YILDIZ TEKNİK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

AKILLI EV OTOMASYONUNUN MİKRODENETLEYİCİ İLE GERÇEKLEŞTİRİLMESİ

Elektrik Müh. Hande KONGAZ

FBE Elektrik Mühendisliği Anabilim Dalı Kontrol ve Otomasyon Programında Hazırlanan

YÜKSEK LİSANS TEZİ

Tez Danışmanı: Prof. Dr. Galip CANSEVER (YTU)

İSTANBUL, 2007

İÇİNDEKİLER

SİMGE LİSTESİ

μ	Üyelik fonksiyonu
w_i	Kuralın çıkış üzerindeki etkisi
r	Fonksiyonun derecesi
z	Çıkış

KISALTMA LİSTESİ

AC	Alternating Current
ADC	Analog Digital Converter
CPU	Central Process Unit
DAC	Digital Analog Converter
DC	Direct Current
DMA	Direct Memory Acsess
DVD	Digital Versatile Disc
EEprpm	Electronically Erasable Programmable REad Only Memory
GHz	Giga Hertz
GPRS	General Packet Radio Service
HVAC	Heating Ventilating Air Conditioning
IEEE	Institute of Electrical and Electronics Engineers
I2C	Inter-Integrated Circuit
IRQ	Interrupt Request
Kbyte	Kilo Byte
kSPS	Kilo Sample Per Second
LCD	Liquid Cristal Display
LDR	Light Dependent Resistor
LED	Light Emiting diyote
MB	Mega Byte
MCU	Microcontrooler Central Unit
MIPS	Million Instruction Per Second
MMS	Multi Media Message Service
mV	Mili Volt
NB	Negatif Büyük
NK	Negatif Küçük
ns	Nano Second
PB	Pozitif Büyük
PK	Pozitif Küçük
PCB	Printed Circuit Board
PLL	Phase Locked Loop
PWM	Pulse Width Module
RAM	Random Access Memory
RF	Radio Frequency
RS232	Recomended Standart 232(EIA232)
RS485	Recomended Standart 485(EIA485)
ROM	Read Only Memory
RX	Recive
S	Sıfır
SMS	Short Message Service
SPI	Serial Peripheral Interface
TIC	Time Internal Counter
TX	Transmit
Uart	Universal Asynchronous Receiver/Transmitter
USB	Univesal Serial Bus
V	Volt
W	Watt
WiFİ	Wireless Fidelity

ŞEKİL LİSTESİ

ÇİZELGE LİSTESİ

ÖNSÖZ

Bu tez çalışmasında evlerde uygulanan kontrol mikroişlemci kullanılarak gerçekleştirilmiştir. Akıllı ev otomasyonunda kontrol edilen parametrelerden sıcaklık, aydınlatma ve güvenlik parametrelerinin kontrolü üzerinde çalışılmıştır.

Bu tez çalışmasında yardımlarını ve desteklerini esirgemeyen aileme ve arkadaşlarıma çok teşekkür ederim. Ayrıca çalışmam sırasında bana her türlü desteği veren ve kaynak gösteren Hocam Prof. Dr. Galip Cansever'e teşekkürü bir borç bilirim.

Aralık,2007 Hande KONGAZ

ÖZET

Akıllı ev otomasyonu günümüzde üzerinde çalışılan en önemli otomasyon konularından biridir. Her yeni yapılan bina bir öncekine göre üstünlüğünü sergileyererek otomasyona yeni bir bakış açısı kazandırmaktadır.

Gerçekleştirilen bu ev otomasyon projesinde maket bir ev modeli üzerinde çalışılmıştır. Odalarda bulunan mikroişlemci ile sıcaklık, aydınlatma ve güvenlik kontrolü yapılmaktadır. Sıcaklık kontrolünde bulanık mantık algoritması esas alınarak ekonomik yönden kazanç hedeflenmiştir. Aydınlatma kontrolünde kapalı çevrim kontrol yapılarak ışık grupları kademeli olarak sürülmüştür. Bu şekilde hem odada her noktada aynı aydınlık seviyesi yakalanmış olur hem de ekonomi yapılmış olunur. Güvenlik kontrolünde ise pencerelere yerleştirilen manyetik röleler kontrol edilir.

Ana panoda bulunan mikroişlemci ise oda panolarından odalardaki sıcaklık ve ışık bilgisini alarak gösterge panelinde göstermektedir. Ana pano üzerinde bulunan tuş takımı ile kullanıcı eve giriş yapmaktadır. Şifre girmede sorun olduğunda ise güvenlik kontrolü devreye girmektedir.

Anahtar kelimeler: Akıllı Ev Otomasyonu, Sıcaklık, Aydınlatma, Güvenlik, Bulanık Mantık

ABSTRACT

At the present days smart building automation is the most important subjects that studied in automaiton subjects. Every new building to show its advantages to previous one, comes with new ideas to building automation .

While working on Building Automation Project which is designed, a model house is used. The modules which is designed with a microcontroller , controls the room's temperature, lightening and security. Fuzzy logic control algorithm is used in temperature control in order to get a good return on financial. In lightening automation system close loop control algorithm is used and in room lightening sources grouped gradually. By gruoping lightening sources gradually, same light level is obtained. This system gets a good return on financial. In security control system, windows positions is controlled with magnetic relays.

In main panel which is used to control all the system that designed, rooms temperatures, lightening status are shown with display unit. With keypad on main panel, user enters to house by his password. If user does not enter any password in definite time, alarm system will be actived.

Key Words: Building Automation Systems, Temerature, Lightening, Security, Fuzzy Logic

1. GİRİŞ

Teknolojinin hızlı gelişimine bağlı olarak artık insanlar hayatlarını kolaylaştıran, ihtiyaçlarına cevap verebilen, kendilerine daha güvenli, daha konforlu ve en önemlisi daha tasarruflu bir yaşam sunan evlere sahip olmak istemektedirler. Bu istekler teknolojinin her geçen gün daha da gelişmesiyle artmaktadır. Buna bağlı olarak eskiden fabrikalarda kullanılan otomasyon sistemleri öncelikle plazalarda uygulanmaya başlanmıştır. Günümüzde ise yeni yapılan tüm binalarda ev otomasyon sistemleri standart hale gelmeye başlamıştır.

Ev teknolojilerinin kişiye özel ihtiyaç ve isteklere göre şekillendirilmesi ile oluşan sisteme ev otomasyonu denir. Ev otomasyon kontrolü ile evde sıcaklık kontrolü, aydınlatma kontrolü, güvenlik kontrolü, elektronik eşya kontrolü yapılmaktadır. Bunların yanında yangın alarm sistemi, bahçe sulama sistemi, perde/jaluzi otomatik kontrolü, kullanıcı tanıma gibi başlıklarda ev otomasyonun çalışma alanlarındandır.

Uygulaması gerçekleştirilen ev otomasyonu sisteminde sıcaklık, aydınlatma ve güvenlik kontrolü gerçekleştirilmiştir. Aydınlatma kontrolünde enerji tasarrufu sağlamak ve odanın her noktasında aynı aydınlık seviyesinin olmasını sağlamak için kademeli ışık kontrolü oransal kontrol ile yapılmıştır. Ayrıca odanın içerisindeki ışık bilgisi sensörler yardımı ile okunarak kapalı çevrim kontrol algoritması ile kontrol edilmiştir. Sıcaklık kontrolünde klasik mantık uygulaması yerine bulanık mantık algoritma uygulaması tercih edilmiştir. Klasik mantık ile gerçekleştirilen sıcaklık uygulamalarında ısıtıcı/soğutucu sürekli belirli bir kademede açık veya tamamen kapalıdır. Odayı az veya çok ısıtılması istenildiğinde tek bir kademede ısıtıcı açılmaktadır. Bulanık mantık algoritmasında ise ısıtıcı/soğutucunun kademeli çalışması esas alınmıştır. Kısaca odanın az ısıtılması istenildiğinde ısıtıcı az, odanın çok ısıtılması istenildiğinde ısıtıcı çok çalıştırılmaktadır. Bulanık mantık ile mikroişlemciye insan hayatında kullanılan dilsel niteleyiciler yüklenmiş olmaktadır. Ayrıca sıcaklık kontrolünün yapılacağı oda için bir transfer fonksiyonu kullanılmadan bulanık mantık algoritması ile kontrol rahatlıkla sağlanır.

Güvenlik kontrolünde gerçekleştirilen sistemlerde evde insan yok iken eve dolu izlenimi vermek için ışıklar önceden ayarlanan bir saatte yanabilir veya müzik sistemi devreye girebilir. Eve izinsiz giriş olduğunda bu giriş algılanarak ev sahibine ve gerekli güvenlik birimlerine haber verilebilir ve ayrıca caydırıcılık amacıyla sesli ve ışıklı bir alarm sistemi kurulmuş olabilir. Gerçekleştirilen sistemde güvenlik için pencere kontrolü yapılmış ve kullanıcı tanıma sistemi gerçekleştirilmiştir.

2. AKILLI EV OTOMASYONU

Günümüzdeki teknolojik gelişmeler hayatın her alanına yenilikler ve kolaylıklar getirmiştir. Teknolojinin hizmet alanı insan olduğu için ve insan için en önemli gündelik yaşam geçirme alanı bina içi olduğundan, zaman içinde geleneksel ev yapısında da değişikliklere gidilmiştir. Otomasyon kelimesi Fransızca kökenli bir kelime olup auto+motion kelimelerinin birleşmesiyle oluşmuştur. Özişler anlamına gelen bu kelime ev kelimesinin sonuna geldiğinde kendi kendine işleyen ev anlamına gelmektedir.

Akıllı ev tanımı genel olarak her şeyi ile kontrol edilebilen, servis kontrol sistemine sahip olan ev şeklinde yapılmaktadır. The Intelligent Building Institution'nın yaptığı tanım da şu şekildedir; akıllı bir ev, çeşitli sistemleri bir arada koordineli bir şekilde kullanarak teknik performansı, yatırımları ve işletim maliyetlerini düşürmeyi, esneklik kazandırmayı en üst seviyeye taşıyan yapıdır.

Akıllı ev denilince birçok farklı kavram akla gelmektedir. Bunun sebebi akıllı ev kavramının sadece yapay us ile ilgili konularda kullanılmaması, bu kavramın evin içine teknolojinin girdiği her konuda kullanılmasıdır. Yapay us ile ilgili kısmı diğerlerinden ayırmak, başarılı bir sınıflandırmaya gitmekle mümkündür [14].

Binalar için akıllı kelimesi ilk olarak Amerika Birleşik Devletlerinde 1980 yılının başlarında kullanılmıştır. Türkiye'deki ilk uygulama ise 1984 yılında yapılmıştır. Bu sistem yalnızca izlemeye yönelik olarak tasarlanmıştır. Geçen yıllar içinde teknolojinin gelişmesine paralel olarak yaşantımızın her alanında ciddi değişimler oldu. Mekatronik sistemlerdeki gelişmeler daha hızlı, daha yüksek kapasiteli kontrol cihazlarının kullanılmasına imkân verdi [12].

Akıllı evlerde çeşitli mekatronik uygulamalar gerçekleştirilebilir, bunlardan bazıları aydınlatma kontrolü, uzaktan kontrol, telefon sistemleri, güvenlik sistemleri, hareket dedektörleri, iklim kontrolü ve benzerleridir.

Akıllı ev otomasyonunu uygulanmasındaki ana amaçlar güvenlik ve emniyetin artırılması, konforun artırılması, basitlik ve kullanım kolaylığı sağlanması ve enerji tasarrufu sağlanmasıdır.

2.1 Akıllı Evlerin Sınıflandırılması

Akıllı evler karakteristiklerine göre üç ana kategoride incelenebilir; kontrol edilebilir evler, programlanabilir evler ve zeki evler. Kontrol edilebilir evler, ev içindeki eşyaların kontrol edildiği evlerdir. Bu tür evlerde teknolojik aletlerin kontrolü esas alınmıştır. Programlanabilir

evler, kontrol edilebilir evlere göre daha gelişmiş bir sınıftır. Bu evler zamana ve basit sensörlere tepki vermektedirler. Örneğin evin içinde ışık seviyesine göre ışıklar yanar veya söner. Zeki evler, programlanabilir evler ile benzerlik göstermektedir. Fakat programlanabilir evlere göre daha gelişmişlerdir. Programlanabilir evlerde senaryo girişi yapılmaktayken zeki evlerde senaryo girişi yapılmaz. Zeki evlerin öğrenme yeteneği vardır. Bu tip evler, kendi kendine inceleyip, buna göre kendi kurallarını ve senaryolarını yaratabilen sistemlere sahiptirler. Bunu yapabilmek için de öğrenme yeteneğine sahip yazılımlar gereklidir. Zeki evler, yaşayanların her günkü hareketlerini izlerler, tekrar eden hareketleri ortaya çıkartırlar. Bu hareketler belirlendikten sonra, o durum için yapılacaklar belirlenir ve bir daha aynı hareket ile karşılaşıldığında uygun eylemler yapılır [10].

2.2 Akıllı Ev Otomasyonunda Kontrol Edilen Parametreler

Akıllı ev otomasyonunu insanı gündelik hayatta etkileyen parametreleri kontrol edilerek daha az stresli, mutlu ve huzurlu bir şekilde yaşamalarını hedefler. Bu parametreler insanların günlük hayattaki iş yaşamlarını, aile hayatlarını doğrudan veya dolaylı olarak etkileyen; sıcaklık, nem, güvenlik, yangın, ışıklandırma, konfor ve benzeridir.

2.2.1 İklimlendirme (HVAC - Heating Ventilating Air Conditioning)

İnsanlar yaşamları boyunca kendi vücut ortamlarına uygun sıcaklıktaki alanlarda yaşamaya çalışmışlar veya yaşadıkları ortamları kendi vücutlarının sıcaklığına uygun hale getirmeye çalışmışlardır. Bu nedenle akıllı ev sistemlerinde temel olarak görülen parametrelerden biri sıcaklık kontrolüdür. Sıcaklığın yanında nem, havalandırma da kontrol edilebilir. Sıcaklık, nem, havalandırmanın birlikte kontrol edilmesi iklimlendirme olarak adlandırılır. Akıllı evlerde iklimlendirme kontrolü her zaman ön planda olmuştur. İklimlendirme kontrolü ile hem optimum sıcaklık, nem, havalandırma değerleri sağlanmış olur hem de ekonomik açıdan tasarruf yapılmış olur.

2.2.2 Işık

Akıllı evlerde kontrol edilen ikinci önemli parametre de ışıktır. Işık şiddetinin değeri insan görme duyusu için önemli bir parametredir. Işık şiddetinin değeri yeterli seviyenin altında olması durumunda göz sağlığının bozulması, iş hayatında performans düşüklüğü gibi sorunlara neden olmaktadır. Işık kontrolünde çeşitli yöntemler kullanılabilir. Sensörler ile ortamın ışık şiddeti ölçülerek seçilen bir algoritma yardımı ile kontrol yapılabilir. Başka bir kontrol yöntemi de zamana bağlı olarak yapılan kontroldür. Bu şekilde yapılan kontrolde daha önceden ayarlanmış zaman dilimlerine göre ışıklar yanmaktadır veya sönmektedir. Akıllı

evlerde yapılan ışık kontrolü ile enerji için yapılan ödemelerde bir tasarrufa gidilmiş olur.

2.2.3 Güvenlik

Güvenlik parametresinin kontrolü akıllı evlerde çeşitli yollarla sağlanabilir. Eve herhangi bir izinsiz giriş olduğunda ev sahibine, polise SMS, MMS, vb iletişim yolları ile haber verilebilir. Aynı şekilde eve izinsiz müdahalede bulunulduğunda evin boş olduğunun anlaşılmaması için ışıklar otomatik olarak caydırma amaçlı yakılabilir veya evdeki müzik sistemi devreye sokulabilir.

2.2.4 Yangın

Yangına müdahalede zaman çok önemlidir. Evde insan olduğunda yangının fark edilmesi uzun sürebilir. Ev boş iken bu süre çok daha fazla olabilir. Akıllı evlerde bulunan yangın veya duman detektörleri sayesinde yangına müdahale zamanı minimuma indirilmiş olur. Duman detektörleri duman algılandığında otomatik olarak fıskiyeler devreye girer ve daha önceden hazırlanan algoritma sayesinde itfaiyeye haber verilmektedir.

2.2.5 Uzaktan Kontrol

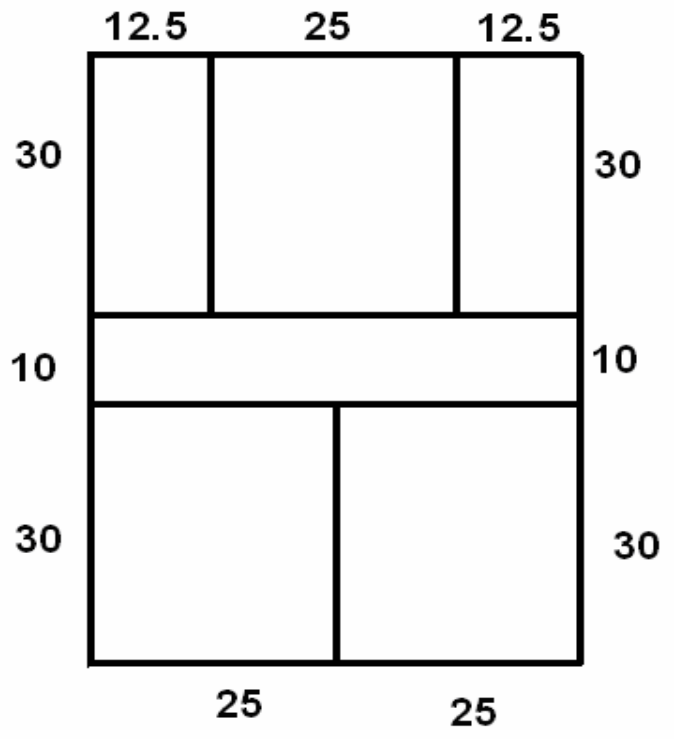
Akıllı yapılarda haberleşme sistemin önemli bir parçasıdır. Haberleşme, sistemin kolları gibi davranarak, kullanıcının veya sistemin göremediği noktalara (uzak noktalara) ulaşarak, o noktalardaki sistemler hakkında bilgi alınmasını sağlar. Uzaktan kontrolün temeli haberleşmedir. Ethernet, Internet, GPRS, SMS**,** WIFI**,** RF**,** RS232**,** RS485 gibi birçok haberleşme protokolü yardımı ile kullanıcının sisteme ulaşması veya ana sistemin uzak noktaya ulaşması sağlanır.

2.2.6 Elektronik Cihazların Kontrolü

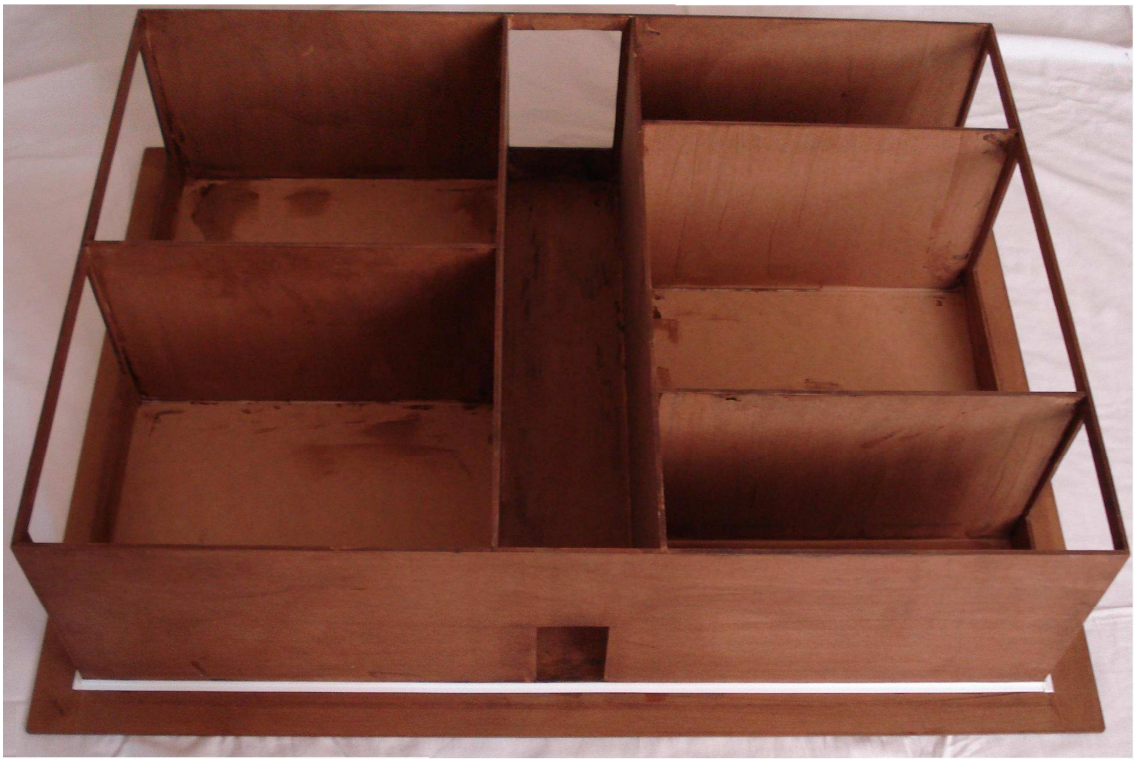
Bir ev içerisinde birden çok elektronik cihaz bulunmaktadır. Bu cihazları kullanıcı belirli bir merkezden yönetmek isteyebilir. Bu cihazlara, su ısıtıcısı, televizyon, DVD, müzik sistemi, vb. örnek olarak verilebilir. Elektronik cihazların kontrolünde merkezden kullanıcının el ile müdahale etmesi veya daha önceden yazılmış senaryo ile kontrolü sayılabilir.

3. ADuC841 MİKRODENETLEYİCİSİ İLE AKILLI EV OTOMASYONU

Bu tezde akıllı ev otomasyonu gerçekleştirilirken yazılan algoritma mikrodenetleyici kullanılarak hayata geçirilmiştir. Ev otomasyon sisteminin model üzerinden gerçekleştirilmesinde bir adet ana pano ve buna bağlı olarak çalışan oda panolarının kontrolü esas alınmıştır. Ana pano evin giriş kapısına yani hole yerleştirilmiştir. Sistemin çalışmasında genel prensip her bir odaya ait olan pano ana pano ile haberleşerek kontrol sağlamasına dayanır. Oda panolarındaki kontrol kartlarında aydınlatma, sıcaklık ve güvenlik kontrolü yapılmaktadır. Ana panoda ise kullanıcı tanıma, şifreleme, izleme işlemleri yapılmaktadır. Uygulama sırasında örnek bir ev modeli oluşturulmuştur. Bu ev modelinde geleneksel Türk ev planı örnek alınmıştır. Model olarak kullanılan ev eş büyüklükte iki oda, bir adet daha küçük oda, bir banyo, bir mutfak ve bir holden oluşmaktadır.



Şekil 3.1 Ev modelinin ölçeklendirilmiş çizimi



Şekil 3.2 Ev modelinin resmi

3.1 Tasarım Esnasında Seçilen MCU

8051 ailesi, INTEL firması tarafından 1980'lerin başında piyasaya sunulan dünyanın en popüler 8-bit mikrokontrolör ailesidir. INTEL 'den sonra, bu MCU (Micro Controller Unit) ailesi ile uyumlu olarak, başta PHILIPS, SIEMENS, ATMEL, DALLAS, OKI, HYUNDAI/HYNIX, WINBOND olmak üzere pek çok üretici firma türev işlemciler üretmiştir. Bunlardan başka kendi özgün mikrokontrolör ailelerini üreten ST, TEXAS INS. gibi çeşitli büyük üreticiler bile 8051 uyumlu MCU lar geliştirmiş ve pazara sunmuştur.

Internet'de yapılan bir araştırmada 55'in üzerinde 8051 üreticisi belirlenmiştir. Bu firmalara ait bir liste bölüm sonunda yer almaktadır. KEIL, IAR, NOHAU, TASKING vb başta olmak üzere çok miktarda diğer firma ise geniş bir donanım ve yazılım geliştirme araçları desteği sunmuştur. Bunun sonucu olarak 8051 ailesi, 1980'lerden bugüne bir endüstri standardı olmuştur.

8051 ailesi bazen MCS–51 ailesi olarak da anılmaktadır. 8051 ve MCS–51 tanımlamaları aynı aileyi belirtmek için kullanılır. Ayrıca "8051" bu ailenin ilk üyelerinden biri olan "Mask ROM" lu modelin de adıdır.

Bugün için çeşitli 8-bitlik mikrokontrolör aileleri arasında 8051 ailesi, gelişmiş türev

ürünleriyle beraber yaklaşık %40 gibi bir piyasa payına sahiptir. 8051 ailesinin başlıca özellikleri aşağıda maddeler halinde verilmiştir.

1 - Geniş Yelpaze: Pek çok üretici firma, orijinal 8051'e çeşitli ek özellikler katarak türev ürünler geliştirmiştir. Her bir üretici firmanın onlarca, hatta bazılarının elliden fazla türev ürünü olduğu dikkate alınırsa ne kadar geniş bir aileden söz edildiği daha rahat anlaşılabilir. Bütün bu ürünler için çeşitli yazılım ve uygulama geliştirme donanımları üreten firmaların da katkılarıyla 8051 bir endüstri standardı haline gelmiştir. Yeryüzünde "Industry Standard" tanımlamasına sahip tek 8-bitlik mikrodenetleyici ailesidir.

2 - Uyumluluk: Çok değişik 8051 türev ürünler bulunmasına rağmen komut seti ve mimari yapı olarak bütün ürünler uyumludur (code compatible). Diğer mikrodenetleyici aileleri, 8051'in sunmuş olduklarını farklı ve uyumlu olmayan işlemcilerle (genellikle tek üretici firma kaynaklı olarak) sağlayabilmektedirler. Bu uyumluluk, kolaylık ve esneklik, program geliştirme araçlarında, eğitiminde ve yazılım desteğinde de bulunmaktadır.

3 - Hız ve Güç:8051 çekirdek mimarisi kontrol uygulamaları için gayet uygun olup hızlı ve güçlüdür. Piyasaya ilk sunuldukları tarihte 12MHz lik modelleri yaklaşık olarak 1 MIPS de (Mega Instruction Per Second) çalışıyor iken günümüzde 24 MIPS, 50 MIPS hatta 100 MIPS'lik hızlara sahip olan türev işlemcilere sahiptir. Bu performans ile 1 makine çevrimlik bir komutu 40ns veya 20ns, hatta 10 ns. gibi bir sürede icra eder. 8-Bitlik işlemci aileleri arasında bu hıza sahip genel bir işlemci ailesi bulunmamaktadır.

4 - Popülerlik: 8051 kullanıcıları için birçok kitap, teknik dokümanlar, yazılım ve donanım gereçleri, pek çok İnternet Web Sayfası mevcuttur. Ürün kolay bir şekilde bulunmakta ve yaygın bir şekilde desteklenmektedir. Eğitim notlarının sonunda 8051 MCU üreticileri, 8051 ailesi için geliştirme araçları donanım ve yazılımları üreten firmalara ait irtibat bilgileri ayrıca çeşitli internet web sitesi adresleri yer almaktadır.

5 - Sürekli Geliştirilme: 1980'lerden bugüne silikon ve tasarım olarak sürekli geliştirilen 8051'lerin hızları, işlem güçleri, on-chip çevre birimleri sayısı ve çeşitliliği artmıştır. Örneğin Analog Devices firması tarafından üretilen bir ürün (ADUC845) Standart 8051 mimarisinde yer alan özelliklerin yanı sıra:

-10 kanal 24 bit rezolüsyona sahip 10 kanal ADC,

-Programlanabilir Gain Amplifier,

-12 bit voltage output DAC,

-Dual PWM çıkışları,

-Voltage reference,

-Current Source,

-Temperature sensor

-Power supply monitör,

-Power-on reset,

-PLL,

-62KB on-chip Flash ROM (In system & In Application Programmable),

-On-chip download / debug interface,

-On-chip 256 Byte + 2KB data RAM,

-4KB data EEprom,

-16MB external data RAM address space,

-UART, SPI, I2C 3 kanal serial interface,

-3 kanal 16-bitlik timer/counter,

-Timer Interval Counter (Real Time Clock),

-Watchdog timer,

-Baud rate generator timer,

-11 Interrupt Vector

gibi özelliklere de sahiptir.

3.1.1 8051/8052 Tarihçesi

Intel firması 1968 yılında hafıza entegre devreleri (Integrated Circuit / tümdevre) yapmak üzere kuruldu. Üretecekleri bir hesap makinesi için CPU entegresi isteyen, hesap makinesi üreten bir firmanın talebi ve üretecekleri bir terminal için özel bir tümdevre isteyen, diğer bir firma Datapoint Corporation'ın isteklerini karşılamak için, Intel firması 4004 (1971) ve 8008 (1972) CPU'larını tasarladı.

Mikroişlemciler ve mikrobilgisayarların sınıflandırılmasında en temel ölçü, mikroişlemcinin entegre üzerinde (On-Chip) işlem yaptığı en uzun verinin bit sayısı, yani kelime uzunluğudur (word length). 4-bitlik işlemci olan 4004 ve 8-bitlik işlemci olan 8008'den başlayarak, mikroişlemciler ve mikrobilgisayarlar için 4-bit, 8-bit, 16-bit, 32-bit, 64-bit gibi veri uzunluk standartları doğmuştur.

Intel, bu ilk ürünlerini başlangıçta sadece o müşterileri için hazırlamıştı. Fakat ilk siparişleri veren firmalar ürünleri kullanmamaya karar verince, piyasaya tanıtım yapıldı. Ciddi bir satış potansiyeli olduğu görülmesi üzerine, aynı zamanda 8008'in 16K'lık hafıza limitini aşmak amacıyla, Intel firması (1974) yılında genel-amaçlı 8080 CPU'sunu üretti. Şaşırtıcı bir şekilde yüksek ilgi gören mikroişlemciler hızla yaygınlaşmaya başladı ve kısa süre içinde 8080, 8-bit mikroişlemcilerde endüstri standardı oldu.

Diğer yarı iletken üreticileri de kendi ürünlerini piyasaya sürdüler, ancak bunların hepsi başaralı olamadı. Başarılı olanlar arasında MOS Technologies'in 6502'si (Apple II bilgisayarlarda kullanıldı), Motorola 6800 ve Zilog Z80 anılabilir. Intel, bir kaç yıl sonra gelişmiş 8-bitlik 8080 işlemcisi olarak adlandırılabilecek, CPU ve çeşitli çevre birimlerinin tek çip üzerinde gerçekleştirildiği, halen yaygın kullanımda olan güçlü bir komut setine sahip, günümüz modern mikrokontrolörlerinin atası 8051'i satışa sundu.

Intel 1978 yılında ilk 16-bit mikroişlemcisi olan 8086'yı üretti. 8086 daha önceki 8080/8085 ürününe bazı yönlerden benzemesine rağmen, iki işlemci ailesi birbiri ile uyumlu değildi. Bir yıl sonra 1979'da üretilen, 8086'nın 8–bit veri yoluna sahip sürümü olan 8088, 1981 yılında üretilen IBM PC mikrobilgisayarının ilk işlemcisi olmuştur. Kısa sürede endüstrinin 16-bit mikroişlemci standardı olan 8086/8088, günümüze kadar uzanan pek çok değişik ürünüyle, x86 ailesi diye adlandırılan çekirdeği(core) oldu.

8086

→ 80286 → 80386 → 80486 → PENTIUM →… … …

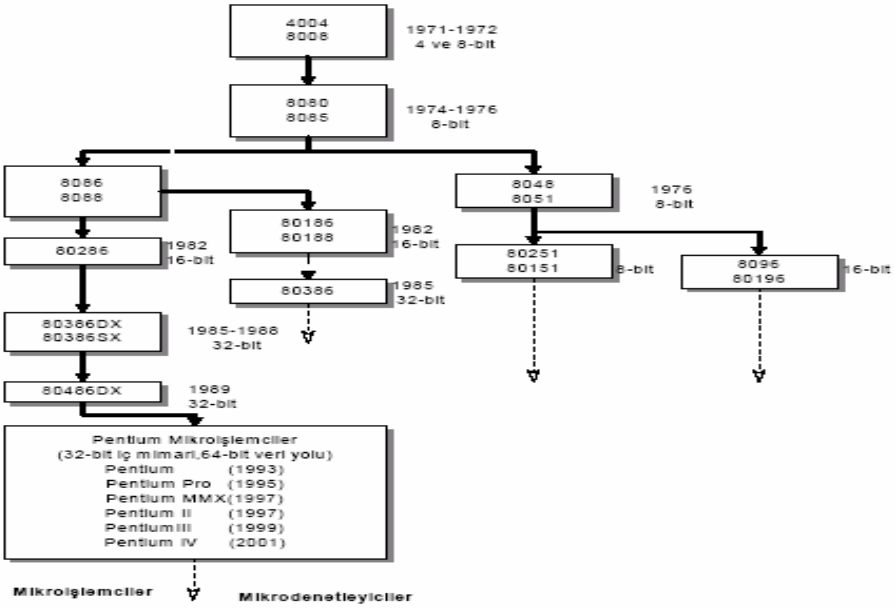
8088

Şekil 3.3 x86 ailesinin çekirdeği

Daha sonraki yıllarda x86 ailesinin diğer ürünleri,80286, 80186 ve 80188 üretildi. 80186 işlemcisi 8086'nın tümdevre-üzere çeşitli çevre birimlerine sahip olan sürümüdür. 80188 işlemcisi ise, 8-bit veri yoluna sahip bir 80186 işlemcisidir. Tasarımlarda fazla çevre birimi

istemeyen 80186/80188 işlemcilerinin, genelde değişmez bir programla, kontrol uygulamaları içinde yer alarak mikrodenetleyici gibi kullanılmaları amaçlanmıştır. Bu iki işlemci yaygın olarak kullanılmıştır.

Uygulama çeşitlerine göre Intel mikroişlemcilerinin sınıflara ayrılması, 80186/80188 ve 8048/8051 işlemcilerden sonra başlamıştır. Genel olarak Intel mikroişlemcileri bugün tekrar programlanabilir mikroişlemciler (Genel Amaçlı İşlemciler) ve mikrodenetleyiciler (Özel Amaçlı İşlemciler) olarak ikiye ayrılır. Intel mikroişlemcileri ve mikrodenetleyicilerinin gelişimi aşağıdaki Şekil 3.4 ve Çizelge 3.1 de verilmiştir.



Şekil 3.4 8051 aile ağacı

8086/8088 işlemcilerinin 1 megabyte hafıza ile sınırlı adres alanı,1980'lerin ortalarına doğru birçok uygulama için ciddi bir problem olmaya başlamıştı. Bu yüzden Intel x86 çekirdeğinin bir üst uyumlu sürümü olan 80286 işlemcisini üretti. Bu işlemci, 16 megabyte'lık adres alanıyla beraber 8086/8088 komut kümesine sahipti. 80286, IBM PC/AT ve orta model PS/2 bilgisayarlarında kullanıldı ve daha önceki 8088 gibi büyük bir başarı kazandı.

Intel için bir sonraki adım, 1985 yılında üretilen, bir entegre devre-üzerinde gerçek 32- bit CPU olan 80386DX oldu. 80286 gibi bu mikroişlemci de çok yaygın olarak kullanıldı. 1988 yılında, harici 16-bit veri yoluna sahip 80386SX işlemcisi üretildi.

80486, 80386'nın bir üst uyumlu modeliydi. Bütün 80386 programları, 80486 makinelerinde bir değişiklik yapılmadan çalışabilecekti. Bu iki işlemci arasındaki temel fark, 80486'nın

80386'ın özelliklerine ek olarak, yardımcı işlemcisi olan bir kayan-nokta birimine (Floating Point Unit-FPU), 8 kilobyte ön hafıza (cache) ve bir hafıza yönetim birimine tümdevre-üzerinde sahip olmasıdır. Ayrıca 80486, 80386'dan çok daha hızlıdır.

Çizelge 3.1 İntel işlemcilerin yıllara göre üretimleri ve özellikleri [18]

İşlemci	Yıl	Saklayıcı/Veri YoluGenişliği	Adres Alanı	Önemli Özellikler
4004	1971	4\4	640 byte	İlk mikroişlemci,2300 transistör,10 mikron
8008	1972	8\8	16K	İlk 8-bit işlemci,108Khz
8080	1974	8\8	64K	İlk genel amaçlı CPU, 6 mikron,6000 transistör
8085	1976	8\8	64K	Gelişmiş 8080,6200 transistör
8086	1978	16\16	1M	İlk 16-bit CPU, 5-10MHz,29000 transistör,3 mikron
8088	1979	16\8	1M	1981'de üretilen IBM PC'deki ilk işlemci,8-bit veri yolu 8086
80186	1982	16\16	1M	8086+I/0
80188	1982	16\8	1M	8088+ I/0
80286	1982	16\16	16M/1G	134000 transistör, 1.5 mikron,IBM PC/AT'nin ilk işlemcisi
80386DX	1985	32\32	4G/64T	Intel'in ilk 32-bit işlmecisi,275000 transistör,1 mikron
80386SX	1988	32\16	4G/64T	80286 yoluna sahip 80386
80486DX	1989	32\32	4G/64T	80386+FPU+cache,1.2 milyon transistör,0.8 mikron
Pentium	1993	32\64	4G/64T	3.1 milyon transistör,0,8 mikron,60-200 MHz, superscalar mimari
Pentium	1995	32\64	64G/64T	5.5 milyon transistör,0.32 mikron, tümdevre üzeri L2 cache, P6 mimarisi: çoklu dallanma tahmin,

Pro				veri akışı analizi ve tahmini yürütme
Pentium MMX	1997	32\64	4G/64T	4.5 milyon transistör, multi-medya ekleri
Pentium II	1997	32\64	64G/64T	7.5 milyon transistör, 0.25 mikron, 300-450 MHz, MMX+ Pro Teknolojisi
Pentium III	1999	32\64	64G/64T	9.5 milyon transistör, 0.18 mikron,450-500 MHz, 3D grafikler ve daha fazla multi-medya desteği

1993 yılında piyasaya sürülen Pentium, temel mimari olarak çok farklı bir mikroişlemci olmayıp, Intel' in her 2-3 yılda bir ürettiği yeni bir x86 işlemcisidir. Bu yapı IA-32 (Intel Architecture) olarak belirtilen 386/486 ile başlayan 32-bit mimarinin bir uzantısıdır.

3.1.2 Genel 8052 Mikrodenetleyicisinin Özellikleri

8051 mikrodenetleyici ailesinin başlıca özellikleri aşağıda verilmiştir;

• Kontrol uygulamaları için optimize edilmiş 8 bitlik CPU

• Genişletilmiş Boolean işleme komutları (tek bitlik lojik komutlar)

• On-chip program hafızası (Program ROM)

• On-chip veri hafızası (Data RAM)

• 4 adet 8 bitlik I/O portu

• Çift yönlü kullanılabilen ve tek tek adreslenebilen I/O pinleri

• 3 kanal 16 bitlik Timer/Counter

• Full Duplex UART (Seri haberleşme kanalı)

• Çok kaynaklı / vektörlü / öncelik seviyeli kesme yapısı (Interrupts)

• On-chip saat osilatör

3.1.3 8052 Türevi Mikrodenetleyiciler

Aşağıda 8051/8052 mimarisini kullanarak mikroişlemci üreten firmaların bazıları

bulunmaktadır. Aynı zamanda bu firmalar 8051/8052 yapısına kendi bünyelerinde yeni özelliklerde eklemişlerdir.

Çizelge 3.2 8051 üreten firmalar ve chipleri [18]

NO:	Firma:	İnternet Adresi:	Genel Özellikler
1	ACER LABS	http://www.ali.com.tw/	8051
2	AEROFLEX	http://www.utmc.com/	8051+Flexible Core Clock İşlemi(1Hz-20Mhz'e harici Clock),(2MHz-20MHz Dahili osilatör)
3	AMD	http://www.amd.com/	8051+Kablosuz Telefon Chip seti
4	ANALOG DEVİCE	http://www.analog.com/	8051+ yüksek çözünürlüklü ADC
5	ATMEL	http://www.atmel.com/	8051
6	CAST	http://www.cast-inc.com	8051+ ortalama 8 kat daha hızlı çekirdek yapısı+ onchip debug
7	CHIPCON	http://www.chipcon.com	8051+ yüksek frekanslı RF Tranciever
8	CONTROL CHIPS	http://www.controlchips.com	Standart 8052
9	CML	http://www.cmlmicro.com	8051+LCD Kontrol Ara yüzü+ İntegral Modem+A/D converter+8*16 klavye+2 Adet PWM Çıkışı
10	CYBRA TECH.	http://www.cybratech.com	8051
11	CYGNAL	http://www.cygnal.com	8051+128Kbyte'a kadar Flash Memory+5 adet 16 bit Timer+ USB 2.0

12	CYPRESS	http://www.cypress.com	8051+Full speed USB bağlantısı
13	DAEWOO	http://www.daewoosemicon.co.kr	8051+2 kademe programlanabilir seri port+ devre üzerinde emülasyon
14	DALLAS	http://www.dalsemi.com	8051+ dual DPTR+ OnChip-Debug+ 16Kbyte EPROM+kristal frekansını 64'den 1024'e kadar bölerek çalışma+ 2 adet seri port+
15	DOLPHIN	http://www.dolphin.fr	8051+ ortalama 9 kat daha hızlı çekirdek yapısı
16	DOMOSYS	http://www.domosys.com	8051+ ortalama 8 kat daha hızlı çekirdek yapısı+ 8 bit A/D conveter ara yüzü SPI'dan
17	GOAL	http://www.goalasic.com	8051+2 capture ve yakalma ünitesi+4 PWM çıkışı+ 2 seri UART+ 4 kanal ADC
18	GOLDSTAR		
19	HONEYWELL	http://www.ssec.honeywell.com	8051+900Hz RF tranciever
20	HYNIX	http://www.hynix.com	8051+ 40MHz'e kadar hız+USART
21	ICSI	http://www.icsi.com.tw/english/	
22	INFINEON	http://www.infineon.com/	8051+ 29 kanal PWM çıkışı+USART+4 kaynaklı 19 interrupt
23	INNOVASIC	http://www.innovasic.com	8051+48MHz'e kadar hız+3.3V ile besleme
24	INTEL	http://www.intel.com	8051
25	ISSI	http://www.issi.com/	8051+iki level öncelik kademeli Flash Memory

26	MACRONIX	http://www.macronix.com/	8051+ dokunmatik ekran controller
27	MAXIM		
28	MICRONAS	http://www.micronas.com	8051+Teletext decoder+tek karakter seti++batı ,doğu avrupa,fars,arap tabanlı dil+programlanabilir ekran boyutu+Ekran kaydırma donanımı+ 14 bit iki kanal PWM
29	MICROTUNE	http://www.microtune.com	8051+bluetooth+ 2GHz frekans tranciever+USB 1.1
30	MENTOR GRAPH.	http://www.mentor.com	8051+On-Chip debug
31	MOSEL-VİTELİC	http://www.moselvitelic.com	8051+LCD'li Voice Smart+ Mouse Controller
32	MYSON CENTRY	http://www.century-semi.com	8051+dijital video encoder+çeşitli monitör tiplerine emülasyon+ üç öncelik kademeli 8 interrupt kaynağı
33	NORDIC	http://www.nvlsi.no/	8051+ 2.4 GHz ISM band radio+ 9 kanal 12 bit ADC
34	OREGANO	http://www.oergano.at	Full senkron devre dizaynı
35	OKI	http://www.oki-europe.de/	8051
36	PHILIPS	http://www.philips.com	
37	RDC	http://www.rdc.com..tw	8051+8 bit IP+ 16 bit network işlemcisi
38	QUICKCORES	http://www.quickcores.com	8051+ Evolution Board
39	SAMSUNG	http://www.intl.samsungsemi.com/	
40	SANYO	http://www.sanyo.com/	8051+10-12 bit ADC Converter

41	SHARP	http://sharp-world.com	8051+ 4 kanal DMA Controller+ üç kanal programlanabilir TIC+ Floppy disk controller
42	SİLİCON STOR.	http://www.sst.com/	
43	SILICONIAS	http://www.siliconias.com	8051
44	SMSC	http://www.smsc.com/	
45	ST	http://www.st.com	8051+ 128Kbyte Flash Memory+ 32 Kbyte Flash Memory+DDC+LVD+ 8Kbyte Battery Backup+PLD+USB1.1+DDCPROM+ keypad ara yüzü+üstün entegre(sadece enerji ve kristal eklenir)
46	SYNCMOS	http://www.syncmos.com.tw	8051+ 40Mhz Clock Frekansı
47	TDK	http://www.tdksemiconductor.com	8051
48	TEMIC	http://www.temic-semi.de	
49	TEXAS INS.	http://www.ti.com	8051+ 8 kanal analog giriş 24 bit çözünürlük+ burn out sensor +16 bit PWM+ 21 interrupt kaynağı
50	TRISCEND	http://www.triscend.com	8051+üç harici 10 dahili interrupt+40Mhz Core+ 10MIPS
51	WINBOND	http://www.winbond.com	8051+1280 byte RAM+64Kbyte ROM+ 1.8V ile 5.5V arası besleme+PWM+ Extra I/0+ wait state control signal
52	WINEDGE	http://www.winedge.com.sg	8051+12 kat hız +hızlı interrupt tepki+ (12+2) interrupt kaynağı + 2 DPTR +4 Kbyte Data RAM

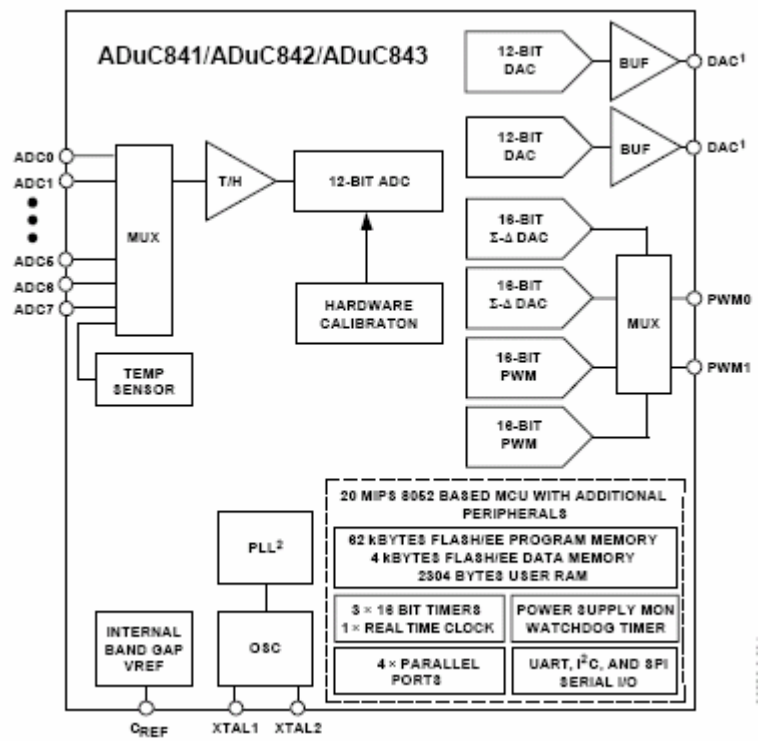
53	WELTREND	http://www.well/trend.com.tw	8051+SYNC sinyal işlemcisi H/V ayrımlı+SYNC sinyal çıkışı (ayrı çalışan)+harici IRQ+14 kanal 8 bit PWM+CRD ve LCDler için karakter görüntüleme+ LCD kontrolü+USB 1.0
54	VERSACHIPS	http://www.vchips.co.kr	8051+ 40 DIP+ 4 kademeli interrupt önceliği+6 interrupt kaynağı+ A/D converter

3.1.4 Tasarlanan Sistemde Kullanılan ADuC841

Tasarlanan sistemde mikroişlemci seçimi yaparken öncelikle işlem hızının yüksek olmasına ve ADC çözünürlüğünün yüksek olması dikkate alındı. Tasarımda kullanılan ADuC841 mikrodenetleyicisi Analog Devices tarafından 8052 ana yapısı üzerine özellikle 8 kanal 12 bit 420 kSPS (kilo sample per second) ADC eklenmiş bir modelidir. Tasarımda ADuC841'in 8 kanal ADCsi kullanılmıştır. Bu sekiz kanal ADC ile oda sıcaklığı üç farklı sensörden, odanın içindeki ışık şiddeti üç farklı noktadaki sensörden ölçülmüş ve kullanıcı ayarlarının girilmesi için de iki ADC kanalı kullanılmıştır.

ADuC841 mikrodenetleyicisinin 8052 yapısı üzerine eklediği PLL (Phase Lock Loop) yapısı ile 8052 ana yapısına 20 MIPS (Million Instruction Per Second) işlem yapabilme özelliği eklenmiştir. Bu özelliği sayesinde ADuC841 ile işlemler hızlı bir şekilde gerçekleştirilmiştir [17].

ADuC841 8052 den farklı olarak üzerinde 2304 byte RAM bulundurmakta ve 62 Kbyte a kadar çıkabilen program hafızasını içinde barındırmaktadır. Büyük RAM ve hafıza alanı yazılım için daha rahat, daha kullanışlı bir geliştirme ortamı sağlamaktadır. Ayrıca mikrodenetleyici üzerinde bulunan on-chip debug özelliği ile mikrodenetleyicinizin üzerinde olduğu karttan, gerçek ortam içinde rahatlıkla hataları ayıklayıp, işlemlerinizi düzenleyebilir [17].



Şekil 3.5 ADuC841 Mikrodenetleyicisinin İç Yapısı [17]

3.2 Ana Pano

Hazırlanan ev otomasyonunda odalardaki ana kontrol işlemleri odalarda bulunan oda kartları üzerinde gerçekleştirilmektedir. Bu oda kartları birbirinden bağımsız olarak çalışmaktadır. Bu oda panoları ana panoya bilgi göndererek, durumlarını belirtirler. Kullanıcı eve girdiğinde odalardaki durumları ana pano üzerinde görebilmektedir.

Ana pano evin girişinde yer almaktadır. Kullanıcı anahtar ile kapıyı açtıktan sonra ana pano üzerinde bulunan tuş takımına şifresini girmek zorundadır. Girilen şifreye göre ilgili oda panosu ve ortak alanlar aktif olmaktadır. Ana pano oda panolarından aldığı bilgilere göre odalardaki sıcaklık ve ışık bilgisini LCD ekranında göstermektedir.

3.2.1 Ana Panoda Gerçekleştirilen İşlemler

3.2.1.1 Kullanıcı Tanıma

Kullanıcı eve giriş yaptıktan sonra ana pano üzerinde bulunan tuş takımını kullanarak kendisine ait olan şifreyi girmesi gerekmektedir. Eğer 30 saniye içerisinde şifreyi giremez ise güvenlik kontrolü aktif olur. Evde yaşayan kullanıcı sayısına göre önceden ayarlanmış şifreler bulunmaktadır. Kullanıcı eve girdiğinde şifresini girerek kendisine ait olan odanın ve ortak alanların aktif edilmesini sağlar. Tuş takımı olarak 16 tuşlu bir donanım tercih edilmiştir.

Şekil 3.6 Tuş takımı

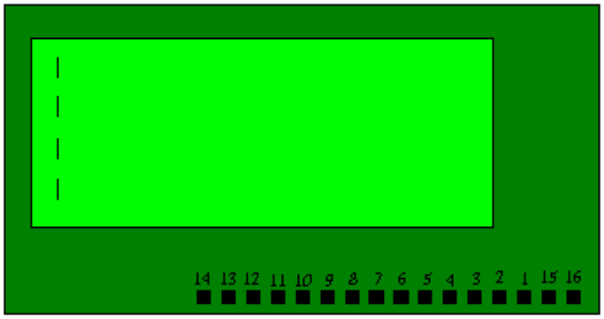
Kullanıcı önce Px tuşunda basarak hangi kullanıcı olduğunu belirtir ve ardından sayı tuşları yardımıyla kendine ait olan şifreyi girer.

3.2.1.2 Tuş Takımı Okuma

Tuş takıma okumada matris tuş okuma yöntemi tercih edilmiştir. Bu tuş takımı okuma yönteminde sütunlar ve satırlar bulunmaktadır. Tuş tarama işlemi yapılırken sütunlar sıra ile sürülür ve bu sürme işleminden sonra aktif olan satıra bakılarak basılan tuş algılanır. Bu algılamada Bouncing denilen, sinyalin tuşa basıldıktan belirli bir süre için gürültülü şekilde algılanmasından dolayı bir Bouncing bekleme süre kullanılmıştır. Bu süre sonunda tekrar aynı satır ve sütun okunarak basıldı bilgisi algılanmış olur.

3.2.1.3 Gösterge Ünitesi

Odalardan alınan oda sıcaklık bilgisi, oda set değer bilgisi, oda ışık şiddeti bilgisi ana pano üzerinde bulunan gösterge ünitesinde kullanıcıya gösterilmektedir. Gösterge ünitesi 4 satır LCD (Liquid Crystal Display) ekrandan oluşmaktadır.



Şekil 3.7 LCD resmi

3.2.1.4 Güvenlik Kontrolü

Güvenlik kontrolünde oda modüllerinden gelen pencere açıldı bilgisi kullanıcının evde olup olmama durumuna göre işlenir. Eğer pencere açıldı bilgisi evde kullanıcı yokken geldiyse eve izinsiz giriş var demektir. Bu izinsiz giriş alarm röle çıkışını aktif eder. Bu röle çıkışına ışık veya alarm ünitesi bağlanabilir.

3.2.1.5 Oda Panoları İle Haberleşme

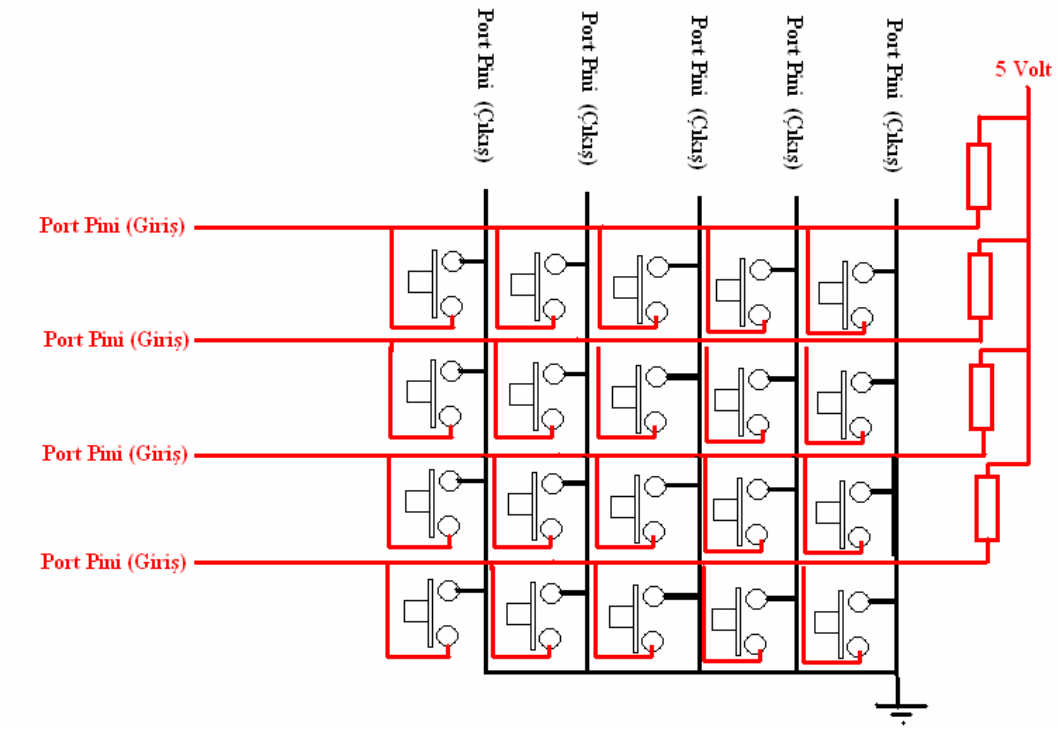
Oda panoları ile haberleşmede RS485 haberleşme protokolü kullanılmıştır. Oda panolarında pencerenin açılıp açılmadığı bilgisi, odanın sıcaklık bilgisi, oda sıcaklık set değeri ve oda ışık değeri gönderilmektedir. Ana panodan oda modüllerine gönderilen bilgi ise eve kullanıcının girdiği ve hangi odanın aktif olacağı bilgisidir.

3.2.1.6 Ana Pano Donanımı

Tasarlanan sistemde ana pano donanımında tuş takımı okuma, LCD sürme ve güvenlik kısımları bulunmaktadır.

3.2.1.6.1 Tuş Takımı Okuma

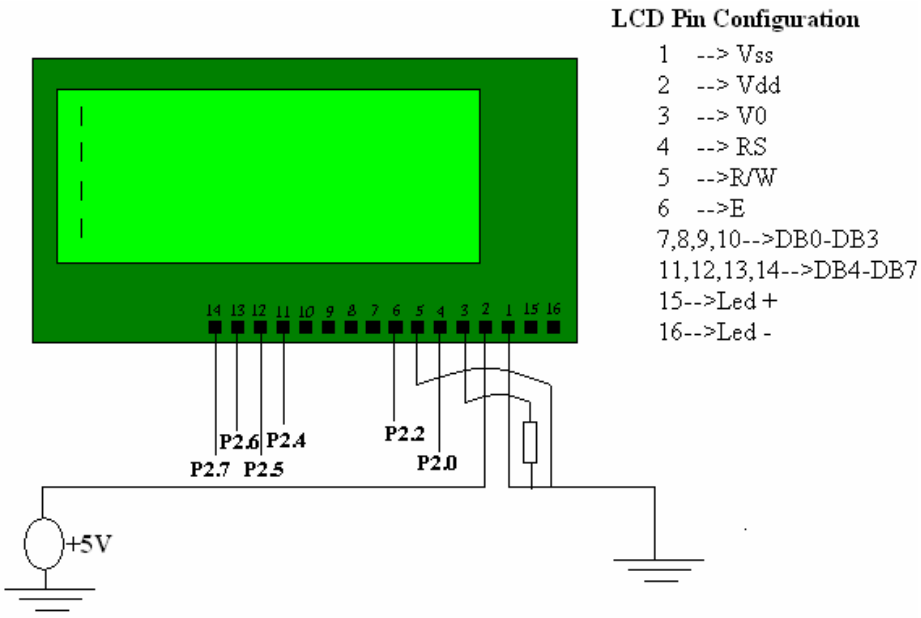
Kullanıcı eve girdiği zaman tuş takımı yardımı ile şifresini girmektedir.



Şekil 3.8 Tuş takımı sürme devresi

3.2.1.6.2 LCD Sürme

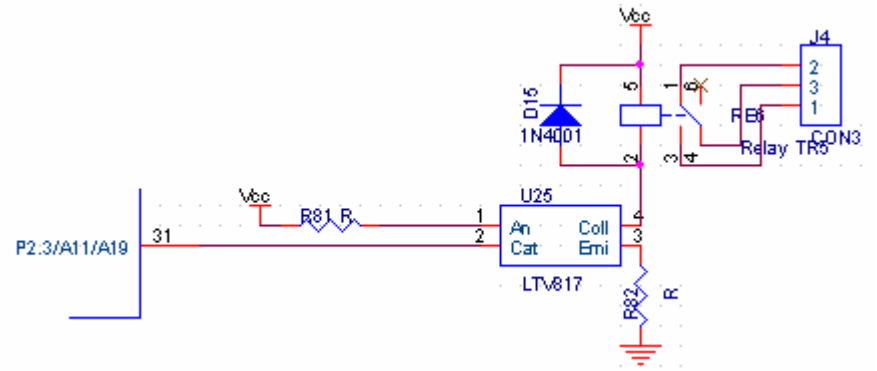
Gösterge odalardaki sıcaklık ve ışık bilgilerinin ana panoda gösterilmesi için kullanılır.



Şekil 3.9 LCD sürme devresi

3.2.1.6.3 Güvenlik Donanımı

Tasarlanan sistemde güvenlik donanımı kısmı için bir röle ve bu rölenin çıkışında bir LED kullanılmıştır.



Şekil 3.10 Röle bağlantı şeması

3.2.1.7 Ana Pano Yazılımı

Tasarlanan sistemde ana pano yazılımında tuş takımı okuma, LCD sürme ve güvenlik yazılım kısımları bulunmaktadır.

3.3 Oda Panosu

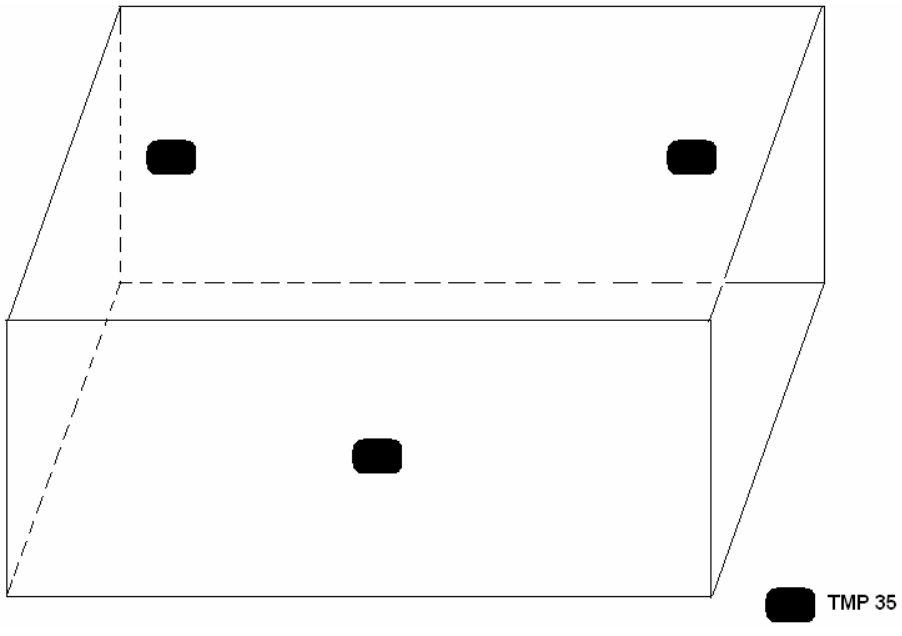
3.3.1 Oda Panosunda Gerçekleştirilen İşlemler

Hazırlanan ev otomasyonunda her bir oda için ayrı bir kontrol kartı düşünülmüştür. Kullanıcı her bir odayı ayrı ayrı kontrol edebilir. Odalar ortak kullanım alanları olan odalar ve kullanıcıya özel olan odalar olarak ikiye ayrılır. Her bir oda için ayarlanan başlangıç değerleri, istenilen değerler farklı olabilir. Oda panosunda gerçekleştirilen ana işlemler sıcaklık kontrolü, aydınlatma kontrolü ve güvenlik kontrolüdür. Ayrıca oda panoları ana pano ile de haberleşmektedirler.

3.3.1.1 Sıcaklık Kontrolü

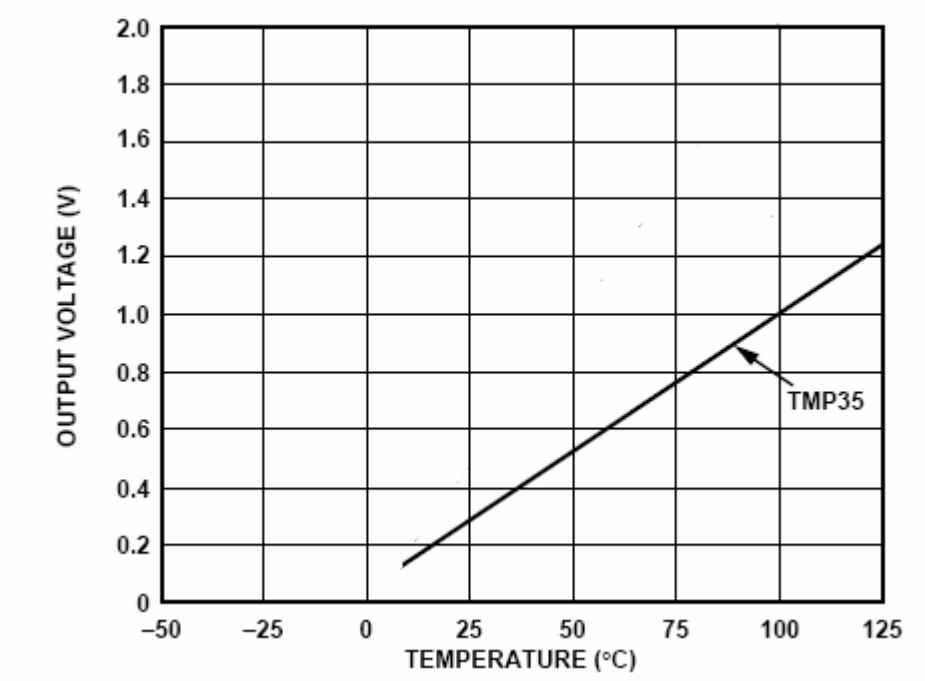
Her odada bulunan mikroişlemci o odanın sıcaklığını kontrol eder. Sıcaklık kontrolünde birçok algoritma kullanılabilir. Bu tez çalışmasında sıcaklık kontrolünde bulanık mantık algoritması tercih edilmiştir. Oda içerisinde bulunan sensörler sayesinde odadan geri besleme alınarak kapalı çevrim kontrol yapılmış olur.

Sıcaklık kontrolünün hazırlanan oda modülünde gerçekleştirilmesi sırasında oda içerisine yerleştirilen üç adet sıcaklık sensörü ile odanın sıcaklık bilgisi çeşitli noktalardan ölçülmektedir. Bu üç noktadan alınan sıcaklık bilgilerinin ortalaması alınarak oda sıcaklık değerine ulaşılmaktadır. Aşağıdaki şekilde oda içerisinde sıcaklık sensörlerinin konumları gösterilmektedir.



Şekil 3.11 Sıcaklık sensörlerinin konumları

Sıcaklık sensörü olarak ADuC841'in üretici firması Analog Devices'ın piyasaya sürdüğü TMP35 sıcaklık sensörü tercih edilmiştir. Bu sıcaklık sensorünün çıkışı 10mV/°C dir. Şekil 3.12 deki grafikte TMP35'ın sıcaklık- Çıkış Gerilim grafiği görülmektedir.



Şekil 3.12 TMP sıcaklık- gerilim grafiği

3.3.1.1.1 Bulanık Mantık

Bulanık mantık algoritması insan diline en yakın olan algoritmadır. Bulanık mantık, klasik mantıkta bulunan kesin önermelerin yerine günlük hayatta kullanılan az, çok, daha fazla gibi önermeleri getirmektedir. Bulanık mantık, kontrol ünitelerinde ara değerler tanımlanmasına imkan sağlayarak, gerçek dünya koşullarına daha yakın kararlar verilmesini sağlar. Örneğin odadaki sıcaklık derecesindeki 1°C'lik değişme klasik mantık ile kontrol edilen bir ısıtıcının çalıştırılmasına yol açmaktadır. Bu odada bulanık mantık algoritması ile kontrol edilen bir ısıtıcı kullanılırsa 1°C'lik bir değişimde ısıtıcı tam güçte değil "birazcık" çalışmasını sağlar. Bu şekilde %90'a varan tasarruflar yapılabilir.

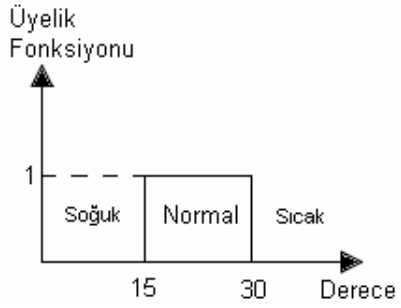
3.3.1.1.2 Bulanık Mantık Tarihçesi

İlk olarak Loutfi Zadeh bulanık mantık üzerinde çalışmıştır. Daha sonra 1974'te E.H. Mamdani Zadeh'in çalışmalarını genişletmiştir. 1980'lerde bulanık mantık kullanarak fiziksel uygulamaların başlamıştır. 1987'de "Sandai" hızlı metro kurulmuştur. 1990 'da ise IEEE 'de ilk konferans "Trans on fuzzy logic" yapılmıştır.

Bu çalışmada, bulanık mantık denetleme sistemlerinin temel ilkelerine iyi bir örnek sistem olması açısından klima sistem modellemesi incelenecektir. Zaman domeni incelemesinde küçük sıcaklık değişimlerinin sürekli olarak sistem çıkışını etkilediği sistemlerde bulanık mantık kontrol sistemlerinin kullanılması büyük önem arz etmektedir [4].

3.3.1.1.3 Bulanık Küme Teorisi

Klasik yani geleneksel mantıkta, herhangi bir nesne seçilen kümeye ya aittir ya da değildir. Yani klasik mantık 1'ler ve 0'lar üzerine kurulmuştur [4]. Şekil 3.13'de görüldüğü gibi 0-15^{0}C soğuk, 15-30^{0}C normal, 30-45^{0}C ise sıcak sınıfına girmektedir.

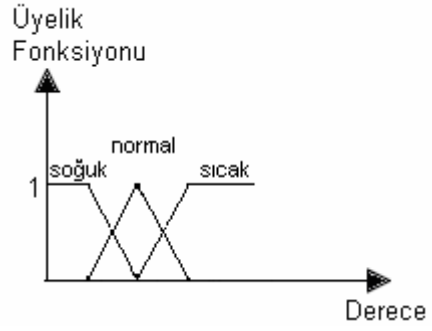


Şekil 3.13 Klasik sistemde üyelik fonksiyonları

Oysa insan 14^{0}C ile 16^{0}C arasına böyle bir ayrım yapmaz, daha az sıcak daha çok sıcak gibi dilsel ifadeler ile ortam sıcaklığını anlatır. İşte bulanık mantık bu dilsel ifadeler üzerine kurulmuştur. Her bir nesnenin üyelik fonksiyonu o nesnenin hangi kümeye daha çok yakın hangisine daha az yakın olduğunu gösterir.

Bulanık Mantık, makinelere insanların deneyimlerinin ve önsezilerinin sembolik ifadelerle aktarılmasıdır.

Sıcaklık için bulanık mantık kümeleri şekil 3.14'deki gibi gösterilebilir.



Şekil 3.14 Bulanık sistemde üyelik fonksiyonları

Yani bulanık mantıkta her elemanın bir üyelik fonksiyonu vardır. Bu değer 0 ile 1 arasında değişir.

$0 \leq \mu F(x) \leq 1$

Boolean veya keskin mantıkta ise karakteristik denklem sıfır veya birdir.

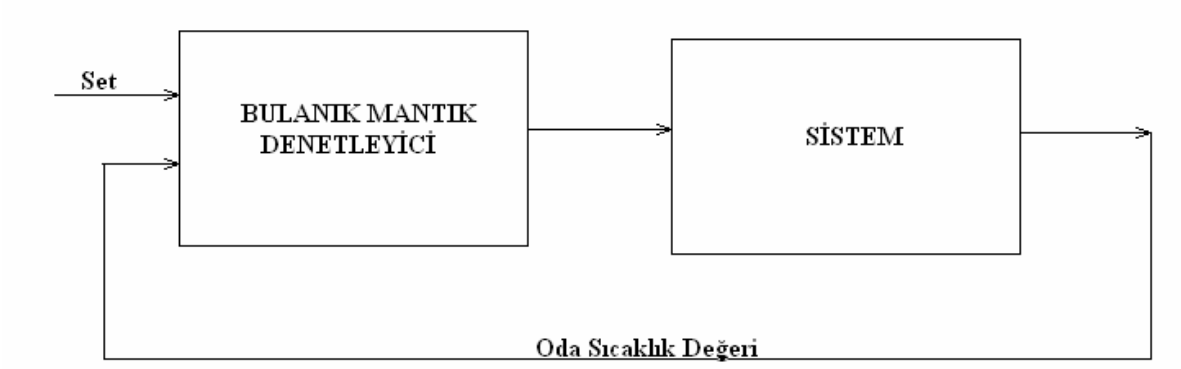
Yani: $\mu A(x) = 1$ x, A kümesinin elemanıdır.

$\mu A(x) = 0$ x, A kümesinin elemanı değildir.

Bulanık mantık kullanıcıya problem çözümü için yeni bir bakış açısı sağlar. Bulanık mantık tam olarak modellenemeyen sistemlerin belli parametrelerinin kontrolünde kullanılır. Bu yüzden tasarımcıya büyük kolaylık ve zamandan tasarruf sağlar.

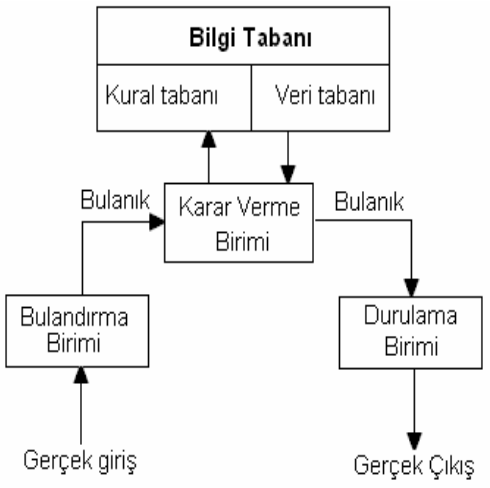
3.3.1.1.4 Bulanık Mantık İle Sıcaklık Kontrolü

Oda panolarında bulunan ADuC841 mikroişlemcisine C kodu ile bulanık mantık kodu yazılmıştır. Odadaki sıcaklık bilgisi ve ayarlanan sıcaklık değeri bulanık mantık algoritmasının girişleridir. Çıkış ise ısıtıcının veya soğutucunun ne kadar sürüleceğini belirten PWM (Pulse Width Modulation) bilgisidir.



Şekil 3.15 Bulanık mantık kapalı çevrim blok şeması

Bulanık mantık denetleyiciler, bilgi tabanı, bulandırma, karar verme ve durulama birimleri olmak üzere dört temel bileşenden oluşmuştur.



Şekil 3.16 Bulanık mantık blok diyagramı

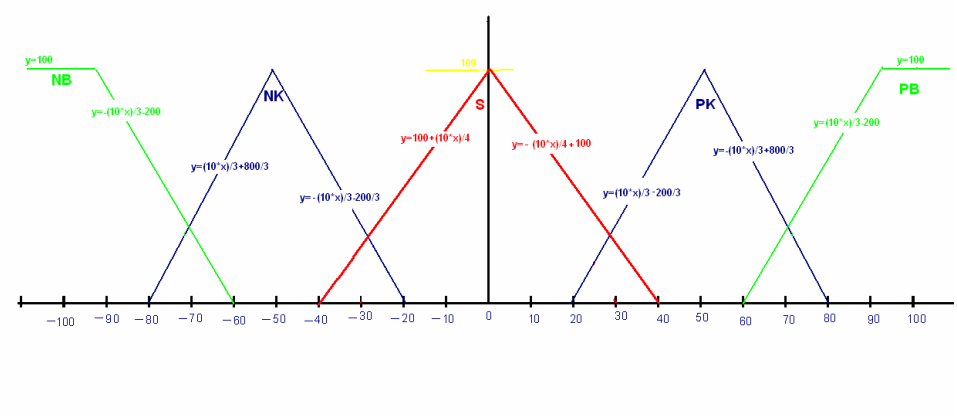
Sistem değişkenleri, denetlenen sistemden ölçülen E giriş değişkeni ve sistemi denetim için bulanık mantık denetleyici tarafından kullanılan U çıkış değişkeni olmak üzere iki çeşittir. Bulandırma birimi en son ölçülen verinin uygun dilsel değerlere dönüştürülmesini sağlar. Bulanık bilgi tabanı bilginin iki ana tipini kapsar: veri tabanı, her bir sistem değişkeninin değerleri gibi kullanılan bulanık kümelerin üyelik işlevlerini tanımlar, kural tabanı ise giriş bulanık değerlerin, çıkış bulanık değerlerine tam olarak eşlenmesini temsil eder. Karar verme birimi bulanık mantık denetleyicinin özüdür ve arzu edilen denetim stratejisine erişmek için, yaklaşık çıkarım sağlaması ile insan gibi karar verme yeteneğine sahiptir. Durulama birim ise karar verme biriminden gelen bulanık bilgileri, gerçek değerlere dönüştürerek, sistemin tanıyabileceği denetim hareketi haline gelmesini sağlar [1].

3.3.1.1.4.1 Bulandırma Birimi

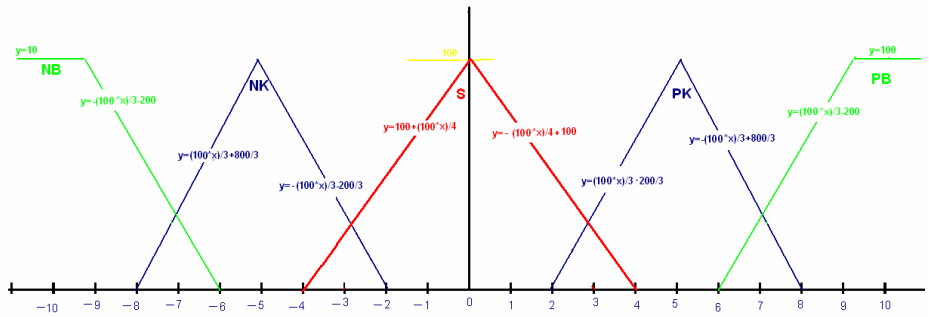
Bulanık mantık denetleyicide temel prensip, gerçek girişi yani kesin olan değerin bulandırılarak işleme alınması ve sonuçta bu bulanık çıkarımın durulanarak kesin yani gerçek değere dönüştürülmesine dayanır. Gerçek girişin bulanık değere dönüştürülmesi bulandırma biriminde yapılmaktadır. Bulandırma biriminde keskin değerin bulanık değere dönüştürülmesinde üyelik fonksiyonları kullanılır. Üyelik fonksiyonları çeşitli geometrik şekillerde olabilir. Bunlardan bazıları üçgen, yamuk, çan eğrisi, gaussiandır. Uygulaması gerçekleştirilen sıcaklık kontrolü için üçgen üyelik fonksiyonları kullanılmıştır. Örneğin giriş değişkenlerinden biri olan set değerinin keskin değeri, tanımlanan üçgen üyelik fonksiyonlarından hangi küme/kümelere ait olduğuna göre üyelik derecesi/dereceleri tespit edilir. Böylece girilen keskin değer üçgen üyelik fonksiyonları yardımıyla dilsel değişkenler

atanmış olur.

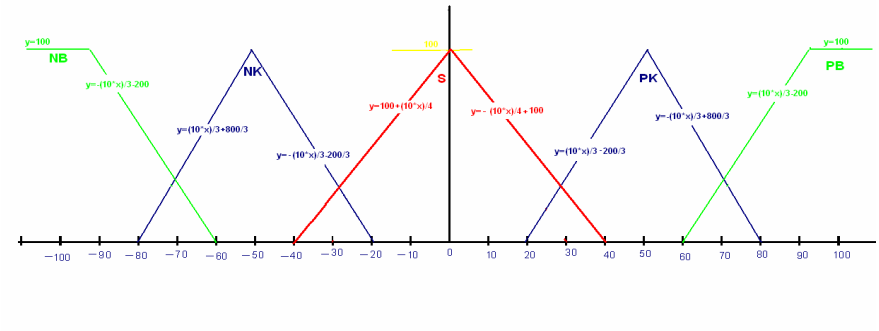
Hazırlanan sistemde giriş değişkenleri oda sıcaklığı ve kullanıcı tarafından girilen set değeridir. Çıkış ise sistemde kullanılan ısıtıcı veya soğutucunun hangi oranda sürüleceğini belirten PWM değeridir. Giriş ve çıkış değişkenleri için üçgen üyelik fonksiyonları aşağıdaki şekillerde gösterildiği gibi seçilmiştir.



Şekil 3.17 Hata üyelik fonksiyonu



Şekil 3.18 Hatanın değişimi üyelik fonksiyonu



Şekil 3.19 Çıkış üyelik fonksiyonu

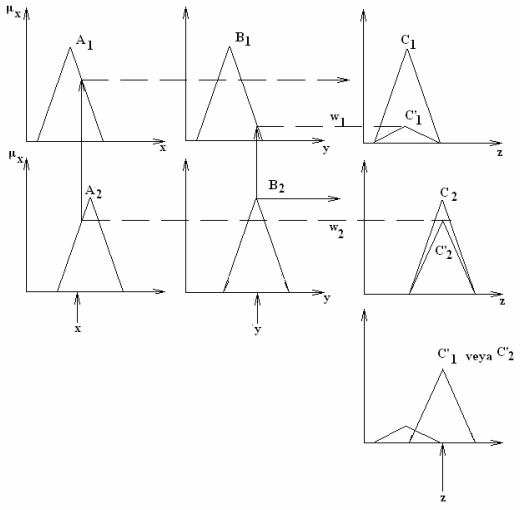
Her üç üyelik fonksiyonunda da beş üçgen üyelik fonksiyonu kullanılmıştır. Bu beş üçgen üyelik fonksiyonu soldan başlamak üzere negatif büyük, negatif küçük, sıfır, pozitif küçük, pozitif büyük olarak adlandırılmıştır.

3.3.1.1.4.2 Karar Verme Birimi

Üyelik fonksiyonlarına giren her bir giriş değişkeni bazen bir bazen birden fazla kümeye ait olabilir. Bu aitlik derecesine μ (mü) denir. Bulandırma işlemi ile sayısal değerlerden, sembolik değerler çıkarılır. Bulanık çıkarım ise denetimi yapılan sistemi kullanan uzman operatörün kullandığı dilsel niteleyiciler ve kurallar kullanılarak sembolik sonuçlar elde edilir. Burada veri tabanı ve karar verme mantığı kullanılmaktadır. Veri tabanı, bulanık kümelerin giriş çıkış değişkenleri ile kural tabanı ise bulanık kural cümlelerini içerir. Bulanık çıkarım için birden fazla yöntem vardır.

3.3.1.1.4.2.1 Max-Dot Çıkarım Yöntemi

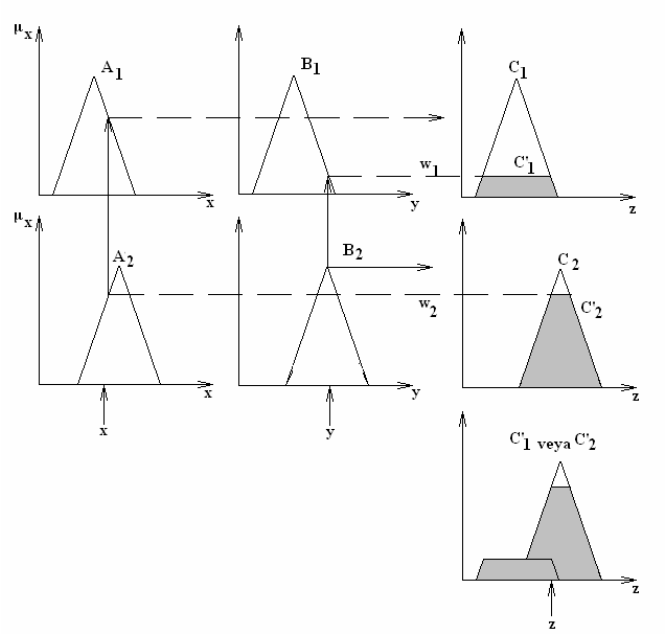
Her bir giriş değeri ait olduğu üyelik işlevindeki üyelik derecesine bağlı olarak, ilgili bulanık kümeyi yeniden ölçeklendirir. Çıkış değeri tüm girişiler için yeniden ölçeklendirilmiş bulanık kümeler içerisindeki maksimum değer alınarak bulunur [1].



Şekil 3.20 Max-dot çıkarım [1]

3.3.1.1.4.2.2 Min-Max Bulanık Çıkarım Yöntemi

Min-Max çıkarım yönteminde çıkış değeri elde edilen bulanık kümelere ağırlık ortalama yönteminin uygulanması ile bulunur [1].



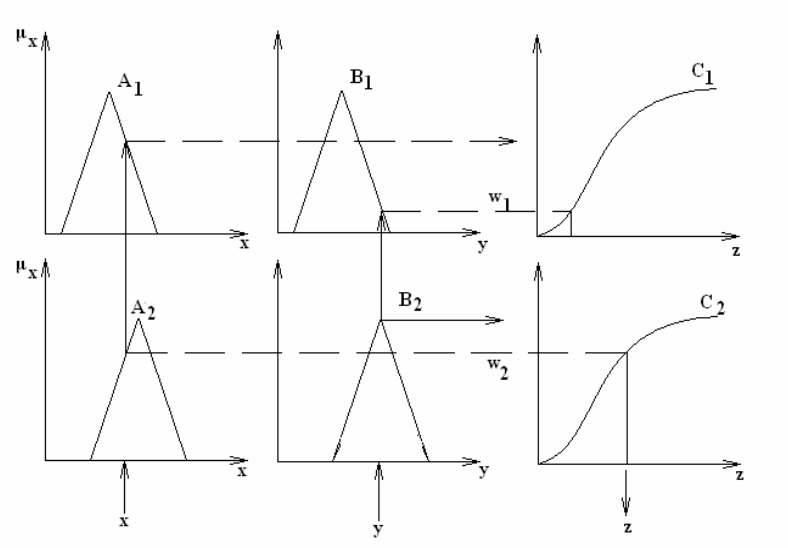
Şekil 3.21 Min-Max çıkarım [1]

$$z = \frac{\sum r_i z_i}{\sum z_i} \tag{3.1}$$

3.3.1.1.4.2.3 Tsukamato Bulanık Çıkarım Yöntemi:

Çıkış üyelik işlevi tek yönlü artan bir işlev olarak seçilir. Çıkış değeri ise her bir kuralın keskin çıkış değerinin ağırlık ortalaması alınarak bulunur [1].

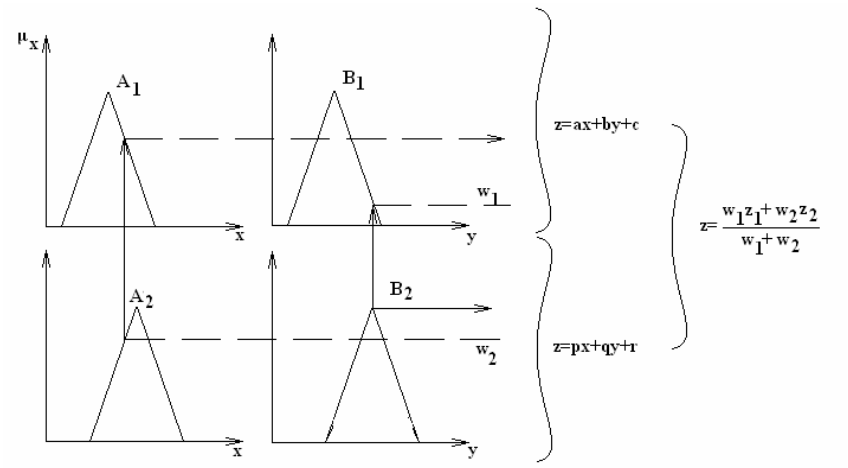
$$z = \frac{\omega_1 z_1 + \omega_2 z_2}{\omega_1 + \omega_2} \tag{3.2}$$



Şekil 3.22 Tsukamato çıkarım [1]

3.3.1.1.4.2.4 Takagi-Sugeno Bulanık Çıkarım Yöntemi

Her bir kuralın çıkışı giriş değerlerinin doğrusal birleşimi ile bulunur. Keskin çıkış değeri ise ağırlık ortalaması ile bulunur [1].



Şekil 3.23 Takagi-Sugeno çıkarım [1]

$$z_1 = f_1(x, y) = ax + by + c \tag{3.3}$$

$$z_2 = f_2(x, y) = ax + by + c \tag{3.4}$$

$$z = \frac{\omega_1 z_1 + \omega_2 z_2}{\omega_1 + \omega_2} \tag{3.5}$$

3.3.1.1.4.3 Kural Tablosu

Uzman kişi giriş değişkenlerinin değişimine göre çıkış değişkenin ne olacağını önceden sistemi inceleyerek tahmin edebilir. Bu tahminler sonucunda bir kural tablosu elde edilmiş olur. Hazırlanan sistemde giriş değişkenlerinin beşer üyelik fonksiyonuna karşılık gelen 5 x 5 lik bir matris şeklinde kural tablosu oluşturulmuştur. Bu kural tablosuna bakarak hata ve hatanın değişimine göre satır sütün işlemleri yapılarak çıkış ifadesi elde edilmiş olur.

Çizelge 3.3 Bulanık Mantık Kural Tablosu

		HATA				
		NB	NK	S	PK	PB
HATANIN DEĞİŞMİ	NB	S	PK	PK	PK	PB
	NK	NK	S	PK	PK	PB
	S	NK	NK	S	PK	PK
	PK	NK	NK	NK	S	PK
	PB	NB	NK	NK	NK	S

3.3.1.1.4.4 Durulama Birimi

Karar verme biriminin sonucu olarak elde edilen bulanık değerin sisteme işlenebilmesi için tekrar keskin değere dönüştürülmesi gerekmektedir. Bu dönüştürme işlemine “Cilalama” adı verilir. Durulama işleminde tercih edilebilecek yöntemlerden bazıları Maksimum üyelik yöntemi, ağırlık merkezi yöntemi, ağırlık ortalaması yöntemi, mean-max üyelik yöntemidir [6].

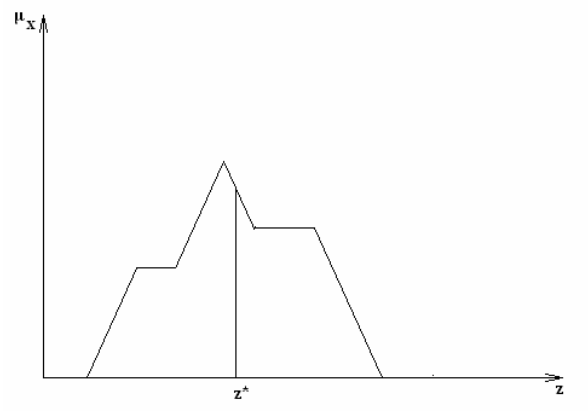
3.3.1.1.4.4.1 Ağırlık Merkezi Yöntemi

Ağırlık merkezi yöntemi en yaygın olarak kullanılan duruluma yöntemidir. Gerçek çıkış değeri, alanın merkezi ve uygulanan bulanık kümenin alanı ile aşağıdaki formüle göre

hesaplanır [5].

Şu formül ile ifade edilir [7];

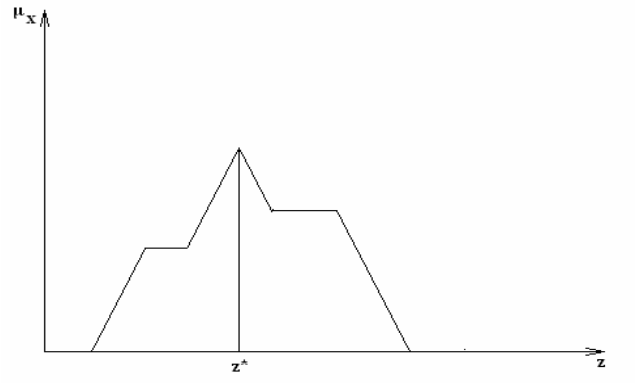
$$z^* = \frac{\int \mu_c(z) z dz}{\int \mu_C(z) sz} \tag{3.6}$$



Şekil 3.24 Ağırlık merkezi yöntemi

3.3.1.1.4.4.2 Maksimum üyelik merkezi yöntemi

Yükseklik yöntemi olarak da adlandırılmaktadır. Bütün üyelik dereceleri içinde en büyük olana eşittir.

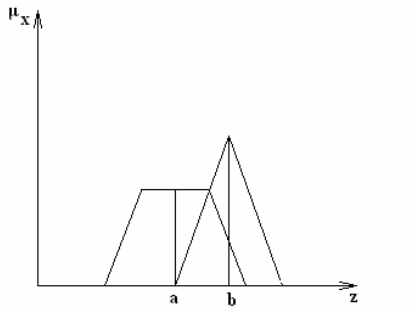


Şekil 3.25 Maksimum üyelik merkezi

$$\mu_C(z^*) \geq \mu_C(z), z \in Z \tag{3.7}$$

3.3.1.1.4.4.3 Ağırlık Ortalaması Yöntemi

Girişlerden elde edilen bütün bulanık değerler ile üyelik değerleri kullanılarak durulama yapılmaktadır [5].



Şekil 3.26 Ağırlık ortalaması yöntemi

Şu formül ile ifade edilir;

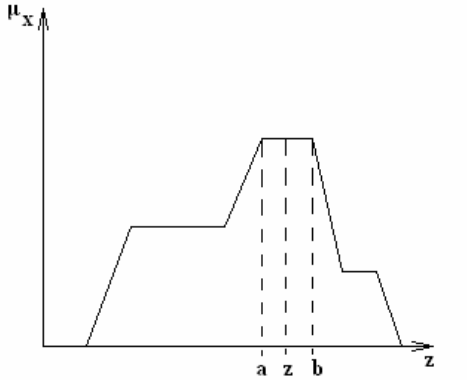
$$z^* = \frac{\sum \mu_C(\overline{z}).\overline{z}}{\sum \mu_C(\overline{z})} \tag{3.8}$$

3.3.1.1.4.4.4 Mean-Max Yöntemi

Maksimum üyelik derecesi tek bir nokta olmayan, düz olan sistemlerde kullanılır.

Şu formül ile ifade edilir;

$$z^* = \frac{a+b}{2} \tag{3.9}$$



Şekil 3.27 Mean-Max yöntemi

Uygulaması gerçekleştirilen sistemde durulama yöntemi olarak ağırlık ortalaması yöntemi seçilmiştir. Bu şekilde elde edilen çıkış gerçek değeri uygulamada kullanılan sürme devresine PWM aktif olma süresi olarak aktarılmıştır.

Mikrodenetleyici üzerine yazılan bulanık mantık kodu devamlı çalışmamaktadır. İnsan vücudu bulunduğu ortam içerisindeki sıcaklıktaki küçük değişimleri hissetmez. Ayrıca ortam sıcaklığı ısıtıcı devreye girdiği anda birden değişememektedir. Bunlar göz önünde bulundurularak tasarlanan sistemde bulanık mantık algoritması sürekli olarak çalıştırılmamakta, belirli süre aralıkları ile sistemdeki değişikliklere göre çıkışlarda uygun değişikliği yapmaktadır.

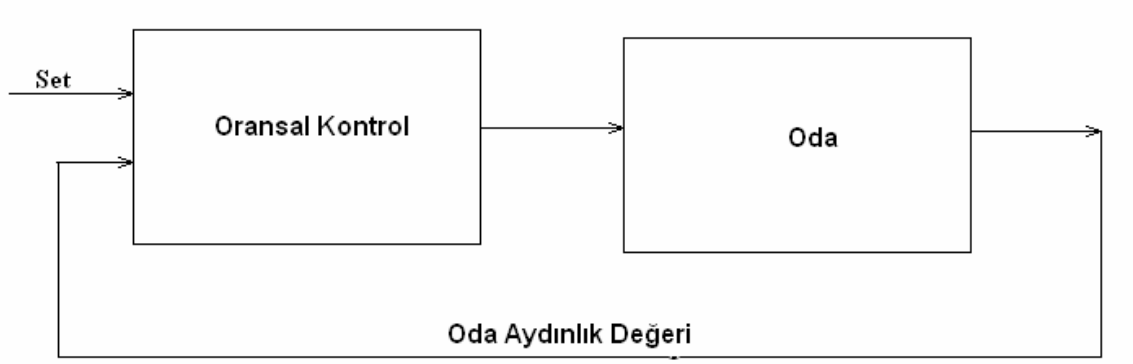
3.3.1.2 Aydınlatma Kontrolü

Ev otomasyonunda kontrol edilen parametrelerden biri de aydınlık seviyesidir. Seçilen algoritmaya göre tüm evin aydınlık kontrolü aynı olabileceği gibi, her odanın aydınlık seviyesi farklı da olabilir. Gerçekleştirilen ev otomasyonunda her odanın aydınlık seviyesi bağımsız düşünülerek tasarlanmıştır. Odalarda bulunan mikroişlemci ile odanın aydınlık seviyesi kontrol edilir.

Evlerde aylık elektrik faturalarının yaklaşık %20'si aydınlatma amaçlı kullanıma aittir. Verimli aydınlatma hem faturalarda hem de gözlerde rahatlama sağlayacağından daha düşük faturalar ve daha kaliteli aydınlatma ile memnun edici sonuçlar elde edilecektir [11].

Bir odadaki aydınlık seviyesinin doğru ayarlanması insan sağlığı açısından çok önemlidir. Doğru bir aydınlatma ile göz sağlığı korunabilir. Bir ortamı ve içerisinde nesneleri, istenilen ölçütlerde görsel algılamaya uygun kılacak şekilde tasarlanmış ışık uygulamaları ”aydınlatma” olarak tanımlanır [8].

Gerçekleştirilen birçok uygulamada aydınlatma kontrolünde on-off kontrol tercih edilir. Ancak on-off kontrolde enerji tasarrufu yapılamaz. Kapalı çevrim kontrol uygulanarak odanın değişen ışık seviyelerine göre ışık kaynakları sürülerek enerji tasarrufu sağlanmaktadır. Bu kapalı çevrim için odanın içerisindeki ışık bilgisine ihtiyaç duyulur. Bu ışık bilgisi ışığa duyarlı sensörler ile ölçülerek mikroişlemciye giriş olarak verilmektedir.



Şekil 3.28 Aydınlatma için kapalı çevrim

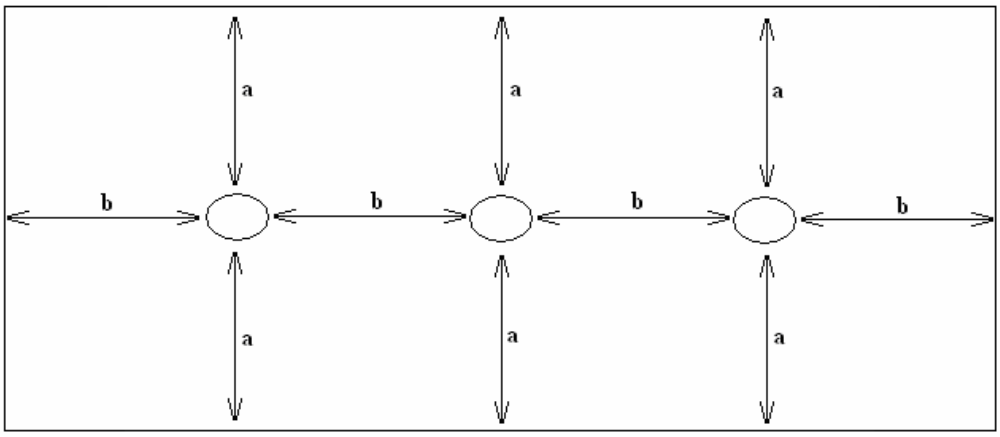
Uygulaması gerçekleştirilen sistemde kapalı çevrim kontrol esas alınmıştır. Ancak kapalı çevrim kontrolde geri besleme bilgisi bir adet sensörden alınıp buna göre bir adet aydınlatma kaynağı sürülürse odanın her noktasında eşit aydınlık seviyesi sağlanamaz. Işık kaynağına yakın olan yerler daha aydınlık, uzak olan yerler ise daha karanlık olur. Bunu önlemek için birden fazla sensör ve birden fazla ışık kaynağı kullanılmıştır. Tüm sensörlerden bilgi alınarak ortalama alınırsa ve bu değere göre ışık kaynakları sürülürse odada tüm noktalarda aynı aydınlık seviyesi yakalanır ancak pencereye yakın yerdeki ışık kaynağı ile pencereye uzakta bulunan ışık kaynağı aynı seviyede yanacağından ekonomik açıdan zarar edilmiş olunur.

Tüm bu durumlar göz önüne alınarak tasarlanan sistemde üç adet sensör ve üç adet ışık kaynağı kullanılmıştır. Bu ışık kaynakları ilgili sensörden alınan bilgiye göre kontrol edilmektedir. Bu kontrolde oransal kontrol tercih edilmiştir.

Aydınlatılan mekanın her noktasında aydınlık seviyesinin elde edilmesi oldukça zordur. Bir oda içerisindeki maksimum, ortalama ve minimum aydınlık seviyeleri arasındaki büyük farklar ışık kaynaklarının yanlış konumlandırılmasından ya da uygulama için uygun olmayan ışık dağılım özelliğine sahip aygıtların kullanılmasından kaynaklanır. Oda içerisindeki E-min/E-ortalama değeri aydınlık seviyesi dağılım faktörü olarak tanımlanır. Bu değer aydınlatmaya olan ihtiyacın düşük olduğu yerlerde 0.4 yüksek olduğu yerlerde ise 0.66 değerlerinden düşük olmamalıdır.

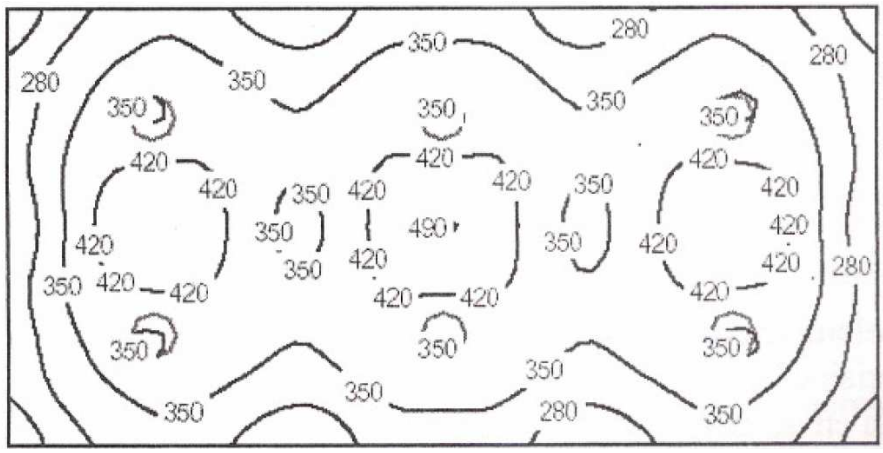
Işık kaynakları aydınlatacakları hacim içersinde belli kuralar doğrultusunda yerleştirilirler. Hacim içerisinde yanlış yerleştirilmiş ışık kaynakları kamaşma gölge oluşumu veya ışık dağılımının boşa harcanması gibi sorunlara yol açabilir.

Tasarlanan sistemde şekil 3.29 deki gibi ışık kaynağı yerleşimi yapılmıştır.



Şekil 3.29 Işık kaynaklarının yerleşimi

Bu durumda aydınlık seviyesi dağılımı şekil 3.30 teki örnek gibi olur.

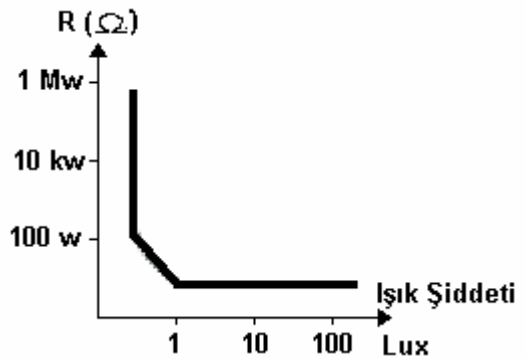


Şekil 3.30 Örnek aydınlık seviyesi dağılımı [8]

Odanın pencereye yakın yani ışık alan bir bölümde bulunan ışık kaynakları sensörden alınan bilgiye göre daha düşük seviyeli yanarken, pencereye uzak yani ışık alan bir bölüme uzak olan ışık kaynakları sensörden alınan bilgiye göre daha yüksek oranda yanar. Bu kademe ışık kaynaklarının PWM ile sürülmesiyle ayarlanır.

Bu şekilde kademeli olarak aydınlatılan bir odada hem her noktada aynı aydınlık şiddeti yakalanmış olur hem de enerji tasarrufu sağlanmış olur. Odanın ışık bilgisinin ölçülmesi için ışığa duyarlı bir direnç türü olan LDR (light dependent resistance, fotodirenç) tercih edilmiştir. Bu dirençler ışıkta az direnç, karanlıkta yüksek direnç gösterirler. Kısaca aydınlıkta LDR'lerin üzerinden geçen akım artmakta, karanlıkta ise azalmaktadır [13].

LDR'lerdeki direnç-ışık grafiği şekil 3.31'de gösterilmiştir.



Şekil 3.31 LDR grafiği

Işık kontrolünde de PWM kullanılmıştır. PWM in aktif olma süresi oransal kontrol algoritmasından elde edilir. Oransal kontrol algoritması LDR lerden aldığı ışık bilgisini ortalamasını alarak kontrol algoritmasına sokmaktadır. Kontrol algoritmasında ışık şiddeti ile kullanıcının ayarlamış olduğu değer arasındaki hataya göre işlem yaparak ışık grubunun sürülmesinde kullanılan PWM aktif olma süresi belirlenir. Bu PWM sürelerine göre ışık grupları ayrı ayrı sürülür.

Oransal kontrol türünde nihai kontrol elemanı kontrol edilen değişkenin değişim miktarına bağlı olarak konumlanır. Kontrol elemanın oransal bandı içinde kontrol edilen değişkenin her değerine karşılık nihai kontrol elemanının bir tek konumu vardır. Başka bir deyişle kontrol edilen değişken ile nihai kontrol elemanı arasında doğrusal bir bağlantı vardır [3].

3.3.1.2.1 Güvenlik Kontrolü

İnsan refah seviyesini artırmak için içinde bulunduğu ortamların güvenliğini sağlamak ister. Ayrıca bu duygu sadece içinde bulunduğu ortamlar içinde olmayıp, değer verdiği, emek vererek sahip olduğu ortamların korunmasını da içerir. İnsanlar için ev otomasyonunda güvenlik kontrolü olmazsa olmazlardandır. Evde yokken evinin güvende olduğunu bilmek, herhangi bir tehlike karşısında kendisine hemen haber geleceğini bilmek insana huzur verir.

Bu koşullar göz önüne alınarak ev otomasyonlarında çeşitli güvenlik kontrol algoritmaları hazırlanmıştır. Eve izinsiz giriş olduğunda yazılan algoritma ile bu izinsiz giriş hem ev sahibine hem de ilgili güvenlik birimlerine bildirilebilir. Başka bir yöntem ise, eve yaklaşan bir tehlikeye karşı, evin boş olduğunun gizlenmesidir. Bu gizleme belli bir saatten sonra ışıkların otomatik olarak yanması, müzik sisteminin açılması şeklinde olabilir. Eve izinsiz giriş olduğunda ise evin ışıkları tehlikeyi gösterecek şekilde yanıp sönebilir ve alarm

çalışabilir.

Gerçekleştirilen ev otomasyon sisteminde ev boş iken eve izinsiz olması pencerelere konan manyetik röleler yardımıyla algılanabilir. Pencere izinsiz açıldığında manyetik role çıkışı aktif olarak oda mikroişlemcisine ilgili sinyali gönderir. Oda mikroişlemcisi ise pencerenin açıldığını RS485 haberleşme protokolü üzerinden bildirir. Ana pano ise evde insan olup olmamasına göre alarm çıkışını aktif eder.

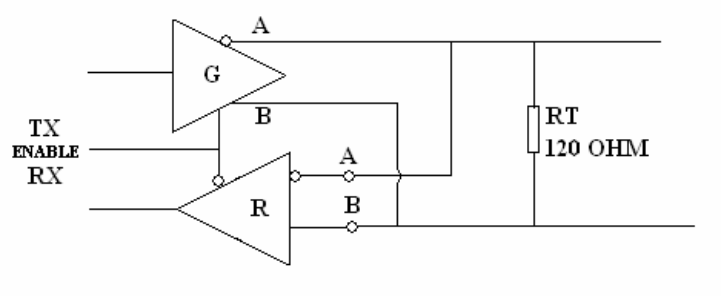
Sistemde pencerelerde kullanılan manyetik röle olarak rit röle seçilmiştir. Rit röle pencere pervazına yerleştirilmiş karşısına gelecek şekilde bir mıknatıs yerleştirilmiştir. Bu sayede rit röle mıknatısın manyetik alanından uzaklaştığında konumunu değiştirerek, bilgi vermektedir. Bu röle mikroişlemcinin dış kesmesine bağlıdır. Bu sayede rit röleden gelen bilgi en önemli öncelik sırasında mikroişlemcide işlenmiş olmaktadır.

3.3.1.2.2 Haberleşme

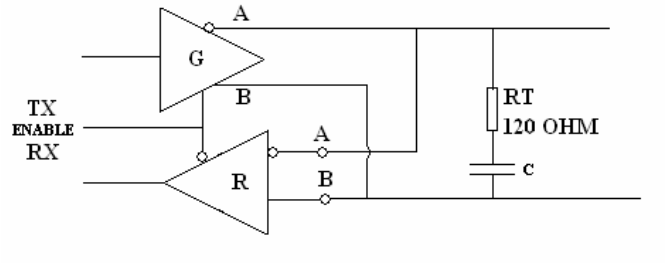
Tasarlanan ev otomasyon sisteminde ana pano ve oda modülleri bulunmaktadır. Oda modülleri yukarıda anlatılan tüm işlemleri kendi üzerlerinde yapabilmektedirler, ancak ana panonun yapılan işlemlerden haberi olamamaktadır. Bu oda modüllerinden bilgi almak için point-to-point haberleşme protokolü kullanılamamaktadır. Çünkü bir ana pano ve birden fazla oda modülü bulunmaktadır. Bu nedenle point-to multipoint haberleşme protokolü olan RS485 haberleşmesi kullanılmıştır. Bu haberleşme yapısı çift hat üzerine full-dupleks olarak kurulmuştur. Hatlardan biri master birimi, transmit hattı diğeri ise receive hattıdır. Bu iki hat sayesinde tüm slaveler ve master full-dupleks olarak, master üzerinden haberleşmektedirler.

RS485 hattının verici ve alıcısı olabilir. Bağlantı uzunluğu ise 1000 metreyi aşabilmektedir. Haberleşme hızı saniyede 10Megabittir. Haberleşme hızı kablo uzunluğuna bağlıdır. Kablo uzunluğu arttıkça haberleşme hızı düşer [16].

RS485 haberleşme yapısının çalışma mantığı, hattı dinlemekte olan kısmın RS485 entegresinin üzerinde 200mV gerilim farkı oluşturmasıdır. Bu nedenle RS485 haberleşmesi diğer haberleşme protokollerine göre daha fazla akım çekmektedir. Bu akım değeri kullanılan haberleşme hat uzunluğuna göre değişmektedir. Genellikle kullanılan sonlandırma direnci değeri 120 Ω dur. 90Ω ‘un altında bir sonlandırma direnci kullanılmalıdır ve sonlandırma dirençleri mutlaka hattın sonu koyulmalıdır. Hattı AC gürültülerden daha da arındırmak için sonlandırma direncine paralel olarak bir 100nF kapasite de takılarak AC sonlandırma yapılabilir [9].



Şekil 3.32 RS485 sonlandırma direnci



Şekil 3.33 RS485 AC sonlandırma direnci ve kapasitesi

RS485 hattında birçok dinleyici modül olabilir, tüm modüller dinleme durumunda iken hattı kontol eden olmadığı için hattı idle (uyku) modunda tutabilmek için bias dirençleri kullanılır. Örnek olarak 10 nodlu bir haberleşme sisteminde 120Ω sonlandırma dirençleri kullanılırsa ve her RS485 entegresinin iç direncini yaklaşık olarak 12KΩ olarak alırsak, hattın eş değer direnci 10 adet 12KΩ paralelliğinden 1.2KΩ ve iki adet 120Ω sonlandırma direncinden 60Ω'un birbirine paralel olması ile hattı eş değer direnci Denklem 3.10 dan 57.15 Ω olarak bulunur. Çok düşük bias akımlarında sonlandırma direncine gerek kalmaz [15].

$$\frac{1}{1200}+\frac{1}{60}=\frac{1}{R} \tag{3.10}$$

RS485 entegresinin A ve B uçları arasında 200mV oluşturmak için gerekli olan akım Denklem 3.11 den 3.5 mA olarak bulunur.

$$\frac{V}{I}=\frac{200mV}{57\Omega}=3.5mA \tag{3.11}$$

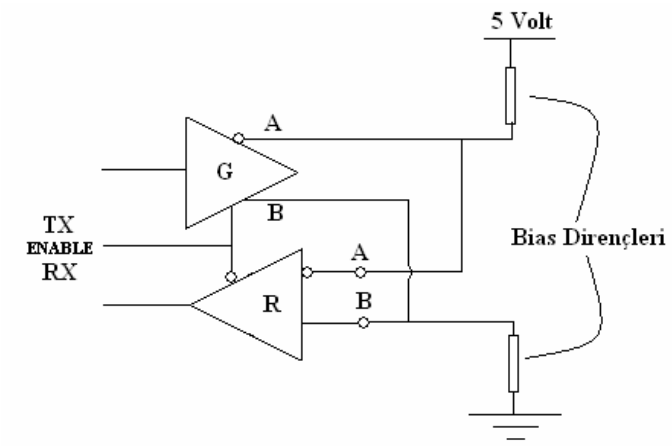
Sistemimiz 5 Volt ile çalıştığı için, bu gerilim seviyesinden bu akımı sınırlamamız için gerekli

olan direnci Denklem 3.12 den 1428 Ω olarak buluruz.

$$\frac{V}{I} = \frac{5V}{3.5mA} = 1428\Omega \tag{3.12}$$

1428Ω' a haberleşme hattının eş değer direnci 57Ω de ekli olduğu için bias dirençlerinin bulmak için Denklem 3.14 deki işlem yapılarak bias akımları bulunur. Şekil 3.34 deki gibi bağlantı yapılır.

$$\frac{(1428 - 57)}{2} = 685\Omega \tag{3.13}$$



Şekil 3.34 RS485 Bias dirençleri

3.3.1.2.2.1 Tasarlanan Sistemdeki RS485 Habeşlemesi

Tasarlanan sistemde üç oda modülü ve bir ana pano bulunmaktadır. Bu üniteler arasında kullanılan RS485 hattının bias direnci hesabı aşağıda bulunmaktadır. Denklem 3.14 te haberleşme hattının eş değer direnci 58.8Ω olarak bulunur. Denklem 3.16 ile akıtılması gerek akım değeri bulunur ve Denklem 3.17de bu akım degeri için gerekli olan toplam direnç bulunur. Denklem 3.18 ile tasarlanan haberleşme hattında kullanılması gereken bias direnç değeri bulunmuş olur.

$$\frac{1}{3000} + \frac{1}{60} = \frac{1}{R} \tag{3.14}$$

$$R = 58.8\Omega \tag{3.15}$$

$$\frac{V}{I} = \frac{200mV}{58\Omega} = 3.44mA \tag{3.16}$$

$$\frac{V}{I} = \frac{5V}{3.44mA} = 1453\Omega \tag{3.17}$$

$$\frac{(1453 - 58)}{2} = 697.5\Omega \tag{3.18}$$

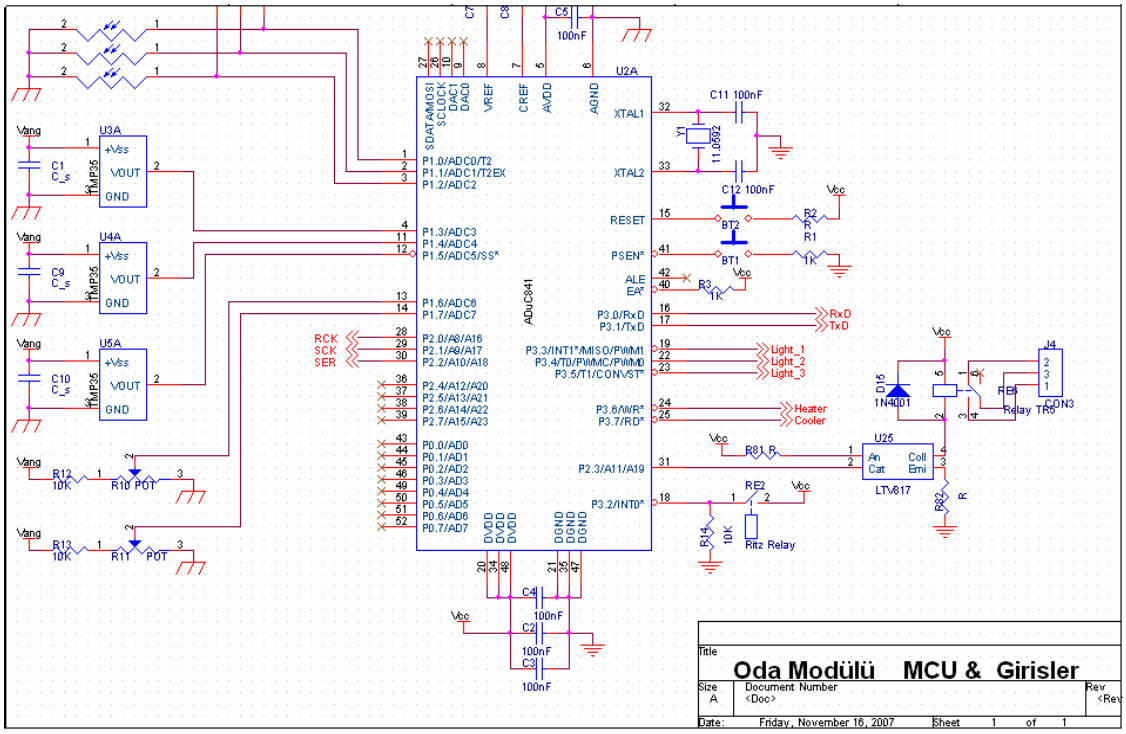
Donanımsal olarak yukarıda belirlenen direnç ile tamamlanan RS485 haberleşme protokolü sistemde Ana pano kontrolü altında çalışmaktadır. Ana pano belirli sürelerde oda modüllerini tarayarak son yapılan işlemler ve durumlar hakkında bilgini tazelemektedir. Bu sistemde oda modülleri slave olarak tanımladığı için kendi aralarında haberleşememektedirler, donanımsal olarak da buna izin verilmemiştir.

3.3.1.3 Oda Panosunun Yazılımı

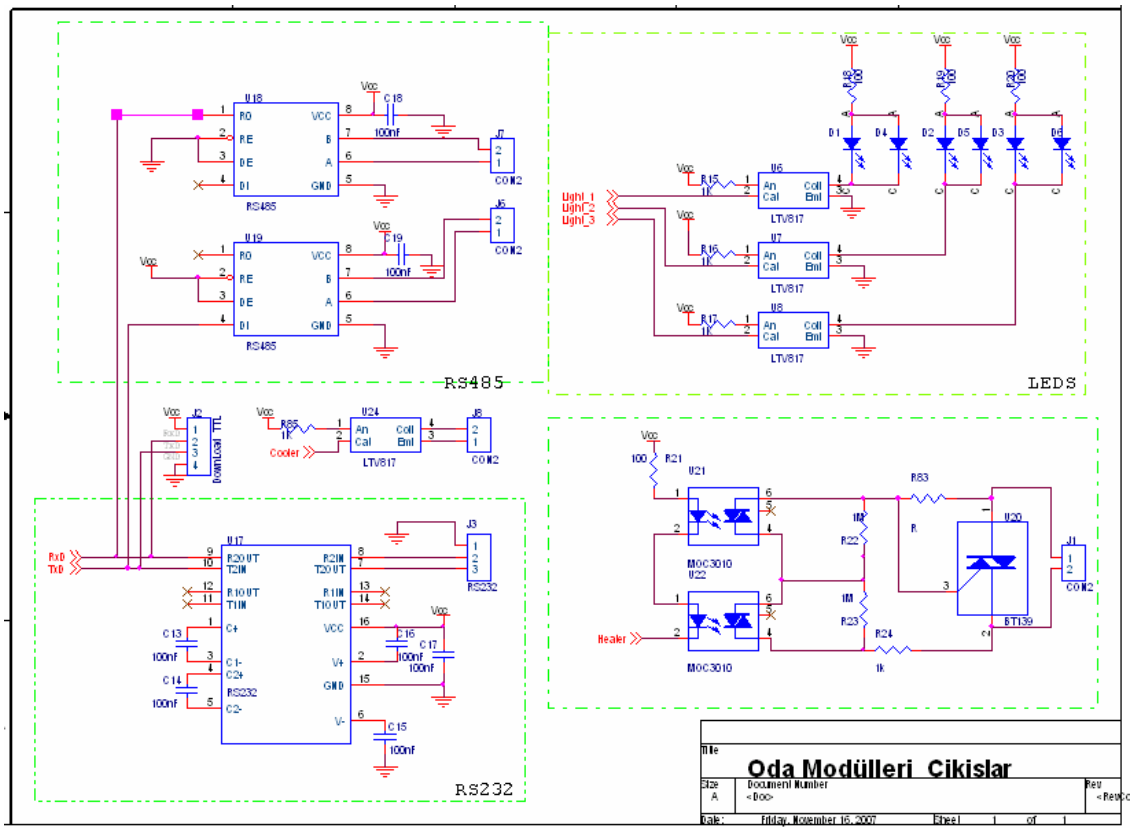
Oda panosunun yazılımı C dili kullanılarak oluşturulmuştur.

3.3.1.4 Oda Panosunun Donanımı

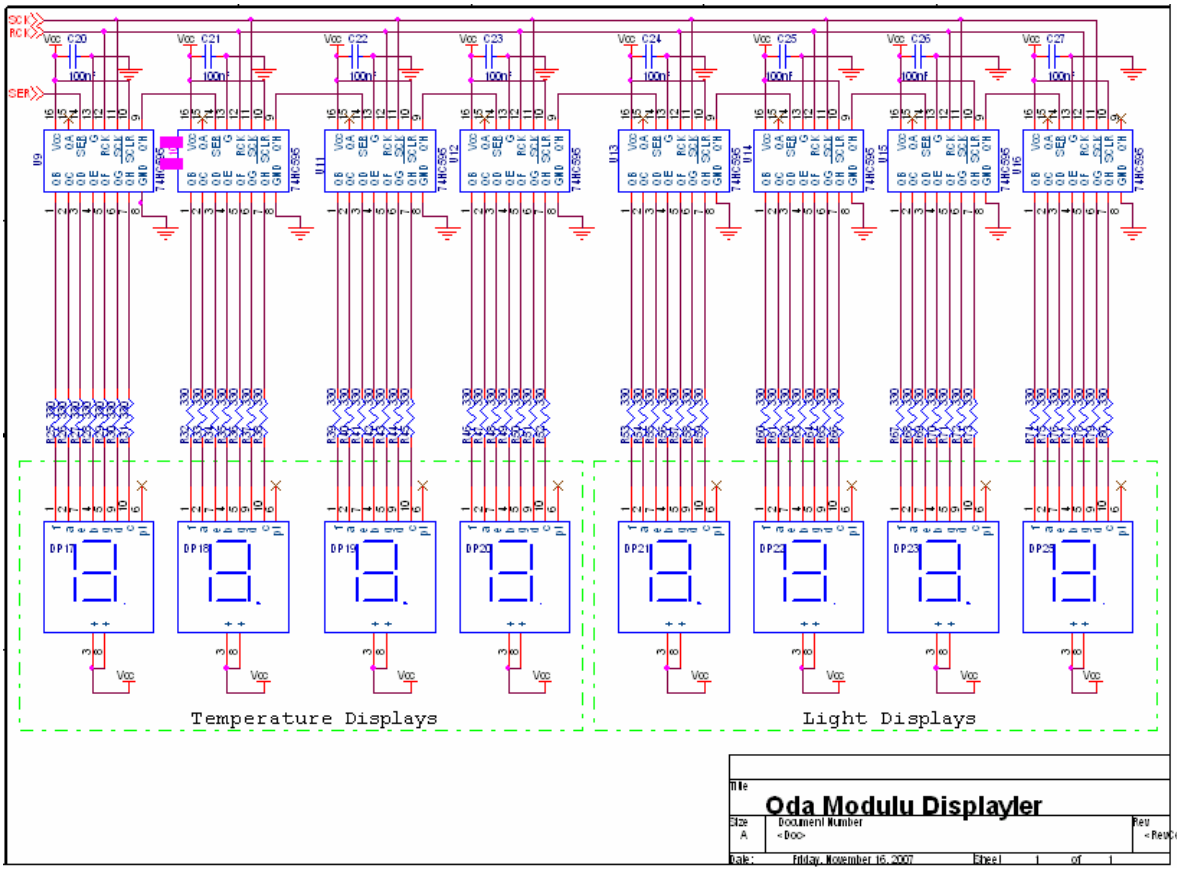
Oda panosunun donanımında baskı devre kart tasarımı yapılmıştır. Bu tasarım için OrCAD programının Capture ve Layout kısımları kullanılmıştır. Önce delikli pertinaks üzerinde donanımlar denenmiş ve hazırlanan devreler OrCAD programı yardımıyla PCB üzerine aktarılmıştır. Şekil 3.35–38 de OrCAD programının CAPTURE'ında çizilmiş olan PCB şematiği ve Şekil 3.39 da OrCAD LAYOUT da çizilmiş PCB gösterilmektedir.



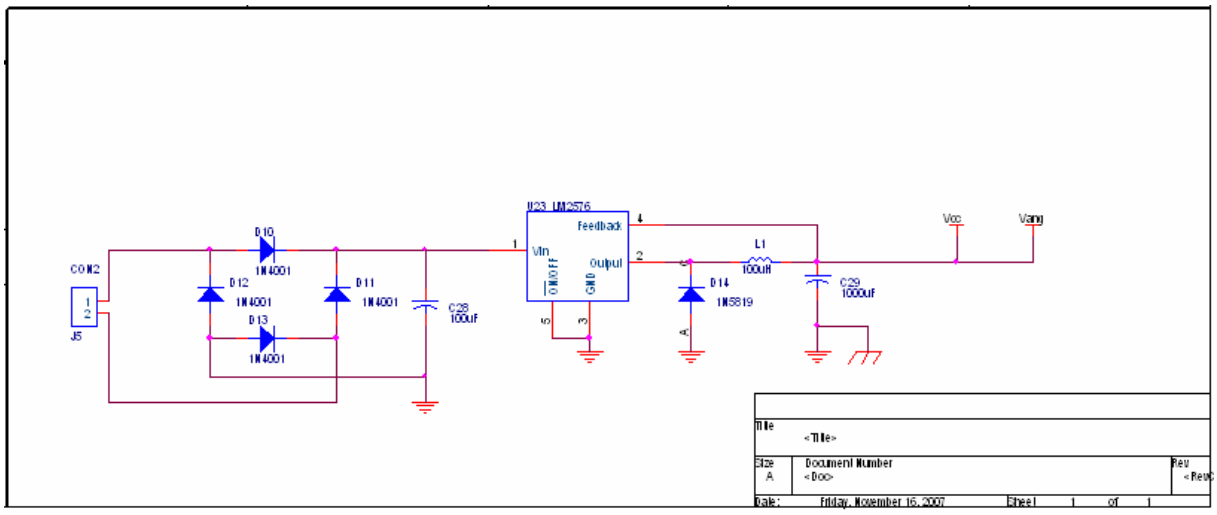
Şekil 3.35 PCB şematiği (MCU ve Girişler)



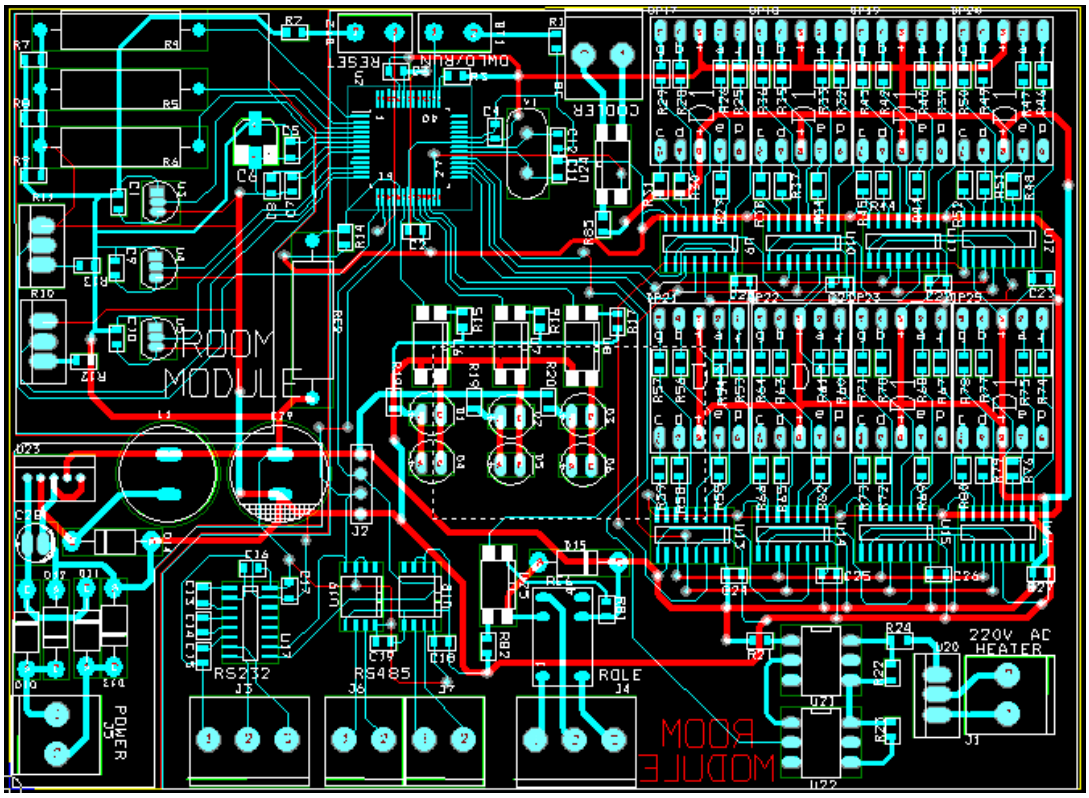
Şekil 3.36 PCB şematiği (Çıkışlar)



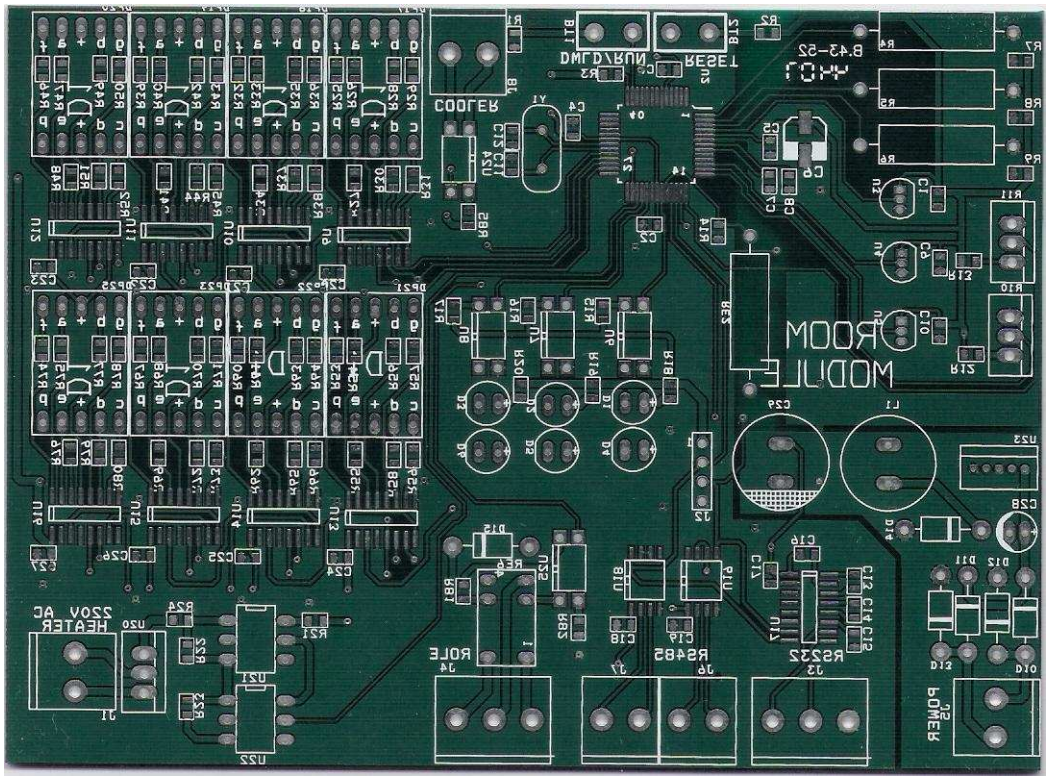
Şekil 3.37 PCB şematiği (Göstergeler)



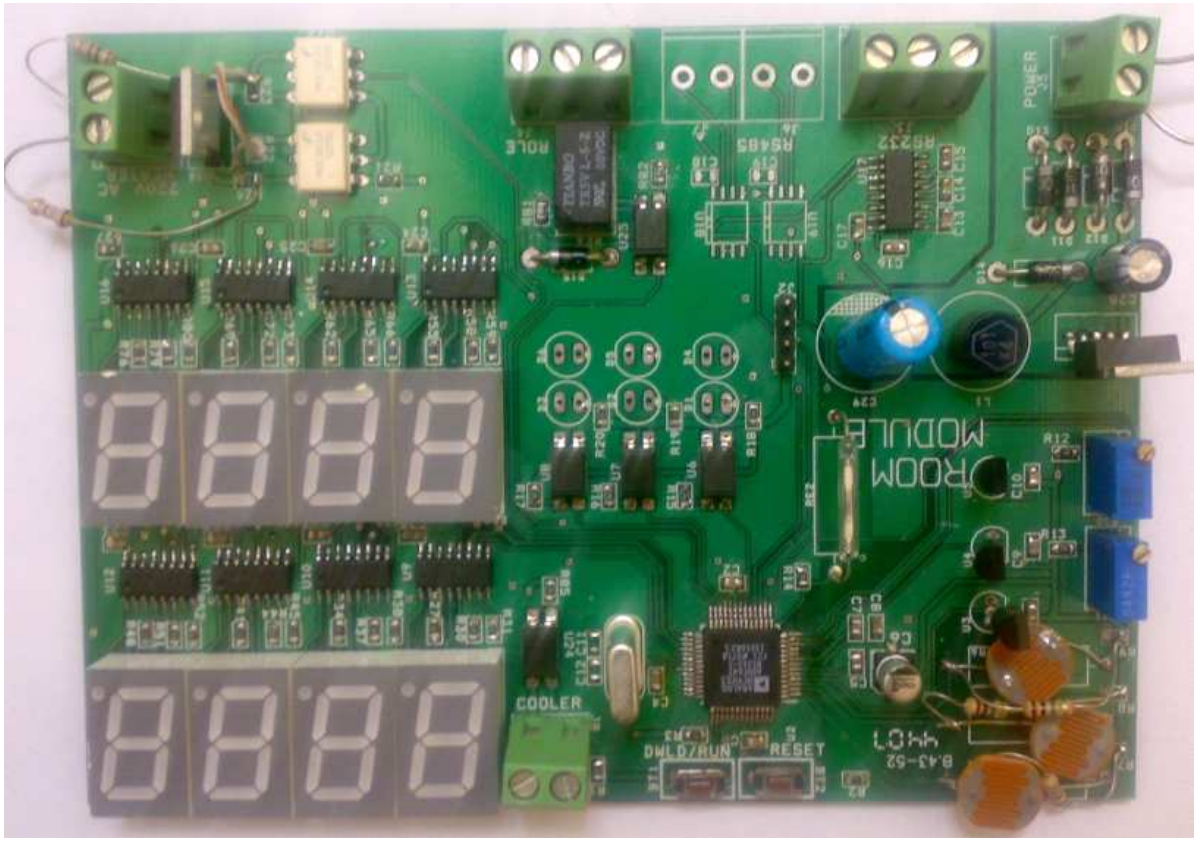
Şekil 3.38 PCB şematiği (Güç)



Şekil 3.39 PCB layout



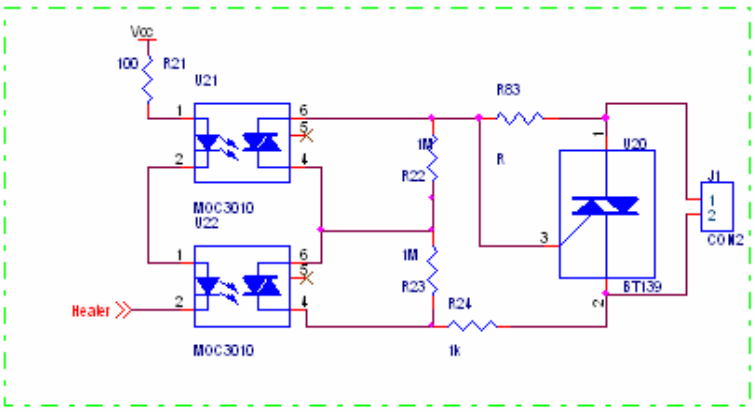
Şekil 3.40 PCB resmi



Şekil 3.41 PCB resmi (dizili)

3.3.1.4.1 Sıcaklık

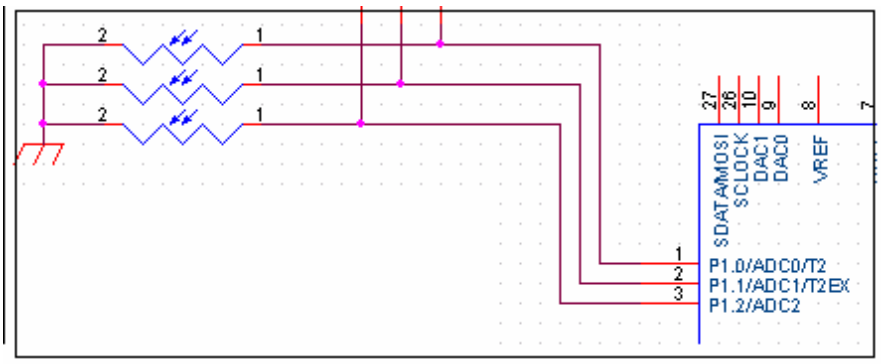
Sıcaklık kontrol devresinde odanın içerisindeki sıcaklık bilgisi TMP35 sıcaklık sensörü yardımı ile okunur. Çıkışta ise ısıtıcı veya soğutucu sürülür. Isıtıcı olarak 220V ile çalışan bir ısıtıcı tercih edilmiştir. 220V, 50W'lık bir ısıtıcı kontrol edebilmek için mikroişlemcinin 5V'luk port pini işlem göremeyeceği için arada bir sürücü devresi kullanılmıştır. 220V kısmını açıp kapatmak için BT139 Triac'ı kullanılmıştır. Mikroişlemcinin portundan bu triac'ı tetiklemek için iki adet MOC3010 kullanılmıştır. Bu MOC3010 optoküplörleri ile mikroişlemci 5V devresi ile 220V devresi birbirinden ayrılmış olur. Bu şekilde mikroişlemcinin 220V'tan gelen gürültülerden arındırılarak sağlıklı çalışması sağlanmıştır.



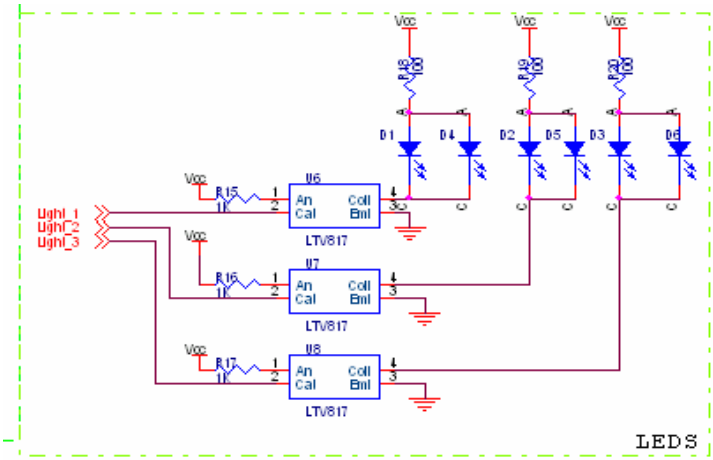
Şekil 3.42 220V ısıtıcı kontrol devresi

3.3.1.4.2 Aydınlatma

Aydınlatma kontrolünde odanın aydınlık bilgisi LDR'lerden mikroişlemcinin ADC birimi yardımı ile dışarıdan alınır. İlgili işlemler yapıldıktan sonra LED gruplar sürülür.



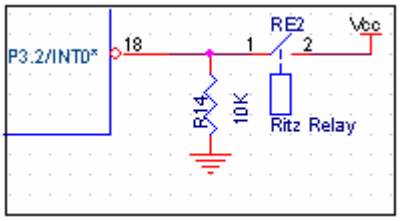
Şekil 3.43 Ldr den okuma



Şekil 3.44 Led grubu sürme şeması

3.3.1.4.3 Güvenlik

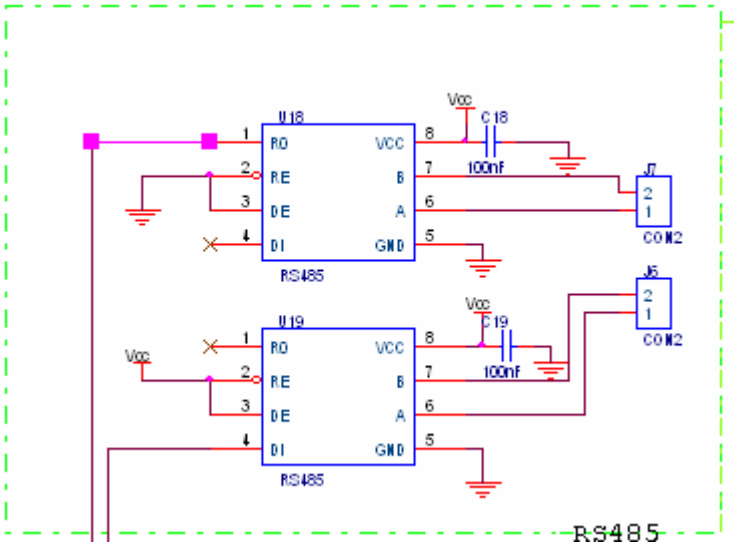
Güvenlik kontrolünde mikroişlemcinin dış kesmesi kullanılmıştır. Dış kesme ile rit röleden gelen sinyale anında tepki verilerek güvenlik işlemi gerçekleştirilmiş olunur.



Şekil 3.45 Rit röle bağlantısı

3.3.1.4.4 Haberleşme

Haberleşme donımında mikroişlemcinin RX ve TX pinlerine birer adet ADM485 entegresi konularak mikroişlemci TTL seviyesinden RS485 haberleşme seviyesine geçiş yapılmıştır.



Şekil 3.46 RS485 devre şeması

SONUÇLAR

Ekonomideki ve endüstrideki gelişmeler enerji talebini arttırmıştır. Teknolojinin sunduğu daha fazla konfor şartları, hem enerjinin daha ekonomik kullanımı hem de daha konforlu yaşam için insanlar her alanda otomasyona yönelmiştir. Her geçen gün teknolojinin evlerimizdeki yeri artmıştır. Ev ve bina otomasyon sistemlerin en önemli bölümleri ısıtıma ve havalandırmadır. Ev otomasyonu sayesinde bu sistemlerin optimum dizaynı yapılabilir. Bu şekilde insanların yaşama ve çalışma ortamlarında sürekli bir rahatlık, enerji tasarrufu sağlanarak elde edilmiş olunur. Bu şekilde yaşam kalitesi artmaktadır.

Bu tez çalışmasında akıllı ev otomasyonu kontrolü mikrodenetleyici kullanılarak gerçekleştirilmiştir. Odalar birbirlerinden ayrı ayrı düşünülerek her bir oda için ayrı bir kontrolör tasarlanmıştır. Her oda kontrolünün gerçekleştirdiği işlemlerin sonuçları ise ana pano üzerinde bulunan mikrodenetleyici sayesinde, gösterge aracılığı ile kullanıcıya sunulmaktadır.

Ev ve bina sistemlerinin tasarımında kapalı çevrim kontrol kullanıcıya her açıdan avantaj getirmektedir. Kullanılan sensörler yardımıyla odalardaki sıcaklık ve aydınlık bilgisi kapalı çevrim kontrol algoritmasının girişi olarak sisteme verilmektedir.

Sıcaklık kontrolünde ve aydınlık kontrolünde kapalı çevrim kontrol algoritmaları kullanılmıştır. Kapalı çevirim algoritmaları sayesinde sistemde harcanan enerji minimuma çekilmiştir. Bunun nedeni, sistemden sürekli veya belirli aralıklarla geri besleme alınması ve ilgili çıkışların bu geri beslemelere göre verilmesidir.

Sıcaklık kontrolünde sistemde bulanık mantık kapalı çevirim kontrol algortiması kullanılmıştır. Bu algoritmanın tercih nedeni, sıcaklık kontrolü yapılacak olan odanın belirli bir transfer fonksiyonun olmayışı ve tasarlanan sistemde ısıtıcı ve soğutucu birimlerinin güçlerinde yapılacak değişikliklerle tüm evlere entegre edilebilmesidir. İnsanların bulundukları ortamları kontrol eden algoritmanın insan diline yakın bir algoritma ile kontrol edilmesi süphesizdir ki insanlar için en iyi sonucu verecektir.

Tasarlanan sistem kullanılan bulanık mantık algortiması sürekli olarak koşturulmamakta, belirli sürelerde bir koşturulmaktadır. Odanın sıcaklığı ısıtıcının sürme gerilimi değiştiğinden hemen değişmediği için yani bir motor gibi gerilimi veya akım arttırınca motorun hızlanması gibi davranmadığı için belirli sürelerde bir çalıştırılmıştır. Bu süre her odaya, odanın şekline, odadaki hava akımına göre değişiklik göstermektedir.

Tasarlanan sistemde ve ev otomasyon sistemlerinde sıcaklık kontrol otomasyonun yapılmasının asıl sebebi, insanlara daha konforlu, maddi açıdan daha uygun olanın hedeflenmesidir.

Sistemde aydınlatma kontrol için bir oda içerisinde farklı çıkışlar ile sürülen ışık grupları oluşturulmuştur. Bunun nedeni insan göz sağlığını korumak için odanın her noktasında aynı ışık şiddetini sağlamak ve gereksiz yere enerji harcanmasını engellemektir. Akşam vakti, evinizin camından içeriye ışık girdiği için camınız önünde ışık istemeyebilir veya yeterli görebilirsiniz, arka tarafta kalan kısımda ise tam tersine ışık ihtiyacınız olabilir. Bu gibi durumlarda odanın aydınlatmasını açtığınız ışık kaynağı sürekli olarak sizin set ettiğiniz değerlerde çalışacaktır. Hangi noktanın ne kadar aydınlık seviyesinde olduğunu bilemeyecek ve sürekli aynı kademede aydınlatacaktır. Oysa tasarlanan sistemde kullanıcı tarafından set edilen aydınlık seviyesi belirli noktalardan ölçülerek o noktaya gerektiği kadar ışık şiddeti sağlanacak ve odanın her noktasında aynı ışık şiddetinin sağlanacaktır. Bu sayede gereksiz yere aydınlatma için kullanılan enerjiden tasarruf edilinmiş olunur.

İnsanoğlunun kalacak yer kadar kaldığı yerin güvenliğide önemlidir. Bu nedenle insanlar her zaman yaşadıkları konutların güvenliğini ön planda tutmuştur. Günümüzdeki ev otomasyon sistemleride güvenlik üzerine de çalışılmaktadır. Alarm sistemleri, telefon ile polise haber verme, izinsiz müdahale olduğunda kullanıcıya haber verme gibi birçok özelliğe sahip sistemler bulunmaktadır. Yapılan bu tezde ise güvenlik kontrolü, izinsiz giriş yapıldığında izinsiz giriş yapan kişiyi caydırma amaçlıdır. Eve izinsiz giriş olduğunda alarm sistemi ile aktif olarak sesli ve görsel uyarı sistemleri devreye alınabilmektedir.

Tasarımı yapılan ev sistemi öncelikle kullanıcının konforunu, sağılığını ve arkasındanda güvenliği sağlamayı hedeflemektedir.

KAYNAKLAR

Elmas Ç., (2003) "Bulanık Mantık Denetleyiciler" , Seçkin Yayınları, Ankara

Ergin, T., (2005) "Akıllı Evler, Yapay Zekanın Günlük Yaşantımızdaki Kullanımı"

Göğşen D. (2001) "Bina Otomasyonu Yüksek Lisans Tezi", İstanbul

Özek A., Sinecen M., (2003) "Klima Sistem Kontrolünün Bulanık Mantık İle Modellenmesi"

Passino K., Yurkovich S. (1998) "Fuzzy Control", Longman, California

Ross T., (1995) "Fuzzy Logic with Engineering Applications", McGraw- Hill, USA

Takagi T., Sugeno M., (1985) "Fuzzy Identification of Systems and Its Application to Modeling and Control", IEEE Transactions on Systems Man and Cybernetics, Vol. 15. No. 1.

Ünal A., (2004) "Aydınlatma Tasarımı ve Proje Uygulamaları", Birsen Yayınları, İstanbul

İNTERNET KAYNAKLARI

B&B Elektronics'in wweb sayfası, http://bb-elec.com/

C-Sharp ile ilgili web sayfası, www.csharpnedir.com

Elektrik işleri etüt idaresi genel müdürlüğü web sayfası, www.eie.gov.tr

Elektikle ilgili web sayfası, www.elektrik.gen.tr

Elektromania web sayfası, www.elektromania.net

Hayata dahiliz biz web sayfası, www.hayatadahiliz.biz

National Semiconductors'ın web sayfası,www.national.com/apnotes

Safetechub web sayfası, www.safetechup.com

EKLER

EK1 Oda Panosunun Yazılımı

```
*=============================================================
Dosya Adi: Main.h
Açiklama:
=============================================================*/
extern unsigned char Temp;
extern signed int Fuzzy_CripValue;

extern unsigned char bdata ProcessFlags;
extern bit Flag_ChangeOnSetTemp;
extern bit Flag_DisplayFlash;
extern bit Flag_RoomActive;
extern bit Flag_FuzzyControl;
extern unsigned char FlashCounter;
extern unsigned char ExCounter;

extern unsigned char ByteCounter;
extern unsigned char *StringPointer;
extern unsigned char *ReceivePointer;
```

```
/*============================================================
Dosya Adi: Main.c
Açiklama:
============================================================*/
//#pragma SRC
//#pragma SYMBOLS CODE DEBUG
#include <ADuC841.h>
//#include <intrins.h>            // rotate,nop islemleri için
#include <main.h>
#include <Interrupts.h>
#include <GeneralFunctions.h>
#include <fuzzy.h>
#include <7segment.h>
#include <Uart.h>


unsigned char Temp;
unsigned char FlashCounter;
signed int Fuzzy_CripValue;

unsigned char bdata ProcessFlags;
sbit Flag_ChangeOnSetTemp=ProcessFlags^0;
sbit Flag_DisplayFlash=ProcessFlags^1;
sbit Flag_RoomActive=ProcessFlags^2;
sbit Flag_FuzzyControl=ProcessFlags^3;
unsigned char FlashCounter;
unsigned char ExCounter;


unsigned char ByteCounter;
unsigned char *StringPointer;
unsigned char *ReceivePointer;


void main (void){
  MCU_Init();    // hiz  üç ikiye bölüyor bu seri portta sorun yaratabilir.
  ADC_Init();
  ADC_Calibration();
  Timer0Init();
  Timer1Init();
  ExtIntInit();
  InitUart();
  InitUartVariables();
  InitVariables();          //  tüm değişkenler atanarak ilk değerleri verilecek.


  while (1)
  {
   SCONV=1;
   ADCI=0;
if (Flag_ChangeOnSetTemp==1)
```

```
                {
                DisplayDrive(SetTemperature,LightSet);
                }
        else
                {
                 DisplayDrive(RoomTemperature,LightSet);
                 }
 if (ADCI==1)
                {
                ADCI=0;
                Temp=TakeADCResult();
                TakeAverage(Temp);
                }

 if ((FuzzyTimeCounter==FuzzyTime)&&(Flag_FuzzyControl==1))
                {
                FuzzyTimeCounter=0x00;
                                                        //----- Fuzzy algortimasi
                CalculateRoomTemperature();
                CalculateError();
                CalculateDeltaError();
                FuzzificationOfError();
                FuzzificationOfDeltaError();
                DecisionCycle();
                Fuzzy_CripValue=Defuzzificaiton();

                if (Fuzzy_CripValue > 0)
                        {
                        Heater=1*((unsigned char)Fuzzy_CripValue);
                        Cooler=0;
                        Cooler_Out=1;
                        }
                else if (Fuzzy_CripValue < 0)
                        {
                        Fuzzy_CripValue=-Fuzzy_CripValue;
                        Cooler=1*((unsigned char)Fuzzy_CripValue);
                        Heater=0;
                        Heater_Out=1;
                        }
            }
if (FlagPacketOK==1)
                {
                FlagPacketOK=0;
                 ReceivePointer=&ReceivedBytes;
                 PrepareUartBuffer();
                 SendingString(27,((unsigned char *)UartTransmitBuffer));
                 SendingChar(0x0D);
                 SendingChar(0x0A);
                }
            }
}
```

```
/*=========================================================
Dosya Adi: GeneralFunctions.h
Açiklama:  General Functions dosyasinda kullanilan fonsksiyonlar ve
                        degiskenlerin tanimlandigi alan.
=========================================================*/
//Prototypes    (Funstions).....................................
extern void Timer0Init(void);
extern void Timer1Init(void);
extern void ExtIntInit(void);
extern void InitVariables(void);
extern void MCU_Init(void);
extern void ADC_Init(void);
extern void ADC_Calibration(void);
extern void ADC_Calibration(void);
extern void TakeAverage(unsigned char ADCResult);
extern unsigned char TakeADCResult(void);
extern void ChangeADCChannel (void);


//   Constantss.....................................................
#define PERIOD     -250                 // 250 clock cycles interrupt period
#define AverageConstant 10
#define LimitTemp    2                                                  //displayi set
esnasinda flash yapmak için kullanilir.
#define P_Control_Mul        40
//Sicakliktaki P control Katsayisi
#define P_Control_Div        100
//Sicakliktaki P control Katsayisi

extern unsigned char Timer;
extern unsigned char Average[10];
extern unsigned char AverageCounter;

extern signed char LDR_1;
extern signed char LDR_2;
extern signed char LDR_3;

extern unsigned char TMP35_1;
extern unsigned char TMP35_2;
extern unsigned char TMP35_3;

extern unsigned char Cooler;
extern unsigned char Heater;
extern unsigned char LightSet;
```

```
/*=========================================================
Dosya Adi: General Functions.c
Açiklama:  Genel amaçli fonksiyonların tanimlandigi dosya
=========================================================*/
#include <ADuC841.h>
#include <intrins.h>            // rotate işlemleri için
#include <main.h>
#include <Interrupts.h>
#include <GeneralFunctions.h>
#include <fuzzy.h>
#include <7segment.h>
#include <Uart.h>

//cikis Fonksiyon adi (giris)
void Timer0Init(void);                  // giris ve cikis değeri yok.
void Timer1Init(void);
void ExtIntInit(void);
void InitVariables(void);
void MCU_Init(void);
void ADC_Init(void);
void ADC_Calibration(void);
void TakeAverage(unsigned char ADCResult); //  giris degeri istiyor.
unsigned char TakeADCResult(void);                  //  cikisda deger veriyor.
void ChangeADCChannel (void);
unsigned char LightControl (unsigned char LightSens,unsigned char
LightValue);


unsigned char Timer;
unsigned char Average[10];
unsigned char AverageCounter;

signed char LDR_1;
signed char LDR_2;
signed char LDR_3;

unsigned char TMP35_1;
unsigned char TMP35_2;
unsigned char TMP35_3;

unsigned char Cooler;
unsigned char Heater;
unsigned char LightSet;

void Timer0Init(void)
{
 TH0   = 0xFF;       // set timer period
 TL0   = 0x21;
 TMOD  |= TMOD | 0x01;              // select mode 1
 TR0   = 1;                         // start timer 0
```

```
 ET0   = 1;                        // enable timer 0 interrupt
 EA    = 1;                        // global interrupt enable
}
void Timer1Init(void)
{
 TH1   = 0xBB;       // set timer period
 TL1   = 0x21;
 TMOD  |= TMOD | 0x10;                // select mode 1
 TR1   = 1;                        // start timer 0
 ET1   = 1;                        // enable timer 0 interrupt
 EA    = 1;                        // global interrupt enable
}
void ExtIntInit(void)
            {
      IT0=1;              //Edge Sevsivity
     EX0=1;               // External Interrupt Enable;
      EA=1;
            }
void InitVariables(void)
{
 Timer0Counter=0x00;
 Timer1Counter=0x00;
 FuzzyTimeCounter=0;
 DisplayFlashTimeCounter=0;

 AverageCounter=0x00;
 Average[0]=0x00;
 Average[1]=0x00;
 Average[2]=0x00;
 Average[3]=0x00;
 Average[4]=0x00;
 Average[5]=0x00;
 Average[6]=0x00;
 Average[7]=0x00;
 Average[8]=0x00;
 Average[9]=0x00;

 LDR_1=0x00;
 LDR_2=0x00;
 LDR_3=0x00;
 TMP35_1=0x00;
 TMP35_2=0x00;
 TMP35_3=0x00;
 LightSet=0x00;
 SetTemperature=0x00;
 PreSetTemperature=0x00;
 Heater=0x00;
 Cooler=0x00;
 Fuzzy_CripValue=0;
 NewError=0;
 OldError=0;
```

```
 Flag_FuzzyControl=0;

 Heater_Out=1;
 Cooler_Out=1;
 Relay_Out=1;
 LedGroup_1=1;
 LedGroup_2=1;
 LedGroup_3=1;


 FlashTime=0;
 Flag_ChangeOnSetTemp=0;
 Flag_DisplayFlash=0;
 Flag_RoomActive=1;
 ExCounter=0x00;
}
void MCU_Init(void)
{
 //PLLCON |=PLLCON |0x03;          // Dis kristall i 8'e böler.
11.092Mhz/8=1.3824Mhz.
 //PLLCON |=PLLCON |0x00;          // Dis kristall i 1'e böler.
11.092Mhz/1=11.092Mhz
 CFG841=CFG841|0x01;
 EWAIT=0x07;
}
void ADC_Init(void)
{
 ADCCON1    = 0x8C; //10001100;
            /*                                  |||||||_EXC
                                      ||||||__T2C
                                            ||||||___AQ0
                                      |||||____AQ1
                                      ||||_____CK0
                                      |||______CK1
                                      ||_______EXT_REF
                                      |________MDI
  */
 ADCCON2 = 0x00; //00000011;
  /*                          |||||||_CS0
                                            ||||||__CS1
                                            |||||___CS2
                                            ||||____CS3
                                            |||_____SCONV
                                            |||______CCONV
                                            ||_______DMA
                                            |________ADCI
  */
ADCDATAH=0x00;
ADCDATAL=0x00;
}
void ADC_Calibration(void)
```

```
{
   ADCI=0;
ADCCON3 = 00000001;
while (ADCI)
          {;;}
  ADCI =0;
ADCCON3 = 00000011;
while (ADCI)
          {;;}
   ADCI=0;
   SCONV=1;
}
void TakeAverage (unsigned char FilterResult)
{
unsigned int Sum;
unsigned char i;
unsigned char Duty;
          i=0;
     Sum=0;
     Duty=0;
Average[AverageCounter]=FilterResult;
AverageCounter=AverageCounter+1;
if (AverageCounter>=AverageConstant)
             {
AverageCounter=0;
for (i=0;i<AverageConstant;i++)
                  {
                  Sum=Sum+Average[i];
                  }
Duty=Sum/AverageConstant;

if (ADCCON2==0x00)
                  {
                  Sum=Duty*100;       // 100 e oranlıyorum
                  Duty=Sum/255;
                  LDR_1=LightControl(Duty,LDR_1);
                  }
else if (ADCCON2==0x01)
                  {
                  Sum=Duty*100;       // 100 e oranlıyorum
                  Duty=Sum/255;
                  LDR_2=LightControl(Duty,LDR_2);
                  }
   else if (ADCCON2==0x02)
                  {
                  Sum=Duty*100;       // 100 e oranlıyorum
                  Duty=Sum/255;
                  LDR_3=LightControl(Duty,LDR_3);
                  }
else if (ADCCON2==0x03)
                  {
```

```
                TMP35_1=Duty;
                }
else if (ADCCON2==0x04)
                {
                TMP35_2=Duty;
                }
else if (ADCCON2==0x05)
                {
                TMP35_3=Duty;
                }
else if (ADCCON2==0x06)
                {
                LightSet=Duty;
                if (LightSet>99)
                        {LightSet=99;}
                }
else if (ADCCON2==0x07)
                {
                SetTemperature=Duty;
                if      (SetTemperature < 11)
                        {SetTemperature=10;}
                if      (SetTemperature >50)
                        {SetTemperature=50;}
                if (PreSetTemperature==0x00)
                        {PreSetTemperature=SetTemperature;}

                if ((PreSetTemperature<(SetTemperature-
LimitTemp))||(PreSetTemperature>(SetTemperature+LimitTemp)))
                        {
                         Flag_ChangeOnSetTemp=1;
                         Flag_DisplayFlash=1;
                         PreSetTemperature=SetTemperature;
                        }
                Flag_FuzzyControl=1;
                }
ChangeADCChannel();
            }
}
unsigned char TakeADCResult(void)
{
unsigned char ADC_H;
unsigned char ADC_L;

  ADC_H  = ADCDATAH;
ADC_H  = _crol_(ADC_H,4);
ADC_H  = ADC_H & 0xF0;
ADC_L  = ADCDATAL;
ADC_L  = _cror_(ADC_L,4);
ADC_L  = ADC_L & 0x0F;
ADC_H  = ADC_H | ADC_L;
```

```
//ADC_L  =0xFF;
//ADC_H  = ADC_L-ADC_H;

return ADC_H;
}
void ChangeADCChannel (void)
{
if (ADCCON2 == 0x00)
                {ADCCON2 = 0x01;}
else if (ADCCON2 == 0x01)
                {ADCCON2 = 0x02;}
else if (ADCCON2 == 0x02)
                {ADCCON2 = 0x03;}
else if (ADCCON2 == 0x03)
                {ADCCON2 = 0x04;}
else if (ADCCON2 == 0x04)
                {ADCCON2 = 0x05;}
else if (ADCCON2 == 0x05)
                {ADCCON2 = 0x06;}
else if (ADCCON2 == 0x06)
                {ADCCON2 = 0x07;}
else if (ADCCON2 == 0x07)
                {ADCCON2 = 0x00;}
}
unsigned char LightControl (unsigned char LightSens,unsigned char
LightValue)
{
unsigned char LightError;
unsigned char LightOutput;
LightError=0;
LightOutput=LightValue;
LightSens=100-LightSens;
LightSens=(LightSens/2);

if (LightSet < LightSens)
            {
LightError=LightSens-LightSet;
if (LightError > LightOutput)
                {       LightOutput=2; }
        else
                {
                LightError=P_Control_Mul*LightError;
                LightError=LightError/P_Control_Div;
                LightOutput=LightOutput-LightError;
                }
            }
else if (LightSet > LightSens)
            {
LightError=LightSet-LightSens;
LightError=P_Control_Mul*LightError;
LightError=LightError/P_Control_Div;
```

```
if (LightOutput<99)
                    {
                    LightOutput=LightOutput+LightError;
                    }
if (LightOutput>98)
                    {
                    LightOutput=99;
                    }
               }
 /*
if (ADCCON2==0x00)
               {
SendingChar('G');
SendingChar((LightSens/100)+0x30);
SendingChar(((LightSens%100)/10)+0x30);
SendingChar((LightSens%10)+0x30);

SendingChar('S');
SendingChar((LightSet/100)+0x30);
SendingChar(((LightSet%100)/10)+0x30);
SendingChar((LightSet%10)+0x30);

SendingChar('H');
if(LightError<0)
                    {LightError=LightError*(-1);}
SendingChar((LightError/100)+0x30);
SendingChar(((LightError%100)/10)+0x30);
SendingChar((LightError%10)+0x30);

SendingChar('C');
SendingChar((LightOutput/100)+0x30);
SendingChar(((LightOutput%100)/10)+0x30);
SendingChar((LightOutput%10)+0x30);
SendingChar(0x0D);
SendingChar(0x0A);
               }
              */
return LightOutput;
}
```

```
/*============================================================
Dosya Adi: Int_header.h
Açiklama:  Interrupt ve Interruptta kullanilan fonksyionları içerir.
============================================================*/
extern unsigned char Timer0Counter;              // variable to count interrupts
extern unsigned char Timer1Counter;              // variable to count interrupts
extern unsigned char FuzzyTimeCounter;
extern unsigned int DisplayFlashTimeCounter;


sbit LedGroup_1 = P3^3;
sbit LedGroup_2 = P3^4;
sbit LedGroup_3 = P3^5;
sbit Heater_Out       = P3^6;
sbit Cooler_Out       = P3^7;

sbit Relay_Out= P2^3;


#define Period     100                  //PWM'in periyodu.
#define FuzzyTime  100                        // fuzzy algoritmasinin kaç saniyede
bir kosması için gerekli.
#define DisplayFlashTime 1000
```

```
/*============================================================
Dosya Adi: Interrupt.c
Açiklama:  Interrupt ve Interruptta kullanilan fonksyionları içerir.
============================================================*/
#include <ADuC841.h>
#include <main.h>
#include <Interrupts.h>
#include <GeneralFunctions.h>
#include <fuzzy.h>
#include <7segment.h>
#include <Uart.h>

 unsigned char Timer0Counter;              // variable to count interrupts
 unsigned char Timer1Counter;              // variable to count interrupts
 unsigned char FuzzyTimeCounter;
 unsigned int DisplayFlashTimeCounter;

void timer0 (void) interrupt 1 using 1{   // Int Vector at 000BH, Reg Bank 1
  TH0   = 0xFD;  //0xC9;
  TL0   = 0xAF;         //0xFF;

  Timer0Counter++;                        // increment interrupt counter
  if (Timer0Counter==Period)
             {
Timer0Counter=0x00;
LedGroup_1=0;
LedGroup_2=0;
LedGroup_3=0;

             }
  if (Timer0Counter==LDR_1)
  { LedGroup_1=1;              }// LED_1 kapatildi.

  if (Timer0Counter==LDR_2)
   { LedGroup_2=1;              }// LED_2 kapatildi.

  if (Timer0Counter==LDR_3)
   { LedGroup_3=1;              }// LED_3 kapatildi.
}
void timer1 (void) interrupt 3 using 1{                   // Int Vector at 000BH,
Reg Bank 1
  TH1   = 0xFA;       // set timer period
  TL1   = 0x00;
  Timer1Counter++;
  if (Timer1Counter==Period)
                  {
                  Timer1Counter=0;
                  if (Heater !=0x00)
                           {Heater_Out=0;}
                  else
                           {Heater_Out=1;}
```

```
                if  (Cooler !=0x00)
                        {Cooler_Out=0;}
                else
                        {Cooler_Out=1;}
                }
  if ((Timer1Counter==Heater)&&(Heater!=0x00))
                { Heater_Out=1;                 }// Heater kapatildi.

  if ((Timer1Counter==Cooler)&&(Cooler!=0x00))
                { Cooler_Out=1;                 }// Cooler kapatildi.

  FuzzyTimeCounter++;
  if (Flag_ChangeOnSetTemp==1)
            {
  DisplayFlashTimeCounter++;
  if (DisplayFlashTimeCounter==DisplayFlashTime)
                {
                DisplayFlashTimeCounter=0;
                if (Flag_DisplayFlash==1)
                        {Flag_DisplayFlash=0;}
                else
                        {Flag_DisplayFlash=1;}
                FlashTime++;
                if (FlashTime==100)
                        {
                        FlashTime=0;
                        Flag_ChangeOnSetTemp=0;
                        Flag_DisplayFlash=0;
                        }
                }
            }
}

void ExtInt (void) interrupt 0 using 1 {
 ExCounter++;
if (ExCounter==0x01)
            {
ExCounter=0;
if (Flag_RoomActive==1)
                        {
                        Relay_Out=0;
                        Flag_RoomActive=0;
                        }
                else
                        {
                        Relay_Out=1;
                        Flag_RoomActive=1;
                        }
            }

}
```

```
/*=========================================================
Dosya Adi: fuzzy.h
Açiklama:  fuzzy.c dosyasında kullanilan degiskenlerin belirtildigi,
                        Fonksiyonlarin prototypelerinin yazildigi alan.
=========================================================*/
//   Constants .....................................................
//Fuzzy Uyelik Fonksiyonlari
#define ResolutionOfDegree              100
#define NotADegree                              0xAA
// Error'un üyelik fonksiyonlarinin başlangiç ve kesisim noktalari
#define ErrorTopOfNB                            10
#define ErrorEndOfNK                            12
#define ErrorStartOfNB                          14
#define ErrorTopOfNK                            15
#define ErrorStartOfZero                16
#define ErrorStartOfNK                          18
#define ErrorTopOfZero                          20
#define ErrorStartOfPK                          22
#define ErrorEndOfZero                          24
#define ErrorTopOfPK                            25
#define ErrorStartOfPB                          26
#define ErrorEndOfPK                            28
#define ErrorTopOfPB                            30

// DeltaError'un üyelik fonksiyonlarinin başlangiç ve kesisim noktalari
#define DeltaErrorTopOfNB               10
#define DeltaErrorEndOfNK               12
#define DeltaErrorStartOfNB                     14
#define DeltaErrorTopOfNK               15
#define DeltaErrorStartOfZero           16
#define DeltaErrorStartOfNK                     18
#define DeltaErrorTopOfZero                     20
#define DeltaErrorStartOfPK             22
#define DeltaErrorEndOfZero                     24
#define DeltaErrorTopOfPK               25
#define DeltaErrorStartOfPB             26
#define DeltaErrorEndOfPK               28
#define DeltaErrorTopOfPB               30

//üyelik fonksiyonlari
#define NB      0
#define NK      1
#define Zero    2
#define PK      3
#define PB      4


//   SubRoutines.....................................................
extern void CalculateRoomTemperature (void);
extern void CalculateError(void);
extern void CalculateDeltaError(void);
```

```
extern void FuzzificationOfError (void);
extern void FuzzificationOfDeltaError (void);
extern void DecisionCycle (void);
extern signed char Defuzzificaiton (void);

//    Ports and Port Pins.....................................

//    Variables  (Bytes, Bits) ..................................
extern signed char xdata NewError;
extern signed char xdata OldError;
extern signed char xdata DeltaError;
extern unsigned char RoomTemperature;
extern unsigned char PreSetTemperature;
extern unsigned char SetTemperature;
extern unsigned char xdata DegreeOfError[4];                    //
DegreeOfError[0]=ilk u degeri,  DegreeOfError[1]=Üyelik fonksiyonu
DegreeOfError[2]=diger u degeri DegreeOfError[3]=diger u degerinin üyelik
fonksiyonu.
extern unsigned char xdata DegreeOfDeltaError[4];    //
DegreeOfDeltaError[0]=ilk u degeri,  DegreeOfDeltaError[1]=Üyelik
fonksiyonu .....
extern unsigned char xdata Decisions[8];
```

```
/*=========================================================
Dosya Adi: Fuzzy.c
=========================================================*/
#include <ADuC841.h>
//#include <intrins.h>          // rotate,nop islemleri için
#include <main.h>
#include <Interrupts.h>
#include <GeneralFunctions.h>
#include <fuzzy.h>
#include <7segment.h>
#include <Uart.h>


//  Subroutines .....................................
void CalculateRoomTemperature (void);
void CalculateError(void);
void CalculateDeltaError(void);
void FuzzificationOfError (void);
void FuzzificationOfDeltaError (void);
unsigned char MaxDotFunction (unsigned char Error,unsigned char DeltaError);
void DecisionCycle (void);
unsigned char MakeDecisionFromTable (unsigned char Error,unsigned char
DeltaError);
signed char Defuzzificaiton (void);
signed char FindCenterOfMembershipFunction (unsigned char
MembershipFunction);

//  Variables  (Bytes, Bits) .....................................
unsigned char RoomTemperature;
unsigned char PreSetTemperature;
unsigned char SetTemperature;

signed char xdata NewError;
signed char xdata OldError;
signed char xdata DeltaError;
unsigned  char xdata DegreeOfError[4];                    //
DegreeOfError[0]=ilk u degeri,  DegreeOfError[1]=Üyelik fonksiyonu
DegreeOfError[2]=diger u degeri DegreeOfError[3]=diger u degerinin üyelik
fonksiyonu.
unsigned  char xdata DegreeOfDeltaError[4];    // DegreeOfDeltaError[0]=ilk u
degeri,  DegreeOfDeltaError[1]=Üyelik fonksiyonu .....
unsigned  char xdata Decisions[8];                        // Kural tablosu ve
karar kuralinin seçtiği u degerleri bulunur.


//Kural Tablosu
unsigned  const code RuleLine1[5]={Zero, PK , PB , PB , PB };
unsigned  const code RuleLine2[5]={ NK ,Zero, PK , PB , PB };
unsigned  const code RuleLine3[5]={ NB , NK ,Zero, PK , PB };
unsigned  const code RuleLine4[5]={ NB , NB , NK ,Zero, PK };
unsigned  const code RuleLine5[5]={ NB , NB , NB , NK , Zero};
```

```
//   FUNCTIONS
/*=========================================================
Fonksiyon Adi: CalculateError
Açiklama:   Kontrol algoritmasi için gerekli hatayi bulur.
                        -Sicaklik degerleri düzeltilip LastTemperature,
PreviousTemperature
                        içerisine koyulmuş olmalidir.
                        -Hata=Set-Ölçülen
Cikis:NewError ve OldError  degerler milisantigrad
=========================================================*/
void CalculateRoomTemperature (void){
unsigned char AvaregeOfTMPs;

AvaregeOfTMPs=TMP35_1;
AvaregeOfTMPs=AvaregeOfTMPs+TMP35_2;
AvaregeOfTMPs=AvaregeOfTMPs+TMP35_3;
AvaregeOfTMPs=AvaregeOfTMPs/3;
RoomTemperature=AvaregeOfTMPs;}
/*=========================================================
Fonksiyon Adi: CalculateError
Açiklama:   Kontrol algoritmasi için gerekli hatayi bulur.
                        -Sicaklik degerleri düzeltilip LastTemperature,
PreviousTemperature
                        içerisine koyulmuş olmalidir.
                        -Hata=Set-Ölçülen
Cikis:NewError ve OldError  degerler milisantigrad
=========================================================*/
void CalculateError (void){
//OldError=NewError;
//NewError=SetTemperature-RoomTemperature;
NewError=SetTemperature;
}
/*=========================================================
Fonksiyon Adi: CalculateDeltaError
Açiklama:   Kontrol algoritmasi için gerekli hatanin degisimini bulur.
                                degerler milisantigrad
=========================================================*/
void CalculateDeltaError (void)
{// DeltaError=(NewError-OldError);
 DeltaError=RoomTemperature; }
/*=========================================================
Fonksiyon Adi: FuzzificationOfError
Açiklama:       NewError degerlerini bulaniklastirir.
degerler milisantigrad
=========================================================*/
void FuzzificationOfError (void){
signed int MemberShip;

MemberShip=0;
if (NewError < ErrorTopOfNB){
```

```
DegreeOfError[0]=ResolutionOfDegree;
DegreeOfError[1]=NB;

DegreeOfError[2]=NotADegree;                    // deger 99
degerinden buyuk ise o DegreeOfErroryoktur.
DegreeOfError[3]=NotADegree; }
else if (NewError < ErrorEndOfNK) {
                MemberShip=0;
                MemberShip=(25*NewError)*(-1);          // y=-25x+350
                MemberShip=MemberShip+350;
                DegreeOfError[0]=MemberShip;
                DegreeOfError[1]=NB;

                DegreeOfError[2]=NotADegree;                    //
deger 99 degerinden buyuk ise o DegreeOfErroryoktur.
   DegreeOfError[3]=NotADegree; }
else if (NewError < ErrorStartOfNB)  {
MemberShip=0;
MemberShip=(25*NewError)*(-1);          // y=-25x+350
MemberShip=MemberShip+350;
DegreeOfError[0]=MemberShip;
DegreeOfError[1]=NB;

MemberShip=0;
MemberShip=100*NewError;                // y=100x-1200
MemberShip=MemberShip-1200;
DegreeOfError[2]=MemberShip;
DegreeOfError[3]=NK; }
else if (NewError < ErrorTopOfNK) {
MemberShip=0;
MemberShip=100*NewError;                // y=100x-1200
MemberShip=MemberShip-1200;
DegreeOfError[0]=MemberShip;
DegreeOfError[1]=NK;

DegreeOfError[2]=NotADegree;                    // deger 99
degerinden buyuk ise o DegreeOfErroryoktur.
DegreeOfError[3]=NotADegree;      }
else if (NewError < ErrorStartOfZero) {
MemberShip=0;
MemberShip=(100*NewError)*(-1);         // y=-(100*x)/3+600
MemberShip=MemberShip/3+600;
DegreeOfError[0]=MemberShip;
DegreeOfError[1]=NK;

DegreeOfError[2]=NotADegree;                    // deger 99
degerinden buyuk ise o DegreeOfErroryoktur.
DegreeOfError[3]=NotADegree; }
else if (NewError < ErrorStartOfNK) {
MemberShip=0;
MemberShip=(100*NewError)*(-1);         // y=-(100*x)/3+600
```

```
MemberShip=MemberShip/3+600;
DegreeOfError[0]=MemberShip;
DegreeOfError[1]=NK;

MemberShip=0;
MemberShip=25*NewError;                    // y=25x-400
MemberShip=MemberShip-400;
DegreeOfError[2]=MemberShip;
DegreeOfError[3]=Zero;       }
else if (NewError < ErrorTopOfZero) {
MemberShip=0;
MemberShip=25*NewError;                    // y=25x-400
MemberShip=MemberShip-400;
DegreeOfError[0]=MemberShip;
DegreeOfError[1]=Zero;
DegreeOfError[2]=NotADegree;                              // deger 99
degerinden buyuk ise o DegreeOfErroryoktur.
DegreeOfError[3]=NotADegree;  }
else if (NewError < ErrorStartOfPK)  {
MemberShip=0;
MemberShip=(25*NewError)*(-1);                  // y=-25*x+600
MemberShip=MemberShip+600;
DegreeOfError[0]=MemberShip;
DegreeOfError[1]=Zero;
DegreeOfError[2]=NotADegree;                              // deger 99
degerinden buyuk ise o DegreeOfErroryoktur.
DegreeOfError[3]=NotADegree;      }
else if (NewError < ErrorEndOfZero) {
MemberShip=0;
MemberShip=(25*NewError)*(-1);                  // y=-25*x+600
MemberShip=MemberShip+600;
DegreeOfError[0]=MemberShip;
DegreeOfError[1]=Zero;

MemberShip=0;
MemberShip=100*NewError;                        // y=100x/3-2200/3 =100x/3-
733
MemberShip=MemberShip/3-733;
DegreeOfError[2]=MemberShip;
DegreeOfError[3]=PK;                }

else if (NewError < ErrorTopOfPK) {
MemberShip=0;
MemberShip=100*NewError;                        // y=100x/3-2200/3 =100x/3-
733
MemberShip=MemberShip/3-733;
DegreeOfError[0]=MemberShip;
DegreeOfError[1]=PK;
DegreeOfError[2]=NotADegree;                              // deger 99
degerinden buyuk ise o DegreeOfErroryoktur.
DegreeOfError[3]=NotADegree;      }
```

```
else if (NewError < ErrorStartOfPB) {
MemberShip=0;
MemberShip=(100*NewError)*(-1);                    // y=-(100*x)/3+2800/3
MemberShip=MemberShip/3+933;
DegreeOfError[0]=MemberShip;
DegreeOfError[1]=PK;
DegreeOfError[2]=NotADegree;                              // deger 99
degerinden buyuk ise o DegreeOfErroryoktur.
DegreeOfError[3]=NotADegree;      }
else if (NewError < ErrorEndOfPK)  {
MemberShip=0;
MemberShip=(100*NewError)*(-1);                    // y=-(100*x)/3+2800/3
MemberShip=MemberShip/3+933;
DegreeOfError[0]=MemberShip;
DegreeOfError[1]=PK;

MemberShip=0;
MemberShip=25*NewError;                    // y=25x-650
MemberShip=MemberShip-650;
DegreeOfError[2]=MemberShip;
DegreeOfError[3]=PB;                       }
else if (NewError < ErrorTopOfPB)  {
MemberShip=0;
MemberShip=25*NewError;           // y=25x-650
MemberShip=MemberShip-650;
DegreeOfError[0]=MemberShip;
DegreeOfError[1]=PB;
DegreeOfError[2]=NotADegree;                              // deger 99
degerinden buyuk ise o DegreeOfErroryoktur.
DegreeOfError[3]=NotADegree;      }
else if (NewError >= ErrorTopOfPB)          {
DegreeOfError[0]=ResolutionOfDegree;
DegreeOfError[1]=PB;                               // deger 99 degerinden buyuk
ise o DegreeOfErroryoktur.
DegreeOfError[2]=NotADegree;                              // deger 99
degerinden buyuk ise o DegreeOfErroryoktur.
DegreeOfError[3]=NotADegree;      }
}
/*============================================================
Fonksiyon Adi: FuzzificationOfDeltaError
Açiklama:       DeltaError degerlerini bulaniklastirir.
degerler milisantigrad
============================================================*/
void FuzzificationOfDeltaError (void)
{
signed int MemberShipDeltaError;
MemberShipDeltaError=0;

if (DeltaError < DeltaErrorTopOfNB){
DegreeOfDeltaError[0]=ResolutionOfDegree;
DegreeOfDeltaError[1]=NB;
```

```
DegreeOfDeltaError[2]=NotADegree;                    // deger 99
degerinden buyuk ise o DegreeOfErroryoktur.
DegreeOfDeltaError[3]=NotADegree;}
else if (DeltaError < DeltaErrorEndOfNK){
                MemberShipDeltaError=0;
                MemberShipDeltaError=(25*NewError)*(-1);
// y=-25x+350
                MemberShipDeltaError=MemberShipDeltaError+350;
                DegreeOfDeltaError[0]=MemberShipDeltaError;
                DegreeOfDeltaError[1]=NB;
                DegreeOfDeltaError[2]=NotADegree;
                DegreeOfDeltaError[3]=NotADegree;}
else if (DeltaError < DeltaErrorStartOfNB){
MemberShipDeltaError=0;
MemberShipDeltaError=(25*NewError)*(-1);             // y=-25x+350
MemberShipDeltaError=MemberShipDeltaError+350;
DegreeOfDeltaError[0]=MemberShipDeltaError;
DegreeOfDeltaError[1]=NB;

MemberShipDeltaError=0;
MemberShipDeltaError=100*NewError;                   // y=100x-1200
MemberShipDeltaError=MemberShipDeltaError-1200;
DegreeOfDeltaError[2]=MemberShipDeltaError;
DegreeOfDeltaError[3]=NK; }
else if (DeltaError < DeltaErrorTopOfNK){
MemberShipDeltaError=0;
MemberShipDeltaError=100*NewError;                   // y=100x-1200
MemberShipDeltaError=MemberShipDeltaError-1200;
DegreeOfDeltaError[0]=MemberShipDeltaError;
DegreeOfDeltaError[1]=NK;
DegreeOfDeltaError[2]=NotADegree;
DegreeOfDeltaError[3]=NotADegree;        }
else if (DeltaError < DeltaErrorStartOfZero){
MemberShipDeltaError=0;
MemberShipDeltaError=(100*NewError)*(-1);            // y=-
100x/3+600
MemberShipDeltaError=MemberShipDeltaError/3+600;
DegreeOfDeltaError[0]=MemberShipDeltaError;
DegreeOfDeltaError[1]=NK;
DegreeOfDeltaError[2]=NotADegree;
DegreeOfDeltaError[3]=NotADegree;        }
else if (DeltaError < DeltaErrorStartOfNK)  {
MemberShipDeltaError=0;
MemberShipDeltaError=(100*NewError)*(-1);            // y=-
100x/3+600
MemberShipDeltaError=MemberShipDeltaError/3+600;
DegreeOfDeltaError[0]=MemberShipDeltaError;
DegreeOfDeltaError[1]=NK;

MemberShipDeltaError=0;
MemberShipDeltaError=25*NewError;                    // y=25x-400
```

```
MemberShipDeltaError=MemberShipDeltaError-400;
DegreeOfDeltaError[2]=MemberShipDeltaError;
DegreeOfDeltaError[3]=Zero;                }
else if (DeltaError < DeltaErrorTopOfZero) {
MemberShipDeltaError=0;
MemberShipDeltaError=25*NewError;                    // y=25x-400
MemberShipDeltaError=MemberShipDeltaError-400;
DegreeOfDeltaError[0]=MemberShipDeltaError;
DegreeOfDeltaError[1]=Zero;
DegreeOfDeltaError[2]=NotADegree;
DegreeOfDeltaError[3]=NotADegree;        }
else if (DeltaError < DeltaErrorStartOfPK){
MemberShipDeltaError=0;
MemberShipDeltaError=(25*NewError)*(-1);                    // y=-25x+600
MemberShipDeltaError=MemberShipDeltaError+600;
DegreeOfDeltaError[0]=MemberShipDeltaError;
DegreeOfDeltaError[1]=Zero;
DegreeOfDeltaError[2]=NotADegree;
DegreeOfDeltaError[3]=NotADegree;        }
else if (DeltaError < DeltaErrorEndOfZero){
MemberShipDeltaError=0;
MemberShipDeltaError=(25*NewError)*(-1);                    // y=-25x+600
MemberShipDeltaError=MemberShipDeltaError+600;
DegreeOfDeltaError[0]=MemberShipDeltaError;
DegreeOfDeltaError[1]=Zero;

MemberShipDeltaError=0;
MemberShipDeltaError=100*NewError;                    // y=(100*x)/3-633
MemberShipDeltaError=MemberShipDeltaError/3-633;
DegreeOfDeltaError[2]=MemberShipDeltaError;
DegreeOfDeltaError[3]=PK; }

else if (DeltaError < DeltaErrorTopOfPK){
MemberShipDeltaError=0;
MemberShipDeltaError=100*NewError;                    // y=(100*x)/3-633
MemberShipDeltaError=MemberShipDeltaError/3-633;
DegreeOfDeltaError[0]=MemberShipDeltaError;
DegreeOfDeltaError[1]=PK;
DegreeOfDeltaError[2]=NotADegree;
DegreeOfDeltaError[3]=NotADegree;        }
else if (DeltaError < DeltaErrorStartOfPB){
MemberShipDeltaError=0;
MemberShipDeltaError=(100*NewError)*(-1);                    // y=-
100x/3+933
MemberShipDeltaError=MemberShipDeltaError/3+933;
DegreeOfDeltaError[0]=MemberShipDeltaError;
DegreeOfDeltaError[1]=PK;
DegreeOfDeltaError[2]=NotADegree;
DegreeOfDeltaError[3]=NotADegree;}
else if (DeltaError < DeltaErrorEndOfPK){
MemberShipDeltaError=0;
```

```
MemberShipDeltaError=(100*NewError)*(-1);                    // y=-
100x/3+933
MemberShipDeltaError=MemberShipDeltaError/3+933;
DegreeOfDeltaError[0]=MemberShipDeltaError;
DegreeOfDeltaError[1]=PK;

MemberShipDeltaError=0;
MemberShipDeltaError=25*NewError;                    // y=25x-650
MemberShipDeltaError=MemberShipDeltaError-650;
DegreeOfDeltaError[2]=MemberShipDeltaError;
DegreeOfDeltaError[3]=PB; }
else if (DeltaError < DeltaErrorTopOfPB){
MemberShipDeltaError=0;
MemberShipDeltaError=25*NewError;                    // y=25x-650
MemberShipDeltaError=MemberShipDeltaError-650;
DegreeOfDeltaError[0]=MemberShipDeltaError;
DegreeOfDeltaError[1]=PB;
DegreeOfDeltaError[2]=NotADegree;
DegreeOfDeltaError[3]=NotADegree;}
else if (DeltaError >= DeltaErrorTopOfPB){
DegreeOfDeltaError[0]=ResolutionOfDegree;
DegreeOfDeltaError[1]=PB;
DegreeOfDeltaError[2]=NotADegree;                    // deger 99
degerinden buyuk ise o DegreeOfErroryoktur.
DegreeOfDeltaError[3]=NotADegree;}                    // deger 99
degerinden buyuk ise o DegreeOfErroryoktur.
}
/*============================================================
Fonksiyon Adi:  DecisionCycle
Açiklama:       BUlanklastirilmis degerleri tabloya göre bulanık sonuç verir.
degerler Bulaniktir.
============================================================*/
void DecisionCycle (void){
unsigned char Error;
unsigned char DeltaError;
Error=DegreeOfError[1];
DeltaError=DegreeOfDeltaError[1];
Decisions[0]=MakeDecisionFromTable(Error,DeltaError);

Error=DegreeOfError[0];
DeltaError=DegreeOfDeltaError[0];
Decisions[1]=MaxDotFunction(Error,DeltaError);

Error=DegreeOfError[1];
DeltaError=DegreeOfDeltaError[3];
Decisions[2]=MakeDecisionFromTable(Error,DeltaError);
Error=DegreeOfError[0];
DeltaError=DegreeOfDeltaError[2];
Decisions[3]=MaxDotFunction(Error,DeltaError);

Error=DegreeOfError[3];
```

```
DeltaError=DegreeOfDeltaError[1];
Decisions[4]=MakeDecisionFromTable(Error,DeltaError);
Error=DegreeOfError[2];
DeltaError=DegreeOfDeltaError[0];
Decisions[5]=MaxDotFunction(Error,DeltaError);

Error=DegreeOfError[3];
DeltaError=DegreeOfDeltaError[3];
Decisions[6]=MakeDecisionFromTable(Error,DeltaError);

Error=DegreeOfError[2];
DeltaError=DegreeOfDeltaError[2];
Decisions[7]=MaxDotFunction(Error,DeltaError);  }

/*============================================================
Fonksiyon Adi:  MakeDecisionFromTable
Açiklama:       Girilen Error ve Delta Error'e göre tabloda karar çikartir.
degerler Bulaniktir.                s
============================================================*/
unsigned char MakeDecisionFromTable (unsigned char Error,unsigned char
DeltaError){
unsigned char Decision;

Decision=0;


if ((DeltaError==NotADegree)||(Error==NotADegree)){
Decision=NotADegree;}
else{
if ((DeltaError)==NB){
                Decision=RuleLine1[Error];}
else if ((DeltaError)==NK){
                Decision=RuleLine2[Error];  }
else if ((DeltaError)==Zero){
                Decision=RuleLine3[Error];}
else if ((DeltaError)==PK){
                Decision=RuleLine4[Error];}
else if ((DeltaError)==PB){
                Decision=RuleLine5[Error];}
            }
return Decision;
}

/*============================================================
Fonksiyon Adi:  MaxDotFunction
Açiklama:       u degerleri içinden büyük olani alir.
degerler Bulaniktir.
============================================================*/
unsigned char MaxDotFunction (unsigned char Error,unsigned char DeltaError){
unsigned char MaxDot;
```

```
if ((DeltaError==NotADegree)||(Error==NotADegree)){
MaxDot=NotADegree;}
else{
 if (DeltaError>Error)
                {MaxDot=DeltaError;}
          else
                {MaxDot=Error;}
             }
return MaxDot;
}

/*============================================================
Fonksiyon Adi:  Defuzzification
Açiklama:       Bulanik degerleri kesin degerlere dönüştürür.
                                Agırlıklı Ortalama Methodu ile dönüsüm
yapilmistir.
============================================================*/
signed char Defuzzificaiton (void){
signed int Mul1;
signed int Mul2;
signed int Mul3;
signed int Mul4;
signed int Crisp;
signed int Addition;

signed int Center;
Center=0;
Crisp=0;
Mul1=0;
Mul2=0;
Mul3=0;
Mul4=0;

if (Decisions[0]!=NotADegree){
Center=FindCenterOfMembershipFunction(Decisions[0]);
Mul1=Center * Decisions[1];}
else
 {Mul1=0;}

if (Decisions[2]!=NotADegree)
{ Center=FindCenterOfMembershipFunction(Decisions[2]);
  Mul2=Center * Decisions[3]; }
else{Mul2=0;}

if (Decisions[4]!=NotADegree){
Center=FindCenterOfMembershipFunction(Decisions[4]);
Mul3=Center * Decisions[5]; }
else{Mul3=0;}

if (Decisions[7]!=NotADegree){
Center=FindCenterOfMembershipFunction(Decisions[6]);
```

```
Mul4=Center * Decisions[7];}
else {Mul4=0;}
Center=Mul1+Mul2;
Center=Center+Mul3;
Center=Center+Mul4;


Addition=0;
if (Decisions[1]!=NotADegree){Addition=Decisions[1];}
if (Decisions[3]!=NotADegree){Addition=Addition+Decisions[3];}
if (Decisions[5]!=NotADegree){Addition=Addition+Decisions[5];}
if (Decisions[7]!=NotADegree){Addition=Addition+Decisions[7];}

Crisp=Center/Addition;

return Crisp;
}
/*============================================================
Fonksiyon Adi:  FindCenterOfMembershipFunction
Açiklama:       Bulanik degerleri kesin degerlere dönüştürür.
                                Agirlikli Ortalama Methodu ile dönüsüm
yapilmistir.
============================================================*/
signed char FindCenterOfMembershipFunction (unsigned char
MembershipFunction){
unsigned char CenterOfMembership;

CenterOfMembership=0;
if (MembershipFunction==NB)
{CenterOfMembership=-90;}
else if  (MembershipFunction==NK)
{CenterOfMembership=-50;}
else if  (MembershipFunction==Zero)
{CenterOfMembership=0;}
else if  (MembershipFunction==PK)
{CenterOfMembership=50;}
else if  (MembershipFunction==PB)
{CenterOfMembership=90;}
return CenterOfMembership;
}
```

```
/*=========================================================

Dosya Adi: GeneralFunctions.h
Açiklama:  General Functions dosyasinda kullanilan fonsksiyonlar ve
                        degiskenlerin tanimlandigi alan.
=========================================================*/
//Prototypes    (Funstions).....................................
extern void Timer0Init(void);
extern void Timer1Init(void);
extern void ExtIntInit(void);
extern void InitVariables(void);
extern void MCU_Init(void);
extern void ADC_Init(void);
extern void ADC_Calibration(void);
extern void ADC_Calibration(void);
extern void TakeAverage(unsigned char ADCResult);
extern unsigned char TakeADCResult(void);
extern void ChangeADCChannel (void);


//   Constantss.....................................................
#define PERIOD     -250                 // 250 clock cycles interrupt period
#define AverageConstant 10
#define LimitTemp    2                                                  //displayi set
esnasinda flash yapmak için kullanilir.
#define P_Control_Mul        40
//Sicakliktaki P control Katsayisi
#define P_Control_Div        100
//Sicakliktaki P control Katsayisi

extern unsigned char Timer;
extern unsigned char Average[10];
extern unsigned char AverageCounter;

extern signed char LDR_1;
extern signed char LDR_2;
extern signed char LDR_3;

extern unsigned char TMP35_1;
extern unsigned char TMP35_2;
extern unsigned char TMP35_3;

extern unsigned char Cooler;
extern unsigned char Heater;
extern unsigned char LightSet;
```

```
/*=========================================================
Dosya Adi: General Functions.c
Açiklama:  Genel amaçli fonksiyonların tanimlandigi dosya
=========================================================*/
#include <ADuC841.h>
#include <intrins.h>              // rotate işlemleri için
#include <main.h>
#include <Interrupts.h>
#include <GeneralFunctions.h>
#include <fuzzy.h>
#include <7segment.h>
#include <Uart.h>



//fonksiyon Protype lari
void SendSCKClock (void);
void SendRCKClock (void);
void SendBitBit(unsigned char Byte);
void Prepare7SegmentData(unsigned char DisplayValue);
void DisplayDrive (unsigned char Temperature, unsigned char Light);

// degiskenler
unsigned char code
SevenSegmentTab[14]={0x84,0xAF,0x98,0x89,0xA3,0xC1,0xC0,0x8F,0x80,0x
81,0x93,0xD4,0xE8,0xFF};
//                                                                    '0'   '1'  '2'
'3'  '4'  '5'  '6'  '7'  '8' '9'   '°'      'C'        '%' 'Blank'
unsigned char FlashTime;

void SendSCKClock (void){
    SCK=0;
    _nop_();
    _nop_();
    _nop_();
    SCK=1;
    _nop_();
    _nop_();
    _nop_();
    SCK=0;
    _nop_();
    _nop_();
    _nop_();
}
void SendRCKClock (void){
    RCK=0;
    _nop_();
    _nop_();
    _nop_();
    RCK=1;
```

```
    _nop_();
    _nop_();
    _nop_();
    RCK=0;
    _nop_();
    _nop_();
    _nop_();
}
void SendBitBit(unsigned char Byte){
unsigned char Temp_Byte;
unsigned char BitCounter;
BitCounter=8;
while (BitCounter > 0)
                {
                Temp_Byte=Byte & 0x01;
                if (Temp_Byte==0x01)
                        { SER=1; }
                else
                        { SER=0; }
                SendSCKClock();
                Byte >>=1;
                BitCounter--;
                }
}
void Prepare7SegmentData(unsigned char DisplayValue){
unsigned char DisplayData;
DisplayData=SevenSegmentTab[DisplayValue];
SendBitBit(DisplayData);
}
void DisplayDrive (unsigned char Temperature, unsigned char Light){
unsigned Temp_Onlar;
unsigned Temp_Birler;
unsigned Light_Onlar;
unsigned Light_Birler;
  Temp_Onlar= Temperature/10;
  Temp_Birler=Temperature-(10*Temp_Onlar);

  Light_Onlar= Light/10;
  Light_Birler=Light-(10*Light_Onlar);

if (Flag_DisplayFlash==1)
            {
Prepare7SegmentData(Blank);
  Prepare7SegmentData(Blank);
  Prepare7SegmentData(Blank);
  Prepare7SegmentData(Blank);
  Prepare7SegmentData(Blank);
  Prepare7SegmentData(Blank);
  Prepare7SegmentData(Blank);
  Prepare7SegmentData(Blank);
  SendRCKClock();
```

```
            }
else
            {
  Prepare7SegmentData(Light_Birler);
  Prepare7SegmentData(Light_Onlar);
  Prepare7SegmentData(Yuzde);
  Prepare7SegmentData(Derece);
  Prepare7SegmentData(C);
  Prepare7SegmentData(Derece);
  Prepare7SegmentData(Temp_Birler);
  Prepare7SegmentData(Temp_Onlar);
  SendRCKClock();

            }
}
```

```
/*============================================================
Dosya Adi: Uart.h
Açiklama:  Uart.c dosyasında kullanilan degiskenlerin belirtildigi,
                        Fonksiyonlarin prototypelerinin yazildigi alan.
============================================================*/
//    SubRoutines.....................................
extern void InitUart(void);
extern void InitUartVariables(void);
extern void SendingChar(unsigned char Karakter);
extern void SendingString(unsigned Long,unsigned char *Pointer);
void PrepareUartBuffer(void);


//    Variables  (Bytes, Bits) .....................................
extern unsigned char xdata ReceivedBytes[10];
extern unsigned char xdata UartTransmitBuffer[32];
extern unsigned char bdata UartFlags;
extern bit FlagSending;
extern bit FlagPacketOK;

//   Constants .....................................................
#define Room 0x31              // '1'
```

```
/*=========================================================
Dosya Adi: Uart.c
Açiklama:  Seri Port kullanma Alt Programlari
=========================================================*/
#include <ADuC841.h>
#include <intrins.h>              // rotate işlemleri için
#include <main.h>
#include <Interrupts.h>
#include <GeneralFunctions.h>
#include <fuzzy.h>
#include <7segment.h>
#include <Uart.h>


//    Subroutines .....................................................
void InitUart(void);
void InitUartVariables(void);
void SendingChar(unsigned char Karakter);
void SendingString(unsigned Long,unsigned char *Pointer);
void PrepareUartBuffer(void);

//    Variables  (Bytes, Bits) .....................................
unsigned char xdata ReceivedBytes[10];
unsigned char xdata UartTransmitBuffer[32];
unsigned char bdata UartFlags;
sbit FlagSending=UartFlags^0;
sbit FlagPacketOK=UartFlags^1;
//_____________________________________________________________________

void Uart (void) interrupt 4 using 1{
if (TI==1)
            {
       TI=0;
FlagSending=0;
if (ByteCounter!=0x00)
                  {
                  ByteCounter--;
                  StringPointer++;
                  SBUF=*StringPointer;
                  FlagSending=1;
                  }
            }
else if (RI==1)
            {
       RI=0;
*ReceivePointer=SBUF;
if ((*ReceivePointer==0x0A)&&((*(ReceivePointer-1)==0x0D)))
                  {
                  if ( (*(ReceivePointer-6)=='<')&&(*(ReceivePointer-
5)=='O')&&(*(ReceivePointer-4)==':')&&(*(ReceivePointer-2)=='>'))
```

```
                        {
                        if ((*(ReceivePointer-3)==Room))
                                {

                                FlagPacketOK=1;
                                ReceivePointer=&ReceivedBytes;
                                }
                        }

                }
        else
                {
         ReceivePointer++;
                FlagPacketOK=0;
                }
           }

}
void InitUart(void){
T3CON=0x86;
T3FD=0X08;
SCON=0x52;
       EA=1;
       ES=1;
}

void InitUartVariables(void){
ByteCounter=0x00;
StringPointer=&ByteCounter;
FlagSending=0;
ReceivePointer=&ReceivedBytes;
}

void SendingChar(unsigned char Karakter){
while (FlagSending==1)
                {;;}
 SBUF=Karakter;
ByteCounter=0x00;
FlagSending=1;

}
void SendingString(unsigned Long, unsigned char *Pointer){
while (FlagSending==1)
                {;;}
ByteCounter=Long;
StringPointer=Pointer;
SBUF=*StringPointer;
FlagSending=1;
}

void PrepareUartBuffer(void)
```

```
{
unsigned char UartChecksum;
unsigned char i;
unsigned char Data_Onlar;
unsigned char Data_Birler;

UartTransmitBuffer[0]='<';
UartTransmitBuffer[1]='O';
UartTransmitBuffer[2]=':';
UartTransmitBuffer[3]=Room;
UartTransmitBuffer[4]=' ';

UartTransmitBuffer[5]='S';
UartTransmitBuffer[6]=':';
Data_Onlar=SetTemperature/10;
Data_Onlar=Data_Onlar+0x30;
Data_Birler=(SetTemperature%10)+0x30;
UartTransmitBuffer[7]=Data_Onlar;
UartTransmitBuffer[8]=Data_Birler;
UartTransmitBuffer[9]=' ';


UartTransmitBuffer[10]='R';
UartTransmitBuffer[11]=':';
Data_Onlar=(RoomTemperature/10)+0x30;
Data_Birler=(RoomTemperature%10)+0x30;
UartTransmitBuffer[12]=Data_Onlar;
UartTransmitBuffer[13]=Data_Birler;
UartTransmitBuffer[14]=' ';

UartTransmitBuffer[15]='L';
UartTransmitBuffer[16]=':';
Data_Onlar=(LightSet/10)+0x30;
Data_Birler=(LightSet%10)+0x30;
UartTransmitBuffer[17]=Data_Onlar;
UartTransmitBuffer[18]=Data_Birler;
UartTransmitBuffer[19]=' ';

UartTransmitBuffer[20]='W';
UartTransmitBuffer[21]=':';
UartTransmitBuffer[22]=' '; // Pencerenin durumu
UartTransmitBuffer[23]=' ';

i=0;
UartChecksum=0;
while (i<24)
            {
UartChecksum=UartChecksum+UartTransmitBuffer[i];
        i++;
            }
UartTransmitBuffer[24]='C';
```

```
UartTransmitBuffer[25]=':';
UartTransmitBuffer[26]=UartChecksum;
UartTransmitBuffer[27]='>';

}
```

EK2 Ana Pano Yazılımı

Tuş Takımı Okuma Altprogramları

```
/*====================================================================
Dosya Adi: key.h
Açiklama:  key.c dosyasında kullanilan degiskenlerin belirtildigi,
           Fonksisyonlarin prototypelerinin yazildigi alan.
====================================================================*
/
//   Constants .....................................................
#define Clear  0;
#define Set              1;
//    SubRoutines.....................................................
extern void InitKeyPins(void);
extern void ReadColumns(void);
extern void ChangeRow(void);
extern void Wait30Ms(void);
extern unsigned char ScanKeys(void);

//    Ports and Port Pins.....................................
sbit Satir_1= P0^0;
sbit Satir_2= P0^1;
sbit Satir_3= P0^2;
sbit Satir_4= P0^3;

sbit Sutun_1= P0^4;     // P2.4 u Column_1 olarak tanimla..
sbit Sutun_2= P0^5;
sbit Sutun_3= P0^6;
sbit Sutun_4= P0^7;
//    Variables  (Bytes, Bits) .....................................
extern unsigned char ActiveRow;
extern unsigned char ActiveColumn;

//Structs  .....................................................
```

```
/*==================================================================
Dosya Adi: key.c
Açiklama:  Matris Tus Okuma Programı
==================================================================*
/
#include <ADuC841.h>
#include <key.h>
#include <intrins.h>              // rotate islemleri için

//    Subroutines .....................................................
void InitKeyPins(void);
void ReadColums(void);
void ChangeRow(void);
void Wait30Ms(void);
unsigned char ScanKeys(void);

//    Variables  (Bytes, Bits) .....................................
unsigned char ActiveRow;
unsigned char ActiveColumn;

//    FUNCTIONS                .....................................

void InitKeyPins(void){
ActiveRow=0x00;
ActiveColumn=0x00;
Sutun_1=1;
Sutun_2=1;
Sutun_3=1;
Sutun_4=1;

Satir_1=0;
Satir_2=0;
Satir_3=0;
Satir_4=0;
}
void  ReadColumns(void){
 if (Sutun_1==0)
          {ActiveColumn=0x10;}
else if(Sutun_2==0)
          {ActiveColumn=0x20;}
else if(Sutun_3==0)
          {ActiveColumn=0x30;}
else if(Sutun_4==0)
          {ActiveColumn=0x40;}
   else
          {ActiveColumn=0x00;}
}
void  ChangeRow(void){
if (ActiveRow==0x01)
      {
```

```
 Satir_1=1;
 Satir_2=0;
 ActiveRow=0x02;
      }
else if (ActiveRow==0x02)
      {
 Satir_2=1;
 Satir_3=0;
 ActiveRow=0x03;
      }
else if (ActiveRow==0x03)
      {
 Satir_3=1;
 Satir_4=0;
 ActiveRow=0x00;
      }
else if (ActiveRow==0x00)
      {
 Satir_4=1;
 Satir_1=0;
 ActiveRow=0x01;
      }
/*else
      {
Satir_1=0;
ActiveRow=0x01;
      }
*/
}
void Wait30Ms(void){
unsigned char Sayac_1;
unsigned char Sayac_2;
unsigned char Sayac_3;
Sayac_1=0x01;
Sayac_2=0xFF;
Sayac_3=0xAA;
while (Sayac_1 > 0)
         {
         while (Sayac_2>0)
                  {
                  while (Sayac_3>0)
                           {
                           Sayac_3--;
                           }
                  Sayac_3=0xAA;
                  Sayac_2--;
                  }
         Sayac_2=0xFF;
         Sayac_1--;
         }
}
```

```
unsigned char  ScanKeys(void){
unsigned char Key;
Key=0x00;
//ActiveColumn=0x00;
 //ActiveRow=0x00;
ChangeRow();// SATIRI DEGISTIR!
ReadColumns();  // KOLONU OKU!
if (ActiveColumn!=0x00)
        {
        Wait30Ms();
        //ReadColumns();
        if (ActiveColumn==0x00)
                {
                Key=0x00;
                }
        else
                {
                Key=ActiveRow+ActiveColumn;
                while (ActiveColumn!=0x00)
                        {
                        Wait30Ms();
                        ReadColumns();
                        }
                ActiveColumn=0x00;
                ActiveRow=0x00;
                }
        }
return Key;
}
```

LCD Sürme Alprogramı

```
/*=====================================================================
Dosya Adi: LCD.h
Açiklama:  LCD.c dosyasında kullanilan degiskenlerin belirtildigi,
           Fonksiyonlarin prototypelerinin yazildigi alan.
=====================================================================*
/
//   Constants .....................................................
#define LCD_Type              16              // 16--> 2x16 LCD
                                                  // 20--> 4x20 LCD
#define ResetCommand     0x3F       // LCD Reset Komutu
#define DisplaySetting          0x32   // DUAL LİNE AND 5X10 DOTS
#define CommSetting             0x2F   // 4-BIT ÜZERİNDEN GÖNDERME AYARI.
#define LCDDDRamAdresSet 0x80 // Ram adresi.

#define Deneme         "112233445566778899AA"
#define Ekran_1 "     Deneme         "
#define Ekran_2 "HOSGELDINIZ.....!!!!"
#define Ekran_3 " LEVENT  BIRGUL     "
```

```
#define Ekran_4        " YILDIZ  TEKNIK     "
#define Ekran_5        "  UNIVERSITESI      "
#define Ekran_6 "0123456789ABCDEF    "

//    SubRoutines.....................................
extern void LCDVariblesInit(void);
extern void LCDPortInitialize(void);
extern void Wait_50usn (void);
extern void Wait_5msn(void);
extern void SendByte(unsigned char Byte);
extern void SendCommand(unsigned char Command);
extern void SendData(unsigned char Data);
extern void PrepareLCDAdress(unsigned char Line);
extern void LCDInitialize (void);
extern void PrepareBuffer(unsigned char  *DisplayPointer);
extern void Send_Data2LCD (unsigned char Line,unsigned char *DataPointer);

//    Ports and Port Pins.....................................
sbit LCD_D7 = P2^7;             // P2.7 yi LCD_D7 olarak tanimla..
sbit LCD_D6 = P2^6;
sbit LCD_D5 = P2^5;
sbit LCD_D4 = P2^4;
sbit LCD_EN = P2^3;
sbit LCD_RS = P2^2;
sbit BUTTON = P3^2;
sbit BUTTON_1 = P3^3;

//    Variables  (Bytes, Bits) .....................................
extern unsigned char  LCD_50USN_SAYACI;
extern unsigned char  LCD_5MSN_SAYACI;
extern unsigned char  LCD_Buffer[LCD_Type];
```

```
/*==================================================================
Dosya Adi: LCD.c
Açiklama:  LCD sürme alt  programlari
==================================================================*
/
#include <ADuC841.h>
#include <LCD.h>
//#include <intrins.h>            // rotate işlemleri için

//    Subroutines .....................................................
void LCDVariblesInit(void);
void LCDPortInitialize(void);
void Wait_50usn (void);
void Wait_5msn(void);
void SendByte(unsigned char Byte);
void SendCommand(unsigned char Command);
void SendData(unsigned char Data);
void PrepareLCDAdress(unsigned char Line);
void LCDInitialize (void);
void PrepareBuffer(unsigned char  *DisplayPointer);
void Send_Data2LCD (unsigned char Line,unsigned char *DataPointer);


//    Variables  (Bytes, Bits) .....................................
unsigned char LCD_50USN_SAYACI;
unsigned char   LCD_5MSN_SAYACI;
unsigned char LCD_Buffer[LCD_Type];




void LCDVariblesInit(void){
LCD_50USN_SAYACI=0x00;
LCD_5MSN_SAYACI=0x00;
LCD_Buffer[0]=0x00;
LCD_Buffer[1]=0x00;
LCD_Buffer[2]=0x00;
LCD_Buffer[3]=0x00;
LCD_Buffer[4]=0x00;
LCD_Buffer[5]=0x00;
LCD_Buffer[6]=0x00;
LCD_Buffer[7]=0x00;
LCD_Buffer[8]=0x00;
LCD_Buffer[9]=0x00;
LCD_Buffer[10]=0x00;
LCD_Buffer[11]=0x00;
LCD_Buffer[12]=0x00;
LCD_Buffer[13]=0x00;
LCD_Buffer[14]=0x00;
LCD_Buffer[15]=0x00;
```

```
}
void LCDPortInitialize(void){
LCD_D7=0;
LCD_D6=0;
LCD_D5=0;
LCD_D4=0;
LCD_EN=0;
LCD_RS=0;
}
void Wait_50usn(void){
LCD_50USN_SAYACI=0xFF;                  // 50usn için gerekli olan sayım degeri.

while (LCD_50USN_SAYACI > 0)
        {
        LCD_50USN_SAYACI--;
        //      _nop_;
        }
}
void Wait_5msn(void){
LCD_5MSN_SAYACI=0xFF;                   // 5msn için gerekli olan sayım degeri.
while (LCD_5MSN_SAYACI > 0)
  {
LCD_50USN_SAYACI=0xFF;
while (LCD_50USN_SAYACI > 0)
        {
        LCD_50USN_SAYACI--;
        //      _nop_;
        }
LCD_5MSN_SAYACI--;
  }
}
void SendByte(unsigned char Byte){
unsigned char Temp_Byte;
//-------------------
Temp_Byte=Byte;
Temp_Byte&=0x80;
if (Temp_Byte==0x80)
        {LCD_D7=1;}
else
        {LCD_D7=0;}

Temp_Byte=Byte;
Temp_Byte&=0x40;
if (Temp_Byte==0x40)
        {LCD_D6=1;}
else
        {LCD_D6=0;}

Temp_Byte=Byte;
Temp_Byte&=0x20;
if (Temp_Byte==0x20)
```

```
        {LCD_D5=1;}
else
        {LCD_D5=0;}

Temp_Byte=Byte;
Temp_Byte&=0x10;
if (Temp_Byte==0x10)
        {LCD_D4=1;}
else
        {LCD_D4=0;}
//-------------------
LCD_EN=1;
Wait_50usn();
LCD_EN=0;
//-------------------
Temp_Byte=Byte;
Temp_Byte&=0x08;
if (Temp_Byte==0x08)
        {LCD_D7=1;}
else
        {LCD_D7=0;}

Temp_Byte=Byte;
Temp_Byte&=0x04;
if (Temp_Byte==0x04)
        {LCD_D6=1;}
else
        {LCD_D6=0;}

Temp_Byte=Byte;
Temp_Byte&=0x02;
if (Temp_Byte==0x02)
        {LCD_D5=1;}
else
        {LCD_D5=0;}

Temp_Byte=Byte;
Temp_Byte&=0x01;
if (Temp_Byte==0x01)
        {LCD_D4=1;}
else
        {LCD_D4=0;}
//-------------------
LCD_EN=1;
Wait_50usn();
LCD_EN=0;
//-------------------
Wait_50usn();
}
void SendCommand(unsigned char Command){
LCD_RS=0;
```

```
SendByte(Command);
}
void SendData(unsigned char Data){
LCD_RS=1;
SendByte(Data);
}
void PrepareLCDAdress(unsigned char Line){
if (Line==1)
        {Line=0x00;}
else if (Line==3)
        {Line=0x14;}
else if (Line==2)
        {Line=0x40;}
else if (Line==4)
        {Line=0x54;}
Line=Line+LCDDDRamAdresSet;
SendCommand(Line);
}
void LCDInitialize (void){
unsigned char i;
i=0;
while (i<6)
  {
Wait_5msn(); // 100msn olamasi gerekir.
i++;
  }

LCD_EN=0;
LCD_RS=0;

i=0;
while (i<4)
  {
Wait_5msn();
i++;
  }

i=0;
while  (i<4)
  {
SendCommand(ResetCommand);
Wait_5msn();
i++;
  }

SendCommand(DisplaySetting);
Wait_5msn();

SendCommand(0x0C);
Wait_5msn();
}
```

```
void PrepareBuffer(unsigned char  *DisplayPointer){
unsigned char  ByteCounter;
ByteCounter=0;
while (ByteCounter <= LCD_Type) // ekranda 20 karakter gösterilir.
        {
        LCD_Buffer[ByteCounter]=*DisplayPointer;
        DisplayPointer++;
        ByteCounter++;
        }
}
void Send_Data2LCD (unsigned char Line,unsigned char *DataPointer){
unsigned char  DataCounter;
unsigned char  Data;
  DataCounter=0;
  PrepareBuffer(DataPointer);
  PrepareLCDAdress(Line);

  while (DataCounter <= LCD_Type)
  {
  Data=LCD_Buffer[DataCounter];
  SendData(Data);
  DataCounter++;
  }

}
```

ÖZGEÇMİŞ

Doğum tarihi	29.03.1983	
Doğum yeri	İstanbul	
Lise	1998-2001	Adnan Menderes Anadolu Lisesi
Lisans	2001-2005	Yıldız Üniversitesi Mühendislik Fak. Elektrik Mühendisliği Bölümü
Yüksek Lisans	2005-2007	Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Elektrik Müh. Anabilim Dalı, Kontrol Otomasyon Programı

Çalıştığı kurum

2006 -Devam ediyor	Phoenix Contact Elektronik Tic. Ltd. Şti.