIK TEMELLERİ

Bulanıklaştırma denildiği zaman evrensel konuşma diline yakın değerler kümesi akla gelir. Klasik mantıkta 1 ya da 0 değerleri işlem görürken, bulanık kontrol denildiği zaman daha geniş bir tanımlama söz konusudur. 1 ve 0 dışında diğer gerçel sayı kümeleri kümesi ile de işlem yapılabilecği ifade edilmiş olunur. Örneğin 0.1 değeri de bir değer, 0.5 değeri de bir değer ve 1 de bir değer ifade eder. Bulanık değerlerin her birisi üyelik fonksiyonu(membership fonction) $\mu A(x)$ şeklinde ifade edilen bir fonksiyonla elde edilir. Üyelik fonksiyonları , yapılacak kontrol algoritmasına göre en iyi sonucu verecek fonksiyonlardan birisi veya birkaçı seçerek yapılır. Üyelik fonksiyonları trimf, trapmf, gbellmf, gaussmf, gauss2mf, sigmf, dsigmf, psigmf, pimf, smf, singleton ve zmf gibi fonksiyonlardır. En çok kullanılan ve en iyi bilinen üyelik fonksiyonları üçgen ve trapez üyelik fonksiyonlarıdır. Bulanık küme içerisindeki her bir elemanın küme içerisinde bir üyelik derecesi vardır. Örneğin kısa denildiğinde sayısalda ya kısa ya da değildir. Ama bulanık ifade de bu ara değerlerle de ifade edilemekte bu da kontrole zenginlik katmaktadır. Kısa dediğimiz zaman "çok kısa", " az kısa"," "çok çok kısa", "çok az kısa" da bir kısa ifadesidir. Bunlarla da kısalık ifade edilebilir. Kısa olma işi ara değerler yayılarak verildiğinden konuya derinlik getirilmiş ve başka bir tarzda ifade etme imkanı tanınmış olur. Soğuk, sıcak, uzun, kısa, hızlı, kirli, yağlı ..vb. gibi ifadeleri bulanık mantıkla ifade ettiğimiz zaman bunu bir matematiksel form ile ifade etmek istersek $A=\{(x, \mu A(x) \mid x \; E \; X\}$ gibi gösterebiliriz. Burada X konuşma evrenini ve $\mu A(x)$ ise 0 ile 1 arasındaki üyelik değerlerini veren üyelik fonksiyonunu(ÜF) göstermektedir. [5,21]

Örnek 2.1

Hız değeri kümesi olarak H hız bulanık kümesi m/sn olarak hız ifadesini en yüksek hız değeri olan 100 m/sn esas alınarak (2.1) deki gibi tanımlanabilir.

$$H=\{0,10,20,30,40,50,60,70,80,90,100\} \qquad (2.1)$$

Hız={0/0+0.1/1+0.2/2+0.3/3+0.4/4+0.5/5+0.6/6+0.7/7+0.8/8+0.9/9+1/10} (2.2)

(2.2) deki ifadeden bulanık küme içerisinde 0 m/sn hız değerinin üyelik derecesi 0 olarak alınmıştır. Bu hız değeri içerisinde hızlı olarak bir değer ifade etmez. 30 m/sn ise 0.3 üyelik değeri ile "az hızlı" şeklinde ifade eder. Dijital mantık olsaydı bu belki de 0 "yok" değerini alacaktı. Fakat şimdi 0.3 üyelik değeri ile bir anlam ifade etmektedir. Aynı şekilde 100 m/sn değeri de 1 üyelik değeri ile "çok hızlı" şeklinde bir bulanık değer ifade etmektedir. Benzer şekilde kontrol edilmek istenilen her türlü fiziksel büyüklük de aynı biçimde bulanık kümelerle ifade edilebilir. [4,21]

Bulanık küme içerisindeki bir eleman kısmi olarak diğer bir kümeye de ait olabilir. Örneğin "yavaş" bulanık kümesine ait olan bir eleman, "normal" bulanık kümesinin bir kısmi üyesi olabilir. Aynı zamanda farklı bir üyelik derecesine sahip olabilir. Bir A bulanık kümesi ve bu küme içerisinde herhangi bir eleman bir üyelik derecesi ile tanımlanabilir.

$$A = \sum^{n} \mu_A(x_i) / x_i \qquad (2.3)$$

2.3 ifadesindeki " / " işareti bir elemanın üyeliğini ve üyelik derecesi arasına konulan bir ayırıcı gibidir. Bölme işlemi anlamına gelmez. [5,17]

$$\mu_{hızlı}(x) = \begin{cases} 1, x \geq 70 \\ (x-4)/2{,}30 \leq x \leq 70 \\ 0, \ddot{o}tekidurumlar \end{cases} \qquad (2.4)$$

(2.4) ifadesinde $\mu(x) \geq 0$ için küme bütün gerçel sayıları kapsar. Bir bulanık küme onun α kesimleriyle de ifade edilebilir. α kesimi, üyelik fonksiyonları belirli bir α değerinden büyük veya eşit olan bulanık kümenin tüm elemanlarını kapsayan kümeye denilir.

Örnek 2.2

H= 0.1/1+ 0.3/2+0.2/4+0.5/5+0/6+0.8/7+0.9/8+1/9 kabul edilirse

$H_\alpha = \{x \in X \mid \mu_A(x) \geq \alpha\}$ ifadesinde α=0.3 e eşit veya büyük α kesimler (2.5) ifadesinde görüldüğü gibi ifade edilebilir.

$H_{0.3}=\{2,5,7,8,9\}$ (2.5)

$H_{0.2}=\{2,4,5,7,8,9\}$ (α=0.2 e eşit veya büyük α kesimler) (2.6)

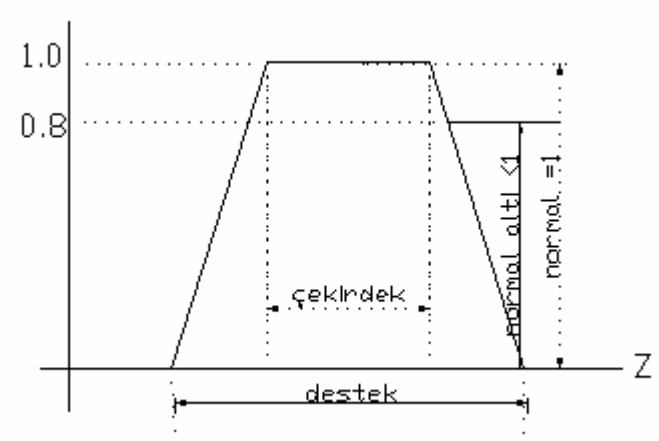
$H_{0.8}=\{7,8,9\}$ (α=0.8 e eşit veya büyük α kesimler) (2.7)

Destek : Bir bulanık kümeninin destek noktaları bütün $x \in X$ için $\mu_A(x) > 0$ yapan noktaların kümesidir.

Çekirdek: Bulanık bir kümenin çekirdeği $\mu_A(x) = 1$ olan değerlerin kümesidir.

Yükseklik : Bulanık bir kümenin yüksekliği α kesim boş değilken $\mu_A(x)$ in aldığı en büyük değerdir. Yüksekliği 1 değerinin altında kalan küme "normal altı", yüksekliği 1 olan değerler kümesi de "normal" olarak ifade edilmektedir.

Şekil 2.1 de şekilde destek, çekirdek ve yükseklik gösterilmiştir. [4]



Şekil 2.1 Destek, çekirdek ve yükseklik

2.1 Bulanık Küme İşlemleri

Bulanık mantık kümeleriyle yapılan işlemler normal kümelerle yapılan işlemlere benzemektedir.

Tümleyen: Bulanık bir H kümesinin tümleyeni $\mu_{\bar{H}} = 1 - \mu_H(x)$ şeklinde gösterilir.

Birleşme : A ve B kümeleri iki bulanık küme olsun ve bunların birleşme işlemi $\mu_{A\cup B}=\max\{\mu_{A(x)}, \mu_{B(x)}\}$ şeklinde ifade edilir.

Kesişim : A ve B kümeleri iki bulanık küme olsun ve bunların kesişme işlemi $\mu_{A\cap B}=\min\{\mu_{A(x)},\mu_{B(x)}\}$ şeklinde ifade edilir.

Yoğunlaşma işlemi : Bu sadece bulanık kümelere has bir işlemdir. Bu işlem $\mu_{CON(A)}(x) = \mu^{2\alpha}{}_A(x)$ şeklinde tarif edilir. Yoğunlaşma işleminde üyelik fonksiyonunun değerini $\mu=1$ dışında azaltıcı etkisi vardır. α sonsuza gidildikçe $\mu_{A(x)}$ sıfıra yaklaşır.

Genişleme işlemi: Bu işlem de bulanık kümelere has bir işlemdir. Bu işlem $\mu_{DIL(A)}(x) = \sqrt{\mu_A(x)}$ şeklinde ifade edilir. Yoğunlaşma işleminin tersidir. üyelik fonksiyonunun değerini $\mu=0$ dışında artırıcı etkisi vardır. Sonsuza (∞) gidildikçe $\mu_{A(x)}$, 1 değerine yaklaşır. [4,17]

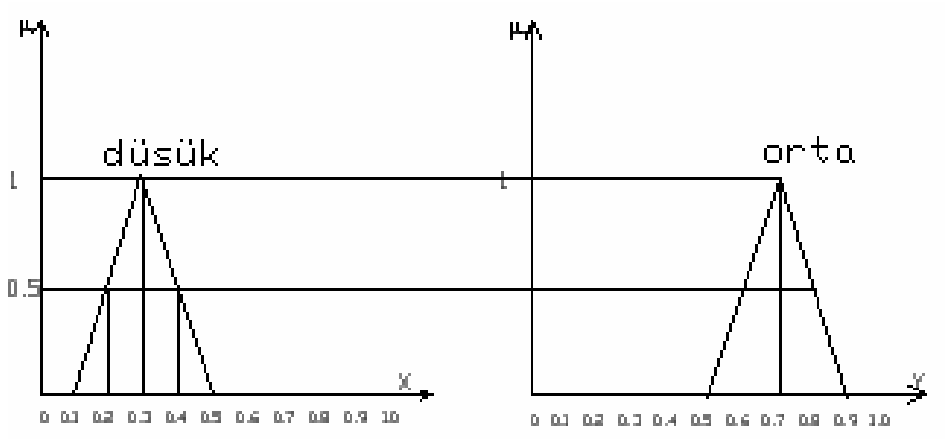
2.2 Bulanık Çıkartım

Bulanık çıkarım, kural tabanına göre giriş değerleri için bulanık denetim sisteminin çıkış keskin değerinin bulunması işlemidir. En çok kullanılan çıkartım mekanizması Mamdani çıkartımına dayanan max-min çıkartımıdır. Eğer cebirsel çarpım yolu tercih edilmiş ise max-prod çıkartımı kullanılır. [4,21]

2.2.1 Max-min çıkartımı

Tek kurala sahip bir denetim sistemi varsa ve x giriş, y ise çıkış değerini temsil ediyorsa

"If x düşük ise y orta " şeklinde bir kural tabanı ifadesi yazabiliriz. Bu durum Şekil 2.2 de görülmektedir.



Şekil 2.2 Max-Min çıkartımı

Tek kurallı denetim sisteminde eğer mamdani çıkartım mekanizması kullanılırsa (2.8) gibi bağıntı yazılabilir.

$$\mu R(x,y) = MIN(\mu düşük(x), \mu orta(y)) \quad (2.8)$$

Kuralın relasyon matrisi (2.9) ve (2.10) gibi sonuç verebilir.

$$\mu R(x=0.2,y=0.7) = MIN(\mu düşük(0.2), \mu orta(0.7)) = \min(0.5,1)=0.5 \quad (2.9)$$

$$\mu R(x=0.3,y=0.7) = MIN(\mu düşük(0.3), \mu orta(0.7)) = \min(1,1)=1 \quad (2.10)$$

Bu kurala ait bağıntı matrisini yazılarak Tablo 2.1 de X=0.4 keskin değerine göre bağıntı matrisi üzerinde çıkış değerleri oluşturulmaktadır.

Tablo 2.1 X=0.4 keskin değerine göre bağıntı matrisi

X	Y					
		0.5	***0.6***	***0.7***	***0.8***	***0.9***
	0.1	0	0	0	0	0
	0.2	0	0.5	0.5	0.5	0
	0.3	0	0.5	1	0.5	0
	0.4	0	0.5	0.5	0.5	0
	0.5	0	0	0	0	0

Tablodaki sonuç bulanık kümesi, Mamdani implikasyonu üyelik fonksiyonundan da bulunabilir.

$\mu R(x_1,y) = MIN(\mu düşük(x_1), \mu orta(y))$

$MIN(\mu düşük(0.4), \mu orta(y)) = MIN(0.5,y)$

y'nin tüm değerleri için;

min(0.5,0)=0

min(0.5,0.5)=0.5 (2.11)

min(0.5,1)=0.5

min(0.5,0.5)=0.5

min(0.5,0)=0 şeklinde sonuç değerlerini elde edilir.

(2.11) sonuçları değerlendirildiği zaman sonuç çıkış değerlerinin 0.5 ile sınırlandırılmış olduğunu görülür. Sonuçtan grafiksel anlamda sonuç bulanık kümesinin 0.5 yüksekliğinde kesildiği görülür.

R ve S relasyon matrisleri olmak üzere ;

Örnek 2.2.1 Max-Min Operatörü ve uygulaması

$$RoS = \begin{bmatrix} 0.4 & 0.6 & 0 \\ 0.9 & 1 & 0.1 \end{bmatrix} o \begin{bmatrix} 0.5 & 0.8 \\ 0.1 & 1 \\ 0 & 0.6 \end{bmatrix} = \begin{bmatrix} 0.4 & 0.6 \\ 0.5 & 1 \end{bmatrix}$$

Tablo 2.2 Relasyon matrisi ve çıkış değerleri

R	**Y1**	**Y2**	**Y3**		**S**	**Z1**	**Z2**
X1	0.4	0.6	0		**Y1**	0.5	0.8
X2	0.9	1	0.1		**Y2**	0.1	1
					Y3	0	0.6

max{min(0.4,0.5), min(0.6, 0.1), min(0, 0)} = max{ 0.4, 0.1, 0} = 0.4
max{min(0.4,0.8), min(0.6, 1), min(0, 0.6)} = max{ 0.4, 0.6, 0} = 0.6
max{min(0.9,0.5), min(1, 0.1), min(0.1, 0)} = max{ 0.5, 0.1, 0} = 0.5
max{min(0.9,0.8), min(1, 1), min(0.1, 0.6)} = max{ 0.8, 1, 0.1} = 1

2.2.2 Max-prod çıkartımı

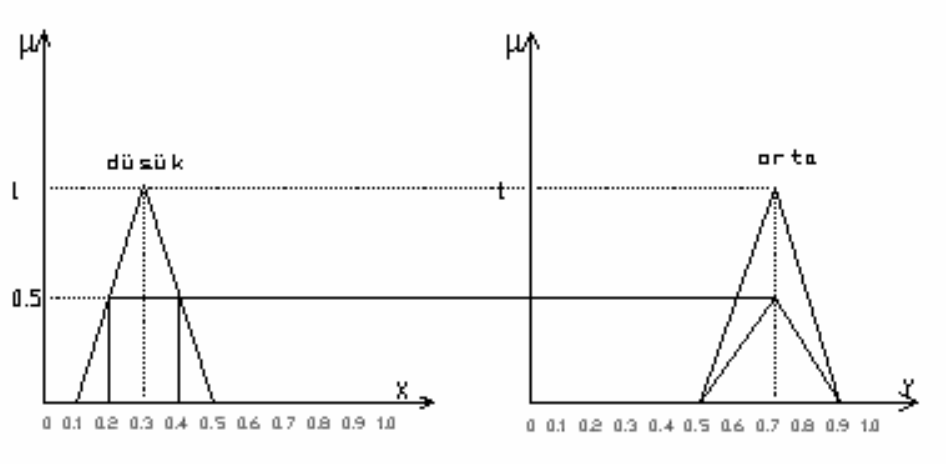
Örnek 2.2.1 de Max-Min çıkartımı ile yapılan örnek şimdi Max-Prod çıkartım yöntemiyle çözülerek, kuralan bağıntı cebirsel çarpım üzerinden bulunur.
$\mu R(x,y) = \mu$düşük$(x).\ \mu$orta(y)

Tablo 2.3 X=0.4 keskin değerine göre max-prod çıkartımı ve çıkış değerleri

		Y				
		0.5	***0.6***	***0.7***	***0.8***	***0.9***
X	***0.1***	0	0	0	0	0
	0.2	0	0.25	0.5	0.25	0
	0.3	0	0.5	1	0.5	0
	0.4	0	0.25	0.5	0.25	0
	0.5	0	0	0	0	0

X =0.4 keskin giriş değeri için bulanık kümeye B denilirse

B(0.4)= {0, 0.25, 0.5 , 0.25 , 0 } bulanık kümesi elde edilir.



Şekil 2.3 Max-prod çıkartımı

Şekil 2.3 deki grafikten de görüleceği üzere Y çıkışı 0.5 değeri ile sınırlandırılmıştır. Fakat y=0.6 ve 0.8 noktasında 0.25 değerini almaktadır.

Örnek 2.2.2 Max-prod çıkartımı ve uygulama örneği

$$\text{RoS} = \begin{bmatrix} 0.4 & 0.6 & 0 \\ 0.9 & 1 & 0.1 \end{bmatrix} o \begin{bmatrix} 0.5 & 0.8 \\ 0.1 & 1 \\ 0 & 0.6 \end{bmatrix} = \begin{bmatrix} 0.06 & 0.6 \\ 0.45 & 1 \end{bmatrix}$$

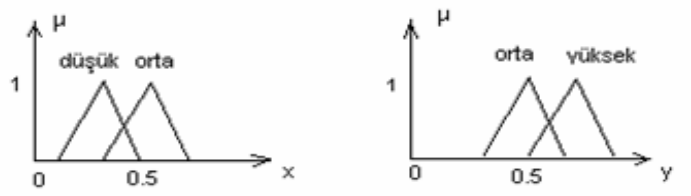
$\max\{0.4 \bullet 0.5, 0.6 \bullet 0.1, 0 \bullet 0\} = \max\{0.02, 0.06, 0\} = 0.06$

$\max\{0.4 \bullet 0.8, 0.6 \bullet 1, 0 \bullet 0.6\} = \max\{0.32, 0.6, 0\} = 0.6$

$\max\{0.9 \bullet 0.5, 1 \bullet 0.1, 0.1 \bullet 0\} = \max\{0.45, 0.1, 0\} = 0.45$

$\max\{0.9 \bullet 0.8, 1 \bullet 1, 0.1 \bullet 0.6\} = \max\{0.72, 1, 0.06\} = 1$

Örnek 2.2 3 İki kurallı denetim sistemi



Şekil 2.4 X ve Y üyelik fonksiyonları

X={0.1,0.15,0.2,0.25,0.3,0.35,0.4,0.45,0.5,0.55,0.6,0.65,0.7}

Y={0.3,0.35,0.4,0.45,0.5,0.55,0.6,0.65,0.7,0.75,0.8,0.85,0.9}

X ve Y temel kümeleri seçilir.

Şart 1: IF X= Düşük THEN Y=Yüksek

Şart 2: IF X= Orta THEN Y=Orta

Birinci kural için
µx düşük(x)={0,0.25,0.5,0.75,1,0.75,0.5,0.25,0,0,0,0,0}

µy yüksek(y)={0,0,0,0,0,0.25,0.5,0.75,1,0.75,0.5,0.25,0}

µR1(x,y)=MIN{µx düşük(x), µy orta(y)}

Her iki vektörün çapraz çarpımı 2.12 de görülmektedir.

$$R1 = \mu_x^T . Düşüş \otimes \mu_y yüksek = \begin{pmatrix} 0 \\ 0.25 \\ 0.5 \\ 0.75 \\ 1 \\ 0.75 \\ 0.5 \\ 0.25 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0.25 & 0.5 & 0.75 & 1 & 0.75 & 0.5 & 0.25 & 0 \end{pmatrix} \quad (2.12)$$

Tablo 2.4 Birinci kural için ilişki(relasyon) matrisi

R1		Y												
		0.3	0.35	0.4	0.45	0.5	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9
X	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0
	0.15	0	0	0	0	0	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0
	0.2	0	0	0	0	0	0.25	0.5	0.5	0.5	0.5	0.5	0.25	0
	0.25	0	0	0	0	0	0.25	0.5	0.75	0.75	0.75	0.5	0.25	0
	0.3	0	0	0	0	0	0.25	0.5	0.75	1	0.75	0.5	0.25	0
	0.35	0	0	0	0	0	0.25	0.5	0.75	0.75	0.75	0.5	0.25	0
	0.4	0	0	0	0	0	0.25	0.5	0.5	0.5	0.5	0.5	0.25	0
	0.45	0	0	0	0	0	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0
	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0
	0.55	0	0	0	0	0	0	0	0	0	0	0	0	0
	0.6	0	0	0	0	0	0	0	0	0	0	0	0	0
	0.65	0	0	0	0	0	0	0	0	0	0	0	0	0
	0.7	0	0	0	0	0	0	0	0	0	0	0	0	0

İkinci kural için üyelik değerlerine göre kümeler yazılırsa

μx orta(x)={0,0,0,0,0,0.25,0.5,0.75,1,0.75,0.5,0.25,0}

μy orta(y)={0,0.25,0.5,0.75,1,0.75,0.5,0.25,0,0,0,0,0}

R2= μx^{T}*orta* $\circ$ μ_{y}orta şeklinde gerekli değerler yerlerine konulursa Tablo 2.5 oluşur.

Tablo 2.5 İkinci kural için relasyon matrisi

R2		Y												
		0.3	0.35	0.4	0.45	0.5	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9
X	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0
	0.15	0	0	0	0	0	0	0	0	0	0	0	0	0
	0.2	0	0	0	0	0	0	0	0	0	0	0	0	0
	0.25	0	0	0	0	0	0	0	0	0	0	0	0	0
	0.3	0	0	0	0	0	0	0	0	0	0	0	0	0
	0.35	0	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0	0	0	0	0
	0.4	0	0.25	0.5	0.5	0.5	0.5	0.5	0.25	0	0	0	0	0
	0.45	0	0.25	0.5	0.75	0.75	0.75	0.5	0.25	0	0	0	0	0
	0.5	0	0.25	0.5	0.75	1	0.75	0.5	0.25	0	0	0	0	0
	0.55	0	0.25	0.5	0.75	0.75	0.75	0.5	0.25	0	0	0	0	0
	0.6	0	0.25	0.5	0.5	0.5	0.5	0.5	0.25	0	0	0	0	0
	0.65	0	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0	0	0	0	0
	0.7	0	0	0	0	0	0	0	0	0	0	0	0	0

R1 ve R2 relasyon matrislerine MAX operatörü uygulandığında tüm kural tabanının bağıntı matrisi elde edilir.

Tablo 2.6 da elde edilen değerler Tablo 2.4 ve Tablo 2.5 in aynı satır-sütuna denk gelen noktalarındaki değerlerin maksimum değerde olanlarının alınmasıyla elde edilmiştir. Böylece iki kurallı denetim sistemine göre her bir kural için elde edilen relasyon(ilişki) matrisleri oluşturulmuş ve bu matrislerin satır-sütun değerlerinden maksimum olanlar alınarak max-prod işlemi yapılmıştır.

Kural tabanında daha çok sayıda kural olsaydı oluşturulacak bağıntı matrisi sayısı, kural tabanındaki kural sayısı kadar olacaktı. Örneğin 12 adet kural olsaydı 12 adet bağıntı matrisi oluşturmak gerekecektir. Daha sonra bu 12 adet bağıntı matrisi için satır-sütun değeri en yüksek olan matrisin üyelik değeri alınarak en son Tablo 2.6 gibi bir matris oluşturularak sonuca gidilecekti.

Tablo 2.6 Max operatörü uygulanmış tablo değerleri

MAX		Y												
		0.3	0.35	0.4	0.45	0.5	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9
X	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0
	0.15	0	0	0	0	0	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0
	0.2	0	0	0	0	0	0.25	0.5	0.5	0.5	0.5	0.5	0.25	0
	0.25	0	0	0	0	0	0.25	0.5	0.75	0.75	0.75	0.5	0.25	0
	0.3	0	0	0	0	0	0.25	0.5	0.75	1	0.75	0.5	0.25	0
	0.35	0	0.25	0.25	0.25	0.25	0.25	0.5	0.75	0.75	0.75	0.5	0.25	0
	0.4	0	0.25	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.25	0
	0.45	0	0.25	0.5	0.75	0.75	0.75	0.5	0.25	0.25	0.25	0.25	0.25	0
	0.5	0	0.25	0.5	0.75	1	0.75	0.5	0.25	0	0	0	0	0
	0.55	0	0.25	0.5	0.75	0.75	0.75	0.5	0.25	0	0	0	0	0
	0.6	0	0.25	0.5	0.5	0.5	0.5	0.5	0.25	0	0	0	0	0
	0.65	0	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0	0	0	0	0
	0.7	0	0	0	0	0	0	0	0	0	0	0	0	0

X=0.45 için gerçekleşen sonuç bulanık kümesi 2.13 de görüldüğü gibi elde edilmiştir.

$\mu_{SON}(0.45) = (0, 0.25, 0.5, 0.75, 0.75, 0.75, 0.5, 0.25, 0.25, 0.25, 0.25, 0.25, 0)$ (2.13)

2.2.3 Birden fazla kısmi şartın olması durumu

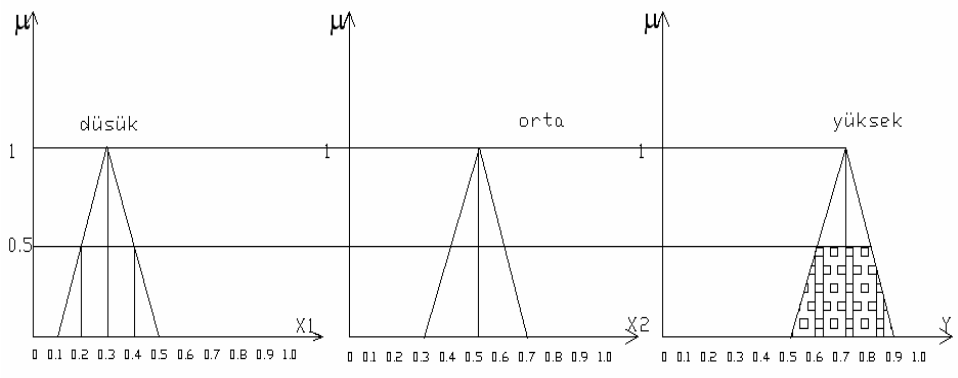
Birden fazla kısmi şartı olan bulanık mantık denetim sistemi 2.14 ifadesinde görüldüğü gibi ifade edilebilir.

IF x1=düşük AND x2=orta THEN y=yüksek (2.14)

Sonuç bulanık kümesi ise 2.15 deki gibi ifade edilebilir.

$\mu R(x1,x2,y) = MIN(\mu düşük(x1), \mu orta(x2), \mu yüksek(y))$ (2.15)

Grafik üzerinde gösterecek olursak Şekil 2.5 de görüldüğü şekilde ifade edilebilir.



Şekil 2.5 Birden fazla kısmi şart durumu

x1=0.4 ve x2=0.5 giriş değerleri için karşılama derecesi

y=MIN(μdüşük(x1), μorta(x2))
y=MIN(μdüşük(0.4), μorta(0.5))=MIN(0.5,1)=0.5

2.2.3.1 İki kurallı bulanık sistem

Max-Min çıkartımda eğer denetim tek bir kurala bağlı ise sadece çıkartımın Min kısmı anlamlı olmaktadır. 2 kurala sahip denetimlerde verilen 2 kural Prensipte birbirlerine OR (veya) operatörü ile bağlı olduklarından çıkartımın Max kısmı da anlam kazanmaktadır.

2.16 da iki kurallı denetim sisteminde max-min çıkartımına bir örnek görülmektedir. [1,2]

2 Kurala sahip bir sistemde

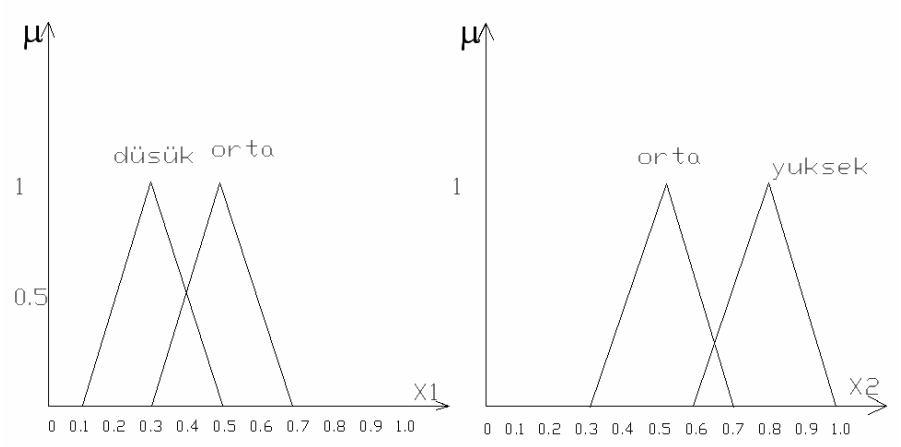
B1: IF x=düşük THEN y=yüksek (2.16)

B2: IF x=orta THEN y= orta

R sonuç ilişkisi R= B1 U B2 olur.

Sonuç bulanık kümesi ifadesi;

μR(x,y) = MAX(μB1(x,y), μB2(x,y))



Şekil 2.6 iki kurallı denetim üyelik fonksiyonları

X1=0.45 giriş değeri seçilirse; 1.kural için karşılama derecesi:

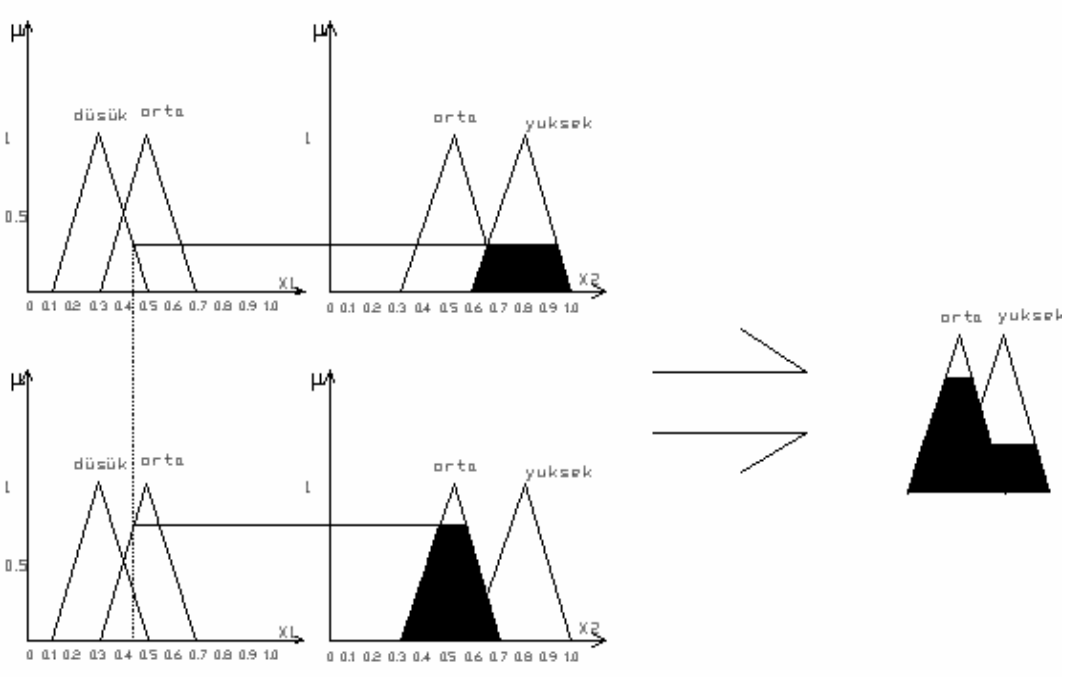
Y1= μxdüşük(0.45)=0.25

2.kural için karşılama derecesi:

Y2= μxorta(0.45)=0.75 olur.

1.kuralda 0.25 yüksekliğinde ve 2.kuralda 0.75 yüksekliğinde sınırlı bulanık küme elde edilir.Her iki küme MAX operatörü (OR işlemi) ile birleştirilir ise sonuç bileşik bulanık kümesi ortaya çıkar. 2.6 şeklinde her iki kuralın oluşturduğu grafiksel birleşim daha net olarak görünmektedir. [4,16,17]

İki kurallı denetim sistemine bir uygulama olarak, Örnek 2.2.3 de iki kurallı max-prod çıkarımına örnek verilmiştir. İki kuarallı denetim mantığını daha iyi görebilmek için örnek 2.2.3 tekrar incelenebilir.



Şekil 2.7 İki kural ile Max-min çıkarımı

2.3 Durulandırma Teknikleri

Bulanık çıkarımında en son adım, durulandırma işlemidir. Amaç çıkışta bir keskin değer elde etmektir. Durulandırma işleminde keskin çıkış değeri elde edilirken, giriş değerlerinin kural tabanındaki her bir kural ile işleme girmesi sonucunda ortaya çıkan çıkış bulanık değerleri uygun şekilde değerlendirilir. Bu değerlendirme işlemi ile durulandırmanın ne şekilde olacağı kararı verilmiş olur. Bir çok durulandırma yöntemi vardır. Hepsinin avantaj ve dezavantajlı yönleri vardır. Uygulama yapılırken hangi durulama yönteminin kullanıldığı önem arzetmektedir. Mikrodenetleyici ya da bilgisayar kullanılıyorsa bulanık setler üzerinde işlem yapmak, zaman ve kaynak gerektireceğinden sonuç keskin değerinin elde edilmesi işlemi çok uzun sürebilir veya kontrol çıkışı hızlı olmayabilir. Bu da istenilmeyen bir durumdur. Kullanıcı bulanık mantık denetimi ile uygulama yaparken hangi durulandırma yönteminin kullanılmasının daha iyi olacağı konusunda araştırma yapmalı ve yapacağı iş ile bulanık mantık denetim mekanizmasını karşılaştırarak uygun yöntemlerden birisini seçmelidir. Genelde istenilen, keskin değer çıkış değerinin en hızlı ve etkili şekilde elde edilmesidir.

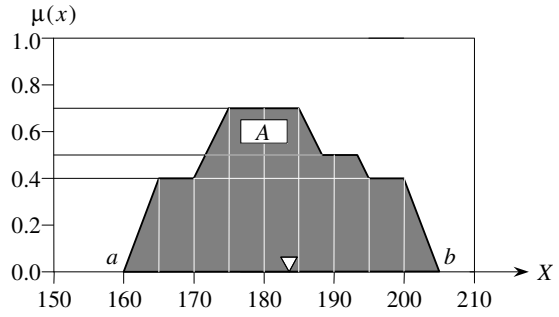
2.3.1 Ağırlık merkezi yöntemi

Birçok durulandırma metodu vardır, fakat bunlar içinde en çok tercih edilenlerden birisi merkezi ağırlık (centroid) tekniğidir. Bu teknikte çıkış bulanık değer setlerinin oluşturduğu toplamsal kütleyi dik bir doğruyla ortadan ikiye ayırıyormuşçasına eşit iki kütleye ayırarak tam orta kütle merkezi noktasını bulur. Matematiksel olarak bu kütle ağırlık merkezi (COG) noktası (2.17) deki gibi ifade edilir. [4,17]

.

$$COG = \frac{\int_a^b \mu_A(x)\,x\,dx}{\int_a^b \mu_A(x)\,dx} \tag{2.17}$$

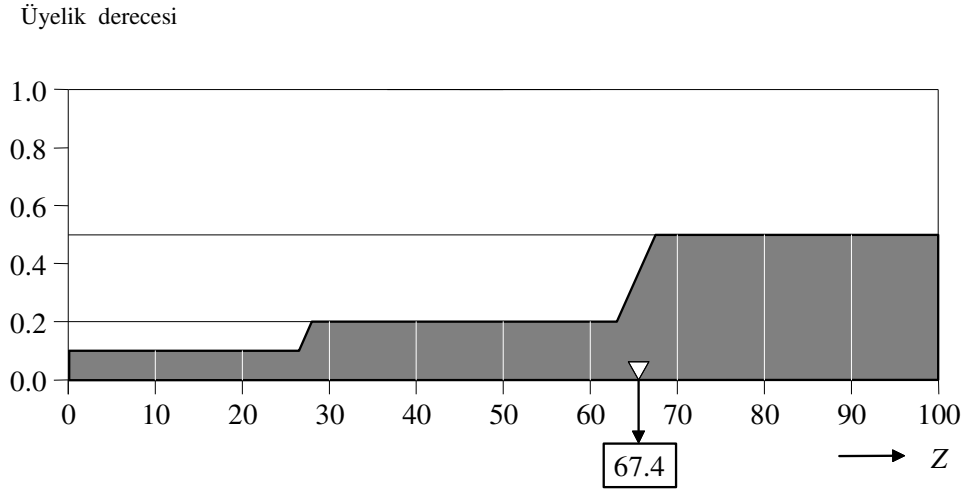
Merkezi ağırlık noktası(COG) durulaştırma metodu, bulanık setin a-b aralığı içerisinde ağırlık merkezini(A noktası) temsil eden bir nokta bulur. a-b aralığı üzerinde örnek noktalar alınarak, kabul edilebilirlik sınırları içerinde olan bir hesaplama yapar. Şekil 2.8 de görüldüğü üzere hesaplamalar her bir geometrik şeklin ağırlık merkezinin toplam ağırlık merkezine olan etkisinin elde edilmesiyle çıkış keskin değeri bulunmaktadır. Burada çıkış fonksiyonlarının simetrik olması gibi bir şart yoktur. Her bir fonksiyonun çıkış grafiği üzerinde etkisi elde edilerek çıkış keskin değeri elde edilir.



Şekil 2.8 Merkezi Ağırlık yöntemi

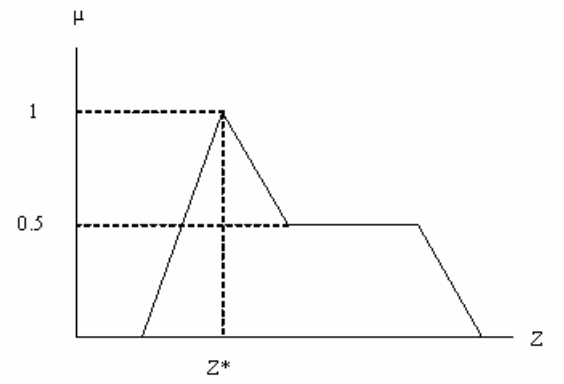
Şekil 2.9 da COG metoduna ait örnek bir çıkış değerinin hesaplama sonucu elde ediliş hali görülmektedir.

$$COG = \frac{(0+10+20)\times 0.1 + (30+40+50+60)\times 0.2 + (70+80+90+100)\times 0.5}{0.1+0.1+0.1+0.2+0.2+0.2+0.2+0.5+0.5+0.5+0.5} = 67.4 \qquad (2.18)$$



Şekil 2.9 Merkezi ağırlık yöntemi ile keskin çıkış elde edilmesi

2.3.2 Maksimum(max) üyelik yöntemi



Şekil 2.10 Max çıkış değeri

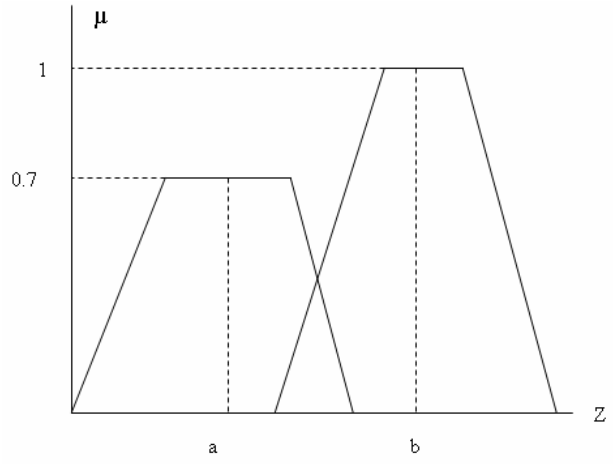
Şekil 2.10 da z* , Z uzayının μ uzayında en yüksek değere sahip olan keskin çıkış değeridir. Keskin çıkış değeri elde edilirken tepe noktası olarak en yüksek değere sahip olan keskin değer alınır. Elde edilen keskin değer belirli bir hata değerine

sahiptir. Tam olarak istenilen veya olması gereken çıkış değerini vermeyebilir fakat hesaplama yaparken işlem hızının yüksek olmasının istenildiği yerlerde teciedilebilir.

2.3.3. Ağırlıklı ortalama yöntemi(weighted average)

Sadece simetrik fonksiyonlarda kullanılır. COG metoduna benzer.

$$WA = \frac{\int_a^b \mu_A(x) * Z}{\int_a^b \mu_A(x)} \tag{2.19}$$



Şekil 2.11 Ağırlıklı ortalama yöntemi

$$Z^* = \frac{0.7 * a + 1 * b}{0.7 + 1} \tag{2.20}$$

Formül 2.20 de Z* keskin çıkış değerinin elde edilişi görülmektedir.

Uygulama projesinde çıkış fonksiyonu olarak singleton tipi üyelik fonksiyonu kullanılmış ve sugeno tipi ağırlıklı ortalama yöntemi ile çıkartım yapılmıştır.

2.3.3.1 Sugeno bulanık çıkartımı ile ağırlıklı ortalaması yöntemi

Mamdani bulanık çıkarımında bir çıkış fonksiyonunun merkezi noktasını bulmaya çalışıldığında bilgisayarda hesaplama açısından zor bir yöntem olmakta ve zaman almaktadır. Michino Sugeno bu zorluğu gördüğü için daha kolay bir çözüm şekli sunmuştur. Sugeno, çıkış üyelik fonksiyonu olarak tek bir singleton-impuls fonksiyonu kullanmıştır. Bir singleton evrensel kümede sadece bir noktada bir değerini almakta diğer bütün heryerde sıfır değerini almaktadır.

Sugeno stili bulanık çıkarım mekanizması mamdani tip bulanık çıkarım mekanizmasına çok benzemektedir. Sugeno, sadece sonuç kuralını değiştirdi. Bir bulanık seti yerine giriş değişkenlerinin bir matematiksel formunu kullandı. Sugeno stili bulanık çıkartım kuralı (2.21) deki gibi gösterilir.

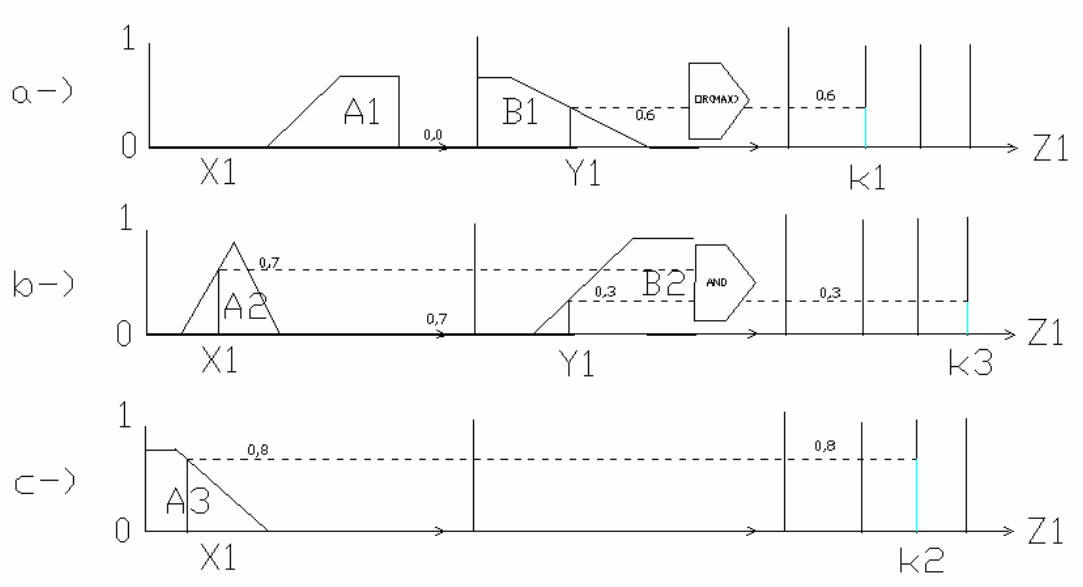
eğer x *A ise*
ve y *B ise* (2.21)
o zaman z $f(x, y)$ dir.

Burada x, y ve z dilsel değişkenlerdir. A ve B ise X ve Y çözüm aralığında bulanık seti ifade etmektedirler. F(x,y) ise matematiksel fonksiyonu ifade etmektedir. En çok kullanılan sıfır-mertebeli Sugeno bulanık modeli (2.22) gibi ifade edilebilir. [17]

eğer x A ise
ve y B ise (2.22)
o zaman z k dır.

(2.22) de ifadesinde k, bir sabit değeri ifade etmektedir. Bu durumda herbir çıkış bulanık seti bir sabit değeri ifade etmektedir. Bütün sonuç üyelik fonksiyonları farklı keskin çıkış değerine denk gelen tek bir singleton darbesiyle ifade edilmektedir.

Şekil 2.12 de görüldüğü üzere a,b ve c şıklarında kural tabanına göre çıkış singleton darbelerinin fonksiyon olarak nasıl seçildiği görülmektedir. a şıkkında kural tabanındaki iki kurallı denetim mekanizması veya ifadesiyle başlanmaktadır. Bu şekilde herhangi bir giriş değerinin en yüksek üyelik değerine sahip olan üyelik fonsiyonu değeri dikkate alınmaktadır. Şekil 2.12 de b şıkkında ise ve ifadesiyele çıkarım yapılmaktadır. Burada da kural tabanındaki iki giriş değerinden en küçük değere sahip olan üyelik fonksiyonu değeri dikkate alınmaktadır. Şekil 2.12 de c şıkkında ise tek kurallı bir giriş kuralı vardır. Bu kuralın üyelik değeri doğrudan çıkıştaki ilgili singleton fonksiyonunu sabit şekilde keserek sonuç çıkış değerinin o kural için çıkışa olan etkinlik değerini belirlemektedir. Her üç şıkta da görüldüğü üzere farklı çıkış değerlerinde kesilmektedir.



Şekil 2.12 Sugeno Bulanık çıkartımı ile şematik gösterimi

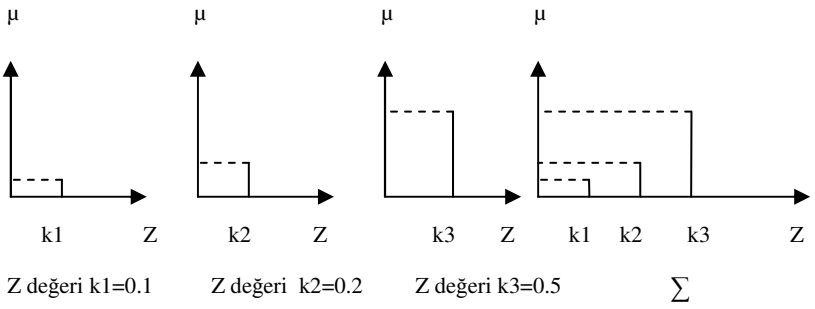
Kural 1: eğer X1, A1 VEYA Y1, B1 ise o zaman Z k1 dir.(veya ile)

Kural 2: eğer X1, A2 ise VE Y1, B2 ise o zaman Z k3 dür. (ve ile)

Kural 3: eğer X1, A3 ise o zaman Z k2 dir. (sabit değer)

Şekil 2.12 de a, b ve c şıklarından elde edilen sonuç çıkış singleton değerleri, durulandırma tekniklerinden uygun birisi ile işlenerek sonuç çıkış keskin değeri elde edilir. Çıkış değeri olarak singleton seçilmesinin sebebi hesaplama kolaylığı olmasındandır. Bilgisayar ile veya mikrodenetleyici ile işlem yaparken bulanık küme kural tabanı üzerinden işlem yaparken singleton çıkış fonksiyonunun kullanılması hesapalama kolaylığı ve hesaplama zamanının kısa olması gibi sebeplerle tercih edilmektedir. Uygulama devresinde bulanık kontrolörleri üzerinde yazılan program yazılımında da bu sebeplerle çıkış üyelik fonksiyonu olarak singleton darbeleri kullanılmıştır.

2.3.3.2 Sugeno yöntemi ile kural tabanı çıkışlarının toplamlarının bulunması



Şekil 2.13 Sugeno yöntemi ile sonuç çıkış değerlerinin toplamlarının çıkarılması

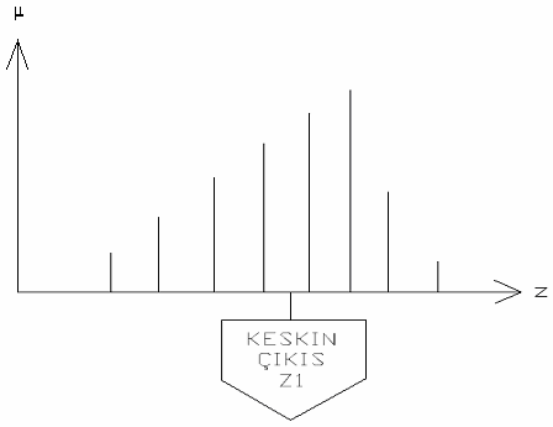
2.3.3.3 Sugeno ağırlıklı ortalama yöntemi

Sugeno yöntemi ile çıkış fonksiyonu olarak singleton pulse fonksiyonları kullanılnatadır. Kural tabanındaki her bir değer için çıkışta bir değre karşılık gelmektedir. Bu karşılık gelen değerler WA(Ağırlıklı ortalama) yöntemi ile 2.23 ifadesinde görüldüğü şekilde yerleine konularak olması gereken çıkış keskin değeri elde edilmektedir. K1, k2 ve k3 sabit değerlerine karşılık gelen üyelik değerleri

formülde yerine konularak keskin çıkış değerleri bulunmaktadır. Çıkış fonksiyonu olarak singleton kullanılmasının temel sebebi simetrik olması ve hesaplama kolaylığı sağlamasıdır.

$$WA = \frac{\mu(k1)\times k1 + \mu(k2)\times k2 + \mu(k3)\times k3}{\mu(k1) + \mu(k2) + \mu(k3)} = \frac{0.1\times 20 + 0.2\times 50 + 0.5\times 80}{0.1 + 0.2 + 0.5} = 65 \quad (2.23)$$

2.3.3.4 Sugeno yöntemi ile durulandırma



Şekil 2.14 Sugeno yöntemi ile durulandırma

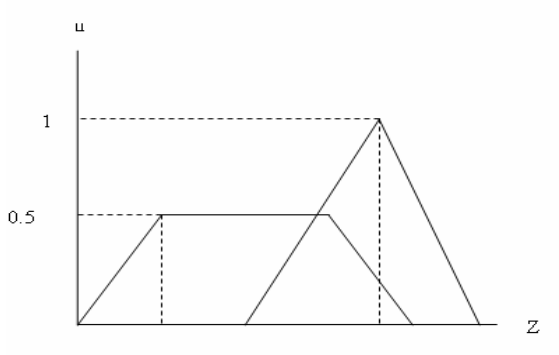
Sugeno yönteminde hesaplama kolay ve hızlıdır. Kontrol problemlerinde hususiyetle de lineer olmayan dinamik sistemlerin kontrol problemlerinde oldukça etkili olarak kullanılmaktadır.

2.3.4 Maksimumların ortalaması yöntemi

Şekil 2.15 de örnek olarak 1 üyelik değerinin olduğu ve 0.5 üyelik değerinin olduğu iki nokta ele alınırsa 2.24 ifadesinde de görüldüğü gibi en yüksek değere sahip olarak çıkış keskin değerinin belirlenmesinde bu noktaların ortalaması kullanılmaktadır.

$$Z^* = \frac{a+b}{2} \tag{2.24}$$

2.3.5 İlk veya son maksimum yöntemleri



Şekil 2.15 İlk veya son maksimum grafik gösterimi

Bu yöntemde maksimum yükseklik bir kıstas ve bant olarak düşünülürse sol kenardan yaklaşım için 0.5 üyelik yükseklik değerine karşılık 2 keskin çıkış değeri olmaktadır. Sağ taraftan yaklaşıldığında da 1 üyelik yükseklik değerine karşılık 8 keskin değeri olmaktadır. Sol veya sağdan yaklaşıldığında en yüksek tepe noktasına sahip olan değer keskin değer olarak alınmaktadır.

2.4 Bulanık Kontrol Yapısı

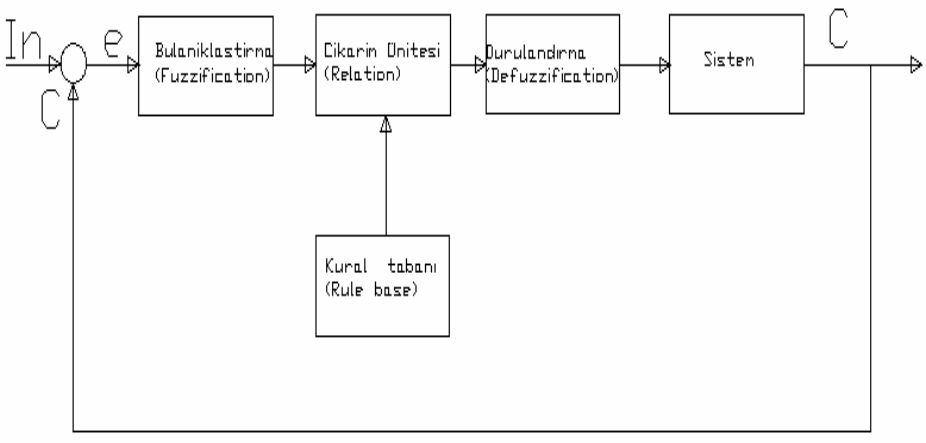
Şekil 2.16 da tipik bir bulanık mantık kontrol sistemine ait klasik kontrol şeması gösterilmiştir. Algoritmaya göre bulanıklaştırma işleminde ilk yapılacak iş, kontrol edilmesi istenilen giriş değerlerinin bulanık hale getirilmesidir. Daha sonra bulanık hale getirilen giriş keskin değerleri çıkarım ünitesi tarafından belirli bir kural tabanı gözetilerek işleme tabi tutulmaktadır. Kural tabanındaki her şart için ayrı bir çıkış bulanık değeri oluşturulmaktadır. Çıkış keskin değerinin bulanık

kontrol mekanizmasında istenilen şekilde uygun değeri alması isteniliyorsa kural tabanının uygun şeklde yapılandırılması gerekmektedir. Bu da biraz zamanla kazanılan tecrübeye ve uygulama yapma sayısına bağlıdır. [5, 7]

Çıkarım ünitesi tasarımcı tarafından seçilen üyelik fonksiyonlarına bağlı olarak kural tabanındaki şart ifadeleri de dikkate alınarak çıkarım yapmaktadır. Burada kullanılacak üyelik fonksiyonlarının seçimi de tasarımcı tarafından en uygun çıkış değerlerini sağlayacak şekilde seçilmelidir. Yoksa istenilmeyen çıkış değerleri alınabilir. Bu da kontrol işlemini sıkıntılı hale getirir.

Durulandırma ünitesi, kural tabanı ile oluşturan bulanık çıkış değerlerinin uygun şekilde seçilen bir yöntemle toplanarak istenilen çıkış keskin değerinin elde edildiği kısımdır. Çıkış keskin değeri elde edildikten sonra istenilen kontrol mekanizmasında kullanılabilir.

Günümüzde Matlab gibi matematiksel işlemler için tasarımlanmış paket programlar Bulanık kontrol uygulamaları başta olmak üzere kontrol problemleri için değişik çözümleri de beraberlerinde sunmaktadırlar. Bu sayede donanımsal olarak kontrol mekanizmalarını gerçekleştirmeden de kontrol çıkışlarının hangi değerleri alacağı veya sonuçların nasıl olacağı konusunda fikir sahibi olunabilmektedir. Hatta kontrol çıkışlarının hangi değerleri alması gerektiği de doğrudan hesaplanabilmekterdir. Bu konuda Matlab haricinde de birçok paket program mevcuttur.



Şekil 2.16 Bulanık kontrol şeması

Şekil 2.16 da görülen blok diyagramda kural tabanı tasarımcının bilgi ve tecrübesi ışığında yapılması istenilen kontrolün hassasiyeti ve mahiyetine göre en uygun şekilde ve etkili kurallar seçilerek oluşturulmaktadır. Kural tabanı ne kadar iyi tasarımlanırsa çıkarım ve durulandırma işlemi de o denli başarılı olmaktadır. Uygun şekilde tasarımı yapılmayan kural tabanından beklenen neticenin alınması zordur.

3. AKILLI TRAFİK SİNYALİZASYONU SİMÜLATÖRÜ

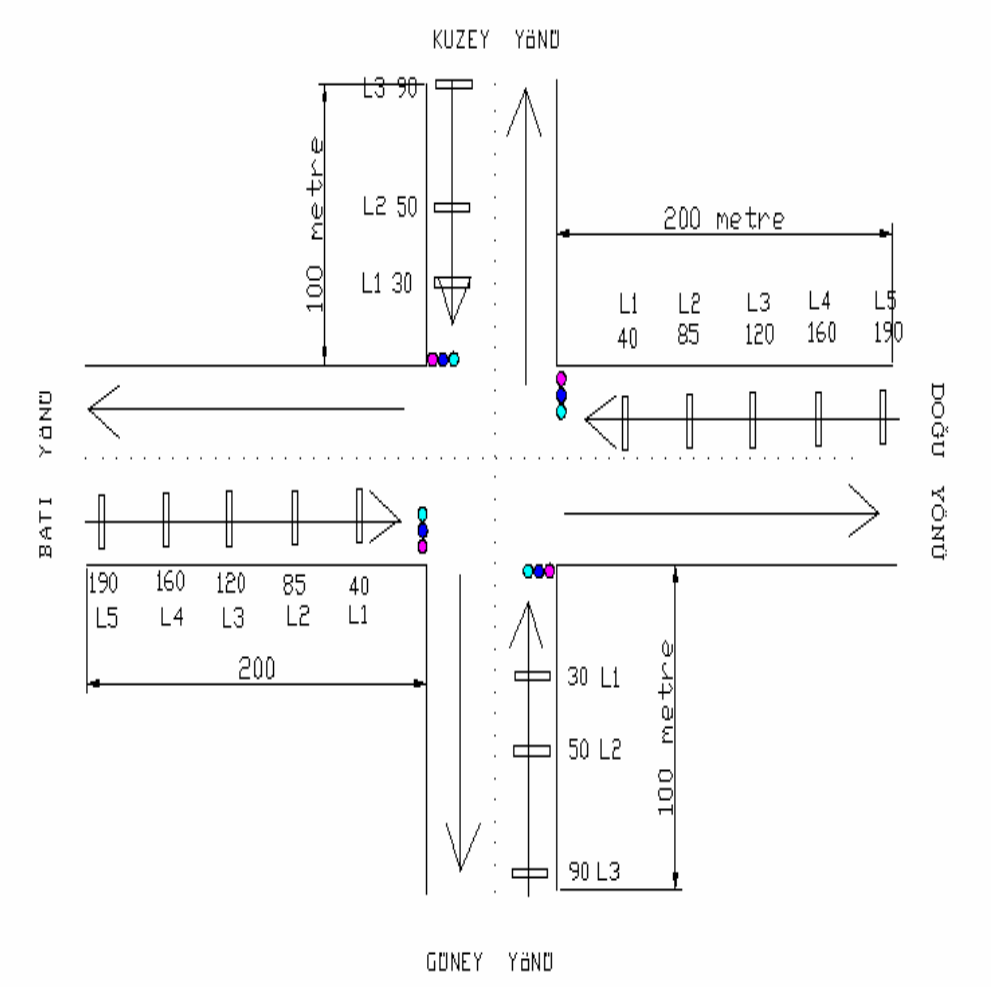
Akıllı Trafik Sinyalizasyonu projesi bulanık mantık denetimi uygulanarak gerçekleştirilmiş bir kontrol uygulamasıdır. Amaç günlük hayatta trafiği ayarlamada ve kontrol etmede kullanılan trafik sinyalizasyonu zaman sürelerini ayarlamak ve trafik akışına kolaylaştırıcı yönde katkı yaparak optimizasyon sağlamaktır. [10, 12,15]

Trafik Sinyalizasyonun günlük uygulamada en çok karşılaşılan şekli, sabit periyotlu trafik ışıkları süresi tayin ederek ışıkları kontrol etme şeklindedir. Uygulama projesi donanımsal bir gerçekleme olduğundan ya trafik altında yapılacak ya da simülasyon ortamı oluşturularak yapılması gerekliydi. Gerçek hayatta trafik altında çalışmak zor ve risk almayı gerektiren bir durumolduğundan ve de bu çalışma ortamını bulmak zor olacağından bir şekilde benzer simülasyon ortamını oluşturarak üzerinde çalışma yapmanın daha iyi bir çözüm olacağı düşünüldü.

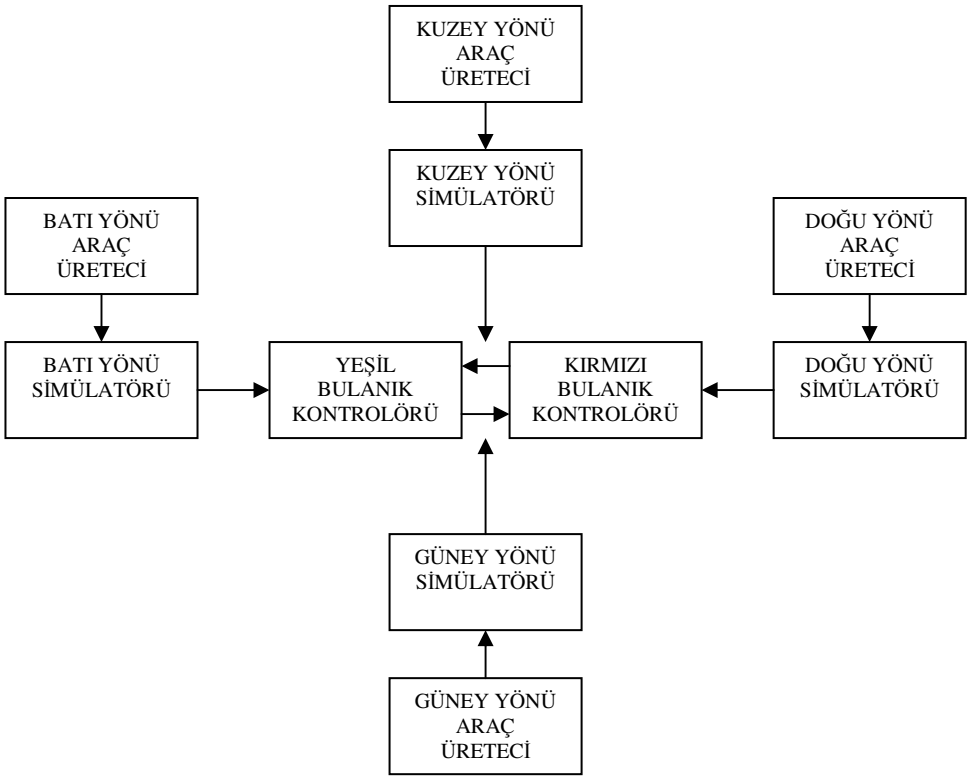
Gerçek hayatta bir kavşak noktası en çok karşılaşan şekillerinden birisi dört yönlü trafik kavşağıdır. Bu yönlerden birisi ana arter diye tabir edilen trafik yükünün en fazla olduğu ve araç sayısının fazla olduğu trafik doğrultusudur. Bu projede Doğu-batı yönü ana taşıyıcı arter olarak düşünülmüştür. Kuzey-Güney yönü ise trafik akışının diğerine göre daha az olduğu tali arter veya ikincil taşıyıcı tabir edilen diğer yöndeki taşıyıcı yol güzergahıdır.

Bir kavşakta dört yön olduğu için her yöne ait kulvarları simüle edebilmek için o yönlere ait araç üretici ve üretilen araçları simüle edebilmek için bir kontrol katı olmak üzere her bir yön için iki adet kontrol devresi olmak üzere dört yön için sekiz edet kontrol devresi kullanılmış ve kavşak içerisindeki araç trafiği kontrol edilmiştir. Kontrol devresi şekil 4.2 de çizilmiştir. Kontrol devresi blok diyagramından görüldüğü üzere her bir yön için bir araç üretici devre ve ilgili yön için simülatör devresinden oluşmaktadır. Merkezde görülen iki adet yeşil ve

kırmızı bulanık kontrolörü devreleri ise yeşil ve kırmızı ışığın uzaman zamanlarını belirleyen bulanık kontrolörü devreleridir. Şekil 3.1 de de uygulama projesinin gerçekleştirilmesinde kullanılan trafik kavşağı ve bu kavşak üzerinde trafik yönlerine göre konulan araç dedektörleri görülmektedir. Ana trafiğin akış yönü Batı-Doğu (B-D) istikametidir. Batı yönü 5 adet, doğu yönü 5 adet olmak üzere B-D istikametinde toplam 10 adet araç dedektörü konulmuştur.

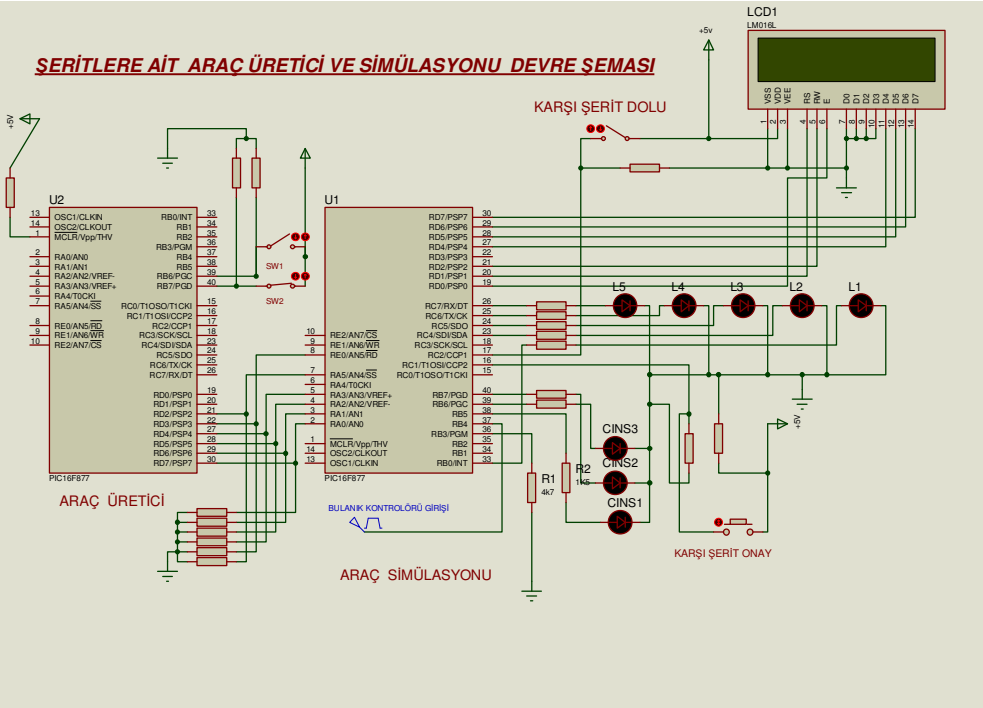


Şekil 3.1 Uygulama projesi trafik kavşağının yapısı



Şekil 3.2 Uygulama projesi blok diyagramı

Kuzey-Güney (K-G) istikametinde de kuzey 3 adet, güney 3 adet olmak üzere toplam 6 adet araç dedektörü kullanılmıştır. Araç dedektörleri ile kulvarlar üzerindeki mevcut araç kuyruk uzunluğu hakkında bilgi alınmaktadır. Elde edilen kuyruk uzunluğu bilgisi bulanık kontrolörlerine giriş bilgisi olarak girilmektedir. Dört yön olduğu için dört adet giriş kuyruk uzunluğu bilgisi gelmektedir. Bu bilgiler bulanık kontrolörlerine girildiğinden trafik ışıklarının süreleri de bu kuyruk uzunluklarına göre belirlenmektedir. Kuyruk uzunlukları giriş bulanık küme işlemlerine tabi tutulduğunda her kuyruk uzunluğu değeri bu küme içerisinde bir değere karşılık gelmektedir. Kısa, orta ve uzun gibi bulanık kümelerinin içerisine girmektedir. [1,2,3,6,11,14]



Şekil 3.3 Araç üretici ve simülasyonu devre şeması

Şekil 3.3 de görülen devre dört yöne ait kulvarların hepsinde kullanılan ortak devre şemasıdır. Bu devreden doğu, batı, kuzey ve güney yönleri için dört adet kullanılmıştır.

3.1 Araç Üreteci Devresi

Devre üzerindeki soldaki kontrolör olan araç üretici pic16F877 kontrolörüdür ve araç üretici devre olarak kullanılmıştır. Bu devre vasıtasıyla dört çeşit araç tipi üretilmektedir. Bu araç tipleri ve özellikleri aşağıdaki gibidir.

1- Taksi : Boyu 5 metredir. Buna önden ve arkadan kuyruğa girdiği zamanki ara boşluk mesafeleri de dahildir.
2- Minibüs: boyu 7 metre oalrak düşünülmüştür. Buna önden ve arkadan kuyruğa girdiği zamanki ara boşluk mesafeleri de dahildir.
3- Otobüs : Boyu 9 metredir. Buna önden ve arkadan kuyruğa girdiği zamanki ara boşluk mesafeleri de dahildir.
4- Tır : Boyu 13 metredir. Buna önden ve arkadan kuyruğa girdiği zamanki ara boşluk mesafeleri de dahildir.

Araç üretici devre bu araçları programlandığı şekilde belli bir plan dahilinde üretmektedir. Eğer kulvarda araç kuyruğu uzunluğu kulvar uzunluğundan fazla değilse yani kulvar, araç alabilecek durumda ise bu durumda kulvar kontrolörü olan ikinci pic16F877 kulvarın boş olduğunu belirten bir sinyal veriyor ve üretilen aracı kulvara alıyor. İşlemler bu şekilde devam ediyor. Her seferinde programda sırada hangi araç var ise o araç kulvara gönderilmektedir. Bu işlem bir sıra ve program doğrultusunda olmaktadır.

Araç üretilme işlemi, trafik yoğunluğuna göre sabah, öğlen ve akşam trafiği olmak üzere yani farklı trafik yoğunlukları düşünülerek yapılmaktadır. Her yön için üretilen araç örneği (paterni) de farklıdır. Yani sabah trafiği için batı yönüne ait araç üretim örneği ile akşam trafiği için kuzey yönüne ait araç örnekleri birbirinden farklıdır.

Araç üretilme işlemi farklı olmasına rağmen bu işlem belirli bir peryot dahilinde sürekli olarak devam etmektedir. Araç üretilme işleminde üretilen araç modeli , bulanık mantık ile yapılan trafik kontrol işlemi ile normal trafik kontrol işleminde aynıdır. Böyle olunca her iki trafik kontrolünü aynı şartlarda karşılaştırma imkanı olmaktadır. Bunun anlamı, tır, taksi, otobüs, taksi … gibi üretilen araç sırası her iki kontrol mantığında da aynı sırada ve aynı şekilde olmaktadır. Böyle olunca da şartlar eşit olduğundan aynı işlem sırasına her iki kontrol sisteminin verdiği tepkiler elde edilerek aradaki fark daha net olarak ortaya konulabilmektedir.

Bu konuda yapılan diğer araştırma ve incelemeler incelendiğinde birçoğu gerçek trafik altında olmasına rağmen farklı araç modelleri geldiğinden deneyler aynı şartlar altında gerçekleşmemektedir.[1,2 ve 3] Günün aynı saatlerine denk gelse bile aynı saat ve aynı dakikada aynı araçlar geçmeyeceğinden araç modelleri tam olarak uyuşmayacaktır. Bununla beraber araçların kırmızı ışık yanarken yapmış oldukları duraklamalar da bütünüyle hassas bir şekilde ölçülemeyeceğinden bu konuda yapılan hesaplamalar da bir nevi teorik veya varsayımlara dayandığından ortaya konulan hesaplamar tam olarak gerçeği yansıtmayabilecektir. Araç

kulvarda ne kadar bekleme yaptı, birim zamanda kaç araç geçti ve oluşan araç kuyruğuna göre trafik ışıkları nasıl tepki verdi bütün bunlar bu projede donanımsal olarak gerçekleştirilip, daha detaylı olarak bilgiler elde edilmiştir.

Araç üreteci devre üzerinde bulunan SW1 ve SW2 anahtarları vasıtasıyla sabah, öğlen ve akşam trafiği için araç modelleri üretilmektedir. SW1 ve SW2 anahtarları dört yöne ait kontrol kartlarında da ortaktır. Hangi trafik modeli seçilirse bütün yönlerde o modele ait trafik yoğunluğuyla ilgili sabah, öğlen veya akşam için araçlar üretilmektedir.

3.2 Araç Simülatörü

Projede kullanılan ikinci kontrolör olan sağdaki pic16F877 ise araç simülasyonu kontrolörüdür. Bu kontrolörün görevi ise araç üretici pic tarafından üretilen araçları kulvara alıp zamana göre kulvardaki kuyruk durumu da göz önünde bulunularak bu araçları metre metre ilerletmektir. İlerleyen araç ışıklara geldiğinde eğer kavşakta yeşil ışık yanıyor ise kulvardaki araç karşı şeride geçirilmektedir. Yeşil ışık yanmıyor ise araç kuyruk oluşturmaya başlamaktadır. Ya da araç önünde kuyruk varsa araç gelip kuyruğa girmektedir. Kuyruktaki araçlar yeşil ışık yanmaya başladığında geçiş yapmaya başlamaktadırlar. Yeşil ışık yanarken karşı şerite geçiş yapan her araç, araç sayısını bir artırmaktadır.

Kulvara araç üretici devre tarafından üretilen araç girdiğinde eğer aracında araç kuyruğu yoksa araç saniyede 12 metre/saniye hızla ilerlemektedir. Bu hız yaklaşık şehriçi trafik ortalamasına yakındır. Hesaplama kolaylığı olması için tamsayı olarak alınmıştır. Küsuratlı rakamlar alınmamıştır. Kuyruktaki araçların hızı daha yavaş olduğundan ortalama kuyruk hızı da 8 metre/saniye olarak alınmıştır.

Herhangi bir kulvardaki araç kulvara girdiğinde eğer önünde hiç kuyruk yoksa 200 metrelik kulvarı tam olarak 16 saniyede terk etmektedir. 100 metrelik kulvarı ise 8 saniyede terk etmektedir. Bu sürelerin üzerinde kulvarı terk eden araçlar için bekleme ya da duraksama zamanı kavramı devreye girmekte ve fazladan kulvarda durulan süre araçlar için ya kırmızı ışık önünde bekleme yaptığı ya da kuyrukta

bekleme yaptığı sonucunu ortaya koymaktadır. Araç önünde kuyruk olduğu zaman ya da kırmızı ışık yanıyorsa araç kulvarı normal zamanında terk edemeyeceğinden kulvar üzerinde yeşil ışık yanana kadar ya da kuyruk bitene kadar bekleme yapacağından normal geçiş zamanından daha uzun bir süre bekleme yapacaktır. Örnek verilecek olursa normalde 200 metrelik kulvarı 16 saniyede geçen bir araç bekleme yaparak gecikmeli olarak kulvarı 70 saniyede terk ettiyse o zaman 70-16 = 54 saniye gecikerek kulvarı terk etmiş olur ya da 54 sn. fazladan kulvar üzerinde bekleme yapmış olur. Bekleme süresinin artması istenilmeyen bir durumdur. Araç sayısı fazla olduğunda bekleme süresinin önemi daha da fazla ortaya çıkacaktır. Bütün mesele bekleme süresini minimuma indirerek kulvardan birim zamanda daha fazla araç geçişini sağlamaktır. Trafik kontol mekanizmasının asıl yapması gereken görevin önemi burada ortaya çıkmaktadır. [1,2]

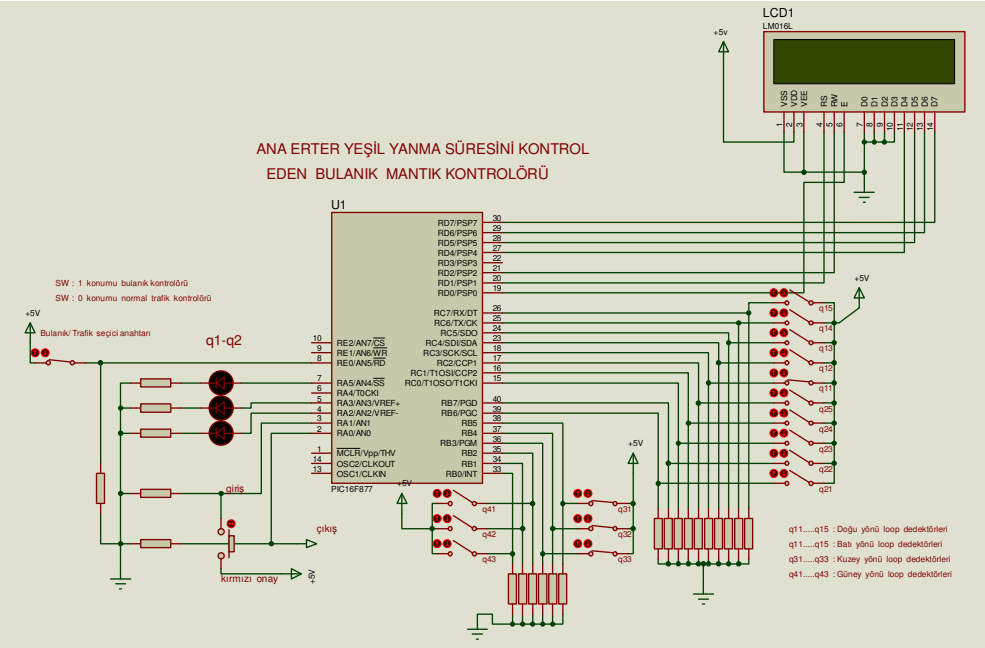
Araç simülatörü olan ikinci pic16F877 mikrodenetleyici, "karşı şerit onay" ve "karşı şerit dolu" pinleri aracılığıyla kulvardaki araç kuyruk durumuna göre sinyal üretmektedir. Karşı şerit boş olduğu zaman karşıda bulunan kulvardan "karşı şerit onay" sinyali verilmekte ve eğer araç varsa bu araç karşı şerite geçirilmektedir. Karşı şerit doluysa ya da uygun değilse "karşı şerit dolu" sinyali verilerek şerite araç kabul edilmeyeceği belirtilmektedir.

Kavşaktaki değişik yönlerdeki kulvarlardan birisi genelde asıl önemli araç trafik akışının gerçekleştiği ana arter tabir edilen birincil öneme sahip kulvardır. Trafik sinyalizasyonu o şekilde tasarlanmalıdır ki ana arterden geçen araç sayısı en fazla olsun ve kulvar üzerinde araç bekleme süreleri de en alt seviyeye indirilebilsin. Şekil 4.4 de ana artere yeşil yanma süresini kontrol eden bulanık kontrol devresinin şeması verilmiştir. Şekil 3.4 deki devrede görülen q11..q15 ve q21, q31, q41 şeklinde ifade edilen semboller sırasıyla doğu, batı, kuzey ve güney yönlerine ait loop dedektörlerini ifade etmektedir. 200 metrelik kulvar üzerinde sırasıyla 40, 85, 120,160 ve 190. metrelerde loop dedektörleri yerleştirilerek batı-doğu doğrultusundaki araç kuyrukları tespit edilmeye çalışılmıştır. [1,6,11,13]

Aynı şekilde 100 metrelik kulvar üzerinde de sırasıyla 30. , 50. ve 90. metrelere loop dedektörleri yerleştirilerek kuzey-güney doğrultusundaki araç kuyrukları öğrenilmeye çalışılmıştır. Ana arter yeşil yanma süresini kontrol eden kontrolör ile tali arterin yeşil yanma süresini(ana arter kırmızı yanma süresi) kontrol eden kontrol devresi yapı olarak birbiriyle hemen hemen aynıdır.

3.3 Trafik Işıkları Kontrol Devresi

Şekil 3.4 de ana arter yeşil yanma süresini kontrol eden devre şeması görülmektedir.

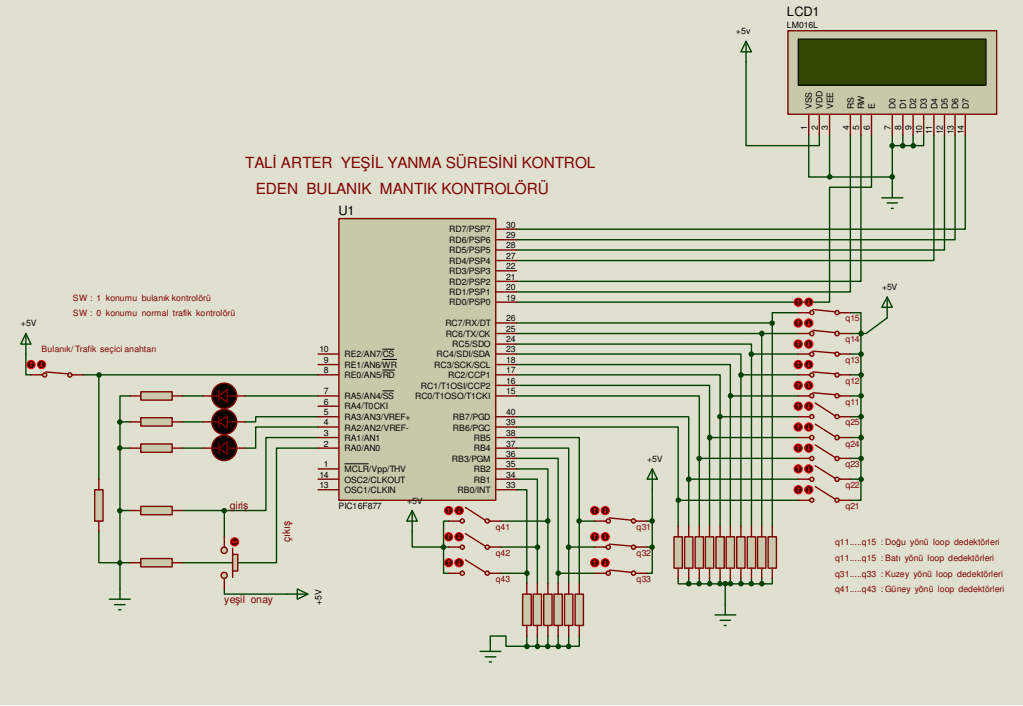


Şekil 3.4 Ana arter yeşil yanma süresini ayarlayan bulanık kontrolörü devre şeması

Ana ve tali arter yeşil yanma sürelerini kontrol eden kontrol devreleri bir bakıma şöyle de ifade edilebilir. Ana arterdeki yeşil yanma süresi Şekil 3.4 deki kontrolör şamasındaki devreyle, kırmızı yanma süresi ise şekil 3.5 deki şemada bulunan devreyle kontrol edilmektedir.

Şekil 3.4 ve Şekil 3.5 deki kontrol devreleri, normal trafik kontrol devresi (sabit peryotlu) olarak çalışırken yeşil ışık yanma süresi 60 saniyedir. Kırmızı ışık

yanma süresi ise 40 saniyedir. Bu süreler bugün büyük şehirlerdeki trafik sinyalizasyonunun yapıldığı kavşaklarda en çok uygulanan ortalama trafik sinyalizasyon süreleridir. Kontrol devreleri bulanık kontrolörü olarak çalıştığında ise yeşil yanma süresi 20 saniyeden başlamakta ve trafikteki kuyruk durumuna bakarak 5(beş) kademede 120 saniyeye kadar çıkabilmektedir. Her bir kademede kuyruk uzunluğuna göre 3 saniye ile 20 saniye arasında artım uygulanmaktadır. Fakat süre artırımı 120 saniyeyi geçmiyecek şekilde dizayn yapılmıştır.



Şekil 3.5 Bulanık kontrolörü devre şeması

Kontrolör devresi bulanık kontrolörü olarak çalıştığında kırmızı süresi de kuyruk durumuna göre 20 saniyeden 120 saniyeye kadar beş kademede süre uzatımı yapılabilmektedir. [1,2]

Bulanık kontrolörü olarak kullanıldığında devrenin en güzel veya olumlu sonucu eğer ilgili kulvarda kuyruk yoksa ışığın fazı değiştirilerek diğer kulvardaki araçlara da geçiş hakkı tanınarak bekleme sürelerini minimuma indirmesi ve geçiş yapan araç sayılarını artırmasıdır. Bu sayede bekleme zamanı da azalmakta ve araç geçişleri en uygun şekilde optimizasyon yapılmaktadır. Bütün yönlerde kontrol aynı anda

yapıldığından bekleme zamanı ve geçen araç sayısı bütün yönlerde aynı anda kontrol edilmekte ve ayarlanmaktadır.

4. UYGULAMA PROJESİNDE KULLANILAN YAZILIMLAR

Devrede kullanılan pic16F877 üzerinde kullanılan yazılım C ile yapılmıştır. Devre üzerinde 10 adet pic16F877 kullanılmıştır. Her bir pic16F877 üzerinde çalışmasını kontrol eden bir yazılım vardır.

4.1 Araç Üretici Devre

Araç üretici devreler her yön için bir adet olmak üzere toplam dört adettir. Araç üretici devrelerin çalışma prensibi birbirleriyle aynıdır. Sadece hangi yöne ait araç üretiyorlarsa, araçların üretilme aralığı ve cinsini belirleyip bir sonraki kat olan araç simülasyon katına üretilen araç tipi iletilir. Sonraki kat bu aracı sanki gerçek trafik altında kulvarda bir araç varmış gibi simülasyon yapar.

Araç üretici devre 30 adet çeşitli tipteki aracı belirli zaman aralıklarıyla peryodik olarak üreterek bir sonraki kata gönderir. Bu 30 aracın üretilme şekli aynıdır. Dört yön olduğu için ve her yöne ait sabah, öğle ve akşam trafiği olmak üzere 4 X 3=12 adet farklı trafik araç üretme modeli oluşturulmuştur.

Araç simülatörü üzerindeki iki adet anahtar ile SW1 ve SW2 sabah, öğle ve akşam trafik modundan birisi seçilmektedir. Şekil 4.1 de C dilinde araç üretici devrenin ilgili kısmındaki trafik seçimiyle ilgili ifade gösterilmiştir. Burada in1, sw1 ile bağlantılı , in2 de sw2 anahtarını ifade etmektedir.

```
" for(;;)
        {  //for başı
  if(!input(in1) && !input(in2)) sabah();// sabah programı
  else if(!input(in1) && input(in2)) oglen();
  else aksam(); "
```

Şekil 4.1 Trafik modlarını çağıran program

Şekil 4.1 deki C ifadesinde sabah() ile sabah trafiği, oglen() ile öğlen trafiği ve akşam() ile de akşam trafiğine ait alt programlar çağrılarak ilgili araç modelleri üretilmektedir. Şekil 4.2 deki C ifadesinde oglen trafiğine ait alt program görülmektedir.

```
"void oglen(void) {
 tir();
 delay_ms(4400);
 taksi();
 delay_ms(2000);
 minibus();
 delay_ms(2600);
 minibus();
 delay_ms(3600);
 taksi();
 delay_ms(2000); "
```

Şekil 4.2 Öğlen Trafik Modu Altprogramı

Şekil 4.2 deki C ifadesinde alt programda tir() ile 13 m. uzunluğundaki tır aracı üretilmekte ve simülatöre gönderilmektedir. Şekil 4.3 deki C ifaesinde de bununla ilgili kısım görülmektedir. Üretilen araçlar arasında C ifadeleriyle gecikme yaptırılarak trafik yoğunluğu ayarlanmaktadır. Deley_ms(3600) ile 3.6 sn lik gecikme sağlanamaktadır. Gecikmeler sabah, öğle ve akşam trafik yoğunlukları göz önünde bulundurularak seçilmiştir.

```
"void tir(void) {
while(input(dolu)) output_low(aracvar);
 do {
  output_high(cins1);
  output_high(cins2);
  output_high(cins3);
  output_high(aracvar);
  }while(input(onay)==0) ;
  output_low(aracvar);
  output_low(cins1);
  output_low(cins2);
  output_low(cins3);
}"
```

Şekil 4.3 Tir aracını üreten alt program

Şekil 4.3 de C dili ifadesinde cins1, cins2 ve cins3 terimleriyle araçların tipleri belirtilmektedir. Araç cinslerine göre rakamlar üretilmekte ve tip ifade edilmektedir. Şekil 4.4 deki C dili ifadesinde araç tiplerinin nasıl belirlendiği görülmektedir. Şekil 4.3 C dili ifadesinde ise cins1, cins2, ve cins3 pic16F877 üzerinde bir çıkış portuna karşılık gelen uçlarıdır. Bu uçlar araç üretici devre üzerindedir. Simülatör devresi bu uçları alarak Şekil 4.4 de C dili ifadesinde görüldüğü şekilde değerlendirerek araç tipini bulmakta ve araç tipine göre devre üzerinde yazılım ile belirlendiği şekilde işlem yapmakta, araç ilerletmekte, kuyruk oluşturmakta, geçen araç sayısını ve bekleme zamanını tespit etmektedir.

```
"unsigned cins_tespit(void)          //kulvara giren aracın cinsini ogreniyor
 {
 if(cins1 && cins2 && cins3)   return(1);
 if(cins1 && cins2 && !cins3)  return(2);
 if(cins1 && !cins2 && !cins3) return(3);
 if(!cins1 && !cins2 && cins3) return(4);   }"
```

Şekil 4.4 Araç Cinsini Tespit eden Altprogram

Şekil 4.4 de C dili ifadesinde tır aracı, simülatör tarafından alınana kadar aracvar biti high (1) konumunda kalmaktadır. Simülatör aracı alır almaz bir sonraki aracın üretilme işlemine geçilir.

4.2 Araç Simülatötörü

Araç simülatörü devresi ise bir önceki kat olan araç üretici devre tarafından gönderilen araç tipi bilgisini alarak kulvar üzerinde sanki araç varmış gibi simülasyon yaparak her saniye aracı belirli miktar kulvar üzerinde ilerletme yapar. Araç eğer kulvar üzerinde kuyruk yoksa yeşil ışık yanıyorken saniyede 12 metre ilerletme yapar. Araç kuyruk üzerinde trafik ışığına geldiğinde yeşil ışık yanıyorsa karşı kulvara geçerek araç sayısını bir artırır. Yeşil ışık yanmıyorsa ya da kulvar üzerinde kuyruk varsa bu durumda araç kuyruk oluşturmaya başlar ya da kuyruğun son kısmına gelerek kuyruğa eklenerek kuyruk mesafesini kendi uzunluğu kadar artırır. Kuyruk, kulvar uzunluğunu geçtiğinde de "kulvar dolu" mesajı yayınlayarak araç üretici devre tarafından üretilen araç simülatöre alınmaz .

Kulvar müsait olana kadar araç alınmaz. Şekil 4.5 deki C ifadesinde , araç simülatörü üzerindeki C yazımlının main() kısmında yer alan ifadeleri göstermektedir. C ifadesinin ilk kısmında araç üretici devre tarafından araç gönderilip gönderilmediği kontrol edilmekte ve araç varsa bu aracın cinsi ve tipi tespit edilerek "arac_ilerlet()" alt programı ile ilgili tipe ait alt programı çağrılarak bu araç üzerinde gerekli simülasyon yapılmaktadır. Bu simülasyon işlemi kulvarda araç olduğu müddetçe her saniye sürekli olarak devam etmektedir.

```
"for(;;)
          {  //for başı
    if(input(aracvar)) {
    araccins=cins_tespit();
    if(((araccins==1) && (q1<=187)) || ((araccins==2) && (q1<=191)) || ((araccins==3) && (q1<=193)) ||
((araccins==4) && (q1<=195))) {
     if((i<=31) && (arac[i].var==0))  {
     output_low(dolu);
     arac[i].cins=araccins;
     arac[i].var=1;
     arac[i].har=1;
     arac[i].zaman=0;
     arac[i].mesafe=0;
     i=i+1;
     if(i==32) i=0;
     output_high(onay);//50 ms sonra onay verecek
     delay_ms(50);
     output_low(onay);
           }    }     }
  arac_ilerlet();
  delay_ms(477);
   SN2+=1;
  if(SN2==2) {
   SN2=0;
  SN1=SN1+1;
   for(k=0;k<=31;k++) {
       if(arac[k].var==1) {
       arac[k].zaman=arac[k].zaman+1;}
    }   }
  if(q1>=40) output_high(LOOPBIR);
   else output_low(LOOPBIR);
  if(q1>=85) output_high(LOOPIKI);
  else output_low(LOOPIKI);
  if(q1>=120) output_high(LOOPUC);
  else output_low(LOOPUC);
  if(q1>=160) output_high(LOOPDORT);
  else output_low(LOOPDORT);
  if(q1>=190) output_high(LOOPBES);
  else output_low(LOOPBES);
  if(q1>=200) output_high(dolu);//q1  doluluk kontrolü
  else output_low(dolu);
  delay_ms(10);
  printf(lcd_putc,"\f%U:%U:%U Q1:%U\ns:%lU tg:%lU ",SA,DK,SN,q1,sayi,top);
}// for döngü sonu"
```

Şekil 4.5 Araç Simülatörü Programı

Şekil 4.5 deki C ifadesinde görülen arac[i].cins, arac[i].har, arac[i].var, arac[i].zaman ve arac[i].mesafe ifadeleri kulvarda bulunan her bir araca ait struct (yapı) ifadeleridir ve Şekil 4.6 daki C ifadesinde tanımlanma şekilleri görülmektedir.

```
"struct yapi
  {
    unsigned mesafe:8;
    unsigned  zaman;
    unsigned var:1;
    byte har:1;
    unsigned cins:3;
  };"
struct yapi arac[32];"
```

Şekil 4.6 Struct Yapının Tanımlanması

Şekil 4.6 ifadesinde 32 araçlık bir struct yapı tanımlanmaktadır. Bu ifadede "mesafe" ile belirtilen bilgi 8 bitlik bilgidir ve aracın kulvar üzerinde aldığı toplam mesafeyi belirtmektedir. "zaman" ile belirtilen, aracın kulvar üzerinde trafik ışıklarına kadar olan mesafe içerisinde kulvar üzerinde geçirdiği toplam zamanı ifade etmektedir. "var" ile de aracın kulvar üzerinde var olup olmadığını ve mevcut ise o araç üzerinde işlem yapılasını sağlamaktadır. "cins" ile de aracın tır, otobüs, minibüs veya taksi araç sınıflarından birisi olduğunu tespit etmektedir. "har" ile aracın hareketli olup olmadığı tespit edilemektedir. Araç karşı kulvara geçtiği andan itibaren Şekil 4.6 daki struct yapıdaki ifadeler temizlenerek araç üretici devreden gelecek bir sonraki araç için pic16F877 üzerinde gerekli RAM hafıza alanı boşaltılır.

```
"arac_ilerlet(void)                    //araç ilerletme kısımı
      {
        for(k=0;k<=31;k++) {
      if(arac[k].var==1) {
          switch(arac[k].cins)
      {
      case 1: tir();    break;
      case 2: otobus(); break;
      case 3: minibus(); break;
      case 4: taksi(); break;
      default: break;
      }
      }
        }
      // yapı sonu
      } "
```

Şekil 4.7 Araç İlerletme Altprogramı

Şekil 4.7 deki C ifadesinde tespit edilen araç cinsine göre 1,2,3 veya 4 ilgili araç için alt programı çağrılarak o araçla ilgili hertürlü işlem yapılmaktadır. Tır, otobus, minibüs ve taksi olmak üzere dört araç modeli vardır ve dört ayrı alt program çağrılarak bu araç modelleri üzerinde gerekli işlemler yapılmaktadır. Şekil 4.8 de C ifadesi ile tespit edilen araç tipine göre üretilen araç için çağrılan tir() alt programı görülmektedir.

Şekil 4.8 deki C ifadesinde programa göre yeşil ışık yanıyorsa araç karşı kulvara geçerek araç sayısı bir artırılmaktadır. Bu işlem "karsiya_aracgirdi()" alt programı ile yapılmaktadır. Yeşil ışık yanmıyorsa Araç kuyruk oluşturmaktadır. Kuyruk uzunluğu 200 m. yi geçerse arac[k].har==0 yapılarak araç kuyrukta olduğu belirtilmektedir. Kuyruktaki bir arac yeşil ışık yanarken 8 m/sn hızla hareket etmektedir. Kuyruk oluştuğunda ise tır aracı 13 m. uzunluğunda kuyruğun sonuna eklenmektedir. 4.9 da C dili ifadesinde araç karşı kulvara girdiği zaman çağrılan "karsiya_aracgirdi(void)" alt programı görülmektedir.

```
"tir(void)
      {
       //önce kuyrukla ilgili kısmı yapıyoruz
       if(arac[k].har==0) {
       if(input(yesil) && !input(karsiserit_dolu)) {
       arac[k].mesafe=arac[k].mesafe+4;
       if(arac[k].mesafe>=200) karsiya_aracgirdi();
        }
         }
       //ilerle buradan itibaren başlıyor
       if((arac[k].har==1) && (q1>0)) {
       if(input(yesil) && !input(karsiserit_dolu))  {
        arac[k].mesafe=arac[k].mesafe+6;
        if(arac[k].mesafe>=(200-q1)) {
                    arac[k].mesafe=(200-q1);
                    arac[k].har=0;
                    q1=q1+13;
                              }
                                }
                                 }
          if((arac[k].har==1) && (q1==0)) {
          if(input(yesil) && !input(karsiserit_dolu))  {
          arac[k].mesafe=arac[k].mesafe+6;
          if(arac[k].mesafe>=200) karsiya_aracgirdi();
                              }
                                }
                           //ilerle sonu
      //2. if
      if(!input(yesil) || input(karsiserit_dolu)) {
      if((arac[k].har==1) && (q1>=0)) {
             arac[k].mesafe=arac[k].mesafe+6;
             if(arac[k].mesafe>=(200-q1)) {
             arac[k].mesafe=(200-q1);
             q1=q1+13;
             if(q1>200) q1=200;
             arac[k].har=0;
                      }
                        }
                           }//2. if sonu
      }//tir sonu"
```

Şekil 4.8 tir() alt programı

```
"karsiya_aracgirdi(void)
      {
       output_high(karsiserit_aracvar);
       if(arac[k].cins==1){
       output_high(karsiserit_cins1);
       output_high(karsiserit_cins2);
       output_high(karsiserit_cins3);}

        if(arac[k].cins==2){
       output_high(karsiserit_cins1);
       output_high(karsiserit_cins2);
       output_low(karsiserit_cins3);}

        if(arac[k].cins==3){
       output_high(karsiserit_cins1);
       output_low(karsiserit_cins2);
       output_low(karsiserit_cins3);}

       if(arac[k].cins==4){
       output_low(karsiserit_cins1);
       output_low(karsiserit_cins2);
       output_high(karsiserit_cins3);}
       delay_ms(150);
       output_low(karsiserit_cins1);
       output_low(karsiserit_cins2);
       output_low(karsiserit_cins3);
       output_low(karsiserit_aracvar);
       sayi=sayi+1;
       if(arac[k].zaman<16) arac[k].zaman=16;
       tg=arac[k].zaman-16;
       if(tg<0) tg=0;
       if(input(yesil) && (tg==1) && (q1==0)) tg=0;
       tor=(tor+tg)/sayi;
       top=top+tg;
       if((q1>0) && arac[k].cins==1) q1=q1-13;
       else if((q1>0) && arac[k].cins==2) q1=q1-9;
       else if((q1>0) && arac[k].cins==3) q1=q1-7;
       else if((q1>0) && arac[k].cins==4 ) q1=q1-5;
       else  ;
       if(q1<0) q1=0;
        arac[k].var=0;
        arac[k].har=0;
        arac[k].zaman=0;
        arac[k].mesafe=0;
        arac[k].cins=0;
            }//program sonu
   // 2. yapının karşıya araç girdisi "
```

Şekil 4.9 Karsiya_aracgirdi() Alt Programı

Şekil 4.9 daki C programı ile araç karşı şeride girdiği zaman araç kuyruğu azaltılarak geçen araç tipine göre kuyruk uzunluğu azaltılmaktadır. Aracın geçiş zamanı ve bekleme süresi hesap edilmektedir. Karşı şeride geçen araçla ilgili "karsiserit_cins1", "karsiserit_cins2" ve "karsiserit_cins3" bitleri 1 veya 0 yapılarak geçen aracın cinsi belirtilmektedir. Araç karşıya geçtiği andan itibaren de o araca ait RAM üzerindeki struct yapıya ait ilgili bitler temizlenerek yeni gelecek araçlar için yer açılmaktadır.

Eğer araç kulvar üzerinde bekleme yaparsa ya da kuyrukta durmak zorunda kalırsa normal geçme zamanından fazla zaman kulvar üzerinde duracağından gecikmeli olarak trafik ışıklarını geçmiş olacaktır. Bu durumda 200 m uzunluğundaki kulvarı 16 sn de geçmesi gerekirken 70 sn de geçtiği zaman 70-16=54 sn bekleme yapmış olmaktadır. Her bir araç için karşı kulvara geçtiği zaman, bekleme zamanı hesap edilerek ve toplam bekleme zamanına eklenerek 1 saat içerisinde geçiş yapan araçlara ait toplam bekleme zamanı bulunmaktadır.

Araç simülasyonu devresinden de 4 adet olduğundan her yöne(kuzey, güney, doğu ve batı) ait toplam araç sayısı ile 1 saat içerindeki yapılan toplam bekleme süresi tespit edilmektedir. Bu sayede hangi yönden kaç adet araç geçtiği ve ilgili yöne ait bekleme süreleri öğrenilmekte ve kulvarın geçiş bilgileri elde edilmektedir.

Araç simülasyon devresi yeşil ışık bilgisine göre hareket etmektedir ve sadece yeşil ışık yanarken araç geçirmektedir. Yeşil ışık bilgisini ise bulanık kontrolörlerinden almaktadır. Batı-doğu simülatör kartları B-D bulanık kontrolöründen yeşil ışık bilgisini almaktadır. Kuzey-güney simülatörleri ise kuzey-güney bulanık kontrolöründen yeşil bilgisini almaktadır.

Bulanık kontrolörleri üzerinde bulunan “fuzzy” seçici anahtarı, bulanık kontrolörlerinin normal kontrolör olarak mı yoksa bulanık kontrolörü olarak mı çalışacağını belirlemektedir. “fuzzy” anahtarı kapalı iken “1” konumunda bulanık kontrolörleri olarak çalışmakta; “fuzzy” anahtarı açık iken “0” konumunda ise normal trafik kontrolörü olarak görev yapmaktadır. Yeşil ışık bilgisi bulanık kontrolörlerinden geldiğinden eğer çalışma şekli normal trafik sinyalizasyonu seçilmişse yeşil ışık süresi sabit olarak 60 sn. olmaktadır. Kırmızı ışık ise 40 sn. olarak seçilmiştir. Bulanık kontrolörü olarak seçildiği zaman da yeşil ışık 20 sn. ile 120 sn. arasında değişmektedir. Kırmızı ışık ta 20 sn. ile 120 sn. arasında değişmektedir. Bulanık kontrolü modunda çalışıldığında giriş değişkenleri dört

yöne ait kuyruk uzunlukları olduğu için yeşil ve kırmızı ışık süresi de kuyruk uzunluklarına göre değişecektir.

Araçlar kuyrukta iken 8 m/sn hızla ilerlemektedir. Bu hız kulvar üzerinde farklı araçlar olduğu için bunların dur, kalk ve hızlanma aralıkları da farklı olduğu için ortalama bir değer alınmıştır ve hesaplama kolaylığı olması için de tam sayı değişken olması tercih edilmiştir.

4.3 Trafik Işıkları Kontrol Devresi

Bulanık kontrolör devreleri batı-doğu(B-D) yeşil zamanı bulanık kontrolörü devresi ve kuzey-güney (K-G) yeşil zamanı bulanık kontrolörü olmak üzere iki adet devreden oluşmaktadır. Heriki devrede de çalışma mantığı benzerdir. Sadece bulanık kural tabanı farklıdır. Batı-doğu devresi yeşil süresinin sonuna geldiğinde diğer kontrolöre bir kontrol sinyali göndererek kuzey-güney yeşil fazının başlamasını sağlar. K-G yeşil fazı bittiğinde aynı şekilde sinyal vererek B-D bulanık kontrolörü yeşil fazını tekrar başlatır. Bu şekil çalışma sürekli tekrar ederek devam eder. Ara geçişteki turuncu fazı 3 sn. olarak kabul edilmiştir. Şekil 4.10 daki C ifadesinde B-D bulanık kontrolörünün main() kısmına ait C dili ifadeleri görülmektedir.

```
"main(void)
{
  set_rtcc(0);
  SET_TRIS_B(0XFF);
        SET_TRIS_A(0X02);
        SET_TRIS_C(0XFF);
        SET_TRIS_D(0X00);
  SET_TRIS_E(0X0D);
        delay_ms(20);
        lcd_init();
  setup_counters(RTCC_INTERNAL,RTCC_DIV_128);
  enable_interrupts(INT_RTCC);
  enable_interrupts(GLOBAL);
        for(;;)
        {  //for başı
  output_low(yonay);
  kuyruk_test();
  if(input(fuzzy)) fuzzy_bak();
  else saykil();
}// for döngü sonu
}//main sonu "
```

Şekil 4.10 B-D Bulanık Kontrolörü Main Kısmı

Şekil 4.10 daki C ifadesinde kuyruk_test() alt programı ile K-G yönlerindeki en uzun kuyruk ile B-D yönlerindeki en uzun kuyruk tespit edilmektedir. Bu tespit işlemi, loop dedektörlerinin belirlenen süre kadar meşgul olup olmamalarına göre yapılmaktadır. Tespit edilen iki adet kuyruk uzunluğu B-D ve K-G yönlerine ait en uzun kuyruk uzunluklarıdır ve bulanık kontrolörlerinin giriş değişkenleri olarak kullanılacaklardır.

Bulanık kontrolörleri iki modda çalışabilmektedir. Modlardan biri bulanık kontrolörü olarak çalıştırmaktır. Diğer modda ise normal trafik kontrolörü gibi çalışmaktadır. Bu sayede bulanık kontrolörü ile normal trafik kontrolörü aynı kart üzerinde yapılarak iki çalışma şekline ait sonuçların alınması sağlanmıştır. Araç modeli her iki kontrolör tipi için sabah, öğlen ve akşam saatlerinde aynı olduğu için her iki kontrolörün davranışı ve yeşil yanma süreleri kolayca aynı kart üzerinden ayarlanabilmektedir. Main() programında bulanık kontrolörleri "input(fuzzy)" ifadesi ile belirtilen "fuzzy" anahtarı bu ayrım işlevini yerine getirmektedir. Anahtar (fuzzy) 1(on) konumunda iken her iki kontrolör bulanık kontrolörü olarak görev yapmaktadır. Bulanık kontrolör olarak çalışırken yeşil yanma süreleri kuyruk uzunluklarına göre belirlenmektedir. Yeşil yanma sürelerinin uzanma fazı 5 adımda yapılmaktadır. Eğer kuyruk yoksa ve uzama değeri 0 değerini alıyorsa ilgili faz sona erdirilerek diğer ışık fazına geçilmektedir.

"Fuzzy_bak()" ile ifade edilen alt program bulanık kontrol işleminin yapıldığı alt programdır. Bu program 9 adet kural tabanına göre kuyruk değerlerini işlemektedir. Amaç, kuyruk uzunluğuna göre ilgili faz için uzama zamanını tespit etmektir. Her fazın son 3 sn. süresinde uzama zamanı tespit edilerek bulanık kontrol işlemi devam etmektedir. Eğer uzama zamanı 0(sıfır) değerini alıyorsa mevcut faz bitirilerek diğer kulvar için yeşil ışık yakılır. 5 adımda gerçekleştirilen uzama zamanı bir faz için toplamda 120 sn. ile sınırlandırılarak diğer yönlere ait bekleyen araçların geçişine imkan verilmiştir.

Anahtar(fuzzy), 0(off) konumuna alındığında da her iki kontrolör normal trafik kontrolörü olarak görev yapmaktadır. Yani yeşil yanma süresi sabit ve 60 sn. ve kırmızı yanma süresi de sabit 40 sn. olarak alınmakta ve turuncu ışık 3 sn. kabul

edilmekte ve peryodik olarak bu işlem devam etmektedir. “Saykil()” C dili ifadesi ile normal trafik kontrol işlemini gerçekleştiren alt program çağrılmaktadır. Bu program ile bugün dünyada en yaygın şekilde kullanılan sabit peryotlu trafik ışığı kontrol işlemi yapılmaktadır.

Şekil 4.11 deki C ifadesinde bu işlemin nasıl yapıldığına ilişkin “kuyruk_test(void)” alt programının C dili ifadeleri görülmektedir.

```
“void kuyruk_test(void)
{
 if(!input(q12)) tq12=0;
 if(!input(q13)) tq13=0;
 if(!input(q14)) tq14=0;
 if(!input(q15)) tq15=0;
 if(input(q11) && !input(q12) && !input(q13) && !input(q14) && !input(q15)) q1=40;
 if(!input(q11) && !input(q12) && !input(q13) && !input(q14) && !input(q15)) q1=0;
 if((input(q12)) && (tq12>=2) && !input(q13) && !input(q14) && !input(q15) && input(q11)) q1=85;
 if((input(q13)) && (tq13>=2) && !input(q14) && !input(q15) && input(q12) && input(q11)) q1=120;
 if((input(q14)) && (tq14>=2) && input(q13)  && !input(q15) && input(q12) && input(q11) ) q1=160;
 if((input(q15)) && (tq15>=2) && input(q13)  && input(q14) && input(q12) && input(q11)) q1=190;
 if(!input(q22)) tq22=0;
 if(!input(q23)) tq23=0;
 if(!input(q24)) tq24=0;
 if(!input(q25)) tq25=0;
 if(input(q21) && !input(q22) && !input(q23) && !input(q24) && !input(q25)) q2=40;
 if(!input(q21) && !input(q22) && !input(q23) && !input(q24) && !input(q25)) q2=0;
 if((input(q22)) && (tq22>=2) && !input(q23) && !input(q24) && !input(q25) && input(q22)) q2=85;
 if((input(q23)) && (tq23>=2) && !input(q24) && !input(q25) && input(q22) && input(q22)) q2=120;
 if((input(q24)) && (tq24>=2) && input(q23)  && !input(q25) && input(q22) && input(q22) ) q2=160;
 if((input(q25)) && (tq25>=2) && input(q23)  && input(q24) && input(q22) && input(q22)) q2=190;
 if(!input(q32)) tq32=0;
 if(!input(q33)) tq33=0;
 if(!input(q31)) tq31=0;
 if(input(q31) && (tq31>=2) && !input(q32) && !input(q33)) q3=30;
 if(!input(q31) && !input(q32) && !input(q33) ) q3=0;
 if((input(q32)) && (tq32>=2) && !input(q33)  && input(q31)) q3=50;
 if((input(q33)) && (tq33>=2)  && input(q32) && input(q31)) q3=90;
 if(!input(q42)) tq42=0;
 if(!input(q43)) tq43=0;
 if(!input(q41)) tq41=0;
 if(input(q41) && !input(q42) && !input(q43)) q4=30;
 if(!input(q41) && !input(q42) && !input(q43)) q4=0;
 if((input(q42)) && (tq42>=2) && !input(q43)  && input(q41)) q4=50;
 if((input(q43)) && (tq43>=2)  && input(q42) && input(q41)) q4=90;
 if(q1>q2) maxq12=q1;
 else maxq12=q2;
 if(q3>q4) maxq34=q3;
 else maxq34=q4;
 } “
```

Şekil 4.11 Kuyruk Uzunluğunu Tespit Eden Alt Program

Şekil 4.11 deki C ifadesinde q1 ve q2 ile Batı ve Doğu yönlerine ait kuyruklar ifade edilmektedir. q3 ve q4 ile de Kuzey ve Güney yönlerine ait kuyruk

uzunlukları ifade edilmektedir. q1 ve q2 nin en uzun olanı ile q3 ve q4 kuyruklarının en uzunu seçilmektedir. Buradaki maksat yönlere ait doğrultudaki en uzun kuyruğu tespit ederek o yöndeki yeşil ışığın yanma süresini kuyruğa göre ayarlamaktır. Şekil 4.11 deki C ifadesinden görüldüğü gibi B-D yönleri 200 m. uzunluğunda olduğu için 40, 85, 120, 160 ve 190. metrelerde loop konulmuştur. K-G yönleri ise 100 m. uzunluğunda olduğundan 30, 50 ve 90. metrelere loop konulmuştur. İnterrupt (kesme) kullanılarak her saniye kuyrukların durumu öğrenilmekte ve zamanlama sayaç değerleri kontrol edilmekte ve üzerlerinde gerekli işlemler yapılarak değerleri düzenlenmektedir. Bütün bu değerler her saniye LCD üzerinde de gösterilerek sayısal değerlerin anlık olarak LCD üzerinden görsel olarak izlenmesine olanak verilmektedir. Şekil 4.12 de ise C dili ifadeleri görülmektedir.

```
"void zaman_test(void) {  //kuyruk zamanlama ile ilgili kısım
tq12=tq12+1;
tq13=tq13+1;
tq14=tq14+1;
tq15=tq15+1;
tq22=tq22+1;
tq23=tq23+1;
tq24=tq24+1;
tq25=tq25+1;
tq31=tq31+1;
tq32=tq32+1;
tq33=tq33+1;
tq41=tq41+1;
tq42=tq42+1;
tq43=tq43+1;
ty1=ty1-1;
tk=tk-1;
if(!input(ykabul)) ty3=ty3+1;
else ty3=0;
if(input(fuzzy))  printf(lcd_putc,"\fTopuz:%U Uz:%U\nQBD:%U QKG:%U ",ty2,uzatma,maxq12,maxq34);
        else printf(lcd_putc,"\fSure:%U \nQBD:%U QKG:%U ",ty3,maxq12,maxq34);
}
  void saat1(void)
{
 TM1=TM1+1;
        if(TM1>=30) {//prescalar küçült
                TM2=TM2+1;
                if(TM2==2) {//saniye artır
                TM1=0;
                TM2=0;
          SN=SN+1;
    zaman_test();
    kuyruk_test();
   if(SN==60)   {//dakika artır
   SN=0;
                DK=DK+1;
                if(DK==60)      {//saat artır
    DK=0;
                SA=SA+1;
                if(SA==24)      SA=0;
   }   }
   }  } }      // interrupt sonu"
```

Şekil 4.12 Zamanlamalarla ilgili Alt Program

Şeki 4.12 deki C ifadelerinde interrupt yazılımı ile aynı zamanda SN, DK ve SA şekilnde zamanlama değerleri de elde edilmektedir. Bu değerler LCD ekranında gösterilerek yapılan kontrol işlemleriyle ilgili detaylı görsel veriler elde edilmektedir.

Şekil 4.13 de LCD üzerinde, yapılan kontrol ile ilgili anlık veriler görülmektedir. LCD üzerindeki görülen değerler

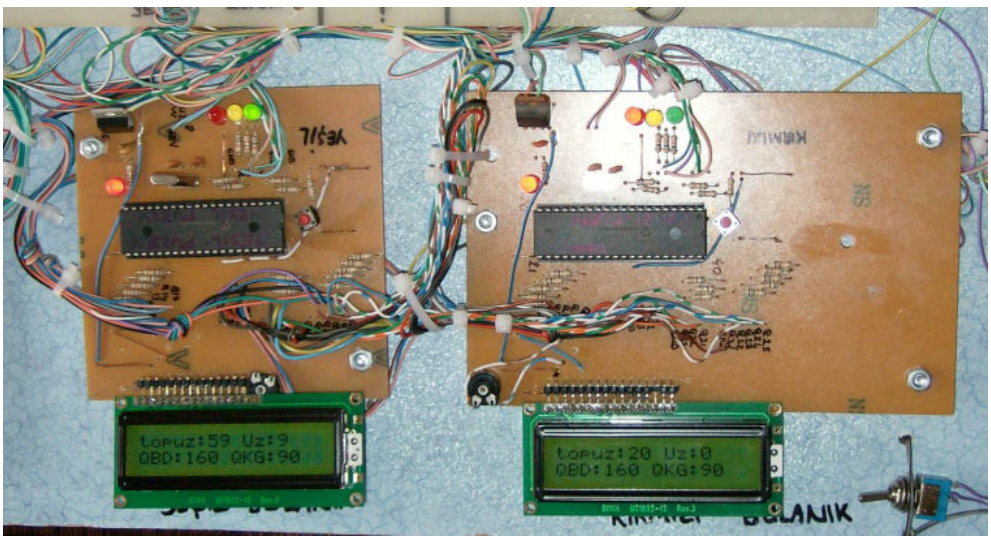
topuz: 0-120 sn arası yapılan toplam ışık fazı uzama değerini ifade etmekte.

Uz: 5 adet ışık fazına ait kuyruk durumuna göre olan 0-20 sn arasındaki fazlara ait uzama değerini göstermektedir.

QBD: B-D yönlerindeki en uzun kuyruğu göstermektedir.

QKG: K-G yönlerindeki en uzun kuyruğu göstermektedir.

Şekil 4.14 deki C dili ifadelerinde bulanık kontrolörlerinde kullanılan üyelik fonksiyonları olan giriş üçgen üyelik fonksiyonlarına ait C dili ifadeleri görülmektedir. Bulanıklaştırma ve keskin değeri elde etmek için çıkarım işlemleri bu fonksiyonlar kullanılarak kural tabanına göre yapılmaktadır.



Şekil 4.13 Bulanık kontrolörleri

```
"#separate
float yq1 q2k(float x) {
 y=(60-x)/60;
 return(y);
 }
 #separate
 float yq1 q2o1(float x) {
 y=(x-50)/50;
 return(y);
 }
 #separate
 float yq1 q2o2(float x) {
 y=(150-x)/50;
 return(y);
 }
 #separate
 float yq1 q2b(float x) {
 y=(x-140)/60;
 return(y);
 }
 #separate
 float maxq3q4k(float x) {
 y=(30-x)/30;
 return(y);
 }
 #separate
 float maxq3q4o1(float x) {
 y=(x-25)/25;
 return(y);
 }
 #separate
 float maxq3q4o2(float x) {
 y=(75-x)/25;
 return(y);
 }
 #separate
 float maxq3q4b(float x) {
 y=(x-70)/30;
 return(y);
 } "
```

Şekil 4.14 Üyelik Fonksiyonlarının Matematiksel İfadeleri

Çıkış üyelik fonsiyonları singleton üyelik fonksiyonlarından seçilerek, hesaplama kolaylığı sağlanarak pic16F877 ile daha hızlı olarak durulandırma yapılmıştır. B-D ve K-G yönlerine ait kontrolörlerin çalışma şekilleri benzer olduğundan ve yazılarak karışıklığa sebep vermemek için bütün C ifadeleri burada gösterilmemiştir. Şekil 4.14 de fonksiyonlarla ilgili yazılım ve C dili ifadeleri ile görülmektedir.

Tablo 4.1 ve Tablo 4.2 de kuzey-güney ve batı-doğu yönlerine ait üyelik fonksiyonları ve aldıkları değerler görülmektedir.

Tablo 4.1 Doğu-Batı yönlerine ait üyelik fonksiyonları ve aldıkları değerler

MESAFE	KISA	ORTA	UZUN
0	1	0	0
10	0,83333	0	0
20	0,66667	0	0
30	0,5	0	0
40	0,33333	0	0
50	0,16667	0	0
60	0	0,2	0
70	0	0,4	0
80	0	0,6	0
90	0	0,8	0
100	0	1	0
110	0	0,8	0
120	0	0,6	0
130	0	0,4	0
140	0	0,2	0
150	0	0	0,16667
160	0	0	0,33333
170	0	0	0,5
180	0	0	0,66667
190	0	0	0,83333
200	0	0	1

Tablo 4.2 Kuzey-Güney yönlerine ait üyelik fonksiyonları ve aldıkları değerler

MESAFE	KISA	ORTA	UZUN
0	1	0	0
5	0,83333	0	0
10	0,66667	0	0
15	0,5	0	0
20	0,33333	0	0
25	0,16667	0	0
30	0	0,2	0
35	0	0,4	0
40	0	0,6	0
45	0	0,8	0
50	0	1	0
55	0	0,8	0
60	0	0,6	0
65	0	0,4	0
70	0	0,2	0
75	0	0	0,16667
80	0	0	0,33333
85	0	0	0,5
90	0	0	0,66667
95	0	0	0,83333
100	0	0	1

4.4 Bulanık Kontrol Kurallarının Tasarlanması

Giriş ve çıkış için tarif edilen fuzzy değerleri kullanılarak Tablo 4.1 ve Tablo 4.2 de tarif edilen değişkenler kullanılarak Tablo 4.3 de yeşil uzama zamanı (ty) için verilen 9 adet fuzzy kontrol kuralları oluşturulmuştur. Tablo 5.4 de kırmızı uzama zamanı (tk) için bulanık değerleri gösterilmiştir. [1,2,3,11]

Tablo 4.4 de kırmızı ışık fazı için verilen 9 adet bulanık kontrol kuralları oluşturulmaktadır. Tablo 4.4 de kırmızı fazı uzama zamanı (tk) için bulanık değerleri gösterilmiştir. [7]

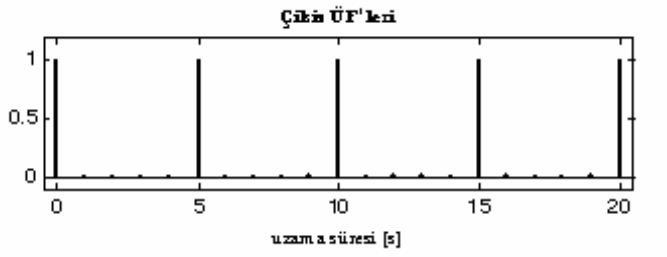
Tablo 4.3 Batı-Doğu yeşil fazı için kural tabanı

Yeşil Işık Kuralları		Kuzey-Güney Kuyruğu		
		kısa	orta	uzun
Batı-Doğu Kuyruğu	KISA	10	5	0
	ORTA	15	10	15
	UZUN	20	20	20

Tablo 4.4 Kuzey-Güney yönü yeşil fazı için kural tabanı

Kırmızı Işık Kuralları		Kuzey-Güney Kuyruğu		
		kısa	orta	uzun
Batı-Doğu Kuyruğu	KISA	0	15	20
	ORTA	0	10	15
	UZUN	0	0	5

Şekil 4.2 de çıkış uzama zamanı bulanık değerler kümesini temsil eden singleton fonksiyonları görülmektedir. [7]



Şekil 4.15 Çıkış üyelik fonksiyonları

Şekil 4.2 den de görüleceği gibi en uzun uzama zamanı 20 saniyedir. En kısa uzama değeri de 0 numaralı üyelik fonksiyonu olan 0 noktasında bulunan üyelik

fonksiyonudur. Sırasıyla 5, 10, 15 ve 20 saniye uzama değerlerine sahip singleton üyelik fonksiyonları görülmektedir. Uygulama projesinde kuyruk uzunlukları dikkate alınırken doğu-batı yönlerine ve kuzey-güney yönlerine ait kuyruk uzunlukları içerisinde en uzun olan kuyruk uzunluğu dikkate alınmaktadır. Bulanık mantık kontrolörü kural tabanı tasarlanırken herhangi bir yöndeki kuyruk uzunlukları içerisinde en uzun olan kuyruk uzunluğu dikkate alınmaktadır.

Her hangi bir zamanda trafik ışıklarının uzaması ayarlanırken yeşil ışık uzama zamanı da kırmızı ışık uzaman zamanı da 20 saniyeden başlamakta ve kuyruk uzunluğuna göre kural tabanı da dikkate alınarak artırılmaktadır. Artım miktarı 0 saniye ile 20 saniye arasında değişmektedir. Uzaman zamanı artım miktarı 5 kere ile sabit tutularak artım miktarı ile ışık zamanlarının maksimum değeri de sınırlandırılmıştır. Bu durumda yeşil ve kırmızı fazın en fazla uzama zamanı Tymax= 20+5*20=120 saniye olmaktadır.

Batı-Doğu yönlerinin minimum yeşil renk için uzama zamanı ise
Tymin=20+5*0=20 saniye olmaktadır. Maksimum uzaman zamanı ise en fazla artım miktarı olan 20 saniyenin 5 kere arka arkaya tekrarlanmasından gelmektedir.

Kırmızı fazı için maksimum uzama zamanı

Tkmax=20+5*20=120 saniyedir.

Minimum uzama zamanı ise

Tkmin=20+5*0=20 saniyedir.

Kırmızı artımın minimum değerinin 20 saniyede tutulmasının asıl sebebi eğer kuzey-güney arterinde araç kuyruğu yoksa kuzey-güney arterindeki ışık rengini kırmızıya çevirerek ana arterden araç geçişini sağlamak içindir. Dolayısıyla kuzey-güney kulvarında araç yok iken kuzey-güney yeşil ışık fazı

sonlandırılarak kırmızı ışık fazına geçilir. Böylece ana arterden daha fazla araç geçişi sağlanabilmektedir. Bulanık mantık denetiminde amaç, ana arterden maksimum araç geçirilirken ortalama bekleme ve duraklama sürelerini de minimuma çekerek yakıt ve zaman tasarrufu sağlarken birçok faydalı optimizasyonu da beraberinde gerçekleştirmektir.

Tablo 4.5 Bulanık mantıkla yapılan sabah trafiği sayım sonuçları

BULANIK MANTIK DENETİMİYLE YAPILAN SABAH TRAFİK SAYIMI	YÖN	ARAÇ SAYISI	GECİKME SÜRESİ(SN)	KUYRUK(m)
15. DAKİKA	BATI YÖNÜ	*589*	*4075*	*115*
	DOĞU YÖNÜ	*427*	*1272*	*0*
	KUZEY YÖNÜ	*169*	*4161*	*77*
	GÜNEY YÖNÜ	*169*	*4127*	*74*
30. DAKİKA	BATI YÖNÜ	*1199*	*8361*	*69*
	DOĞU YÖNÜ	*853*	*2479*	*0*
	KUZEY YÖNÜ	*345*	*8620*	*95*
	GÜNEY YÖNÜ	*347*	*8730*	*69*
45. DAKİKA	BATI YÖNÜ	*1760*	*11357*	*160*
	DOĞU YÖNÜ	*1286*	*4428*	*53*
	KUZEY YÖNÜ	*522*	*13469*	*0*
	GÜNEY YÖNÜ	*521*	*13625*	*0*
60. DAKİKA	BATI YÖNÜ	*2367*	*15762*	*72*
	DOĞU YÖNÜ	*1723*	*5701*	*0*
	KUZEY YÖNÜ	*690*	*17417*	*79*
	GÜNEY YÖNÜ	*689*	*17636*	*76*

Tablo 4.6 Sabit saykıl ile yapılan sabah trafiği sayım sonuçları

NORMAL TRAFİK IŞIKLARI İLE YAPILAN SABAH TRAFİK SAYIMI	YÖN	ARAÇ SAYISI	GECİKME SÜRESİ(SN)	KUYRUK(m)
15. DAKİKA	BATI YÖNÜ	474	6227	141
	DOĞU YÖNÜ	422	4232	45
	KUZEY YÖNÜ	172	3574	60
	GÜNEY YÖNÜ	173	3567	61
30. DAKİKA	BATI YÖNÜ	962	2597	97
	DOĞU YÖNÜ	840	8327	55
	KUZEY YÖNÜ	345	7278	58
	GÜNEY YÖNÜ	348	7386	43
45. DAKİKA	BATI YÖNÜ	1476	19152	41
	DOĞU YÖNÜ	1283	12538	13
	KUZEY YÖNÜ	527	11275	31
	GÜNEY YÖNÜ	532	11462	39
60. DAKİKA	BATI YÖNÜ	1957	25476	199
	DOĞU YÖNÜ	1703	16930	102
	KUZEY YÖNÜ	694	14622	63
	GÜNEY YÖNÜ	693	14941	92

Tablo 4.7 Bulanık mantıkla yapılan öğlen trafiği sayım sonuçları

BULANIK MANTIK DENETİMİYLE YAPILAN ÖĞLEN TRAFİK SAYIMI	YÖN	ARAÇ SAYISI	GECİKME SÜRESİ(SN)	KUYRUK(m)
15. DAKİKA	BATI YÖNÜ	431	2555	54
	DOĞU YÖNÜ	433	2973	72
	KUZEY YÖNÜ	310	4108	63
	GÜNEY YÖNÜ	310	3942	32
30. DAKİKA	BATI YÖNÜ	845	5030	51
	DOĞU YÖNÜ	850	5610	65
	KUZEY YÖNÜ	616	8271	27
	GÜNEY YÖNÜ	617	7411	14
45. DAKİKA	BATI YÖNÜ	1278	7532	0
	DOĞU YÖNÜ	1284	8217	0
	KUZEY YÖNÜ	920	12423	91
	GÜNEY YÖNÜ	919	11200	83
60. DAKİKA	BATI YÖNÜ	1709	9752	0
	DOĞU YÖNÜ	1717	10714	0
	KUZEY YÖNÜ	1218	17112	88
	GÜNEY YÖNÜ	1217	15566	97

Tablo 4.8 Sabit saykıl ile yapılan öğlen trafiği sayım sonuçları

NORMAL TRAFİK IŞIKLARI İLE YAPILAN ÖĞLEN TRAFİK SAYIMI	YÖN	ARAÇ SAYISI	GECİKME SÜRESİ(SN)	KUYRUK(m)
15. DAKİKA	BATI YÖNÜ	413	4159	90
	DOĞU YÖNÜ	406	4204	100
	KUZEY YÖNÜ	214	5242	90
	GÜNEY YÖNÜ	204	5203	92
30. DAKİKA	BATI YÖNÜ	844	8416	33
	DOĞU YÖNÜ	819	8435	35
	KUZEY YÖNÜ	428	10621	100
	GÜNEY YÖNÜ	407	10398	97
45. DAKİKA	BATI YÖNÜ	1271	12584	0
	DOĞU YÖNÜ	1251	12772	0
	KUZEY YÖNÜ	643	16076	98
	GÜNEY YÖNÜ	613	15584	98
60. DAKİKA	BATI YÖNÜ	1697	16844	0
	DOĞU YÖNÜ	1670	17143	0
	KUZEY YÖNÜ	855	21427	85
	GÜNEY YÖNÜ	819	20743	31

Tablo 4.9 Bulanık mantıkla yapılan akşam trafiği sayım sonuçları

BULANIK MANTIK DENETİMİYLE YAPILAN AKŞAM TRAFİK SAYIMI	YÖN	ARAÇ SAYISI	GECİKME SÜRESİ(SN)	KUYRUK(m)
15. DAKİKA	BATI YÖNÜ	413	2372	90
	DOĞU YÖNÜ	416	2560	90
	KUZEY YÖNÜ	303	4374	9
	GÜNEY YÖNÜ	303	4000	9
30. DAKİKA	BATI YÖNÜ	852	4687	9
	DOĞU YÖNÜ	854	5026	21
	KUZEY YÖNÜ	597	9046	36
	GÜNEY YÖNÜ	596	8380	45
45. DAKİKA	BATI YÖNÜ	1279	6845	5
	DOĞU YÖNÜ	1281	7238	23
	KUZEY YÖNÜ	888	13542	88
	GÜNEY YÖNÜ	887	12500	97
60. DAKİKA	BATI YÖNÜ	1704	8990	60
	DOĞU YÖNÜ	1711	9620	58
	KUZEY YÖNÜ	1188	18242	56
	GÜNEY YÖNÜ	1187	16928	43

Tablo 4.10 Sabit saykıl ile yapılan akşam trafiği sayım sonuçları

NORMAL TRAFİK IŞIKLARI İLE YAPILAN AKŞAM TRAFİK SAYIMI	YÖN	ARAÇ SAYISI	GECİKME SÜRESİ(SN)	KUYRUK(m)
15. DAKİKA	BATI YÖNÜ	411	4159	95
	DOĞU YÖNÜ	405	4223	100
	KUZEY YÖNÜ	214	5242	98
	GÜNEY YÖNÜ	204	5202	92
30. DAKİKA	BATI YÖNÜ	842	8400	45
	DOĞU YÖNÜ	827	8434	55
	KUZEY YÖNÜ	428	10621	100
	GÜNEY YÖNÜ	407	10497	97
45. DAKİKA	BATI YÖNÜ	1273	12578	0
	DOĞU YÖNÜ	1253	12771	0
	KUZEY YÖNÜ	643	16078	98
	GÜNEY YÖNÜ	623	15980	98
60. DAKİKA	BATI YÖNÜ	1689	16838	0
	DOĞU YÖNÜ	1671	17142	0
	KUZEY YÖNÜ	854	21371	98
	GÜNEY YÖNÜ	818	20870	93

Tablo 4.5-4.10 daki tablolarda uygulama projesinin çalıştırılması sonucu elde edilen sonuçlar görülmektedir. Tablo 4.11 da karşılaştırmalı sonuçların yer aldığı tablo vardır.

Tablo 4.11 Bulanık mantık kontrolörü ile sabit saykıl ile yapılan kontrol arasındaki karşılaştırmalı sonuçlar tablosu

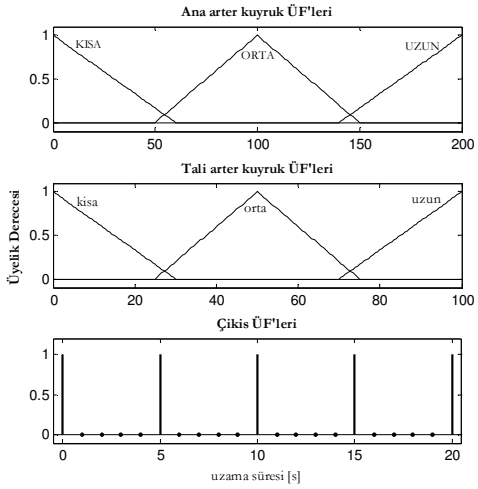
KONTROL TİPİ	YÖN	ARAÇ SAYISI	GECİKME SÜRESİ(SN)	ARAÇ SAYISI (BULANIK - TRAFİK)	ARAÇ SAYI % FARK	GECİKME SÜRESİ FARK (TRAFİK-BULANIK)	GECİKME SÜRESİ % FARK (TR-BUL/TR)
BULANIK SABAH	BATI YÖNÜ	*2367*	*15762*	**410**	**20,95%**	**9714**	**38,13%**
	DOĞU YÖNÜ	*1723*	*5701*	**20**	**1,17%**	**11229**	**66,33%**
	KUZEY YÖNÜ	*690*	*17417*	**-4**	**-0,58%**	**-2795**	**-19,12%**
	GÜNEY YÖNÜ	*689*	*17636*	**-4**	**-0,58%**	**-2695**	**-18,04%**
TRAFİK SABAH	BATI YÖNÜ	*1957*	*25476*				
	DOĞU YÖNÜ	*1703*	*16930*				
	KUZEY YÖNÜ	*694*	*14622*				
	GÜNEY YÖNÜ	*693*	*14941*				
BULANIK ÖĞLEN	BATI YÖNÜ	*1709*	*9752*	**12**	**0,71%**	**7092**	**42,10%**
	DOĞU YÖNÜ	*1717*	*10714*	**47**	**2,81%**	**6429**	**37,50%**
	KUZEY YÖNÜ	*1218*	*17112*	**363**	**42,46%**	**4315**	**20,14%**
	GÜNEY YÖNÜ	*1217*	*15566*	**398**	**48,60%**	**5177**	**24,96%**
TRAFİK ÖĞLEN	BATI YÖNÜ	*1697*	*16844*				
	DOĞU YÖNÜ	*1670*	*17143*				
	KUZEY YÖNÜ	*855*	*21427*				
	GÜNEY YÖNÜ	*819*	*20743*				
BULANIK AKŞAM	BATI YÖNÜ	*1704*	*8990*	**15**	**0,89%**	**7848**	**46,61%**
	DOĞU YÖNÜ	*1711*	*9620*	**40**	**2,39%**	**7522**	**43,88%**
	KUZEY YÖNÜ	*1188*	*18242*	**334**	**39,11%**	**3129**	**14,64%**
	GÜNEY YÖNÜ	*1187*	*16928*	**369**	**45,11%**	**3942**	**18,89%**
TRAFİK AKŞAM	BATI YÖNÜ	*1689*	*16838*				
	DOĞU YÖNÜ	*1671*	*17142*				
	KUZEY YÖNÜ	*854*	*21371*				
	GÜNEY YÖNÜ	*818*	*20870*				

Tablo 4.11 deki karşılaştırmalı tablo yorumlandığında bulanık kontrolör ile yapılan sinyalizasyon ile normal trafik sinyalizasyonu arasındaki fark daha açık şekilde görülmektedir. Tablodaki deney sonuçlarından sadece sabah trafiğinde, normal trafik sinyalizasyonu gecikme ve bekleme süresinde % 19 gibi bir üstünlük sağladı fakat geri kalan diğer bütün durumlarda bulanık kontrolörü ile yapılan trafik sinyalizasyonundan alınan sonuçlar açık şekilde olumlu neticeler ortaya koymaktadır. Araç sayılarında bazı durumlarda bulanık kontrolörü yapılan trafik sinyalizasyonu % 45 e varan oranlarda üstünlük sağlamakta ; bekleme ve

gecikme zamanında da bazı trafik saatlerinde % 66 oranlarına varan üstünlük sağlamaktadır.

Tablo 4.11 deki değerler sadece 1 saatlik süre içerisinde alınan değerlerdir. Bariz şekilde aradaki olumlu fark görünmektedir. Özellikle trafiğin yoğun olduğu saatlerde bulanık kontrolörü ile yapılan sinyalizasyonun araç sayıları ve bekleme zamanlarını nasıl etkilediği görülmektedir.

Uygulama projesinde kullanılan matlab çalışması :



Şekil 4.16 B-D ve K-G kulvarlarına ait giriş-çıkış üyelik fonksiyonları

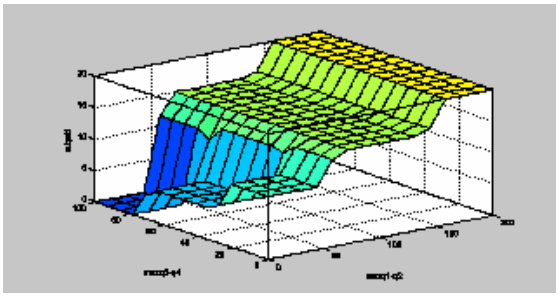
Şekil 4.16 da B-D ve K-G kulvarlarına ait giriş üyelik fonksiyonları ve çıkış üyelik fonksiyonları görülmektedir. Akıllı trafik sinyalizasyonu projesinde ve(and) metodu olarak "minimum", veya(or) metodu olarak "maksimum", çıkarım(implication) olarak "minimum", çıkış üyelik fonksiyonlarının toplanmasında "maksimum" ve çıkış keskin değerin bulunması için durulandırma(defuzzification) için de ağırlıklı ortalama yöntemi(WA) kullanılmıştır. [8,9]

Ana arter yeşil yanma süresini kontrol eden bulanık kontrolörünün matlab editöründe yazılı kural tabanı Şekil 4.17 de ifade edilmiştir.

1. if(maxq1-q2 is küçük) and (maxq3-q4 is küçük) then (output1 is O)
2. 2. if(maxq1-q2 is küçük) and (maxq3-q4 is orta) then (output1 is K)
3. if(maxq1-q2 is küçük) and (maxq3-q4 is büyük) then (output1 is ÇK)
4. if(maxq1-q2 is orta) and (maxq3-q4 is küçük) then (output1 is B)
5. if(maxq1-q2 is orta) and (maxq3-q4 is orta) then (output1 is B)
6. if(maxq1-q2 is orta) and (maxq3-q4 is büyük) then (output1 is B)
7. if(maxq1-q2 is büyük) and (maxq3-q4 is küçük) then (output1 is ÇB)
8. if(maxq1-q2 is büyük) and (maxq3-q4 is orta) then (output1 is ÇB)
9. if(maxq1-q2 is büyük) and (maxq3-q4 is büyük) then (output1 is ÇB)

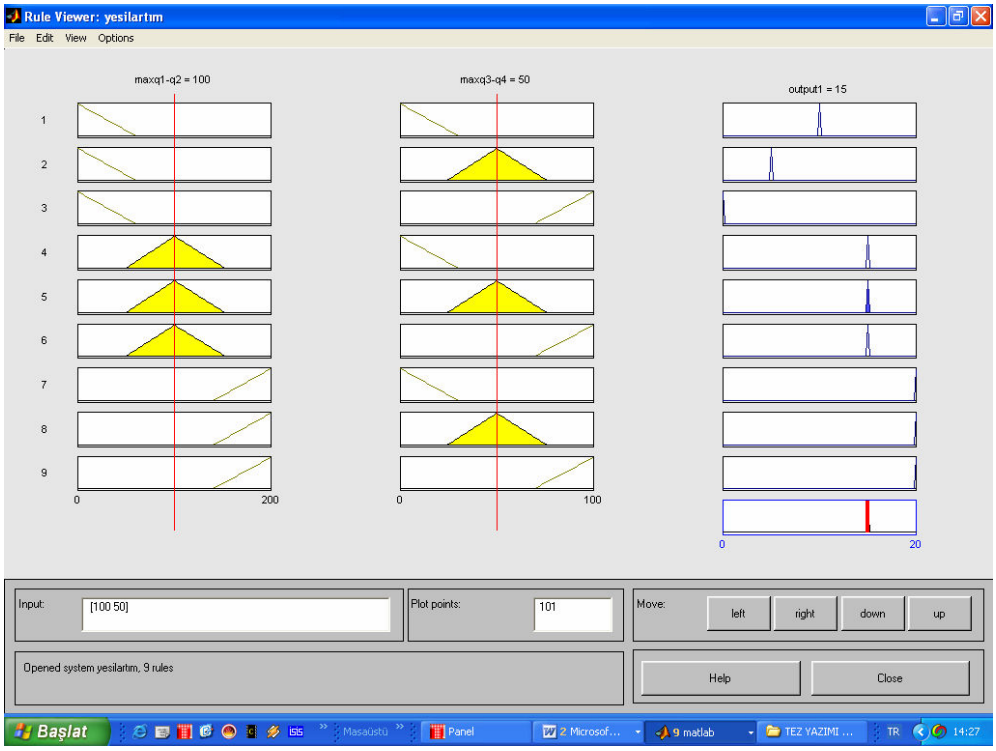
Şekil 4.17 B-D Kulvarı Yeşil Yanma Süresi Kural Tabanı

D-B yönüne ait uzama zamanı yüzey şekli şekil 4.18 de gösterilmiştir. [7]



Şekil 4.18 Doğu-Batı yönü çıkış yüzey şekli

D-B yönüne ait ait çıkış keskin değerinin aldığı örnek bir değer, Şekil 4.5 de görülmektedir. [7]



Şekil 4.19 Batı-Doğu yönü kural tabanına göre keskin değer elde edilişi

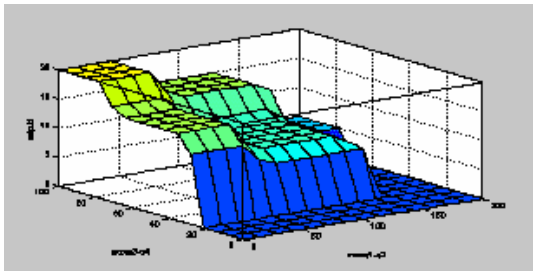
Şekil 4.19 da görüldüğü gibi yeşilartım matlab "rule Viewer" şeklinden giriş kuyruk uzunluk değerlerine göre çıkış keskin değerinin almış olduğu değer görülmektedir. Giriş B-D kuyruk uzunluğunun 100 metre olduğu ve K-G kuyruk uzunluğunun 50 metre olduğu iki adet giriş değerinin kural tabanına göre çıkış keskin değerinin output1=15 sn olduğu görülmektedir. Bunun anlamı Şekil 5.5 da görülen kuyruk uzunluklarının olduğu bir anda bulanık mantık kontrolörünün çıkış uzama faz değerinin alacağı keskin değerinin 15 sn olacağıdır. Yani mevcut yeşil fazı 15 sn. daha fazla uzatılarak devam ettirilecek demektir. 15 sn. süre bitiminde tekrar kuyruk uzunluklarına bakılarak yeni uzama fazı hesap edilecektir. Bu şekilde döngü ifadesi 5 kere tekrar eder. Eğer uzama zamanı sıfır değerini alırsa bir sonraki ışık fazına geçilerek ışık yanması değişir. Yeşil yanıyorsa turuncuya ve sonra da kırmızıya döner. Her durumda yeşil ve kırmızı ışık maksimum yanma süreleri 120 sn. süresini geçmeyecek şekilde yazılım ayarlanmıştır.

Kuzey-güney yönü yeşil süresi artımı ya da kırmızı fazı süre artımı için kural tabanı da Şekil 4.20 de görülmektedir.

1. if(maxq1-q2 is küçük) and (maxq3-q4 is küçük) then (output1 is ÇK)
2. 2. if(maxq1-q2 is küçük) and (maxq3-q4 is orta) then (output1 is B)
3. if(maxq1-q2 is küçük) and (maxq3-q4 is büyük) then (output1 is ÇB)
4. if(maxq1-q2 is orta) and (maxq3-q4 is küçük) then (output1 is ÇK)
5. if(maxq1-q2 is orta) and (maxq3-q4 is orta) then (output1 is O)
6. if(maxq1-q2 is orta) and (maxq3-q4 is büyük) then (output1 is B)
7. if(maxq1-q2 is büyük) and (maxq3-q4 is küçük) then (output1 is ÇK)
8. if(maxq1-q2 is büyük) and (maxq3-q4 is orta) then (output1 is ÇK)
9. if(maxq1-q2 is büyük) and (maxq3-q4 is büyük) then (output1 is K)

Şekil 4.20 K-G Kulvarı Yeşil Yanma Süresi Kural Tabanı

Kırmızı artıma ait yüzey şekli Şekil 4.21 de görülmektedir.

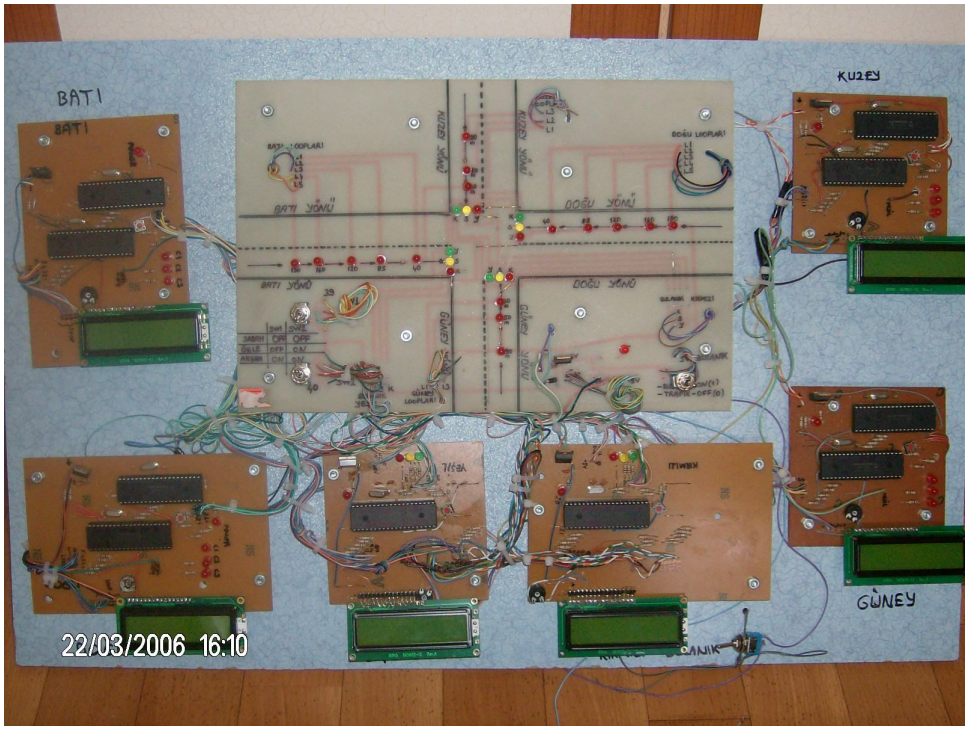


Şekil 4.21 Kuzey-Güney yönü yeşil süresi artımı için yüzey şekli

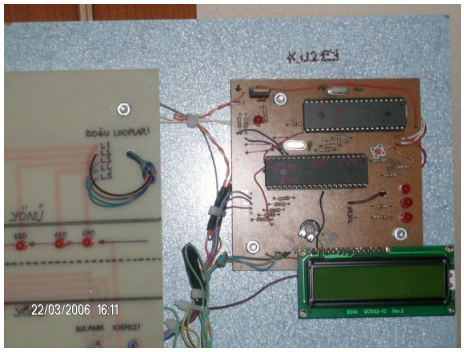
Uygulama projesinde donanımsal gerçekleştirilen değerlerle matlab sonuçları da karşılaştırılarak sonuçlar test edilmiştir. Değerler birbirine yakın çıkmıştır. Üçüncü bir test etme şekli de uygulama yapılmadan bütün sonuçlar Proteus paket programı üzerinde denenerek çıkması gerekli muhtemel sonuçlar görülerek donanımsal gerçekleşmeye geçilmiştir. Burada da sonuçlar hemen hemen aynı çıkmıştır.

Şekil 4.22 de uygulama projesi trafik sinyalizasyonuna ait fotoğraf görülmektedir. Şekil 4.23 ve Şekil 4.24 de ise kuzey ve güney yönlerine ait simülasyon kartları görülmektedir. Şekil 4.22 de ortadaki büyük kart ise dört

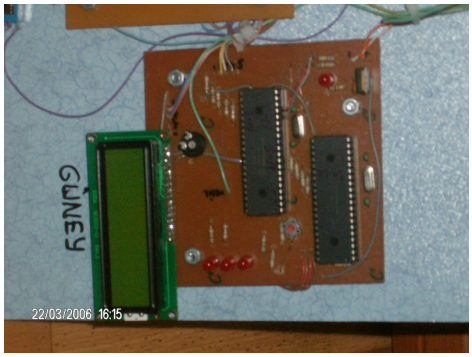
yöne ait kavşak yapısını gösteren ve üzerinde bulunan kırmızı ledlerle araç dedektörlerinin yerlerini gösteren ve trafik ışıklarını gösteren simülasyon ortamını göstermektedir. Uygulama projesi çalışırken geçen araç sayıları, kuyruk durumları ve gecikme zamanları her yöne ait simülasyon kartından ve bulanık kontrolörleri kartları üzerindeki LCD ekranlardan görülebilmektedir. Bu şekilde her bir yöne ait araç geçişleri görsel olarak takip edilebilmekte ve bu değerler alınarak 1 saatlik süre içinde bütün yönlere ait geçişler öğrenilebilmektedir.



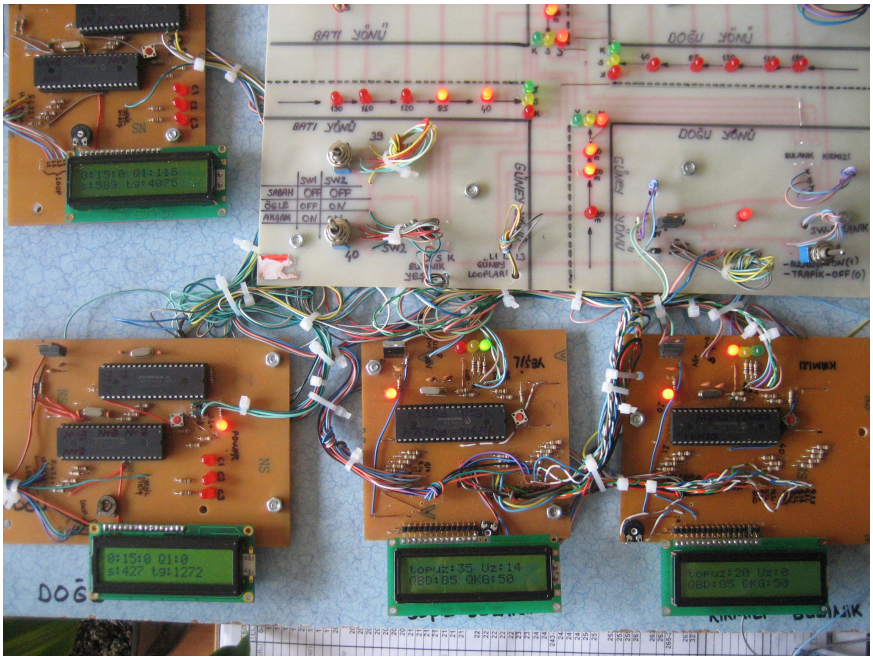
Şekil 4.22 Uygulama devresi bitmiş hali



Şekil 4.23 Uygulama devresi kuzey yönü araç üreteci ve simülatörü kartı



Şekil 4.24 Uygulama devresi güney yönü araç üreteci ve simülatörü kartı

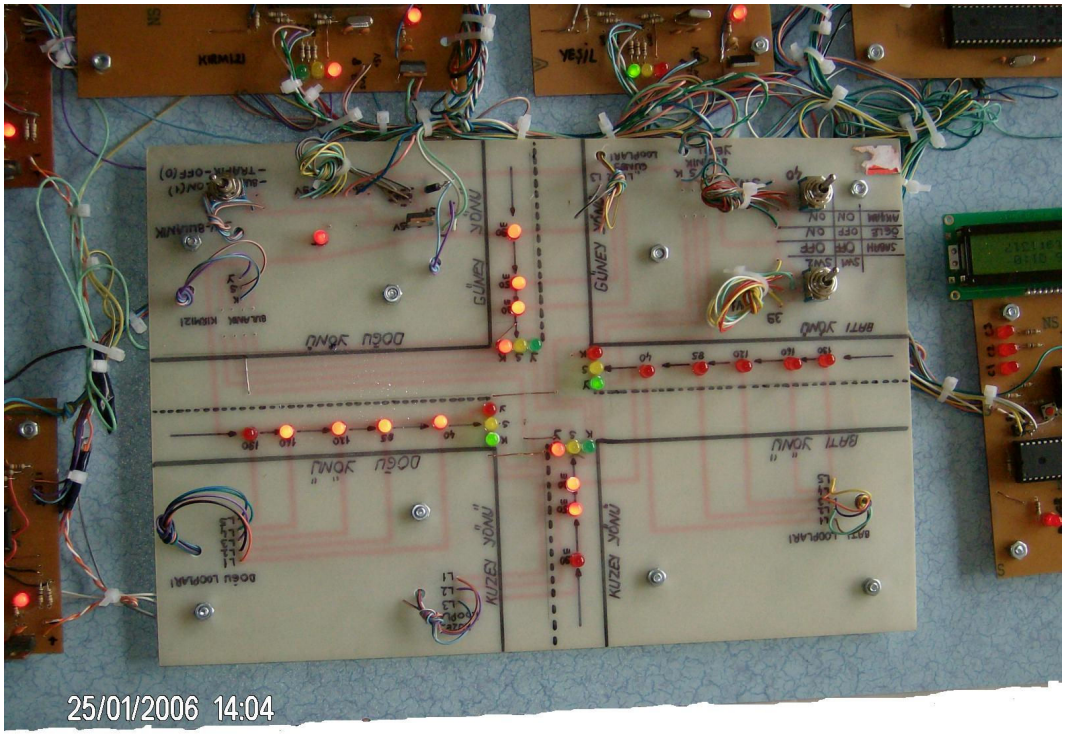


Şekil 4.25 Uygulama devresi ledlerin durumu ve LCD panellerin görünümü

Şekil 4.25 de uygulama projesi çalışırken kırmızı ve yeşil renkli bulanık kontrolörleri LCD ekranlarında devre çalışırken değerler görülmektedir. LCD ekran üzerinden de görüldüğü gibi topuz ile toplam uzama zamanı, uz ile o anki uzama zamanını, QBD ile batı-doğu yönüne ait kuyruk uzunluğu değeri ve QKG ile de kuzey-güney yönüne ait kuyruk uzunluğu değeri görülmektedir.

Şekil 4.25 de kuzey, güney ve doğu yönüne ait çalışma anındaki görüntü şekli görülmektedir. Şekilden görüldüğü gibi doğu yönüne ait kuyruk uzunluğu 160. metrede görülmektedir. Güney yönü 100 metre mesafe bütünüyle dolu olduğu yanan ledlerden görülmektedir. Kuzey yönünde ise 50. metreye kadar araç

kuyuğu olduğu yanan ledlerden görülmektedir. Çalışma anında B-D yönüne ait trafik ışıkları yeşil renk yanmaktadır.



Şekil 4.26 Uygulama devresi çalışırken yönlere ait ledlerin ve anahtarların durumu

Şekil 4.26 da SW1 ve SW2 şeklinde görülen iki adet anahtar ile sabah, öğlen ve akşam trafik modu seçimi yapılmaktadır.

Şekil 4.22 de görülen devrede 6 adet baskı devre kartı görülmektedir. Bu kartlar üzerinde bulunan pic16F877 mikrodenetleyicileri üzerinde C dili ile yazılan yazılımlar bulunmaktadır. Pic16F877 mikronkontrolörleri üzerinde araç üretici yazılımı olarak sırasıyla Şekil 4.27 de görülen C programları HEX yazılımı kodları yüklüdür.

Batı yönü araç üreteci devresi	W_üretici.hex
Doğu yönü araç üretici devresi	E_üretici.hex
Kuzey yönü araç üretici devresi	N_üretici.hex
Güney yönü araç üretici devresi	S_üretici.hex

Şekil 4.27 Derlenmiş Araç Üretici Devre Hex Kod Dosya İsimleri

Araç simülatörü olarak sırasıyla Şekil 4.28 de görülen yazılımlar yüklüdür.

Batı yönü araç simülatörü	E_W_kontrolörü.hex
Doğu yönü araç simülatörü	E_W_kontrolörü.hex
Kuzey yönü araç simülatörü	N_S_kontrolörü.hex
Güney yönü araç simülatörü	N_S_kontrolörü.hex

Şekil 4.28 Derlenmiş Simülatör Hex Kod Dosya İsimleri

Bulanık kontrolörü olarak da sırasıyla Şekil 4.29 da görülen yazılımlar yüklüdür.

Yeşil ışık bulanık kontrolörü	fuzzy_logic_yeşil.hex
Kırmızı ışık bulanık kontrolörü	fuzzy_logic_kırmızı.hex

Şekil 4.29 Derlenmiş Bulanık Kontrolör Hex Kod Dosya İsimleri

Bütün C dili ifadeleri ve yazılımların HEX kodu ifadeleri tezin bulunduğu kompakt CD de bulunan "osman_demirci_yazılımlar_ek.pdf " dosyasında bulunmaktadır. C dili ifadeleri HEX uzantılı derlenmiş dosya isimleriyle aynı isme sahiptir; tek fark uzantı dosya isimler. ".c" şeklindedir. Anahtar pozisyonlarına göre trafik modu seçimi Tablo 4.12 de görüldüğü şekilde yapılmaktadır.

Tablo 4.12 Trafik modunu seçen anahtar durumları

	SW1	**SW2**
SABAH	OFF(0)	OFF
ÖĞLEN	OFF	ON
AKŞAM	ON(1)	ON

Şekil 4.26 da görülen "fuzzy" anahtarı ile bulanık kontrolü ve normal trafik kontrolörü arasında seçim yapılmaktadır. "fuzzy" anahtarı "1 (on)" konumunda iken devre bulanık kontrolörü olarak görev yapmaktadır. "fuzzy" anahtarı "0 (off)" konumunda iken devre normal trafik kontrolörü olarak görev yapmaktadır. Ekler kısmında ise uygulama devresinde kullanılan C dili yazılımların kaynak kodları verilmiştir.

5. PROJE SONUÇLARI VE YORUMLANMASI

Tablo 5.1, 5.2 ve Tablo 5.3 de 1 saatlik süre için sabah, öğlen ve akşam trafik durumlarına göre uygulama projesi çalıştırılarak alınan deneysel sonuçlar tablo haline getirilmiştir. 1 saatlik süre için bulanık mantık ve normal trafik kontrolü ile yapılan trafik kontrollerinden elde edilen araç geçiş sayıları ile kırmızı ışık yanarken ortaya çıkan araç kuyruklarından dolayı oluşan bekleme ve duraklama zamanları yazılmıştır.

Tablolarda belirtilen batı-doğu ile batı ve doğu kulvarındaki toplam araç sayıları ve bekleme zamanlarının olduğu kulvarları ifade etmekte; aynı şekilde kuzey-güney ile de kuzey ve güney yönlerine ait değerler ifade edilmektedir.

Tablo 5.1 Sabah saati için elde edilen değerler

değerlendirme tablosu (1 saat için)		
	Bulanık Sabah	
	araç sayısı(adet)	**bekleme zamanı(sn)**
Batı-doğu	4090	21463
Kuzey-güney	1379	35053
toplam	**5469**	**56516**
	Normal trafik Sabah	
	araç sayısı(adet)	**bekleme zamanı(sn)**
Batı-doğu	3660	42406
Kuzey-güney	1387	29563
toplam	**5047**	**71969**
farklar	***422***	***-15453***
365 günlük fark	154030	-5640345
2 şerit olması durumunda	308060	-11280690
3 şerit olması durumunda	462090	-16921035

Tablo 5.2 Öğlen saati için elde edilen değerler

değerlendirme tablosu (1 saat için)		
	Bulanık Öğlen	
	araç sayısı(adet)	**bekleme zamanı(sn)**
Batı-doğu	3426	20466
Kuzey-güney	2435	32678
toplam	**5861**	**53144**
	Normal trafik Öğlen	
	araç sayısı(adet)	**bekleme zamanı(sn)**
Batı-doğu	3367	33987
Kuzey-güney	1674	24170
toplam	**5041**	**58157**
farklar	***820***	***-5013***
365 günlük fark	299300	-1829745
2 şerit olması durumunda	598600	-3659490
3 şerit olması durumunda	897900	-5489235

Tablo 5.3. Akşam saati için elde edilen değerler

değerlendirme tablosu (1 saat için)		
	Bulanık Akşam	
	araç sayısı(adet)	**bekleme zamanı(sn)**
Batı-doğu	3415	18610
Kuzey-güney	2375	35170
toplam	**5790**	**53780**
	Normal trafik Akşam	
	araç sayısı(adet)	**bekleme zamanı(sn)**
Batı-doğu	3360	33980
Kuzey-güney	1672	42241
toplam	**5032**	**76221**
farklar	***758***	***-22441***
365 günlük fark	276670	-8190965
2 şerit olması durumunda	553340	-16381930
3 şerit olması durumunda	830010	-24572895

Tablo 5.4 Sabah, öğlen ve akşam saatleri için 3 saatlik sürenin toplam değerleri

değerlendirme tablosu (3 saat için)		
	araç sayısı(adet)	**bekleme zamanı(sn)**
farklar	***2000***	***-42907***
365 günlük fark	730000	-15661055
2 şerit olması durumunda	1460000	-31322110
3 şerit olması durumunda	2190000	-46983165

Tablo 5.1 de 1 saatlik sabah trafiği incelendiğinde bulanık mantık ve normal trafik kontrolü ile yapılan trafik sonuçlarına bakılırsa sabah trafiği için bulanık mantık kontrolü ile fazladan 422 araç geçirildiği görülmektedir. 422 araç fazladan geçerken aynı zamanda 1 saatlik zaman içerisinde toplamda 15453 sn daha az bekleme zamanı olmaktadır. Bir yıl için sonuçların toplamı değerlendirildiğinde yıl içerisinde fazladan 154030 araç fazladan geçmekte ve 5640345 sn daha az bekleme zamanı olmaktadır.

Tablo 5.2 de 1 saatlik öğlen trafiği incelendiğinde bulanık mantık ve normal trafik kontrolü ile yapılan trafik sonuçlarına bakılırsa öğlen trafiği için bulanık mantık kontrolü ile fazladan 820 araç geçirildiği görülmektedir. 820 araç fazladan geçerken aynı zamanda 1 saatlik zaman içerisinde toplamda 5013 sn daha az bekleme zamanı olmaktadır. Bir yıl için sonuçların toplamı değerlendirildiğinde yıl içerisinde 299300 araç fazladan geçmekte ve 1829745 sn daha az bekleme zamanı olmaktadır.

Tablo 5.3 de 1 saatlik akşam trafiği incelendiğinde bulanık mantık ve normal trafik kontrolü ile yapılan trafik sonuçlarına bakılırsa öğlen trafiği için bulanık mantık kontrolü ile fazladan 758 araç geçirildiği görülmektedir. 758 araç fazladan geçerken aynı zamanda 1 saatlik zaman içerisinde toplamda 22441 sn daha az bekleme zamanı olmaktadır. Bir yıl için sonuçların toplamı değerlendirildiğinde yıl içerisinde fazladan 276670 araç fazladan geçmekte ve 8190965 sn daha az bekleme zamanı olmaktadır.

Tablo 5.4 incelendiğinde de 3 saatlik sabah, öğlen ve akşam trafiği tablo sonuçlarının toplamları alınarak incelendiğinde bulanık mantık ve normal trafik

kontrolü ile yapılan trafik sonuçlarına bakılırsa 3 saatlik toplam bulanık kontrolörü ile yapılan trafik kontrolü ile fazladan 2000 araç geçirildiği görülmektedir. 2000 araç fazladan geçerken aynı zamanda 3 saatlik zaman içerisinde toplamda 42907 sn daha az bekleme zamanı olmaktadır. Bir yıl için sonuçların toplamı değerlendirildiğinde yıl içerisinde fazladan 730000 araç geçmekte ve 15661055 sn daha az bekleme zamanı olmaktadır.

Tablo 5.4 e göre eğer kulvar iki şeritli olsaydı bu sonuçlar yaklaşık 2 ile çarpılacaktı o zaman da 1460000 adet araç fazladan geçerken 31322110 sn daha az bekleme zamanı oluşacaktı. Tablo 5.4 e göre kulvar 3 şeritli olması durumunda da bu 2190000 araç fazladan geçerken , 46983165 sn daha az bekleme süresi olabilecekti.

Tablo 5.1, 5.2 ve Tablo 5.3 deki sonuçlar sadece günün 3 saatlik sonuçlarıdır. 24 saat dışında 21 saat daha vardır. Burada oluşacak araç geçişi ve bekleme zamanları da alınsa ortaya çıkacak fark çok daha fazla olacaktır. Tablo 5.1, 5.2, 5.3 ve Tablo 5.4 deki sonuçların yorumunda da görüldüğü gibi fazladan geçen her araç trafik akışını ve yükünü rahatlatacağı gibi kulvarda yapılan daha az bekleme zamanı ile de yakıt tüketiminin daha az seviyede olması sağlanmaktadır. Bekleme yaparken araçların motorların rölantide çalışması da gereksiz yere yakıt tüketimine sebep olduğu için bekleme sürelerini azaltarak yakıt tüketimi de azaltılmış olmaktadır.

Trafikte bekleme süresinin azaltılması ile, klakson sesi, motor gürültüsü gibi gürültü kirliliği de azaltılmış olacaktır. Bekleme süresinin azaltılmasına paralel olarak egzosdan çıkan zararlı gazlar azalacağı için çevre kirliliği de bir nebze olsun azalacaktır.

Bekleme süresi azalacağından ve geçen araç sayısı artacağından trafik sıkışıklıkları da nispeten daha az olacak ve sürücülerin psikolojisi üzerindeki olumsuz etkilerde daha az olacaktır.

Bulanık mantık bugün dünyada gittikçe daha fazla kullanım alanı bulmakta ve hayatın her alanına girmektedir. Trafikte de henüz hak ettiği yeri alamamasına rağmen üzerinde yapılan çalışmalar yakın zamanda trafik kontrollerinde de bulanık mantık uygulamalarının giderek yaygınlaşacağını göstermektedir.

KAYNAKLAR

1. Chih-Hsun Chou-Jen-Chao Teng , Chung-Hua University ,A fuzzy logic controllor for traffic junction signals, ***Information Sciences***,143,73–97, (2002)

2. Jarkko Niittymaki From Helsinki Universiyt,İnstallation and experiances of field testing a fuzzy signal controller, ***European Jurnal of Operational Research***, 131, 273-281, (2001)

3. Mohamed Trabia-Mohamed Kaseko-Murali Ando , A Two stage Fuzzy Logic Controller for traffic signals, ***Transportation Research Part***, C7, 353-367, (1999)

4. İbrahim, A., Gömülü sistemlerle bulanık mantık", Bileşim Yayınevi, 1-70, (2004)

5. Şenol, F., Lisans Tezi, **Gazi *Üniversitesi Teknik Eğitim Fakültesi***, Ankara, 1-35, (2000)

6. Shahariz bin Abdul Aziz, 1996**,** Fuzzy Traffic Light Controller, http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol2/sbaa/article2.html , (Ziyaret tarihi: 9 Şubat 2006)

7. Jantzen, J., Design of Fuzzy Controller, Technical University Of Danimark***, Tech. report no 98-E 864***, 1-27, 2-24, (19 Ağustos 1998)

8. Gonzales ve diğerleri (2000), "A Fuzzy Logic Approach For Car-Following modelling", Vol 42(5), 735-755 (2000)

9. Khalid, M., "The need for artificial intelligance and its applications in industry", Center Of Artificial İntelligance and Robotic(CAIRO) University Teknology Malaysia, Kuala Lumpur

10. Halle ve Draa, (2005) "A colloborative driving system based on multiagent modelling and simulations", www.elsevier.com/locate/trc, Transportation Research Part , C 13 , 320–345(ziyaret tarihi: 9 Ağustos 2006).

11. Khiang ve diğerleri, İntelligent Traffic Lights Control by Fuzzy Logic, ***Malaysian Journal of Computer Science***, Vol. 9 No., pp. 29-352, (December 1996)

12. Sheu, Jiuh-Bing, "A fuzzy clustering approach to real-time deman-responsive bus dispatching and Transportation", ***Fuzzy Sets and systems***, 150, pp. 437-455, (2005)

13. Lee ve Kim (2000), "A Lingitutional control system for a platon of vehicles using a fuzzy-sliding mode algorithm", ***Mechatronic***, 12, 97-118, (2002)

14. Selekwa ve diğerleri (2003), "Application of LQ modelling and optimization in Urban traffic control", ***Optimal Control Application and Methods***, 24, pp. 331-345, (2002)

15. Wiering ve diğerleri, "İntelligent traffic Light Control", ***Utrecht University technical Report UU-CS-2004-029***, pp. 1-31, (2004)

16. Godoy, Simoes, Colorado School of Mines Engineering Division, "Fuzzy Control System", http://egweb.mines.edu/faculty/msimoes/tutorials/Introduction_ fuzzy_logic/Intro_Fuzzy_Logic.pdf, (ziyaret tarihi: 9 Ağustos 2006).

17. Goebel, Greg, 01.Jun.2003, "An Introduction To Fuzzy Logic", http:// www.facstaff.bucknelloedu/mastascu/eControlHTML/Fuzzy/Fuzzy1.html, (ziyaret tarihi: 2 Mart 2006).

EKLER

EK.1 E_W_Kontrolörü.c Kaynak Kodları

```
#ifdef __PCH__
#device PIC16F877
#endif

#include <16F877.H>
#device  *=16
#fuses XT,NOWDT
#use delay(clock=4000000)
#include <lcd.c>
#define LOOPBES pin_C7
#define LOOPDORT pin_C6
#define LOOPUC pin_C5
#define LOOPIKI pin_C4
#define LOOPBIR pin_B0
#define aracvar pin_A0
#define cins1 pin_A1 //tır=12 m   otobus=8 m
#define cins2 pin_A2  //minibus=6 taksi=4 m
#define cins3 pin_A3
#define onay pin_E0
#define dolu pin_A5 // kulvar doluysa 1 oluyor
#define karsiserit_dolu pin_C2//karşı şerit durumunu alıyor
#define karsiserit_onay pin_C1// karşı şerit onay
#define karsiserit_cins1 pin_B5
#define karsiserit_cins2 pin_B7
#define karsiserit_cins3 pin_B6
#define karsiserit_aracvar pin_B3
#define yesil pin_B4
              unsigned int TM1=0;
              unsigned int  TM2=0;
              unsigned int  DK=0;
  unsigned int  SA=0;
              unsigned int SN=0;
  unsigned int q1=0;
  unsigned int i=0;
  unsigned int m=0;
  unsigned int j=1;
  unsigned int t=1;
  unsigned int k=1;
  unsigned int aralik=0;
  long int sayi=0;
  unsigned int ison=1;
  unsigned int araccins;
  unsigned int tor=0;
  unsigned int tg=0;
  long int top=0;
  short int cift=0;
  unsigned int sn1=0;
  unsigned int sn2=0;
  struct yapi
  {
    unsigned mesafe:8;
    unsigned  zaman;
    unsigned var:1;
    byte har:1;
    unsigned cins:3;
  };
  struct yapi arac[32];
      cins_tespit(void);
      saatayarla(void);
      dkayarla(void);
      saat1(void);
      #INT_RTCC
```

```
       saat_isr() { // saat interruptı burada başlıyor
         saat1();
       }
#separate
        karsiya_aracgirdi(void)
        {
         output_high(karsiserit_aracvar);
         if(arac[k].cins==1) {
         output_high(karsiserit_cins1);
         output_high(karsiserit_cins2);
         output_high(karsiserit_cins3);}

          if(arac[k].cins==2) {
         output_high(karsiserit_cins1);
         output_high(karsiserit_cins2);
         output_low(karsiserit_cins3);}

          if(arac[k].cins==3) {
         output_high(karsiserit_cins1);
         output_low(karsiserit_cins2);
         output_low(karsiserit_cins3);}

         if (arac[k].cins==4){
         output_low(karsiserit_cins1);
         output_low(karsiserit_cins2);
         output_high(karsiserit_cins3);}

         delay_ms(150);
         output_low(karsiserit_cins1);
         output_low(karsiserit_cins2);
         output_low(karsiserit_cins3);
         output_low(karsiserit_aracvar);
         sayi=sayi+1;
         if(arac[k].zaman<16) arac[k].zaman=16;
         tg=arac[k].zaman-16;
         if(tg<0) tg=0;
         if(input(yesil) && (tg==1) && (q1==0)) tg=0;
         tor=(tor+tg)/sayi;
         top=top+tg;
         if((q1>0) && arac[k].cins==1) q1=q1-13;
         else if((q1>0) && arac[k].cins==2) q1=q1-9;
         else if((q1>0) && arac[k].cins==3) q1=q1-7;
         else if((q1>0) && arac[k].cins==4 ) q1=q1-5;
         else  ;
         if(q1<0) q1=0;

          arac[k].var=0;
          arac[k].har=0;
          arac[k].zaman=0;
          arac[k].mesafe=0;
          arac[k].cins=0;
              }//program sonu
    // 2. yapının karşıya araç girdisi

#separate
       tir(void)
       {
        //önce kuyrukla ilgili kısmı yapıyoruz
        if(arac[k].har==0) {
        if(input(yesil) && !input(karsiserit_dolu)) {
        arac[k].mesafe=arac[k].mesafe+4;
        if(arac[k].mesafe>=200) karsiya_aracgirdi();
         }
          }
        //ilerle buradan itibaren başlıyor
        if((arac[k].har==1) && (q1>0)) {
        if(input(yesil) && !input(karsiserit_dolu))  {
         arac[k].mesafe=arac[k].mesafe+6;
         if(arac[k].mesafe>=(200-q1)) {
                        arac[k].mesafe=(200-q1);
                        arac[k].har=0;
                        q1=q1+13;
                                }
                                  }
```

```
                                    }
          if((arac[k].har==1) && (q1==0)) {
          if(input(yesil) && !input(karsiserit_dolu))  {
          arac[k].mesafe=arac[k].mesafe+6;
          if(arac[k].mesafe>=200) karsiya_aracgirdi();
                                  }
                                    }
                                //ilerle sonu
        //2. if
        if(!input(yesil) || input(karsiserit_dolu)) {
        if((arac[k].har==1) && (q1>=0)) {
                arac[k].mesafe=arac[k].mesafe+6;
                if(arac[k].mesafe>=(200-q1)) {
                arac[k].mesafe=(200-q1);
                q1=q1+13;
                if(q1>200) q1=200;
                arac[k].har=0;
                        }
                          }
                             }//2. if sonu
        }//tir sonu

         //-----------------------
#separate
        otobus(void)
        {
         //önce kuyrukla ilgili kısmı yapıyoruz
         if(arac[k].har==0) {
         if(input(yesil) && !input(karsiserit_dolu)) {
         arac[k].mesafe=arac[k].mesafe+4;
         if(arac[k].mesafe>=200) karsiya_aracgirdi();
          }
           }
         //ilerle buradan itibaren başlıyor
         if((arac[k].har==1) && (q1>0)) {
         if(input(yesil) && !input(karsiserit_dolu))  {
          arac[k].mesafe=arac[k].mesafe+6;
          if(arac[k].mesafe>=(200-q1)) {
                          arac[k].mesafe=(200-q1);
                          arac[k].har=0;
                          q1=q1+9;
                                  }       }
                                    }
          if((arac[k].har==1) && (q1==0)) {
          if(input(yesil) && !input(karsiserit_dolu))  {
          arac[k].mesafe=arac[k].mesafe+6;
          if(arac[k].mesafe>=200) karsiya_aracgirdi();

                                  }
                                    }
                                //ilerle sonu
        //2. if
        if(!input(yesil) || input(karsiserit_dolu)) {
        if((arac[k].har==1) && (q1>=0)) {
                arac[k].mesafe=arac[k].mesafe+6;
                if(arac[k].mesafe>=(200-q1)) {
                arac[k].mesafe=(200-q1);
                q1=q1+9;
                if(q1>200) q1=200;
                arac[k].har=0;
                        }
                          }
                             }//2. if sonu
         }
 #separate
        minibus(void)
        {
        //önce kuyrukla ilgili kısmı yapıyoruz
         if(arac[k].har==0) {
         if(input(yesil) && !input(karsiserit_dolu)) {
         arac[k].mesafe=arac[k].mesafe+4;
         if(arac[k].mesafe>=200) karsiya_aracgirdi();
          }
           }
```

```
        //ilerle buradan itibaren başlıyor
        if((arac[k].har==1) && (q1>0)) {
        if(input(yesil) && !input(karsiserit_dolu))  {
         arac[k].mesafe=arac[k].mesafe+6;
         if(arac[k].mesafe>=(200-q1)) {
                        arac[k].mesafe=(200-q1);
                        arac[k].har=0;
                        q1=q1+7;
                                }
                                  }
                                    }
           if((arac[k].har==1) && (q1==0)) {
           if(input(yesil) && !input(karsiserit_dolu))  {
           arac[k].mesafe=arac[k].mesafe+6;
           if(arac[k].mesafe>=200) karsiya_aracgirdi();

                                }
                                  }
                               //ilerle sonu
       //2. if
       if(!input(yesil) || input(karsiserit_dolu)) {
       if((arac[k].har==1) && (q1>=0)) {
                arac[k].mesafe=arac[k].mesafe+6;
                if(arac[k].mesafe>=(200-q1)) {
                arac[k].mesafe=(200-q1);
                q1=q1+7;
                if(q1>200) q1=200;
                arac[k].har=0;
                        }
                          }
                             }//2. if sonu
        }
#separate
       taksi(void)
       {
       //önce kuyrukla ilgili kısmı yapıyoruz
        if(arac[k].har==0) {
        if(input(yesil) && !input(karsiserit_dolu)) {
        arac[k].mesafe=arac[k].mesafe+4;
        if(arac[k].mesafe>=200) karsiya_aracgirdi();
         }
          }
        //ilerle buradan itibaren başlıyor
        if((arac[k].har==1) && (q1>0)) {
        if(input(yesil) && !input(karsiserit_dolu))  {
         arac[k].mesafe=arac[k].mesafe+6;
         if(arac[k].mesafe>=(200-q1)) {
                        arac[k].mesafe=(200-q1);
                        arac[k].har=0;
                        q1=q1+5;
                                }
                                  }
                                    }
           if((arac[k].har==1) && (q1==0)) {
           if(input(yesil) && !input(karsiserit_dolu))  {
           arac[k].mesafe=arac[k].mesafe+6;
           if(arac[k].mesafe>=200) karsiya_aracgirdi();

                                }
                                  }
                               //ilerle sonu
       //2. if
       if(!input(yesil) || input(karsiserit_dolu)) {
       if((arac[k].har==1) && (q1>=0)) {
                arac[k].mesafe=arac[k].mesafe+6;
                if(arac[k].mesafe>=(200-q1)) {
                arac[k].mesafe=(200-q1);
                q1=q1+5;
                if(q1>200) q1=200;
                arac[k].har=0;
                        }
                          }
                             }//2. if sonu
        }
```

```
#separate
       arac_ilerlet(void) //araç ilerletme kısımı
       {
          for(k=0;k<=31;k++) {
        if(arac[k].var==1) {

            switch(arac[k].cins)
        {
        case 1: tir();    break;
        case 2: otobus(); break;
        case 3: minibus(); break;
        case 4: taksi(); break;
        default: break;
        }
        }
          }
       // yapı sonu
       }
//saat kısmı burada

  void saat1(void)
{
  TM1=TM1+1;
            if(TM1>=30) {//prescalar küçült

                        TM2=TM2+1;
                        if(TM2==2) {//saniye artır
                        TM1=0;
                        TM2=0;
               SN=SN+1;
    if(SN==60)        {//dakika artır
    SN=0;
    SN1=0;
                        DK=DK+1;

                        if(DK==60)            {//saat artır
     DK=0;
                        SA=SA+1;
                        if(SA==24)            SA=0;

    }
    }
    }
  }
}

      // interrupt sonu
 byte cins_tespit(void)//kulvara giren aracın cinsini ogreniyor
 {
  if(input(cins1) && input(cins2) && input(cins3))    return(1);
  if(input(cins1) && input(cins2) && !input(cins3))   return(2);
  if(input(cins1) && !input(cins2) && !input(cins3))  return(3);
  if(!input(cins1) && !input(cins2) && input(cins3))  return(4);
 }
#separate
 main(void)
{
  set_rtcc(0);
  SET_TRIS_B(0X10);
            SET_TRIS_A(0X0F);
            SET_TRIS_C(0X06);
            SET_TRIS_D(0X00);
  SET_TRIS_E(0X00);
  setup_adc_ports(NO_ANALOGS);
  setup_psp(PSP_DISABLED);
  setup_adc(ADC_OFF);
            lcd_init();
  delay_ms(10);
  setup_counters(RTCC_INTERNAL,RTCC_DIV_128);
  enable_interrupts(INT_RTCC);
  enable_interrupts(GLOBAL);
  output_low(onay);//pin_E0
  output_low(pin_B6);
  output_low(pin_B7);
```

```
  output_low(pin_C0);
  output_low(dolu);
  for(j=0;j<=31;j++) {
       arac[j].zaman=0;
       arac[j].mesafe=0;
       arac[j].cins=0;
       arac[j].har=0;
       arac[j].var=0;
             }
             // 2. yapı için
      i=0;
      m=0;

              for(;;)
              {  //for başı
    if(input(aracvar)) {
    araccins=cins_tespit();
    if(((araccins==1) && (q1<=187)) || ((araccins==2) && (q1<=191)) || ((araccins==3) && (q1<=193)) || ((araccins==4) &&
(q1<=195))) {

      if((i<=31) && (arac[i].var==0))  {
      output_low(dolu);
      arac[i].cins=araccins;
      arac[i].var=1;
      arac[i].har=1;
      arac[i].zaman=0;
      arac[i].mesafe=0;
      i=i+1;
      if(i==32) i=0;
      output_high(onay);//50 ms sonra onay verecek
      while(input(aracvar)) ;
      output_low(onay);
      delay_ms(50);
             }
         }
          }
  arac_ilerlet();
  delay_ms(477);
   SN2+=1;
  if(SN2==2) {
   SN2=0;
  SN1=SN1+1;
   for(k=0;k<=21;k++) {
       if(arac[k].var==1) {
       arac[k].zaman=arac[k].zaman+1;}
    }   }
  if(q1>=40) output_high(LOOPBIR);
   else output_low(LOOPBIR);
  if(q1>=85) output_high(LOOPIKI);
  else output_low(LOOPIKI);
  if(q1>=120) output_high(LOOPUC);
  else output_low(LOOPUC);
  if(q1>=160) output_high(LOOPDORT);
  else output_low(LOOPDORT);
  if(q1>=190) output_high(LOOPBES);
  else output_low(LOOPBES);
  if(q1>=200) output_high(dolu);//q1 doluluk kontrolü
  else output_low(dolu);
  delay_ms(10);
  printf(lcd_putc,"\f%U:%U:%U Q1:%U\ns:%lU tg:%lU ",SA,DK,SN,q1,sayi,top);
}// for döngü sonu
}//main sonu
```

EK.2 N_S_kontrolörü.c kaynak kodları

```
#ifdef __PCH__
#device PIC16F877
#endif
#include <16F877.H>
#device  *=16
#fuses XT,NOWDT
#use delay(clock=4000000)
#include <lcd.c>
#define LOOPBES pin_C7
#define LOOPDORT pin_C6
#define LOOPUC pin_C5
#define LOOPIKI pin_C4
#define LOOPBIR pin_B0
#define aracvar pin_A0
#define cins1 pin_A1 //tır=12 m   otobus=8 m
#define cins2 pin_A2  //minibus=6 taksi=4 m
#define cins3 pin_A3
#define onay pin_E0
#define dolu pin_A5 // kulvar doluysa 1 oluyor
#define karsiserit_dolu pin_C2//karşı şerit durumunu alıyor
#define karsiserit_onay pin_C1// karşı şerit onay
#define karsiserit_cins3 pin_C0
#define karsiserit_cins2 pin_B7
#define karsiserit_cins1 pin_B6
#define karsiserit_aracvar pin_B5
#define yesil pin_B4
              byte TM1=0;
              byte  TM2=0;
              byte  DK=0;
  byte bos=1;
  byte alindi1=0;
              unsigned int  SA=0;
              unsigned int SN=0;
  unsigned int q1=0;
  unsigned int i=0;
  unsigned int m=0;
  unsigned int j=1;
  unsigned int t=1;
  unsigned int k=1;
  unsigned int aralik=0;
  long int sayi=0;
  unsigned int ison=1;
  unsigned int araccins;
  unsigned int tor=0;
  unsigned int tg=0;
  long int top=0;
  short int cift=0;
  unsigned int SN1=0;
  unsigned int SN2=0;

  struct yapi
  {
    unsigned mesafe:8;
    unsigned  zaman;
    unsigned var:1;
    byte har:1;
    unsigned cins:3;
  };
  struct yapi arac[22];

      cins_tespit(void);
      saatayarla(void);
      dkayarla(void);
      saat1(void);
```

```
      #INT_RTCC
      saat_isr() { // saat interruptı burada başlıyor
                  saat1();
      }
#separate
       karsiya_aracgirdi(void)
       {
        output_high(karsiserit_aracvar);
        if(arac[k].cins==1) {
        output_high(karsiserit_cins1);
        output_high(karsiserit_cins2);
        output_high(karsiserit_cins3);}

         if(arac[k].cins==2) {
        output_high(karsiserit_cins1);
        output_high(karsiserit_cins2);
        output_low(karsiserit_cins3);}

         if(arac[k].cins==3) {
        output_high(karsiserit_cins1);
        output_low(karsiserit_cins2);
        output_low(karsiserit_cins3);}

        if (arac[k].cins==4){
        output_low(karsiserit_cins1);
        output_low(karsiserit_cins2);
        output_high(karsiserit_cins3);}

        delay_ms(250);
        output_low(karsiserit_cins1);
        output_low(karsiserit_cins2);
        output_low(karsiserit_cins3);
        output_low(karsiserit_aracvar);
        sayi=sayi+1;
        if(arac[k].zaman<8) arac[k].zaman=8;
        tg=arac[k].zaman-8;
        if(tg<0) tg=0;
        if(input(yesil) && (tg==1) && (q1==0)) tg=0;
        tor=(tor+tg)/sayi;
        top=top+tg;
        if((q1>0) && arac[k].cins==1) q1=q1-13;
        else if((q1>0) && arac[k].cins==2) q1=q1-9;
        else if((q1>0) && arac[k].cins==3) q1=q1-7;
        else if((q1>0) && arac[k].cins==4 ) q1=q1-5;
        else  ;
        if(q1<0) q1=0;

         arac[k].var=0;
         arac[k].har=0;
         arac[k].zaman=0;
         arac[k].mesafe=0;
         arac[k].cins=0;
             }//program sonu
    // 2. yapının karşıya araç girdisi
#separate
       tir(void)
       {
        //önce kuyrukla ilgili kısmı yapıyoruz
        if(arac[k].har==0) {
        if(input(yesil) && !input(karsiserit_dolu)) {
        arac[k].mesafe=arac[k].mesafe+4;
        if(arac[k].mesafe>=100) karsiya_aracgirdi();
         }
          }
        //ilerle buradan itibaren başlıyor
        if((arac[k].har==1) && (q1>0)) {
        if(input(yesil) && !input(karsiserit_dolu))  {
         arac[k].mesafe=arac[k].mesafe+6;
         if(arac[k].mesafe>=(100-q1)) {
                     arac[k].mesafe=(100-q1);
                     arac[k].har=0;
                     q1=q1+13;
                                }
                                  }
```

```
                                    }
          if((arac[k].har==1) && (q1==0)) {
          if(input(yesil) && !input(karsiserit_dolu))  {
          arac[k].mesafe=arac[k].mesafe+6;
          if(arac[k].mesafe>=100) karsiya_aracgirdi();
                                  }
                                    }
                                //ilerle sonu
        //2. if
        if(!input(yesil) || input(karsiserit_dolu)) {
        if((arac[k].har==1) && (q1>=0)) {
                arac[k].mesafe=arac[k].mesafe+6;
                if(arac[k].mesafe>=(100-q1)) {
                arac[k].mesafe=(100-q1);
                q1=q1+13;
                if(q1>100) q1=100;
                arac[k].har=0;
                        }
                          }
                             }//2. if sonu

        }//tir sonu

         //-----------------------
#separate
        otobus(void)
        {
         //önce kuyrukla ilgili kısmı yapıyoruz
         if(arac[k].har==0) {
         if(input(yesil) && !input(karsiserit_dolu)) {
         arac[k].mesafe=arac[k].mesafe+4;
         if(arac[k].mesafe>=100) karsiya_aracgirdi();
          }
           }
         //ilerle buradan itibaren başlıyor
         if((arac[k].har==1) && (q1>0)) {
         if(input(yesil) && !input(karsiserit_dolu))  {
          arac[k].mesafe=arac[k].mesafe+6;
          if(arac[k].mesafe>=(100-q1)) {
                          arac[k].mesafe=(100-q1);
                          arac[k].har=0;
                          q1=q1+9;
                                    }
                                      }
                                        }
          if((arac[k].har==1) && (q1==0)) {
          if(input(yesil) && !input(karsiserit_dolu))  {
          arac[k].mesafe=arac[k].mesafe+6;
          if(arac[k].mesafe>=100) karsiya_aracgirdi();

                                  }
                                    }
                                //ilerle sonu
        //2. if
        if(!input(yesil) || input(karsiserit_dolu)) {
        if((arac[k].har==1) && (q1>=0)) {
                arac[k].mesafe=arac[k].mesafe+6;
                if(arac[k].mesafe>=(100-q1)) {
                arac[k].mesafe=(100-q1);
                q1=q1+9;
                if(q1>100) q1=100;
                arac[k].har=0;
                        }
                          }
                             }//2. if sonu
         }
 #separate
        minibus(void)
        {
        //önce kuyrukla ilgili kısmı yapıyoruz
         if(arac[k].har==0) {
         if(input(yesil) && !input(karsiserit_dolu)) {
         arac[k].mesafe=arac[k].mesafe+4;
         if(arac[k].mesafe>=100) karsiya_aracgirdi();
```

```
        }
         }
        //ilerle buradan itibaren başlıyor
        if((arac[k].har==1) && (q1>0)) {
        if(input(yesil) && !input(karsiserit_dolu))  {
         arac[k].mesafe=arac[k].mesafe+6;
         if(arac[k].mesafe>=(100-q1)) {
                        arac[k].mesafe=(100-q1);
                        arac[k].har=0;
                        q1=q1+7;
                                }
                                  }
                                   }
           if((arac[k].har==1) && (q1==0)) {
           if(input(yesil) && !input(karsiserit_dolu))  {
           arac[k].mesafe=arac[k].mesafe+6;
           if(arac[k].mesafe>=100) karsiya_aracgirdi();

                                }
                                  }
                               //ilerle sonu
       //2. if
       if(!input(yesil) || input(karsiserit_dolu)) {
       if((arac[k].har==1) && (q1>=0)) {
                arac[k].mesafe=arac[k].mesafe+6;
                if(arac[k].mesafe>=(100-q1)) {
                arac[k].mesafe=(100-q1);
                q1=q1+7;
                if(q1>100) q1=100;
                arac[k].har=0;
                        }
                          }
                             }//2. if sonu
        }
#separate
       taksi(void)
       {
       //önce kuyrukla ilgili kısmı yapıyoruz
        if(arac[k].har==0) {
        if(input(yesil) && !input(karsiserit_dolu)) {
        arac[k].mesafe=arac[k].mesafe+4;
        if(arac[k].mesafe>=100) karsiya_aracgirdi();
         }
          }
        //ilerle buradan itibaren başlıyor
        if((arac[k].har==1) && (q1>0)) {
        if(input(yesil) && !input(karsiserit_dolu))  {
         arac[k].mesafe=arac[k].mesafe+6;
         if(arac[k].mesafe>=(100-q1)) {
                        arac[k].mesafe=(100-q1);
                        arac[k].har=0;
                        q1=q1+5;
                                }
                                  }
                                   }
           if((arac[k].har==1) && (q1==0)) {
           if(input(yesil) && !input(karsiserit_dolu))  {
           arac[k].mesafe=arac[k].mesafe+6;
           if(arac[k].mesafe>=100) karsiya_aracgirdi();

                                }
                                  }
                               //ilerle sonu
       //2. if
       if(!input(yesil) || input(karsiserit_dolu)) {
       if((arac[k].har==1) && (q1>=0)) {
                arac[k].mesafe=arac[k].mesafe+6;
                if(arac[k].mesafe>=(100-q1)) {
                arac[k].mesafe=(100-q1);
                q1=q1+5;
                if(q1>100) q1=100;
                arac[k].har=0;
                        }
                          }
```

```
                        }//2. if sonu
        }


        // buradan itibaren yapının ikinci kısmı var

#separate
       arac_ilerlet(void) //araç ilerletme kısımı
       {
          for(k=0;k<=21;k++) {
        if(arac[k].var==1) {

            switch(arac[k].cins)
        {
        case 1: tir();    break;
        case 2: otobus(); break;
        case 3: minibus(); break;
        case 4: taksi(); break;
        default: break;
        }
        }
          }

        // yapı sonu
       }
//saat kısmı burada
    void saat1(void)
{
  TM1=TM1+1;
            if(TM1>=30) {//prescalar küçült


                        TM2=TM2+1;

                        if(TM2==2) {//saniye artır
                        TM1=0;
                        TM2=0;
                SN=SN+1;
     if(SN==60)         {//dakika artır
     SN=0;
     SN1=0;

                        DK=DK+1;

                        if(DK==60)              {//saat artır
      DK=0;
                        SA=SA+1;
                        if(SA==24)              SA=0;

     }
     }
     }
  }
}

       // interrupt sonu

//************************************

  byte cins_tespit(void)//kulvara giren aracın cinsini ogreniyor
  {
   if(input(cins1) && input(cins2) && input(cins3))    return(1);
   if(input(cins1) && input(cins2) && !input(cins3))   return(2);
   if(input(cins1) && !input(cins2) && !input(cins3))  return(3);
   if(!input(cins1) && !input(cins2) && input(cins3))  return(4);
  }

#separate
 main(void)
{
  set_rtcc(0);
  SET_TRIS_B(0X10);
            SET_TRIS_A(0X0F);
            SET_TRIS_C(0X06);
```

```
              SET_TRIS_D(0X00);
  SET_TRIS_E(0X00);
              delay_ms(50);
              lcd_init();
  setup_counters(RTCC_INTERNAL,RTCC_DIV_128);
  enable_interrupts(INT_RTCC);
  enable_interrupts(GLOBAL);
  output_low(onay);//pin_E0
  output_low(pin_B6);
  output_low(pin_B7);
  output_low(pin_C0);
  output_low(dolu);
  for(j=0;j<=21;j++) {
       arac[j].zaman=0;
       arac[j].mesafe=0;
       arac[j].cins=0;
       arac[j].har=0;
       arac[j].var=0;
            }
      i=0;
              for(;;)
              {  //for başı

     if(input(aracvar)) {
     araccins=cins_tespit();
     if(((araccins==1) && (q1<=87)) || ((araccins==2) && (q1<=91)) || ((araccins==3) && (q1<=93)) || ((araccins==4) &&
(q1<=95))) {
      alindi1=0;
      bos=0;
       if((i<=21) && (arac[i].var==0))  {
       output_low(dolu);
       arac[i].cins=araccins;
       arac[i].var=1;
       arac[i].har=1;
       arac[i].zaman=0;
       arac[i].mesafe=0;
       alindi1=1;
       i=i+1;
       if(i==22) i=0;

       output_high(onay);//50 ms sonra onay verecek
       while(input(aracvar)) ;
       output_low(onay);
       delay_ms(50);
              }
          }
           }
  arac_ilerlet();
  delay_ms(487);
  SN2+=1;
  if(SN2==2) {
   SN2=0;
  SN1=SN1+1;
   for(k=0;k<=21;k++) {
       if(arac[k].var==1) {
       arac[k].zaman=arac[k].zaman+1;}
     }
   }
  if(q1>=10) output_high(LOOPBIR);
   else output_low(LOOPBIR);
  if(q1>=30) output_high(LOOPIKI);
  else output_low(LOOPIKI);
  if(q1>=50) output_high(LOOPUC);
  else output_low(LOOPUC);
  if(q1>=90) output_high(LOOPDORT);
  else output_low(LOOPDORT);
  if(q1>=100) output_high(LOOPBES);
  else output_low(LOOPBES);
  if(q1>=100) output_high(dolu);//q1 doluluk kontrolü
  else output_low(dolu);
  delay_ms(10);
  printf(lcd_putc,"\f%U:%U:%U Q1:%U\ns:%lU tg:%lU ",SA,DK,SN,q1,sayi,top);
}// for döngü sonu
}//main sonu
```

EK.3 E_üretici.c kaynal kodları

```
#ifdef __PCH__
#device PIC16F877
#endif

#include <16F877.H>
#device  *=16
#fuses XT,NOWDT,NOPROTECT,NOLVP
#use delay(clock=4000000)
#define dolu pin_D2// kulvar doluysa 1 oluyor
#define onay pin_D3// karşı şerit onay
#define cins3 pin_D4//tır=12 m   otobus=8 m
#define cins2 pin_D5//minibus=6 taksi=4 m
#define cins1 pin_D6
#define aracvar pin_D7
#define in1 pin_B6
#define in2 pin_B7
              unsigned int TM1=0;
              unsigned int TM2=0;
              unsigned int DK=0;
              unsigned int  SA=0;
              unsigned int SN=0;
  unsigned int q1=0;
  unsigned int q2=0;
  unsigned int i=1;
  unsigned int j=1;
      saatayarla(void);
      dkayarla(void);
      saat1(void);


 unsigned cins_tespit(void)//kulvara giren aracın cinsini ogreniyor
  {
  if(cins1 && cins2 && cins3)    return(1);
  if(cins1 && cins2 && !cins3)   return(2);
  if(cins1 && !cins2 && !cins3)  return(3);
  if(!cins1 && !cins2 && cins3)  return(4);
  }

void tir(void) {
while(input(dolu)) output_low(aracvar);
 do {

  output_high(cins1);
  output_high(cins2);
  output_high(cins3);
  output_high(aracvar);
  }while(input(onay)==0) ;

  output_low(aracvar);
  output_low(cins1);
  output_low(cins2);
  output_low(cins3);

}

otobus(void) {
  while(input(dolu)) output_low(aracvar);
  do {

  output_high(cins1);
  output_high(cins2);
  output_low(cins3);
  output_high(aracvar);
     } while(input(onay)==0)  ;
  output_low(aracvar);
  output_low(cins1);
```

```
 output_low(cins2);
 output_low(cins3);
 }

minibus(void) {
 while(input(dolu)) output_low(aracvar);
  do {

 output_high(cins1);
 output_low(cins2);
 output_low(cins3);
 output_high(aracvar);
    } while(input(onay)==0) ;
 output_low(aracvar);
 output_low(cins1);
 output_low(cins2);
 output_low(cins3);

 }

taksi(void) {
 while(input(dolu)) output_low(aracvar);
  do {

 output_low(cins1);
 output_low(cins2);
 output_high(cins3);
 output_high(aracvar);
  } while(input(onay)==0) ;
 output_low(aracvar);
 output_low(cins1);
 output_low(cins2);
 output_low(cins3);

}

void sabah(void) {
 taksi();
 delay_ms(1500);
 taksi();
 delay_ms(1500);
 taksi();
 delay_ms(1500);
 minibus();
 delay_ms(1950);
 minibus();
 delay_ms(1950);
 taksi();
 delay_ms(1500);
 otobus();
 delay_ms(2150);
 taksi();
 delay_ms(1500);
 minibus();
 delay_ms(1950);
 minibus();
 delay_ms(1950);
 tir();
 delay_ms(3300);
 taksi();
 delay_ms(1500);
 otobus();
 delay_ms(2150);
 taksi();
 delay_ms(1500);
 minibus();
 delay_ms(1950);
 minibus();
 delay_ms(1950);
 minibus();
 delay_ms(1950);
 taksi();
 delay_ms(1500);
 otobus();
```

```
 delay_ms(2150);
 taksi();
 delay_ms(1500);
 tir();
 delay_ms(3300);
 minibus();
 delay_ms(1950);
 minibus();
 delay_ms(1950);
 otobus();
 delay_ms(2150);
 otobus();
 delay_ms(2150);
 taksi();
 delay_ms(1500);
 taksi();
 delay_ms(1500);
 taksi();
 delay_ms(1500);
 taksi();
 delay_ms(1500);
 otobus();
 delay_ms(2150);

}

void oglen(void) {
 tir();
 delay_ms(4400);
 taksi();
 delay_ms(2000);
 minibus();
 delay_ms(2600);
 minibus();
 delay_ms(3600);
 taksi();
 delay_ms(2000);
 tir();
 delay_ms(4400);
 minibus();
 delay_ms(2600);
 minibus();
 delay_ms(2600);
 minibus();
 delay_ms(2600);
 taksi();
 delay_ms(2000);
 taksi();
 delay_ms(2000);
 tir();
 delay_ms(4400);
 otobus();
 delay_ms(3000);
 minibus();
 delay_ms(2600);
 minibus();
 delay_ms(2600);
 taksi();
 delay_ms(2000);
 otobus();
 delay_ms(3000);
 minibus();
 delay_ms(3600);
 tir();
 delay_ms(4400);
 taksi();
 delay_ms(2000);
 taksi();
 delay_ms(2000);
 taksi();
 delay_ms(3000);
 minibus();
 delay_ms(2600);
 taksi();
```

```
 delay_ms(2000);
 minibus();
 delay_ms(2600);
 minibus();
 delay_ms(2600);
 taksi();
 delay_ms(2000);
 minibus();
 delay_ms(3600);
 taksi();
 delay_ms(2000);
 tir();
 delay_ms(4400);
}
void aksam(void) {
 otobus();
  delay_ms(2250);
  otobus();
  delay_ms(2750);
  minibus();
  delay_ms(1850);
  taksi();
  delay_ms(1500);
  minibus();
  delay_ms(1850);
  minibus();
  delay_ms(1650);
  otobus();
  delay_ms(2250);
  otobus();
  delay_ms(2250);
  taksi();
  delay_ms(1500);
  taksi();
  delay_ms(1500);
  tir();
  delay_ms(3100);
  minibus();
  delay_ms(1850);
  otobus();
  delay_ms(2250);
  taksi();
  delay_ms(1500);
  otobus();
  delay_ms(2750);
  minibus();
  delay_ms(1850);
  taksi();
  delay_ms(1500);
  taksi();
  delay_ms(1500);
  taksi();
  delay_ms(1500);
  otobus();
  delay_ms(2250);
  tir();
  delay_ms(3100);
  minibus();
  delay_ms(1850);
  minibus();
  delay_ms(1850);
  otobus();
  delay_ms(1750);
  otobus();
  delay_ms(2250);
  taksi();
  delay_ms(1500);
  taksi();
  delay_ms(1500);
  taksi();
  delay_ms(1500);
  otobus();
  delay_ms(2250);
  minibus();
```

```
 delay_ms(1850);
}
 main(void)
{
   set_rtcc(0);
   SET_TRIS_B(0XC0);
                SET_TRIS_A(0X0F);
                SET_TRIS_C(0X04);
                SET_TRIS_D(0X0C);
                delay_ms(50);
                disable_interrupts(INT_RTCC);
   disable_interrupts(GLOBAL);
   setup_adc_ports(NO_ANALOGS);
   setup_psp(PSP_DISABLED);
   setup_adc(ADC_OFF);
                for(;;)
                {  //for başı
    if(!input(in1) && !input(in2)) sabah();// sabah programı
    else if(!input(in1) && input(in2)) oglen();
    else aksam();
   }//for sonu
}//main sonu
```

EK.4 W_üretici.c kaynak kodları

```
#ifdef __PCH__
#device PIC16F877
#endif
#include <16F877.H>
#device  *=16
#fuses XT,NOWDT,NOPROTECT,NOLVP
#use delay(clock=4000000)
#define dolu pin_D2// kulvar doluysa 1 oluyor
#define onay pin_D3// karşı şerit onay
#define cins3 pin_D4//tır=12 m   otobus=8 m
#define cins2 pin_D5//minibus=6 taksi=4 m
#define cins1 pin_D6
#define aracvar pin_D7
#define in1 pin_B6
#define in2 pin_B7
                unsigned int TM1=0;
                unsigned int TM2=0;
                unsigned int DK=0;
                unsigned int  SA=0;
                unsigned int SN=0;
  unsigned int q1=0;
  unsigned int q2=0;
  unsigned int i=1;
  unsigned int j=1;
 unsigned cins_tespit(void)//kulvara giren aracın cinsini ogreniyor
  {
  if(cins1 && cins2 && cins3)    return(1);
  if(cins1 && cins2 && !cins3)   return(2);
  if(cins1 && !cins2 && !cins3)  return(3);
  if(!cins1 && !cins2 && cins3)  return(4);
  }

void tir(void) {
while(input(dolu)) output_low(aracvar);
 do {

  output_high(cins1);
  output_high(cins2);
  output_high(cins3);
  output_high(aracvar);
  }while(input(onay)==0) ;

  output_low(aracvar);
  output_low(cins1);
  output_low(cins2);
  output_low(cins3);
}

otobus(void) {
  while(input(dolu)) output_low(aracvar);
  do {

  output_high(cins1);
  output_high(cins2);
  output_low(cins3);
  output_high(aracvar);
     } while(input(onay)==0)  ;
  output_low(aracvar);
  output_low(cins1);
  output_low(cins2);
  output_low(cins3);
  }
minibus(void) {
 while(input(dolu)) output_low(aracvar);
```

```
  do {
  output_high(cins1);
  output_low(cins2);
  output_low(cins3);
  output_high(aracvar);
     } while(input(onay)==0) ;
  output_low(aracvar);
  output_low(cins1);
  output_low(cins2);
  output_low(cins3);

  }
taksi(void) {
  while(input(dolu)) output_low(aracvar);
   do {

  output_low(cins1);
  output_low(cins2);
  output_high(cins3);
  output_high(aracvar);
   } while(input(onay)==0) ;
  output_low(aracvar);
  output_low(cins1);
  output_low(cins2);
  output_low(cins3);
}
void sabah(void) {
  otobus();
  delay_ms(1200);
  otobus();
  delay_ms(1200);
  minibus();
  delay_ms(1000);
  taksi();
  delay_ms(650);
  minibus();
  delay_ms(800);
  minibus();
  delay_ms(1000);
  otobus();
  delay_ms(1400);
  otobus();
  delay_ms(1400);
  taksi();
  delay_ms(650);
  taksi();
  delay_ms(1000);
  tir();
  delay_ms(2300);
  minibus();
  delay_ms(1300);
  otobus();
  delay_ms(1400);
  taksi();
  delay_ms(650);
  otobus();
  delay_ms(1900);
  minibus();
  delay_ms(800);
  taksi();
  delay_ms(1250);
  taksi();
  delay_ms(650);
  taksi();
  delay_ms(650);
  otobus();
  delay_ms(1400);
  tir();
  delay_ms(2300);
  minibus();
  delay_ms(1200);
  minibus();
  delay_ms(1200);
  otobus();
```

```
 delay_ms(1900);
 otobus();
 delay_ms(900);
 taksi();
 delay_ms(1000);
 taksi();
 delay_ms(650);
 taksi();
 delay_ms(650);
 otobus();
 delay_ms(1700);
 minibus();
 delay_ms(800);

}

void oglen(void) {
 tir();
 delay_ms(4300);
 taksi();
 delay_ms(2500);
 minibus();
 delay_ms(1950);
 minibus();
 delay_ms(1950);
 taksi();
 delay_ms(1500);
 tir();
 delay_ms(4300);
 minibus();
 delay_ms(2950);
 minibus();
 delay_ms(1950);
 minibus();
 delay_ms(2950);
 taksi();
 delay_ms(1500);
 taksi();
 delay_ms(2500);
 tir();
 delay_ms(4300);
 otobus();
 delay_ms(3150);
 minibus();
 delay_ms(2950);
 minibus();
 delay_ms(1950);
 taksi();
 delay_ms(2500);
 otobus();
 delay_ms(3150);
 minibus();
 delay_ms(2950);
 tir();
 delay_ms(4300);
 taksi();
 delay_ms(2500);
 taksi();
 delay_ms(2500);
 taksi();
 delay_ms(2500);
 minibus();
 delay_ms(2950);
 taksi();
 delay_ms(2500);
 minibus();
 delay_ms(1950);
 minibus();
 delay_ms(2950);
 taksi();
 delay_ms(2500);
 minibus();
 delay_ms(1950);
 taksi();
```

```
 delay_ms(1500);
 tir();
 delay_ms(4300);
 }

 void aksam(void) {
 taksi();
 delay_ms(1500);
 taksi();
 delay_ms(1500);
 taksi();
 delay_ms(1500);
 minibus();
 delay_ms(1950);
 minibus();
 delay_ms(1950);
 taksi();
 delay_ms(1500);
 otobus();
 delay_ms(2150);
 taksi();
 delay_ms(1500);
 minibus();
 delay_ms(1950);
 minibus();
 delay_ms(1950);
 tir();
 delay_ms(3300);
 taksi();
 delay_ms(1500);
 otobus();
 delay_ms(2150);
 taksi();
 delay_ms(1500);
 minibus();
 delay_ms(1950);
 minibus();
 delay_ms(1950);
 minibus();
 delay_ms(1950);
 taksi();
 delay_ms(1500);
 otobus();
 delay_ms(2150);
 taksi();
 delay_ms(1500);
 tir();
 delay_ms(3300);
 minibus();
 delay_ms(1950);
 minibus();
 delay_ms(1950);
 otobus();
 delay_ms(2150);
 otobus();
 delay_ms(2150);
 taksi();
 delay_ms(1500);
 taksi();
 delay_ms(1500);
 taksi();
 delay_ms(1500);
 taksi();
 delay_ms(1500);
 otobus();
 delay_ms(2150);
 }
 main(void)
 {
   set_rtcc(0);
   SET_TRIS_B(0XC0);
              SET_TRIS_A(0X0F);
              SET_TRIS_C(0X04);
              SET_TRIS_D(0X0C);
```

```
            delay_ms(50);
            disable_interrupts(INT_RTCC);
  disable_interrupts(GLOBAL);
  setup_adc_ports(NO_ANALOGS);
  setup_psp(PSP_DISABLED);
  setup_adc(ADC_OFF);
            for(;;)
            {  //for başı
   if(!input(in1) && !input(in2)) sabah();// sabah programı
   else if(!input(in1) && input(in2)) oglen();
   else aksam();



  }//for sonu
}//main sonu
```

EK.5 N_üreteci.c kaynak kodları

```
#ifdef __PCH__
#device PIC16F877
#endif
#include <16F877.H>
#device  *=16
#fuses XT,NOWDT,NOPROTECT,NOLVP
#use delay(clock=4000000)
#define dolu pin_D2// kulvar doluysa 1 oluyor
#define onay pin_D3// karşı şerit onay
#define cins3 pin_D4//tır=12 m   otobus=8 m
#define cins2 pin_D5//minibus=6 taksi=4 m
#define cins1 pin_D6
#define aracvar pin_D7
#define in1 pin_B6
#define in2 pin_B7
                unsigned int TM1=0;
                unsigned int TM2=0;
                unsigned int DK=0;
                unsigned int  SA=0;
                unsigned int SN=0;
  unsigned int q1=0;
  unsigned int q2=0;
  unsigned int i=1;
  unsigned int j=1;
      saatayarla(void);
      dkayarla(void);
      saat1(void);
 unsigned cins_tespit(void)//kulvara giren aracın cinsini ogreniyor
  {
   if(cins1 && cins2 && cins3)    return(1);
   if(cins1 && cins2 && !cins3)   return(2);
   if(cins1 && !cins2 && !cins3)  return(3);
   if(!cins1 && !cins2 && cins3)  return(4);
  }
void tir(void) {
while(input(dolu)) output_low(aracvar);
 do {

  output_high(cins1);
  output_high(cins2);
  output_high(cins3);
  output_high(aracvar);
  }while(input(onay)==0) ;
  output_low(aracvar);
  output_low(cins1);
  output_low(cins2);
  output_low(cins3);
}
otobus(void) {
  while(input(dolu)) output_low(aracvar);
  do {

  output_high(cins1);
  output_high(cins2);
  output_low(cins3);
  output_high(aracvar);
     } while(input(onay)==0)  ;
  output_low(aracvar);
  output_low(cins1);
  output_low(cins2);
  output_low(cins3);
  }
minibus(void) {
 while(input(dolu)) output_low(aracvar);
   do {
```

```
 output_high(cins1);
 output_low(cins2);
 output_low(cins3);
 output_high(aracvar);
    } while(input(onay)==0) ;
 output_low(aracvar);
 output_low(cins1);
 output_low(cins2);
 output_low(cins3);
 }

taksi(void) {
 while(input(dolu)) output_low(aracvar);
  do {

 output_low(cins1);
 output_low(cins2);
 output_high(cins3);
 output_high(aracvar);
  } while(input(onay)==0) ;
 output_low(aracvar);
 output_low(cins1);
 output_low(cins2);
 output_low(cins3);
}
void sabah(void) {
 taksi();
 delay_ms(4000);
 otobus();
 delay_ms(5000);
 minibus();
 delay_ms(5700);
 otobus();
 delay_ms(4000);
 minibus();
 delay_ms(5700);
 taksi();
 delay_ms(5500);
 taksi();
 delay_ms(4500);
 minibus();
 delay_ms(5700);
 taksi();
 delay_ms(5500);
 tir();
 delay_ms(6400);
 minibus();
 delay_ms(5700);
 taksi();
 delay_ms(2500);
 otobus();
 delay_ms(6000);
 otobus();
 delay_ms(6000);
 taksi();
 delay_ms(2000);
 minibus();
 delay_ms(7700);
 tir();
 delay_ms(8400);
 otobus();
 delay_ms(2000);
 taksi();
 delay_ms(7500);
 taksi();
 delay_ms(1500);
 taksi();
 delay_ms(2500);
 minibus();
 delay_ms(6700);
 minibus();
 delay_ms(5700);
 taksi();
 delay_ms(3500);
```

```
 tir();
 delay_ms(4300);
 otobus();
 delay_ms(4000);
 otobus();
 delay_ms(3000);
 taksi();
 delay_ms(5500);
 taksi();
 delay_ms(3500);
 minibus();
 delay_ms(6700);
}

void oglen(void) {
 tir();
 delay_ms(1500);
 taksi();
 delay_ms(700);
 minibus();
 delay_ms(1000);
 minibus();
 delay_ms(1000);
 taksi();
 delay_ms(700);
 tir();
 delay_ms(1500);
 minibus();
 delay_ms(900);
 minibus();
 delay_ms(900);
 minibus();
 delay_ms(1000);
 taksi();
 delay_ms(700);
 taksi();
 delay_ms(1700);
 tir();
 delay_ms(1500);
 otobus();
 delay_ms(1250);
 minibus();
 delay_ms(1000);
 minibus();
 delay_ms(1000);
 taksi();
 delay_ms(700);
 otobus();
 delay_ms(1250);
 minibus();
 delay_ms(2000);
 tir();
 delay_ms(1500);
 taksi();
 delay_ms(700);
 taksi();
 delay_ms(700);
 taksi();
 delay_ms(1500);
 minibus();
 delay_ms(1000);
 taksi();
 delay_ms(700);
 minibus();
 delay_ms(1000);
 minibus();
 delay_ms(900);
 taksi();
 delay_ms(700);
 minibus();
 delay_ms(2000);
 taksi();
 delay_ms(900);
 tir();
```

```
 delay_ms(2500);
}
void aksam(void) {
 tir();
 delay_ms(3000);
 otobus();
 delay_ms(2500);
 minibus();
 delay_ms(2000);
 otobus();
 delay_ms(2500);
 tir();
 delay_ms(3000);
 taksi();
 delay_ms(1700);
 taksi();
 delay_ms(1700);
 minibus();
 delay_ms(2000);
 taksi();
 delay_ms(1700);
 tir();
 delay_ms(3000);
 minibus();
 delay_ms(2600);
 taksi();
 delay_ms(1700);
 otobus();
 delay_ms(2500);
 otobus();
 delay_ms(2500);
 tir();
 delay_ms(3000);
 minibus();
 delay_ms(2000);
 tir();
 delay_ms(3000);
 otobus();
 delay_ms(3000);
 taksi();
 delay_ms(1700);
 taksi();
 delay_ms(2500);
 taksi();
 delay_ms(1700);
 minibus();
 delay_ms(2000);
 minibus();
 delay_ms(2000);
 taksi();
 delay_ms(1700);
 tir();
 delay_ms(3300);
 otobus();
 delay_ms(2500);
 otobus();
 delay_ms(2500);
 taksi();
 delay_ms(1700);
 taksi();
 delay_ms(1700);
 tir();
 delay_ms(3000);
}
 main(void)
{
  set_rtcc(0);
  SET_TRIS_B(0XC0);
              SET_TRIS_A(0X0F);
              SET_TRIS_C(0X04);
              SET_TRIS_D(0X0C);
              delay_ms(50);
              disable_interrupts(INT_RTCC);
  disable_interrupts(GLOBAL);
```

```
  setup_adc_ports(NO_ANALOGS);
  setup_psp(PSP_DISABLED);
  setup_adc(ADC_OFF);
              for(;;)
              {  //for başı
   if(!input(in1) && !input(in2)) sabah();// sabah programı
   else if(!input(in1) && input(in2)) oglen();
   else aksam();
  }//for sonu
}//main sonu
```

EK.6 S.üretici.c kaynak kodları

```
#ifdef __PCH__
#device PIC16F877
#endif
#include <16F877.H>
#device  *=16
#fuses XT,NOWDT,NOPROTECT,NOLVP
#use delay(clock=4000000)
#define dolu pin_D2// kulvar doluysa 1 oluyor
#define onay pin_D3// karşı şerit onay
#define cins3 pin_D4//tır=12 m   otobus=8 m
#define cins2 pin_D5//minibus=6 taksi=4 m
#define cins1 pin_D6
#define aracvar pin_D7
#define in1 pin_B6
#define in2 pin_B7
                unsigned int TM1=0;
                unsigned int TM2=0;
                unsigned int DK=0;
                unsigned int  SA=0;
                unsigned int SN=0;
  unsigned int q1=0;
  unsigned int q2=0;
  unsigned int i=1;
  unsigned int j=1;
      saatayarla(void);
      dkayarla(void);
      saat1(void);
 unsigned cins_tespit(void)//kulvara giren aracın cinsini ogreniyor
  {
  if(cins1 && cins2 && cins3)    return(1);
  if(cins1 && cins2 && !cins3)   return(2);
  if(cins1 && !cins2 && !cins3)  return(3);
  if(!cins1 && !cins2 && cins3)  return(4);
  }
void tir(void) {
while(input(dolu)) output_low(aracvar);
 do {
  output_high(cins1);
  output_high(cins2);
  output_high(cins3);
  output_high(aracvar);
  }while(input(onay)==0) ;

  output_low(aracvar);
  output_low(cins1);
  output_low(cins2);
  output_low(cins3);

}

otobus(void) {
  while(input(dolu)) output_low(aracvar);
  do {
  output_high(cins1);
  output_high(cins2);
  output_low(cins3);
  output_high(aracvar);
     } while(input(onay)==0)  ;
  output_low(aracvar);
  output_low(cins1);
  output_low(cins2);
  output_low(cins3);
  }
minibus(void) {
 while(input(dolu)) output_low(aracvar);
   do {
```

```
 output_high(cins1);
 output_low(cins2);
 output_low(cins3);
 output_high(aracvar);
    } while(input(onay)==0) ;
 output_low(aracvar);
 output_low(cins1);
 output_low(cins2);
 output_low(cins3);
 }
taksi(void) {
 while(input(dolu)) output_low(aracvar);
  do {

 output_low(cins1);
 output_low(cins2);
 output_high(cins3);
 output_high(aracvar);
  } while(input(onay)==0) ;
 output_low(aracvar);
 output_low(cins1);
 output_low(cins2);
 output_low(cins3);
}
void sabah(void) {
 taksi();
delay_ms(5000);
 taksi();
 delay_ms(4500);
 minibus();
 delay_ms(4700);
 tir();
 delay_ms(5800);
 minibus();
 delay_ms(5700);
 minibus();
 delay_ms(4900);
  tir();
 delay_ms(6400);
 otobus();
 delay_ms(6400);
 taksi();
 delay_ms(4500);
 taksi();
 delay_ms(6500);
 taksi();
 delay_ms(6500);
 minibus();
 delay_ms(4700);
  minibus();
 delay_ms(5700);
 taksi();
 delay_ms(5600);
 tir();
 delay_ms(3300);
 otobus();
 delay_ms(2000);
 taksi();
 delay_ms(4500);
 minibus();
 delay_ms(3700);
  taksi();
 delay_ms(3800);
 otobus();
 delay_ms(5000);
  otobus();
 delay_ms(4500);
 taksi();
 delay_ms(4500);
 taksi();
 delay_ms(4500);
 minibus();
 delay_ms(5700);
 taksi();
```

```
 delay_ms(4800);
 otobus();
 delay_ms(4300);
 minibus();
 delay_ms(4900);
 otobus();
 delay_ms(4400);
  taksi();
 delay_ms(4500);
 taksi();
 delay_ms(3700);
  otobus();
 delay_ms(4400);
}
void oglen(void) {
 tir();
 delay_ms(2600);
 taksi();
 delay_ms(1700);
 minibus();
 delay_ms(2000);
 minibus();
 delay_ms(2000);
 taksi();
 delay_ms(700);
 tir();
 delay_ms(2000);
 minibus();
 delay_ms(2000);
 minibus();
 delay_ms(1000);
 minibus();
 delay_ms(2000);
 taksi();
 delay_ms(1700);
 taksi();
 delay_ms(1700);
 tir();
 delay_ms(2000);
 otobus();
 delay_ms(1200);
 minibus();
 delay_ms(1000);
 minibus();
 delay_ms(2000);
 taksi();
 delay_ms(700);
 otobus();
 delay_ms(2200);
 minibus();
 delay_ms(1000);
 tir();
 delay_ms(1600);
 taksi();
 delay_ms(1700);
 taksi();
 delay_ms(1700);
 taksi();
 delay_ms(1700);
 minibus();
 delay_ms(2000);
 taksi();
 delay_ms(700);
 minibus();
 delay_ms(1000);
 minibus();
 delay_ms(1000);
 taksi();
 delay_ms(700);
 minibus();
 delay_ms(1000);
 taksi();
 delay_ms(1700);
 tir();
```

```
 delay_ms(2100);
}

void aksam(void) {
 tir();
 delay_ms(3000);
 otobus();
 delay_ms(2500);
 minibus();
 delay_ms(2000);
 otobus();
 delay_ms(2500);
 tir();
 delay_ms(3000);
 taksi();
 delay_ms(2000);
 taksi();
 delay_ms(2500);
 minibus();
 delay_ms(2000);
 taksi();
 delay_ms(2000);
 tir();
 delay_ms(3000);
 minibus();
 delay_ms(2600);
 taksi();
 delay_ms(2000);
 otobus();
 delay_ms(2500);
 otobus();
 delay_ms(2500);
 tir();
 delay_ms(3000);
 minibus();
 delay_ms(2600);
 tir();
 delay_ms(5500);
 otobus();
 delay_ms(2500);
 taksi();
 delay_ms(2500);
 taksi();
 delay_ms(2000);
 taksi();
 delay_ms(2000);
 minibus();
 delay_ms(2600);
 minibus();
 delay_ms(2600);
 taksi();
 delay_ms(2000);
 tir();
 delay_ms(3300);
 otobus();
 delay_ms(2500);
 otobus();
 delay_ms(3000);
 taksi();
 delay_ms(2000);
 taksi();
 delay_ms(2000);
 tir();
 delay_ms(3000);
}
 main(void)
{
  set_rtcc(0);
  SET_TRIS_B(0XC0);
            SET_TRIS_A(0X0F);
            SET_TRIS_C(0X04);
            SET_TRIS_D(0X0C);
            delay_ms(50);
            disable_interrupts(INT_RTCC);
```

```
  disable_interrupts(GLOBAL);
  setup_adc_ports(NO_ANALOGS);
  setup_psp(PSP_DISABLED);
  setup_adc(ADC_OFF);
              for(;;)
              {  //for başı
   if(!input(in1) && !input(in2)) sabah();// sabah programı
   else if(!input(in1) && input(in2)) oglen();
   else aksam();



  }//for sonu
}//main sonu
```

EK.7 Fuzzy_logic_yeşil.c kaynak kodları

```
#ifdef __PCH__
#device PIC16F877
#endif
#include <16F877.H>
#device  *=16
#fuses XT,NOWDT
#use delay(clock=4000000)
#include <stdlib.h>
#include <math.h>
#fuses XT,NOWDT,NOPROTECT,NOLVP
#include <lcd.c>
#define q15 pin_C7
#define q14 pin_C6
#define q13 pin_C5
#define q12 pin_C4
#define q11 pin_C3
#define fuzzy pin_E0
#define q12k pin_A5 // kulvar doluysa 1 oluyor
#define q12s pin_A3
#define q12y pin_A2
#define yonay pin_A0
#define ykabul pin_A1
#define q25 pin_C2//karşı şerit durumunu alıyor
#define q24 pin_C1// karşı şerit onay
#define q23 pin_C0
#define q22 pin_B7
#define q21 pin_B6
#define q31 pin_B5
#define q32 pin_B4
#define q33 pin_B3
#define q41 pin_B2
#define q42 pin_B1
#define q43 pin_B0
              byte TM1=0;
              byte  TM2=0;
              byte  DK=0;
              unsigned int  SA=0;
              unsigned int SN=0;
  unsigned int q1=0;
  unsigned int q2=0;
  unsigned int q3=0;
  unsigned int q4=0;
  unsigned int i=1;
  unsigned int j=1;
  unsigned int t=1;
  unsigned int k=1;
  unsigned int maxq12;
  unsigned int maxq34;

  byte tq12=0;
  byte tq13=0;
  byte tq14=0;
  byte tq15=0;
  byte tq22=0;
  byte tq23=0;
  byte tq24=0;
  byte tq25=0;
  byte tq31=0;
  byte tq32=0;
  byte tq33=0;
  byte tq41=0;
  byte tq42=0;
  byte tq43=0;
  float x1=0;
  float x2=0;
  float y1=0;
```

```
  float y2=0;
  float maxs1=0;
  float maxs2=0;
  float max=0;
  float y;
  float x;
  float z1=0;
  float z2=0;
  float z3=0;
  float z4=0;
  float z5=0;
  float z6=0;
  float z7=0;
  float z8=0;
  float z9=0;
  float pay;
  float payda;
  unsigned int ty1=20;
  unsigned int ty2=0;
  byte tk=0;
  byte ty3=0;
  unsigned int uzatma;
  unsigned int dongu=0;
      loop_kontrol(void);
      cins_tespit(void);
      saatayarla(void);
      dkayarla(void);
      kuyruk_test(void);
      saat1(void);
      #INT_RTCC
      saat_isr() { // saat interruptı burada başlıyor


                  saat1();

      }

//saat kısmı burada
 void zaman_test(void) {  //kuyruk zamanlama ile ilgili kısım
 tq12=tq12+1;
 tq13=tq13+1;
 tq14=tq14+1;
 tq15=tq15+1;
 tq22=tq22+1;
 tq23=tq23+1;
 tq24=tq24+1;
 tq25=tq25+1;
 tq31=tq31+1;
 tq32=tq32+1;
 tq33=tq33+1;
 tq41=tq41+1;
 tq42=tq42+1;
 tq43=tq43+1;
 ty1=ty1-1;
 tk=tk-1;
 if(!input(ykabul)) ty3=ty3+1;
 else ty3=0;
 if(input(fuzzy))  printf(lcd_putc,"\ftopuz:%U Uz:%U\nQBD:%U QKG:%U ",ty2,uzatma,maxq12,maxq34);
          else printf(lcd_putc,"\fsure:%U \nQBD:%U QKG:%U ",ty3,maxq12,maxq34);
 }
   void saat1(void)
{
  TM1=TM1+1;
            if(TM1>=30) {//prescalar küçült


                        TM2=TM2+1;

                        if(TM2==2) {//saniye artır
                        TM1=0;
                        TM2=0;

                SN=SN+1;
     zaman_test();
```

```
      kuyruk_test();
     if(SN==60)         {//dakika artır
     SN=0;
                        DK=DK+1;

                        if(DK==60)              {//saat artır
      DK=0;
                        SA=SA+1;
                        if(SA==24)              SA=0;

     }
     }
     }
   }
}

       // interrupt sonu
#separate
float yq1q2k(float x) {
 y=(60-x)/60;
 return(y);
 }
 #separate
 float yq1q2o1(float x) {
 y=(x-50)/50;
 return(y);
 }
 #separate
 float yq1q2o2(float x) {
 y=(150-x)/50;
 return(y);
 }
 #separate
 float yq1q2b(float x) {
 y=(x-140)/60;
 return(y);
 }
 #separate
 float maxq3q4k(float x) {
 y=(30-x)/30;
 return(y);
 }
 #separate
 float maxq3q4o1(float x) {
 y=(x-25)/25;
 return(y);
 }
 #separate
 float maxq3q4o2(float x) {
 y=(75-x)/25;
 return(y);
 }
 #separate
 float maxq3q4b(float x) {
 y=(x-70)/30;
 return(y);
 }

#separate
 float out11(float x) {
 y=4-4*x;
 return(y);
 }
 #separate
 float out21(float x) {
 y=4*x;
 return(y);
 }
#separate
 float out22(float x) {
 y=8-4*x;
 return(y);
 }
 #separate
```

```
float out31(float x) {
y=4*x+4;
return(y);
}
#separate
float out32(float x) {
y=12-4*x;
return(y);
}
#separate
float out41(float x) {
y=4*x+8;
return(y);
}
#separate
float out42(float x) {
y=16-4*x;
return(y);
}
#separate
float out51(float x) {
y=4*x+12;
return(y);
}
#separate
float out52(float x) {
y=20-4*x;
return(y);
}
#separate
float out61(float x) {
y=4*x+16;
return(y);
}

#separate
void fonksiyon1(void) { // 27 adet kural için ayrı ayrı fonksiyonları çağırıp
                //işlem yaptıracağım
//1. kural için

      if((0<=maxq12) && (maxq12<=60)) x1=yq1q2k(maxq12);
      else x1=0;
        if((0<=maxq34) && (maxq34<=30)) x2=maxq3q4k(maxq34);
       else x2=0;
       if(x1>x2) x1=x2;//x1 x2'den küçük
                    //en kucuk degeri x1 e atayarak yer tasarrufu yaptım
      if((x1==0) || (x2==0)) {x1=0;
                          y1=10; }
       else {
       y1=10;  // 1 nolu signum x=0 noktasında
       }//1. kurala ait keskin değeri buldum
       pay=(x1*y1);
       payda=x1;
// 2. kural burada başlar
     if((0<=maxq12) && (maxq12<=60)) x1=yq1q2k(maxq12);
      else x1=0;
     if((25<=maxq34) && (maxq34<50)) x2=maxq3q4o1(maxq34);
     else if ((50<=maxq34) && (maxq34<=75)) x2=maxq3q4o2(maxq34);
     else x2=0;
       if(x1>x2) x1=x2;//x1 x2'den küçük
                    //en kucuk degeri x1 e atayarak yer tasarrufu yaptım
      if((x1==0) || (x2==0)) {x1=0;
                    y1=5; }
       else {
       y1=5;  // 1 nolu signum x=0 noktasında
       }//2. kurala ait keskin değeri buldum
       pay=pay+(x1*y1);
       payda=payda+x1;
// 3. kural için
      if((0<=maxq12) && (maxq12<=60)) x1=yq1q2k(maxq12);
      else x1=0;
      if((70<=maxq34) && (maxq34<=100)) x2=maxq3q4b(maxq34);
       else x2=0;
      if(x1>x2) x1=x2;//x1 x2'den küçük
```

```
                    //en kucuk degeri x1 e atayarak yer tasarrufu yaptım
      if((x1==0) || (x2==0)) {  x1=0;
                          y1=0; }
       else {
         // 0 nolu signum x=0 noktasında
       y1=0;
       }//3. kurala ait keskin değeri buldum
       pay=pay+(x1*y1);
       payda=payda+x1;


}
#separate
void fonksiyon2(void) {
// 4. kural için
      if((50<=maxq12) && (maxq12<100))  x1=yq1q2o1(maxq12);
       else if((100<=maxq12) && (maxq12<=150)) x1=yq1q2o2(maxq12);
       else x1=0;
       if((0<=maxq34) && (maxq34<=30)) x2=maxq3q4k(maxq34);
        else x2=0;
       if(x1>x2) x1=x2;//x1 x2'den küçük
                    //en kucuk degeri x1 e atayarak yer tasarrufu yaptım
       if((x1==0) || (x2==0)) z4=0;
        else {
        z4=x1;  //
        }//4. kurala ait keskin değeri buldum
//5.kural
      if((50<=maxq12) && (maxq12<100))  x1=yq1q2o1(maxq12);
       else if((100<=maxq12) && (maxq12<=150)) x1=yq1q2o2(maxq12);
       else x1=0;
       if((25<=maxq34) && (maxq34<50)) x2=maxq3q4o1(maxq34);
      else if ((50<=maxq34) && (maxq34<=75)) x2=maxq3q4o2(maxq34);
      else x2=0;
       if(x1>x2) x1=x2;//x1 x2'den küçük
                    //en kucuk degeri x1 e atayarak yer tasarrufu yaptım
       if((x1==0) || (x2==0)) z7=0;
        else {
        z7=x1;
        }//5. kural
//6. kural için

      if((50<=maxq12) && (maxq12<100))  x1=yq1q2o1(maxq12);
       else if((100<=maxq12) && (maxq12<=150)) x1=yq1q2o2(maxq12);
       else x1=0;
       if((70<=maxq34) && (maxq34<=100)) x2=maxq3q4b(maxq34);
        else x2=0;
       if(x1>x2) x1=x2;//x1 x2'den küçük
                    //en kucuk degeri x1 e atayarak yer tasarrufu yaptım
       if((x1==0) || (x2==0)) {y1=15;
                          z6=0; }
        else {
        z6=x1;
        y1=15;  // 4 nolu signum x=15 noktasında
        }//6. kurala ait keskin değeri buldum
        if(z4>z7) max=z4;
        else max=z7;
        if(z6>max) max=z6;
        x1=max;
        pay=pay+(x1*y1);
        payda=payda+x1;
}

#separate
void fonksiyon3(void) {
//7. kural için
      if((140<=maxq12) && (maxq12<=200)) x1=yq1q2b(maxq12);
       else x1=0;
       if((0<=maxq34) && (maxq34<=30)) x2=maxq3q4k(maxq34);
        else x2=0;
       if(x1>x2) x1=x2;//x1 x2'den küçük
                    //en kucuk degeri x1 e atayarak yer tasarrufu yaptım
       if((x1==0) || (x2==0)) z1=0;
        else {
        z1=x1;  // 5 nolu signum x=20 noktasında
```

```
       }//7. kurala ait keskin değeri buldum
//8. kural için
       if((140<=maxq12) && (maxq12<=200)) x1=yq1q2b(maxq12);
        else x1=0;
       if((25<=maxq34) && (maxq34<50)) x2=maxq3q4o1(maxq34);
      else if ((50<=maxq34) && (maxq34<=75)) x2=maxq3q4o2(maxq34);
      else x2=0;
       if(x1>x2) x1=x2;//x1 x2'den küçük
                    //en kucuk degeri x1 e atayarak yer tasarrufu yaptım
        if((x1==0) || (x2==0)) z2=0;

        else {
        z2=x1;  // 4 nolu signum x=15 noktasında
         }//8. kurala ait keskin değeri buldum
//9. kural için
       if((140<=maxq12) && (maxq12<=200)) x1=yq1q2b(maxq12);
        else x1=0;
       if((70<=maxq34) && (maxq34<=100)) x2=maxq3q4b(maxq34);
        else x2=0;
       if(x1>x2) x1=x2;//x1 x2'den küçük
                    //en kucuk degeri x1 e atayarak yer tasarrufu yaptım
       if((x1==0) || (x2==0)) {z3=0;
                          y1=20; }
        else {
        z3=x1;
        y1=20;  // 3 nolu signum x=10 noktasında
        }//9. kurala ait keskin değeri buldum
        if(z1>z2) max=z1;
        else max=z2;
        if(z3>max) max=z3;
        x1=max;
        pay=pay+(x1*y1);
        payda=payda+x1;

        // COG metodu kullanılarak keskin değer hesaplatacağız.


         uzatma= (int) pay/payda;  // burada en son keskin değer hesaplanıyor
}

#separate
void isik_yak(void)  {
 dongu=0;
 ty1=20;
 ty2=20;
 yesil1:
 fonksiyon1();
 fonksiyon2();
 fonksiyon3();
 output_high(q12y);
 output_low(q12s);
 output_low(q12k);
 output_low(yonay);

 if(ty1>3) goto yesil1;
  fonksiyon1();
  fonksiyon2();
  fonksiyon3();
 if(uzatma>0) {
        ty2=ty2+uzatma;
        ty1=ty1+uzatma;
        dongu=dongu+1;
        if(dongu<6) goto yesil1;
        else {
        dongu=1;
        ty2=20;
        ty1=ty1-uzatma; }
        }

  while(ty1>0) {
  output_high(yonay);
  output_high(q12s);
  output_high(q12y);
  output_low(q12k); }
```

```
 output_low(q12s);
 output_low(q12y);
 output_high(q12k);

while(!input(ykabul)) {
 output_high(yonay);
 output_low(q12s);
 output_low(q12y);
 output_high(q12k); }

 while(input(ykabul)) {
 output_high(yonay);
 output_high(q12k);
 output_low(q12s);
 output_low(q12y); }

 output_low(yonay);
 output_high(q12s);
 output_low(q12y);
 output_high(q12k);
 delay_ms(3000);
 output_low(yonay);
 output_low(q12s);
 output_low(q12k);
 output_low(q12y);
 ty2=20;
 }

void fuzzy_bak(void)
{
 do {
   isik_yak();
 }while(input(fuzzy==1));
}
void saykil(void)
{
if(!input(ykabul)) {//1. if
output_low(yonay);
output_high(q12y);
output_low(q12s);
output_low(q12k);
delay_ms(57000);
output_high(q12s);
output_high(q12y);
output_low(q12k);
output_high(yonay);
delay_ms(3000);
while(!input(ykabul));
while(input(ykabul)) {
output_low(q12y);
output_low(q12s);
output_high(q12k);
output_high(yonay);
                  }
output_low(yonay);
output_low(q12y);
output_high(q12s);
output_high(q12k);
delay_ms(3000);
          } // if sonu

                  }
void kuyruk_test(void)
{
 if(!input(q12)) tq12=0;
 if(!input(q13)) tq13=0;
 if(!input(q14)) tq14=0;
 if(!input(q15)) tq15=0;
 if(input(q11) && !input(q12) && !input(q13) && !input(q14) && !input(q15)) q1=40;
 if(!input(q11) && !input(q12) && !input(q13) && !input(q14) && !input(q15)) q1=0;
 if((input(q12)) && (tq12>=2) && !input(q13) && !input(q14) && !input(q15) && input(q11)) q1=85;
 if((input(q13)) && (tq13>=2) && !input(q14) && !input(q15) && input(q12) && input(q11)) q1=120;
 if((input(q14)) && (tq14>=2) && input(q13)  && !input(q15) && input(q12) && input(q11) ) q1=160;
```

```
if((input(q15)) && (tq15>=2) && input(q13)  && input(q14) && input(q12) && input(q11)) q1=190;

if(!input(q22)) tq22=0;
if(!input(q23)) tq23=0;
if(!input(q24)) tq24=0;
if(!input(q25)) tq25=0;
if(input(q21) && !input(q22) && !input(q23) && !input(q24) && !input(q25)) q2=40;
if(!input(q21) && !input(q22) && !input(q23) && !input(q24) && !input(q25)) q2=0;
if((input(q22)) && (tq22>=2) && !input(q23) && !input(q24) && !input(q25) && input(q22)) q2=85;
if((input(q23)) && (tq23>=2) && !input(q24) && !input(q25) && input(q22) && input(q22)) q2=120;
if((input(q24)) && (tq24>=2) && input(q23)  && !input(q25) && input(q22) && input(q22) ) q2=160;
if((input(q25)) && (tq25>=2) && input(q23)  && input(q24) && input(q22) && input(q22)) q2=190;

if(!input(q32)) tq32=0;
if(!input(q33)) tq33=0;
if(!input(q31)) tq31=0;
if(input(q31) && (tq31>=2) && !input(q32) && !input(q33)) q3=30;
if(!input(q31) && !input(q32) && !input(q33) ) q3=0;
if((input(q32)) && (tq32>=2) && !input(q33)  && input(q31)) q3=50;
if((input(q33)) && (tq33>=2)  && input(q32) && input(q31)) q3=90;

if(!input(q42)) tq42=0;
if(!input(q43)) tq43=0;
if(!input(q41)) tq41=0;
if(input(q41) && !input(q42) && !input(q43)) q4=30;
if(!input(q41) && !input(q42) && !input(q43)) q4=0;
if((input(q42)) && (tq42>=2) && !input(q43)  && input(q41)) q4=50;
if((input(q43)) && (tq43>=2)  && input(q42) && input(q41)) q4=90;

if(q1>q2) maxq12=q1;
else maxq12=q2;
if(q3>q4) maxq34=q3;
else maxq34=q4;
}
main(void)
{
  set_rtcc(0);
  SET_TRIS_B(0XFF);
            SET_TRIS_A(0X02);
            SET_TRIS_C(0XFF);
            SET_TRIS_D(0X00);
  SET_TRIS_E(0X0D);
            delay_ms(20);
            lcd_init();
  setup_counters(RTCC_INTERNAL,RTCC_DIV_128);
  enable_interrupts(INT_RTCC);
  enable_interrupts(GLOBAL);
            for(;;)
            {  //for başı
  output_low(yonay);
  kuyruk_test();
  if(input(fuzzy)) fuzzy_bak();
  else saykil();
}// for döngü sonu
}//main sonu
```

EK.8 Fuzzy_logic_kırmızı.c kaynak kodları

```
#ifdef __PCH__
#device PIC16F877
#endif
#include <16F877.H>
#device  *=16
#fuses XT,NOWDT
#use delay(clock=4000000)
#include <stdlib.h>
#include <math.h>
#fuses XT,NOWDT,NOPROTECT,NOLVP
#include <lcd.c>
#define q15 pin_C7
#define q14 pin_C6
#define q13 pin_C5
#define q12 pin_C4
#define q11 pin_C3
#define fuzzy pin_E0
#define q12k pin_A5 // kulvar doluysa 1 oluyor
#define q12s pin_A3
#define q12y pin_A2
#define yonay pin_A0
#define ykabul pin_A1
#define q25 pin_C2//karşı şerit durumunu alıyor
#define q24 pin_C1// karşı şerit onay
#define q23 pin_C0
#define q22 pin_B7
#define q21 pin_B6
#define q31 pin_B5
#define q32 pin_B4
#define q33 pin_B3
#define q41 pin_B2
#define q42 pin_B1
#define q43 pin_B0
              byte TM1=0;
              byte  TM2=0;
              byte  DK=0;
              unsigned int  SA=0;
              unsigned int SN=0;
  unsigned int q1=0;
  unsigned int q2=0;
  unsigned int q3=0;
  unsigned int q4=0;
  unsigned int i=1;
  unsigned int j=1;
  unsigned int t=1;
  unsigned int k=1;
  unsigned int maxq12;
  unsigned int maxq34;

  byte tq12=0;
  byte tq13=0;
  byte tq14=0;
  byte tq15=0;
  byte tq22=0;
  byte tq23=0;
  byte tq24=0;
  byte tq25=0;
  byte tq31=0;
  byte tq32=0;
  byte tq33=0;
  byte tq41=0;
  byte tq42=0;
  byte tq43=0;
  float x1=0;
  float x2=0;
  float y1=0;
```

```
  float y2=0;
  float maxs1=0;
  float maxs2=0;
  float max=0;
  float y;
  float x;
  float z1=0;
  float z2=0;
  float z3=0;
  float z4=0;
  float z5=0;
  float z6=0;
  float z7=0;
  float z8=0;
  float z9=0;
  float pay;
  float payda;
  unsigned int ty1=20;
  unsigned int ty2=0;
  byte tk=0;
  byte ty3=0;
  unsigned int uzatma;
  unsigned int dongu=0;
      loop_kontrol(void);
      cins_tespit(void);
      saatayarla(void);
      dkayarla(void);
      kuyruk_test(void);
      saat1(void);
      #INT_RTCC
      saat_isr() { // saat interruptı burada başlıyor


                  saat1();

      }
//saat kısmı burada
 void zaman_test(void) {  //kuyruk zamanlama ile ilgili kısım
 tq12=tq12+1;
 tq13=tq13+1;
 tq14=tq14+1;
 tq15=tq15+1;
 tq22=tq22+1;
 tq23=tq23+1;
 tq24=tq24+1;
 tq25=tq25+1;
 tq31=tq31+1;
 tq32=tq32+1;
 tq33=tq33+1;
 tq41=tq41+1;
 tq42=tq42+1;
 tq43=tq43+1;
 ty1=ty1-1;
 tk=tk-1;
 if(input(ykabul)) ty3=ty3+1;
 else ty3=0;
 if(input(fuzzy))  printf(lcd_putc,"\ftopuz:%U Uz:%U\nQBD:%U QKG:%U ",ty2,uzatma,maxq12,maxq34);
          else printf(lcd_putc,"\fsure:%U \nQBD:%U QKG:%U ",ty3,maxq12,maxq34);

 }
   void saat1(void)
{
  TM1=TM1+1;
            if(TM1>=30) {//prescalar küçült


                        TM2=TM2+1;

                        if(TM2==2) {//saniye artır
                        TM1=0;
                        TM2=0;

                SN=SN+1;
     zaman_test();
```

```
      kuyruk_test();
     if(SN==60)         {//dakika artır
     SN=0;
                        DK=DK+1;

                        if(DK==60)              {//saat artır
      DK=0;
                        SA=SA+1;
                        if(SA==24)              SA=0;

     }
     }
     }
   }
}

       // interrupt sonu
#separate
float yq1q2k(float x) {
 y=(60-x)/60;
 return(y);
 }
 #separate
 float yq1q2o1(float x) {
 y=(x-50)/50;
 return(y);
 }
 #separate
 float yq1q2o2(float x) {
 y=(150-x)/50;
 return(y);
 }
 #separate
 float yq1q2b(float x) {
 y=(x-140)/60;
 return(y);
 }
 #separate
 float maxq3q4k(float x) {
 y=(30-x)/30;
 return(y);
 }
 #separate
 float maxq3q4o1(float x) {
 y=(x-25)/25;
 return(y);
 }
 #separate
 float maxq3q4o2(float x) {
 y=(75-x)/25;
 return(y);
 }
 #separate
 float maxq3q4b(float x) {
 y=(x-70)/30;
 return(y);
 }


 #separate
 float out11(float x) {
 y=4-4*x;
 return(y);
 }
 #separate
 float out21(float x) {
 y=4*x;
 return(y);
 }
#separate
 float out22(float x) {
 y=8-4*x;
 return(y);
 }
```

```
#separate
float out31(float x) {
y=4*x+4;
return(y);
}
#separate
float out32(float x) {
y=12-4*x;
return(y);
}
#separate
float out41(float x) {
y=4*x+8;
return(y);
}
#separate
float out42(float x) {
y=16-4*x;
return(y);
}
#separate
float out51(float x) {
y=4*x+12;
return(y);
}
#separate
float out52(float x) {
y=20-4*x;
return(y);
}
#separate
float out61(float x) {
y=4*x+16;
return(y);
}
#separate
void fonksiyon1(void) { // 27 adet kural için ayrı ayrı fonksiyonları çağırıp
                  //işlem yaptıracağım
       pay=0;
       payda=0;
//1. kural için

      if((0<=maxq12) && (maxq12<=60)) x1=yq1q2k(maxq12);
      else x1=0;
        if((0<=maxq34) && (maxq34<=30)) x2=maxq3q4k(maxq34);
       else x2=0;
       if(x1>x2) x1=x2;//x1 x2'den küçük
                  //en kucuk degeri x1 e atayarak yer tasarrufu yaptım
      if((x1==0) || (x2==0)) {
                        z1=0;}
       else {
       z1=x1;
        // 1 nolu signum x=0 noktasında
       }//1. kurala ait keskin değeri buldum
// 4. kural için
      if((50<=maxq12) && (maxq12<100))  x1=yq1q2o1(maxq12);
      else if((100<=maxq12) && (maxq12<=150)) x1=yq1q2o2(maxq12);
      else x1=0;
      if((0<=maxq34) && (maxq34<=30)) x2=maxq3q4k(maxq34);
       else x2=0;
      if(x1>x2) x1=x2;//x1 x2'den küçük
                  //en kucuk degeri x1 e atayarak yer tasarrufu yaptım
      if((x1==0) || (x2==0)) {
                        z2=0;}
       else {
       z2=x1;
         // 1 nolu signum x=0 noktasında
       }//4. kurala ait kesk

//7. kural için
      if((140<=maxq12) && (maxq12<=200)) x1=yq1q2b(maxq12);
       else x1=0;
      if((0<=maxq34) && (maxq34<=30)) x2=maxq3q4k(maxq34);
       else x2=0;
```

```
      if(x1>x2) x1=x2;//x1 x2'den küçük
                  //en kucuk degeri x1 e atayarak yer tasarrufu yaptım
      if((x1==0) || (x2==0)) {
                     z3=0;}
       else {
       z3=x1;
         // 1 nolu signum x=0 noktasında
       }//7. kurala ait kesk


//8. kural için
      if((140<=maxq12) && (maxq12<=200)) x1=yq1q2b(maxq12);
       else x1=0;
      if((25<=maxq34) && (maxq34<50)) x2=maxq3q4o1(maxq34);
     else if ((50<=maxq34) && (maxq34<=75)) x2=maxq3q4o2(maxq34);
     else x2=0;
      if(x1>x2) x1=x2;//x1 x2'den küçük
                  //en kucuk degeri x1 e atayarak yer tasarrufu yaptım
      if((x1==0) || (x2==0)) {y1=0;
                     z4=0;}
       else {
       z4=x1;
       y1=0;  // 1 nolu signum x=0 noktasında
       }//8. kurala ait kesk
       if(z1>z2) max=z1;
       else max=z2;
       if(z3>max) max=z3;
       if(z4>max) max=z4;
       x1=max;
       pay=x1*y1;
       payda=x1;
}

 #separate
void fonksiyon2(void) {

// 2. kural burada başlar
     if((0<=maxq12) && (maxq12<=60)) x1=yq1q2k(maxq12);
      else x1=0;
     if((25<=maxq34) && (maxq34<50)) x2=maxq3q4o1(maxq34);
     else if ((50<=maxq34) && (maxq34<=75)) x2=maxq3q4o2(maxq34);
     else x2=0;
      if(x1>x2) x1=x2;//x1 x2'den küçük
                  //en kucuk degeri x1 e atayarak yer tasarrufu yaptım
      if((x1==0) || (x2==0))  {
                     z5=0;}

       else {
       z5=x1; // 1 nolu signum x=0 noktasında
       }//2. kurala ait keskin değeri buldum

 //6. kural için

      if((50<=maxq12) && (maxq12<100))  x1=yq1q2o1(maxq12);
      else if((100<=maxq12) && (maxq12<=150)) x1=yq1q2o2(maxq12);
      else x1=0;
      if((70<=maxq34) && (maxq34<=100)) x2=maxq3q4b(maxq34);
       else x2=0;
      if(x1>x2) x1=x2;//x1 x2'den küçük
                  //en kucuk degeri x1 e atayarak yer tasarrufu yaptım
      if((x1==0) || (x2==0))  {y1=15;
                     z6=0;}
       else {
       z6=x1;
       y1=15;  // 1 nolu signum x=0 noktasında
       }//6. kurala ait keskin değeri buldum
       if(z5>z6) x1=z5;
       else x1=z6;
       pay=pay+(x1*y1);
       payda=payda+x1;

// 3. kural için
      if((0<=maxq12) && (maxq12<=60)) x1=yq1q2k(maxq12);
      else x1=0;
```

```
      if((70<=maxq34) && (maxq34<=100)) x2=maxq3q4b(maxq34);
       else x2=0;
      if(x1>x2) x1=x2;//x1 x2'den küçük
                  //en kucuk degeri x1 e atayarak yer tasarrufu yaptım
      if((x1==0) || (x2==0)) y1=0;
       else {
       y1=20;  // 1 nolu signum x=0 noktasında
       }//3. kurala ait keskin değeri buldum
       pay=pay+(x1*y1);
       payda=payda+x1;

//5.kurif((60<=maxq12) && (maxq12<80))  x1=yq1q2o1(maxq12);
      if((50<=maxq12) && (maxq12<100))  x1=yq1q2o1(maxq12);
      else if((100<=maxq12) && (maxq12<=150)) x1=yq1q2o2(maxq12);
      else x1=0;
      if((25<=maxq34) && (maxq34<50)) x2=maxq3q4o1(maxq34);
     else if ((50<=maxq34) && (maxq34<=75)) x2=maxq3q4o2(maxq34);
     else x2=0;
      if(x1>x2) x1=x2;//x1 x2'den küçük
                  //en kucuk degeri x1 e atayarak yer tasarrufu yaptım
      if((x1==0) || (x2==0)) y1=0;
       else {
       y1=10;  // 1 nolu signum x=0 noktasında
       }//5. kurala ait keskin değeri buldum
       pay=pay+(x1*y1);
       payda=payda+x1;
}
#separate
void fonksiyon3(void) {

//9. kural için
      if((140<=maxq12) && (maxq12<=200)) x1=yq1q2b(maxq12);
       else x1=0;
      if((70<=maxq34) && (maxq34<=100)) x2=maxq3q4b(maxq34);
       else x2=0;
      if(x1>x2) x1=x2;//x1 x2'den küçük
                  //en kucuk degeri x1 e atayarak yer tasarrufu yaptım
      if((x1==0) || (x2==0)) y1=0;
       else {
       y1=5;  // 1 nolu signum x=0 noktasında
       }//6. kurala ait keskin değeri buldum
       pay=pay+(x1*y1);
       payda=payda+x1;

       // COG metodu kullanılarak keskin değer hesaplatacağız.
        uzatma= (int) pay/payda;  // burada en son keskin değer hesaplanıyor
}
#separate
void isik_yak(void)  {
 dongu=0;
 ty2=20;
 while(!input(ykabul)) {
 output_low(q12y);
 output_low(q12s);
 output_high(q12k);
 output_low(yonay);
 }
 ty1=20;
 if(input(ykabul)) {
 output_low(q12y);
 output_high(q12s);
 output_high(q12k);
 output_high(yonay);
 delay_ms(3000);
 }
yesil1:
  fonksiyon1();
  fonksiyon2();
  fonksiyon3();
 output_low(q12s);
 output_low(q12k);
 output_high(q12y);
 output_high(yonay);
 if(ty1>3) goto yesil1;
```

```
 fonksiyon1();
 fonksiyon2();
 fonksiyon3();
if(uzatma>0) {
      ty2=ty2+uzatma;
      ty1=ty1+uzatma;
      dongu=dongu+1;
      if(dongu<6) goto yesil1;
      else {
      dongu=1;
      ty2=20;
      ty1=ty1-uzatma; }
      }
 while(ty1>0) {
 output_low(yonay);
 output_high(q12s);
 output_high(q12y);
 output_low(q12k); }

 output_low(yonay);
 output_low(q12s);
 output_low(q12y);
 output_high(q12k);
ty2=20;
 }

void fuzzy_bak(void)
{
 do {
   isik_yak();
 }while(input(fuzzy==1));
}
void saykil(void)
{
while(!input(ykabul)) {
output_high(q12k);
output_low(q12s);
output_low(q12y);  }
output_high(yonay);

if(input(ykabul)) {
output_high(q12k);
output_high(q12s);
output_low(q12y);
output_high(yonay);
delay_ms(3000);
while(input(ykabul)) {
output_low(q12k);
output_low(q12s);
output_high(q12y);
output_high(yonay);
delay_ms(37000);
output_low(yonay);
output_low(q12k);
output_high(q12s);
output_high(q12y);
delay_ms(3000);
while(input(ykabul)) {
output_high(q12k);
output_low(q12s);
output_low(q12y);
             }
           }
         }
output_low(yonay);
output_high(q12k);
output_low(q12y);
output_low(q12s);
               }


void kuyruk_test(void)
{
 if(!input(q12)) tq12=0;
```

```
if(!input(q13)) tq13=0;
if(!input(q14)) tq14=0;
if(!input(q15)) tq15=0;
if(input(q11) && !input(q12) && !input(q13) && !input(q14) && !input(q15)) q1=40;
if(!input(q11) && !input(q12) && !input(q13) && !input(q14) && !input(q15)) q1=0;
if((input(q12)) && (tq12>=2) && !input(q13) && !input(q14) && !input(q15) && input(q11)) q1=85;
if((input(q13)) && (tq13>=2) && !input(q14) && !input(q15) && input(q12) && input(q11)) q1=120;
if((input(q14)) && (tq14>=2) && input(q13)  && !input(q15) && input(q12) && input(q11) ) q1=160;
if((input(q15)) && (tq15>=2) && input(q13)  && input(q14) && input(q12) && input(q11)) q1=190;

if(!input(q22)) tq22=0;
if(!input(q23)) tq23=0;
if(!input(q24)) tq24=0;
if(!input(q25)) tq25=0;
if(input(q21) && !input(q22) && !input(q23) && !input(q24) && !input(q25)) q2=40;
if(!input(q21) && !input(q22) && !input(q23) && !input(q24) && !input(q25)) q2=0;
if((input(q22)) && (tq22>=2) && !input(q23) && !input(q24) && !input(q25) && input(q22)) q2=85;
if((input(q23)) && (tq23>=2) && !input(q24) && !input(q25) && input(q22) && input(q22)) q2=120;
if((input(q24)) && (tq24>=2) && input(q23)  && !input(q25) && input(q22) && input(q22) ) q2=160;
if((input(q25)) && (tq25>=2) && input(q23)  && input(q24) && input(q22) && input(q22)) q2=190;

if(!input(q32)) tq32=0;
if(!input(q33)) tq33=0;
if(!input(q31)) tq31=0;
if(input(q31) && (tq31>=2) && !input(q32) && !input(q33)) q3=30;
if(!input(q31) && !input(q32) && !input(q33) ) q3=0;
if((input(q32)) && (tq32>=2) && !input(q33)  && input(q31)) q3=50;
if((input(q33)) && (tq33>=2)  && input(q32) && input(q31)) q3=90;

if(!input(q42)) tq42=0;
if(!input(q43)) tq43=0;
if(!input(q41)) tq41=0;
if(input(q41) && !input(q42) && !input(q43)) q4=30;
if(!input(q41) && !input(q42) && !input(q43)) q4=0;
if((input(q42)) && (tq42>=2) && !input(q43)  && input(q41)) q4=50;
if((input(q43)) && (tq43>=2)  && input(q42) && input(q41)) q4=90;

if(q1>q2) maxq12=q1;
else maxq12=q2;
if(q3>q4) maxq34=q3;
else maxq34=q4;
}
main(void)
{
  set_rtcc(0);
  SET_TRIS_B(0XFF);
            SET_TRIS_A(0X02);
            SET_TRIS_C(0XFF);
            SET_TRIS_D(0X00);
  SET_TRIS_E(0X0D);
            delay_ms(20);
            lcd_init();
  setup_counters(RTCC_INTERNAL,RTCC_DIV_128);
  enable_interrupts(INT_RTCC);
  enable_interrupts(GLOBAL);
  fonksiyon1();
 fonksiyon2();
 fonksiyon3();
  output_low(yonay);
  output_high(q12k);
  output_low(q12s);
  output_low(q12y);
            for(;;)
            {  //for başı

  kuyruk_test();
  if(input(fuzzy)) fuzzy_bak();
  else saykil();


}// for döngü sonu
}//main sonu
```

ÖZGEÇMİŞ

Osman Demirci 1971 yılında Ankara'da doğdu. İlk, orta ve lise öğrenimini İzmit'te tamamladıktan sonra 1989 yılında girdiği İstanbul Üniversitesi Elektronik Mühendisliği Bölümünden 1993 yılında mezun oldu. 2005 yılında girdiği Kocaeli Üniversitesi Fen Bilimleri Enstitüsü Elektronik ve Haberleşme Bölümündeki yüksek lisans öğrenimi 2007 yılında tamamlamıştır. 1997 yılından itibaren Karayolları 17. Bölge Müdürlüğünde Elektronik Mühendisi olarak çalışmakta olup evli ve iki çocuk babasıdır.