T.C.

TRAKYA ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

PIC MİKRODENETLEYİCİ İLE ÇALIŞAN OTOMATİK BAHÇE SULAMA SİSTEMİ

RECEP ÇALIŞIR

YÜKSEK LİSANS TEZİ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

Tez Danışmanı: Dr. Öğretim Üyesi Özlem UÇAR

EDİRNE- 2021

RECEP ÇALIŞIR'In hazırladığı "**PIC MİKRODENETLEYİCİ İLE ÇALIŞAN OTOMATİK BAHÇE SULAMA SİSTEMİ**" başlıklı bu tez, tarafımızca okunmuş, kapsam ve niteliği açısından **Bilgisayar Mühendisliği** Anabilim Dalında bir **Yüksek lisans tezi** olarak kabul edilmiştir.

Jüri Üyeleri (Ünvan, Ad, Soyad):	İmza
Dr. Öğr. Üyesi Özlem UÇAR	……
Doç. Dr. İlhan UMUT	……
Dr. Öğr. Üyesi Erkan ÖZHAN	……
…….	……
…….	……

Tez Savunma Tarihi: 30/06/2021

Bu tezin Yüksek Lisans/Doktora tezi olarak gerekli şartları sağladığını onaylarım.

İmza

Dr. Öğr. Üyesi Özlem UÇAR
(Ünvanı, Ad, Soyad)
Tez Danışmanı ……

Trakya Üniversitesi Fen Bilimleri Enstitüsü onayı

Prof. Dr. Hüseyin Rıza Ferhat KARABULUT
(Ünvan, Ad, Soyad)
Fen Bilimleri Enstitüsü Müdürü

T.Ü.FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
YÜKSEK LİSANS PROGRAMI
DOĞRULUK BEYANI

Trakya Üniversitesi Fen Bilimleri Enstitüsü, tez yazım kurallarına uygun olarak hazırladığım bu tez çalışmasında, tüm verilerin bilimsel ve akademik kurallar çerçevesinde elde edildiğini, kullanılan verilerde tahrifat yapılmadığını, tezin akademik ve etik kurallara uygun olarak yazıldığını, kullanılan tüm literatür bilgilerinin bilimsel normlara uygun bir şekilde kaynak gösterilerek ilgili tezde yer aldığını ve bu tezin tamamı ya da herhangi bir bölümünün daha önceden Trakya Üniversitesi ya da farklı bir üniversitede tez çalışması olarak sunulmadığını beyan ederim.

30/06/2021

Ad, Soyad

Recep ÇALIŞIR

İmza

Yüksek Lisans Tezi
PIC Mikrodenetleyici ile Çalışan Otomatik Bahçe Sulama Sistemi
T.Ü. Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

ÖZET

Teknolojinin gelişmesiyle birlikte modern tarım ve günlük hayattaki etkileri de azımsanamayacak ölçüde artmıştır. Otomasyon ile kullanılan sistemler gün geçtikçe artmakla birlikte, kullanıcılar açısından zaman, maliyet ve rahatlık konularında pozitif etki ederek vazgeçilmez duruma gelmiştir. Sadece belirli bir amaca yönelik oluşturulmuş sabit sistemler, kullanıcı ihtiyaçları doğrultusunda değişim gösteremediklerinden; hem kullanıcı bakımından zamanla işlevsiz duruma gelmekte, hem de maddi bakımdan bir yük durumuna düşmektedirler.

Bu tez çalışmasında, bir mikrodenetleyici türü olan PIC ile tasarlanan ve kullanıcı isteklerine göre çalışma şekli değişebilen otomatik bahçe sulama sistemi geliştirilmesi amaçlanmıştır. Buna istinaden, düşük güç tüketimine sahip, sisteme entegre edilecek motorun çalışma sistemi ile uyumlu olan ve tüm ek bileşenleri barındıran bir mikrodenetleyici üzerinde araştırma yapılmıştır. Yüksek performans elde etme amacıyla uygun derleyici ve programlayıcı seçimleri üzerine odaklanılmıştır. İstenen amaca yönelik bir sistem tasarımının gerçekleştirilmesi için algoritmalar geliştirilmiştir. Hatasız çalışan bir sistem elde edilmesi amacıyla, derleyiciden elde edilen çıktı, simülasyon ortamında entegrasyon testlerine konulmuştur.

Bu kapsamda, çalışmanın ikinci bölümünde; mikrodenetleyici yapısı ve kullanım nedenleri hakkında bilgi verilmiştir. Çalışmanın üçüncü bölümünde; sistem tasarlanırken kullanılmış olan geliştirme araçlarının özellikleri ve kullanım şekillerinden bahsedilmiştir. Çalışmanın dördüncü bölümünde; sistemin araştırma konuları ve geliştirme yöntemi hakkında detaylı bilgi verilerek PICOBS (PIC Otomatik Bahçe

Sulama Sistemi) özellikleri açıklanmıştır. Çalışmanın son bölümü olan beşinci bölümde ise, çalışmanın sonuçlarından ve üzerine geliştirme yapılabilecek özelliklerden bahsedilmiştir.

Yıl : 2021
Sayfa Sayısı : 80
Anahtar Kelimeler : Otomatik Sulama, Zaman Ayarlı Sistem, Mikrodenetleyici

Master Thesis
Automatic Garden Irrigation System Working with PIC Microcontroller
Trakya University Institute of Naturel Sciences
Computer Engineering Department

ABSTRACT

With the development of technology, its effects on modern agriculture and daily life have increased considerably. Although the systems used with automation are increasing day by day, they have become indispensable by having a positive effect on time, cost and comfort for users. Since fixed systems created only for a specific purpose cannot change in line with user needs; they both become dysfunctional for the user over time and also become a financial burden.

In this thesis, it is aimed to develop an automatic garden irrigation system, which is designed with PIC, a type of microcontroller, and can be changed according to user requests. Based on this, research has been carried out on a microcontroller with low power consumption, compatible with the operating system of the motor to be integrated into the system and including all additional components. The focus is on choosing the appropriate compiler and programmer to achieve high performance. Algorithms have been developed to realize a system design for the desired purpose. In order to obtain an error-free system, the output obtained from the compiler was put into integration tests in the simulation environment.

In this context, in the second part of the study; information about microcontroller structure and usage reasons are given. In the third part of the study; the features and usage patterns of the development tools used while designing the system are mentioned. In the fourth part of the study; PICOBS (PIC Automatic Garden Irrigation System) features are explained by giving detailed information about the research subjects and development method of the system. In the fifth chapter, which is the last part of the study, the results of the study and the features that can be improved upon are mentioned.

Year : 2021
Number of Pages : 80
Keywords : Automatic Irrigation, Timed Sysem, Microcontroller

TEŞEKKÜR

Çalışmalarım boyunca değerli yardım ve katkılarıyla beni yönlendiren danışmanım Dr. Öğr. Üyesi Özlem UÇAR'a teşekkürlerimi sunarım.

Tezimin her aşamasında beni yalnız bırakmayan aileme ve Hanım ÇALIŞIR'a sonsuz sevgi ve saygılarımı sunarım.

Recep ÇALIŞIR

Edirne, 2021

İÇİNDEKİLER

ÖZET iv
ABSTRACT vi
TEŞEKKÜR viii
İÇİNDEKİLER ix
ÇİZELGELER DİZİNİ xi
ŞEKİLLER DİZİNİ xii
SİMGELER VE KISALTMALAR DİZİNİ xiii
BÖLÜM 1 1
GİRİŞ 1
BÖLÜM 2 3
KURAMSAL BİLGİLER 3
2.1. Mikrodenetleyici Hakkında 3
2.2. Mikrodenetleyici Kullanım Nedenleri 5
2.3. Mikrodenetleyici Mimarisi 6
2.4. PIC Mikrodenetleyici 8
2.4.1. PIC 16F877A Genel Özellikleri 9
2.4.1.1. PIC 16F877A Besleme Gerilimi 10
2.4.1.2. Osilatör Ayarları ve Kondansatör Seçimi 11
2.4.1.3. PIC 16F877A Sıfırlama Ayarları 13
2.4.1.4. Kesmeler 13
2.4.1.5. Analog Sayısal Dönüştürücü 14
2.4.1.6. Zamanlayıcı Modülleri 15
2.4.1.7. PIC Bellek Yapısı 16
BÖLÜM 3 19
MATERYAL VE YÖNTEM 19
3.1. CCS- C Derleyici Programı 19
3.2. C Programlama Dili 20
3.3. Proteus Design Suite Simulasyon Programı 22
3.4. PICKIT Programlayıcısı 24
BÖLÜM 4 27
ARAŞTIRMA BULGULARI 27
4.1. Araştırma Kapsamındaki Alanlar 27

4.2. Sistem İçerisinde Geliştirilen Özellikler 28
4.3. PICOBS Özellikleri 31
4.4. Proteus Simülasyon Ortamı İçinde PICOBS 32
4.4.1. Elektronik Bileşenlerin Seçimi 32
4.4.2. Devre Elemanlarının Birleşimi 33
4.4.3. Simülasyon Devre Çalışma Mantığı 34
4.4.4. Kodun Simülasyon Ortamına Entegresi 34
4.4.5. Ekran Tasarımları, Geçişleri ve Kullanıcı Veri Girişi 35
4.4.5.1. Alan Büyüklüğü Belirleme Ekranı 35
4.4.5.2. Tekrar Sayısı Belirleme Ekranı 37
4.4.5.3. Toplam Çalışma Süresi Belirleme Ekranı 37
4.4.5.4. Mevcut Saat Belirleme Ekranı 38
4.4.5.5. Çalışma Saati Belirleme Ekranı 38
4.4.5.6. Ana Ekran 39
4.5. PICOBS Çalışma Mantığı 40
BÖLÜM 5 41
SONUÇLAR VE YAPILACAK ÇALIŞMALAR 41
5.1. Sonuçlar 41
5.2. Yapılacak Çalışmalar 41
KAYNAKLAR 43
EKLER 46
EK A: PICOBS PROGRAM FONKSİYONLARI 46
EK B: KAYNAK KODLARI 48
ÖZGEÇMİŞ 66

ÇİZELGELER DİZİNİ

Çizelge 2.1. Osilatör için uygulanabilir kondansatör miktarları……..11
Çizelge 2.2. Osilatör için kullanılabilir kondansatör miktarları...…….....12

ŞEKİLLER DİZİNİ

Şekil 2.1. Flip- Flop devresi 3
Şekil 2.2.Von Neumann mimarisi 6
Şekil 2.3. Harvard mimari yapısı 7
Şekil 2.4. Popüler PIC mikrodenetleyiciler 8
Şekil 2.5. PIC 16F877A pin görünüş şeması 9
Şekil 2.6. PIC 16F877A besleme bağlantıları 10
Şekil 2.7. Osilatör bağlantı şeması 12
Şekil 2.8. Kesme kullanımının etkisi 14
Şekil 2.9. Analog dijital dönüştürücü kullanımı 15
Şekil 2.10. Zamanlayıcı blok diyagramı 16
Şekil 2.11. Çipteki program belleği 17
Şekil 3.1. CCS- C derleyici 20
Şekil 3.2. CC-C derleyici içinde C Programlama kullanımı 21
Şekil 3.3. Proteus Professional simulasyon geliştirme ortamı 23
Şekil 3.4. PICKIT bağlantı gösterimi 24
Şekil 3.5. PICKIT 3 pinleri 25
Şekil 3.6. PICKIT 3 bileşenleri 26
Şekil 4.1. Alan büyüklüğü kod parçacığı 27
Şekil 4.2. Kontrol devresi blok gösterimi 28
Şekil 4.3. Çalışma başlangıcı belirleme algoritması 29
Şekil 4.4. Sistemi oluşturan fonksiyonların gösterimi 30
Şekil 4.5. Kullanılan kütüphaneler 31
Şekil 4.6. Elektronik devre elemanları 32
Şekil 4.7. Simülasyon devresi 33
Şekil 4.8. Program entegresi 35
Şekil 4.9. Alan bilgisi giriş ekranı 36
Şekil 4.10. Alan ekran geçişi 36
Şekil 4.11. Tekrar sayısı ekranı 37
Şekil 4.12. Gün aralığı ekranı 38
Şekil 4.13. Mevcut saat ekranı 38
Şekil 4.14. Başlangıç saati ekranı 39
Şekil 4.15. Kurulum tamamlandı ekranı 39
Şekil 4.16. Ana ekran 39

SİMGELER VE KISALTMALAR DİZİNİ

Kısaltmalar

ADC	Analog to Digital Converter
ALU	Arithmetic Logical Unit
BOR	Brown Out Reset
CAN	Controller Area Network
CPU	Central Processing Unit
DAC	Digital to Analog Converter
DSP	Digital Signal Processor
EEPROM	Electrically Erasable Programmeble Read Only Memory
ICSP	In Circuit Serial Programming
IDE	Integrated Development Environment
ISR	Interrupt Service Routine
LCD	Liquid Crystal Display
LED	Light Emitting Diode
MCU	Micro Controller Unit
OST	Oscillatör Start- Up Timer
PCB	Printed Circuit Board
PIC	Peripheral Interface Controller
PICOBS	PIC Otomatik Bahçe Sulama Sistemi
POR	Power On Reset
PVSM	Proteus Virtual System Modeling
PWM	Pulse Width Modulation

PWRT	Power Up Timer
RAM	Random Access Memory
RISC	Reduced Instruction Set Computing
ROM	Read Only Memory
SPI	Serial Peripheral Interface
UART	Universal Asynchronous Receiver/ Transmitter
USB	Universal Serial Bus

BÖLÜM 1

GİRİŞ

Teknoloji dünyasında, gelişim ivmesi en yüksek alanlardan biri elektroniktir. Elektronik alanında yeni sistem ve ürünlerin geliştirilmesi artarak devam etmektedir. Gerçekleşen bu yeniliklerin takibi gün geçtikçe daha zorlu bir iş haline gelmektedir. Bu noktada mikrodenetleyiciler, karmaşık ve zorlu yapının üstesinden gelinmesi için çok önemli bir rol oynamaktadır. Mikrodenetleyiciler hem donanım hem de yazılım olarak bu ihtiyacı rahatlıkla karşılayabilmektedir (Özkan, 2018).

Geçmişten günümüze canlı yaşamında ve sanayide kullanılan su vazgeçilmez bir kaynaktır. Bitkisel ürün gelişimi için hayati bir öneme sahiptir. Suyun eksikliği bitki üretimini kısıtlayan çok önemli bir etmendir (Tekinel, Kanber & Çetin, 2000).

Yang, Lu, Okushima ve Sase (2004), gerçekleştirdikleri çalışmada akustik sensör kullanarak bitki gövdesindeki sıvı akışını ölçme sayesinde optimum ve otomatik sulama sistemi tasarlamışlardır. Bu sistemde kullanılan sensör gürültüden etkilenmektedir.

Üretilmesi planlanan bitkilerin sulama ihtiyacının gerçekçi bir şekilde belirlenebilmesi için söz konusu bitkilerin belirli dönemlerdeki su ihtiyaçlarının belirlenmesi gerekmektedir. Bu belirleme sonucu sulama yapılacak alandaki su tüketimi hesaplanabilmektedir (Koç ve Güner, 2005).

Kırnak (2006), topraktaki nem miktarına göre otomatik sulama yapan bir sistem geliştirmiştir. Bu sistem sayesinde aşırı sulamanın ve gereksiz su tüketiminin önüne geçmiştir.

Su tüketiminde modern yöntemlerin kullanılmaması, aşırı tüketim ve savurganlık, yanlış kullanım yöntemleri, küresel ısınma ve iklim değişiklikleri gibi etkenler ülkemiz de dahil olmak üzere su sorunlarının ortaya çıkmasına sebep olmaktadır. Yaşanabilecek su krizlerinin önüne geçebilmek için dünya genelinde

önlem almak gerekmektedir. Kalıcı önlemler alınmadığı takdirde yaşanabilecek krizin boyutları endişe verici olmakla birlikte suya olan talebin giderek artacağı görülmektedir (Alparslan, Tanık & Dölgen, 2008).

Fidan ve Karasekreter (2011), kısa mesaj ile kontrol edilebilen sulama otomasyon kontrol birimini geliştirmişlerdir. Bu çalışmaları ile kullanıcı girdisine göre sistemi çalıştırabilmektedir ve yağmur yağdığı takdirde kullanıcı bilgilendirme özelliği de bulunmaktadır. Bu sistem sayesinde su ve enerjiden tasarruf sağlamayı başarmışlardır.

Bakibillah, Rahman ve Zaman (2014), motorun hız kontrolünün ihtiyaca göre ayarlanabildiği bir sistem geliştirmişlerdir. Bu sistem sayesinde gereksiz donanım kullanımının önüne geçmiş ve enerjiden tasarruf sağlanmıştır.

Hermansson ve Lundblad (2019), geliştirdikleri bitkiler için otomatik sulama sistemi sayesinde sensör yardımı ile azalan bitki suyunu otomatik olarak vermek üzerine odaklanmışlardır. Bu tür bir sistemin geniş bir alana uygulanması ve esnekliği uygulanabilirlik açısından zorluk çıkarabilmektedir.

Yapılan bu çalışmada önceki çalışmalardan farklı olarak, belirlenen zamanda başlayan çalışma süresini sulanacak alan büyüklüğüne göre otomatik hesaplayabilmesi ve çalışma frekansının değişken ve tekrar eder şekilde ayarlanabilir olmasıdır. Bu çalışmada, PIC (Peripheral Interface Controller) mikrodenetleyici ile çalışan, çoklu görev yürütmesi yaparak gerçek zamanlı çalışabilen, zaman ayarlaması imkânı veren, belirlenen süre boyunca otomatik ayarlanan hızda çalışarak kaynakları mümkün olduğu kadar düşük seviyede kullanan bir sistem geliştirilmesi amaçlanmıştır. Bu sistemin bahsedilen şekilde geliştirilmesi sayesinde kullanıcı isteklerine direkt olarak yanıt verebilecek bir ürün ortaya konulmuştur. Kullanışlı bir ürün olarak tasarlanan bu sistem ile mevcut uygulamalardaki verimliliğin artması sağlanacaktır. Böylece sabit bir biçimde, ihtiyaç olmadığı halde bile çalışmaya devam eden sistemlerin yerini alabilecek bir alternatif üretilmiş olacaktır.

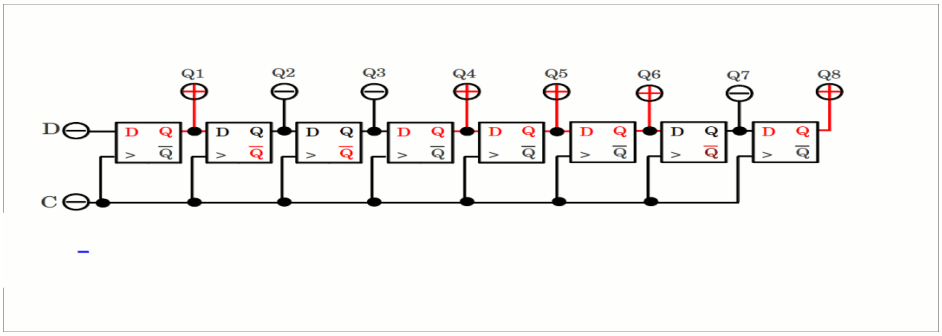
BÖLÜM 2

KURAMSAL BİLGİLER

2.1. Mikrodenetleyici Hakkında

Günümüzde elektronik cihazlara olan talepteki artma ile bu cihazlar çok daha fazla işlevi gerçekleştirebilir hale gelmiştir. Örneğin 90'lı yılların sonunda, hatta 2000'li yılların başlarında sadece sesli ya da yazılı iletişim ile ihtiyacımızı karşılarken, günümüzde yurtdışındaki insanlarla dahi görüntülü konuşmak mümkündür (Karaca, 2014). Tüm bu gelişmeler sonucunda üreticiler için birçok işlevi gerçekleştirebilen ve az yer kaplayan cihazların geliştirilmesi kaçınılmaz bir hal almıştır. Bu kapsamda, mikroişlemci üretilmiş fakat giriş/ çıkış birimleri ve hafızasının farklı yerlerde olması sonucu az yer kaplayan cihazlar üretmek için uygun olmadığının farkına varılmıştır. Farklı yerlerde olan bu birimlerin birleştirilmesi sayesinde aynı işlevi yerine getiren ve daha az yer kaplayan mikrodenetleyiciler ortaya çıkmıştır.

Transistörler birleşerek bir bitlik veriyi tutmaya yarayan "Flip- Flop" yapılarını oluşturmaktadır ve bu yapılar, basit bir şekilde ifade etmek gerekirse, bir sonraki bit gelene kadar çıkıştaki biti saklayan ve girişte veri değişikliği olduğunda "clock" darbesiyle birlikte çıkıştaki değeri de değiştiren mantıksal devre olarak tanımlanabilir (Balım, 2015). Bura yapılan işin tam karşılığı bir bitin saklanması şeklinde açıklanabilir (Şekil 2.1).



Şekil 2.1. Flip- Flop devresi.

Daha çok verinin depolanabilmesini mümkün kılan saklayıcıların oluşturulabilmesi ise bu yapıların birleşmesi sonucu gerçekleşmektedir. Saklayıcının kaç bitlik bir yapıya sahip olacağı, mikrodenetleyicinin yapısına göre belirlenmektedir. İşlemci hafıza birimleri ise saklayıcıların bir araya gelmesi sonucu oluşmaktadır.

Hafıza birimlerinde yer alan verilerin okunması ve işlenmesi bu birimler dışında da gerçekleşmektedir. Sistemde yer almakta olan bütün bileşenlerin saklayıcılardan oluşması ile gerçekleşmekte olan bütün işlemler saklayıcı üzerinde tutulmakta olan verilerin mantıksal devreler üzerinden kendi aralarında ya da türlü işlemlere maruz kalması şeklinde gerçekleşmektedir.

Matematiksel hesaplamalar, ALU (Arithmetic Logical Unit) tarafından gerçekleştirilmektedir. Dışarıdan gelen verinin alınması işlemi giriş birimleri tarafından, mikrodenetleyici tarafından üretilen verinin tutulma işlemi ise çıkış birimleri tarafından sağlanmaktadır.

Sistem mimarisi kavramı, mikrodenetleyiciye sahip sistemleri kapsayan saklayıcılar kümesi ve bu kümeler arasındaki bilgi transferi şeklinde değerlendirilmektedir. Mikrodenetleyici mimarisini belirleyen faktörler ise kullanılan komutların çeşitleri ve çalışma süreleri, saklayıcı tipleri ve transfer yöntemleri şeklinde tanımlanmaktadır.

Sık yapılan hatalardan biri ise mikrodenetleyici ve mikroişlemci kavramlarının birbiri ile karıştırılmasıdır. Mikronedetleyiciyi oluşturan çevre birimleri; dijital- analog çevirici, hafıza ve zamanlayıcı gibi birimlerken, mikroişlemci ise sadece hafıza ve işlem gibi birimlerden meydana gelmektedir. Mikroişlemcilerin kullanım alanının büyük çoğunluğu kişisel bilgisayarlardır.

Mikrodenetleyiciler, daha çok endüstriyel alanda gerçekleştirilen otomasyon işlemlerine yönelik tasarlanan kapsamlı mikroişlemciler şeklinde tanımlanmaktadır. Mikroişlemcilerin çalışma hızı daha fazla olmasına rağmen mikrodenetleyicilerin üzerinde daha fazla birim yer almaktadır.

Bir gömülü sistem tasarlamamız için mikrodenetleyiciler, mikroişlemcilere göre işimizin çok daha kolay olmasını sağlamaktadır çünkü gömülü sistem için mikroişlemci kullanmamız durumunda burada bahsettiğimiz RAM (Random Access Memory), program hafızası, ADC (Analog to Digital Converter) gibi tüm birimleri satın alıp bunlar

ve mikroişlemci arasında veri yolu, adres yolu bağlantılarını kurmamız gerekmektedir (Balım, 2018). Mikrodenetleyiciden, söz konusu birimlerin tümünü içinde barındıran bir yapı olarak bahsedilebilmektedir.

2.2. Mikrodenetleyici Kullanım Nedenleri

Mikrodenetleyici, gelen veriyi ve programı hafızasına alarak derleyip, sonuç olarak çıktı üreten bir bilgisayardır. Programlandığı yazılım hafızaya kaydedilerek, işlenebilecek şekilde derlenerek bir çıktı üretilmektedir. Çıkıştaki porta bağlı bir motor olduğu düşünülürse, bu motor mikrodenetleyiciden gelen çıkış sinyaline göre çalışacaktır. Yani, mikrodenetleyiciler elektronik devrelerde beyin görevi görerek, elektronik sistemlerin kontrol edilmesini sağlamaktadır (Semiz, 2018).

Zamanla değişen kullanıcı ihtiyaçlarının karşılanması amacıyla, gerektiğinde ihtiyaca göre çalışabilecek şekilde tasarlanan sistemler ön plana çıkmaktadır. Kullanıcının kendi ihtiyaçları doğrultusunda çalışma prensibini ayarlayabildiği sistemler kullanışlı ve uzun ömürlü olmaktadır. Kullanıcı ihtiyacı doğrultusunda geliştirilen sistemlerin ek parçaya ihtiyaç duymadan çalışan donanımlara entegre edilmesi ile ortaya yüksek performanslı, düşük maliyetli ve kullanışlı sistemler çıkmaktadır. Mikroişlemci ile oluşturulan sistemlerde işlemleri gerçekleştirmek için ek ünitelere ihtiyaç vardır. Bu ünitelerin kendi aralarında veri alışverişi yapmasını mümkün kılmak için bir veri yoluna ihtiyaç vardır. Bütün bu etkenler maliyetin daha çok artmasını sağlamakla birlikte devredeki boş yer kapasitesinin azalmasına neden olmaktadır. Mikrodenetleyicide ise bu durum, gereken tüm birimlerin yongada barındırılması şeklinde çözülmektedir (Özkan, 2008). Dolayısıyla tasarlanacak sistemlerde mikrodenetleyici kullanmak hem maliyet hem de programlamada bir katkı sağlamaktadır.

Mikrodenetleyicilerin kontrol edilebilmesi ve istenilen amaç doğrultusunda çalışabilmesi için komut satırlarından meydana gelen bir yazılım gerekmektedir. Birçok farklı mikrodenetleyici türü olmakla birlikte, her bir mikrodenetleyici türü için değişik derleyici türleri bulunmaktadır.

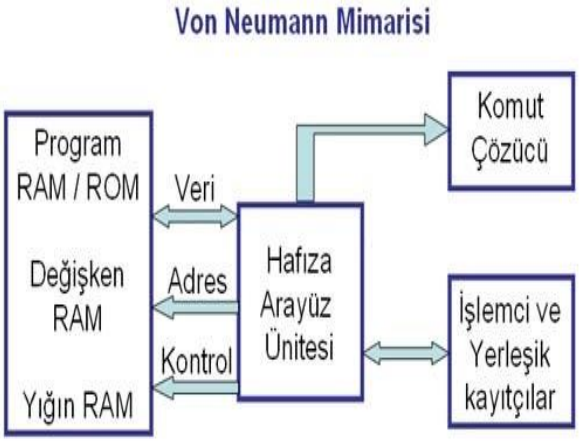
Mikrodenetleyici kullanımının birçok sebebi bulunmaktadır. Bu sebeplerden başlıca olanları; küçük boyutlu olmaları, tek mikrodenetleyici ile birçok çözüm

üretebilme imkânı, güç tüketiminin düşük olması, program depolayabilme ve istenilen zamanda çalıştırabilme ile düşük maliyetli ve yüksek performanslı olmalarıdır. Bahsedilen özellikler sayesinde belirli bir amaca yönelik kullanışlı bir sistem oluşturmak mümkündür. Çalışma şeklinin kullanıcı girdisine göre ayarlanabilmesi özelliği ile esnek bir sistem tasarlanmıştır. İstenen görevlerin belirli aralıklar ile otomatik bir şekilde gerçekleştirilmesiyle zaman tasarrufu sağlanması amaçlanmıştır. Bu sistemin kontrolünde kullanılacak mikrodenetleyici türü, günümüzde kolaylıkla elde edilebilen ve en çok tercih edilen türlerden biri olan PIC mikrodenetleyici türü olacaktır.

2.3. Mikrodenetleyici Mimarisi

Mimari sınıflandırması söz konusu olduğunda komut işleme tekniği ve hafıza organizasyonu gibi etkenler dikkate alınmaktadır.

Mikrodenetleyiciler, hafıza organizasyonuna göre Harvard ve Von Neuman şeklinde iki mimarı sınıflandırmasına sahiptir. Geçmişte Von Neuman mimarisi tercih edilse de 1970'li yılların sonlarında Harvard mimarisi mikrodenetleyici tasarımında standart hale gelmiştir ve günümüzde bu iki mimari yapının özelliklerini de içeren mikrodenetleyiciler (MAXQ ailesi) de bulunmaktadır (İlker, 2013).



Şekil 1.2. Von Neumann mimarisi.

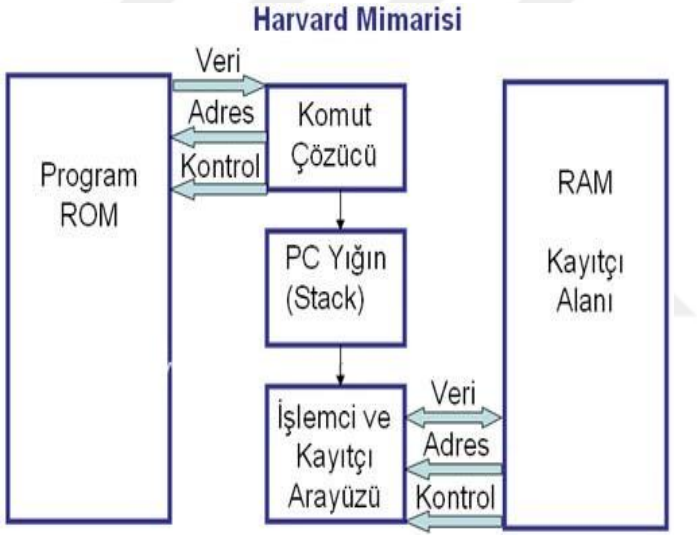
Bu mimari yapısı; merkezi işlem birimi, giriş ve çıkış birimlerinden oluşmaktadır. Birimlerin kendi aralarında gerçekleşmekte olan haberleşme ise iletişim kanalları sayesinde gerçekleşmektedir.

Von Neuman Mimari yapısında bütün halde bir parçalı bellek bulunmaktadır.

Bu demektir ki eş bir hafıza alanında yer alan verinin yanında program alanı da bulunmaktadır.

Bu mimaride hafıza ve işlem birimleri arasında bir ayrıma gidilmiş olması çok önemli bir özellik olarak nitelendirilmektedir. Veri iletimi ve komut alma gibi operasyonlar ortak hafıza birimi üzerinden gerçekleştirildiği için bu veriler ortak bir yol kullanılarak iletilmektedir. Dolayısıyla, bu operasyonların aynı anda gerçekleştiği durumlarda bir gecikme durumu söz konusu olabilmektedir.

Harvard Mimarisi yapısında birbirinden bağımsız çalışmakta olan iki bellek bulunmaktadır. Bu yüzden, veri iletimi ve komut alma operasyonlarının gerçekleştirildiği üniteler farklı yerlerde bulunmaktadır (Şekil 2.3).



Şekil 2.2. Harvard mimari yapısı.

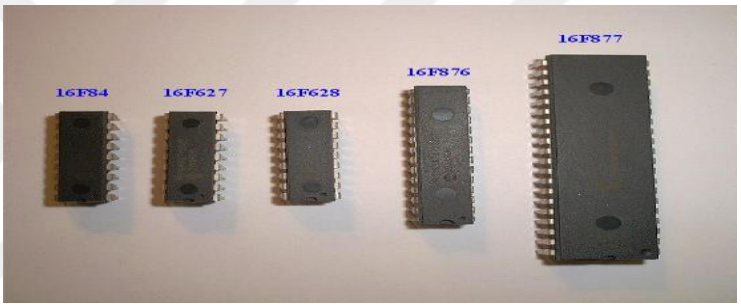
Komut aktarımı ve veri iletişimini gerçekleştiren operasyonların farklı yolları kullanması sayesinde bu iki operasyonun eş zamanlı olarak gerçekleştirilmesi mümkündür. Bu demektir ki, hafıza biriminden komutların okunması sırasında bu operasyon için gerekli olan veri aynı zamanda bekleme olmadan okunabilmektedir.

Performans bakımından kritik öneme sahip sistemler üzerinde bu mimari oldukça tercih edilmektedir. Ayrıca hassas veri işlemesinde kullanılan ve DSP (Digital Signal Processor) devrelere sahip mikrodenetleyicilerde Harvard mimarisi tercihi, yüksek performans sunmasından dolayı oldukça fazladır.

2.4. PIC Mikrodenetleyici

General Instruments tarafından geliştirilmiş olan PIC mikrodenetleyici, giriş-çıkış işlemlerini çok hızlı bir şekilde gerçekleştirebilen bir mikrodenetleyici olarak üretilmiştir.

PIC mikrodenetleyicinin sahip olduğu mimari türü Harvard mimarisidir. Bu mimaride, komut ve bilgi saklama bellekleri birbirinden bağımsız yapıdadır ve bunun sonucu olarak diğer mimari türündeki mikrodenetleyicilere göre maliyeti daha fazladır fakat teknolojik gelişimlerin yardımı ile fiyat farkı zamanla ortadan kalkmıştır. Veri belleğinde kullanılan statik RAM, program belleğinde kullanılan "flash" belleğe kıyasla oldukça hızlıdır ve bu nedenle Harvard mimarisindeki mikrodenetleyicilerin çok daha hızlı olduğu bilinmektedir (Robotiksistem, 2009).



Şekil 2.3. Popüler PIC mikrodenetleyiciler.

Mikrodenetleyicinin RAM, ROM (Read Only Memory), CPU (Central Processing Unit), TIMER ve COUNTER yapılarından oluşan bütünleşmiş bir çip olduğu bilinmektedir (PIC, 2017). PIC, bunlara ek olarak ADC (Analog Dijital Dönüştürücü), DAC (Dijital to Analog Converter) de içeren bir mikrodenetleyicidir. PIC mikrodenetleyici ayrıca, ek çevre birimleriyle ara yüz oluşturmak için UART (Universal Asynchronous Receiver/ Transmitter), SPI (Serial Peripheral Interface), ve CAN (Controller Area Network) gibi protokolleri de desteklemektedir.

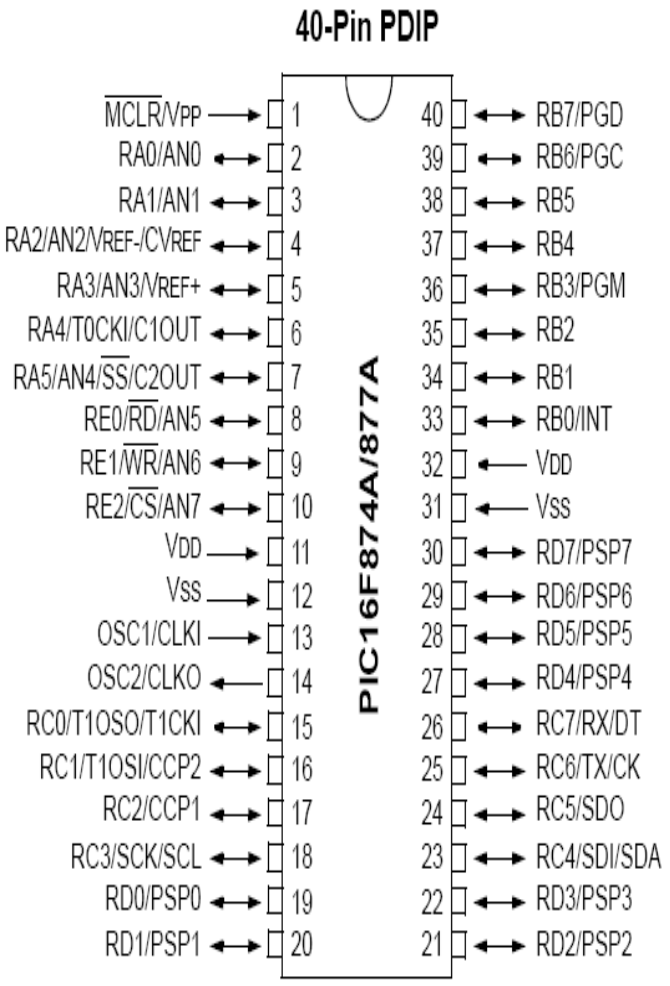
PIC mikrodenetleyici ailesinin tasarımında RISC (Reduced Instruction Set Computing) işlemci türü kullanılması sayesinde yüksek hızda çalışma mümkün kılınmıştır. PIC mikrodenetleyicilerde az sayıda komut bulunması ve bunların bir çevirim içinde işleme girmesi sayesinde yüksek hız elde edilmektedir. Bu mikrodenetleyici türünde bir çevirimde işlenmeyen tek komutlar "call" ve "goto"

komutlarıdır.

2.4.1. PIC 16F877A Genel Özellikleri

PIC, bir çevre birimleri denetleyici şeklinde tanımlanmaktadır ve mikrodenetleyici kategorisinde değerlendirilmektedir. Yapısında giriş- çıkış, PWM (Pulse Width Modulation), ADC, RAM, CPU bulunduran birleşik bir sistem olmakla birlikte, içerisine yazılmış olan koda göre giriş- çıkış birimlerini kontrol ederek yönlendirebilen elektronik bir beyin olarak karşımıza çıkmaktadır.

16F877A 8 bitlik bir mikrodenetleyicidir. 16F877A'nın üzerindeki 40 pinden 33 tanesi giriş- çıkış pinleridir (Şekil 2.5).



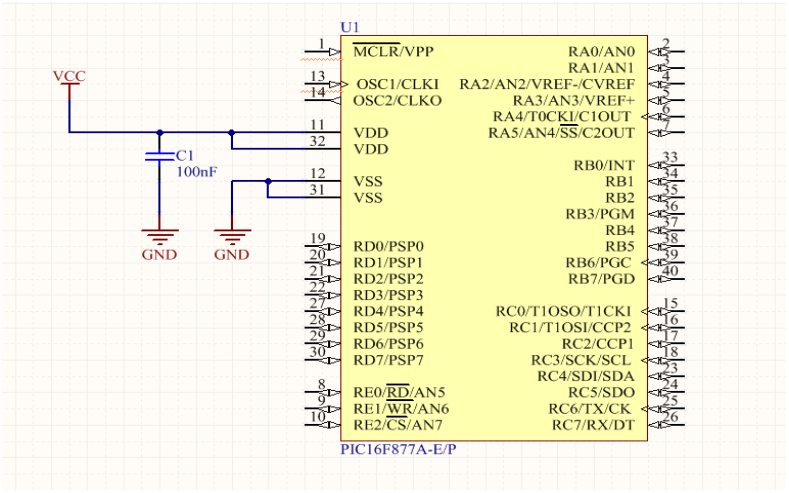
Şekil 2.4. PIC 16F877A pin görünüş şeması.

PIC 16F877; 6 bitlik A portu, her biri 8 bitlik B, C ve D portları ve 3 bitlik E portu olmak üzere 5 porta sahiptir. Giriş- çıkış pinlerinin gerekli konfigürasyonlar yapılarak başka amaçlarla kullanılması da mümkündür (Robotiksistem, 2009). Bu mikrodenetleyicinin ulaşabileceği en yüksek hız 20 Mhz'dir ve bu demektir ki bir komut 200 ns hızında çalışmaktadır. 14 kaynaktan kesme yapabilmektedir ve 1.000.000 defa programlanabilen 8 Kword değerindeki programlama belleğine sahiptir. Ayrıca, enerji tasarrufu sağlayan uyku modu özelliği olmakla birlikte programlama işlemi sırasında küçük gerilimle çalışabilmektedir. Programlanabilme aşamasında enerji girişi olarak sadece 5V değerinde bir ihtiyaç ile seri bağlanarak çalışmaktadır. Besleme akımı 25 ma değerindedir ve düşük güç tüketimi ile yüksek sıcaklıklarda çalışabilmektedir.

PIC 16F877A mikrodenetleyici sekiz kattan oluşan yığın yapısındadır ve bu durum birbiri içine geçmiş şekilde sekiz alt program çağrılabilmesine olanak vermektedir (Bodur, 2014). Ayrıca CPU çalışmasına etki eden birkaç farklı nitelik bulunmaktadır. Yalnızca iki pin kullanılarak ICSP (In Circuit Serial Programming) özelliği ile PIC 16F877A, uygulama devresi üzerindeyken programlanabilmektedir.

2.4.1.1. PIC 16F877A Besleme Gerilimi

Besleme bağlantılarının çizilmesi, PIC ile oluşturulan sistemin tasarım aşamasında yapılması gereken ilk iştir. Mikrodenetleyicinin 12 ve 31 numaralı pinleri Vss negatif, 11 ve 32 numaralı pinleri ise Vdd pozitif beslemedir (Şekil 2.6).



Şekil 2.5. PIC 16F877A besleme bağlantıları.

Devreye enerji verildiği zaman oluşabilecek dalgalanmaları dengeleyebilmek amacıyla Vss ile Vdd arasına 100nF dekuplaj kapasitörü konulmalıdır.

Besleme gerilimleri duruma göre değişiklik göstermekle birlikte 2 ve 5.5V arasında olabilmektedir. Örnek akım çekme durumları frekans ve beslemeye göre değişiklik göstermektedir (Çizelge 2.1).

Çizelge 2.1. Osilatör için uygulanabilir kondansatör miktarları.

Çalışma Frekansı	Besleme	Akım
4 Mhz	5.5V	1.6 mA
32 Khz	3V	20 µA
Boşta Bekleme Anı		1 µA

2.4.1.2. Osilatör Ayarları ve Kondansatör Seçimi

PIC 16F877A mikrodenetleyicilerde kullanılan kondansatör seçimi saat stabilizasyonunu sağlamak bakımından oldukça önemlidir. PIC mikrodenetleyici için kondansatör değerlerinin ayarlanmasında 3 mod bulunmaktadır ve bu modlardaki değerler üretici tarafından test edilmektedir.

Osilatör tipi tasarlanacak devrenin çeşidine göre değişmektedir ve farklı devrelere göre LP, XT ve HS tip osilatörler bulunmaktadır (Çizelge 2.2).

Çizelge 2.2. Osilatör için kullanılabilir kapasitör miktarları.

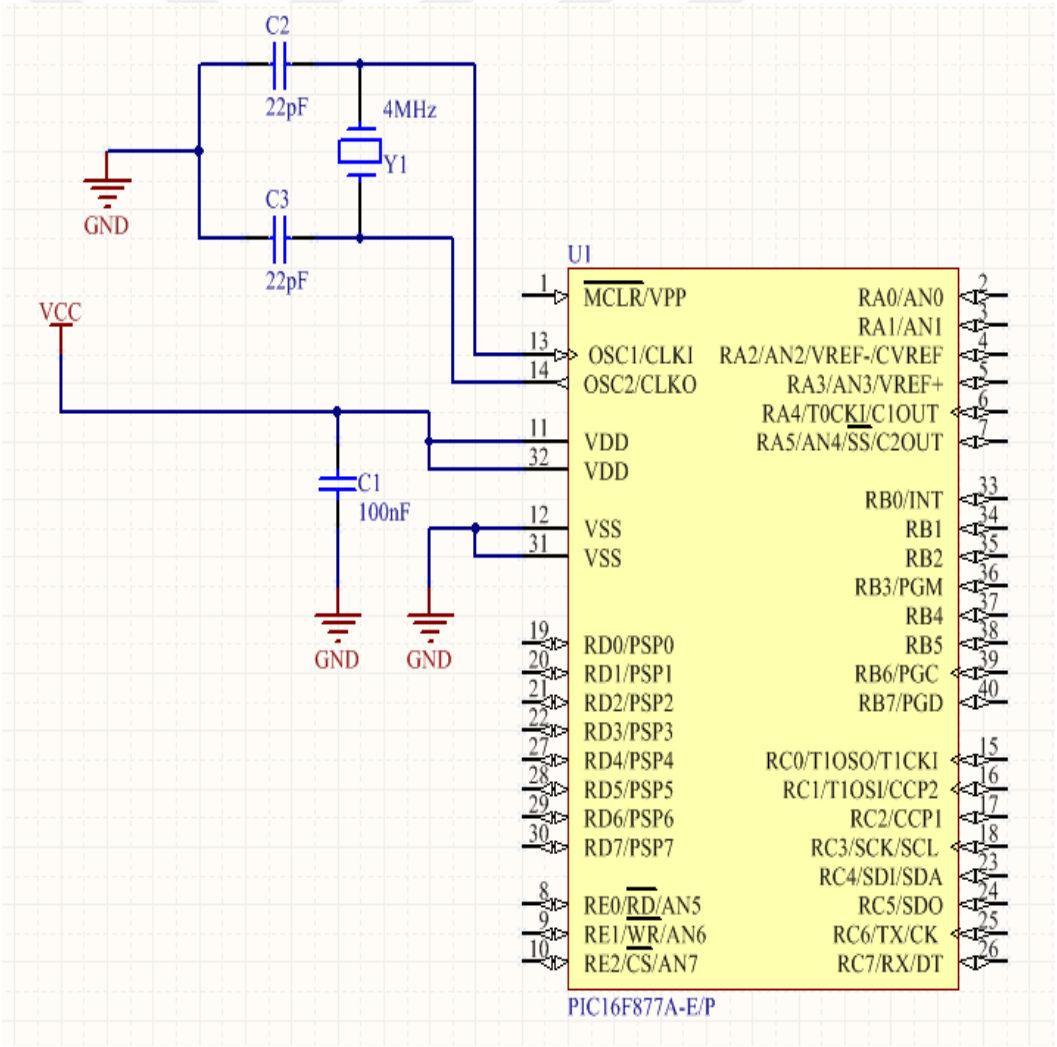
Mod	Frekans	OSC1 / C1	OSC2 / C2
LP	32kHz	68-100pF	68-100pF
	200kHz	15-33pF	15-33pF
XT	100kHz	100-150pF	100-150pF
	2.0MHz	15-33pF	15-33pF
	4.0MHz	15-33pF	15-33pF
HS	4.0MHz	15-33pF	15-33pF
	10.0MHz	15-33pF	15-33pF

Osilatör ayarlamaları, temel bağlantı yapma aşamasında gelen ikinci adımdır. Osilatörün işlevi, komutların işlenebilmesi için bir “clock” sinyali üretmektir. PIC 16F877A mikrodenetleyicisi üzerinde çalıştırılabilmesi için osilatör, 13 ve 14 numaralı

pinlere bağlanmalıdır (Şekil 2.7).

Mikrodenetleyiciye bağlanmış olan bütün osilatörlerin değer frekanslarına ulaşma zamanı, enerji verildiği andan itibaren kısa süreli bir gecikmeden sonra gerçekleşmektedir. Osilatörün istenen frekansta "clock" sinyali üretmeye başlaması, besleme gerilimi normal değere eriştiğinde gerçekleşmektedir.

Kristal frekansının 4 parçaya bölünerek tersinin alınması ile komutların her birinin işlem zamanı hesaplanabilmektedir. Örnek vermek gerekirse; 20 MHz kristalin olduğu bir sistemde frekans değeri 5Mhz olmaktadır. Böylece, bir komutun işlenme zamanı 1/5Mhz = 0.2 usn olarak hesaplanmaktadır. Mikrodenetleyicinin gereksinim duyduğu saat sinyali, kristal osilatörden başka harici osilatör veya R/C osilatör bağlanması ile de sağlanabilmektedir.



Şekil 2.6. Osilatör bağlantı şeması.

2.4.1.3. PIC 16F877A Sıfırlama Ayarları

PIC 16F877A mikrodenetleyici içinde birden fazla sıfırlama tipi barındırmaktadır. Bunlar:

- OST (Oscillatör Start- Up Timer): Denetleyiciyi, 1024 çevrimlik gecikme zamanı süresince sıfırlama konumunda tutmaktadır. Kristal osilatörün kararlı duruma gelebilmesi için bu süre gereklidir.
- PWRT (Power Up Timer): Mikrodenetleyici ilk çalıştığında devreye girmesini engellemek amacıyla 72 ms kadar bir süre boyunca sistemin sıfırlama durumunda kalmasını sağlamaktadır. Bu süre boyunca besleme geriliminin yeterli seviyeye gelmesi sağlanmış olmaktadır. Bu tip sıfırlama yöntemi, kararlı bir şekilde çalışmanın önemli olduğu sistemlerde tercih edilmektedir.
- BOR (Brown Out Reset): Sıfırlamanın, sistemde gerilim düşüşü gerçekleştiği durumlarda oluşmasıdır.
- POR (Power On Reset): Sıfırlamanın oluşması, sisteme beslemenin gelmesiyle birlikte başlamaktadır. POR sinyali, Vdd pininde 1.2- 1.7V aralığında bir gerilim oluşması algılandığında oluşmaktadır.

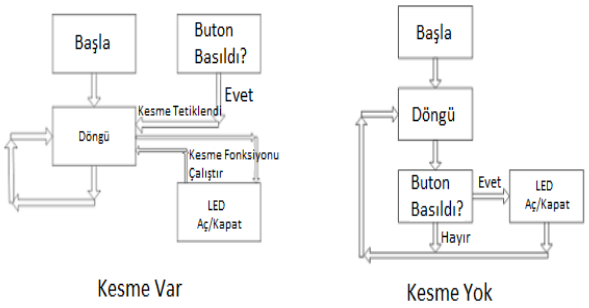
2.4.1.4. Kesmeler

Mikrodenetleyici çalışma mekanizması, hafızaya yazılan programın satır satır okunması ve her satırdaki program komutunun işlenmesi şeklinde ilerlemektedir. Programın tasarlanma şekli ve akışına göre ‘subroutine’ adı verilen alt program parçalarına geçiş yapılabilmektedir. Fakat gereken durumlarda programın normal akışının dışına çıkılması gereken zamanlar olabilmektedir. Kesmeler bu durumların gerçekleştiği zamanlarda devreye girmektedir.

Burada dikkat edilmesi gereken nokta, alt program ile kesmenin birbirinden tamamen farklı olmasıdır. Programın hangi durumlarda alt programlara geçmesi gerektiği, program yazılırken belirlenirken, kesme ise ayrı bir uyarıcı sayesinde

program dışında gerçekleşmektedir. Bu uyarıcı, kullanılan zamanlayıcının dolması ile oluşmuş bir sinyal, seri haberleşme biriminden alınan bir veri veya pine bağlı sensörden alınmış 5V değerinde bir sinyal olabilmektedir. Kesme sinyali ulaştığı anda programın kesme vektörüne dallanması ile bu kısımdaki komutların işlenmeye başlamasıdır.

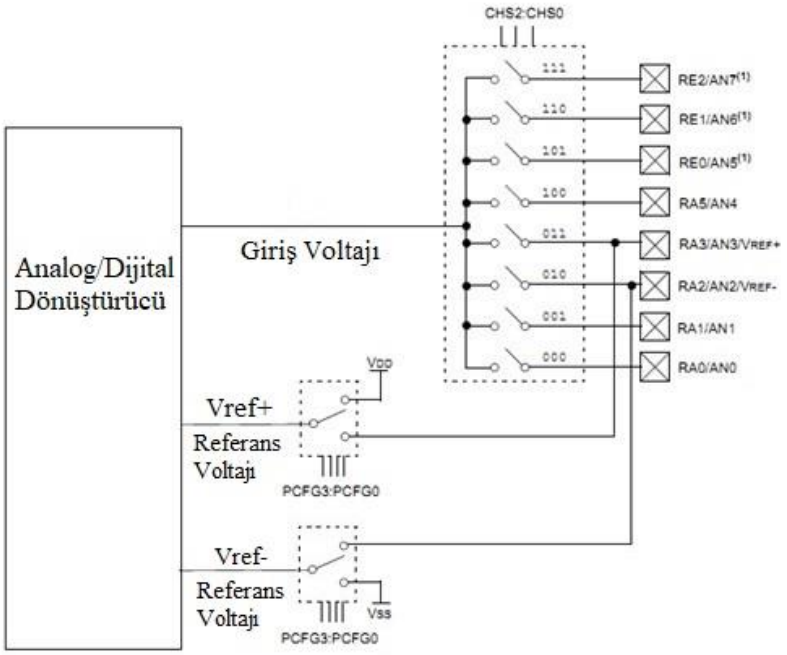
Kesme mantığı oluşturulan görseller sayesinde daha kolay anlaşılabilmektedir (Şekil 2.8).



Şekil 2.7. Kesme kullanımının etkisi.

2.4.1.5. Analog Sayısal Dönüştürücü

ADC (Analog Dijital Dönüştürücü) birimlerinin en önemli özelliklerinin bit sayıları olduğu bilinmektedir. 8 bitlik bir mikrodenetleyicinin üretebildiği değerler 0-255 arasında iken, 10 bitlik bir ADC 0- 1023 aralığında değerler üretmektedir. Çözünürlük, ADC birimindeki bit sayısı ile doğru oranda artmaktadır. Örneğin; çıkış genliği 0- 5 V aralığında değişmekte olan bir basınç sensörü ve 10 bitlik bir ADC birimi ile 5V/1023 = 0.0048875855327468 değerinde bir sonuç elde edilmektedir. Bu sonuç, ADC biriminin hassasiyetini göstermektedir. Bu demektir ki ADC birimi 4.8m V değerindeki voltaj değişimlerini algılamaktadır. ADC biriminden okunulan değerin 15 olduğu durumda ise girişteki voltaj değeri 15*0.0048875855327468 =~ 2 V değerinde olmaktadır.

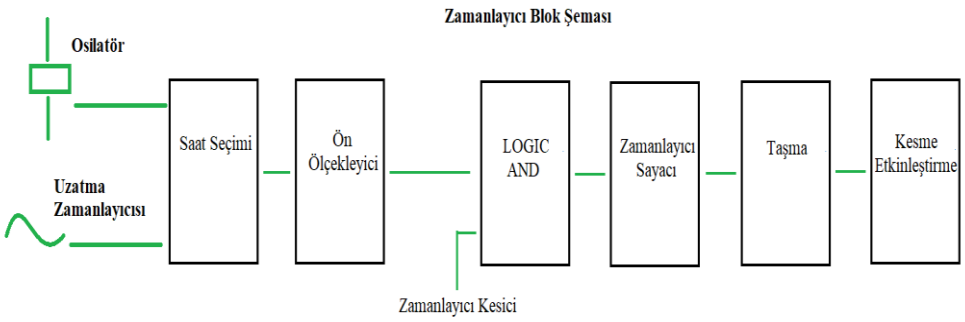


Şekil 2.8. Analog dijital dönüştürücü kullanımı.

2.4.1.6. Zamanlayıcı Modülleri

Doğru bir biçimde zaman gecikmelerinin oluşturulması ve zamanın ölçülmesinin gerçekleştirilmesini sağlamak amacıyla zamanlayıcı modülleri kullanılmaktadır. Zamanlayıcı modüllerin haricinde mikrodenetleyici içinde tekrar eden döngüler oluşturularak, istenen zaman gecikmeleri elde edilip ölçülebilmektedir fakat bu yöntemin mikrodenetleyiciye gereksiz yük eklemesinden dolayı tercih edilmemesi gerekmektedir. Zamanlayıcının çalışma mantığı, basit bir ikili sayacın saat atımlarını sayması ve en yüksek değere vardığında sıfırlanması şeklinde gerçekleşmektedir.

Adından da anlaşılacağı gibi, zamanlayıcı modülleri zamanı ölçmek veya doğru zaman gecikmesini oluşturmak için kullanılmaktadır. Mikrodenetleyici, ayrıca döngüleri çalıştırarak gerekli zaman gecikmelerini üretip ölçebilmektedir. Fakat, zamanlayıcı, CPU'yu bu fazlalık ve tekrarlayan görevden kurtararak diğer görevler için maksimum işlem süresi tahsis edilebilmesine olanak sağlamaktadır (Şekil 2.10).



Şekil 2.9. Zamanlayıcı blok diyagramı.

Zamanlayıcı hesaplaması, osilatör frekans değerinin, kontrolörü beslemesiyle gecikme aralığının oluşturulması için "preskalar" değere bölünmesi ile zamanlayıcı sayısının değerinin 1 artması şeklinde belirlenmektedir.

2.4.1.7. PIC Bellek Yapısı

PIC mikrodenetleyicilerde veri belleği, EEPROM (Electrically Erasable Programmeble Read Only Memory) ve program belleği olmak üzere 3 çeşit hafıza bulunmaktadır.

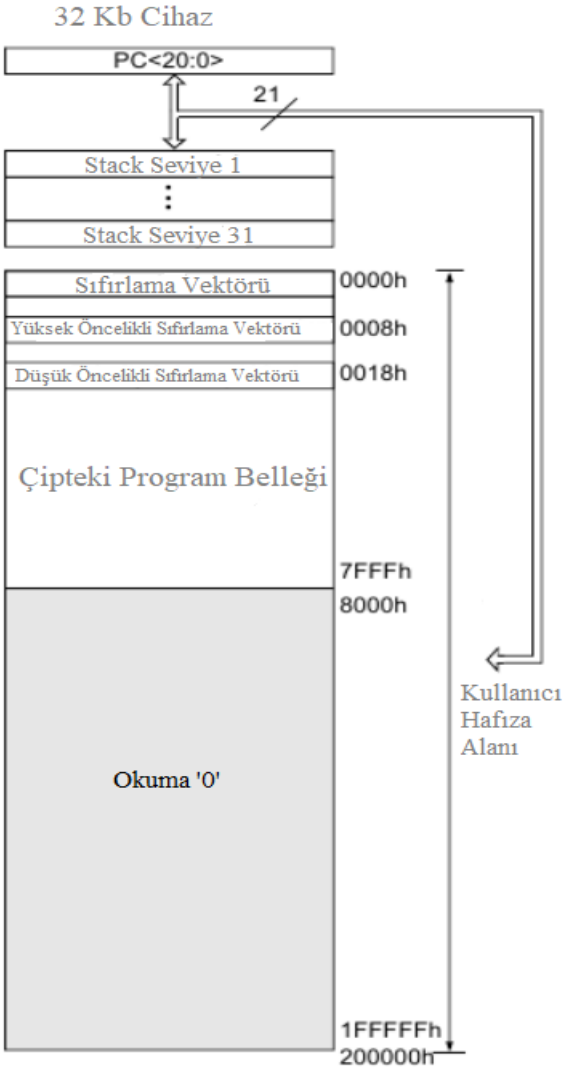
Birçok PIC mikrodenetleyici türünde EEPROM bellek bulunması sayesinde, ihtiyaç duyulan durumlarda veriler kaybolmadan yazılıp okunabilmektedir çünkü elektrik beslemesinin kesildiği durumlarda bile saklanan veriler silinmemektedir.

"CALL" komutu, "RETURN" komutu ile sonlandırılması gereken bir alt rutini atlamak için kullanılmaktadır. "CALL" komutu, işlenen olarak alt yordamdaki ilk talimatın adresine sahiptir. "CALL" komutu yürütüldüğünde, hedef adres "PC" içerisine kopyalanmaktadır. Bir "CALL" komutu yürütüldüğünde veya bir kesme dallanmaya neden olduğunda, "PC" yığının üzerine itilmektedir. Yığın, bir "RETURN", "RETLW" veya "RETFIE" komutlarının yürütülmesi durumunda "POP" (yığından çekme işlemi) gerçekleşmektedir.

Yığın, dairesel bir tampon olarak çalışmaktadır. Bu, yığın sekiz kez "PUSH" (yığına itme işlemi) yapıldıktan sonra, dokuzuncu itmenin ilk itmeden itibaren saklanan değerin üzerine yazılacağı anlamına gelmektedir. Onuncu itme, ikinci

itmenin üzerine yazılmaktadır.

Program komutlarının saklandığı hafıza türü olan program hafızası, “FLASH” bellek yapısında olmasından dolayı tekrar tekrar silinerek yeniden programlanabilmektedir. PIC 16F877A mikrodenetleyicisinde, bu mikrodenetleyiciye özgün 8K(8*1024 = 8192) program belleği bulunmaktadır. Bu demektir ki 16F877A, hafızada 8192 adet 14 bitlik komut saklama özelliği bulunmaktadır. PIC 16F877A mikrodenetleyicisinin hafıza haritasında 000Fh- 1FFFh adresleri arasındaki kısım, “On- Chip” program belleği için ayrılmıştır (Şekil 2.11).



Şekil 2.10. Çipteki program belleği.

Bellekteki adreslere erişim, program denetleyici tarafından yapılmaktadır. PIC 16F877A mikrodenetleyicisinde 13 bitlik program denetleyici bulunmaktadır ve

2^13= 8192 adresleme yapılmaktadır.

Ana program çalışması her başladığında 0000 adresinde sıfırlama vektörü, 0004 adresinde ISR (Interrupt Service Routine) için ayrılmıştır.

Bir kesme kullanılması planlanıyor ise, programın kesinti vektöründen sonra başlayacaktır; değilse sıfırlama vektörünün başından yazmaya başlamaktadır. Belleğin bir kısmı, programı bunlara yazmak için tasarlanmış sayfalara bölünmüştür; kalan bellek (Yığın, Kesinti Vektörü ve Sıfırlama Vektörü) donanım kayıtlarıdır.

BÖLÜM 3

MATERYAL VE YÖNTEM

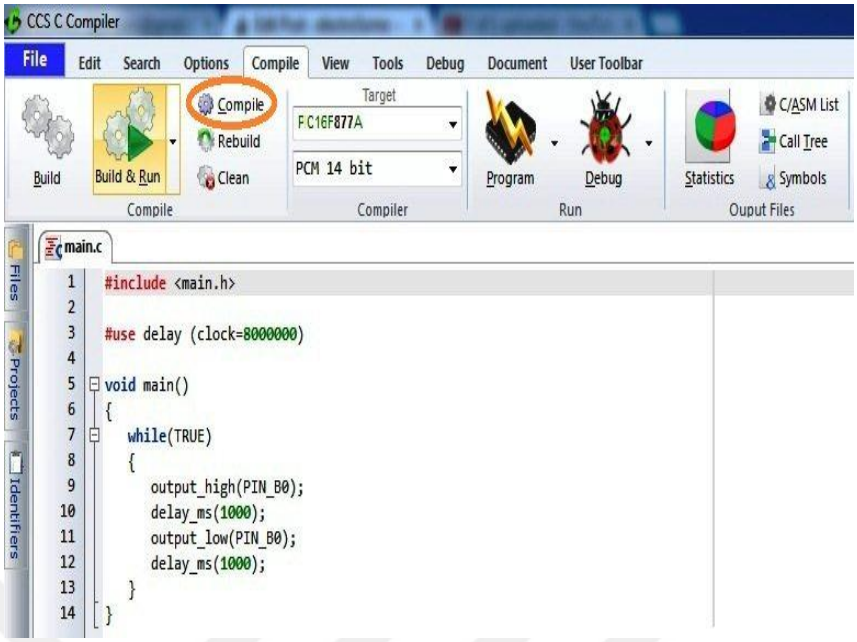
Bu sistemin geliştirilmesi sırasında simülasyon ortamı olarak Proteus Design Suite, derleyici olarak CCS- C Derleyici, USB (Universal Serial Bus) programlayıcı olarak PIC KIT Programlayıcı ve programlama dili olarak C Programlama Dili kullanılmıştır.

3.1. CCS- C Derleyici Programı

Diğer tüm mikrodenetleyiciler gibi PIC mikrodenetleyici, Assembly Dili kullanılarak programlanabilmektedir. Bir PIC Mikroişlemcisini programlamak için CCS- C, MPLAB ve MikroC gibi birçok yüksek seviyeli dil derleyicisi bulunmaktadır. CCS, bir Microchip PIC Micro Controller Tool Solutions şirketi olan Custom Computer Services'in kısaltmasıdır ve C Programlama dili ile programlandığı için bu adı almıştır.

MikroC ve CCS C derleyicileri, derin dahili mimari bilgisi olmadan bir PIC mikrodenetleyicisini programlamamıza olanak tanıyan çok sayıda yerleşik kütüphane içerdiğinden, yeni başlayanlar için en iyi derleyicilerdir. PIC10, PIC12, PIC14, PIC16 ve PIC18 mikrodenetleyicileri için CCS C Derleyicisi, MCU (Micro Controller Unit) donanımına erişmek için 307 yerleşik fonksiyona sahiptir. Verimli ve yüksek düzeyde optimize edilmiş kod üretmektedir (Şekil 3.1).

Derleyici, satır içi işlevleri ve yeniden kullanılabilir yazmaçlarda parametre geçişini işleyebilmektedir. Kullanıcı için şeffaf olan derleyici, sayfaları otomatik olarak ele alarak RAM ve ROM kullanımını optimize etmek için program yapısını analiz ederek ağaç işlemlerini çağırmaktadır.



Şekil 11 CCS- C derleyici.

Derleyicinin kurulabilmesi için lisans ve kayıt dosyaları mutlaka bulunmalıdır. Ayrıca lisans dosyalarının program kurulum dosyasında yer alması da gerekmektedir.

3.2. C Programlama Dili

C programlama dili hem çok yönlü hem de popüler olan ve çok çeşitli uygulamalarda ve teknolojilerde kullanılmasına izin veren bir programlama dilidir. Örneğin, işletim sistemleri, çok daha karmaşık programlar ve aradaki her şey için kod yazmak için kullanılabilmektedir. Sadeliği ve esnekliği büyük ölçüde, endüstrinin temel programlama dillerinden biri haline gelmesine neden olan makinelerden bağımsız olarak çalışabilmesinden kaynaklanmaktadır.

C dilini anlamak, Java ve C ++ gibi C'de kullanılan özellikleri ve sözdizimini ödünç alarak temel olarak C'yi kullanan çok çeşitli diğer programlama dillerini kolayca öğrenmenize ve kullanmanıza olanak tanımaktadır.

C, başlangıçta işletim sistemlerini yazmak için geliştirilmiş, oldukça verimli ve basit bir programlama dilidir. Onu çok esnek ve kullanımı kolay kılan birçok faydası ve özelliği arasında, hafızaya düşük seviyeli erişim, temiz ve özlü bir stil ve basit bir anahtar kelime kümesi vardır. Ek olarak, bir sistem için C kullanılarak yazılan kaynak kodu, herhangi bir değişiklik yaşamadan başka bir işletim sisteminde aynı etkinlikte

çalışabilmektedir.

C, UNIX işletim sistemleri için bir programlama dili olarak geliştirilmiş olsa da artık hemen hemen tüm donanım platformlarında ve işletim sistemlerinde kullanılmasına izin veren birçok derleyiciye sahiptir. İlk popüler olmaya başladığında, ANSI olarak da bilinen Amerikan Ulusal Standartlar Enstitüsü, programlama dili için ticari bir standart oluşturmayı gerekli bulmuştur. O zamandan beri, Uluslararası Standartlar Organizasyonu tarafından da onaylandı ve şimdi bazen "ANSI C" olarak anılmaktadır.

Şekil 3.2'de RS232 üzerinden ADC örneklerini okumak için CCS- C kullanan açıklamalı örnek bir program gösterilmiştir.

```
#include <16F877A.h>
definitions
#fuses NOPROTECT
#use delay(clock=20000000)
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)
void main() {
unsigned int8 i, value, min, max;
printf("Sampling:"); // Printf from the RS232 library

setup_adc_ports(AN0); // Make AN0 a analog pin
setup_adc(ADC_CLOCK_INTERNAL); // Start up the ADC
set_adc_channel(0); // Set ADC channel to AN0

do {
min=255;
max=0;
for(i=0; i<=30; ++i) {
delay_ms(100);
library
value = read_adc();
if(value<min)
min=value;
if(value>max)
max=value;
}

printf("\r\nMin: %2X Max: %2X\n\r",min,max);
} while (TRUE);
}
```

Şekil 3.2. CC-C derleyici içinde C Programlama kullanımı.

Her C programı, program yürütmenin başlangıç noktası olan bir ana işlevi içermelidir. Program, amaçlarına göre birden fazla işleve bölünebilir ve işlevler ana

veya alt işlevlerden çağrılabilmektedir. Büyük bir projede farklı kütüphanelerdeki fonksiyonların kullanılabilmesi için bu kütüphaneler ana dosyaya dahil edilebilmektedir. CCS- C ayrıca, aygıta özgü işlevselliği dahil etmek için "#include" yönergesini kullanarak uygun aygıt dosyasını dahil etmeyi gerektirmektedir. Ayrıca, yonga için sigortaları belirtmek için "#fuses" ve saat hızını belirtmek için "#use" gecikmesi gibi bazı ön işlemci yönergeleri de bulunmaktadır. Fonksiyonlar veri bildirimlerini, tanımları ve ifadeleri içermektedir. Derleyici ayrıca çok sayıda standart C kitaplığının yanı sıra programlara dahil edilebilen ve kullanılabilen diğer aygıt sürücülerini de sağlamaktadır. CCS ayrıca, PIC mikrodenetleyicide bulunan çeşitli çevre birimlerine erişmek için çok sayıda yerleşik işlev sağlamaktadır.

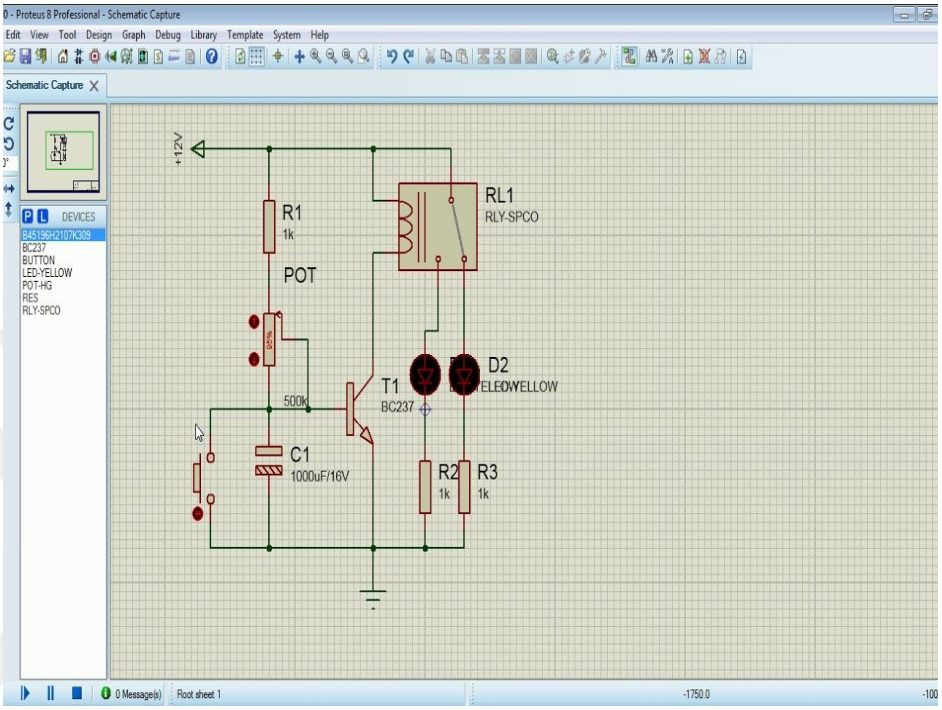
3.3. Proteus Design Suite Simulasyon Programı

Proteus; şematik yakalama, simülasyon ve PCB (Printed Circuit Board) yerleşim modülleri içeren bir elektronik devre tasarım yazılımıdır. PCB tasarımı için kullanılmıyor olsa bile, bileşen seçilirken PCB düzeni ayrı ayrı görüntülenebilmesi sayesinde, PCB içerisindeki bileşenlerin lehimlemesi sırasında yardımcı olmaktadır.

PVSM (Proteus Virtual System Modeling), mikrodenetleyici tabanlı tasarımların tamamının birlikte simülasyonunu kolaylaştırmak için karışık mod "SPICE" devre simülasyonunu, animasyonlu bileşenleri ve mikrodenetleyici modellerini birleştirmektedir. Şimdiye kadar ilk kez, bu tür tasarımları fiziksel bir prototip yapılmadan önce geliştirmek ve test etmek mümkün olmuştur. Bu durum LED (Light Emitting Diode) ve LCD (Liquid Crystal Display) ekranlar gibi ekran göstergelerinin, anahtar ve düğme gibi çalıştırıcıları kullanarak tasarımla etkileşim kurabilmesi sayesinde mümkündür. Proteus, ayrıca hem montaj kodu hem de yüksek seviyeli dil kaynağı için kesme noktaları, tek adımlama ve değişken ekran dahil olmak üzere kapsamlı hata ayıklama olanakları sağlamaktadır. Proteus, tasarım girişi ve geliştirme için ortam sağlamak için kanıtlanmış ISIS Şema Yakalama yazılımını kullanmaktadır. ISIS, köklü bir üründür ve kullanım kolaylığını güçlü düzenleme araçlarıyla birleştirmektedir. Hem simülasyon hem de PCB tasarımı için şematik yakalamayı destekleyebilmektedir.

Mikrodenetleyici modeli, ürün tasarımının diğer unsurları ile şemaya

oturmaktadır. Tıpkı gerçek bir çip gibi makine kodunun yürütülmesini taklit etmektedir. Program kodu bir porta yazıyorsa, devredeki mantık seviyeleri buna göre değişmektedir. Devre işlemcinin pinlerinin durumunu değiştirirse bu gerçek hayatta olduğu gibi program kodu tarafından görülmektedir (Şekil 3.3).



Şekil 3.3. Proteus Professional simulasyon geliştirme ortamı.

Proteus, tamamlanmış mikrodenetleyici sistemlerinin gerçek zamanlı simülasyonlarını çalıştırma kabiliyetinde neredeyse kusursuz olsa da gerçek gücü bu simülasyonları tek adımlı modda gerçekleştirme yeteneğinden gelmektedir. Bu da demektir ki en iyi yazılım hata ayıklayıcısı gibi çalışmaktadır. Kod tek adımda tutulduğunda, mikrodenetleyicinin dışındaki tüm tasarım üzerindeki etkinin gözlemlenebilmesidir.

Simülasyon CPU modelleri, desteklenen her işlemcide bulunan giriş- çıkış bağlantı noktalarını, kesintileri, zamanlayıcıları ve diğer tüm çevre birimlerini tam olarak taklit edebilmektedir. Basit bir yazılım simülatöründen başka bir şey değildir, çünkü tüm bu çevre birimlerinin dış devre ile etkileşimi tamamen dalga biçimi seviyesinde modellenmiştir.

Proteus, bir Mantık Analizörü, Fonksiyon Üreteci, Model Üreteci ve Sanal

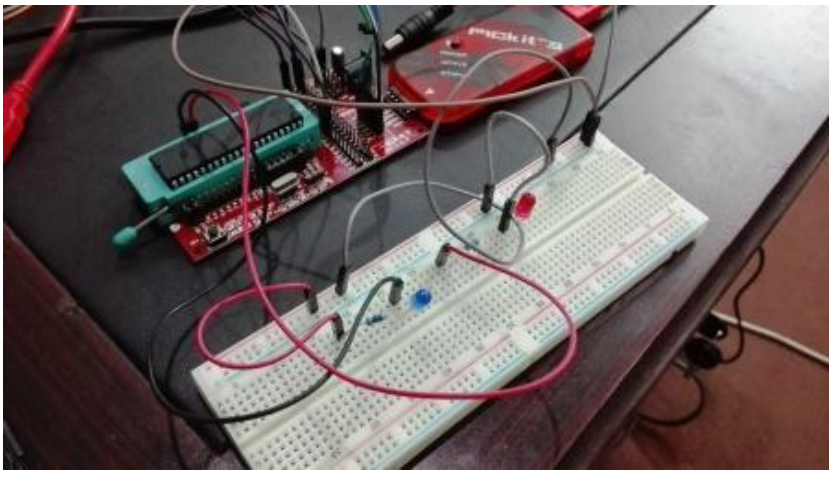
Terminal'in yanı sıra basit voltmetreler ve ampermetreler dahil olmak üzere bir dizi sanal enstrümanı içermektedir. Ek olarak, simülatör, renkli noktalar kullanarak her pinde mantık durumunu görüntüleyebilmektedir. Bu, tek aşamalı giriş- çıkış kodu oluştururken son derece yararlı olmaktadır. Proteus programının en iyi özelliklerinden biri, bir mikrodenetleyici üzerinde çalışan yazılım ile ona bağlı herhangi bir analog veya dijital elektronik arasındaki etkileşimi canlandırma yeteneğidir.

3.4. PICKIT Programlayıcısı

PIC mikrodenetleyicilerini programlamak için, bilgisayar ile programcı olarak bilinen mikrodenetleyici arasında iletişim kurabilen bir donanım parçasına ihtiyacımız vardır. Piyasada halihazırda birkaç programlayıcı bulunmaktadır, ancak çok yönlülüğü nedeniyle PICKIT en çok tercih edilen programlayıcılar arasında sayılmaktadır.

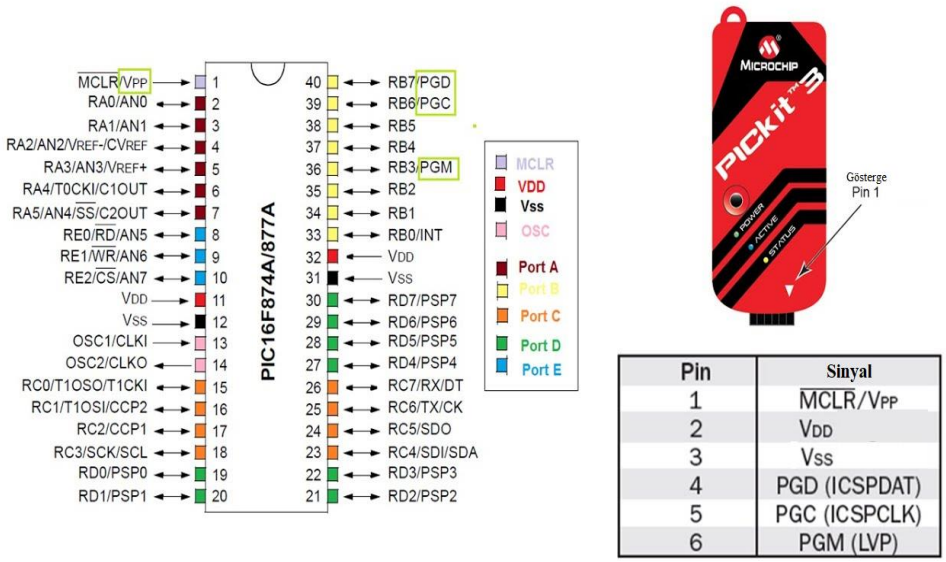
Bazı 8, 14 ve 18 pinli cihazlar, hata ayıklama için özel bir MCU'ya sahip küçük başlık kartları kullanmaktadır. Bu özel MCU, PICKIT iletişimi için ekstra pinlere sahiptir ve bu nedenle uygulama için parçadaki tüm pinlerin kullanımına izin vermektedir. Başlık panosu programlama için kullanılmaz veya gerekli olmamaktadır. Ancak, bu cihazlarda hata ayıklanırken başlık kullanılmalıdır.

Şekil 3.4'te mikrodenetleyicinin PICKIT3 üzerinde kullanım şekli ve bağlantı kablolarının yerleştirme yöntemi gösterilmiştir. Mikrodenetleyici direkt kit üzerine yerleştirilerek, bilgisayar tarafından görülebilmesi sayesinde kullanıcıya kullanım kolaylığı sağlamaktadır.



Şekil 3.4. PICKIT bağlantı gösterimi.

PICKIT, IDE'nin (Integrated Development Environment) güçlü grafik kullanıcı arabirimini kullanarak PIC ve flash mikrodenetleyicilerinin en uygun fiyat noktasında hata ayıklama ve programlanmasına izin vermektedir. PICKIT, tasarım mühendisinin bilgisayarına yüksek hızlı bir USB ara yüzü sayesinde bağlanarak, hedefe bir hata ayıklama bağlayıcısı aracılığıyla bağlanabilmektedir. Bağlayıcı, devre içi hata ayıklama ve devre içi seri programlama uygulamak için iki cihaz giriş- çıkış pini ve sıfırlama hattını kullanmaktadır (Şekil 3.5).



Pin	Sinyal
1	MCLR/VPP
2	VDD
3	Vss
4	PGD (ICSPDAT)
5	PGC (ICSPCLK)
6	PGM (LVP)

Şekil 3.5. PICKIT 3 pinleri.

PICKIT3 aşağıdaki bileşenlerle birlikte gelmektedir;

- Kordon Bağlantısı
- USB Bağlantı Noktası
- Pin 1 İşaretleyici
- Programlama Bağlayıcı
- Durum LED'leri
- Basma Butonu



Şekil 3.6. PICKIT 3 bileşenleri.

BÖLÜM 4

ARAŞTIRMA BULGULARI

4.1. Araştırma Kapsamındaki Alanlar

Bu tezde gerçekleştirilen iş, sıfırdan geliştirilen bir algoritma tasarımı sayesinde kullanıcı isteklerine göre şekillenebilen, dinamik ve esnek olarak kullanılan bir sistem ortaya konulmasıdır.

Araştırılan diğer bahçe sulama sistemlerinde bir düğme yardımı ile aç/ kapat şeklinde yürütülebilen sistemlerden farklı olarak tasarlanan algoritmalar sayesinde, sistem kendi zaman sayacını içinde barındırması ile harici bir ekipman veya modüle ihtiyaç duymadan zaman sayımı ve ayarı yapabilmektedir. Tekdüze çalışma mantığından kurtulmak amacıyla eklenen gün kontrol mantığı, çalışmayı farklı kılan özelliklerdendir.

Tezin konusunu kapsayan ve araştırma konusu olan bir diğer özellik, programın kullanıcı tarafından belirlenen metrekare verisine göre hesaplama yapan algoritması ile çalışma şeklinin otomatik olarak değişebilme yeteneğidir.

Şekil 4.1'de kullanıcıdan alan büyüklüğü girdisinin alındığı fonksiyonun küçük bir kısmı olan ekran çıktısını kontrol eden kod parçacığının içeriği görülmektedir.

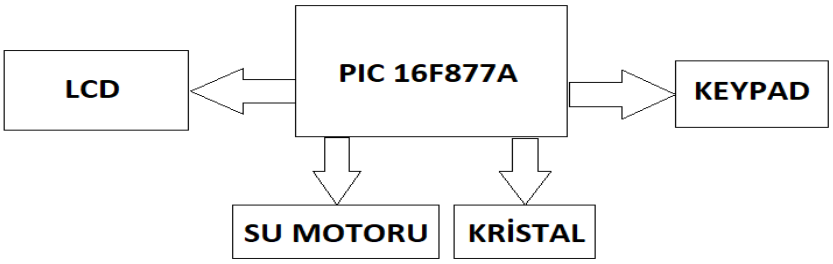
```
void FieldMod()
{
retryFieldMod:
lcd_gotoxy(1,1);
printf(lcd_putc,"\fMETREKARE(1-100)");
lcd_gotoxy(21,2);
printf(lcd_putc, "DEVAM: *");
delay_ms(5);
```

Şekil 4.1. Alan büyüklüğü kod parçacığı.

4.2. Sistem İçerisinde Geliştirilen Özellikler

Bu çalışmada eklenmesi gereken özellikler hakkında araştırmalar tamamlandıktan sonra çeşitli mantıksal ve matematiksel hesaplamalar yapılarak, algoritmaların oluşturulması ile geliştirme kısmına geçilerek uygulamaya konulmuştur.

Sistem ilk çalıştırıldığı anda, sulanacak alanın metrekare cinsinden büyüklüğü, tekrar frekansının değeri, maksimum çalışılacak gün sayısı, mevcut saat bilgisi ve sistemin çalışma saati bilgilerinin istendiği bir dizi geçişli ekran çıkmaktadır. Bu ekranlarda girdi ve hata kontrolü birlikte yapılmaktadır. Bu sayede yanlış veya hatalı bir girdi olması durumunda bir sonraki ekrana geçiş engellenmiş olmakla birlikte, bütün girdi değerleri alınmadan ana ekrana geçiş ve sistemin çalışmaya başlamasının önüne geçilmiş olmaktadır. Sisteme enerji verilmeden önce devre bağlantıları şekil 4.2'de belirtilen şekilde yapılmalıdır.



Şekil 4.2. Kontrol devresi blok gösterimi.

Tezdeki sisteme benzer şekilde çalışmakta olan, PIC mikrodenetleyici ile bahçe sulama işlemi gerçekleştiren sistemler mevcuttur. 2007 yılında Çakır ve Çalış tarafından gerçekleştirilen, otomatik biçimde çalışan sulama sistemi tasarımı mevcut sistemlerin incelenebilmesi için yararlı bir örnek olma özelliği taşımaktadır. Gerçekleştirdikleri çalışmada PIC 16F877 mikrodenetleyicisi üzerinde çalışarak sistemin kontrolünü uzak noktadaki bir kumanda aracılığıyla sağlamışlardır. Sistem, telefon hattı üzerinden elle veya otomatik sulama yapacak şekilde geliştirilmiştir. Sistemin otomatik çalışma şekli, toprağa yerleştirilen nem sensörleri sayesinde topraktaki nem değerinin belirlenen değer altına düşmesiyle birlikte sulama motorunun çalışmaya başlaması ve istenen nem değerine geldiğinde durması şeklindedir. Tasarlanan bu sistemin otomatik çalışma şekli zaman ve enerjiden tasarruf edilmesini

sağlamıştır.

Tezin uygulanış kısmını mevcut sistemlerden ayıran ve geliştirme kapsamında olan 3 ana özellik bulunmaktadır.

Bu özelliklerden ilki, sistemin sabit zaman aralığı içinde başla/ bitir mantığı ile değil, kullanıcı tarafından belirlenen zaman aralıklarına uygun olarak dönemsel şekilde çalışabilme yeteneğidir. Mevcut sistemlerde zaman ayarlı olarak çalışma işlevi bulunmasına rağmen, bu işlem başlangıç ve bitiş zamanlarının girdi olarak alınmasından sonra, sabit gün tekrarı ile sonsuz döngü şeklinde gerçekleşmektedir. Yapılan çalışmada ise mevcut sistemlerden farklı olarak, mevcut zaman ve istenen başlangıç zamanı girdi olarak alınmakla birlikte, sistemin hangi frekansta çalıştırılmak istendiği ve en uzun çalışma gün sayısı da girdi olarak istenmektedir. Bu demektir ki, girilen başlangıç zamanında çalışmaya başlayan sistem, tekrar girdisine göre belirlenen periyodlarla sürekli bir şekilde çalışmaktadır (Şekil 4.3).

```
Input the fieldSize
Input the frequency
Input the maxDay
Input the currentHour
Input the startingHour
Print out main screen message
    If currentHour is equal to startingHour
        Set MotorOn to true
        Initialize SystemOnTime to zero
        Increment SystemOnTime according to time passed
        Call Timer(arguments: fieldSize)
            Return(FinishTime)
                If(SystemOnTime>=FinishTime)
            Set MotorOn to false
            Endif
     Else
        Print out system idle message
        Set MotorOn to false
    Endif
```

Şekil 4.3. Çalışma başlangıcı belirleme algoritması.

Tezin benzer sistemlerden bir diğer farkı, sulanacak alanın büyüklüğüne göre sistemin çalışma süresinin değişkenlik gösterebilme özelliğinin olmasıdır. Sisteme bağlı olan motorun çalışma süresi, alanın büyüklüğüne göre değişmektedir. Bu demektir ki, sistem sabit bir çalışma mantığına bağlı tek bir basit algoritma ile değil, birbiri ile

bağlantılı ve birden fazla operasyonun sıralı çalışma mantığı ile birleşmesi sonucu, parametre verilerine göre değişken bir çıktı üretebilme kapasitesine sahiptir. Bunun sonucu olarak etkili bir çalışma dinamiği sağlanmıştır.

Sistem birden fazla fonksiyonun tek bir yerden çağrılarak aynı anda koşturulması ile, her biri kendi görevini yapıp çıktı verdikten sonra bu çıktılara göre işlem yaparak çalışmaktadır (Şekil 4.4).

```
void main()
{
   kbd_init();
   port_b_pullups(true);
   set_tris_b(0xF0);

   lcd_init();

   FieldMod();

   RepeatDayMod();

   MaxDayMod();

   CurrentHourMod();

   startHourMod();
```

Şekil 4.4. Sistemi oluşturan fonksiyonların gösterimi.

Araştırma konusu olan üçüncü özellik, sistem çıktısının çalıştırdığı motorun sabit bir hızda değil, “PWM” özelliği kullanılarak değişken hızlarda çalışabilmesini mümkün kılmak olmuştur. Böylelikle kullanılan motor sürekli tam kapasitede sabit bir hızla çalıştırılmak yerine, en ideal hızda çalışacak şekilde tasarlanarak güç tasarrufu elde edilmesi sağlanmıştır.

Sistemin istenen görevleri eksiksiz bir şekilde yerine getirmesi, derleyici içine gömülü kütüphanelerin programa dahil edilmesi ile mümkün olmaktadır. Bu kütüphaneler sayesinde mikrodenetleyici üzerinde hangi pinlerin aktif edilmesi gerektiği belirlenmektedir. Ayrıca ekran üzerindeki pin giriş ve çıkışlarının hangi görevleri yapmaları gerektiği atanmaktadır. Kullanıcı girdisini mümkün kılan klavyenin tuşlarına

karşılık gelen pin bağlantıları, kütüphane dahil edilmesi sırasında belirlenmektedir (Şekil 4.5).

```
#use delay(clock=4000000)
#fuses XT, NOWDT, NOPUT, NOLVP, NOCPD, NOPROTECT, NODEBUG, NOBROWNOUT, NOWRT
#include <kbd1.c>

#define LCD_RS_PIN PIN_D1
#define LCD_RW_PIN PIN_D2
#define LCD_ENABLE_PIN PIN_D0
#define LCD_DATA4 PIN_D4
#define LCD_DATA5 PIN_D5
#define LCD_DATA6 PIN_D6
#define LCD_DATA7 PIN_D7
#include <lcd.c>


#define sut1 pin_b4
#define sut2 pin_b5
#define sut3 pin_b6
#define sut4 pin_b7

#define sat1 pin_b0
#define sat2 pin_b1
#define sat3 pin_b2
#define sat4 pin_b3

#define ENABLE PIN_C4
#define ENABLE PIN_C5
#use fast_io(c)
```

Şekil 4.5. Kullanılan kütüphaneler.

Algoritmaların geliştirilme mantığı ve çalışma şekli ile ilgili detaylı bilgiye, tez raporunun çalışma mantığı ve kod açıklamalarıyla ilgili kısımlarından ulaşmak mümkündür.

4.3. PICOBS Özellikleri

İhtiyaca göre şekillenen sulama gereksinimleri, belirli zaman aralıklarında ve kullanıcı tarafından belirlenen süre boyunca yapılacak sulama sistemini ortaya çıkarmıştır. Bu gereksinimlerden hareketle tasarlanan PICOBS, PIC 16F877A mikrodenetleyici kullanılarak geliştirilmiştir.

PICOBS, geliştirme süreci boyunca kullanıcı odaklı olması hedef alınarak, en yüksek fayda ve düşük güç tüketimi konularında avantaj sağlayacak şekilde tasarlanmıştır. Kullanımı oldukça kolay, boyutları küçük ve montajı basittir. Bütün fonksiyonlar ve değişkenler İngilizce olarak tanımlanmıştır. Kullanıcı ara yüzü için ise Türkçe dili seçilmiştir. Kodlar, CSS C derleyicisi kullanılarak yazılmıştır.

Simülasyon ortamı, Proteus programında tasarlanmış ve gerçekleştirilmiştir.

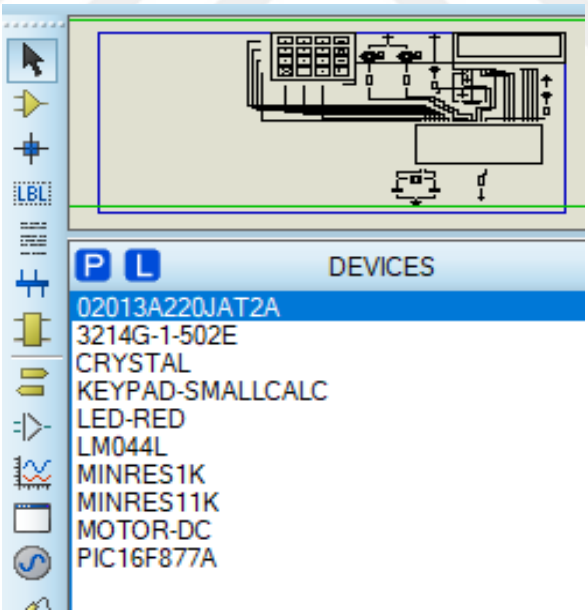
Aynı anda tek bir kullanıcının işlem yapmasına olanak vermektedir. Mikrodenetleyici hafızası, oluşturulan görevi işleyebilecek ve kaydedebilecek düzeydedir. Programın çalışma şekli kullanıcı tarafından belirlenmektedir. Belirlenen zamanda çalışma, belirlenen süre boyunca motoru çalıştırma, açık kalma periyodu gibi özellikler kullanıcı ara yüzü menüsünden seçilmektedir. Programın yazılım mantığı ve geliştirme özellikleri sayesinde işlemci üzerindeki yük azalmıştır ve daha az RAM ve ROM bellek tüketimi mümkün kılınmıştır.

Aşağıdaki bölümlerde PICOBS özelliklerinin nasıl geliştirildiği ve kullanılabileceği açıklanmıştır.

4.4. Proteus Simülasyon Ortamı İçinde PICOBS

4.4.1. Elektronik Bileşenlerin Seçimi

Programın simülasyon ortamında gerçekte yürüttüğü özelliklerin görülebilmesi ve test edilmesi amacıyla oluşturulan sistemin içindeki elektronik devre bileşenleri birbirleri ile uyumlu çalışacak şekilde seçilmiştir (Şekil 4.6).



Şekil 4.6. Elektronik devre elemanları.

Seçilmiş olan elektronik bileşenler şu şekildedir;

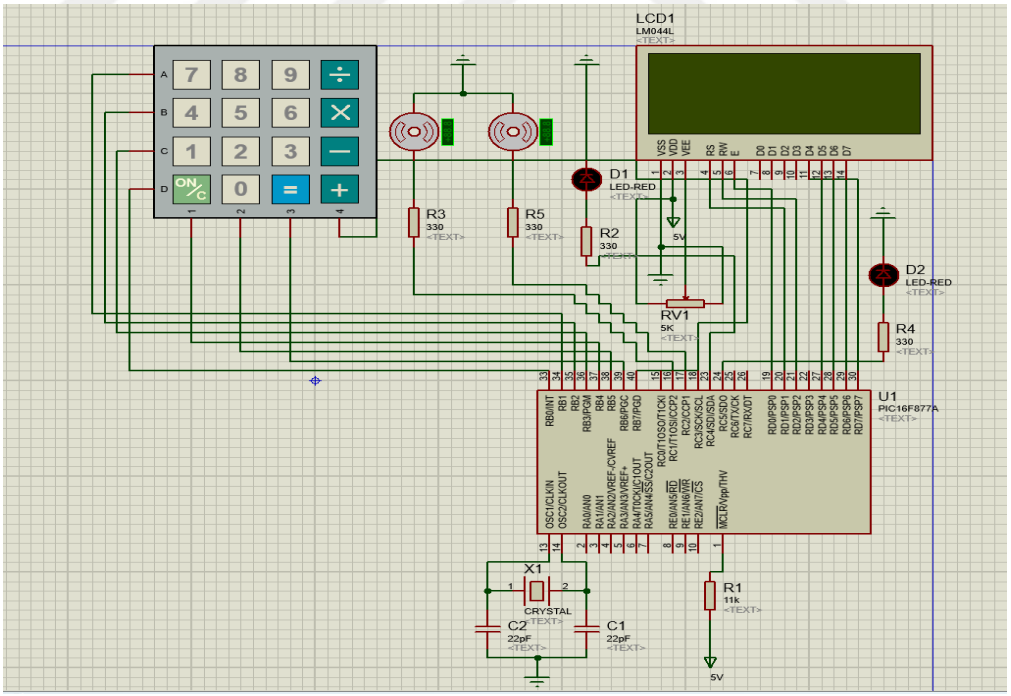
- DC Motor
- Led- Red

- Keypad – SmallCalc
- Crystal
- 16F877A Mikrodenetleyici
- LM044L LCD
- Güç Kaynağı
- 1K ve 11K Dirençler

4.4.2. Devre Elemanlarının Birleşimi

Simülasyonun eksiksiz ve hatasız bir şekilde çalıştırılabilmesi amacıyla elektronik devreleri PCB üzerine hatlar çekilerek uygun bir biçimde bağlanmıştır. Devre elemanları arasında oluşabilecek uyumsuzluklar, elektronik prensipleri kullanılarak giderilmiştir.

Mikrodenetleyici B portuna karşılık gelen pin uçlarına, kullanıcı tarafından bilgi girilmesine olanak veren tuş takımı bağlanmıştır. Mikrodenetleyici C portuna karşılık gelen pin uçlarına, motor kontrolünü sağlayan devre elemanları bağlanmıştır. Yönergelerin olduğu ve verinin gösterildiği ekran ise D portundaki pin uçlarına bağlanmaktadır (Şekil 4.7).



Şekil 4.7. Simülasyon devresi.

Mikrodenetleyicinin frekans değerinin ayarlanması osilatör sayesinde gerçekleşmektedir. 22 pF değerinde iki kondansatör kullanılarak tasarlanan osilatör 20 mHz değerinde çalışmaktadır.

4.4.3. Simülasyon Devre Çalışma Mantığı

Simülasyon devresinde giriş bilgisi tuş takımından gelen veri ile sağlanmaktadır. Tuş takımının doğru veriyi iletebilmesi, 4x4 tuş takımı ayarlama kütüphanesi sayesinde gerçekleşmektedir. Bu kütüphane, CCS – C derleyicisi içinden erişim verilerek programa dahil edilmektedir.

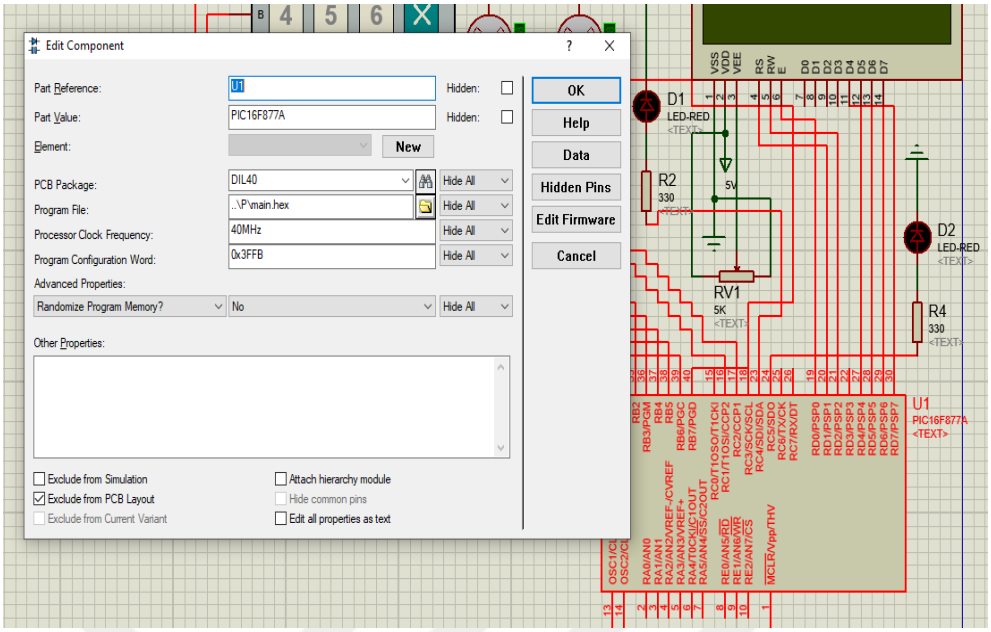
Kullanıcıya sürekli bilgi ve programın hangi kısımda olduğunun verisinin verilmesi LCD ekran sayesinde gerçekleşmektedir. 4x20 tipinde bir ekran tercih edilerek, programın 4 ana başlık altında bilgi sunması sağlanmıştır.

Sulamayı gerçekleştirecek motorun komut alması mikrodenetleyicinin çıkış pin uçlarından gelen sinyaller ile mümkün olmaktadır. Aynı şekilde motordan gelen ve motorun durum bilgisine vermekte olan sinyaller ise mikrodenetleyicinin giriş pin uçları ile toplanmaktadır.

4.4.4. Kodun Simülasyon Ortamına Entegresi

Mikrodenetleyicinin, C programlama dili ile yazılan ve simülasyonun nasıl davranacağını belirleyen kodlara erişimi bir hex dosyasının simülasyon ortamına dahil edilmesi ile mümkün olmaktadır.

Program kodu CCS- C derleyicisi içinde hatasız bir şekilde derlenebilirse, program dosyasının çıktısının bulunduğu klasörde bir hex dosyası oluşmaktadır. Bu hex dosyası, Proteus simülatörünün içinde mikrodenetleyici üzerine çift tıklama yapıldıktan sonra, bulunduğu yerdeki dosya yolu belirtilerek simülatöre dahil edilmektedir (Şekil 4.8).



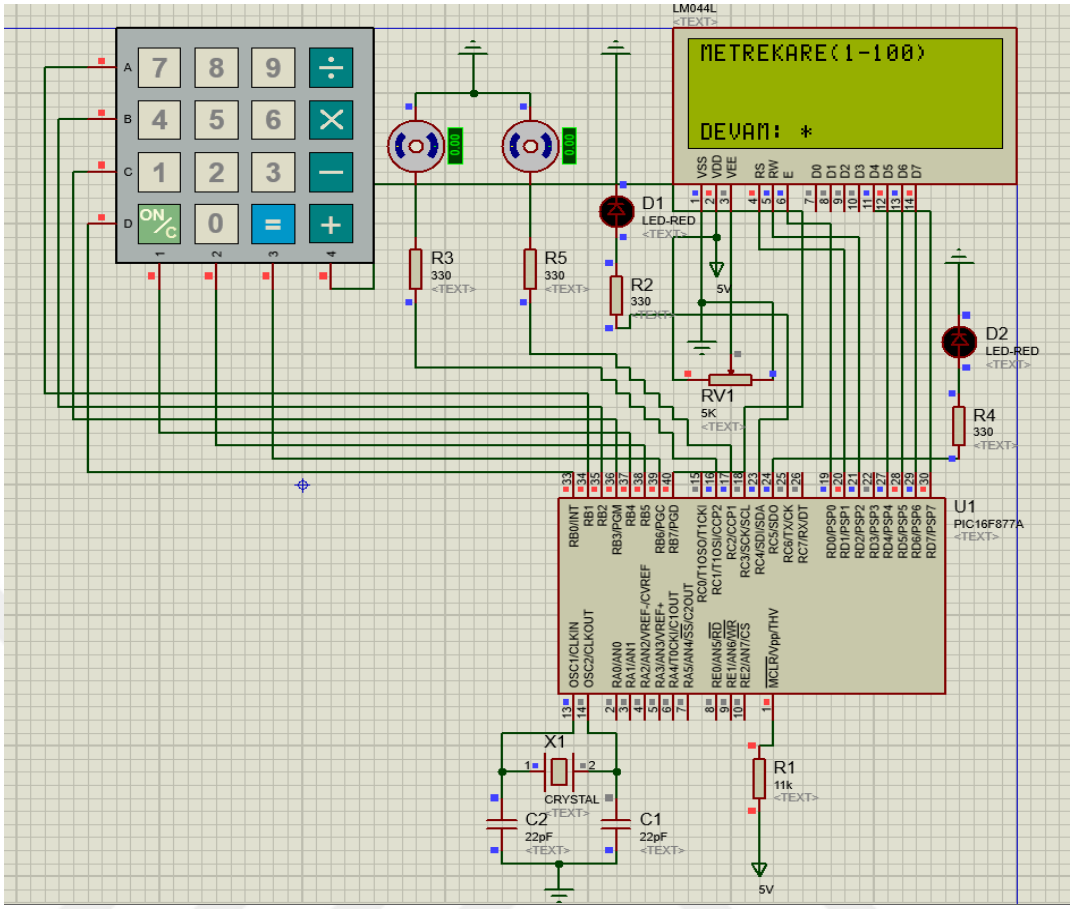
Şekil 4.8. Program entegresi.

4.4.5. Ekran Tasarımları, Geçişleri ve Kullanıcı Veri Girişi

Programın çalışabilmesi ve kod parçacıklarında belirlenen mantığa göre işlemesi için gereken veri kullanıcı tarafından sağlanmalıdır. Bu durum programın kullanıcı ihtiyaçlarına göre şekillenebilecek, esnek ve değiştirilebilir bir ürün ortaya çıkarma amacıyla tasarlanmıştır. Kullanıcının girdiği verilerin kontrolü aynı ekran üzerinde iken sağlanmaktadır ve hatalı girdi durumunda sonraki ekrana geçişe izin verilmemektedir.

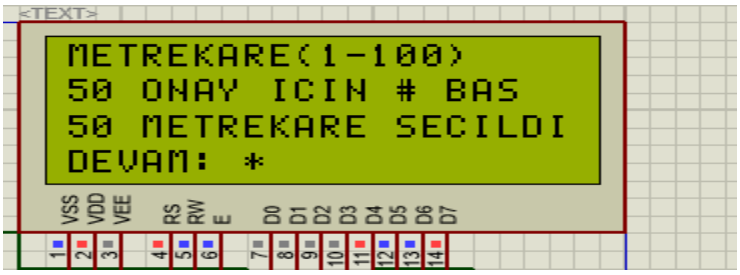
4.4.5.1. Alan Büyüklüğü Belirleme Ekranı

Sistem çalıştırıldıktan sonra karşılaşılan ilk ekranda, kullanıcıdan ne kadar büyüklükte bir alana sulama yapılacağı bilgisinin girilmesi beklenmektedir (Şekil 4.9).



Şekil 4.9. Alan bilgisi giriş ekranı.

Kullanıcı tarafından belirtilen alanın büyüklüğüne göre, sulama süresi programda belirtilen mantığa uygun bir şekilde arka planda hesaplanmaktadır. Kullanıcının girebileceği alan sınırları önceden belirlenmiştir. Sınırlar dahilinde olmayan bir giriş yapıldığı takdirde ekranda uyarı yazısı belirmekte ve yeniden giriş yapılması istenmektedir. Sınırlar dahilinde doğru bir giriş yapıldıktan sonra ekranda belirtilen yönerge tuşları ile sonraki ekranlara geçiş yapılabilmektedir (Şekil 4.10).

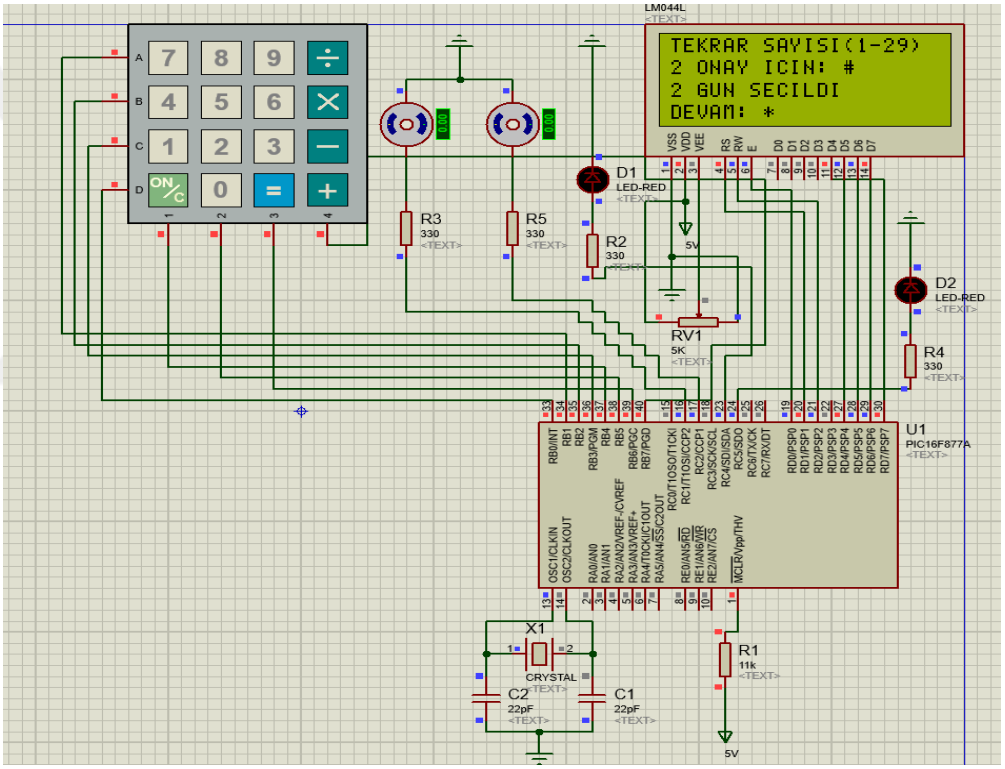


Şekil 4.10. Alan ekran geçişi.

4.4.5.2. Tekrar Sayısı Belirleme Ekranı

Programın bir diğer özelliği olan belirlenen sayıda tekrar özelliği sayesinde kullanıcıdan alınan bilgiye göre, belirlenen gün sayısı kadar süre boyunca program aynı işlevleri yerine getirmekle yükümlü olmaktadır.

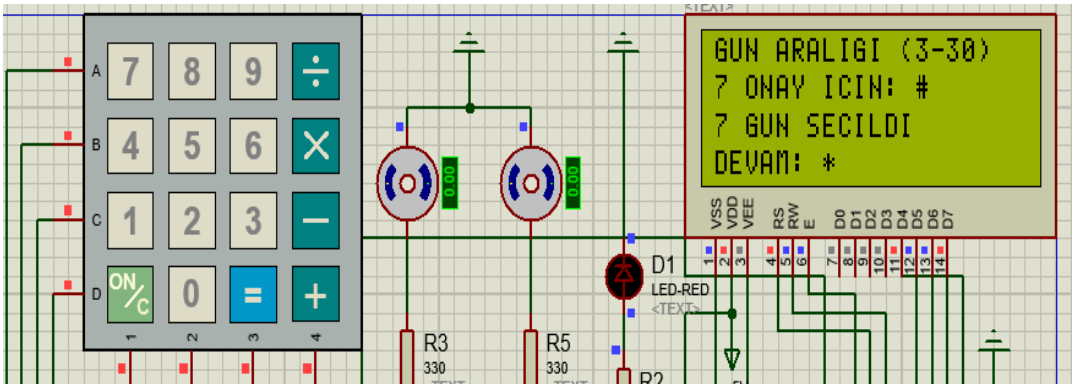
Örneğin, kullanıcı tarafından tekrar sayısı olarak 2 girildiği takdirde; program her 2 günde bir motoru çalıştırarak sulama yapacaktır. Her gün girişi sonrasında fazladan bir gün işleme katıldığından dolayı en büyük tekrar giriş sayısı 29 olarak belirlenmiştir (Şekil 4.11).



Şekil 4.11. Tekrar sayısı ekranı.

4.4.5.3. Toplam Çalışma Süresi Belirleme Ekranı

Sistem, toplam çalışma süresi 30 gün olacak şekilde tasarlanmıştır. Sistem açık unutulduğu takdirde veya istenen durumdan daha fazla çalışmasının önüne geçilmesi amacıyla bu şekilde bir tasarım mantığı belirlenmiştir.



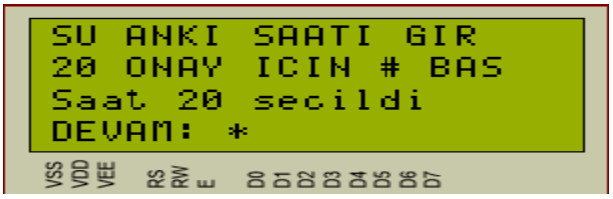
Şekil 4.12. Gün aralığı ekranı.

Gün aralığı, diğer ekranlar için açıklanan mantık ile paralel bir şekilde işlemektedir. Kullanıcıdan belirli bir aralıkta, programın çalışacağı maksimum sürenin girilmesi istenecek şekilde tasarlanmıştır (Şekil 4.12).

4.4.5.4. Mevcut Saat Belirleme Ekranı

Sistemin çalışma mantığı, mevcut saatin kullanıcı tarafından girilmesine dayanmaktadır. Mevcut saat girildikten sonra, motoru çalıştırılacağı saatin girilmesi gereken ekrana geçiş yapılacaktır. Bu da demektir ki mevcut saat ve çalışmaya başlama saati eşit duruma geldiğinde sistem çalışmaya başlamaktadır.

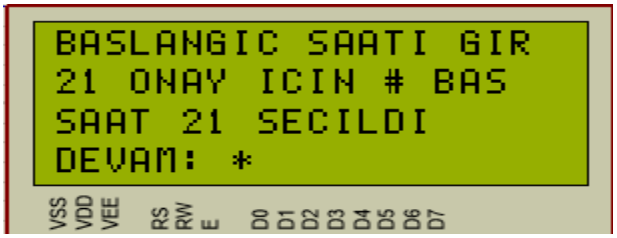
Mevcut saatin kullanıcı tarafından girilmesi, doğru ve hatasız çalışma bakımından bir sorun teşkil etmemektedir çünkü program kod parçacıkları içerisinde saatler 24 saat olarak tanımlanmakla birlikte bir döngü şeklinde çalışmaktadır. Bu da demektir ki sistem her durumda kullanıcının belirlediği zaman aralıklarında, belirtilen zaman farkları ile çalışacaktır (Şekil 4.13).



Şekil 4.13. Mevcut saat ekranı.

4.4.5.5. Çalışma Saati Belirleme Ekranı

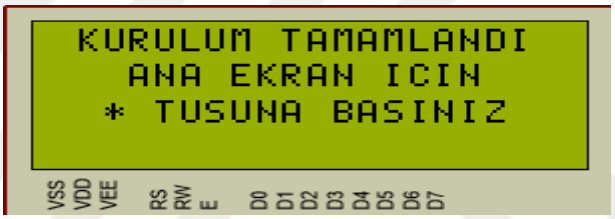
Mevcut saat girişi yapıldıktan sonra, kullanıcıdan sistemin çalışmaya başlayacağı saat değerinin girişinin yapıldığı ekrana geçiş sağlanacaktır (Şekil 4.14).



Şekil 4.14. Başlangıç saati ekranı.

Örneğin, kullanıcı tarafından mevcut saatin 20, başlangıç saatinin 21 olarak girildiği bir durumda sistem bir saat sonra motoru çalıştıracak ve başta girilmiş alan büyüklüğü ile doğru orantılı olacak şekilde bir süre boyunca açık tuttuktan sonra kapatacaktır.

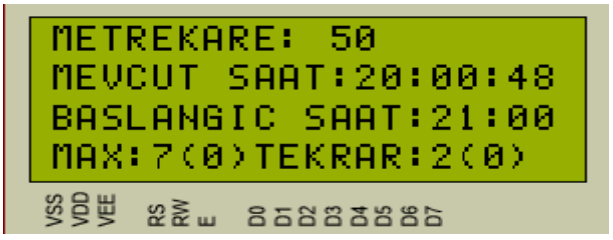
Bütün değerler hatasız girildikten sonra kurulum tamamlandı ekranına geçiş yapılmaktadır (Şekil 4.15).



Şekil 4.15. Kurulum tamamlandı ekranı.

4.4.5.6. Ana Ekran

Kullanıcı girdilerinin kontrolü her ekranda girdi yapıldığı anda gerçekleştiğinden, ana ekrana geçiş yapıldığında girdi kaynaklı bir hata ile karşılaşılmasının önüne geçilmiş olmaktadır. Ana ekranda önceden girilen bilgilerin tümü gösterilmekle birlikte; ilerleyen zaman, sistemin kaç gündür açık olduğu ve kaçıncı tekrarda olduğu bilgisi de gösterilmektedir (Şekil 4.16).



Şekil 4.16. Ana ekran.

4.5. PICOBS Çalışma Mantığı

Sistem kendi içerisinde frekansa bağlı çalışmakta olan bir zamanlayıcı barındırmaktadır. Bu zamanlayıcı, her bir program döngüsünde bir değişken içerisindeki değeri arttırmaktadır. Böylece artan bu değişken değerine göre saniye, dakika ve saatler hesaplanmaktadır. Hesaplanan zamana göre gerçek zamanlıya benzer bir zamanlayıcı elde edilmektedir.

Kullanıcıdan alınan veriler ile, mevcut zaman ile sistemin çalışması istenen zaman eşitlendiği anda motor çalışmaktadır. Motorun çalışma süresi kullanıcı tarafından verilmiş olan alan büyüklüğüne göre hesaplanmaktadır.

Sistem, belirlenen periyodlar ile belirlenmiş toplam süre boyunca açık kaldıktan sonra bir uyarı mesajı ile kendini kapatmaktadır.

BÖLÜM 5

SONUÇLAR VE YAPILACAK ÇALIŞMALAR

5.1. Sonuçlar

Bu çalışmada; PIC 16F877A mikrodenetleyici türü ile bahçe sulama sistemi olarak kullanılmak üzere, kullanıcı isteklerine göre şekillenerek, istenilen zaman aralıklarında ve süre boyunca esnek yapıda çalışmakta olan, az yer kaplayan, kullanıcı ara yüzü Türkçe olarak tanımlanmış kullanımı kolay bir sistem tasarlanmıştır. Bu sistem, yapılan çalışmalar sonrasında çeşitli ünite testlerine konularak kullanıcı girdisi alma, çoklu görev yürütme, gerçek zaman kontrolü, periyodik olarak çalışma gibi özellikleri bu testler ile doğrulanmıştır. Sistemin testlerinin simülasyon ortamında gerçekleştirilmesi, gerçek hayatta nasıl tepkiler ortaya konulacağı açısından fikirler vermiştir. Kontrol edici olarak mikrodenetleyici kullanılması sayesinde maliyet oldukça aza indirilmekle birlikte güç tüketimi açısından da oldukça faydalı bir seçim yapılmıştır. PICOBS, PIC 16F877A mikrodenetleyicisi ile aynı yazmaç yapısına sahip olan ailelerin hepsinde kullanılabilmektedir.

Performans ile ilgili yapılan deneylerde PICOBS tepki süresinin 3 us olarak oldukça düşük ve beklenen düzeyin altında olduğu tespit edilmiştir. Görevler arası geçişler ise 430 komut çevirimine karşılık gelmektedir. Ölçümler 20 Mhz kristal osilatör ile yapılmış olup PIC için komut işletim süresi harici frekansın dörtte birine karşılık geldiğinden komut 0.5 us sürede işletilmektedir. Sistemde bulunan kristal osilatörün değeri 20 Mhz olduğundan dolayı, ölçümler bu osilatör değeri üzerinden yapılmıştır ve komut işletilme süresi 0.5 us olarak ölçülmüştür.

5.2. Yapılacak Çalışmalar

Tasarlanmış olan sulama sisteminin gerçekleştirdiği fonksiyonlar eksiksiz ve hatasız bir biçimde tamamlanmış olmakla birlikte, bu sistemde yer almayan fakat

üzerine geliştirme yapılarak sistemi bir üst noktaya taşıyabilecek mesaj ile durum kontrol bilgisi ve güneş enerjisi ile çalışma özellikleri bu çalışmada ele alınmamıştır. Gelecek çalışmalarda güneş enerji paneli ve devresi eklenmesi, kısa mesaj ile bildirim gönderilmesi özelliklerinin PICOBS içerisine eklenmesi ile birlikte kullanılabilirliğin arttırılması ve sistemin yer ve enerji gibi bağımlılıklardan kurtarılması yapılabilecek çalışmalar arasında sayılabilmektedir.

KAYNAKLAR

Alparslan, N., Tanık, A., Dölgen, D. (2008). *Türkiye'de Su Yönetiminin Durumu: Sorunlar ve Öneriler*. Türk Sanayicileri ve İş Adamları Derneği, İstanbul.

Bakibillah, A.S.M., Rahman, N., Zaman, Md.A.U. (2014). *Microcontroller based Closed Loop Speed Control of DC Motor using PWM Technique*. Doi:10.5120/18979-0402.

Balım, M.A. (2015). 2 Mayıs 2021 tarihinde https://www.elektrikport.com/teknik-kutuphane/mikrodenetleyiciler-nasil-calisir-1-bolum/14785#ad-image-0 adresinden erişildi.

Balım, M.A. (2018). 4 Mayıs 2021 tarihinde https://www.alperbalim.com/mikrodenetleyici-programlamaya-baslamak/ adresinden erişildi.

Bodur, M. (2014). 16 Ocak 2021 tarihinde https://www.emo.org.tr/ekler/d9b816ba072629f_ek.pdf adresinden erişildi.

Çakır, A., Çalış, H. (2007). *Uzaktan Kontrollü Otomatik Sulama Sistemi Tasarımı ve Uygulaması*. Süleyman Demirel Üniversitesi, Fen Bilimleri Enstitüsü Dergisi, 11(3), 258-261.

Çotuk, H. (2008). 15 Şubat tarihinde http://www.earsiv.etu.edu.tr/bitstream/handle/20.500.11851/351/TZ00023.pdf?sequence=1&isAllowed=y adresinden erişildi.

Elprocus. (2014). 25 Nisan 2021 tarihinde https://www.elprocus.com/pic-microcontroller-programming-using-c-language/ adresinden erişildi.

Exploreembeded. 25 Şubat 2021 tarihinde https://www.exploreembedded.com/wiki/PIC16f877a_Timer adresinden erişildi.

Fidan, U., Karasekreter, N. (2011). *GSM/SMS Tabanlı Sulama Otomasyon Kontrol Biriminin Geliştirilmesi ve Uygulaması*. E-Journal of Nem Sciences Academy, 6(1), 71-77.

Gupta, S. (2018). 12 Nisan 2021 tarihinde https://circuitdigest.com/microcontroller-projects/4x4-keypad-interfacing-with-pic16f877a adresinden erişildi.

Hermansson, H., Lundblad, L. (2019). *Automatic Irrigation System For Plants*. TRITA-ITM-EX 2019:57.

İlker, Ü. (2013). 4 Mayıs 2021 tarihinde https://www.ilkerunal.com/wp-content/uploads/2013/11/teorik.pdf adresinden erişildi.

Karaca, O. (2014). 29 Nisan 2021 tarihinde http://www.oguzozkaraca.com/mikrodenetleyici-nedir/ adresinden erişildi.

Kırnak, H. (2006). *Automatıc Irrigatıon Based on Soil Moisture for Nursery Crops*. GAP V. Mühendislik Kongresi, Şanlıurfa, 1540-1547.

Koç, A., Güner, Ü. (2005). Mevcut Sulama Projelerinin FAO Kriterleriyle Yeniden Değerlendirilmesi: Tavas Ovası Örneği. Dumlupınar Üniversitesi Fen Bilimleri Enstitüsü Dergisi, 009, 93-106.

Microchip. 27 Nisan 2021 tarihinde http://www.microchip.com, adresinden erişildi.

Öter, A., Bahar, M.Ş. (2018). *Programlanabilir Denetleyici Kontrollü Sulama Sistemi*. Doi:21(4):329-333.

Özkan, E. (2018, Kasım 28). 28 Nisan 2021 tarihinde https://www.muhendisbeyinler.net/mikrodenetleyici-nedir-ne-ise-yarar/ adresinden erişildi.

PIC. (2005). Wikipedia, Özgür Ansiklopedi içinde. 27 Mart 2021 tarihinde http://tr.wikipedia.org/wiki/PIC adresinden erişildi.

PIC Microcontroller. (2017). 26 Nisan 2021 tarihinde https://pic-microcontroller.com/how-to-simulate-pic-microcontroller-in-proteus-design-suite-8/ adresinden erişildi.

Robotiksistem. (2009). 10 Ocak 2021 tarihinde https://www.robotiksistem.com/mikrodenetleyici_nedir_pic_ozellikleri.html adresinden erişildi.

Robotiksistem. (2009). 11 Ocak 2021 tarihinde http://www.robotiksistem.com/pic_mikrodenetleyiciler.html adresinden erişildi.

Robotiksistem. (2009). 15 Ocak 2021 tarihinde http://www.robotiksistem.com/pic16f877_mikrodenetleyici_ozellikleri.html adresinden erişildi.

Roboturka. (2015). 15 Mart 2021 tarihinde https://roboturka.com/pic-assembly-pic-c/bellek-kullanimina-gore-mikroislemci-mimarileri/ adresinden erişildi.

Semiz, T.Y. (2018). 4 Mayıs 2021 tarihinde https://maker.robotistan.com/mikrodenetleyici-mikroislemci/ adresinden erişildi.

Simple Circuit. (2016). 18 Mart 2021 tarihinde https://simple-circuit.com/pic16f877a-

ds1307-alarm-ccs-c/ adresinden erişildi.

Simple Circuit. (2016). 11 Nisan 2021 tarihinde https://simple-circuit.com/pic16f877a-pwm-adc-ccs-c/ adresinden erişildi.

Tekinel, O., Kanber, R., Çetin, M. (2000). *Su Kaynaklarının Geliştirme ve Kullanımı*. TMMOB Ziraat Mühendisliği Odası Türkiye Ziraat Mühendisleri V. Teknik Kongresi, Ankara, 17-21.

Yang, S., Lu, P., Okushima, L., Sase, S. (2004). *Precision Irrigation System Based on Detection of Crop Water Stress with Acoustic Emission Technique*. Proceedings of the 2004 International Conference on Information Acquisition ICIA 2004, Hefei, China, 444-447.

EKLER

EK A: PICOBS PROGRAM FONKSİYONLARI

Timer_1

Osilatör değerine bağlı olarak programın çalışma frekansına göre gerçek zamanı hesaplamayı sağlayan zamanlayıcı fonksiyondur. Bu fonksiyon ile frekans değerine göre saniye ve dakikalar hesaplanarak, mikrodenetleyicinin zamanı kendi içinde hesaplayabilme özelliği kazandırılarak dışarıya bağımlılığının önüne geçilmektedir. Saat/ sayaç PIC16'larda ve PIC18'lerde 16 bittir. Saymayı gerçekleştirir ve ayrıca taşmada bir kesinti sağlamaktadır. Mevcut seçenekler her cihazda farklıdır ve cihaz başlık dosyasında listelenmiştir.

FieldMod

Motorun ne kadar süre çalıştırılacağı hesaplamasında kullanılacak olan alan büyüklüğü bilgisini işleyen fonksiyondur. Kullanıcı tarafından girilen alan bilgisi bu fonksiyon ile motor çalışma süresine dönüştürülmektedir.

CurrentHourMod

Mevcut saat ile ilgili tüm işlemlerin gerçekleştirildiği fonksiyondur.

StartHourMod

Sistemin çalışmaya başlayacağı saat ile ilgili tüm işlemlerin gerçekleştirdiği fonksiyondur.

RepeatDayMod

Sistemin hangi periyodlar ile çalışacağını belirleyen ilgili işlemlerin yer aldığı fonksiyondur.

MaxDayMod

Sistemin açık kalacağı toplam süre ile ilgili operasyonların tanımlandığı fonksiyondur.

MainMenu

Sistem çalışmaya başladıktan sonra güncel bilgilerin kullanıcıya gösterilmesi ve bilgilendirme yapılması ile ilgili işlemlerin gerçekleştirildiği fonksiyondur.

Main

Bütün fonksiyonların kontrol edildiği, ekran geçişlerinin hangi sıra ile yapılması gerektiğinin belirlendiği, sıralamanın belirlendiği ve programın işleyişinin başlangıç yeri olan ana fonksiyondur.

EK B: KAYNAK KODLARI

Main.c Dosyası

```
#include <16f877A.h>
#use delay(clock=4000000)
#fuses XT, NOWDT, NOPUT, NOLVP, NOCPD, NOPROTECT, NODEBUG,
NOBROWNOUT, NOWRT
#include <kbd1.c>

#define LCD_RS_PIN PIN_D1
#define LCD_RW_PIN PIN_D2
#define LCD_ENABLE_PIN PIN_D0
#define LCD_DATA4 PIN_D4
#define LCD_DATA5 PIN_D5
#define LCD_DATA6 PIN_D6
#define LCD_DATA7 PIN_D7
#include <lcd.c>

#define sut1 pin_b4
#define sut2 pin_b5
#define sut3 pin_b6
#define sut4 pin_b7

#define sat1 pin_b0
#define sat2 pin_b1
#define sat3 pin_b2
#define sat4 pin_b3

#define ENABLE PIN_C4
#define ENABLE PIN_C5
```

```
#use fast_io(c)

int16 A=0;
int Y=0;
int Z=0;

int B=0;
int R=0;
int currentHourControl=0;

int M=0;
int N=0;
int startHourControl=0;

int16 C=0;
int F=0;
int G=0;

int Q=0;
int W=0;
int E=0;

int U=0;

const char mainMenuState = '*';
boolean daysDone = false;

  char K;
  char T;

boolean fieldModExit = false;
boolean currentHourModExit = false;
```

```
boolean startHourModExit = false;
boolean repeatDayModExit = false;
boolean maxDayModExit = false;

int32 cont = 0;
int contSec=0;

int sec=0, nowMin=0, hour=0;

int hStart=0;
const int min=0;
int32 feedingMin=0;

const int minPerField = 1;
int16 fieldNumber = 3;
const int maxFieldNumber = 100;
int32 finishTime = 0;

boolean Motor = false;
boolean startFeeding = false;

int day=0;
int repeatingDayCount = 0;

int16 repeatingDay = 1;
int maxDay = 2;

boolean closeSystem = false;
boolean motorRun = false;

boolean startWaitingTime = false;
int countWaitingTime = 0;
```

```
int32 waitingSec = 0, waitingMin = 0;
const int waitingTimeFinish = 60;//60 mins

boolean reverseMotor = false;
int countReverse = 0;
int32 reverseSec = 0, reverseMin=0;
int reverseMotorFinish = 0;

boolean equalHours = false;

#int_timer1
void timer_1()
{
set_timer1(60506);

if((fieldNumber>0) && (repeatingDay>0) && (maxDay>0))
{
contSec++;
}
if(contSec == 50)
{
sec++;
contSec=0;
}
if(sec==60)
{
nowMin++;
sec=0;
}
if(nowMin==60)
{
hour++;
```

```
nowMin=0;
}
if(hour==24)
{
day++;
repeatingDayCount++;
hour=0;
}

if(repeatingDayCount == repeatingDay)
{
repeatingDayCount=0;
feedingMin=0;
}
if(day==maxDay)
{
closeSystem = true;
daysDone = true;
}

if(hStart == hour)
{
startFeeding = true;
equalHours = true;
}
if((repeatingDayCount == 0) && (closeSystem == false) && (startFeeding == true))
{
cont++;
motorRun = true;
}
if(cont == 3000)
{
```

```
feedingMin++;
cont=0;
}
if((motorRun) && (repeatingDayCount == 0)
{
 motor = true;
}
if(motor == true)
{
output_high(pin_c4);
startWaitingTime = false
}
if((feedingMin >= finishTime) ||  (repeatingDayCount != 0)  || (closeSystem == true))
{
motor = false;
motorRun = false;
startFeeding = false;
cont=0;
startWaitingTime = true;
}
if(motor == false)
{
output_low(pin_c4);
}
if(repeatingDay == repeatingDayCount)
{
feedingMin = 0;
}
if(feedingMin == 0)
{
reverseMin = 0;
}
```

```
}

void FieldMod()
{
retryFieldMod:
lcd_gotoxy(1,1);
printf(lcd_putc,"\fMETREKARE(1-100)");
lcd_gotoxy(21,2);
printf(lcd_putc, "DEVAM: *");
delay_ms(5);
Z=1;
do
{
K = kbd_getc();
T = K - 48;
if((K!=0) && (K!='A') && (K!='B') && (K!='C') && (K!='D')&& (K != '*') && (K !=
'#'))
{
A = (A*10) + T;
Y++;
lcd_gotoxy(Y,2);
printf(lcd_putc,"%d ONAY ICIN # BAS",T);
delay_ms(5);
}
if(K=='#')
{
fieldNumber = A;
if((A>=1) && (A<=maxFieldNumber))
{
lcd_gotoxy(21,1);
printf(lcd_putc,"%ld METREKARE SECILDI", fieldNumber);
finishTime = minPerField * fieldNumber;
```

```
reverseMotorFinish = (minPerField * fieldNumber)/3;
delay_ms(5);
}
else
{
lcd_gotoxy(21,2);
printf(lcd_putc,"YANLIS SAYI, LIMIT 100");
Y=0;
A=0;
delay_ms(5);
goto retryFieldMod;
}
}
 if((A>=1) && (A<=maxFieldNumber)){
if(K == '#')
{
fieldModExit = true;
}
}
if((K=='*') && (fieldModExit==true))
{
Y=0;
A=0;
Z=0;
}
}while(Z==1);
}
void CurrentHourMod()
{
retryCurrentHourMod:
currentHourControl=1;
lcd_gotoxy(1,1);
```

```
printf(lcd_putc,"\fSU ANKI SAATI GIR");
lcd_gotoxy(21,2);
printf(lcd_putc, "DEVAM: *");
do
{
K = kbd_getc();
T = K - 48;
if((K!=0) && (K!='A') && (K!='B') && (K!='C') && (K!='D')&& (K != '*') && (K
!='#'))
{
B = (B*10) + T;
R++;
lcd_gotoxy(R,2);
printf(lcd_putc,"%d ONAY ICIN # BAS",T);
delay_ms(5);
}
if(K=='#')
{
if((B>=0) && (B<=24))
{
hour = B;
lcd_gotoxy(21,1);
printf(lcd_putc,"Saat %d secildi",hour);
delay_ms(5);
}
else
{
lcd_gotoxy(21,2);
printf(lcd_putc,"YANLIS SAAT");
B=0;
R=0;
delay_ms(5);
```

```
goto retryCurrentHourMod;
}
}
 if((B>=0) && (B<=24))
 {
 if(K == '#')
{
 CurrentHourModExit = true;
}
}
if((K=='*') && (CurrentHourModExit==true))
{
R=0;
B=0;
currentHourControl=0;
}
}while(currentHourControl==1);
}
void startHourMod()
{
retryStartHourMod:
startHourControl=1;
lcd_gotoxy(1,1);
printf(lcd_putc,"\fBASLANGIC SAATI GIR\n");
lcd_gotoxy(21,2);
printf(lcd_putc, "DEVAM: *");
do
{
K = kbd_getc();
T = K - 48;
if((K!=0) && (K!='A') && (K!='B') && (K!='C') && (K!='D')&& (K != '*') && (K !=
'#'))
```

```
{
M = (M*10) + T;
N++;
lcd_gotoxy(N,2);
printf(lcd_putc,"%d ONAY ICIN # BAS",T);
delay_ms(5);
}

if(K=='#')
{
if((M>=0) && (M<=24))
{
hStart = M;
lcd_gotoxy(21,1);
printf(lcd_putc,"SAAT %d SECILDI",hStart);
delay_ms(100);
}
else
{
lcd_gotoxy(21,2);
printf(lcd_putc,"YANLIS SAAT");
M=0;
N=0;
delay_ms(5);
goto retryStartHourMod;
}
}
  if((M>=0) && (M<=24))
  {
if(K == '#'){
startHourModExit = true;
}
```

```
}
if((K=='*') && (startHourModExit==true))
{
 M=0;
N=0;
startHourControl=0;

lcd_gotoxy(1,1);
printf(lcd_putc,"\f KURULUM TAMAMLANDI");
lcd_gotoxy(1,2);
printf(lcd_putc,"   ANA EKRAN ICIN");
lcd_gotoxy(21,1);
printf(lcd_putc,"  * TUSUNA BASINIZ");
}
}while(startHourControl==1);
}
void RepeatDayMod()
{
retryRepeatDayMod:
lcd_gotoxy(1,1);
printf(lcd_putc,"\fTEKRAR SAYISI(1-29)");
lcd_gotoxy(21,2);
printf(lcd_putc, "DEVAM: *");
delay_ms(5);
G=1;
do
{
K = kbd_getc();
T = K - 48;
if((K!=0) && (K!='A') && (K!='B') && (K!='C') && (K!='D')&& (K != '*') && (K
!='#'))
{
```

```
C = (C*10) + T;
F++;
lcd_gotoxy(F,2);
printf(lcd_putc,"%d ONAY ICIN: #",T);
delay_ms(5);
}
if(K=='#')
{
if((C>=1) && (C<=29))
{
repeatingDay = C;
lcd_gotoxy(21,1);
printf(lcd_putc,"%ld GUN SECILDI", repeatingDay);
delay_ms(5);
}
else
{
lcd_gotoxy(21,2);
printf(lcd_putc,"YANLIS GUN, 0-30 GIR");
C=0;
F=0;
delay_ms(5);
goto retryRepeatDayMod;
}
}
if((C>=1) && (C<=29))
{
 if(K == '#')
{
 repeatDayModExit = true;
}}
if((K=='*') && (repeatDayModExit==true))
```

```
{
F=0;
C=0;
G=0;
}
}while(G==1);
}

void MaxDayMod()
{
retryMaxDayMod:
lcd_gotoxy(1,1);
int maxDayBottomLimit = repeatingDay+1;
printf(lcd_putc,"\fGUN ARALIGI (%d-30)",maxDayBottomLimit);
lcd_gotoxy(21,2);
printf(lcd_putc, "DEVAM: *");
delay_ms(5);
E=1;
do
{
K = kbd_getc();
T = K - 48;
if((K!=0) && (K!='A') && (K!='B') && (K!='C') && (K!='D')&& (K != '*') && (K
!='#'))
{
Q = (Q*10) + T;
W++;
lcd_gotoxy(W,2);
printf(lcd_putc,"%d ONAY ICIN: #",T);
delay_ms(5);
}
if(K=='#')
```

```
{
if((Q>=maxDayBottomLimit) && (Q<=30))
{
maxDay = Q;
if((Q%repeatingDay) == 0 )
{
maxDay = maxDay + 1;
lcd_gotoxy(21,1);
printf(lcd_putc,"%d+1=%d GUN SECILDI", (maxDay-1), maxDay);
delay_ms(5);
}else
{
lcd_gotoxy(21,1);
printf(lcd_putc,"%d GUN SECILDI", maxDay);
delay_ms(5);
}
}
else
{
lcd_gotoxy(1,1);
printf(lcd_putc,"/fYANLIS GUN GIRDINIZ");
Q=0;
W=0;
delay_ms(5);
goto retryMaxDayMod;
}
}
  if((Q>=maxDayBottomLimit) && (Q<=30))
{
if(K == '#')
{
maxDayModExit = true;
```

```
}
}
if((K=='*') && (maxDayModExit==true))
{
Q=0;
W=0;
E=0;
}
}while(E==1);
}
void MainMenu()
{
 lcd_gotoxy(1,1);
 printf(lcd_putc,"\fMETREKARE: %ld",fieldNumber);
 delay_ms(100);
do
{
lcd_gotoxy(1,2);
printf(lcd_putc,"MEVCUT SAAT:%02d:%02d:%02d",hour,nowMin,sec);
lcd_gotoxy(21,1);
printf(lcd_putc,"BASLANGIC SAAT:%02d:%02d",hStart,min);
lcd_gotoxy(21,2);

printf(lcd_putc,"MAX:%d(%d)TEKRAR:%ld(%d)",maxDay,day,repeatingDay,repeatin
gDayCount);

if(daysDone == true)
{
printf(lcd_putc,"\fSistem Kapandi");
delay_ms(3000);
return;
}
```

```
}while(U==1);
}
void main()
{
kbd_init();
port_b_pullups(true);
set_tris_b(0xF0);
lcd_init();
FieldMod();
RepeatDayMod();
MaxDayMod();
CurrentHourMod();
startHourMod();
while(true)
{
K = kbd_getc();
T = K - 48;
set_tris_c(0x0F);
output_c(0x00);
enable_interrupts(GLOBAL);
setup_timer_2(T2_DISABLED,0,1);
setup_timer_1(T1_INTERNAL | T1_DIV_BY_4);
set_timer1(60506);
enable_interrupts(int_timer1);
switch(K)
{
case mainMenuState:
U=1;
MainMenu();
break;
}
}
```

}

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı : ÇALIŞIR, Recep

Uyruğu : T.C.

Medeni hali : Bekar

Eğitim

Derece	**Eğitim Birimi**	**Mezuniyet tarihi**
Lisans	Ankara Üniversitesi/Bilgisayar Mühendisliği	2016
Lise	Beşiktaş Anadolu Lisesi/Sayısal	2010

İş **Deneyimi**

Yıl	**Yer**	**Görev**
2021-	Siemens	Yazılım Geliştirme Mühendisi
2019- 2021	Ronwell Digital	Ar-Ge Mühendisi
2018 – 2019	Technology Partners Enerji	Otomasyon Mühendisi

Yabancı Dil

İngilizce