**KOCAELİ ÜNİVERSİTESİ * FEN BİLİMLERİ ENSTİTÜSÜ**

# KABLOSUZ ALGILAYICI/EYLEYİCİ AĞLARLA DENETİM SİSTEMİ TASARIMI

**YÜKSEK LİSANS TEZİ**

**Faruk AKTAŞ**

**Anabilim Dalı: Elektronik ve Bilgisayar Eğitimi**

**Danışman: Doç. Dr. Celal ÇEKEN**

**KOCAELİ, 2012**

**KOCAELİ ÜNİVERSİTESİ * FEN BİLİMLERİ ENSTİTÜSÜ**

# KABLOSUZ ALGILAYICI/EYLEYİCİ AĞLARLA DENETİM SİSTEMİ TASARIMI

**YÜKSEK LİSANS TEZİ**

**Faruk AKTAŞ**

**Tezin Enstitüye Verildiği Tarih: 19 Aralık 2011**

**Tezin Savunulduğu Tarih: 13 Ocak 2012**

**Tez Danışmanı**

**Doç. Dr. Celal ÇEKEN**

(.....)

**Üye**

**Prof. Dr. Kadir ERKAN**

(.....)

**Üye**

**Yrd. Doç. Dr. Ali ÇALHAN**

(.....)

**KOCAELİ, 2012**

## ÖNSÖZ VE TEŞEKKÜR

Son yıllarda bilgi kaynaklarının giderek artmasıyla verilerin toplanması, analiz edilmesi ve saklanması büyük önem kazanmıştır. Teknolojideki son gelişmeler, fiziksel dünyayı gözlemleme yeteneğine sahip, veri işleyebilen, karar verme tabanlı uygun işlemleri gerçekleştirebilen dağıtılmış kablosuz algılayıcı ve eyleyici ağların ortaya çıkmasına yol açmıştır.

Yüksek lisans eğitimim süresince değerli birikimlerini benimle paylaşan, tezimin her aşamasında sorunlarımı dinleyerek, çalışmalarıma yön veren ve yoğun akademik yaşamında değerli zamanını her türlü problemimi çözmeye ayıran tez danışmanım saygıdeğer hocam Doç. Dr. Celal ÇEKEN'e, haftalık rutin toplantılarda tez çalışmama yol gösteren hocalarım, sayın Prof. Dr. Kadir ERKAN ve sayın Doç. Dr. Mehmet YILDIRIM'a, değerli görüşleri ile tez çalışmama katkıda bulunan sayın Ahmet KIZILHAN'a, tez çalışmasının başından sonuna kadar manevi desteklerini benden esirgemeyen başta oda arkadaşım sayın Arş. Gör. Alper KARAHAN ve tüm iş arkadaşlarıma ve tüm NCS grubuna teşekkürlerimi sunarım.

Bugünlere gelmemi sağlayan anneme, babama ve kardeşlerime saygı, sevgi ve sonsuz teşekkürler.

**İÇİNDEKİLER**

## ŞEKİLLER DİZİNİ

## TABLOLAR DİZİNİ

## SİMGELER

| | |
|---|---|
| A | : Kabul sinyali |
| Ah | : Amper saat |
| $^{o}C$ | : Santigrat derece |
| F(s) | : Çıkış Büyüklüğü (Laplace domeni) |
| K | : Kazanç |
| Kb | : Kilo bayt |
| kbit | : Kilobit |
| Kbps | : Saniyede iletilen kilobit sayısı |
| L | : Ölü zaman (sn) |
| mA | : Mili amper |
| Mbit | : Megabit |
| MHz | : Megahertz |
| mV | : Milivolt |
| P | : Stop bitiş durumu ($I^2C$) |
| pf | : Pikofarad |
| R | : Direnç (ohm) |
| R | : Read ($I^2C$ okuma) |
| $R_f$ | : Referans direnci (ohm) |
| RF | : Radyo frekansı |
| Rx | : Sinyal alım frekansı |
| s | : Laplace değişkeni |
| S | : Start başlama durumu ($I^2C$) |
| Sr | : Tekrarlı start durumu ($I^2C$) |
| T | : Zaman sabiti (sn) |
| Tx | : Sinyal gönderim frekansı |
| V | : Volt |
| $V_{CC}$ | : Besleme gerilimi (volt) |
| $V_{EE}$ | : Besleme gerilimi (volt) |
| $V_{out}$ | : Çıkış gerilimi (volt) |
| $V_{ref(+)}$ | : Pozitif referans voltajı |
| $V_{ref(-)}$ | : Negatif referans voltajı |
| W | : Read ($I^2C$ yazma) |
| X(s) | : Giriş büyüklüğü (Laplace domeni) |

**Kısaltmalar**

| | |
|---|---|
| ACK | :Acknowledge (Kabul) |
| ADC | :Analog-Digital Converter (Analog-Dijital Çevirici) |
| CDMA | :Code Division Multiple Access (Kod Bölmeli Çoklu Erişim - KBÇE) |
| CMOS | :Complementary Metal-Oxide Silicon (Tamamlayıcı Metal-Oksit Silikon) |
| DAC | :Digital Analog Converter (Dijital Analog Çevirici) |
| DAQ | :Data Acquisition Board (Veri Edinim Bordu) |
| DPM | :Dynamic Power Management (Dinamik Güç Yönetimi) |
| DVS | :Dynamic Voltage Scaling (Dinamik Voltaj Ölçeklendirme) |
| EEPROM | :Electrically Erasable Programmable Read Only Memory (Elektrik ile Yazılıp Silinebilen Sadece Okunabilen Bellek) |
| GA | :Gerilim Aralığı |
| GAS | :Gerilim Alt Sınır |
| GG | :Gelen Gerilim |
| GPRS | :General Packet Radio Services (Genel Paket Radyo Servisi) |
| GUI | :Graphic User Interface (Grafik Kullanıcı Arayüzü) |
| $I^2C$ | :Inter-Integrated Circuit (Entegre Arası Cihaz) |
| LSB | :Least Significant Bit (En Düşük Değerlikli Bit) |
| MATLAB | :Matrix Laboratory (Matris Laboratuvarı) |
| MSB | :Most Significant Bit (En Yüksek Değerlikli Bit) |
| NACK | :Non-acknowledge (Kabul hayır) |
| NCS | :Networked Control Systems (Ağ Kontrol Sistemi – AKS) |
| NMOS | :N Channel Metal-Oxide Silicon (N Kanal Metal-Oksit Silikon) |
| ÖS | :Ölçülen Sıcaklık |
| PERL | :Practical Extraction and Report Language (Pratik Çıkarım ve Raporlama Dili) |
| PWM | :Puls Width Modulation (Darbe Genişlik Modülasyonu) |
| RAM | :Random Accsess Memory (Rastgele Erişilebilir Bellek) |
| RISC | :Reduced Instruction Set Computer (Azaltılmış Komut Seti) |
| SA | :Sıcaklık Aralığı |
| SCL | :Serial Clock (Seri Saat) |
| SDA | :Serial Data (Seri Veri) |
| SFR | :Special Function Register (Özel İşlem Yazmacı) |
| SPI | :Serial Peripheral Interface (Seri Çevresel Arayüz) |
| SÜS | :Sıcaklık Üst Sınırı |
| TTL | :Transistor-Transistor Logic (Transistör-Transistör Lojik) |
| USART | :Universal Asynchronous Receiver/Transmitter (Evrensel asenkron Alıcı/Verici) |
| WSAN | :Wireless Sensor and Actuator Networks (Kablosuz Algılayıcı ve Eyleyici Ağlar – KAEA ) |
| WSN | :Wireless Sensor Networks (Kablosuz Algılayıcı Ağlar – KAA) |
| XML | :Extensible Markup Language (Uzatılabilir İşaretleme Dili) |
| XMLRPC | :Extensible Markup Language Remote Procedure Calls (Uzatılabilir İşaretmele Dili Uzaktan Yordam Çağrısı) |

**KABLOSUZ ALGILAYICI/EYLEYİCİ AĞLARLA DENETİM SİSTEMİ TASARIMI**

**Faruk AKTAŞ**

**Anahtar Kelimeler:** Ağ Tabanlı Denetim, Kablosuz Algılayıcı ve Eyleyici Ağlar, Ölü Zamanlı Sistem

**Özet:** Kablosuz Algılayıcı Ağ (KAA) uygulamaları endüstriyel, askeri, medikal ve çevresel alanlarda özellikle ortam izleme ve görüntüleme amaçlı olarak kullanılmaktadır. Bu uygulama alanlarının yanı sıra, özellikle son zamanlarda, KAA'ların denetim amaçlı olarak kullanımı giderek yaygınlaşmaktadır. Bu çalışmada, elektronik sistemlerin denetiminde kullanılmak üzere KAA altyapısını kullanan bir test düzeneği fiziksel olarak gerçekleştirilmiştir. Ayrıca, geliştirilen test düzeneğinin çalışmasını doğrulamak üzere ölü zamanlı bir sistemin denetimi değişik çalışma koşulları altında test edilmiştir. Geliştirilen uygulamada, ilk olarak, denetimi gerçekleştirilen sistem üzerindeki algılayıcıdan gelen sıcaklık bilgisi mikrodenetleyici tarafından alınarak analog-dijital çevirici ile sayısal bilgiye dönüştürülmektedir. Sistemin çalışmasını istediğimiz sıcaklık bilgisi (referans büyüklüğü) ise bilgisayar yardımıyla kablosuz olarak MDA320 veri edinim borduna gönderilmektedir. Veri edinim bordu gelen sıcaklık bilgisini $I^2C$ çıkışları üzerinden mikrodenetleyiciye göndermektedir. Ölçülen sıcaklık bilgisi ile referans bilgisi, denetim işlemini gerçekleştiren mikrodenetleyici tarafından değerlendirilerek denetim bilgisi üretilmekte ve mikrodenetleyicinin çıkışlarına gönderilmektedir. Mikroişlemcinin çıkışına bağlı bulunan sistem ise gelen denetim sinyali uyarınca çalışmasını değiştirmektedir. Yapılan çalışmalar sonucunda kablosuz algılayıcı ağların ortam izleme ve görüntüleme işlevlerinin yanı sıra denetim amaçlı olarak da kullanılabileceği gösterilmiştir.

**WIRELESS SENSOR/ACTUATOR NETWORKS CONTROL SYSTEM DESIGN**

**Faruk AKTAŞ**

**Keywords**: Networked Control, Wireless Sensor and Actuator Networks, Dead Time System

**Abstract**: Wireless Sensor Network ( WSN ) applications commonly are available in industrial, medical, military, and environmental areas, especially for monitoring and tracking purposes. In addition to these widely used applications, recently, WSNs are also being employed for the control of the systems. In this study, a test bed which has a WSN structure and will be used in the control of the electronic systems is realized. In order to validate the test bed developed, a first order plus dead time process control system is tested in different working conditions. In the case study, first of all, the temperature data from the sensor deployed on the system controlled is converted to digital value by ADC of the microcontroller. The set point value is sent by PC to the MDA320 data acquisition board over the wireless medium. The acquisition board is then delivered the converted temperature value to the microcontroller using $I^2C$ bus. Measured temperature and set point values are evaluated and control information is generated considering these values by the microcontroller. The system connected to the output of the microcontroller is changed its working condition according to this control information generated. The results show that, the WSNs can be used efficiently for the control purpose in addition to its common applications, i.e. monitoring and tracking.

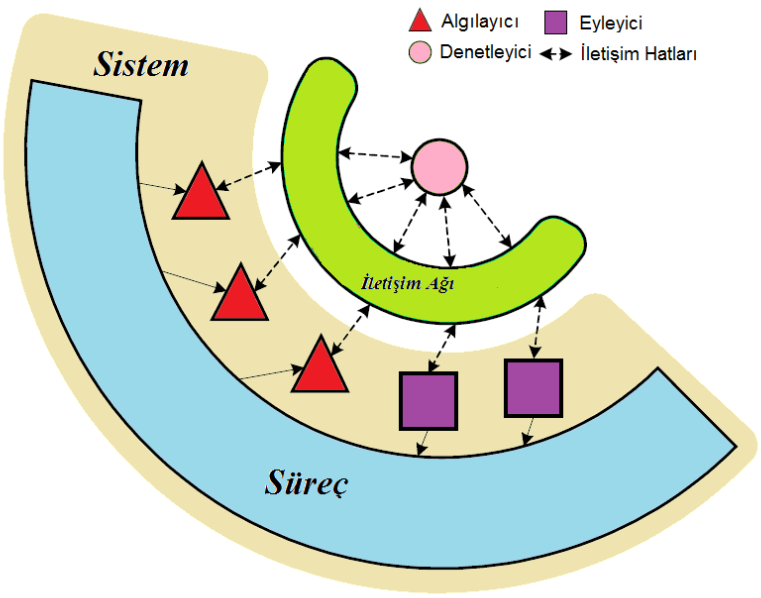
# 1. GİRİŞ

Mikroelektronik ve kablosuz haberleşme teknolojilerindeki gelişmeler küçük boyutlu, hareketli, düşük maliyetli, düşük enerji gereksinimli ve çok fonksiyonlu algılayıcı düğümlerin yaygın olarak kullanılmasına olanak sağlamıştır [1].

Kablosuz algılayıcı ağ (KAA); mikrodenetleyici, algılayıcı, kablosuz haberleşme arabirimi ve güç biriminden oluşan düğümlerin bir araya gelerek oluşturduğu ağ yapısıdır. KAA'daki düğümler, bulunduğu ortamdan topladığı fiziksel büyüklükleri merkezi bir erişim noktasına göndermektedir. Algılama verileri, çeşitli yapılar kullanılarak (çoklu atlama, tek atlama) kablosuz ortam üzerinden merkezi düğüme iletilir [2].

KAA'lar oldukça fazla uygulama potansiyeline sahiptir [2−3]. KAA'lar günümüzde çevresel, endüstriyel, askeri ve sağlık ile ilgili alanlarda özellikle ortam izleme amaçlı olarak kullanılmaktadırlar. KAA'ların ortam izleme amaçlı kullanımlarının yanı sıra denetim amaçlı olarak kullanılması da giderek yaygınlaşmaktadır.

Son yıllardaki iletişim teknolojilerindeki devrimsel denebilecek değişiklikler ile, gömülü veri işleme ve ağ tabanlı iletişim, kontrol sistemleri mühendisliğinde giderek önemli rol oynamaktadır. Gerçekleştirilen uygulamalarda, sisteme yerleştirilen ağ elemanlarının sayısı, genel amaçlı bilgisayarlardan (masaüstü ve dizüstü bilgisayar) çok daha fazladır. Bu devreler otomasyon, süreç denetimi ve izlenmesi gibi birçok uygulama alanında kullanılırlar. Bu bağlamda, gömülü ağ kontrol ve görüntüleme sistemleri, kısaca ağ kontrol sistemleri (AKS- Networked Control Systems (NCS)), eşi benzeri görülmemiş şekilde popüler olmuşlardır. Şekil 1.1'de ağ kontrol sistemi örneği görülmektedir [4].




Şekil 1.1: Ağ kontrol sistemi örneği

Şekilde görüldüğü gibi ağ kontrol sisteminde, düğümler/bileşenler onları bir araya getiren ortak bir ağ üzerinden iletişim kurabilirler. Böylece algılayıcılar ve denetleyiciler aralarında, denetleyiciler kendi aralarında, algılayıcılar kendi aralarında, denetleyiciler ve eyleyici düğümler kendi aralarında iletişim kurabilirler. Bu iletişimin amacı, denetim sisteminin başarımını arttırmaya yöneliktir. Başarım, optimal kontrol veya tahmin örneğinde olduğu gibi, bir başarım kriteri açısından tanımlanmış, ölçülebilir bir miktar veya istenilen bir davranış olarak açıklanan niteliksel bir ölçü olabilir. Ağ kontrol sisteminde her bir düğüm iletişimin yanı sıra karar verici olarak davranabilir ve kontrol yapabilir [4].

Bu çalışmada elektronik sistemlerin denetimini gerçekleştiren kablosuz algılayıcı/eyleyici ağ (KAEA) tabanlı bir test düzeneği geliştirilmiştir. Ayrıca, geliştirilen test düzeneğinin çalışmasını doğrulamak üzere ölü zamanlı bir sistemin denetimi değişik çalışma koşulları altında test edilmiştir. Geliştirilen uygulamada, ilk olarak, denetimi gerçekleştirilen sistem üzerindeki algılayıcıdan gelen sıcaklık bilgisi mikrodenetleyici tarafından alınarak analog-dijital çevirici ile sayısal bilgiye dönüştürülmektedir. Sistemin çalışmasını istediğimiz sıcaklık bilgisi (referans büyüklüğü) ise bilgisayar yardımıyla kablosuz olarak MDA320 veri edinim borduna

gönderilmektedir. Veri edinim bordu gelen sıcaklık bilgisini $I^2C$ çıkışları üzerinden mikrodenetleyiciye göndermektedir. Ölçülen sıcaklık bilgisi ile referans bilgisi, denetim işlemini gerçekleştiren mikrodenetleyici tarafından değerlendirilerek denetim bilgisi üretilmekte ve mikrodenetleyicinin çıkışlarına gönderilmektedir. Mikrodenetleyicinin çıkışına bağlı bulunan sistem ise gelen denetim sinyali uyarınca çalışmasını değiştirmektedir. Bu çalışma ile KAA'ların geleneksel kullanım alanlarının yanı sıra denetim amaçlı olarak ta kullanılabileceği gösterilmiştir. KAEA tabanlı bir denetim sistemi tasarımı gerçekleştirilmiştir.

## 1.1. Literatür Taraması

Akyıldız ve diğ. 2001'de, KAA kavramını tanımlayarak, kablosuz ağ tasarımını etkileyen faktörleri açıklamışlardır [2]. Bu çalışma KAA alanındaki en temel çalışmalardan biridir.

Yang Peng ve arkadaşları (2008), aktif bir yanardağ olan Helen Yanardağı'ndan gelen bilimsel verileri doğru olarak izleme amaçlı akıllı algılama sistemi tasarlamışlardır [6]. Tasarladıkları sistem ile çevrimiçi olarak algılayıcı sürücü hizmetleri, arıza tespiti, veri önceliklendirme ve zaman senkronizasyonu sağlamışlardır. Bu uygulama ile kaynak kullanımı, durum farkındalığı ve sistem başarımını iyileştirmişlerdir. Uygulamada tez çalışmasında kullanılan MDA320 veri edinim bordu ortam izleme amaçlı uygulamaya uygun olarak kullanılmıştır.

Jren-Chit Chin ve arkadaşları (2009), yapıların sağlamlık durumlarını gerçek zamanlı olarak izleme amaçlı algılayıcı ağ tasarlamışlardır. Bu çalışma binalardaki çatlakların tespit edilmesinde kullanılmıştır [7].

Kamran Khakpour ve M.H. Shenessa (2008), endüstriyel denetim sistemlerinde, ZigBee standardı kullanan KAA'ların başarımını gösteren bir çalışma yapmışlardır [8]. Yapılan çalışma özellikle dokuma fabrikalarındaki dokuma tezgahlarını izleme ile ilgili yapılmıştır. Çalışma sonucunda ZigBee ile Bluetooth ve Wi-Fi karşılaştırıldığında ZigBee iletişim protokolünün daha güvenilir, daha güçlü ve yüksek başarıma sahip olduğu görülmüştür.

Xiuhong Li ve diğ. (2006), seralar için uzaktan izleme sistemi önermişlerdir [9]. Veriler kablosuz modül GPRS-CDMA 4 üzerinden gerçek zamanlı olarak uzak sunucuya iletilebilmektedir.

Orazio Farrugia ve arkadaşları (2008), KAA kullanarak, deniz çevresindeki sınırlı bir alandaki su kalitesini izleme amaçlı sistem tasarlamışlardır. Sudaki tuzluluk oranı, pH değeri ve suyun sıcaklığı gibi değerler izlenmiştir [10]. Sistemde tezde kullanılan MDA320 veri edinim bordu ortam izleme amaçlı uygulamaya uygun olarak kullanılmıştır. Uygulamaya özel olarak tasarlanmış sistemde MDA320 veri edinim bordunun harici algılayıcı girişleri kullanılmıştır.

## 1.2. Tez Çalışmasının Amacı ve Başlatılma Sebepleri

Literatürde yapılan çalışmaların büyük çoğunluğu KAA'ların geleneksel uygulama alanlarından biri olan ortam izleme amaçlı olarak gerçekleştirilmiştir. Yapılan çalışmaların bazılarında, tez çalışmamızda kullanılan MDA320 veri edinim bordu ortam izleme amaçlı olarak kullanılmıştır.

KAA'ların geleneksel uygulama alanlarının yanı sıra denetim amaçlı olarak kullanılması ihtiyacı doğmuştur. Teknolojideki son gelişmeler, fiziksel dünyayı gözlemleme yeteneğine sahip, veri işleyebilen, karar verme tabanlı uygun işlemleri gerçekleştirebilen dağıtılmış kablosuz algılayıcı ve eyleyici ağların ortaya çıkmasına yol açmıştır. Elektronik sistemlerin uzaktan izlenmesinin yanı sıra denetlenmesi ihtiyacına yönelik olarak KAEA tabanlı bir test düzeneği fiziksel olarak gerçekleştirilmiştir.

Test düzeneğinin gerçekleştirilmesinde algılayıcı/eyleyici ağ elemanı olarak MDA320 veri edinim bordu kullanılmaktadır. MDA320 veri edinim bordunun analog girişleri kullanılarak algılama işlemi, PIC16F877 mikrodenetleyicisi ile haberleşerek ise ortak bir biçimde eyleyici işlemi gerçekleştirilmektedir. Eyleyici kısmın karar verme birimi PIC16F877 mikrodenetleyicisidir. Mikrodenetleyiciye denetim ile ilgili parametreleri gönderme işlemini MDA320 veri edinim bordu

gerçekleştirmektedir. Mikrodenetleyici ve MDA320 veri edinim bordu $I^2C$ haberleşme uçları ile birbirlerine bağlanmışlardır.

Çalışmamızın ana amacı, KAA'ların geleneksel kullanım alanlarının yanı sıra denetim amaçlı olarak da kullanılabilmesine yönelik olarak KAEA tabanlı bir test düzeneği geliştirmektir. Denetlenen sistem olan süreç denetim sisteminin ölü zaman parametrelerinin iyileştirilmesiyle ilgili bir çalışma yapılmamıştır.

### 1.3. Tez Çalışmasının Katkıları

Tez çalışmalarının katkıları dört ana başlık altında ifade edilebilir:

- Elektronik sistemlerin KAEA ile gözlemlenmesi ve denetlenmesini sağlayan bir test düzeneği geliştirilmiştir.
- KAA, izleme amaçlı kullanılmasının yanında denetim amaçlı olarak da kullanılmıştır.
- Algılayıcı ve eyleyici düğüm olarak MDA320 veri edinim bordu kullanılmıştır.
- MDA320 veri edinim bordu ve PIC16F877 cihazlarının $I^2C$ haberleşmesi ile iletişimi sağlanmıştır.

### 1.4. Tez Düzeni

Tez çalışmaları, beş ana bölümde sunulmaktadır;

Bölüm 2'de KAA'lar hakkında genel bilgiler verilmekte, düğümlerin mimarilerinden bahsedilmekte ve genel uygulama alanları belirtilmektedir.

Bölüm 3'te sistemde kullanılan yazılım ve donanım bileşenleri ile ilgili geniş bilgiler yer almaktadır.

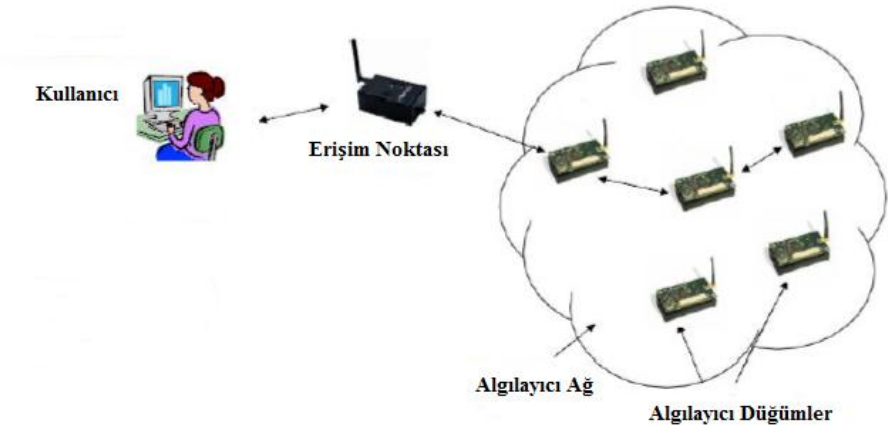
Bölüm 4'te ise gerçekleştirilen KAEA tabanlı denetim uygulaması anlatılmaktadır.

Yapılan tez çalışmalarının sonuçları ve katkıları son bölümde değerlendirilmektedir.

# 2. KABLOSUZ ALGILAYICI AĞLAR

## 2.1. Giriş

Mikroelektronik ve kablosuz haberleşme teknolojilerindeki gelişmeler küçük boyutlu, hareketli, düşük maliyetli, düşük enerji gereksinimli ve çok fonksiyonlu algılayıcı düğümlerin yaygın olarak kullanılmasına olanak sağlamıştır [1]. Algılayıcı düğümlerin verileri işlemek, çevreyi gözetlemek ve fiziksel dünya ile iletişimi sağlamak amacıyla kablosuz olarak haberleştiği ortama kablosuz algılayıcı ağlar (KAA) denilmektedir [11]. KAA'lar günümüzde çevresel, sağlık, askeri ve endüstriyel alanlarda özellikle ortam izleme amaçlı olarak kullanılmaktadır [2]. Şekil 2.1'de genel KAA yapısı görülmektedir.




Şekil 2.1: KAA yapısı

## 2.2. Kablosuz Algılayıcı ve Eyleyici Ağlar

Teknolojideki son gelişmeler, fiziksel dünyayı gözlemleme yeteneğine sahip, veri işleyebilen, karar verme tabanlı uygun işlemleri gerçekleştirebilen dağıtılmış kablosuz algılayıcı ve eyleyici ağların (KAEA – Wireless Sensor and Actuator

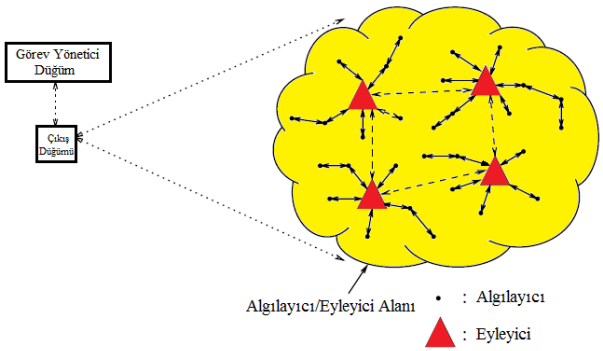
Networks (WSAN)) ortaya çıkmasına yol açmıştır. Bu ağlar, savaş gözleme, yapıların mikroiklimlendirme kontrolleri, nükleer, kimyasal ve biyolojik saldırı tespiti, ev otomasyonu ve çevresel izleme sistemlerinde kullanılırlar.

Örneğin bir yangın durumunda, yangın kontrol edilemez duruma gelmeden önce, algılayıcı röleler ile yangının tam yeri ve yoğunluğuna göre yağmurlama sistemi çalıştırılarak yangın kolayca söndürülebilir. Benzer bir şekilde, bir odadaki hareket ve ışık sensörleri insan varlığını algılarlar. Daha sonra, önceden belirlenmiş kullanıcı tercihlerine göre belirlenmiş eylemleri yürütmek için eyleyicilere komut gönderirler.

KAEA'larda algılama ve hareket işlemleri, sırasıyla algılayıcı ve eyleyici düğümler tarafından yapılır. Algılayıcılar, düşük maliyetli, sınırlı algılama ve hesaplama yeteneğine sahip, kablosuz iletişim yapabilen düşük güçlü devrelerdir. Eyleyiciler ise daha iyi işlem kapasitesi ve pil ömrüne sahip, daha yüksek güçte iletişim yapabilen düğümlerdir. Hedef alana yerleştirilen düğümlerin sayısı yüzlerce hatta binlerce olabilirken, daha yüksek yeteneklere sahip eyleyici düğümler için bu sayı söz konusu değildir. Çünkü eyleyici düğümler geniş alanda hareket edebilirler [5].

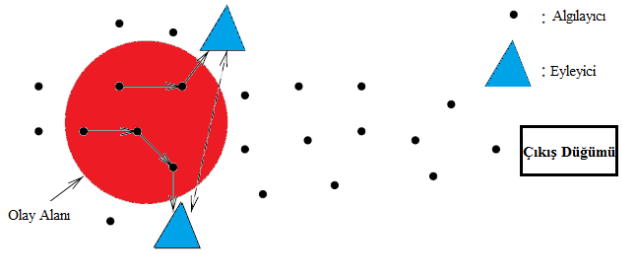
### 2.2.1. Kablosuz algılayıcı ve eyleyici ağların fiziksel karakteristikleri

KAEA'larda algılayıcı ve eyleyici düğümlerin rolleri sırasıyla, ortamdan verileri toplamak ve toplanan bu verilere göre uygun eylemleri gerçekleştirmektir. Şekil 2.2'de alana dağıtılmış, algılayıcı ve eyleyici düğümlerin çıkış düğümü ve yönetici düğümleri ile bağlantısını gösterir fiziksel mimari görülmektedir [5].

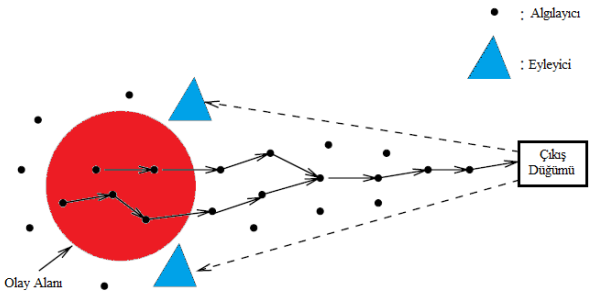



Şekil 2.2: KAEA fiziksel mimarisi

Fiziksel mimari ikiye ayrılmaktadır. Bunlar otomatik ve yarı otomatik diye adlandırılmış mimarilerdir. Otomatik mimari, merkezi denetleyici (çıkış düğümü) olmayışından dolayı bu şekilde adlandırılmıştır. Yarı otomatik mimaride ise veri toplayıp, işleyen bir merkezi denetleyici (çıkış düğümü) bulunmaktadır. Şekil 2.3'te otomatik, Şekil 2.4'te ise yarı otomatik mimariler görülmektedir [5].




Şekil 2.3: Otomatik mimari

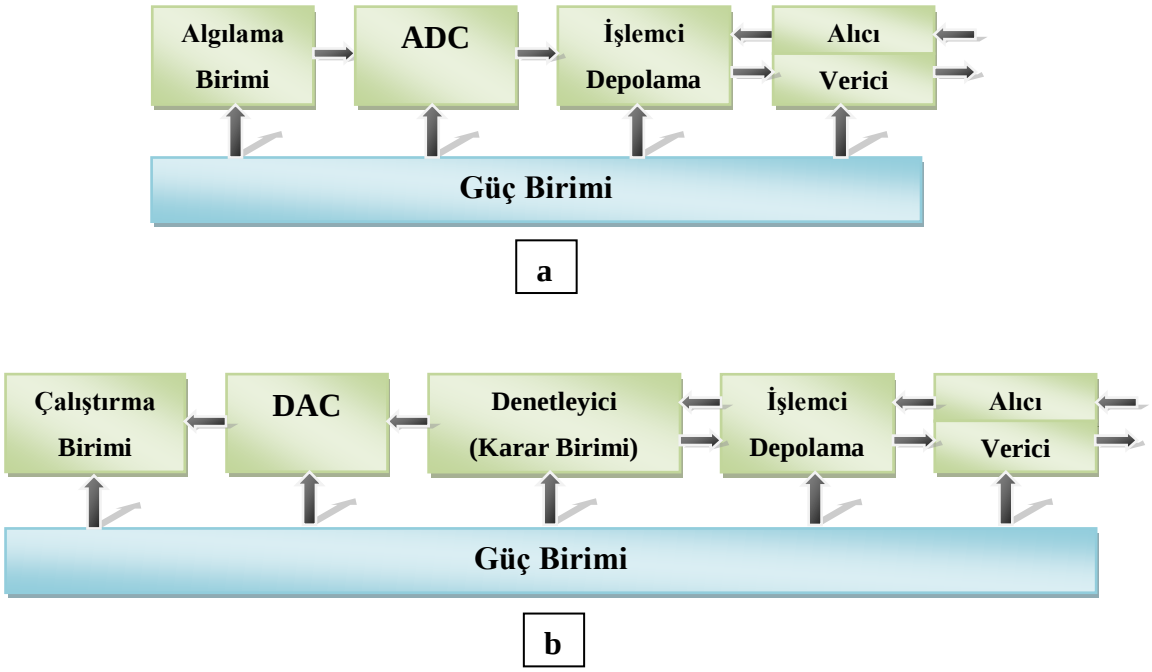



Şekil 2.4: Yarı otomatik mimari

Uygulama türüne bağlı olarak, bu mimarilerden yalnızca biri kullanılır. Yarı otomatik mimarinin avantajı, halen kullanılmakta olan kablosuz algılayıcı ağ uygulamalarına benzemesidir. Böylece iletişim yapma ve koordinasyonu için yeni algoritma ve protokoller geliştirilmesine gerek kalmamaktadır.

Otomatik mimaride eyleyici düğümler ve algılayıcı düğümler birbirlerine yakın oldukları için algılanan bilgi hemen aktarılabilmektedir. Bu durum bilgi gecikmesini en az düzeye indirmektedir. Yarı otomatik mimaride ise her nerede olay olursa, olay bilgisi düğümler üzerinden tek tek atlayarak merkezi düğüme gönderilmektedir.

Böylece düğümler üzerine aşırı bir yönlendirme külfeti binmektedir. Düğümlerin herhangi birinde arıza meydana gelmesi durumunda iletişim kaybolabilir ve ağ kullanılmaz hale gelebilir.




Şekil 2.5: Bileşenler a) Algılayıcılar b) Eyleyiciler

Şekil 2.5'de KAEA uygulamalarında kullanılan algılayıcı ve eyleyici düğümlerin bileşenleri görülmektedir. Algılayıcı düğüm güç birimi, algılama birimi, analog-dijital çevirici (ADC), işlemci ve depolama birimi ve iletişim alt birimi (alıcı/verici) kısımlarından oluşmaktadır. Algılama birimi ısı, ışık ve ses gibi olayları gözlemler. Toplanan veriler ADC tarafından analog veriye çevrilir. Çevrilen veri işlemci tarafından analiz edildikten sonra yakındaki eyleyiciye iletilir.

Karar verme birimi (denetleyici) fonksiyonları, algılayıcı okumalarını giriş olarak alır, eylem komutlarını ise çıkış olarak üretir. Eylem komutları dijital-analog çevirici (DAC) tarafından analog sinyale çevrilir ve çalıştırma birimine gönderilir [5].

Bazı uygulamalarda entegre edilmiş algılayıcı/eyleyici düğümler, eyleyici düğümlerin yerini alabilirler. Çünkü entegre algılayıcı/eyleyici düğümler algılama ve

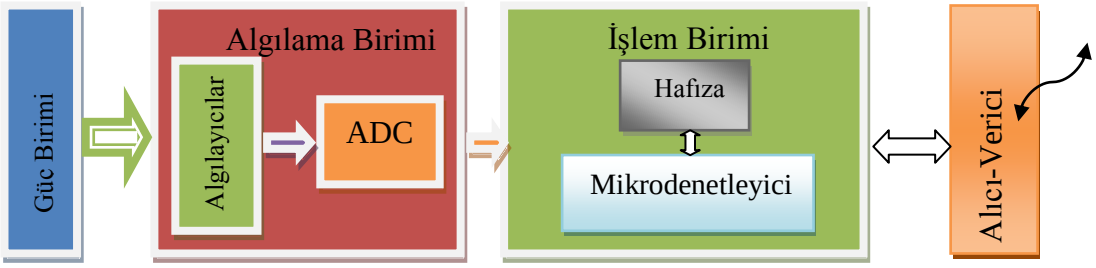
eylem işlemlerinin her ikisini birden yapabilirler. Bu düğümler algılama ve ADC birimlerine ek olarak eyleyici düğümlerin bütün bileşenlerine sahiptirler.

Algılayıcı/eyleyici düğümlere örnek olarak robot verilebilir. Ancak tek başına robot, tüm etkinlik alanı içerisinde yeterli bir algılama yeteneğine sahip olmayabilir. Bundan dolayı robotların (algılayıcı/eyleyici düğüm), daha güvenilir hareketlerde bulunabilmeleri için, kendi algılayıcı okumalarının yanı sıra yakındaki diğer algılayıcı düğümlerden gelen verileride değerlendirmesi gerekmektedir. Daha sonra karar verme birimi uygun kararları alır ve çalıştırma birimi bir eyleyici düğüm gibi eylemleri gerçekleştirir.

Entegre algılayıcı/eyleyici düğümlerin kullanılması genel KAEA mimarisini etkilemez. Gerçek uygulamaların çoğunda entegre algılayıcı/eyleyici düğümler, özellikle robotlar, eyleyici düğümlerin yerine kullanılırlar [5].

### 2.3. Kablosuz Algılayıcı Düğüm Yapısı

Kablosuz algılayıcı ağlarda kullanılan algılayıcı düğümler, hesaplama, algısal bilgi toplama ve ağdaki diğer bağlantılı düğümlerle haberleşme yeteneklerine sahip düğümlerdir [12]. Algılama, veri işleme, iletişim ve güç birimlerinden meydana gelen bu küçük algılayıcı düğümlerin ortak olarak çalışması, kablosuz algılayıcı ağlarının temel çalışma prensiplerini oluşturmaktadır. Şekil 2.6'da bir algılayıcı düğümün genelleştirilmiş mimarisi görülmektedir [2-13]. Algılayıcı düğümler genelde 6 tip bileşenden oluşur. Bunlar; işlemci, bellek ünitesi, güç kaynağı, algılayıcı ve/veya erişim düzeneği ve son olarak, haberleşme alt sistemidir.




Şekil 2.6: Algılayıcı düğüm mimarisi

### 2.3.1. Güç birimi

Güç birimi, dolaylı olarak tüm ağın ömrünü belirlemesi sebebiyle algılayıcı düğümlerinin en önemli birimidir. Algılayıcı düğümlerde başlıca kullanılan güç kaynakları, boyut sınırlaması nedeniyle genellikle standart AA piller veya kristal hücrelerdir. Bazı uygulamalarda güneş enerjisi ile şarj olabilen piller tercih edilebilmektedir. Böylelikle bir düğümün ömrü çok uzun sürelere çıkabilmektedir [13]. Bir algılayıcı düğümündeki enerji tüketimi algılama, iletişim ve veri işleme nedeniyle olmaktadır. Bu üç işlemden en fazla enerji tüketimine neden olan veri iletimidir. 1 Kb veriyi 100 metrelik bir uzaklığa iletmek için gereken enerji, yaklaşık olarak saniyede 100 milyon komut işleyen bir işlemcide 3 milyon komut işlemek için gereken enerjiye eşittir. Günümüzdeki düğümler yenilenebilir enerji kaynaklarını da (güneş enerjisi, ısı enerjisi, titreşim enerjisi vb.) kullanabilecek şekilde geliştirilmektedir. Kullanılan en önemli iki güç koruma politikası dinamik güç yönetimi (Dynamic Power Management DPM) ve dinamik voltaj ölçeklendirme (Dynamic Voltage Scaling - DVS)'dir. DPM kullanılmayan veya etkin olmayan parçaları kapatma görevini gerçekleştirir, DVS yaklaşımı kararlı olmayan iş yüküne bağlı olarak güç seviyeleri arasında geçişler yaparak çalışır [14].

### 2.3.2. Algılama birimi

Algılayıcılar ve ADC (Anolog/Digital Converter- Analog/Sayısal Çevirici)'lerden meydana gelen algılama birimi ışık, nem v.b. fiziksel büyüklüklerin ortamdan elde edilmesi ve bu büyüklüklerin işlem birimi tarafından işlenebilecek formata getirilmesinden sorumludur [13].

Algılayıcı düğümleri küçük boyutlu, düşük enerji tüketimli, otonom, gözetimsiz çalışabilen, ortama uyum sağlayabilen özelliklere sahip olmalıdır. Kablosuz algılayıcı düğümler sadece düşük güç tüketen sınırlı güç kaynağına sahip (0,5 Ah ve 1,2 V gibi) mikro-elektronik algılayıcı aygıtlarını kullanabilir. Algılayıcılar üç kategori şeklinde sınıflandırılmaktadır.

Pasif, her yöne açık (yönsüz) algılayıcılar: Pasif algılayıcılar, ortamı aktif araştırma ile değiştirmeden, verileri toplayan algılayıcılardır. Kendi enerjilerine sahiptirler ve enerji analog sinyali yükseltmek için gereklidir. Bu ölçümlerde "yön" şeklinde bir kavram yoktur.

Pasif, dar ışınlı algılayıcılar: Bu algılayıcılar pasiftir ancak iyi tanımlanmış ölçüm yönü kavramına sahiptir. Bu algılayıcıya örnek olarak kamera verilebilir.

Aktif algılayıcılar: Bu gruptaki algılayıcılar ortamı aktif olarak araştırırlar, örnek olarak sonar veya radar algılayıcıları veya küçük patlamalarla şok dalgaları üreterek çalışan bazı sismik algılayıcı tipleri verilebilir [14].

### 2.3.3. İşlem birimi

Mikrodenetleyici ve hafıza birimlerinden oluşan işlem birimi, kod bellekte yüklü olan ve düğümlerin ağ içerisinde yapmakla yükümlü olduğu görev komutlarının işlenmesinden sorumlu kısımdır [13].

Mikrodenetleyici veriyi işler ve algılayıcı düğüm içerisindeki diğer bileşenlerin işlevselliğini denetler. Maliyet ve düşük güç tüketimi gibi avantajlarından dolayı mikrodenetleyiciler algılayıcı düğüm için en uygun seçimdir. Genel amaçlı mikroişlemciler mikrodenetleyicilerden daha fazla enerji harcamaktadırlar [14].

### 2.3.4. Alıcı-Verici birimi

Alıcı ve vericinin işlevselliği alıcı-verici adı verilen tek bir aygıt içerisinde birleştirilmiştir. Alıcı-vericiler genelde gönderme (Transmit), alma (Receive), boşta bekleme (Idle) ve uyku (Sleep) olmak üzere dört farklı işlem moduna sahiptir. Yeni nesil alıcı/vericiler bu işlem modlarını otomatik olarak gerçekleştiren gömülü durum makinelerine sahiptir. Boş modda çalışan alıcı/vericilerin güç tüketimi neredeyse alma modundaki enerji tüketimine eşittir. Bu yüzden alma veya iletme işlemi yapmayan alıcı/vericinin boş moda geçirilmesi yerine uyku moduna geçmesi enerji

açısından daha elverişli bir çözümdür. Ayrıca paket iletimi için uyku modundan iletim moduna geçerken de önemli miktarda enerji (alıcı/verici) harcanmaktadır [14].

## 2.4. Kablosuz Algılayıcı Ağ Uygulama Alanları

Mikro elektronik ve mekaniksel sistem tasarımındaki gelişmeler sıcaklık, nem, basınç, titreşim, ses, görüntü ve kimyasal sızıntı gibi fiziksel büyüklüklerin sezilmesini sağlayan algılayıcıların kablosuz haberleşebilen düğümlere entegre edilebilmesini mümkün kılmaktadır. Kablosuz algılayıcı ağ uygulamalarını iki temel başlık altında sınıflandırmak mümkündür. Bunlar izleme ve görüntülemedir. Bu çalışmada bu iki sınıfın yanında, sistemlerin KAA ile görüntülenmesi ve denetlenmesini sağlayan bir test düzeneği geliştirilmiştir. Şekil 2.7'de KAA uygulama alanlarını gösteren akış şeması görülmektedir [16].

Uygulama alanları aşağıdaki gibi listelenebilir [15].

➢Endüstriyel otomasyon

- Süreç izleme ve kontrol
- Enerji hatlarının izlenmesi ve bütünlüğünün sağlanması
- Benzin-Gaz üretimi ve taşımacılığı
- Titreşim izleme

➢Üretim, depolama ve taşımacılık

- Ürün takibi
- Trafik izleme
- Ürün yer tayini
- Akıllı taşıyıcılar ( deformasyon vs.)

➢Yapı otomasyon

- İzleme ve kayıt
- Yer tayini (çalışan, malzeme, araç…)
- Işıklandırma kontrolü
- Yangın alarmı
- Deprem tahmini

➢Çevresel takip

- Tarım-Sulama-Seracılık
- Gıda kalitesi
- Hava durumu
- Hayvancılık

➢Sağlık

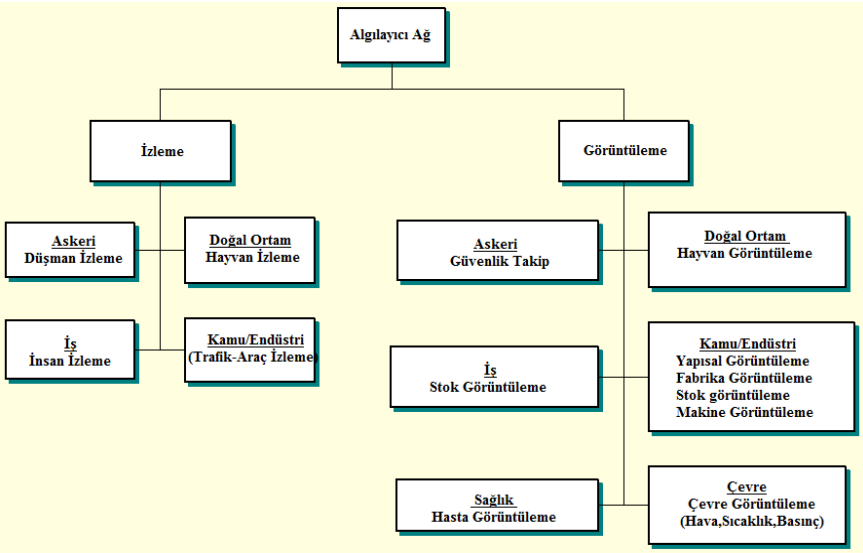
- Sağlık parametreleri izleme
- Yaşlı kişilerin takibi

➢Elektronik ve bilgisayar

- Akıllı evler
- TV-DVD-VCR
- Kablosuz PC yan ürünleri
- Cep telefonları

➢Askeri sistemler

- Düşman izleme
- Alçak mesafe ses radarları
- Denizaltı algılayıcıları
- Personel ve taşıt izleme




Şekil 2.7: KAA uygulama alanları

# 3. GELİŞTİRİLEN SİSTEMİN DONANIM VE YAZILIM BİLEŞENLERİ

## 3.1. Donanım Bileşenleri

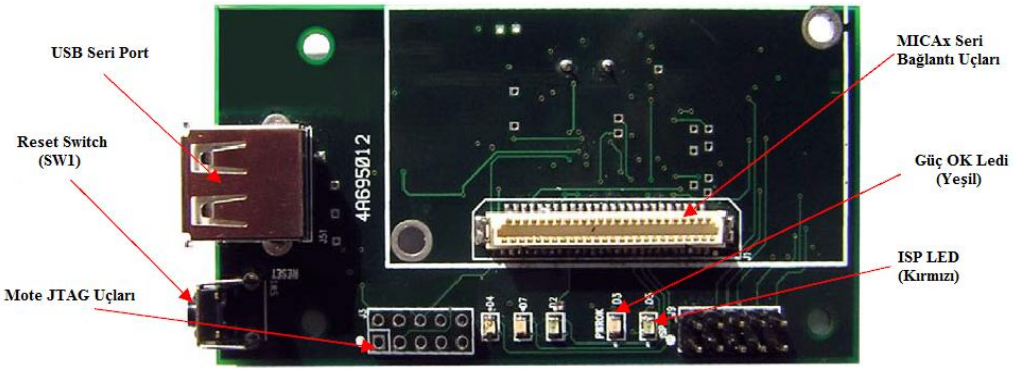
Tez çalışmasında donanım bileşenleri olarak kablosuz algılayıcı ağ elemanları, birinci dereceden ölü zamanlı sistem düzeneği, dijital-analog çevirici (DAC) ve PIC16F877 mikrodenetleyicisi kullanılmıştır.

### 3.1.1. Kablosuz algılayıcı ağ bileşenleri

Tez çalışmasında kullanılan KAA bileşenleri MIB520 programlama kartı ve MDA320 veri edinim bordudur.

#### 3.1.1.1. MIB520 USB arayüz kartı

USB kablo üzerinden bilgisayara bağlanan MIB520, MICA ve IRIS düğüm ailesindeki cihazların iletişimi ve programlaması için kullanılan çok amaçlı bir karttır [17]. Sistemde MIB520, MDA320 veri edinim bordu ile bilgisayar arasındaki bağlantıyı sağlayan erişim noktası ve MDA320 üzerinde bulunan MPR2600 cihazların programlanması için kullanılmıştır.

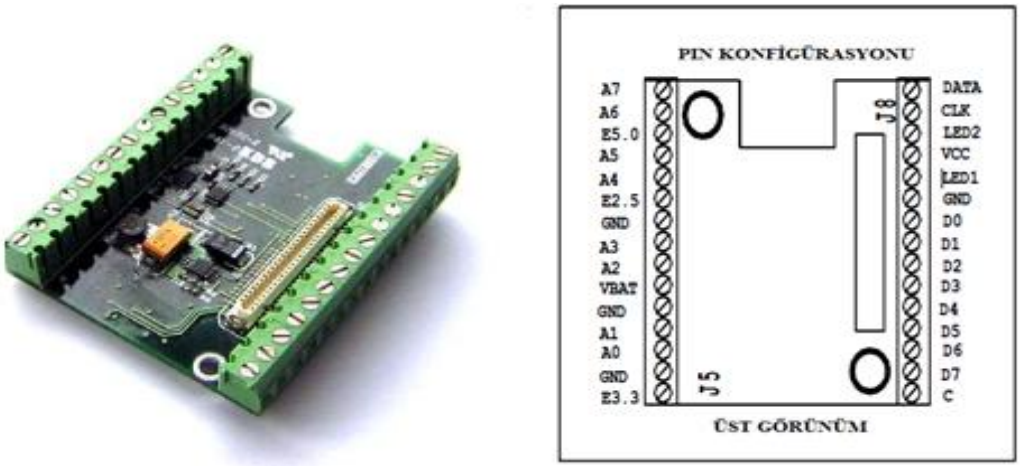



Şekil 3.1: MIB520CA üstten görünüşü

### 3.1.1.2. MDA320 veri edinim bordu

Kablosuz algılayıcı ağ bileşenlerinden biri de MDA320 veri edinim bordudur (Data acquisition board, DAQ). MDA320 veri edinim bordu üzerinde her biri 8 bit çözünürlüklü 8 analog giriş, 8 dijital giriş – çıkış kanalı, dışarıdan algılayıcı bağlanabilen 3 farklı seviye ucu ve harici $I^2C$ haberleşme uçları bulunmaktadır.

MDA320 bordunun öncelikli kullanım alanları, düşük güçlü ölçümler, hava ölçüm sistemleri, hassas tarım ve sulama kontrolleri, ekolojik izlemeler, toprak analizleri ve uzaktan süreç denetimi olarak gösterilebilir [18]. Şekil 3.2'de MDA320 veri edinim bordu görülmektedir. Tablo 3.1'de MDA320 veri edinim bordunun uçlarının fonksiyonları verilmektedir [18].




Şekil 3.2: MDA320 veri edinim bordu ve uç konfigürasyonu (üstten görünüş)

Tablo 3.1: MDA320 uçlarının fonksiyonları

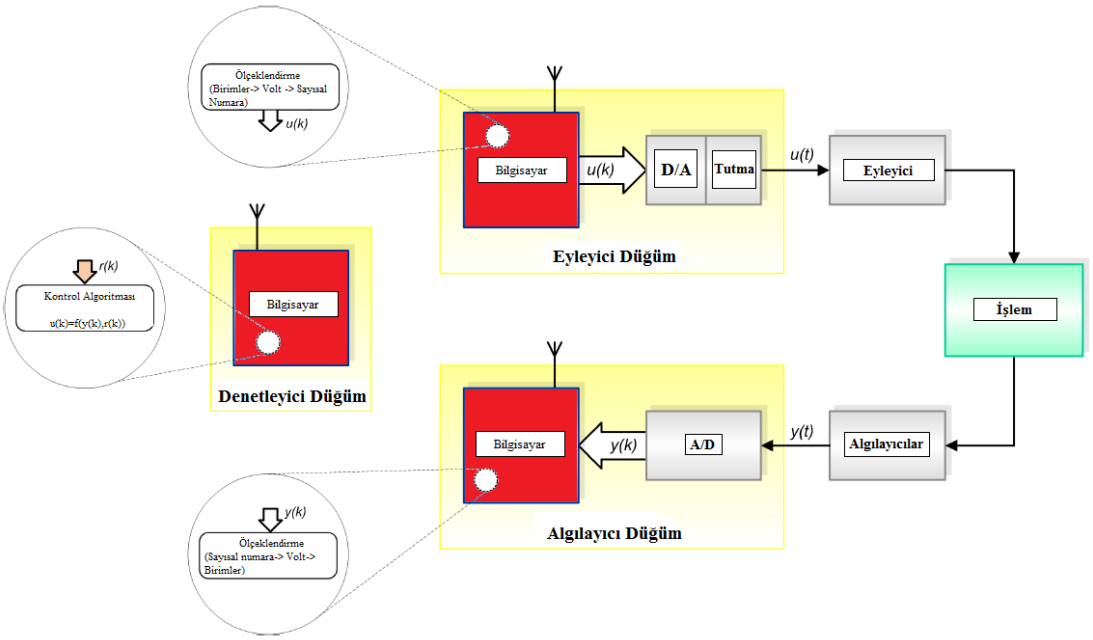
| UÇ | TANIMLAMALAR |
|---|---|
| **A7** | Tek uçlu Analog kanal 7 veya ayrışık analog kanal 11 pozitif kenarı |
| **A6** | Tek uçlu Analog kanal 6 veya ayrışık analog kanal 11 negatif kenarı |
| **E5.0** | 5.0 V Çıkış |
| **A5** | Tek uçlu Analog kanal 5 veya ayrışık analog kanal 10 negatif kenarı |

Tablo 3.1: MDA320 uçlarının fonksiyonları (Devamı)

| UÇ | TANIMLAMALAR |
|---|---|
| **A4** | Tek uçlu Analog kanal 4 veya ayrışık analog kanal 10 pozitif kenarı |
| **E2.5** | 2.5 V Çıkış |
| **GND** | Elektriksel Toprak |
| **A3** | Tek uçlu Analog kanal 3 veya ayrışık analog kanal 9 negatif kenarı |
| **A2** | Tek uçlu Analog kanal 2 veya ayrışık analog kanal 9 pozitif kenarı |
| **VBAT** | Beslemenin pozitif terminalindeki gerilim |
| **A1** | Tek uçlu Analog kanal 1 veya ayrışık analog kanal 8 negatif kenarı |
| **A0** | Tek uçlu Analog kanal 0 veya Ayrışık analog kanal 8 pozitif kenarı |
| **GND** | Elektriksel Toprak |
| **E3.3** | 3.3 V Çıkış |
| **DATA** | $I^2C$ Veri Hattı |
| **CLK** | $I^2C$ Saat Hattı |
| **LED2** | Yeşil Led |
| **Vcc** | Cihazın gerilimi |
| **LED1** | Kırmızı Led |
| **GND** | Elektriksel Toprak |
| **D0-D7** | Dijital Hatlar D0-D7 |
| **C** | Sayıcı Kanal |

### 3.1.2. Ağ kontrol sistemi ve birinci dereceden ölü zamanlı sistemler

Klasik gerçek zamanlı kontrol sistemi, kontrol edilebilen endüstriyel süreç, bir veya daha fazla denetleyici, algılayıcılar ve eyleyicilerden meydana gelmektedir. Süreç ölçümleri, denetleyici tarafında bulunan analog-dijital çevirici birimi ile algılanıp örneklenerek toplanırlar. Örneklenen ölçümler daha sonra içerisinde kontrol algoritmaları bulunan fiziksel birimler için ölçeklendirilirler. Kontrol işlemleri daha sonra analog sinyale çevrilerek eyleyicilere ulaştırılırlar. Şekil 3.3'te kablosuz ağ kontrol sistemi görülmektedir [4].



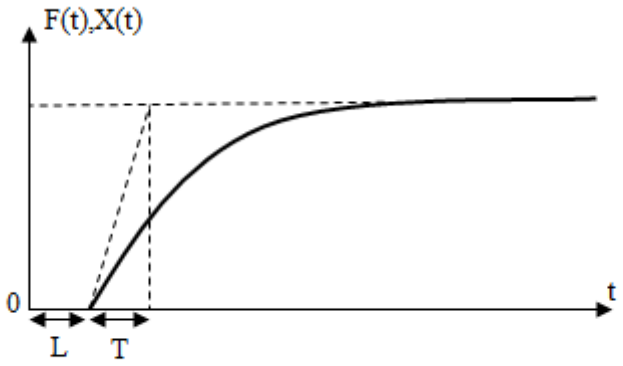

Şekil 3.3: Kablosuz ağ kontrol sistemi

Ağ kontrol sisteminde, dağıtılmış klasik fonksiyonlara sahip gerçek zamanlı kontrol sistemi ile birkaç ağ/işleyici düğüm arasında, bir iletişim ağı tarafından kablolu veya kablosuz olarak bağlantı kurulur. Klasik kontrol sistemleri ile karşılaştırıldığında, ağ kontrol sistemlerinde tüm işlemler, kablosuz ağ ile iletişim kurabilen mekana dağıtılmış işlemci düğümler tarafından yapılır [4].

Ölü zamanlı sistemler üzerine yapılan araştırmalar artmakta ve bu sistemler için tasarlanan denetleyiciler gün geçtikçe çeşitlenerek gelişmektedir. Bu sistemlerin matematiksel modelleri kolay elde edilemediğinden mühendislik uygulamalarını yapmak oldukça zordur [19]. Sanayide kullanılan endüstriyel kontrol sistemlerinin büyük bir bölümü ölü zamanlı sistemlerdir. Ölü zamanlı sistemler için X(s) giriş büyüklüğü F(s) çıkış büyüklüğü olmak üzere transfer fonksiyonu aşağıdaki gibi tanımlanabilir.

$$\frac{F(s)}{X(s)} = \frac{K.e^{-sL}}{1+Ts} \tag{3.1}$$

Burada K terimi kazancı, T terimi zaman sabitini, L terimi ise ölü zamanı ifade etmektedir. Şekil 3.4'te ölü zamanlı sistemin birim basamak cevabı görülmektedir [20].

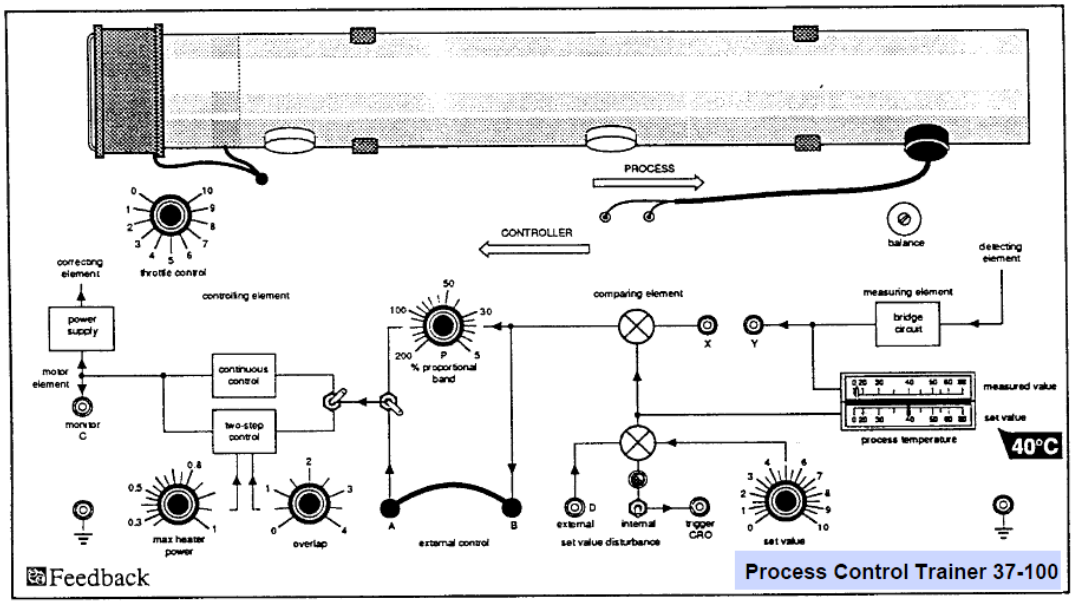



Şekil 3.4: Ölü zamanlı sistemin birim basamak cevabı

Şekilde görüldüğü gibi sistem L zamanı kadar bir gecikme ile birim basamak fonksiyonuna cevap vermektedir. Bunun sebebi sistemde bulunan $e^{-sL}$ ifadesidir. Bu sistem herhangi bir denetleyici ile kontrol edilmek istenirse sistemde meydana gelen gecikmeler önemli sorunlar yaratabilir. Denetleyen sistem eski değerleri değerlendirerek hatalı kontrol işaretleri üretebilir. Gecikme, giriş işaretindeki değişiminin etkisinin çıkış işaretinde gecikmeli olarak görülmesinden kaynaklanır [21].

Tez çalışmasında FeedBack firmasının ürettiği süreç denetim sistemi (Process Control Trainer) kullanılmıştır. Kullanılan 37-100 süreç denetim sistemi üzerindeki aksam ile kendi kendine denetlenebilen bir süreç ve denetim donanımıdır. Temel özellikleri arasında büyük bir denetlenen sistem, mesafe/hız gecikmesinin sağlanması, transfer gecikmesi, sistem cevabı, oransal ve iki adımlı denetim gösterilebilir. Oldukça hızlı cevap vermesi nedeniyle, ayar değeri ve ölçülen değer arasındaki değişimler osilaskop ekranından izlenebilir [22]. Şekil 3.5'te süreç denetim sistemi görülmektedir [22].

Sistem kendi üzerindeki donanım ile kontrol edilebildiği gibi harici girişleri ile de kontrol edilebilmektedir.




Şekil 3.5: Süreç denetim sistemi (Process Control Trainer 37-100)

### 3.1.3. PIC16F877 mikrodenetleyici

Tez çalışmasında denetleme elemanı olarak mikrodenetleyici kullanılmıştır. Mikrodenetleyici olarak ise $I^2C$ ve ADC modülleri olması nedeniyle PIC16F877 mikrodenetleyicisi seçilmiştir. Aşağıdaki bölümlerde PIC16F877 mikrodenetleyicisi ile ilgili geniş bilgiler bulunmaktadır.

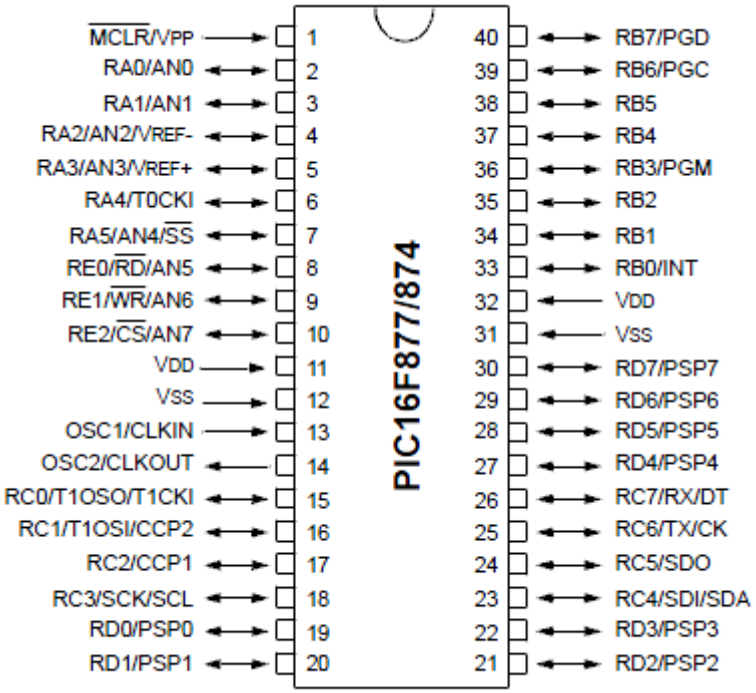
#### 3.1.3.1. PIC16F877 mikrodenetleyicisinin genel özellikleri

- İçerisinde yüksek performanslı azaltılmış komut setine sahip (RISC) mikroişlemci bulunmaktadır. Programlamada kullanılan 35 komut vardır ve komutlardan her biri 14 bit uzunluktadır.
- Dallanma komutları iki saat darbesi süresinde, diğerleri ise bir saat darbesi süresinde çalışır.
- Kristal frekansı 20 MHz'e kadar arttırılabilir.
- Veri yolu 8 bittir.

- 8 KB flash program belleği, 368 Byte veri belleği (RAM), 256 Byte EEPROM veri belleği vardır ve 1 milyon kez programlanabilirler.
- Port A,B,C,D ve E olmak üzere 5 adet giriş çıkış portu bulunur.
- 54 adet SFR olarak adlandırılan özel işlem yazmacı vardır ve bu yazmaçlar statik RAM üzerindedir.
- 14 kaynaktan kesme yapabilir.
- Power-on Reset (Enerji verildiğinde sistemi resetleme özelliği) vardır.
- Power-up Timer (Power-up zamanlayıcı), Osilatör Start-up Timer (Osilatör başlatma zamanlayıcısı), Watch-dog Timer (Bekçi köpeği zamanlayıcısı) vardır.
- Devre içi Debugger (Hata ayıklamakta kullanılabilecek modül) bulunur.
- Düşük gerilimli ve sadece 2 pinle programlama özelliği.
- Uyku modu.
- Seçimli osilatör özelliklerine sahiptir.
- 2.0 V – 5.0 V arasında besleme gerilimi aralığına sahiptir.
- 25 mA'lik kaynak akımı standardı.
- Geniş sıcaklık aralığında çalışabilme özelliği.
- Düşük güçle çalışabilme.
- TMR0:8 bitlik zamanlayıcısı, 8 bit önbölücülü olarak kullanılabilir.
- TMR1 önbölücülü 16 bit zamanlayıcı içeririr. Uyuma modundayken kontrol edilebilir ve değeri arttırılabilir.
- TMR2 8 bitlik zamanlayıcı hem önbölücü hem de son bölücü sabitine sahiptir.
- İki tane Yakalama / Karşılaştırma / PWM modülüne sahiptir. Yakalama ve karşılaştırma 16 bit, PWM ise maksimum 10 bit çözünürlükle yapılabilir.
- 10 bit çok kanallı A/D çeviriciye sahiptir.
- Ana Senkron seri port (MSSP) modülü, SPI (Master mod) ve I2C (Master Slave) modlarında kullanılabilir.
- Asenkron seri iletişim için USART seri iletişim ara birimine sahiptir.
- Paralel haberleşme için 8 bit genişlikte Paralel Slave Portu bulunur, bu port ile dış RD, WR, CS kontrollerine sahiptir.
- BOR (Brown Out) Reset (Voltaj Düşmesi Reset) özelliğine sahiptir [23].

### 3.1.3.2. PIC16F877 mikrodenetleyicisi bacak yapısı ve uç fonksiyonları

Şekil 3.6'da PIC16F877 mikrodenetleyicisinin bacak yapısı, Tablo 3.2'de ise uç fonksiyonları verilmektedir.




Şekil 3.6: PIC16F877 mikrodenetleyici bacak yapısı

Tablo 3.2: PIC16F877 mikrodenetleyicinin uç fonksiyonları

| PİN ADI | PİN NO | PİN TİPİ | TAMPON TİPİ | AÇIKLAMALAR |
|---|---|---|---|---|
| OSC1/CLKIN | 13 | I | ST/CMOS(4) | Osilatör saat darbesi girişi (Kristal veya harici kaynak) |
| OSC1/CLKOUT | 14 | O | - | Osilatör kristal çıkış ucu |

Tablo 3.2: PIC16F877 mikrodenetleyicinin uç fonksiyonları (Devamı)

| PİN ADI | PİN NO | PİN TİPİ | TAMPON TİPİ | AÇIKLAMALAR |
|---|---|---|---|---|
| MCLR/Vpp | 1 | I/P | ST | Resetleme girişi/Programlama anında programlama gerilimi girişi (Resetleme için uç lojik 0 yapılmalı) |
| RA0/AN0 | 2 | I/O | TTL | Çift yönlü giriş/çıkış. Analog giriş 0. |
| RA1/AN1 | 3 | I/O | TTL | Çift yönlü giriş/çıkış. Analog giriş 1. |
| RA2/AN2/$V_{REF-}$ | 4 | I/O | TTL | Çift yönlü giriş/çıkış. Analog giriş 2.<br>Negatif analog referans gerilimi. |
| RA3/AN3/$V_{REF+}$ | 5 | I/O | TTL | Çift yönlü giriş/çıkış. Analog giriş 3.<br>Pozitif analog referans gerilimi. |
| RA4/T0CKI | 6 | I/O | ST | Çift yönlü giriş/çıkış. RA4 Zamanlayıcı0 zamanlayıcı/sayıcı saat darbe giriş ucu olarak kullanılabilir. Çıkış modunda açık drain tipindedir. |
| RA5/SS/ AN4 | 7 | I/O | TTL | Çift yönlü giriş/çıkış. SSP için slave seçme pini. Analog giriş 4. |
| RB0/INT | 33 | I/O | TTL/ST(1) | Çift yönlü giriş/çıkış. Dış kesme girişi olarak seçilebilir. |
| RB1 | 34 | I/O | TTL | Çift yönlü giriş/çıkış. |
| RB2 | 35 | I/O | TTL | Çift yönlü giriş/çıkış. |
| RB3/PGM | 36 | I/O | TTL | Çift yönlü giriş/çıkış. Düşük gerilimli programlamada da kullanılabilir. |
| RB4 | 37 | I/O | TTL | Çift yönlü giriş/çıkış. PortB kesme pini. |
| RB5 | 38 | I/O | TTL | Çift yönlü giriş/çıkış. PortB kesme pini. |
| RB6/PCG | 39 | I/O | TTL/ST(2) | Çift yönlü giriş/çıkış. PortB kesme pini. Devre içi hata ayıklama pini. Seri programlamada saat darbesi girişi. |
| RB7/PGD | 40 | I/O | TTL/ST(2) | Çift yönlü giriş/çıkış. PortB kesme pini. Devre içi hata ayıklama pini. Seri programlamada veri girişi. |

Tablo 3.2: PIC16F877 mikrodenetleyicinin uç fonksiyonları (Devamı)

| PİN ADI | PİN NO | PİN TİPİ | TAMPON TİPİ | AÇIKLAMALAR |
|---|---|---|---|---|
| RC0/T1OS0/T1CK1 | 15 | I/O | ST | Çift yönlü giriş/çıkış. Zamanlayıcı1 osilatör çıkışı veya zamanlayıcı1 saat darbe girişi. |
| RC1/T1OS1/CCP2 | 16 | I/O | ST | Çift yönlü giriş/çıkış. Zamanlayıcı1 osilatör girişi. Yakalama2 (Capture2) girişi. Karşılaştırma2 (Compare2) çıkışı. PWM2 çıkışı. |
| RC2/CCP1 | 17 | I/O | ST | Çift yönlü giriş/çıkış. Yakalama1 (Capture1) girişi. Karşılaştırma1 (Compare1) çıkışı. PWM1 çıkışı. |
| RC3/SCK/SCL | 18 | I/O | ST | Çift yönlü giriş/çıkış. SPI VE $I^2C$ modları için senkron seri saat darbesi giriş/çıkışı. |
| RC4/SD1/SDA | 23 | I/O | ST | Çift yönlü giriş/çıkış. SPI modda SPI veri girişi veya $I^2C$ modda veri girişi ya da çıkışı. |
| RC5/SDO | 24 | I/O | ST | Çift yönlü giriş/çıkış. SPI modda SPI veri çıkışı. |
| RC6/TX/CK | 25 | I/O | ST | Çift yönlü giriş/çıkış. USART veya senkron saat darbesi. |
| RC7/RX/DT | 26 | I/O | ST | Çift yönlü giriş/çıkış. USART asenkron alma veya senkron veri. |
| RD0/PSP0 | 19 | I/O | ST/TTL(3) | Çift yönlü giriş/çıkış. Paralel Slave Port 0 (PSP0) |
| RD1/PSP1 | 20 | I/O | ST/TTL(3) | Çift yönlü giriş/çıkış. PSP1. |
| RD2/PSP2 | 21 | I/O | ST/TTL(3) | Çift yönlü giriş/çıkış. PSP2. |
| RD3/PSP3 | 22 | I/O | ST/TTL(3) | Çift yönlü giriş/çıkış. PSP3. |
| RD4/PSP4 | 27 | I/O | ST/TTL(3) | Çift yönlü giriş/çıkış. PSP4. |
| RD5/PSP5 | 28 | I/O | ST/TTL(3) | Çift yönlü giriş/çıkış. PSP5. |
| RD6/PSP6 | 29 | I/O | ST/TTL(3) | Çift yönlü giriş/çıkış. PSP6. |
| RD7/PSP7 | 30 | I/O | ST/TTL(3) | Çift yönlü giriş/çıkış. PSP7. |
| RE0/RD/AN5 | 8 | I/O | ST/TTL(3) | Çift yönlü giriş/çıkış. Analog giriş 5. Paralel slave port için okuma kontrol ucu. |
| RE1/WR/AN6 | 9 | I/O | ST/TTL(3) | Çift yönlü giriş/çıkış. Analog giriş 6. Paralel slave port için yazma kontrol ucu. |

Tablo 3.2: PIC16F877 mikrodenetleyicinin uç fonksiyonları (Devamı)

| PİN ADI | PİN NO | PİN TİPİ | TAMPON TİPİ | AÇIKLAMALAR |
|---|---|---|---|---|
| RE2/CS/AN7 | 10 | I/O | ST/TTL(3) | Çift yönlü giriş/çıkış. Analog giriş 7. Paralel slave port için seçme kontrol ucu. |
| $V_{SS}$ | 12, 31 | P | | Toprak Ucu. |
| $V_{DD}$ | 11, 32 | P | | Pozitif Uç. |

I: Input (Giriş) O: Output(Çıkış) I/O: Input/Output (Giriş/Çıkış) P: Power (Güç)

TTL: TTL Giriş ST: Schmitt Trigger -: Kullanılmıyor

Not: 1. Bu tampon PORTB dış kesme olarak ayarlanmışsa Schmitt Trigger giriştir.

2. Bu tampon seri programlama modunda Schmitt Trigger girişidir.

3. Bu tampon genel amaçlı giriş-çıkış olarak kullanıldığında Schmitt Trigger girişi ve PSP modunda TTL giriştir.

4. Bu tampon RC osilatör modunda Schmitt Trigger girişidir. Diğer modlarda CMOS girişidir.

### 3.1.4. Dijital-analog çevirici

Dijital analog çeviriciler (DAC), girişinde bulunan sayısal değerlere karşılık analog bir gerilim veya akım üretmektedir. Analog dijital çeviriciler'de (ADC) olduğu gibi bit çözünürlüğü, adım büyüklüğü gibi birçok kavram DAC'lar için de geçerlidir. DAC'lar ile sayısal olarak çalışan mikroişlemci ve mikrodenetleyiciler ile analog mantıkla çalışan cihazların kontrolünü gerçekleştirmek mümkündür.

8 bit çözünürlüklü bir DAC, maksimum $2^8$=256 adet farklı sayısal değerin analog karşılığını üretebilmektedir. DAC'nin referans gerilimleri olan Vref(+) = 5V ve Vref(-) = 0 ise her bir sayısal değer için üretilecek analog sinyalin karşılığı şu şekilde hesaplanabilir.

$$DAC\,Adım\ Büyüklüğü = \frac{Referans\ Gerilimi}{2^{BitSayısı}} \tag{3.2}$$

$$DAC\,Adım\ Büyüklüğü = \frac{5}{256} = 0{,}01953125 \tag{3.3}$$

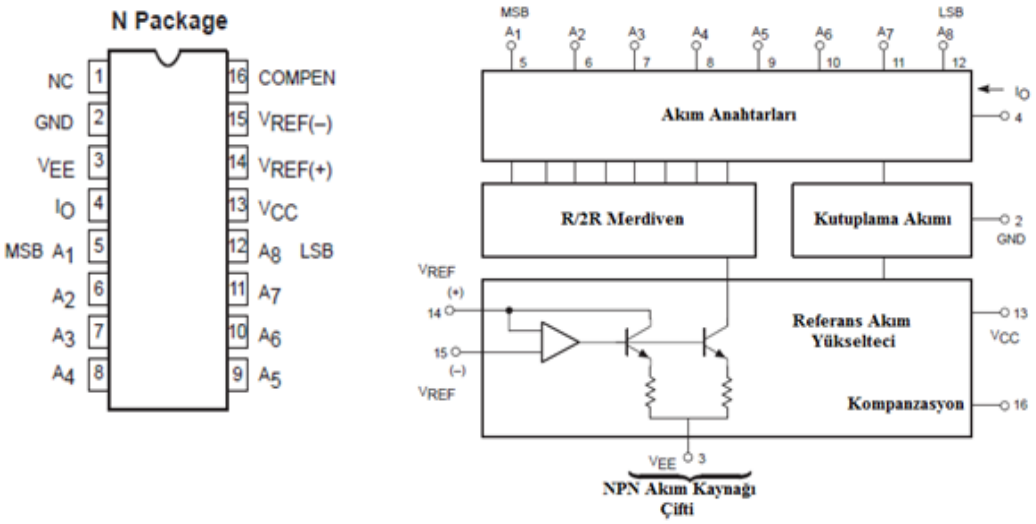
Tablo 3.3: 8 bit çözünürlüklü bir DAC için çıkış değerleri

| Dijital Giriş | Analog Çıkış (V) |
|---|---|
| **00000000** | 0 |
| **00000001** | 0,01953125 |
| **00000010** | 0,0390625 |
| **00000011** | 0,05859375 |
| **…** | … |
| **…** | … |
| **11111111** | 4,98046875 |

Tez çalışmasında, DAC gerçeklemesi yapılırken en popüler ve ucuz DAC'lardan biri olan MC1408 entegresi kullanıldı. Bu entegrenin eşdeğeri DAC0808 entegresidir. MC1408 standart 16 bacak DIP paket bir entegredir ve +5 V'luk +Vcc ile en az -5 V, en fazla -15 V'luk VEE gerilimi ile beslenir. MC1408 entegresinin içinde bulunan R/2R merdiven tipi D/A çevirici, akım yükseltecinden gelen referans akımını, 8 ikilik ağırlıklı akıma böler. Bipolar transistör anahtarlar olan A1-A8 girişlerindeki ikilik bilgiye göre ikilik ağırlıklı akımları çıkış hattına bağlar. En yüksek değerlikli biti taşıyan giriş A1, en düşük değerlikli biti taşıyan giriş A8 ile gösterilmiştir. Şekil 3.7'da MC1408 bacak yapısı ve blok diyagramı görülmektedir [24].

MC1408'in bir işlemsel yükselteç (op-amp) ve bir dirençle gerilime çevrilebilen akım çıkışı vardır. Çıkış gerilimi genel olarak şöyle hesaplanabilir.

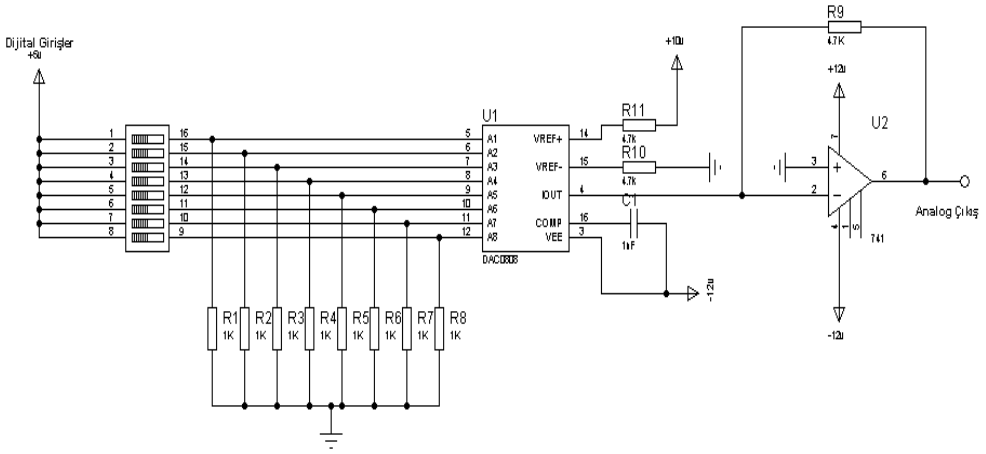
$$Vout = \frac{Vref}{R14} + Rf\left(\frac{A1}{2} + \frac{A2}{4} + \frac{A3}{8} + \frac{A4}{16} + \frac{A5}{32} + \frac{A6}{64} + \frac{A7}{128} + \frac{A8}{256}\right) \tag{3.4}$$




Şekil 3.7: MC1408 bacak bağlantıları ve blok diyagramı

MC1408 DAC entegresi akım çıkışlıdır. Entegrenin dört numaralı bacağından şaseye doğru bir direnç çekilerek gerilim çıkışı okunabilir. Devrenin çıkışını Vref gerilimi direkt olarak etkilemektedir. Devreden elde edilmek istenen çıkışa göre Vref gerilimi ayarlanmaktadır. Devre çıkışındaki opampın görevi DAC çıkışından gelen akımı, Rf direncinin değerinin kazanca olan etkisine göre katlamaktır. DAC entegresinin akım çıkışı opampın eviren girişine bağlanmıştır. Opamp simetrik olarak beslenmiştir. Opamp besleme gerilimi +12V ve -12V olarak ayarlanmıştır. Çıkışta evirmeden 0-10V arası gerilim görülmesi için opampın dört numaralı bacağından verilen negatif besleme toprağa çekilebilir.

Yukarıdaki formülden de anlaşılabileceği gibi girişler ağırlıkları oranında çıkışı etkilerler. Burada A1 girişi MSB (En yüksek değerlikli bit), A8 girişi ise LSB ( En düşük değerlikli bit)'dir. A8 girişindeki değişim çıkışı çok küçük bir miktarda etkilerken A1 girişindeki değişim, formülden de anlaşılacağı gibi çıkışı yarı yarıya etkilemektedir. Girişlerin değişimine göre çıkışta $2^8$=256 farklı seviye oluşmaktadır. Çıkışta görülmek istenen değere göre girişler ayarlanabilmektedir. Şekil 3.8'de test düzeneği için kurulan dijital-analog çevirici devresi görülmektedir.




Şekil 3.8: Dijital-analog çevirici devresi

## 3.2. Yazılım Bileşenleri

Tez çalışmasında yazılımsal olarak $I^2C$ haberleşme protokolü, Perl, Matlab, nesC ve XSERVE kullanılmıştır.

### 3.2.1. $I^2C$ haberleşme protokolü

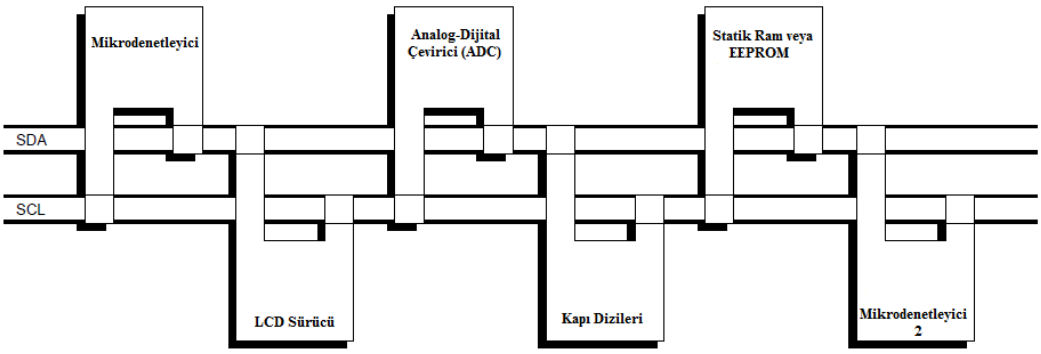
$I^2C$ (Inter-Integrated Circuit), mikrodenetleyiciler ve mikroişlemci tabanlı küçük çipler arasında kolayca iletişim kurulmasını sağlayan entegreler arası seri haberleşme protokolüdür. PHILIPS firması tarafından geliştirilmiştir.

$I^2C$'de, seri veri giriş-çıkış (SDA) ve seri saat sinyali (SCL) olmak üzere iki hat üzerinden iletişim gerçekleşmektedir. SDA hattındaki veri giriş çıkış işlemleri SCL hattındaki saat sinyali sayesinde eş zamanlı olarak gerçekleşmektedir. Bu haberleşmede entegreler arasında yönetici (master), yönetilen (slave) ilişkisi bulunmaktadır. Master, veri iletimini başlatır, durdurur ve iletişim için gerekli olan saat sinyalini üretir. Hattaki her elemanın kendine özel bir adresi vardır ve bu adresle tanınırlar (Mikrodenetleyici, LCD, klavye vb.). Bu cihazlar özelliklerine göre gönderici (Transmitter), alıcı (Receiver) veya hem gönderici hem de alıcı olarak

(hatta veri gönderebilir, alabilir veya her ikisini birden yapabilir) çalışırlar [25]. Slave, saat sinyali üretmez sadece veri alışverişi yapar. Slave cihazlar yönetici tarafından adreslenirler.

$I^2C$ protokolü ile yavaş (slow), hızlı (fast) ve yüksek hızlı (high speed) olmak üzere çeşitli hızlarda veri iletişimi sağlanabilir. Yavaş (slow) hızda standart, veri iletişim hızı 100 kbit/s'ye kadar, hızlı (fast) modda veri iletişim hızı 400 kbit/s'ye kadar, yüksek hızlı iletişimde ise veri iletişim hızı 3,4 Mbit/s'ye kadar çıkabilmektedir [26].

$I^2C$, çok yöneticili bir protokoldür. Yani hattı birden fazla elemanın kontrol etme kabiliyeti vardır. Çok masterli $I^2C$ sistemlerde masterlar aynı anda veri iletimini başlatmaya çalışabilirler. Bu durumda arbitrasyon (arbitration) işlemi kullanılır. Bu durumda yöneticiler tarafından üretilen saat sinyalleri, arbitrasyon işlemi boyunca SCL hattının VE_bağlantılı lojik hattı (giriş hatlardan herhangi biri düşük (Low) seviyeye çekildiği anda çıkışı düşük seviyeye çekme) ile eşzamanlı hale getirilir [25]. Şekil 3.9'da örnek $I^2C$ bağlantısı görülmektedir [26].




Şekil 3.9: Örnek $I^2C$ bağlantısı

Tablo 3.4'te $I^2C$ protokolünde kullanılan terimler ve tanımlamaları verilmektedir [26].

Tablo 3.4: $I^2C$ protokolünde kullanılan terimler ve tanımlamaları

| Terim | Tanımlama |
|---|---|
| **Gönderici (Transmitter)** | Hatta veri gönderen elemana denir. |
| **Alıcı (Receiver)** | Hattan veri alan elemana denir. |
| **Yönetici (Master)** | $I^2C$ haberleşmesinde veri iletimini başlatan, durduran ve veri iletişimi için gerekli saat sinyallerini üretene elemana denir. |
| **Yönetilen(Slave)** | Yönetici tarafından adreslenen elemana denir. |
| **Çoklu-Yönetici (Multi-Master)** | Birden fazla yöneticinin aynı anda hattaki veriyi bozmadan hattın kontrolünü ele almaya çalışmasıdır. |
| **Arbitrasyon (Arbitration)** | Eğer hatta birden fazla yönetici varsa ve bunlar aynı anda hattın kontrolünü ele almaya çalışırlarsa, bunlardan sadece birinin hattı kontrol etmesi sağlanır ve veri kesilmeden iletilir. |
| **Senkronizasyon (Synchronization)** | İki veya daha fazla saat sinyalinin eşzamanlı hale getirilmesidir. |

#### 3.2.1.1. $I^2C$ haberleşmesinin gerçekleşmesi

İki senaryo için basit olarak $I^2C$ haberleşmesi şu şekillerde gerçekleşir.

1- Mikrodenetleyici A'nın mikrodenetleyici B'ye bilgi göndermek istediği düşünüldüğünde,

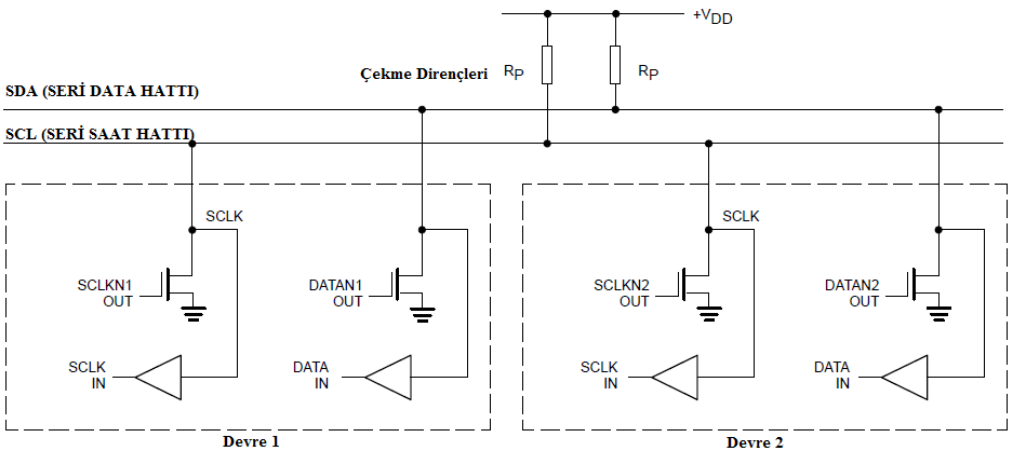
- Mikrodenetleyici A (Yönetici (Master)) mikrodenetleyici B'ye (Yönetilen (Slave)) adres atar.
- Mikrodenetleyici A (Yönetici-Gönderici) mikrodenetleyici B'ye (Yönetilen-Alıcı) veriyi gönderir.
- Mikrodenetleyici A veri iletişimini sonlandırır.

2- Mikrodenetleyici A mikrodenetleyici B'den bilgi almak istediği düşünüldüğünde,

- Mikrodenetleyici A (Yönetici (Master)) mikrodenetleyici B'ye (Yönetilen (Slave)) adres atar.
- Mikrodenetleyici A (Yönetici-Alıcı) mikrodenetleyici B'den (Yönetilen-Gönderici) veriyi alır.
- Mikrodenetleyici A veri iletişimini sonlandırır [25].

### 3.2.1.2. $I^2C$ genel karakteristikleri

$I^2C$ haberleşmesinde SDA ve SCL hatlarının her ikisi de iki yönlü çalışan hatlardır ve Şekil 3.10'da görüldüğü gibi çekme dirençleri ile pozitif gerilim kaynağına bağlanırlar. Hatlarda iletişim olmadığında (boşta beklerken), her iki hatta yüksek (HIGH) seviyededir. Hatta bağlanacak elemanların çıkış uçlarının, SDA ve SCL wired-AND fonksiyonunu gerçekleştirebilmesi için açık-oluk (open-drain) veya açık-kollektör (open-collector) olması gerekmektedir. Bus'a bağlanacak eleman sayısı bus'ın kapasitesi ile sınırlıdır (400 pf) [25]. Şekil 3.10'da $I^2C$ elemanlarının hatta bağlanışları gösterilmektedir [26].




Şekil 3.10: Standart ve hızlı moddaki $I^2C$ elemanlarının Bus'a bağlanışı

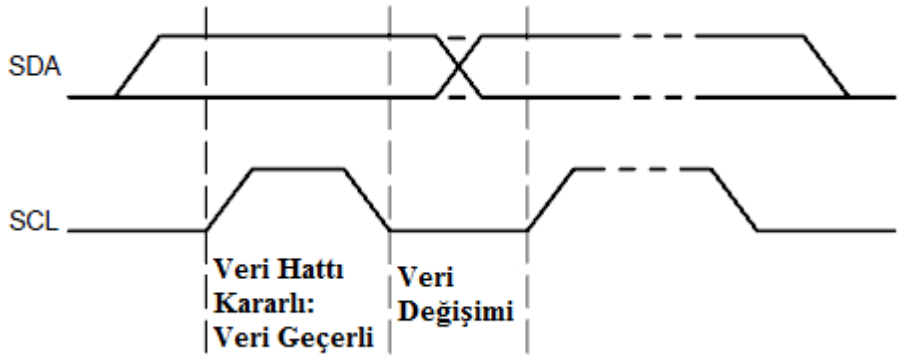
### 3.2.1.3. Bit iletimi

$I^2C$ bus'a bağlı elemanlar CMOS, NMOS veya bipolar gibi değişik teknolojilere sahip olabileceğinden hattaki lojik-0 (LOW) ve lojik-1 (HIGH) sinyal seviyeleri sabit değildir ve kaynak gerilimi (Vdd) seviyesine göre değişmektedir. Her bir veri bitinin iletimi için bir saat sinyali üretilir [26].

### 3.2.1.4. Bilginin geçerliliği

$I^2C$'de verinin geçerli olabilmesi için, saat sinyalinin yüksek periyodu süresince, SDA hattındaki verinin kararlı durumda olması gerekmektedir. Saat sinyalinin düşük

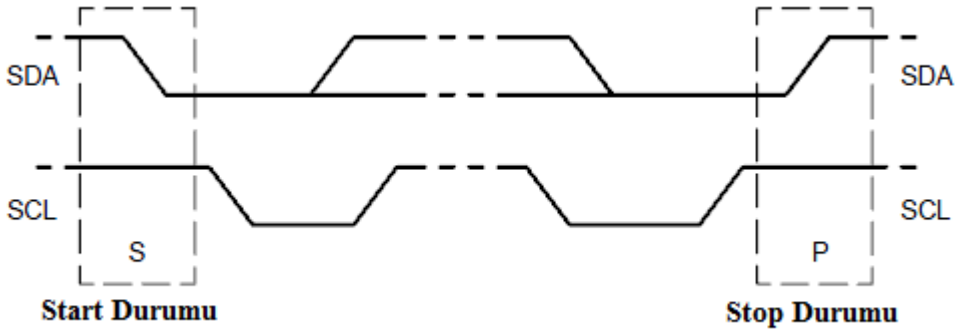
olduğu süre boyunca SDA hattı, düşük veya yüksek konumlarını değiştirebilirler. Şekil 3.11'de $I^2C$ bit iletişimi görülmektedir [26].




Şekil 3.11: $I^2C$ bit iletişimi

### 3.2.1.5. START ve STOP durumları

Şekil 3.12'de $I^2C$'nin bu özel durumları görülmektedir. SDA hattı yüksek seviyeden (HIGH) düşük seviyeye (LOW) geçerken, saat sinyali yüksek seviyede olursa, bu durum START durumunu belirtmektedir [26].




Şekil 3.12: $I^2C$ START ve STOP durumları

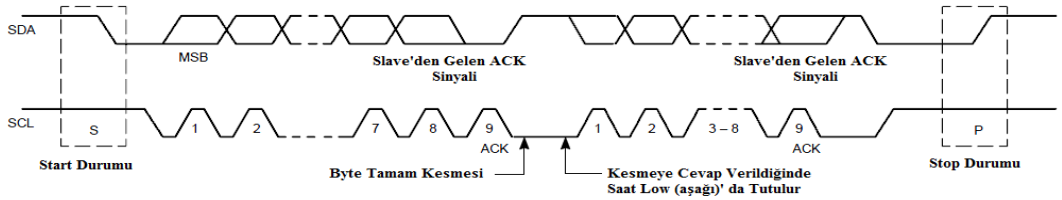
SDA hattı düşük seviyeden yüksek seviyeye geçerken, saat sinyali yüksek seviyede kalırsa bu durum da STOP durumunu göstermektedir (Şekil 3.12).

START ve STOP durumları daima master tarafından üretilir. START durumundan sonra hattın meşgul olduğu, STOP durumundan sonra ise hattın boş olduğu değerlendirilir. Ayrıca $I^2C$'de STOP durumu yerine tekrar START durumu gönderilebilir.

Gerekli arayüz donanıma sahip iletim hattına bağlı elemanların, START ve STOP durumlarını algılaması kolay olmaktadır. Böyle bir arayüzü olmayan elemanlar, geçişi hissedebilmeleri için SDA hattını her saat periyodunda en az iki kere örneklemek durumundadır [25].

### 3.2.1.6. Veri iletimi

#### 3.2.1.6.1. Byte formatı

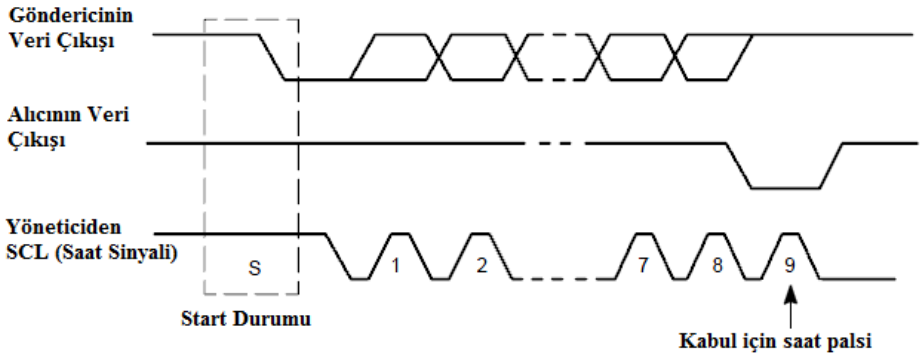



Şekil 3.13: $I^2C$'de veri iletimi

$I^2C$'de iletilecek her veri 8-bit uzunluğunda olmalıdır. İletilecek bayt sayısında bir sınırlama yoktur. İletilen her bir bayt'ı kabul biti (ACK) takip eder. Veri iletimine ilk olarak MSB bitinden başlanır. Eğer slave başka bir işten dolayı (harici kesme cevabı) baytların tamamını alamıyor veya gönderemiyorsa bu durumda saat, master tarafından düşük (LOW) seviyesinde tutulur ve slave hazır hale geldikten sonra iletilir. Bazı durumlarda, $I^2C$ formatından başka formatlarında kullanılmasına müsaade edilir. Bu durumda her bir bayt iletim süresince (CBUS uyumlu elemanlar için) START bitiyle başlar ve STOP bitiyle son bulur. Bu arada NACK (non-acknowledge) biti üretilir (Şekil 3.13) [26].

### 3.2.1.7. Kabul (ACK) Sinyali

$I^2C$ haberleşmesinde veri iletişiminin gerçekleşmesi için kabul sinyali (ACK) kullanmak zorunludur. Kabul için gerekli saat sinyali yönetici tarafından üretilir. Gönderici, kabul saat sinyali süresince SDA hattını yüksek seviyeye çekerek hattı serbest bırakır. Alıcı kabul saat sinyali süresince SDA hattını düşük seviyeye çeker. (Şekil 3.14) Kabul saat sinyalinin yüksek periyodu süresince SDA hattı kararlı düşük seviyededir [26].

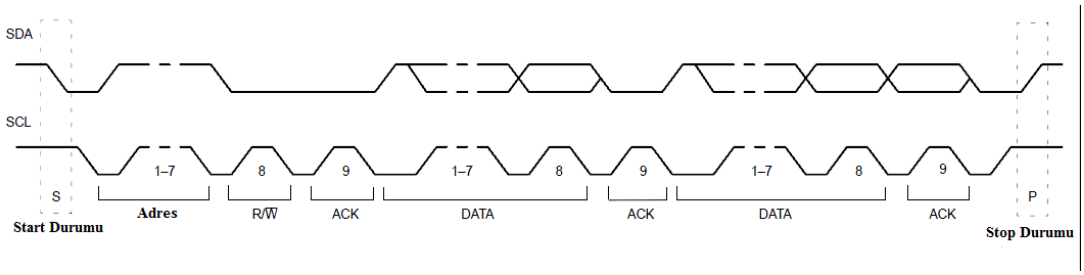



Şekil 3.14: $I^2C$ kabul sinyali

Adreslenmiş olan alıcı, her bir baytı aldıktan sonra ACK üretmek zorundadır. Slave, yönetilen adres baytından sonra ACK göndermezse, SDA hattı yönetilen tarafından yüksek seviyeye çekilir ve veri iletimi gerçekleşmez. Bu durumda master iletimi bitirmek için ya STOP durumu, ya da yeni iletim için tekrarlı-START (Repeated START (Sr)) durumu üretir.

Eğer slave-alıcı, masterın ürettiği slave−adrese ACK gönderir, fakat iletimden bir süre sonra veri iletimine cevap vermezse (ACK üretmezse) master iletimi bitirir. Bu slave cihazın ilk bayttan sonra NACK sinyali ürettiğini göstermektedir. Bu durumda slave, veri hattını yüksek seviyeye çeker ve master STOP veya tekrarlı−START üretir. Bir master−alıcı bilgi iletimine katılmışsa, slave−göndericiye gelen bilginin sonunda bir ACK göndererek işaretlemelidir.

### 3.2.1.8. 7 Bit adresleme ile formatlar




Şekil 3.15: I2C'de bütün bir veri iletimi

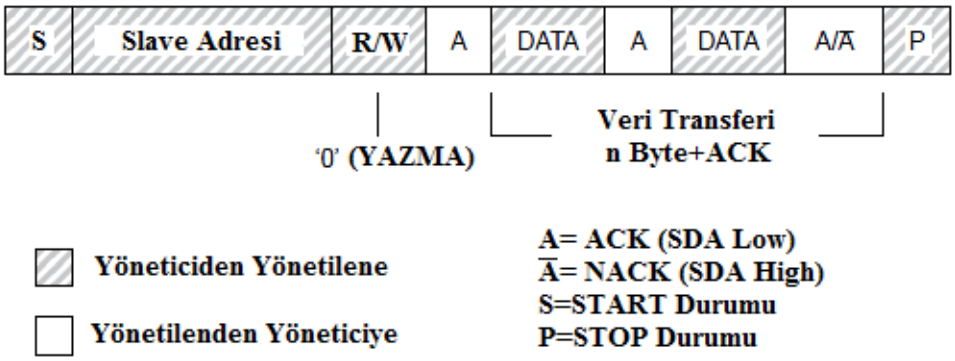
$I^2C$ veri iletimi Şekil 3.15'de görülmektedir [26]. START (S) durumundan sonra, bir slave adresi gönderilir. 7 bit uzunluğundaki bu bitlerden hemen sonra veri yön biti olan R/W biti gelmektedir. Bu bitin '0' olması yazma işlemi, '1' olması ise okuma işlemi yapılacağını göstermektedir. Veri iletimi daima master tarafından üretilen STOP (P) durumu ile sonlandırılmaktadır. Bununla birlikte eğer master iletimi devam ettirmek isterse tekrarlı−START durumu üretebilir ve STOP durumunu üretmeden bir başka slave'i adresleyebilir.

İletim sırasında R/W formatlarının değişik kombinasyonları vardır. Mümkün olan veri iletim formatları şöyledir [26].

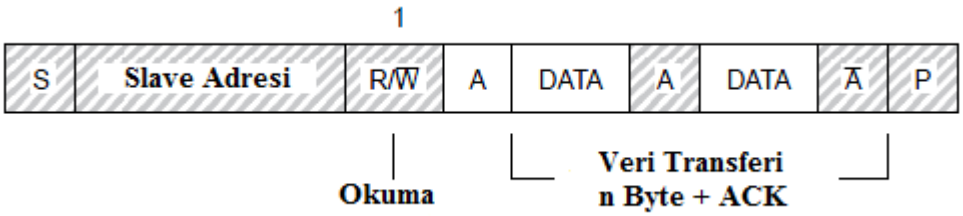
- Master−Gönderici (Transmitter), slave−alıcıya (Receiver) gönderir. Bu sırada iletimin yönü değiştirilemez.

- Master ilk bayttan sonra slave'i okur. İlk ACK'den sonra master−gönderici, master−alıcı olur. Slave−alıcı ise slave−gönderici olur. İlk ACK sadece slave tarafından üretilir. STOP durumu, NACK bitinden sonra master tarafından üretilir.

- Şekil 3.18'de birleşik format görülmektedir [26]. İletim yönünün değişiminde, START durumu ve slave adreslemesi tekrarlanır, R/W biti ise terslenir. Eğer master−alıcı tekrarlı−START durumu gönderirse öncesinde NACK biti gönderir.




Şekil 3.16: Master−göndericinin slave−alıcıyı adreslemesi

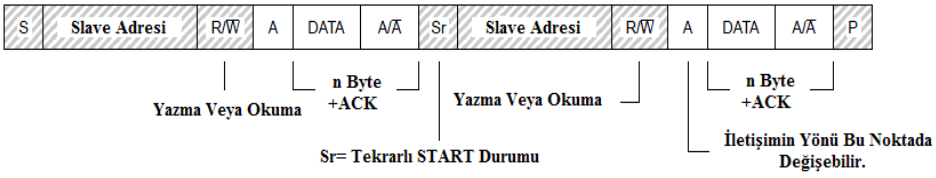



Şekil 3.17: Master'ın ilk bayttan sonra slave'i okuması

- Bileşik format seri hafızayı kontrol etmek için kullanılabilir. Bu durumda ilk veri baytında dahili hafıza adresi yazılmalıdır. START durumu ve slave adresinden sonra veri iletimi yapılabilir.

- Hafıza ile iletişimde hafıza adresleri otomatik olarak bir arttırılabilmekte veya azaltılabilmektedir. Bu işlem, ilgili $I^2C$ elemanını üreten firma tarafından belirlenir.

- Her baytı A veya \A ile gösterilen ACK biti takip eder.

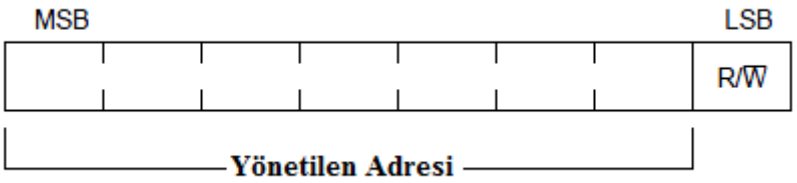



Şekil 3.18: Bileşik format

### 3.2.1.9. 7 Bit Adresleme

$I^2C$ adresleme işlemi genellikle, START durumundan sonra gönderilen bayt ile belirlenir. Master tarafından gönderilen bu bayt ile hangi slave'in seçileceği belirlenir. Genel çağrı (General Call) adresi (00000000) bunların dışındadır ve hatta bağlı olan tüm elemanları adresleyebilir. Teoride, bu adres kullanıldığında, bütün elemanlar ACK ile cevap verirler. Bununla birlikte iletim hattına bağlı elemanlar bu adresi görmezden de gelebilirler [26].

#### 3.2.1.9.1. İlk Bayttaki Bitlerin Tanımlanması




Şekil 3.19: START durumundan sonraki ilk bayt

İlk baytın ilk 7 biti slave adresini oluşturur. 8. bit LSB bitidir ve iletilecek mesajın yönünü belirler (Şekil 3.19). Bu bitin '0' olması, master'ın ilgili slave'e yazma işlemi yapacağını, '1' olması ise ilgili slave'den okuma yapılacağını gösterir. Adres gönderildiğinde sisteme bağlı tüm elemanlar START durumundan sonraki 7 bit ile kendi adreslerini karşılaştırırlar. Eğer adresler birbirine eşitse, bu eleman master tarafından R/W bitine bağlı olarak master−alıcı veya slave−gönderici olacak şekilde seçilir.

### 3.2.2. Soket programlama için kullanılan yazılım

Tez çalışmasında PERL dili, soket programlama ve XMLRPC yöntemi ile bilgisayardan MDA320 veri edinim borduna referans sinyalini gönderme amacıyla kullanılmıştır.

Perl, NASA'da sistem yöneticisi olarak çalışan dil bilimci Larry Wall tarafından geliştirilmiş bir programlama dilidir. Bu programlama dili yoğun şekilde metin işleme ve görüntü tanıma söz konusu olduğunda kullanılabilecek en güçlü ve pratik programlama dilidir. Yirmi iki yıldır geliştirilen ve özgür yazılım çerçevesinde kaynak kodu açık olarak sunulan Perl programlama dili hemen hemen tüm işletim sistemlerinde çalışmaktadır.

Perl ismi bir kısaltma olmayıp açılımı yoktur. Bu yüzden PERL olarak yazılmaz. Ancak Perl kelimesine karşılık olarak daha sonradan çeşitli açılımlar teklif edilmiştir. Bunların en çok bilineni "Pratik Çıkarım ve Raporlama Dili" dir (Practical Extraction and Report Language) [27].

### 3.2.3. Arayüz için kullanılan yazılım

MATLAB (MATrix LABoratory – Matris Laboratuarı), temel olarak teknik ve bilimsel hesaplamalar için yazılmış yüksek performansa sahip bir yazılımdır. 1970'lerin sonunda Cleve Moler tarafından yazılan Matlab programının tipik kullanım alanlarından birkaçı aşağıda sunulmaktadır:

- Algoritma geliştirme ve kod yazma yani programlama,
- Matematiksel (nümerik ve sembolik) hesaplama işlemleri,
- Doğrusal cebir, istatistik, Fourier analizi, filtreleme, optimizasyon, sayısal integrasyon vb. konularda matematik fonksiyonlar,
- 2D ve 3D grafiklerinin çizimi,
- Modelleme ve benzetimi,
- Grafik oluşturma,
- Veri analizi ve kontrolü,
- Gerçek dünya şartlarında uygulama geliştirme,

  şeklinde özetlenebilir.

MATLAB, matematik−istatistik, optimizasyon, sinir ağları, bulanık mantık, işaret ve görüntü işleme, kontrol tasarımları, yöneylem çalışmaları, tıbbi araştırmalar, finans ve uzay araştırmaları gibi çok çeşitli alanlarda kullanılmaktadır. MATLAB,

kullanıcıya hızlı bir analiz ve tasarım ortamı sağlar [28]. Avantajlarından bazıları aşağıda sunulmaktadır;

- Matlab programı kodu C/C++ diline dönüştürebilir,
- 20. dereceden bir denklemin köklerini bulabilir,
- 100x100 boyutlu bir matrisin tersi alınabilir,
- Bir elektrik motoru gerçek zamanda kontrol edilebilir,
- Bir otobüsün süspansiyon benzetimi yapılabilir.

### 3.2.4. Düğümler içerisindeki işletim sistemi ve programlama dili

Tez çalışmasının Kablosuz algılayıcı ağ kısmını Crossbow firmasının ürettiği MDA320 veri edinim bordu (Data Acquisition Board) ve MIB520 programlama kartı oluşturmaktadır. Bu cihazların her biri ağı oluşturan düğümlerdir. MDA320 veri edinim bordu ve MIB520 cihazları, MPR2600 cihazları ile tümleşik olarak kullanılırlar. MPR2600 cihazları üzerinde 128 KB kod belleği, 4 KB veri belleği, 4 KB EEPROM ve 16 MHz ATMEGA128L işlemci, IEEE 802.15.4 standardı ile uyumlu Chipcon CC2420 devresi bulunmaktadır. Bu devreyi kullanarak 2,4 GHz frekans bandında 250 Kbps hızında iletişim yapabilmektedirler [18].

Düğümlerin içerisinde TinyOS işletim sistemi bulunmaktadır. TinyOS açık kaynak kodlu bir işletim sistemidir ve bileşen tabanlıdır. TinyOS programları yazılım bileşenleri üzerine kurulmakta ve bu bileşenler düğümlerin donanımlarını kontrol etmektedir. Yani MPR2600 cihazı içerisine uygun yazılım yüklenerek MDA320 veri edinim bordunun donanımları kullanılabildiği gibi, MPR2600 devresi içerisine uygun kod yüklenerek MIB520 programalama kartı ile tümleşik hale getirilip erişim noktası olarak da kullanılabilmektedir. TinyOS işletim sisteminde, klasik işletim sistemlerinden farklı olarak, çekirdek ve kullanıcı uygulamaları diye bir ayrım bulunmamaktadır. Bu yapı, programın tamamının analizinin ve eniyilemenin daha etkin bir şekilde yapılabilmesine olanak sağlar. İşletim sisteminin çekirdeği sınırlı boyutta kod ve veri içerir. TinyOS statik bellek yönetimini kullanır. Böylece dinamik bellek yönetimlerinden oluşabilecek bellek sızıntısı da engellenmiş olur [29].

TinyOS işletim sistemi, güç tüketimini azaltmak ve düğümlerin çalışma ömrünü arttırmak için "acele et ve uyu" diye bilinen bir çalışma stratejisi izler. Bu strateji, olabildiğince az güç harcamak için mikrodenetleyicinin mümkün olduğunca uyuması prensibine dayanır. Görev, olabilecek en kısa sürede bitirildikten sonra, işlemci en az güç harcayacağı uyku modunda bekletilir. Ek olarak, uygulamaları sonsuz döngülerde bekletmek yerine kesmeler veya zamanlayıcı/sayıcı fonksiyonları kullanılır ve işlem süresini kısaltarak güç tüketimini azaltmayı hedefler. Bunların gerçekleştirilebilmesi için, nesneye yönelik programlama yöntemi kullanılır [29].

Bileşenler birbirlerine arayüzler ile bağlanmaktadır. Arayüz ve bileşenler paket haberleşmesi, yönlendirme, algılama, harekete geçirme ve depolama gibi işlemleri gerçekleştirebilmektedir. TinyOS'ta bileşenler arasındaki etkileşim çift yönlüdür. Bir bileşen kullandığı diğer bileşenin komutlarını çağırabilirken, kullanılan bileşen de olayları sinyalleyerek diğer bileşeni gerçekleşen olaylardan haberdar edebilir [30].

Düğümlere yüklenecek programlar genel olarak konfigürasyon ve modül dosyalarından oluşmaktadır. Konfigürasyon dosyası, bileşenleri ve arayüzlerini birbirlerine bağlamayı (wiring) sağlamaktadır. Bağlama işlemi sayesinde çeşitli bileşenler birbirlerinin arayüzlerini yani görevlerini, komutlarını ve olaylarını çağırıp kullanabilmektedirler. Modül dosyasında ise düğüm içerisinde çalışacak ve gerekli işlemlerin yapılacağı kodlar bulunmaktadır [31].

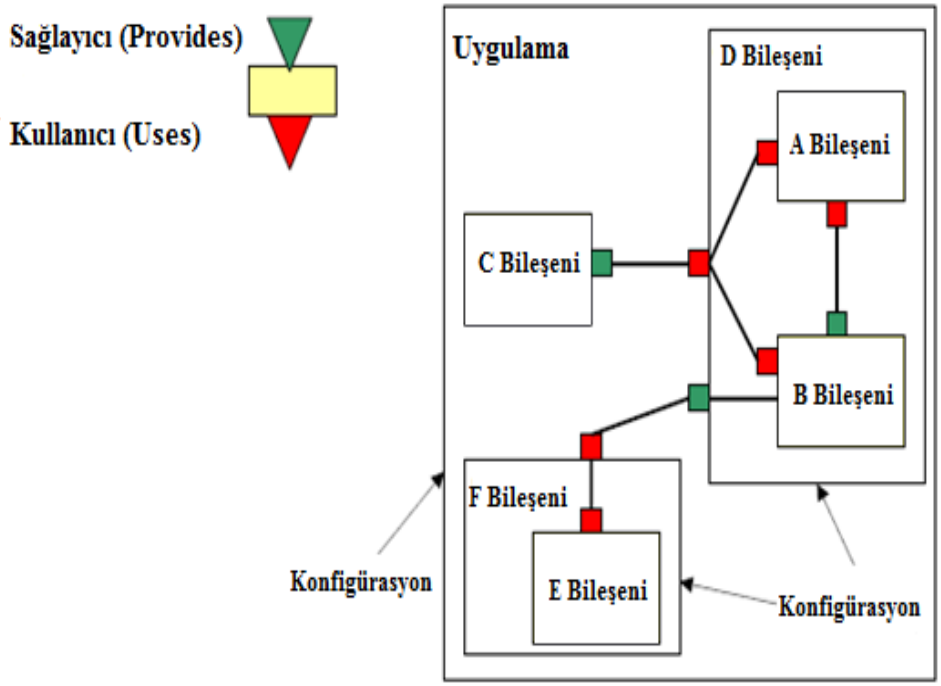
#### 3.2.4.1. İşletim sistemi programlama dili

Bir sistem programlama dili olan nesC, gömülü sistemler için kullanılmaktadır. Basit ve modüler olması, optimize (en uygun şeklide ayarlanması) edilebilmesi ve güçlü bir yapısının olması önemli özelliklerindendir.

Bileşen tabanlı bir dil olan nesC olay tabanlı bir işletim mekanizmasına sahip olan kablosuz algılayıcı ağları için uygun bir programlama dilidir. nesC dilindeki bileşenler nesneye dayalı programlama paradigmasındaki nesnelere benzemektedirler. Bileşenler bir durum bilgisi ve o durum bilgisini işleyen işlevselliği barındırmaktadırlar [30].

#### 3.2.4.1.1. Kullanılan programlama dili uygulama yapısı

Şekil 3.20'da nesC uygulama yapısı görülmektedir [32].




Şekil 3.20: nesC uygulama yapısı

#### 3.2.4.1.2. Konfigürasyonlar Ve Modüller

Konfigürasyonlar, bileşenlerin birbirleri ile olan ilişkilerini içermektedir. Arayüzler ise bir bileşenin başka bileşenler ile olan ilişkilerini düzenlemektedir. Bu ilişkiler hizmet verme veya hizmet alma şeklinde olabilmektedir. Modüllerin birbirleri ile etkileşime geçebilmesini sağlayan bağlama işlemi "configuration" anahtar kelimesi ile tanımlanan yapılandırma bileşenlerince gerçekleştirilir. Aşağıda MyApp uygulaması için oluşturulmuş konfigürasyon ve modül dosyaları görülmektedir.

```
configuration MyApp {
}
implementation
{
  components Main, MyAppM, TimerC, LedsC;
  Main.StdControl -> TimerC.StdControl;
  Main.StdControl -> MyAppM.StdControl;
  MyAppM.Timer -> TimerC.Timer[unique("Timer")];
  MyAppM.Leds -> LedsC.Leds;
}
  module MyAppM {
  provides {
  interface StdControl;
  }
  uses {
  interface Timer;
  interface Leds;
  }
  }
  implementation {
  command result_t StdControl.init() {
  call Leds.init();
  return SUCCESS;
  }
  command result_t StdControl.start() {
  return call Timer.start(TIMER_REPEAT, 1000);
  }
  command result_t StdControl.stop() {
  return call Timer.stop();
  }
  event result_t Timer.fired()
  {
  call Leds.redToggle();
  return SUCCESS;
  }
  }
```

Şekil 3.21: Konfigürasyon ve modül dosyası kodları

StdControl, Leds ve Timer arayüzlerini kullanmaktadır. MyApp uygulamasının yaptığı iş belli bir süre ile düğüm üzerinde bulunan kırmızı ledi terslemektir. Bu işlemleri gerçekleştiren arayüzler Leds ve Timer arayüzleridir. Timer arayüzünde tanımlananan sürenin dolması ile bir olay meydana gelmiş ve bu olay sonucunda

kırmızı ledin yanıp sönmesi sağlanmıştır. Görüldüğü gibi modül, başka bir modül tarafından sinyallenerek aşağıdan yukarıya bir etkileşim sağlanmaktadır ve bu etkileşim “event” anahtar kelimesi ile belirtilen olaylar sayesinde olmaktadır. Leds arayüzü ise sistemdeki ledlerin yakılıp söndürülmesi işlevini yerine getirecek modüller tarafından gerçekleştirilecektir. Modülün “implementation” kısmındaki C kodu gerçekleştirimlerinde MyApp modülü, bu arayüzün ledleri yakan komutunu çağırmaktadır. Yani yukarıdan aşağıya bir etkileşim söz konusudur. Komutlar “command” anahtar kelimesi ile belirtilirler ve çağırımları “call” anahtar kelimesi ile yapılmaktadır. MyAppM modülü hangi arayüzleri kullanacağını kendi gerçekleştiriminde belirtmişti. Ancak bu modülün belirttiği arayüzleri kullanabilmesi için, bu arayüzleri sağlayan modüllere bağlanması gerekmektedir. Modüllerin birbirleri ile etkileşime geçebilmesini sağlayan bağlama işlemi “configuration” anahtar kelimesi ile tanımlanan yapılandırma bileşenlerince gerçekleştirilirler. MyApp yapılandırıcı modülü, MyAppM bileşenin kullandığı arayüzleri sunan StdControl, LedsC ve TimerC.Timer[unique("Timer")] isimli sistem bileşenlerini “->” operatörü ile MyAppM bileşenlerine bağlamaktadır. Bu bileşenler “provides” anahtar kelimesi ile sundukları arayüzleri belirtmişlerdir. Örneğin LedsC bileşeni “provides interface Leds” ile Leds arayüzünün komutlarını gerçekleştirdiğini belirtmektedir. “MyAppM.Leds -> LedsC.Leds;” kod parçacığı ile MyAppM bileşeni ile LedsC bileşeni birbirine bağlanmaktadır. Böylelikle MyApp bileşeni, LedsC bileşeninin Leds arayüzü üzerinden sunduğu komutları çağırabilmektedir. “MyAppM.Timer -> TimerC.Timer[unique("Timer")];” kod parçacığı ile TimerC bileşeninin Timer arayüzü üzerinden sunduğu komutları çağırabilmektedir [30].

#### 3.2.4.2. Algılayıcı düğüme kod yükleme aşamaları

Algılayıcı düğüme kod gömme işlemi sırasında şu aşamalar izlenmiştir.

- /MoteWorks/apps/tutorials/ klasörü içine bir alt klasör açılmalı. İlk uygulamanın klasör ismi MyApp seçildi.
- Yapılacak bir uygulama için hazırlanan klasör içerisinde en az 5 tane dosyası olması gerekmektedir [33].

- ✓ Makefile
- ✓ Makefile.component
- ✓ nesC'de yazılmış uygulama konfigürasyonu
- ✓ nesC'de yazılmış uygulama modülü
- ✓ Readme dosyası

#### 3.2.4.2.1. Makefile oluşturma

Makefile oluşturmak için PN2 programında boş bir çalışma dosyası açılarak içine aşağıdaki kodlar yazılmıştır. Makefile çalışma sırasında ve uygulamanın çalışma adımları içerisinde diğer çalışma dosyalarına olan bağlılıkları içerir. MoteWorks içerisindeki belirli uygulamaların alt klasörleri içerisindeki Makefile'lar aynıdır (/xmesh,/xsensor,/examples,/general,/tutorials). Aşağıdaki kodda uygulamada ihtiyaç duyulacak diğer dosyalar belirtilmektedir [33].

```
include Makefile.component
include $(TOSROOT)/apps/MakeXbowlocal
include $(MAKERULES)
```

"include $(TOSROOT)/apps/MakeXbowlocal" komut satırı düğümlerin kullanacağı grup ID numarası, Rx (Sinyal Alım) ve Tx (Sinyal Gönderme) frekansları ve RF iletişim gücünü içerir.

Yukarıdaki kodlar yazıldıktan sonra çalışma dosyası oluşturduğumuz MyApp klasörünün içine kaydedilmelidir. Makefile içinde uygulamada sırasında kullanılacak diğer dosyaların isimleri bulunmaktadır. Kayıt şu şekilde yapılmalıdır [33].

| | |
|---|---|
| File name | Makefile |
| Save as type | All files (".") |
| File format | No change to file format |

Şekil 3.22: Makefile kayıt formatı

#### 3.2.4.2.2. Makefile.component oluşturma

Yeni bir boş çalışma belgesi açılarak içerisine aşağıdaki kodlar yazılmıştır. Bu dosya içinde Moteworks düzeni içerisindeki belirli uygulamalar için belirli bağlılıklar gösterilmektedir. Aşağıdaki kodda kullanılacak bileşen isminin ne olacağı ile derleme yapılacak algılayıcı kart isimleri belirtilmiştir [33].

```
COMPONENT=MyApp
SENSORBOARD=mts400
```

Yukarıdaki kodlar yazıldıktan sonra çalışma dosyası oluşturduğumuz MyApp klasörünün içine kaydedilmelidir. Bu dosya içerisinde kullanılacak sensör bord tanımlanmıştır. Kayıt şu şekilde yapılmalıdır [33].

| | |
|---|---|
| File name | Makefile.componet |
| Save as type | All files (“.”) |
| File format | No change to file format |

Şekil 3.23: Makefile.component kayıt formatı

#### 3.2.4.2.3. Üst düzey konfigürasyonların oluşturulması

MyApp içerisindeki kodlar aşağıdaki gibidir. Tüm uygulamalar üst düzey konfigürasyon dosyası gerektirir. MyApp.nc içerisinde MyAppM.nc içerisindeki modüllere ve diğer bileşenlere bağlantılar yapılmaktadır. Tanımlamalar parantez içindeki kısımlarda yapılmaktadır (used veya provide arayüzler gibi). Buradaki konfigürasyon bloğu içerisinde herhangi bir tanımlama yapılmamıştır. MyApp içerisindeki diğer bir kod bloğu “implementation” (gerçekleme) bloğudur. Burada bileşenlerin birbiri ile bağlantısı gösterilmektedir [33].

```
configuration MyApp {
}
implementation {
components Main, MyAppM, TimerC, LedsC; // Uygulama içerisinde
kullanılacak bileşenler tanımlanmıştır.
Main.StdControl -> TimerC.StdControl; // Main.StdControl interface'i ile
TimerC.StdControl interface'i birbirine bağlanmıştır.
Main.StdControl -> MyAppM.StdControl; // Main.StdControl ile MyAppM
içerisindeki Std.Control interface'i bağlanmıştır.
MyAppM.Timer -> TimerC.Timer[unique("Timer")]; // MyAppM içerisindeki
Timer interface'i ile daha önce Nesc içerisinde tanımlı bulunan Timer interface'i
birbirine bağlanmıştır.
MyAppM.Leds -> LedsC.Leds; // MyAppM içerisindeki Leds interface'i ile
daha önce Nesc içerisinde tanımlı bulunan Leds interface'i birbirine
bağlanmıştır.
```

Şekil 3.24: Konfigürasyon dosyası kodları açıklamaları

Yukarıdaki kodlar yazıldıktan sonra çalışma dosyası oluşturduğumuz MyApp klasörünün içine kaydedilmelidir. Her uygulama için en az bir tane StdControl ara yüzü tanımlanmalıdır. StdControl arayüzü hizmet (provides) arayüzü olup TinyOS işletim sisteminin temel fonksiyonlarını içerir. 3 kısmı bulunmaktadır. Bunlar initialized, started ve stopped dir. Yazılan koddaki son iki satır uygulama modülündeki LedsC ve TimerC modüllerine bağlantı sağlar. Bu modüllerin içindeki TimerC ve LedsC fonksiyonları ile Timer ve LED devreleri kontrol edilebilir. Dosyanın kaydı şu şekilde yapılmalıdır [33].

| | |
|---|---|
| File name | MyApp.nc |
| Save as type | All files (“.”) |
| File format | No change to file format |

Şekil 3.25: Konfigürasyon dosyası kayıt formatı

**3.2.4.2.4. Module oluşturma**

Module dosyası içine uygulamanın programlama kodları da yazılmıştır. Yani uygulamanın program kodları ile modül kodları aynı yerdedir. Bu uygulamada kırmızı led belli bir zamana göre terslenecektir (Toggle).

Uses (Kullanıcı) ara yüzüne ilişkin önemli bir notta şudur. Uses arayüz olarak tanımlanan arayüzlerin alt fonksiyonlarıda özel bir tanımlamaya gerek kalmadan açıkça çağrılabilir. Örneğin MyAppM modülü içerisindeki Leds arayüzünün Leds.init() fonksiyonu MyAppM.init() olarak açıkça çağrılabilir. Ok işaretleri arayüzler arasındaki ilişkileri belirlerler. ” -> “ şeklinde gösterilmektedir. Okun sağ tarafında bir arayüz, sol tarafında ise bir gerçekleştirme (implementation) vardır. Diğer bir deyişle, uses (kullanıcı) arayüz solda, provides (sağlayıcı) arayüz sağdadır.

```
MyAppM.Timer -> TimerC.Timer[unique("Timer")];
```

Kod satırında sol tarafta bulunan MyAppM.Timer, Timer (/MoteWorks/tos/interfaces/timer.nc) arayüzüne atıfta bulunurken, sağ tarafta bulunan TimerC.Timer ise Timer gerçeklemesine atıfta bulunmaktadır. nesC aynı arayüzün çoklu uygulamalarını destekler. Aynı arayüz as ifadesi kullanılarak birçok kere kullanılabilir. Örneğin Timer arayüzü gibi. Kullanımı aşağıdaki gibi olabilir.

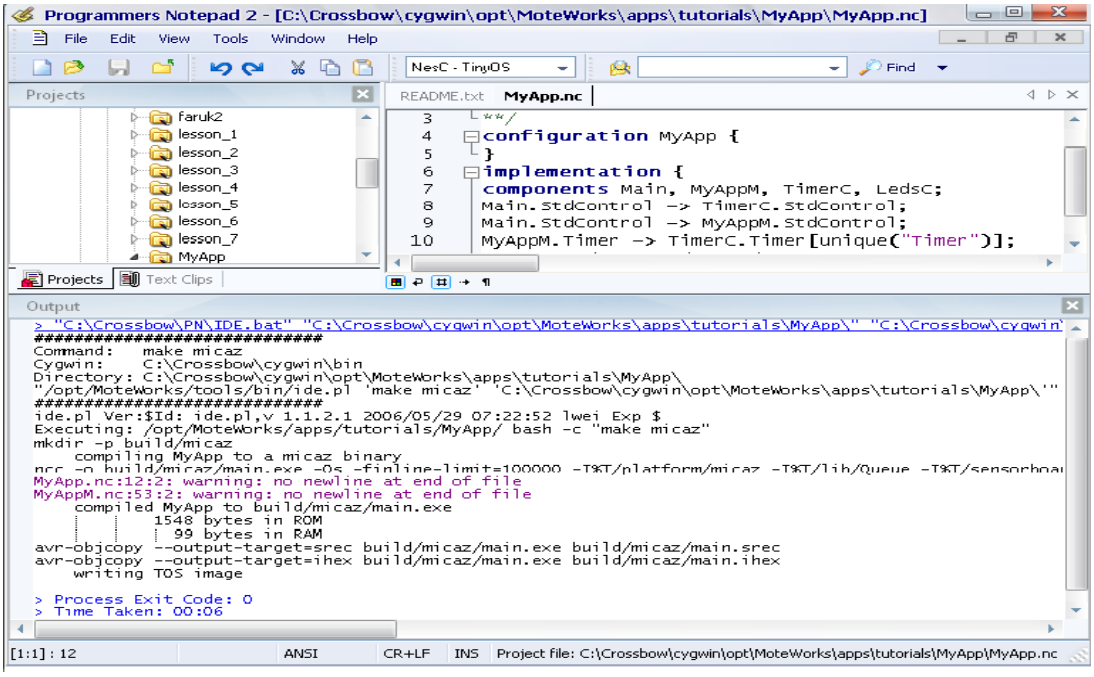
```
interface Timer;
interface Timer as TO_Timer;
```

| | |
|---|---|
| File name | MyAppM.nc |
| Save as type | All files (“.”) |
| File format | No change to file format |

Şekil 3.26: Module dosyası kayıt formatı

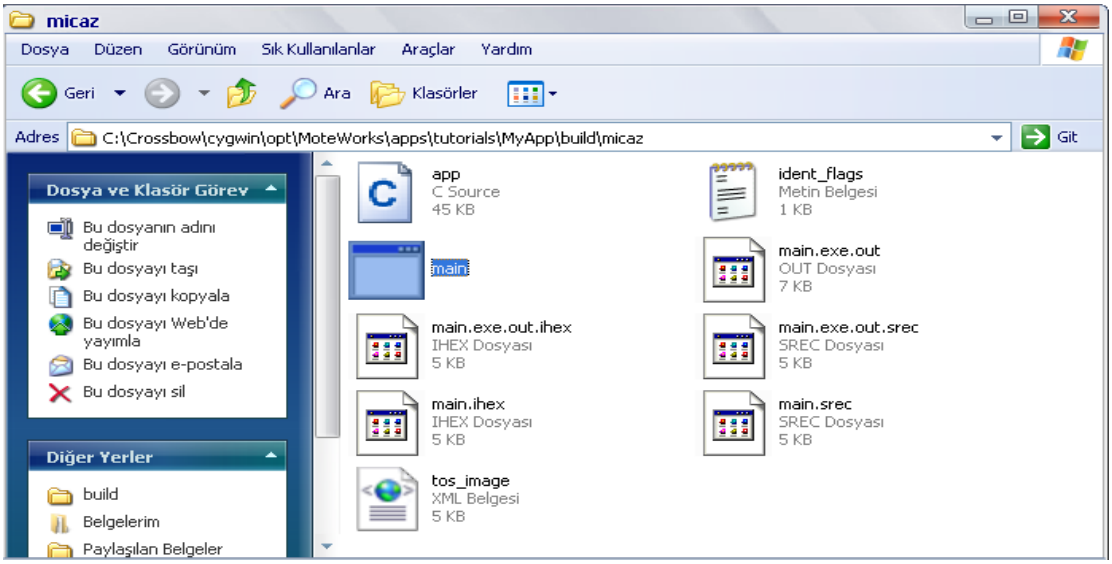
### 3.2.4.3. Yazılan kodların derlenmesi

Yukarıda yazılan kodların algılayıcı düğüme yüklenebilmesi için derlenmesi gerekmektedir. Bunun için; PN2 (Programmers NotePad−2) programının Tools menüsünden > make micaZ seçilmelidir.(Bizim düğümlerimiz micaz olarak geçtiği için make micaZ seçilmiştir.) Uygun seçim yapıldıktan sonra program derleme işlemine başlamaktadır. Görünüm Şekil 3.27’deki gibi oluşmaktadır.




Şekil 3.27: Derleme işlemi

Output kısmında "Writing TOS image" yazısının görülmesi derlemenin başarılı olduğu anlamına gelmektedir. Derleme sonucu oluşan dosya main.exe dosyasıdır. Ve bu dosya derlenen dosyaların bulunduğu klasörün içinde derleme sonucu oluşan build/micaz/ alt klasörlerinin içerisindedir. Görünümü aşağıdaki şekildeki gibidir.



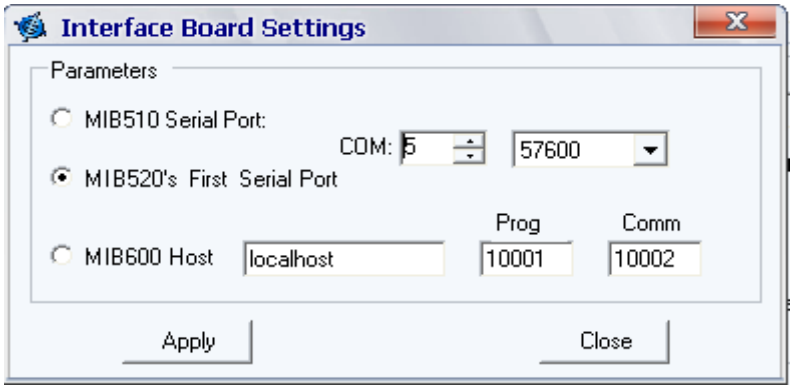

Şekil 3.28: Derleme sonrası dosyaların bulunduğu kısım

### 3.2.4.4. main.exe dosyasının düğüme yüklenmesi

Program kodunun algılayıcı düğüme yüklenmesi için MIB520 USB programlayıcının bilgisayara bağlanması gerekmektedir. MIB 520 bilgisayara bağlandığında iki tane sanal seri port açılmaktadır. Bu portlar COM5 ve COM6'dır. (Erişim noktası bağlandığında COM3 ve COM4 sanal seri portları açılmaktadır.) Bu portlardan COM5 programlama için kullanılmaktadır. COM6 ise haberleşme için kullanılmaktadır.

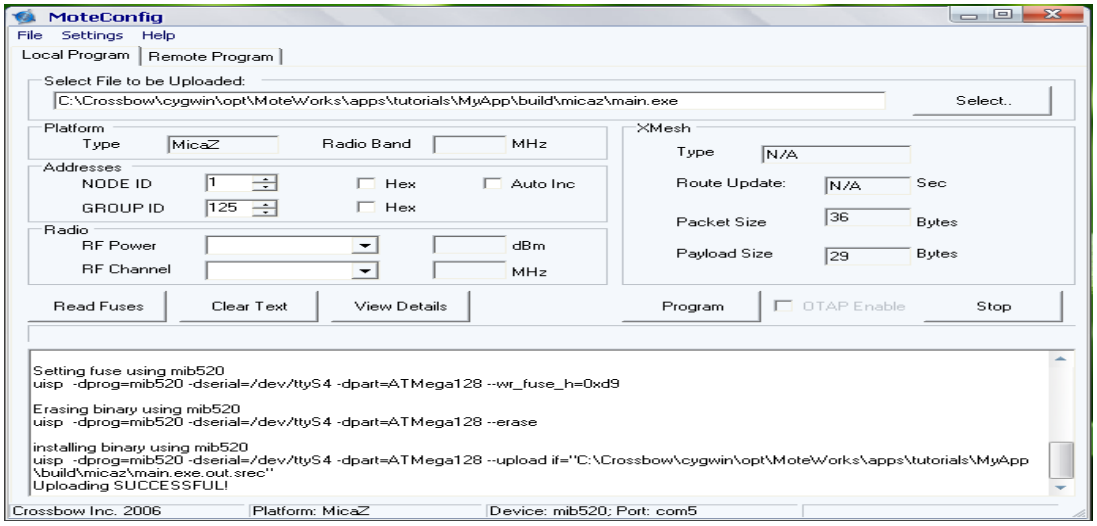
Yükleme yapılabilmesi için MTS400 algılayıcı düğümü aşağıdaki şekilde gösterilen MICA−series connector kısmına bağlanmaktadır. MIB 520 üzerinde güç led'inin yanında 3 tane test ledi bulunmaktadır. Bu ledler programın çalışıp çalışmadığı hakkında fikir vermektedir.

MIB520 bağlantısı yapıldıktan sonra yükleme programı olan MoteConfig programı çalıştırılmalıdır. MoteConfig programı açıldıktan sonra programda ara yüz ayarlarının yapılması gerekmektedir. Bunun için Setting menüsünden İnterface Board seçeneği seçilmelidir. Ayarlar aşağıdaki gibi yapılmalıdır.




Şekil 3.29: MoteConfig ayarları

Daha sonra yüklenecek dosya seçilir. Dosya ile ön yükleme yapılır. Ön yükleme bittikten sonra program tuşuna basılarak program algılayıcı düğüme yüklenir. Yükleme durumu
Şekil 3.30'teki gibidir.



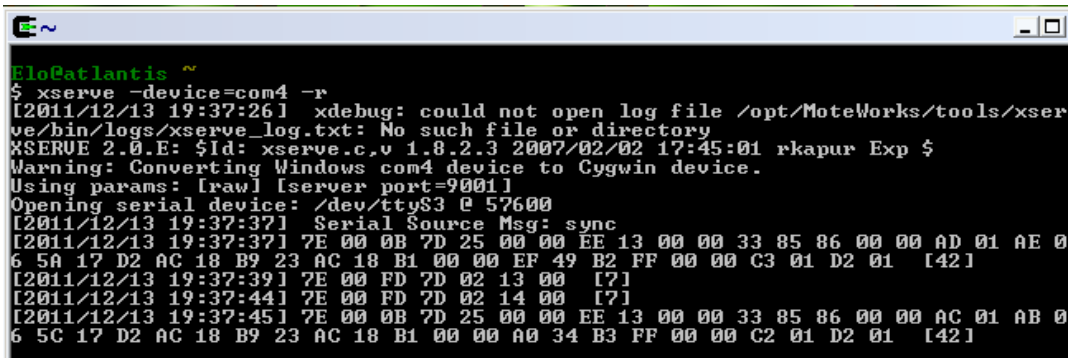

Şekil 3.30: Dosyanın düğüme yüklenmesi işlemi

### 3.2.5. XSERVE ağ geçidi

XSERVE mesh kablosuz ağları ve kurumsal uygulamalar ağı ile etkileşimi arasında birincil ağ geçidi olarak hizmet vermektedir. Yüksek düzeyde hizmet XML tabanlı konfigürasyon dosyaları ve yüklenebilir eklenti modülleri kullanılarak özelleştirilebilir [35].

XSERVE'de veriler 3 şekilde yönetilir.

- Satır (Raw) Format: Satır formatında verinin her baytı 16lık sayı sistemine göre 1 çift sayı şeklinde görüntülenir [34]. Satır formatlı verilerin görünümü Şekil 3.31'de görülmektedir.




Şekil 3.31: Satır formatlı veri görünümü

- Çözümlenmiş (Parsed) Format: Çözümlenmiş formatta veriler ham olarak 16'lık sayı sistemine göre gösterilir. Verilerin yanına ölçülen özel değerleri yazılır. Çözümlenmiş formatlı verilerin görünümü Şekil 3.32'de görülmektedir.

```
Elo@atlantis ~
$ xserve -device=com4 -p
[2011/12/13 19:47:37]  xdebug: could not open log file /opt/MoteWorks/tools/xser
ve/bin/logs/xserve_log.txt: No such file or directory
XSERVE 2.0.E: $Id: xserve.c,v 1.8.2.3 2007/02/02 17:45:01 rkapur Exp $
Warning: Converting Windows com4 device to Cygwin device.
Using params: [parsed] [server port=9001]
Opening serial device: /dev/ttyS3 @ 57600
[2011/12/13 19:47:40]  Serial Source Msg: sync
[2011/12/13 19:47:40] amtype=0xfd,
[2011/12/13 19:47:41] MTS400 [sensor data converted to engineering units]:
   health:          node id = 0x13ee
   battery:                 = 0x1ad mv
   humid:                 = 0x6d6%
   Temperature:             = 0x1781 degC
   IntersemaTemperature:    = 0x00 degC
   IntersemaPressure:       = 0x4897 mbar
   Light :                  = 0xffb2 lux
   X-axis Accel:            = 0x1c2 mg
   Y-axis Accel:            = 0x1c1 mg
```

Şekil 3.32 : Çözümlenmiş formatlı veri görünümü

- Çevrilmiş Format: Her değer ham değerinden ölçü birimine uygun şekilde dönüştürülmüş halde gösterilir. Çevrilmiş formatlı verilerin görünümü Şekil 3.33'de görülmektedir.

```
Elo@atlantis ~
$ xserve -device=com4 -c
[2011/12/13 19:53:37]  xdebug: could not open log file /opt/MoteWorks/tools/xser
ve/bin/logs/xserve_log.txt: No such file or directory
XSERVE 2.0.E: $Id: xserve.c,v 1.8.2.3 2007/02/02 17:45:01 rkapur Exp $
Warning: Converting Windows com4 device to Cygwin device.
Using params: [converted] [server port=9001]
Opening serial device: /dev/ttyS3 @ 57600
[2011/12/13 19:53:41]  Serial Source Msg: sync
[2011/12/13 19:53:41] MTS400 [sensor data converted to engineering units]:
   health:          node id = 5102
   battery:                 = 2898 mv
   humid:                 = 54%
   Temperature:             = 21 degC
   IntersemaTemperature:    = -168.931244 degC
   IntersemaPressure:       = 738.263733 mbar
   Light :                  = 59.570000 lux
   X-axis Accel:            = 0.000000 mg
   Y-axis Accel:            = 0.000000 mg
[2011/12/13 19:53:41] amtype=253,
```

Şekil 3.33: Çevrilmiş formatlı veri görünümü

### 3.2.5.1. XSERVE veri sunuş formatları

XSERVE'de veriler 4 yol ile sunulur [34].

- Çıktı (Print): Tüm veriler XSERVE terminal ekranında görüntülenir.

- Dosya Çıkışı (Export File): Her veri paketi kendi özel veri dosyasına çıkartılır. Her paket dosyaya tek bir satır olarak girilir ve her alan sınırlı bir özel sınırlayıcı kullanıcı tarafından kullanılır.

- Veri Günlüğü (Data Logging): Her veri paketi, kendine ait özel veri tabanı tablolarında saklanır. Her paket tabloda tek bir satırda temsil edilir. Paket ya yeni bir satır olarak eklenir ya da mevcut bir satırı güncellemek için kullanılabilir.

- XML Veri Akışı (XML Stream): Her veri paketi XML belge olarak sunulur. XML belgesi XML soket bağlantısı üzerinden dinlenerek yayınlanmaktadır. XML veri akışı için XSERVE ekranında yazılan parametreler ve anlamları şu şekildedir.

➢ –xmlr: Satır (raw) formatlı XML veri çıkışı olur.
➢ –xmlp: Çözümlenmiş (parsed) formatta XML veri çıkışı olur.
➢ –xmlc: Çevrilmiş (converted) formatta XML veri çıkışı olur.

#### 3.2.5.2. Tez çalışması için kullanılan parametreler

Tez çalışmasına yönelik olarak XSERVE terminaline yazılan parametreler ve anlamları aşağıda belirtilmektedir. Çalışma ekranındaki;

xserve -device=COM4 -h –c -xmlc –xmlport=9002 ifadesindeki parametrelerin anlamları şu şekildedir.

-device=COM6 : Erişim noktası ile bilgisayarın hangi sanal portu üzerinden haberleşileceğidir. Açılan sanal portlardan büyük numaralı olanı ile haberleşilir. Çalışılan bilgisayarda COM6 portu büyük numaralı olarak açılan porttur.

-h: XSERVE' in Web arayüzüne erişim için yazılır. Web sunucusu üzerinden veri görüntülenir.

-c: Ekranda verilerin çevrilmiş formatta görüntülenmesi için yazılmıştır.

-xmlc: Çevrilmiş formatta XML veri çıkışı olması için yazılmıştır.

-xmlport=9002: XML sunucusunun port numarasıdır.

#### 3.2.5.3. XCommand

XCommand ile, uzaktan düğümlerin çalışma parametreleri değiştirilebilir, durum değişkenleri sorgulanabilir, düğümler üzerinde bulunan led veya buzzer gibi devreler çalıştırılabilir veya durdurulabilir ya da bir düğümün örnekleme hızı değiştirilebilir. Bu işlemler XCommand'in ara yüzü Xserveterm ile gerçekleştirilmektedir. Tablo 3.5'te Xcommand ile yapılabilecek işlemler ve tanımlamaları bulunmaktadır [34].

Tablo 3.5: XCommand kategorileri ve tanımlamaları

| **Komut Kategorisi** | **Komut** | **Parametreler** | **Tanımlamalar** |
|---|---|---|---|
| Güç Yönetimi | RESET<br>SLEEP<br>WAKEUP | | Uyuma ve Uyanma zamanlarının resetlenmesi için. |
| Temel Oranları Güncelleştirme | SET_RATE | | Oranları alır veya günceller. set_rate komutu cihazın veri edinim oranını değiştirir.İlk parametre milisaniye cinsinden yeni veri aralığını belirtir. |
| Cihaz Konfigürasyon Parametreleri ve Ayarları | GET_CONFIG<br>SET_NODEID<br>SET_GROUP<br>SET_RF_POWER<br>SET_RF_CHANNEL | | Bu komut setleri radyo frekansı ve gücünün değerinin alınması, ayarlanması, ve radyo kanallarını içerir. |
| Çalıştırma | ACTUATE<br>- SET_LED<br><br>- SET_SOUND<br><br>- SET_RELAY | <br>0= OFF, 1=ON, 2=TOOGLE<br><br>0=OFF, 1=ON<br><br>0=OFF, 1=ON | <br>Bu komut seti ile her led ayrı ayrı düzenlenebileceği gibi üçü birden çalıştırabilir.<br><br>Bu komut ses devresinin açar veya kapatır.<br><br>Bu komut röleleri açar veya kapatır. |

### 3.2.5.4. XserveTerm kullanımı

XSERVE çalışırken CYGWIN programı tekrar çalıştırılarak yeni bir çalışma penceresi açılır. Açılan yeni çalışma penceresindeki komut satırına aşağıdaki formatta kod yazılması gerekmektedir.

xserveterm –server=<host:port>

Yukarıdaki formata uygun olarak yazılmış kod Şekil 3.34'de görülmektedir.

```
Elo@atlantis ~
$ xserveterm -server=localhost:9003

XServTerm:>
```

Şekil 3.34: XserveTerm çalışma komutu

XserveTerm çalıştıktan sonra yapılabilecek işlemler listesi, komut ekranına help komutu yazılmasıyla ekrana dökülür. Şekil 3.35'de ekran görüntüsü görülmektedir.

```
~
Available Commands:
   set_timeout <timeout ms>
   set_starting_sequence <sequence number>
   set_default_group <group id>
   get_config  <destination address>
   set_rate <destination address> <new rate>
   set_nodeid <destination address> <new node id>
   set_groupid <destination address>  <new group id>
   set_rfchannel <destination address>  <new rf channel>
   set_rfpower <destination address>  <new rfpower>
   sleep <destination address>
   wake <destination address>
   reset <destination address>
   xserve.shutdown
   custom_action <destination address>  <type>  <value>
                 type[1:bias | 2:threshold | 3:N | 4:T]
   actuate <destination address> <device> <state>
                 device ids              states
                 ----------              ------
                 green led       0       off      0
                 yellow led      1       on       1
                 red led         2       toggle   2
                 all leds        3
                 sounder         4
                 relay1          5
                 relay2          6
                 relay3          7

XServTerm:>
```

Şekil 3.35: XserveTerm ile yapılabilecekler listesi

### 3.2.5.5. XserveTerm komutları ve kullanım örnekleri

#### 3.2.5.5.1. get_config <destination address>

Hedef adresi (destination address) kısmına yazılan aktif durumdaki düğümün konfigürasyon ayarlarını gösterir. Şekil 3.36'da 5109 düğümüne ait bilgilerin yer aldığı örnek kod ekran çıktısı görülmektedir.

```
XServTerm:> get_config 5109
SUCCESS: Command executed successfully. (SUCCESS:Success)
NODE ID: 5109
UID: 01A247F51000000D
GROUP ID: 125
RADIO CHANNEL: 25
RADIO POWER: 31
```

Şekil 3.36: get_config komutu

#### 3.2.5.5.2. set_rate <destination address> < new rate>

Hedef adresi (destination address) kısmına yazılan aktif durumdaki düğümün örnekleme süresini değiştirir. Yazılacak değer milisaniye cinsindendir. Örnek kod Şekil 3.37'deki gibidir.

```
XServTerm:> set_rate 5109 1000
SUCCESS: Command executed successfully. (SUCCESS:Success)
NODE ID: 5109
SAMPLE RATE: 1000
```

Şekil 3.37: set_rate komutu

İlgili kod sonucunda 5109 düğümünün yeni örnekleme hızı 1000 ms (1 saniye) olmuştur. İlgili değişiklik Xserve kısmından takip edilebilmektedir.

#### 3.2.5.5.3. set_nodeid <destination address> <new node id>

Hedef adresi (destination address) kısmına yazılan aktif durumdaki düğümün NODE ID'si değiştirilir. Örnek kod Şekil 3.38'deki gibidir.

```
XServTerm:> set_nodeid 5109 5100
SUCCESS: Command executed successfully. (SUCCESS:Success)
NODE ID: 5100
```

Şekil 3.38: set_nodeid komutu

İlgili kod sonucunda 5109 numaralı düğümün ID'si 5100 olarak değiştirilmiştir. Bundan sonra bu düğümle yapılacak işlemlerde 5100 bilgisi kullanılmalıdır.

#### 3.2.5.5.4. set_groupid <destination address> <new group id>

Hedef adresi (destination address) kısmına yazılan aktif durumdaki düğümün bağlı olduğu grup numarası değiştirilir. Örnek kod:

set_groupid 5100 145

Benzer görevlere sahip iki komut daha vardır. Bunlar hedef adresi yazılan düğümün radyo frekansı (Rf) kanalını ve gücünü değiştiren komutlardır. Formatları aşağıdaki gibidir.

set_rfchannel <destination address> <new rf channel>
set_rfpower <destination address> <new rfpower>

Üstteki üç komutu kullanmak pek sağlıklı olmayabilir. Hedef adresteki düğümde yapılacak bu değişiklikler ile ilgili düğüm ile erişim noktası arasındaki bilgiler uyuşmayacağından hedef adresteki düğümden bilgi alınamamaktadır.

#### 3.2.5.5.5. sleep <destination address>

Hedef adresi (destination address) kısmına yazılan aktif durumdaki düğümü uyku moduna geçirir. Uyku modunda radyo işlemleri durmaz, sadece örnekleme uygulamaları durur. Şekil 3.39'da örnek kod görülmektedir.

```
XServTerm:> sleep 5109
SUCCESS: Command executed successfully. (SUCCESS:Success)
```

Şekil 3.39: sleep komutu

Bu komutu yazdıktan sonra 5109 ID numaralı düğümden artık bilgi gelmemektedir.

#### 3.2.5.5.6. wake <destination address>

Hedef adresi (destination address) kısmına yazılan aktif durumdaki düğümü uyku modunda ise çıkarır. Şekil 3.40'da örnek kod görülmektedir.

```
XServTerm:> wake 5109
SUCCESS: Command executed successfully. (SUCCESS:Success)
```

Şekil 3.40: wake komutu

Bu komut yazıldıktan sonra 5109 numaralı düğüm uykudan uyanarak baz istasyonuna tekrar bilgi göndermeye devam eder.

#### 3.2.5.5.7. reset <destination address>

Hedef adresi (destination address) kısmına yazılan aktif durumdaki düğümü resetler. Şekil 3.41'de örnek kod görülmektedir.

```
XServTerm:> reset 5109
SUCCESS: Command executed successfully. (SUCCESS:Success)
```

Şekil 3.41: reset komutu

#### 3.2.5.5.8. xserve.shutdown

Xserve ile ilgili yapılan işlemler sonlandırılır ve çevresel cihazlar ile olan tüm haberleşme durur.

#### 3.2.5.5.9. actuate <destination address> <device> <state>

Hedef adresi (destination address) kısmında belirtilen aktif durumdaki düğüm üzerindeki devrelerin kontrolü yapılır.

Tablo 3.6: actuate komutu devre ayarlamaları

| **Devre** | **Devre Numarası** | **Devre Durumları** |
|---|---|---|
| YEŞİL LED | 0 | OFF=0, ON=1, TOGGLE=2 |
| SARI LED | 1 | OFF=0, ON=1, TOGGLE=2 |
| KIRMIZI LED | 2 | OFF=0, ON=1, TOGGLE=2 |
| TÜM LEDLER | 3 | OFF=0, ON=1, TOGGLE=2 |
| SES DEVRESİ | 4 | OFF=0, ON=1, TOGGLE=2 |
| RÖLE 1 | 5 | OFF=0, ON=1, TOGGLE=2 |
| RÖLE 2 | 6 | OFF=0, ON=1, TOGGLE=2 |
| RÖLE 3 | 7 | OFF=0, ON=1, TOGGLE=2 |

Örneğin : 5100 numaralı düğüm üzerindeki sarı ledi yakmak için gerekli olan kod satırı Şekil 3.42'deki gibidir.

```
XServTerm:> actuate 5100 1 1
SUCCESS: Command executed successfully. (SUCCESS:Success)
```

Şekil 3.42: actuate komutu örnek 1

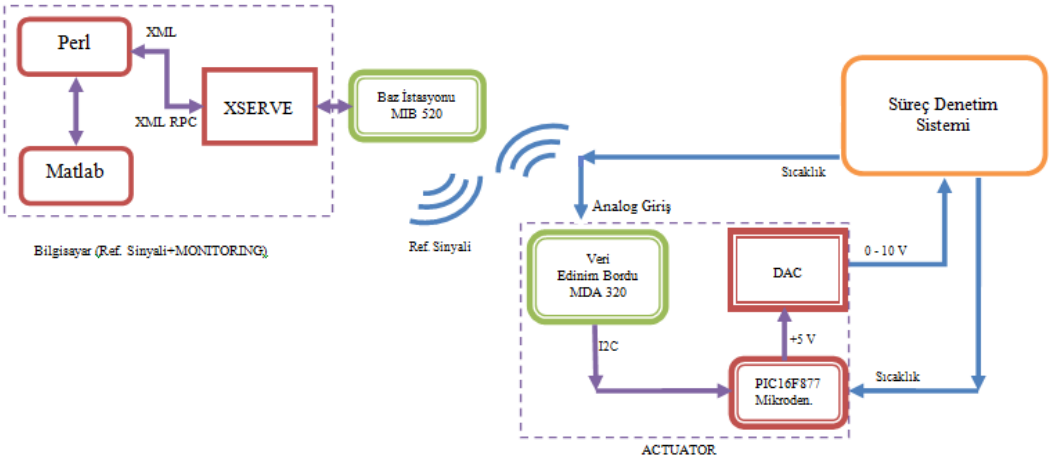
Veya tüm ledleri terslemek için yazılması gerekli kod satırı Şekil 3.43'deki gibidir.

```
XServTerm:> actuate 5100 3 2
SUCCESS: Command executed successfully. (SUCCESS:Success)
```

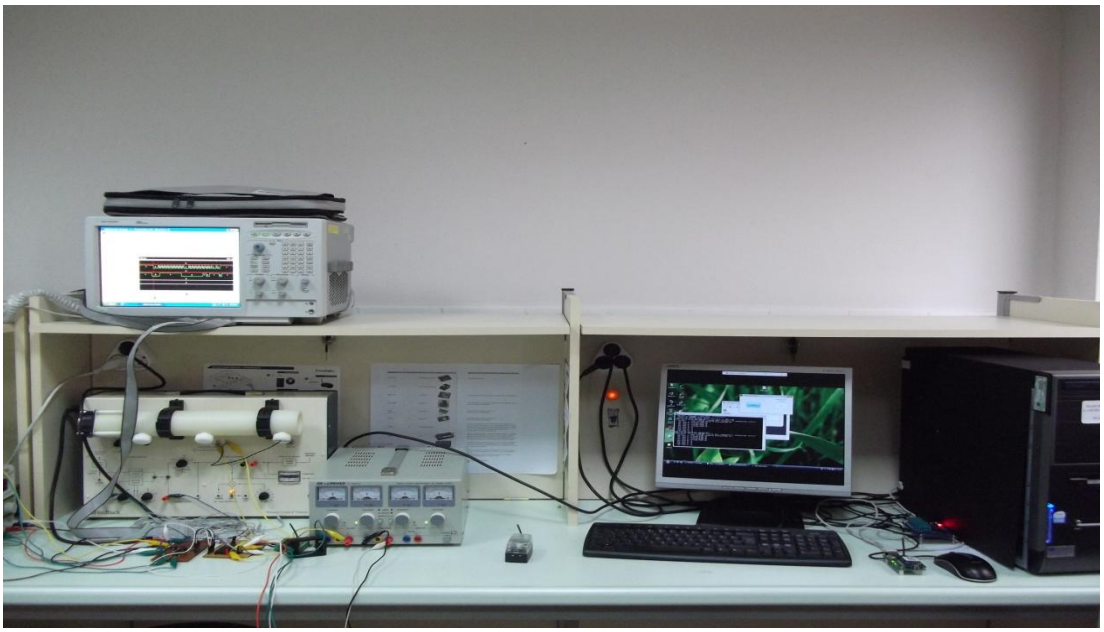
Şekil 3.43: actuate komutu örnek 2

# 4. GELİŞTİRİLEN DENETİM SİSTEMİ TASARIMI

Tez çalışmasında uygulaması gerçekleştirilen sistemin blok şeması Şekil 4.1'de görülmektedir. Şekil 4.2'de ise sistemin çalışması için kurulan düzenek görülmektedir.

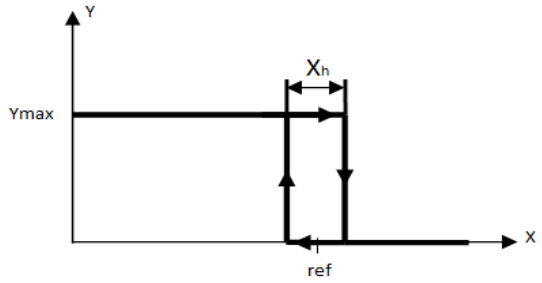



Şekil 4.1: Sistemin blok şeması



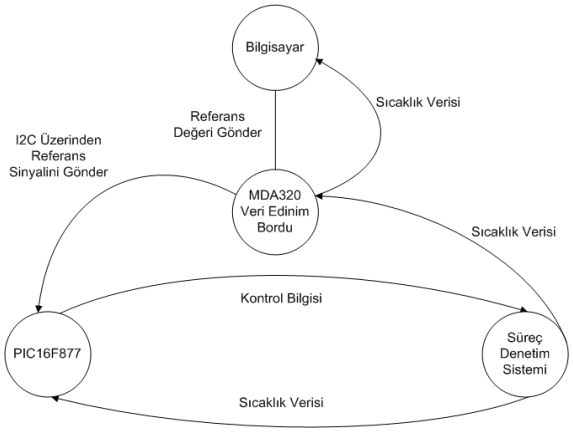
Şekil 4.2: Sistem için kurulan düzenek

Tez çalışmasında sistemin aç−kapa (on−off) denetimi gerçekleştirilmiştir. Aç−kapa denetimde, kontrol cihazı belirlenen referans değerinin üzerinde veya altında ise enerjiyi açar ya da kapatır. Enerji ya tamamen açıktır ya da tamamen kapalıdır. Aç-kapa denetim sıcaklık kontrolü gibi çok hassas olmayan sistemlerde kullanılırlar. Bu tip denetimde en çok karşılaşılan sorun, referans değerine ulaşan sistem çıkışında sürekli bir aç−kapa durumunun oluşması ve bunun kontrol kısmında bulunan mekanik donanıma zarar vermesidir. Bu sorunu aşmak amacıyla sistemin histerezis ile aç−kapa denetimi gerçekleştirilmiştir. Histerezis denetimde referans değeri aşılır aşılmaz enerji kesilmez (off durumu). Daha önceden belirlenen bir değer aşılırsa enerji kesilir. Aynı şekilde referans değerinin altına düşüldüğünde sisteme enerji verilmez (on durumu). Daha önce belirlenen bir değerin altına düşüldüğünde sisteme enerji verilir. Böylelikle aç−kapa durumları geciktirilerek mekanik sistemin sık aç−kapa yapmasının önüne geçilmektedir. Şekil 4.3'te histerezis grafiği görülmektedir [20].



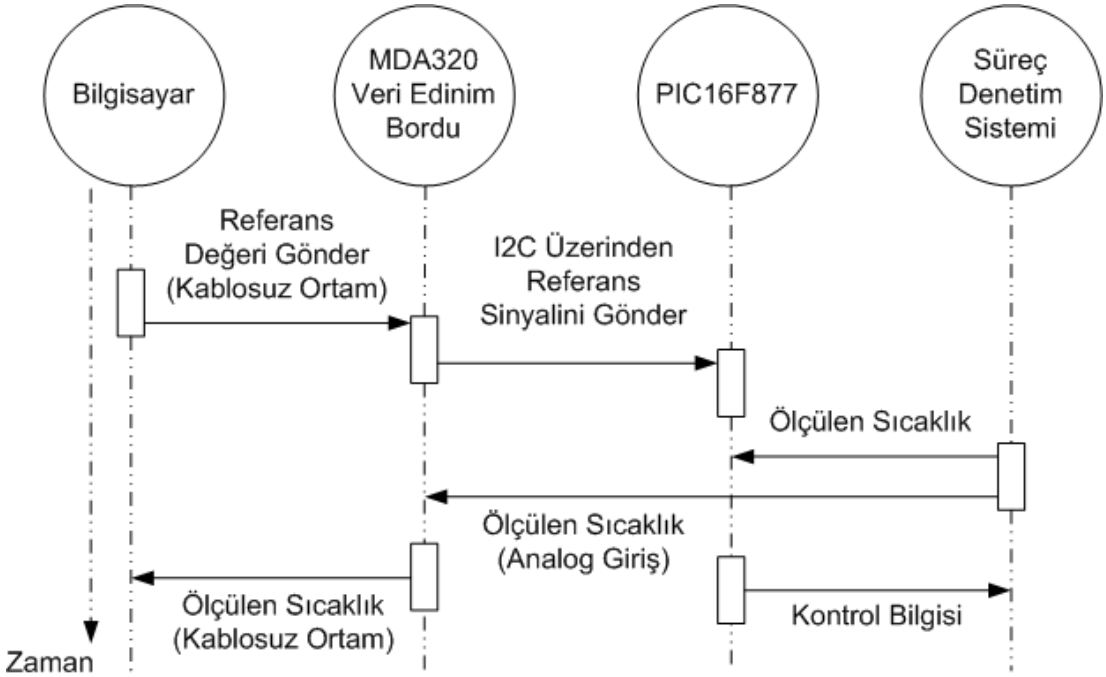

Şekil 4.3: Histerezis grafiği

Sistemin çalışmasını gösteren basit durum diyagramı Şekil 4.4'te görülmektedir.




Şekil 4.4: Sistemin çalışmasının basit durum diyagramı

Gerçekleştirilen sistemde işlemlerin hangi sırayla yapıldığını gösteren zaman akış diyagramı aşağıdaki şekilde görülmektedir.



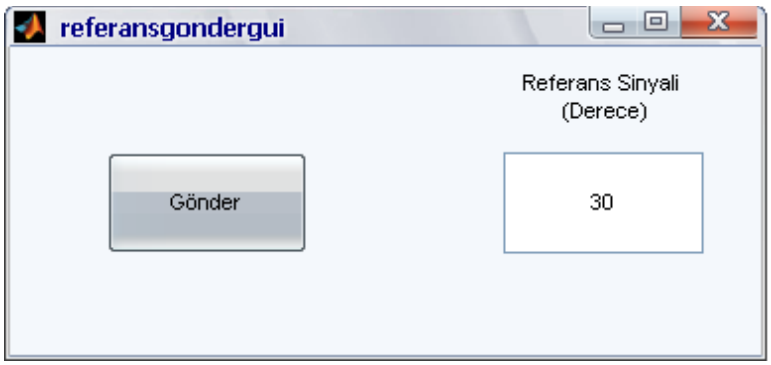

Şekil 4.5: Sistemin zaman akış diyagramı

Geliştirilen sistemde ilk olarak sistemin denetlenmek istediği referans sıcaklık belirlenerek bilgisayar üzerinden MDA320 veri edinim borduna kablosuz olarak gönderilmektedir. MDA320 veri edinim bordundan alınan bu referans bilgisi $I^2C$ haberleşmesi ile PIC16F877 mikrodenetleyicisine gönderilmektedir. Süreç denetim sisteminden alınan analog sıcaklık bilgisi yine mikrodenetleyici tarafından alınarak analog−digital çevirici ile dijital veriye çevrilmektedir. Gönderilen referans değeri ile ölçülen sıcaklık değeri karşılaştırılarak mikrodenetleyici tarafından kontrol bilgisi üretilmekte ve mikrodenetleyicinin dijital çıkışlarına gönderilmektedir. Dijital çıkışlardaki bu bilgi ise dijital−analog çevirici ile analog sinyale çevrilmektedir. Üretilen analog sinyal sistemin harici girişine bağlanarak sistemin denetimi gerçekleştirilmektedir. Aynı zamanda izleme yapılması amacıyla, süreç denetim sistemi üzerindeki algılayıcının çıkışı MDA320 veri edinim bordunun analog girişine bağlanmaktadır. Analog girişten alınan veriler kablosuz ortam aracılığıyla bilgisayara gönderilerek veriler işlenmiş ve çalışmanın grafiği elde edilmiştir. Geliştirilen test düzeneği ile sistemin hem denetlenmesi hem de izlenmesi sağlanmaktadır.

## 4.1. Referans Bilgisinin Gönderilmesi

Sistem çalışmasının ilk adımı, süreç denetim sisteminin denetlenmek istendiği sıcaklık değerinin MDA320 veri edinim borduna gönderilmesidir. Süreç denetim sisteminin denetlenmek istendiği sıcaklık değeri bilgisayar üzerinden MDA320 veri edinim borduna gönderilmektedir.

Referans bilgisinin gönderilmesi işi MATLAB programı üzerinde çalışan grafik kullanıcı ara yüzü (Graphichal User Interface (GUI)) üzerinden yapılmaktadır. Tasarlanmış ara yüz ile sistemin denetlenmek istendiği referans değeri MDA320 veri edinim borduna gönderilmektedir. Şekil 4.6'da MATLAB'da hazırlanmış ara yüz görülmektedir.




Şekil 4.6: Referans değeri gönderme ara yüzü

Örnek olarak sistem 30 derece de denetlenmek istenirse gönderilmesi gereken değer yukarıdaki gibi ayarlanmalıdır. Gönder butonuna basıldıktan sonra alınan bu referans değeri XMLRPC (eXtensible Markup Language Remote Procedure Calls) yöntemi ile MDA320 veri edinim borduna erişim noktası üzerinden gönderilmektedir. Gönderilmek istenen veri PERL dilinde yazılmış olan XCommandCustomAction.pl dosyasına parametre olarak gitmektedir. Bu dosyanın yaptığı iş, XMLRPC yöntemini kullanarak aldığı bu parametreyi veri edinim borduna göndermektir. PERL dili bu çalışmada soket programlama amacıyla kullanılmaktadır. Veri edinim borduna enerji verildikten sonra erişim noktası ile iletişime geçmektedir. Şekil 4.7'de veri edinim

borduyla erişim noktasının iletişime geçtikten sonra veri edinim bordundan gelen veriler görülmektedir.

```
Elo@atlantis ~
$

Elo@atlantis ~
$ xserve -device=com6 -c
[2011/12/14 17:07:35]  xdebug: could not open log file /opt/MoteWorks/tools/xser
ve/bin/logs/xserve_log.txt: No such file or directory
XSERVE 2.0.E: $Id: xserve.c,v 1.8.2.3 2007/02/02 17:45:01 rkapur Exp $
Warning: Converting Windows com6 device to Cygwin device.
Using params: [converted] [server port=9001]
Opening serial device: /dev/ttyS5 @ 57600
[2011/12/14 17:07:38]  Serial Source Msg: sync
[2011/12/14 17:07:38] amtype=253,
[2011/12/14 17:07:38] MDA320 [sensor data converted to engineering units]:
   health:     node id=5109 parent=0 battery=5566 mV
   adc chan 0: voltage=1818 mV
   adc chan 1: voltage=1665 mV
   adc chan 2: voltage=1727 mV
   adc chan 3: voltage=1638 mV
   dig chan 0: =1
   dig chan 1: =1
   dig chan 2: =1
   dig chan 3: =1
```

Şekil 4.7: Veri edinim bordundan gelen bilgiler

MATLAB uygulamasından gelen veriyi MDA320 veri edinim borduna gönderen PERL dilinde yazılmış kodun bir bölümü Şekil 4.8'de görülmektedir.

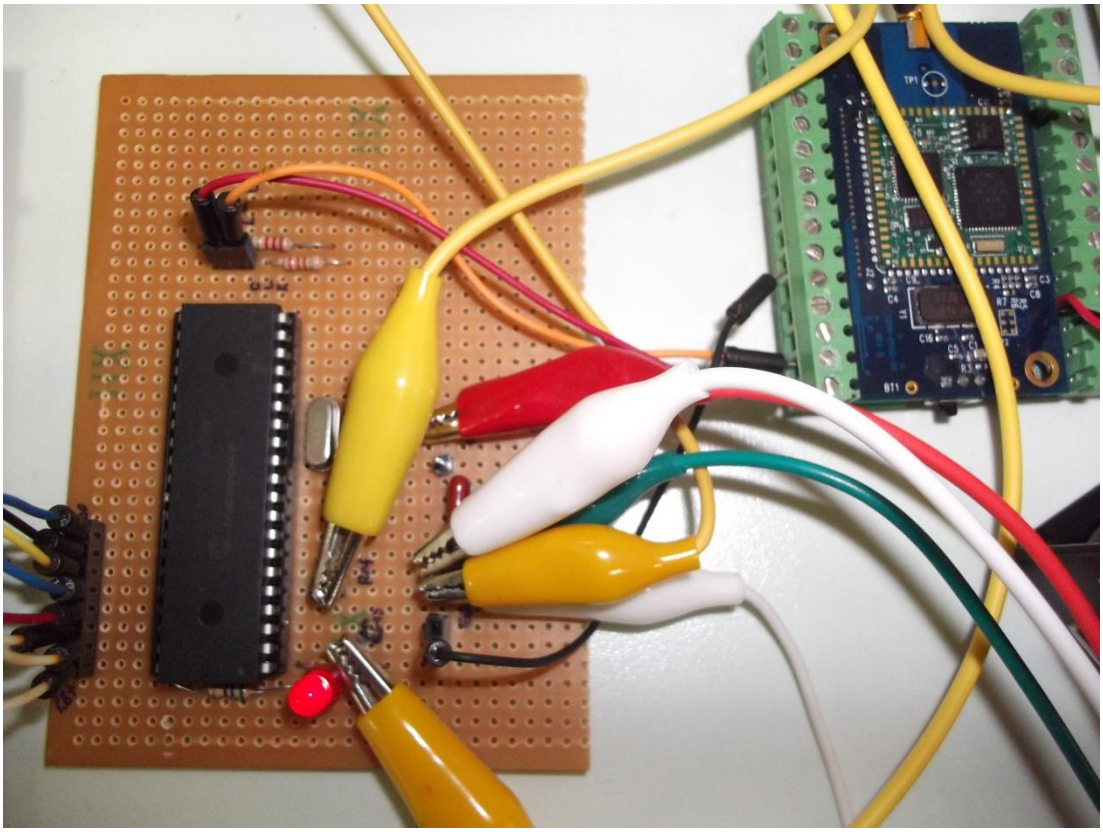
```
XCommandCustomAction.pl
60    $len=length($msg_out);
61    $lenstr=pack("l",$len);
62
63    print STDOUT "mesaj genisligi: $len ve paketlenmis hali $lenstr\n";
64    syswrite(CLIENT,$lenstr,4);
65    syswrite(CLIENT,$msg_out,$len);
66    close CLIENT;
67
68    }
69
70    sub main(){
71
72
73        print STDOUT "ilk parametre: $ARGV[0].\n";
74
75        #my $name=$_[1];
76        #my $type=$_[2];
77
78        $| = 1;                    # enable auto-flushing
79        print "Content-Type: text/html\n\n";
80        my $cmdnum=0;
81        my $val=0;
82        #my $exstr="";
83        my $setstr="";
84        my $cmdstr="";
85        my $tmpstr="";
86        my $xmlmsgstr="<?xml version=\"1.0\"  encoding=\"UTF-8\"?>".
87            "<methodCall><methodName>xmesh.custom_action</methodName><params>".
88            "<param><value><struct><member><name>destAddress</name><value>".
89            "<int>$ARGV[0]</int></value></member><member><name>groupId</name>".
90            "<value><int>125</int></value></member><member><name>type</name>".
91            "<value><int>$ARGV[1]</int></value></member><member><name>value</name>".
92            "<value><int>$ARGV[2]</int></value></member></struct></value></param>".
93            "</params></methodCall>";
94        my $pid=0;
95        my $len=0;
96        my $url="";
97
```

Şekil 4.8: XCommandCustomAction.pl dosyası

## 4.2. Referans Bilgisinin PIC16F877 Mikrodenetleyicisine Gönderimi

Bilgisayar üzerinden veri edinim borduna gönderilen referans bilgisi veri edinim bordunun EEPROM bölgesine yazılmaktadır. Veri edinim bordunun hafıza bölgesine yazılan bu veri $I^2C$ bus hattı üzerindende seri olarak okunabilmektedir.

Veri edinim bordu ile PIC16F877 mikrodenetleyicisi üzerlerinde bulunan $I^2C$ seri haberleşme uçları ile birbirleriyle haberleşmektedir. $I^2C$ haberleşmesi için cihazlarda bulunan SDA ve SCL uçları birbirleri ile birleştirilmiş ve birleştirildikleri noktalar çekme dirençleri ile +5v besleme seviyesine bağlanmışlardır. İletişim bu iki hat üzerinden yapılmaktadır. Geliştirilen sistemde MDA320 veri edinim bordu yönetici (master) olarak, PIC16F877 mikrodenetleyicisi ise yönetilen (slave) cihaz olarak kullanılmaktadır. Bu durumda iletişimin gerçekleşmesi için gerekli olan START, STOP durumlarını ve iletişim için gerekli olan saat sinyallerini veri edinim bordu üretmektedir. PIC16F877 ve MDA320 veri edinim bordunun bağlantısı Şekil 4.9'da görülmektedir.



Şekil 4.9: PIC16F877 ve MDA320 veri edinim bordu $I^2C$ bağlantısı

Veri edinim bordu PIC16F877'yi yönetilen (slave) adresi olarak 16'lık (hexadecimal) sayı sistemine göre 0x3E şeklinde adreslemektedir. Veri edinim bordu 0x3E yönetilen adresini SDA hattına koyduğunda mikrodenetleyici bu adresi kendi adresi ile karşılaştırmakta ve kabul sinyali (ACK) göndermektedir. Kabul sinyalinden sonra mikrodenetleyici veri almaya ve göndermeye hazır hale gelmektedir. Veri edinim bordundan bir baytlık veri gönderimi aşağıdaki şekilde olmaktadır.

Tablo 4.1: Bir bayt veri gönderimi

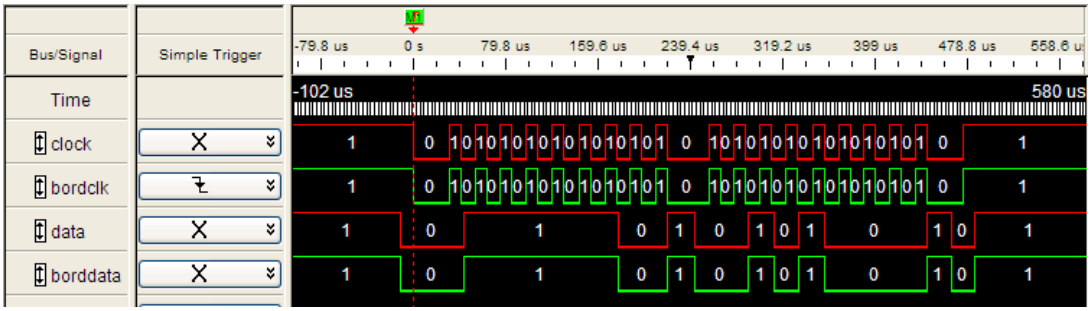
| | | | | | |
|---|---|---|---|---|---|
| Start | Slave Adresi 7 bit=0x3E | R/W 1 bit | Kabul (ACK) | Veri (8 bit) | Stop |

Veri edinim bordu yönetilen adresini gönderirken 7 bit adresleme metodu ile göndermektedir. Buna göre adres bilgisi sola bir bit ötelenir ve sonuna okuma/yazma (read/write) biti eklenir. Eğer yönetilen cihazdan okuma yapılacaksa okuma/yazma biti 1 olarak ayarlanır. Bu durumda START durumundan sonra hatta bırakılan 8 bitlik veri 0x7F olmaktadır. Eğer yönetilen cihaza veri gönderilecekse okuma/yazma biti 0 olarak ayarlanır. Bu durumda START durumundan sonra hatta bırakılan veri 0x7E olmaktadır.

PIC16F877 mikrodenetleyicisinin adresi yazılımsal olarak kendi içerisindeki fonksiyonlar ile ayarlanmaktadır. Adres bilgisi 8 bit olarak kontrol edilmektedir. Bu yüzden adres bilgisi belirlenirken 7 bitlik adres bilgisinin sonundaki okuma/yazma biti de hesaba katılmaktadır.

Yapılan çalışmada veri edinim bordundan mikrodenetleyiciye veri gönderileceği için, mikrodenetleyici içerisinde kendi adresi 0x7E olarak ayarlanmıştır. 0x7E bilgisi geldiğinde mikrodenetleyici kabul sinyali göndermektedir. Kabul sinyalinden sonra veri edinim bordundan gelen veri, veri hattından 8 bitlik veri şeklinde okunarak bir değişkene atanmaktadır.

Veri edinim bordundan mikrodenetleyiciye gönderilen veriler lojik analizör ile izlenmektedir. Seçilen herhangi bir referans değeri için gönderilen verilerin ekran görüntüsü Şekil 4.10'da görülmektedir.

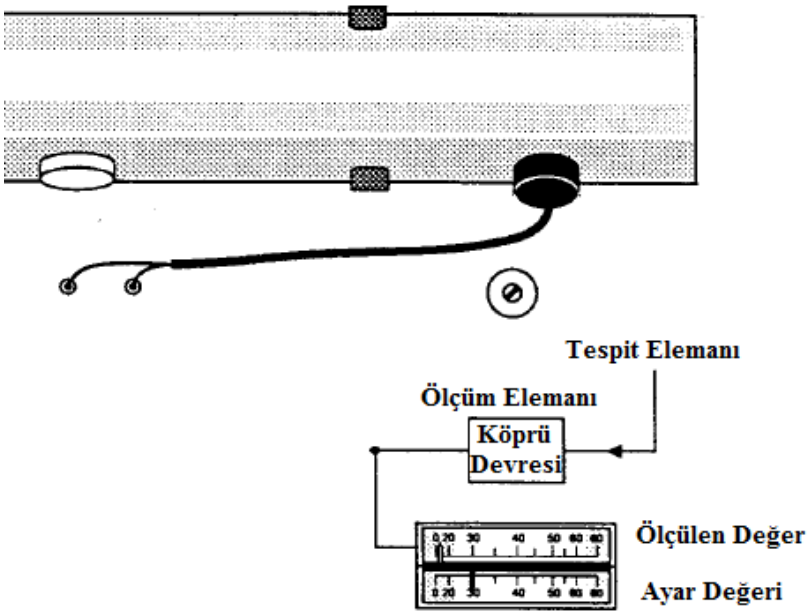



Şekil 4.10: Lojik analizörden $I^2C$ hattı görüntüsü

Şekil 4.10'da görüldüğü gibi ilk olarak SDA hattına master cihaz olan veri edinim bordundan START durumunu ifade eden bit bırakılmaktadır. SCL hattının yüksek seviyede olduğu durumdayken SDA hattının düşük seviyeden yüksek seviyeye geçmesi START durumunu oluşturmaktadır. Daha sonra ise 7 bit slave adresi ve okuma/yazma biti hatta bırakılmaktadır. Veri edinim bordu mikrodenetleyiciye veri göndereceği için okuma/yazma biti 0 olarak gönderilmektedir. 9. saat sinyali ise ACK içindir. ACK da, master SDA hattını 9. saat sinyali boyunca yukarı seviyede (serbest bırakır) tutar. Mikrodenetleyici ise adres bilgisinin uyuşması halinde hattı 9. saat sinyali boyunca aşağı konuma çekerek ACK'yi göndermektedir. ACK'dan sonra referans değeri SDA hattına bırakılmaktadır. Bu değer mikrodenetleyici tarafından alındıktan sonra tekrar ACK gönderilmektedir. Daha sonra ise veri iletimini bitirmek için master olan veri edinim bordu STOP bitini SDA hattına bırakmaktadır. SCL sinyalinin yüksek seviyede olduğu durumdayken, SDA hattının düşük seviyeden yüksek seviyeye çıkması STOP durumunu oluşturmaktadır. Bu şekilde referans sinyali veri edinim bordundan mikrodenetleyiciye $I^2C$ hattı üzerinden gönderilmektedir.

### 4.3. Ölçülen Sıcaklık Bilgisinin Alınması

Geliştirilen test düzeneği ile süreç denetim sistemi denetlenmektedir. Denetleme işlemi ile eğitim seti üzerinde bulunan rezistans açılıp kapatılmaktadır. Rezistansın yaydığı sıcaklık bilgisi deney seti üzerinde bulunan algılayıcı ile algılanmaktadır. Burada algılayıcı eleman negatif katsayılı bir ısıl dirençtir. Bu elemanın uçları bir köprü elemanına bağlanarak ölçme işlemi yapılmaktadır. Tespit elemanı ve ölçüm elemanın bağlanışı Şekil 4.11'de görülmektedir.




Şekil 4.11: Süreç denetim sistemi ölçme bağlantıları

Sistemden gelen sıcaklık bilgisi köprü elemanı uçları üzerinden analog değer olarak okunabilmektedir. Süreç denetim sisteminin ölçüm aralığı 0 ile 80 $^{o}$C arasındadır. Sistemin köprü elemanı çıkışlarında rezinstansın yaydığı sıcaklık değerine ters orantılı olarak gerilim değerleri okunmaktadır. Bunun sebebi tespit elemanının (algılayıcı) negatif katsayılı ısıl direnç olmasından dolayıdır. Köprü elemanı çıkışlarında ölçüm aralığına göre 0−80 $^{o}$C sıcaklık değerlerine karşılık 250 mV ile 400 mV arasında sıcaklık değerleri okunmaktadır. 0 $^{o}$C'de ölçüm elemanı çıkışında 400 mV okunurken 80 $^{o}$C sıcaklık değerinde 250 mV okunmaktadır.

Ölçülen bu sıcaklık bilgisinin geliştirilen sistemde kullanılabilmesi için sayısal veriye çevrilmesi gerekmektedir. Bunun sebebi, bilgisayardan gelen referans değerinin sayısal bir veri olması ve karşılaştırma yapılabilmesi için, ölçülen sıcaklık bilgisininde sayısal veri olması gerekliliğidir.

Gelen sıcaklık bilgisinin analog bilgiden dijital bilgiye çevrilmesi işlemini mikrodenetleyici gerçekleştirmektedir. Bu işlem için PIC16F877 mikrodenetleyicisinin analog−dijital çevirici (ADC) modülü kullanılmaktadır. Bunun avantajı harici bir devre veya entegre kullanmadan analog sinyallerin dahili ADC

modülü sayesinde dijital sinyallere çevrilmesidir. Mikrodenetleyici içerisindeki ADC modülü 10 bitliktir ve $2^{10}$ = 1024 adet değer ile bir analog işareti örnekleyebilmektedir. ADC biriminin elde ettiği dijital bilginin bit sayısı ADC modülünün çözünürlüğünü ifade eder. Çözünürlük ne kadar yüksekse o kadar iyi dönüşüm yapılabilmektedir.

Bilgisayar tarafından gönderilen referans bilgisi 8 bitlik olduğundan değerlerin karşılaştırılabilmesi için mikrodenetleyici içerisindeki ADC modülü de 8 bitlik olarak kullanılmıştır. Mikrodenetleyici 8 bitlik çözünürlükle, $2^8$ = 256 adet farklı değer şeklinde bir analog işareti örnekleyebilmektedir. ADC' nin dijital bilgiye çevireceği sinyalin maksimum gerilim değeri yani belirlenen $V_{REF}$ referans değeri 5 volt ise ADC adım büyüklüğü şu şekilde hesaplanmaktadır.

$$ADC\ Adım\ Büyüklüğü = \frac{Sinyalin\ Maksimum\ Gerilim\ Değeri}{2^{BitSayısı}} \tag{4.1}$$

$$ADC\ Adım\ Büyüklüğü = \frac{5}{256} = 0{,}01953125 \tag{4.2}$$

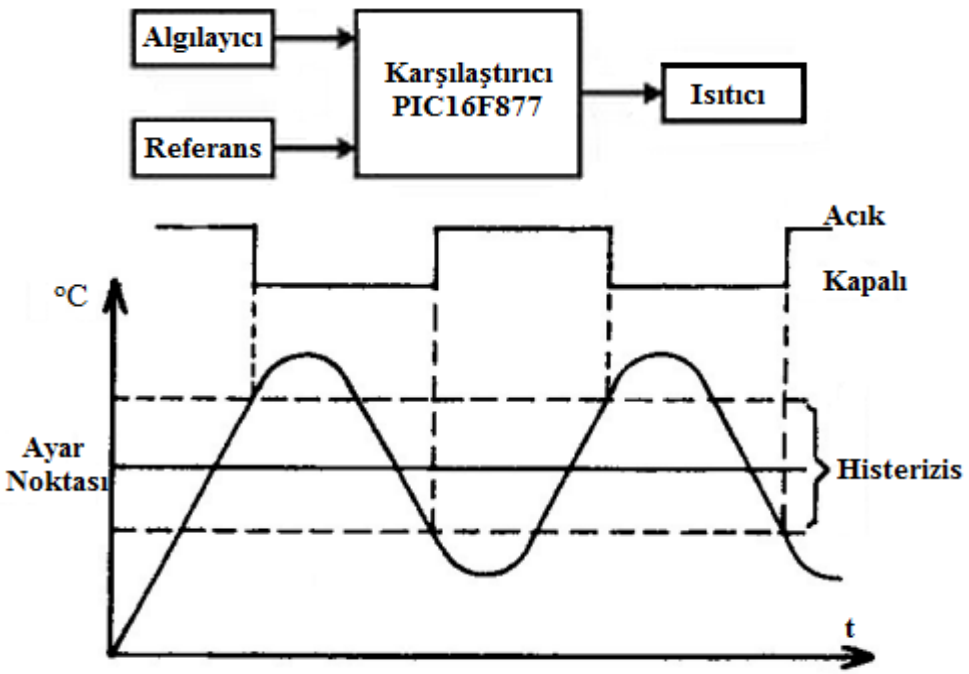
5 voltluk referans gerilimi için 0,019 volt'luk çözünürlük olmaktadır. 0,019 volt 19 mV olmaktadır. Bu sistem için 19 mV örnekleme aralığı yeterli çözünürlüğü sağlamamaktadır. Bu nedenle referans gerilimi 2 volt olarak seçilmiştir. Seçilen referans gerilimi için 0,007 voltluk çözünürlük olmaktadır. Bu şekilde her iki dereceye karşılık dijital bilgi üretilmektedir. Örnek olarak 25 $^{o}$C sıcaklık değerine karşılık gelen dijital bilgi 00101000 8 bitlik bilgidir. Bu sayının 16 'lık (hexadecimal) sayı sistemi karşılığı 0x28, 10 'luk (desimal) sayı sistemi karşılığı 40 tır. Mikrodenetleyici içerisinde yazılımsal olarak bu üç bilgide kullanılabilmektedir.

### 4.4. Değerlendirme ve Kontrol İşaretinin Üretilmesi ve Sisteme Uygulanması

Geliştirilen sistemle sistemin histerezis aralıkta aç−kapa denetimi gerçekleştirilmiştir. Normal aç−kapa denetimde, kontrol cihazı, gelen bilgi belirlenen referans değerinin üzerinde veya altında ise enerjiyi açar ya da kapatır. Enerji ya tamamen açıktır ya da tamamen kapalıdır. Bu tip denetim sıcaklık kontrolü gibi çok

hassas olmayan sistemlerde kullanılırlar. Bu tip denetimde en çok karşılaşılan sorun, referans değerine ulaşan sistem çıkışında sürekli bir aç−kapa durumunun oluşması ve bunun kontrol kısmında bulunan mekanik aksama zarar vermesidir. Bu sorunu aşmak amacıyla sistemin histerezis ile aç−kapa denetimi gerçekleştirilmiştir. Histerezis denetimde referans değeri aşılır aşılmaz enerji kesilmez (kapa (off) durumu). Daha önceden belirlenen bir değer aşılırsa enerji kesilmektedir. Aynı şekilde referans değerinin altına düşüldüğünde sisteme enerji verilmez (aç (on) durumu). Daha önce belirlenen bir değerin altına düşüldüğünde sisteme enerji verilmektedir. Histerezis aralık ile referans noktası etrafında sabit bir bant oluşturulmaktadır. Böylelikle aç−kapa durumları geciktirilerek mekanik sistemin sık aç−kapa yapmasının önüne geçilmektedir.

Geliştirilen test düzeneğinin denetleme elemanı mikrodenetleyicidir. Mikrodenetleyici içerisinde yazılmış rutinle sistemin histerezis aralıkta aç−kapa (on−off) denetimi yapılmaktadır. Şekil 4.12'de geliştirilen sistemle yapılan denetimin grafiksel gösterimi görülmektedir [35].




Şekil 4.12: Histerezis aralıklı aç−kapa denetim grafiği

Sistemin denetlenmek istendiği ayar noktası ve histerezis bant aralığı belirlendikten sonra, referans değeri bilgisayar tarafından ilk önce MDA320 veri edinim borduna gönderilmektedir. Daha sonra bu veri, veri edinim bordundan $I^2C$ ile PIC16F877 mikrodenetleyicisine gönderilmektedir. Süreç denetim sisteminden gelen bilgi ile referans bilgisi mikrodenetleyici içerisinde karşılaştırılarak aç ya da kapa denetim bilgisi üretilmektedir. Denetim bilgisi belirlenen referans değeri ve histerezis aralığa göre üretilmektedir. Histerezis aralık ile, seçilen referans değeri orta nokta olacak şekilde üst ve alt sınır noktaları belirlenmektedir. Denetim bilgisinin üretildiği değerler bu sınır değerleridir. Aç−kapa denetimde referans değerinin üzerine çıkıldığı anda kapa denetim bilgisi üretilirken, histerezis ile aç−kapa denetimde ise referans+histerezis değerine gelindiği anda kapa denetim bilgisi üretilmektedir. Aynı şekilde aç denetim bilgisi de referans−histerezis değerinde üretilmektedir. Örneğin, referans değeri 25 $^oC$, histerezis aralık 4 $^oC$ belirlendiğinde üst sınır 25+2=27 $^oC$, alt sınır ise 25-2=23 $^oC$ olmaktadır. Bu çalışma değerleri için sistem üzerindeki rezistansın kapanma değeri üst sınır olan 27 $^oC$, rezistansın açılma değeri ise alt sınır olan 23 $^oC$'dir. Rezistans bu bant aralığında gelen denetim bilgisine göre çalışmaktadır.

Üretilen denetim bilgisi 8 bitlik bir bilgidir ve mikrodenetleyicinin dijital çıkışlarına yazılmaktadır. Dijital çıkışlara aç denetim bilgisi için 16'lık (hexadesimal) sayı sistemine göre 0xFF (ikilik- 11111111) bilgisi, kapa denetim bilgisi içinse 16'lık (hexadesimal) sayı sistemine göre 0x00 (ikilik- 00000000) bilgisi dijital çıkışlara yazılmaktadır. Süreç denetim sistemini iki seviyeli (0/10 V) denetime ek olarak oransal olarak da denetleyebilmek için dijital−analog çevirici (Dijital Analog Converter – DAC) kullanılmaktadır. DAC girişlerine aç denetim bilgisi (11111111B) bilgisi gönderildiğinde DAC'ın çıkış değeri, $V_{REF}$ değerine göre alabileceği en yüksek değerini (10 V) almaktadır. DAC girişlerine kapa denetim bilgisinin gelmesi ile (00000000B) DAC 'ın çıkış değeri alabileceği minimum değeri (0 V) almaktadır.

DAC devresi çıkışı süreç denetim sisteminin harici girişine bağlanarak süreç denetim sistemi denetlenmektedir.

## 4.5. Geliştirilen Sistemin İzlenmesi

### 4.5.1. Sıcaklık bilgisinin bilgisayara aktarılması

Tez çalışması için geliştirilen test düzeneğinin çalışması, süreç denetim sisteminden gelen sıcaklık bilgisinin algılayıcıdan alınarak bilgisayar ortamına aktarılmasıyla izlenebilmektedir.

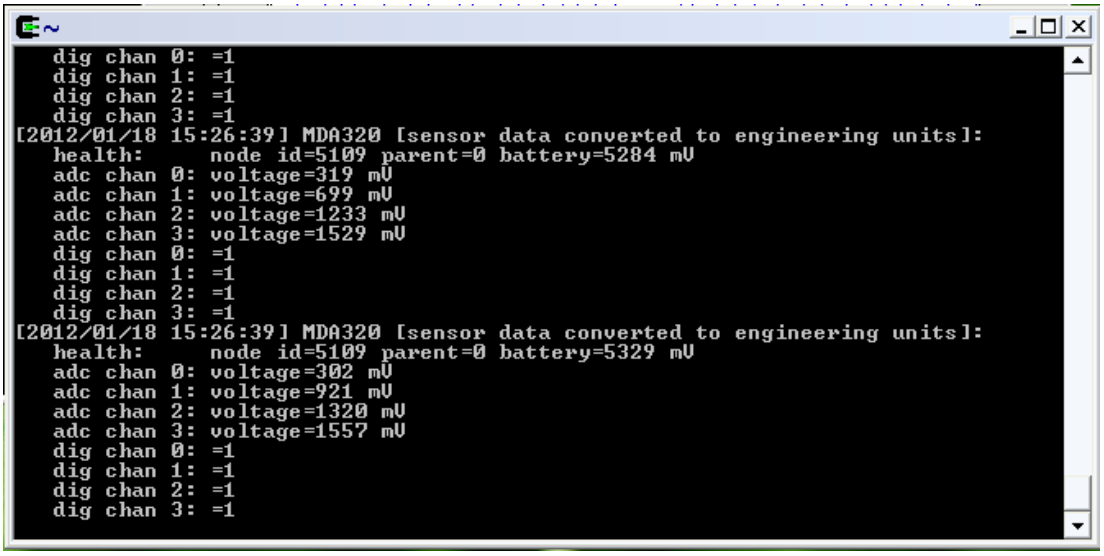
Geliştirilen test düzeneğinin çalışmasının izlenmesi için, ilk aşamada ölçülen sıcaklık bilgilerinin bilgisayara aktarılması gerekmektedir. Bu amaçla sistem üzerinde bulunan algılayıcının ölçtüğü sıcaklık bilgileri bilgisayara aktarılmaktadır. Ölçülen sıcaklık ile ilgili sistemden alınan anlamlı veri, köprü elemanı uçlarındaki her bir sıcaklık değerine karşılık gelen gerilim değerleridir.

Süreç denetim sisteminin rezistansının yaydığı sıcaklık bilgisi sistem üzerinde bulunan algılama elemanıyla algılanmaktadır. Algılama elemanı, boncuk tipi bir ısıl dirençtir. Isıl direnç uçları sistem üzerinde bulunan köprü elemanına bağlanmaktadır. Rezistansın sıcaklığına göre ısıl elemanın direnci ve buna bağlı olarak köprü elemanın uçlarındaki gerilim değeri değişmektedir. Isıl direncin negatif katsayılı olmasından dolayı sıcaklık arttıkça köprü uçlarındaki gerilim değeri düşmektedir. Örneğin sıcaklığın 0 $^{o}$C ölçüldüğü durumda köprü elemanının uçlarında 400 mV gerilim bulunurken, sıcaklığın 80 $^{o}$C olduğu durumda ise köprü elemanının uçlarında 250 mV bulunmaktadır.

Sıcaklık bilgisinin alınması için köprü elemanının ucu MDA320 veri edinim bordunun analog girişine bağlanmıştır. Analog girişten okunan gerilim değeri kablosuz ortam aracılığıyla baz istasyonuna gönderilmektedir. Baz istasyonuna gelen bu veriler XML formatında yayınlanmaktadır. PERL dili kullanılarak yazılan kod ile XML formatındaki verilerin yayınlandığı port dinlenmektedir. Gelen veriler arasından analog girişten gelen gerilim değerinin olduğu veri alınarak dosyaya yazılmaktadır.

### 4.5.2. Geliştirilen sistem için örnek uygulama

Geliştirilen sistemin çalışmasını izlemek amacıyla, belirlenen değerlere göre örnek bir uygulama gerçekleştirilmiştir. Örnek uygulama için referans sıcaklık 40 $^{o}$C, histerezis aralık ise 6 $^{o}$C olarak seçilmiştir. Bu değerlere göre, referans değer orta nokta olacak şekilde sistemin denetleneceği üst sınır, referans+histerezis=40+3=46 $^{o}$C, alt sınır ise referans-histerezis=40-3=37 $^{o}$C'dir. Seçilen referans değeri ve histerezis aralık değerlerine göre mikrodenetleyici içerisinde denetim rutini tekrar düzenlenmiştir. Referans değeri MDA320 veri edinim borduna kablosuz olarak MATLAB GUI üzerinden gönderildikten sonra sistemin denetlenmesine başlanmıştır. Denetimin başlamasından sonra MDA320 veri edinim bordundan bilgisayara gelen veriler Şekil 4.13'te görülmektedir.




Şekil 4.13:Veri edinim bordundan gelen bilgiler

Şekil 4.13'te görülen değerler arasından adc chan 0 değeri, köprü devresinin bağlı olduğu analog kanaldan gelen gerilim değeridir. XML formatlı olarak gelen bu bilgi PERL dili ile yazılmış kod ile alınarak hem çalışma ekranına hem de dosyaya yazdırılmaktadır. PERL ile alınmış bilgiler Şekil 4.14'te görülmektedir.


Şekil 4.14: Perl ile alınan veriler

PERL ile alınan bu verilerin anlamlı bir şekilde grafiklendirilebilmesi için gelen gerilim değerleri ile bu değerlere karşılık gelen sıcaklık değerleri eşleştirilmektedir. Eşleştirme işlemi için ilk önce hangi sıcaklık değerinin hangi gerilim değerine denk geldiği belirlenmiştir. Seçilen referans değerlere karşılık gelen gerilim değerleri Tablo 4.2'de görülmektedir.

Tablo 4.2: Sıcaklık−Gerilim ilişkisi

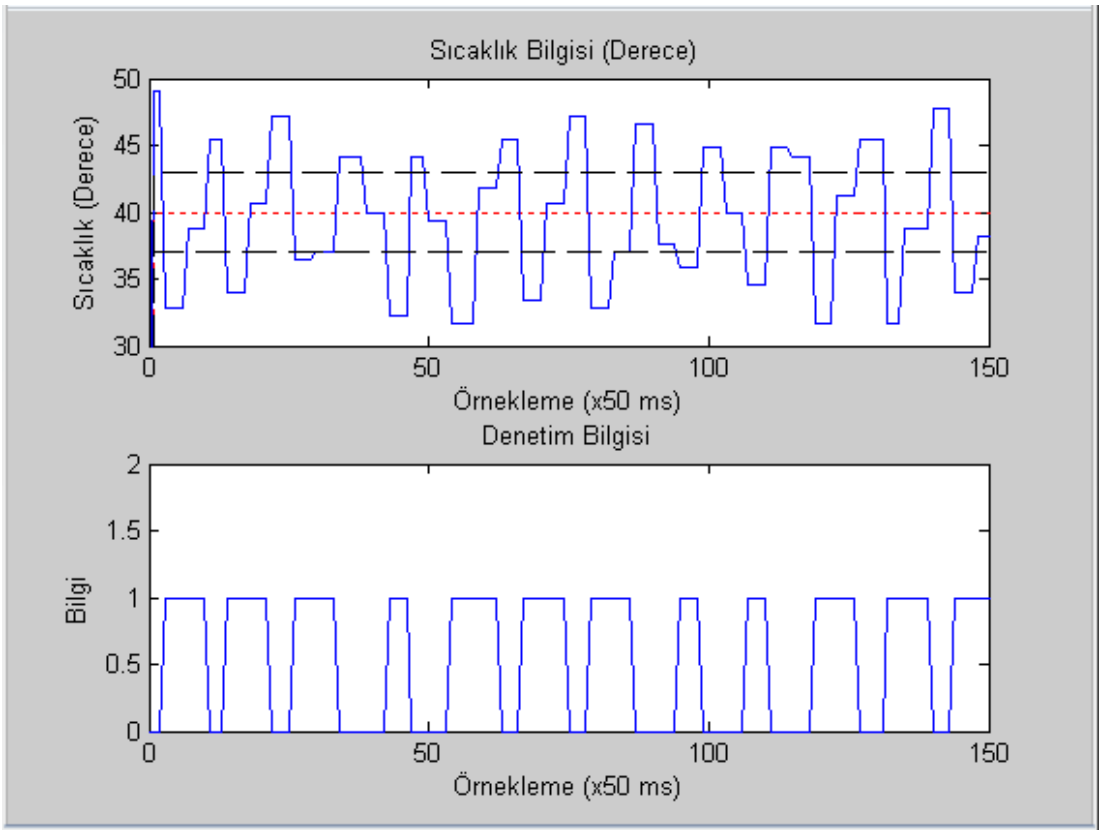
| **Seçilen Değer** | **Sıcaklık ($^{o}$C)** | **Gerilim (mV)** |
|---|---|---|
| Üst Sınır (ref+his) | 43 | 312 |
| Referans Değeri | 40 | 320 |
| Alt Sınır (ref-his) | 37 | 328 |

Tabloda görülen değerlere göre ölçülen ara değerlere göre eşleştirme için bir formül üretilmiştir. Üretilen formül aşağıdaki gibidir.

$$ÖS = SÜS - (GG - GAS) * \frac{SA}{GA} \tag{4.3}$$

Bu formülde ÖS algılayıcının ölçtüğü sıcaklık, SÜS sıcaklık üst sınır değeri, GG köprü elemanından gelen gerilim, GAS gerilim alt sınırı, SA sıcaklık aralığı, GA ise gerilim aralığını ifade etmektedir.

PERL dili ile alınıp dosyaya yazılan verilerin sıcaklık eşleştirme işlemi MATLAB programı içerisinde yapılarak sistemin çalışması grafiklendirilmiştir. Sistemin çalışmasını gösteren grafik Şekil 4.15'te görülmektedir.




Şekil 4.15 : Sistem çalışmasının izlenmesini gösteren grafik

Grafikte sistemin ürettiği ilk sıcaklık değeri, üst sınır olan 43 $^{o}$C' nin üstündedir. İlk anda bu sıcaklık değerine göre sisteme "ısıtıcıyı kapa" denetim bilgisi gitmektedir. Sıcaklık değeri histerezis aralığın alt sınırı olan 37 $^{o}$C'ye kadar azalmakta, bu bölgede ise sisteme sürekli "ısıtıcıyı kapa" bilgisi gönderilmektedir. Sıcaklık değeri alt sınır olan 37 $^{o}$C'nin altına indiği anda sisteme "ısıtıcıyı aç" denetim bilgisi gönderilmektedir. "Isıtıcıyı aç" denetim bilgisi ile sistem üzerindeki rezistans açılmaktadır. Sıcaklık artışı histerezis aralığın üst sınırı olan 43 $^{o}$C'ye kadar devam etmektedir. Bu bölgede sisteme sürekli "ısıtıcıyı aç" denetim bilgisi gönderilmektedir. Sıcaklık değeri üst sınır olan 43 $^{o}$C ölçüldüğü anda sisteme tekrar "ısıtıcıyı kapa" denetim bilgisi gönderilmektedir. Sistem denetimi bu şekilde

herhangi bir müdahele gerektirmeksizin devam etmektedir. Grafik sistemden gelen gerçek zamanlı verilere göre çizdirilmektedir.

### 4.6. Sonuç

Bu bölümde Bölüm 3'te bahsedilen yazılım ve donanım bileşenleri kullanılarak gerçekleştirilen test düzeneğinin çalışması anlatılmış ve süreç denetim sisteminin çalışmasına örnek bir uygulama gösterilmiştir. Örnek uygulamadan elde edilen sonuçlar uygulamanın gerçek zamanlı çalışmasıyla örtüşmektedir. Bu çalışma ile KAEA ile sistemlerin uzaktan izlenmesi ve denetlenmesi amacıyla test düzeneği geliştirilmiş, alınan gerçek zamanlı verilere göre grafikleri elde edilerek çalışması doğrulanmıştır.

## SONUÇLAR VE ÖNERİLER

Bu tez çalışmasında, KAEA kullanılarak birinci dereceden ölü zamanlı bir sistemin denetlenmesi ve izlenmesine yönelik fiziksel olarak gerçekleştirilen test düzeneği, bu test düzeneğini oluşturan yazılımsal ve donanımsal bileşenler ve son kısımda işlemlerin hangi sırayla yapıldığı anlatılmaktadır. Test düzeneğinin çalışması ile ilgili örnek bir çalışma yapılmış, elde edilen sonuçlar ile sistemin gerçek zamanlı çalışmasının örtüştüğü görülmüştür.

Son yıllarda kablosuz haberleşme sistemlerinin hızla gelişimi KAA'ların ortaya çıkmasına sebep olmuştur. KAA bir mikrodenetleyici, algılayıcı, kablosuz haberleşme arabirimi ve güç ünitesinden oluşan düğümlerin bir araya gelerek oluşturduğu ağ yapısıdır. KAA'daki düğümler, bulunduğu ortamdan topladığı fiziksel büyüklükleri merkezi bir erişim noktasına göndermektedir. Algılama verileri çeşitli yapılar kullanılarak (multi hop, single hop) kablosuz ortam üzerinden merkezi düğüme iletilmektedir.

Teknolojideki son gelişmeler, fiziksel dünyayı gözlemleme yeteneğine sahip, veri işleyebilen, karar verme tabanlı uygun işlemleri gerçekleştirebilen dağıtılmış kablosuz algılayıcı ve eyleyici ağların (KAEA) ortaya çıkmasına yol açmıştır. KAEA, KAA'ların genişletilmiş halidir.

Tez çalışması ile KAEA ile uzaktan denetlenmesini ve izlenmesini sağlayan bir test düzeneği oluşturulmuştur. Oluşturulan test düzeneği ile birinci dereceden ölü zamanlı bir sistemin uzaktan denetimi gerçekleştirilmiş ve sistemin çalışması izlenmiştir. Geliştirilen test düzeneğinin modüler olması denetlenmek istenen sistemler ile ilgili esneklik sağlamaktadır.

Tez çalışmasının ana katkıları aşağıdaki gibi maddeler halinde sıralanabilir;

- KAA, geleneksel kullanım alanlarının yanı sıra denetim amaçlı kullanılmıştır.
- KAEA tabanlı denetim sistemi gerçekleştirilmiştir.
- Geliştirilen test düzeneğinde kullanılan donanımsal ve yazılımsal bileşenler ile ilgili bilgiler sunulmuştur.
- Sistemin çalışmasının aşamaları sunulmuştur.
- Örnek bir uygulama yapılarak test düzeneğinin çalışması doğrulanmış, izlenen veriler ile gerçek zamanlı çalışmanın sonuçlarının örtüştüğü görülmüştür.

Bu tez çalışması ile sistemlerin uzaktan denetlenmesi ve izlenmesine yönelik olarak test düzeneği oluşturulmuştur. Test düzeneği oluşturulurken birçok yazılım ve donanım bileşeni kullanılmıştır. Kullanılan bu bileşenlerin yanı sıra $I^2C$ ve XMLRPC gibi haberleşme yöntemleride test düzeneğinin gerçekleştirilmesinde kullanılmıştır. Geliştirilen test düzeneği modüler ve esnek yapıdadır. Test düzeneği çıkışına denetlenecek farklı sistemler de bağlanabilir. Bu çalışmanın yanı sıra ileride gerçekleştirilmesi planlanan bazı yenilikler aşağıda sunulmaktadır;

- Tez çalışmasında denetleme elemanı olarak PIC16F877 mikrodenetleyicisi kullanılmıştır. Bu denetleyici yerine ihtiyaçlara göre farklı denetleyicilerde kullanılabilir.

- Tez çalışmasında üretilen kontrol bilgisi mikrodenetleyicinin çıkışlarına yazılmaktadır. Aynı şekilde üretilen kontrol bilgisi $I^2C$ haberleşmesi ile tekrar MDA320 veri edinim borduna gönderilerek veri edinim bordunun dijital çıkışlarına yazılabilir ve denetleme elemanı olarak veri edinim bordu kullanılabilir.

- Tez çalışmasında birinci dereceden ölü zamanlı sistemin histerezis aralıkta aç−kapa denetimi gerçekleştirilmiştir. İleriye yönelik çalışmalarda mikrodenetleyici içerisinde oransal, PID, bulanık (fuzzy), adaptif gibi farklı denetim algoritmaları gerçekleştirilebilir.

- Geliştirilen test düzeneği gecikme duyarlı sistemler içinde, gerekli düzenlemeler yapılarak kullanılabilir.

## KAYNAKLAR


[1]Ceken, C., “An Energy Efficient and Delay Sensitive Centralized MAC Protocol for Wireless Sensor Networks”, ***Computer Standards & Interfaces***, 30, 1-2, 20-31, (2008).

[2]Akyıldız, I. F., Su, W., Sankasubramaniam, Y., Çayırcı, E., “Wireless Sensor Networks: A Survey”, ***Computer Networks***, 393 – 422, (2002).

[3]Alippi, C., Anastasi, G., Galperti, C., Mancini, F., Roveri, M., “Adaptive Sampling for Energy Conservation in Wireless Sensor Networks for Snow Monitoring Applications”, ***IEEE International Workshop on Mobile Ad Hoc and Sensor Systems for Global and Homeland Security (MASS-GHS 2007)***, Pisa, Italy, (2007).

[4]Chiaia, I. C., “Active Fault-Tolerance in Wireless Networked Control Systems”, Doktora Tezi, ***Duisburg-Essen Üniversitesi***, (2010).

[5]Akyıldız, I. F., Kasımoğlu, İ. H., “Wireless Sensor And Actor Networks: Research Challenges”, ***Ad Hoc Networks 2***, 351 – 367, (2004).

[6] Peng, Y., Lahusen, R., Shirazi, B., Song, W., “Design Of Smart Sensing Component For Volcano Monitoring”, ***Intelligent Environments, 2008 IET 4th International Conference on,*** Seatle, WA, (2008).

[7] Chin, J-C., Rautenberg, J. M., Ma, C. Y. T., Pujol, S., Yau, D.K.Y., “An Experimental Low-cost, Low-data-rate Rapid Structural Assessment Network”, ***Sensors Journal, IEEE,*** Vol. 9, 1361-1369, (2009).

[8] Khakpour K., Shenassa, M. H., “Industrial Control Using Wireless Sensor Networks”, ***IEEE 3rd International Conference on Information and Communication Technologies: From Theory to Applications, 2008. ( ICTTA 2008),*** Damascus, Syra, (2008) .

[9] Li, X., Sun, Z., Huang, T., Du, K., Wang, Q., Wang, Y., “ Embedded Wireless Network Control System: an Appliacation of Remote Monitoring System for Greenhouse Environment”, ***IMACS Multiconference on "Computational Engineering in Systems Applications" (CESA)***, Beijing, China, (2006).

[10] Farruggia, O., Fodero, F., Paola, A. D., Ortolani, M., Re, C. L., “Wireless Sensor Networks for Marine Environment Monitoring”, ***”GEOGRID OPEN DAY AT THE UNIVERSITY OF PALERMO”,*** 22-26**,** Palermo, ITALY, ( 2008).

[11] Solak, S., “Kablosuz Algılayıcı Ağlarda Kullanılan MAC Protokollerinin Karşılaştırılmalı Başarım Analizi”, Yüksek Lisans Tezi, ***Kocaeli Üniversitesi Fen Bilimleri Enstitüsü***, (2008).

[12] Turhan, E., “Kablosuz Algılayıcı Ağlar İçin Matlab İle Kullanıcı Arayüz Tasarımı”, Yüksek Lisans Tezi, ***Kocaeli Üniversitesi Fen Bilimleri Enstitüsü***, (2011).

[13] Çakıroğlu, M., “Kablosuz Algılayıcı Ağlar İçin Dinamik Kanal Atlamalı Güvenlik Sistemi Tasarımı”, Doktora Tezi, ***Sakarya Üniversitesi Fen Bilimleri Enstitüsü***, (2008).

[14] Kuş, M., “Kablosuz Algılayıcı Ağlar İçin İnternet Tabanlı İzleme Sistemi Tasarımı”, Yüksek Lisans Tezi, ***Sakarya Üniversitesi Fen Bilimleri Enstitüsü***, (2010).

[15] Harmankaya, A. O., “Kablosuz Algılayıcı Ağ Yönlendirme Protokollerinin Karşılaştırılması”, Yüksek Lisans Tezi, ***Kocaeli Üniversitesi Fen Bilimleri Enstitüsü***, (2007).

[16] Yick, J., Mukherjee, B., Ghosal, D., “Wireless sensor network survey”, ***Computer Networks,*** 2292-233, (2008).

[17] Crossbow, “MPR/MIB User’s Manual”, ***Crossbow Technology,*** Revision A, (2007).

[18] Crossbow, “MTS/MDA Sensor Board User’s Manual”, ***Crossbow Technology,*** Revision A, (2007).

[19] Han, P., Liu, J., Kai, P., “A set of simple and effective control methods for dead time systems”, ***Proceedings of the IEEE International Conference on Automation and Logistics***, 1674-1677, (2008).

[20] Aktaş, F., Çeken, C., Erkan, K., Yıldırım, M., “Kablosuz Algılayıcı Ağlar Kullanılarak Birinci Dereceden Ölü Zamanlı Bir Sistemin Denetimi”, ***6th International Advanced Technologies Symposium (IATS’ 11)***, Elazığ, Türkiye, (2011).

[21] Yücelen, T., “Uzun Ölü Zamanlı Sistemler İçin Smith Öngörücüsü Yöntemi İle PI-P Kontrolör Tasarımı”, ***III.OTOMASYON SEMPOZYUMU VE SERGİSİ,*** Denizli, Türkiye, (2005).

[22] FeedBack, “Process Control Trainer 37-100”, ***FeedBack,*** Manuel 37-100, (2007).

[23] MICROCHIP, “PIC16F87X Data Sheet”, ***Microchip Technology Inc***., DS30292C, (2001).

[24] Philips Semiconductors, “Data Sheet MC1408-8 8 bit multiplying D/A Converter”, ***PHILIPS*** , IC11 Handbook, (2003).

[25] Atmaca, S., “$I^2C$-Bus Seri İletişim Protokolü İçin Veri İzleme Sistemi”, Yüksek Lisans Tezi, ***Sakarya Üniversitesi Fen Bilimleri Enstitüsü***, (2002).

[26] Philips Semiconductors, “The I2C-bus and how to use it”, ***PHILIPS*** , Including Specifications, (1995).

[27] Online, http://tr.wikipedia.org/wiki/Perl, (**Ziyaret Tarihi*:* 2011**)

[28] Online, http://www.controlworld.tk/web_project_files/c8d4216c1ebe78f8eb6b31d93c85a9d/kaynaklar/matlab.pdf, (**Ziyaret Tarihi*:* 2011**).

[29] Dener, M., Bay, O. F., “Kablosuz Algılayıcı Ağlarda Düğümler Arasında Veri Haberleşmesi”, ***6th International Advanced Technologies Symposium (IATS’ 11)***, Elazığ, Türkiye, (2011).

[30] Yıldırım, K. S., Kantarcı, A., “Kablosuz Algılayıcı Ağlar İçin TinyOS ile Uygulama Geliştirme”, ***Akademik Bilişim’10***, Muğla, Türkiye, (2010).

[31] Çalhan, A., Çeken, C., "Ortam İzleme Amaçlı Kablosuz Algılayıcı Ağların Benzetimi ve Uygulaması", ***Haberleşme Teknolojileri ve Uygulamaları Sempozyumu (Habtekus 08),*** İstanbul, Türkiye, (2008).

[32] Online, http://www.cs.utsa.edu/~korkmaz/teaching/cn−resources/tinyos/tinyosnesc_pres.pdf, (**Ziyaret Tarihi: 2011**)

[33] Crossbow, “MoteWorks Getting Started Guide”, ***Crossbow Technology,*** Revision D, (2007).

[34] Crossbow, “XServe User’s Manual”, ***Crossbow Technology,*** Revision D, (2007).

[35] Online, http://afguven.com/depo/dersnot/bahar22/Bkontrol/bk5.pdf, (**Ziyaret Tarihi: 2011**)

## EK−A GELİŞTİRİLEN SİSTEM İÇİN YAZILAN KAYNAK KODLAR

MDA320 içerisinde çalışan nesC dilindeki kodlar

**DioM dosyası**

```
module DioM {
   provides
{
      interface StdControl;
      interface Dio[uint8_t channel];
}
   uses
{
      interface StdControl as I2CPacketControl;
      interface Leds;
      interface I2CPacket;
}
}

implementation {

  uint8_t state;      //keep state of our State Machine
  uint8_t io_value;        //keep track of what is actually on the chip
  uint8_t mode[8];      //keep track of the mode of each channel
  uint16_t count[8];   //we can count the number of pulses
  uint8_t bitmap_high,bitmap_low,bitmap_toggle;
  uint8_t i2c_data;    //the data read from the chip
  uint8_t intflag=0;
  uint8_t i2cwflag=0;
  uint8_t i2crflag=0;
  uint8_t gonderilecekVeri;

#define XOR(a,b)  ((a) & ~(b))|(~(a) & (b))
#define  testbit(var, bit)   ((var) & (1 <<(bit)))      //if zero then return zero and if one
not equal zero
#define  setbit(var, bit)    ((var) |= (1 << (bit)))
#define  clrbit(var, bit)    ((var) &= ~(1 << (bit)))

//Interrupt definition
#define INT_ENABLE()  sbi(EIMSK , 4)
#define INT_DISABLE() cbi(EIMSK , 4)
```

```
enum {GET_DATA, SET_OUTPUT_HIGH, SET_OUTPUT_LOW,
SET_OUTPUT_TOGGLE , GET_THEN_SET_INPUT, IDLE , INIT};
   command result_t StdControl.init()
{
        mode[0] = RISING_EDGE;
        mode[1] = RISING_EDGE;
        mode[2] = RISING_EDGE;
        mode[3] = RISING_EDGE;
        mode[4] = RISING_EDGE;
        mode[5] = RISING_EDGE;
        mode[6]=DIG_OUTPUT;
        mode[7]=DIG_OUTPUT;
       io_value=0x0;
       state=INIT;
      call I2CPacketControl.init();
      return SUCCESS;
 }
   task void init_io()
     {
      atomic i2crflag=1;
         if(call I2CPacket.readPacket(1,0x03) == FAIL)
           {
                atomic i2crflag=0;
                post init_io();
           }
     }

   task void read_io();
   command result_t StdControl.start()
 {
      cbi(DDRE,4);          //Making INT pin input
      //cbi(EICRB,ISC40);      //Making INT sensitive to falling edge
      //sbi(EICRB,ISC41);
      //INT_ENABLE();          //probably bus is stable and now we are ready
      post init_io();
      return SUCCESS;
 }
   command result_t StdControl.stop()
 {
    return SUCCESS;
 }
   command result_t Dio.setparam[uint8_t channel](uint8_t modeToSet)
      {
                mode[channel]=modeToSet;
         if( ((modeToSet & RISING_EDGE) == 0) & ((modeToSet &
FALLING_EDGE) == 0) ) mode[channel] |= RISING_EDGE;
         if((modeToSet & DIG_LOGIC)!=0)
         {
         if(intflag==0)
```

```
        {
        state=IDLE;
        post read_io();
        }
        }
        return FAIL;
    }
 task void set_io_high_cceken()
   {
    uint8_t status;
    status = FALSE;
    if(state==IDLE) state= SET_OUTPUT_HIGH;
    else { status=TRUE; post set_io_high_cceken(); }
    if(status==TRUE) return;
    i2c_data=gonderilecekVeri
    atomic i2cwflag=1;
    if( (call I2CPacket.writePacket(1,(char*) &i2c_data, 0x01)) == FAIL)
      {
        atomic i2cwflag=0;
        state=IDLE;
        post set_io_high_cceken();
      }
    else
     {
         bitmap_high=0x0;
         io_value=i2c_data;
     }
   }
command result_t Dio.high_cceken[uint8_t channel](uint8_t veri)
   {
        gonderilecekVeri=~veri;
        post set_io_high_cceken();
        return SUCCESS;
   }
 command result_t Dio.getData[uint8_t channel]()
   {
    uint16_t counter;
    counter = count[channel];
    if(RESET_ZERO_AFTER_READ & mode[channel]) {count[channel]=0;}
    signal Dio.dataReady[channel](counter);
    return SUCCESS;
   }
 default event result_t Dio.dataReady[uint8_t channel](uint16_t data)
    {
        return SUCCESS;
    }
 task void read_io()
    {
       uint8_t status;
```

```
        status = FALSE;
           if(state==IDLE) state=GET_DATA;
           else { status=TRUE; post read_io(); }
        if(status==TRUE) return;
        atomic i2crflag=1;
        if(call I2CPacket.readPacket(1,0x03) == FAIL)
           {
         atomic i2crflag=0;
                     state=IDLE;
                     post read_io();
           }
      }
   event result_t I2CPacket.writePacketDone(bool result) {
     if(i2cwflag==0) return SUCCESS;
     atomic i2cwflag=0;
      if(result) {
         if ( state == SET_OUTPUT_HIGH || state == SET_OUTPUT_LOW || state
== SET_OUTPUT_TOGGLE) {
           state = IDLE;
         }
      }
      return SUCCESS;
   }
   event result_t I2CPacket.readPacketDone(char length, char* data)
 {
      uint8_t ChangedState;
      int i;
      if(i2crflag==0) return SUCCESS;
      atomic i2crflag=0;
      i2c_data=*data;
      if (length != 1)
         {
           state = IDLE;
             INT_ENABLE();
             return FALSE;
         }
      if(state==INIT)
        {
           io_value=i2c_data;
           state=IDLE;
           INT_ENABLE();
        }
      if(state==GET_DATA) {
        intflag=1;
        ChangedState = XOR(io_value,i2c_data);     //see those one who has changed
        for(i=0;i<8;i++){
          if( !( mode[i] & DIG_OUTPUT) ){ if((mode[i] & DIG_LOGIC))
           {
              if(testbit(i2c_data,i)!=0)
```

```
                count[i]=1;
              else
                count[i]=0;
              signal Dio.dataReady[i](count[i]);
              continue;
            }
            if(testbit(ChangedState,i)) {       //find the channels which are realy changed
              if( mode[i] & RISING_EDGE )
              {
                if(testbit(io_value,i)==0 && testbit(i2c_data,i)!=0) {
                  if(EVENT & mode[i]) signal Dio.dataReady[i](count[i]);
                  //                        if (count[i] == 0xffff) signal Dio.dataOverflow[i]();
                  count[i]++;
                }
              }
              if( mode[i] & FALLING_EDGE )
              {
                if(testbit(io_value,i)!=0 && testbit(i2c_data,i)==0) {
                  if(EVENT & mode[i]) signal Dio.dataReady[i](count[i]);
                  //                        if (count[i] == 0xffff) signal Dio.dataOverflow[i]();
                  count[i]++;
                }
              }
            }
          }
        }
          io_value=i2c_data;
          INT_ENABLE();
          state = IDLE;
      }
      return SUCCESS;
    }

  TOSH_SIGNAL(SIG_INTERRUPT4)
      {
        INT_DISABLE();
        if(!post read_io()) INT_ENABLE();
        return;
      }
}
```

**I2CPacketM dosyası**

```
module I2CPacketM
{
  provides
{
    interface StdControl;
    interface I2CPacket[uint8_t id];
```

```
 }
  uses
 {
    interface I2C;
    interface StdControl as I2CStdControl;
    interface Leds;
  }
}
implementation
{
   enum {IDLE=99,
        I2C_START_COMMAND=1,
        I2C_STOP_COMMAND=2,
        I2C_STOP_COMMAND_SENT=3,
        I2C_WRITE_ADDRESS=10,
        I2C_WRITE_DATA=11,
        I2C_READ_ADDRESS=20,
        I2C_READ_DATA=21,
        I2C_READ_DONE=22};

        enum {STOP_FLAG=0x01, /* send stop command at the end of packet? */
        ACK_FLAG =0x02, /* send ack after recv a byte (except for last byte) */
        ACK_END_FLAG=0x04, /* send ack after last byte recv'd */
        ADDR_8BITS_FLAG=0x80, // the address is a full 8-bits with no terminating
readflag
   };
  char* data;
  char length;
  char index;
  char state;
  char addr;
  char flags;
  char temp[10];
command result_t StdControl.init()
{
    call I2CStdControl.init();
    atomic {
        state = IDLE;
        index = 0;
              }
    return SUCCESS;
  }
command result_t StdControl.start()
 {
     return SUCCESS;
 }

command result_t StdControl.stop() {
     return SUCCESS;
```

```
  }
command result_t I2CPacket.writePacket[uint8_t id](char in_length,char* in_data,
char in_flags)
{
     uint8_t status;
     atomic {
     status = FALSE;
     if (state == IDLE)
        {
           /*  reset variables  */
           addr =id;
           data = in_data;
           index = 0;
           length = in_length;
           flags = in_flags;
           state = I2C_WRITE_ADDRESS;
           status = TRUE;
        }
     }
     if(status == FALSE ) {
        return FAIL;
     }

     if (call I2C.sendStart())
        {
           return SUCCESS;
        }
     else
        {
           atomic { state = IDLE; }
           return FAIL;
        }
  }
   command result_t I2CPacket.readPacket[uint8_t id](char in_length, char in_flags)
{
     uint8_t status;
     atomic {
     status = FALSE;
     if (state == IDLE)
        {

           addr = id;
           index = 0;
           length = in_length;
           flags = in_flags;
           state = I2C_READ_ADDRESS;
           status = TRUE;
        }
     }
```

```
      if(status == FALSE ) {
         return FAIL;
      }
      if (call I2C.sendStart())
         {
            return SUCCESS;
         }
      else
         {
            atomic { state = IDLE; }
            return FAIL;
         }
  }
  event result_t I2C.sendStartDone() {
       if(state == I2C_WRITE_ADDRESS){
          state = I2C_WRITE_DATA;
          call I2C.write( (flags & ADDR_8BITS_FLAG) ? addr : ((addr << 1) + 0) );
       }
       else if (state == I2C_READ_ADDRESS){
          state = I2C_READ_DATA;
          call I2C.write( (flags & ADDR_8BITS_FLAG) ? addr : ((addr << 1) + 1) );
          index++;
       }
    return 1;
 }
 event result_t I2C.sendEndDone() {
    char* out_data;
    char out_length;
    char out_addr;
    out_addr=addr;
    out_length=length;
    out_data=data;
    if (state == I2C_STOP_COMMAND_SENT) {
       // success!
       state = IDLE;
       signal I2CPacket.writePacketDone[out_addr](SUCCESS);
    }
    else if (state == I2C_READ_DONE) {
       state = IDLE;
       signal I2CPacket.readPacketDone[out_addr](out_length, out_data);
    }

    return SUCCESS;
 }
 event result_t I2C.writeDone(bool result) {
      if(result == FAIL) {
         state = IDLE;
         signal I2CPacket.writePacketDone[addr](FAIL);
         return FAIL;
```

```
    }
    if ((state == I2C_WRITE_DATA) && (index < length))
      {
        index++;
        if (index == length) {
          state = I2C_STOP_COMMAND;
        }
        return call I2C.write(data[index-1]);
      }
    else if (state == I2C_STOP_COMMAND)
      {
        state = I2C_STOP_COMMAND_SENT;
        if (flags & STOP_FLAG)
          {
            return call I2C.sendEnd();
          }
        else {
          state = IDLE;
          return signal I2CPacket.writePacketDone[addr](SUCCESS);
        }
      }
    else if (state == I2C_READ_DATA)
      {
        if (index == length)
          {
            return call I2C.read((flags & ACK_END_FLAG) == ACK_END_FLAG);
          }
        else if (index < length)
          return call I2C.read((flags & ACK_FLAG) == ACK_FLAG);
      }
    return SUCCESS;
  }

event result_t I2C.readDone(char in_data) {
  temp[index-1] = in_data;
  index++;
  if (index == length)
    call I2C.read((flags & ACK_END_FLAG) == ACK_END_FLAG);
  else if (index < length)
    call I2C.read((flags & ACK_FLAG) == ACK_FLAG);
  else if (index > length)
      {
        state = I2C_READ_DONE;
        data = (char*)(&temp);
        if (flags & STOP_FLAG){
          call I2C.sendEnd();}
        else
          {
```

```
                state = IDLE;
                signal I2CPacket.readPacketDone[addr](length, data);
            }
        }
    return SUCCESS;
  }
default event result_t I2CPacket.readPacketDone[uint8_t id](char in_length, char*
in_data) {
   return SUCCESS;
  }
  default event result_t I2CPacket.writePacketDone[uint8_t id](bool result) {
   return SUCCESS;
  }
}
```

**XMDA320 Dosyası Xcommand.Received Kod Bölümü**

```
  event result_t XCommand.received(XCommandOp *opcode) {
  switch (opcode->cmd) {
  case XCOMMAND_SET_RATE:
  timer_rate = opcode->param.newrate;
     call Timer.stop();
     call Timer.start(TIMER_REPEAT, timer_rate)
     break;
  case XCOMMAND_SLEEP:
     sleeping = TRUE;
     call StdControl.stop();
     call Timer.stop();
     call Leds.set(0);
     break;
  case XCOMMAND_WAKEUP:
     // Wake up from sleep state.
     if (sleeping)
    {
     initialize();
     call Timer.start(TIMER_REPEAT, timer_rate);
     sleeping = FALSE;
     }
     break;
  case XCOMMAND_RESET:
     // Reset the mote now.
     break;
  case XCOMMAND_ACTUATE:
    if (opcode->param.actuate.device != XCMD_DEVICE_SOUNDER) break;
    timer_rate = 1000;
    call Timer.stop();
    call Timer.start(TIMER_REPEAT, timer_rate)
    break;
        }
```

```
case XCOMMAND_CUSTOM_ACTION: {
call Leds.yellowOn();
call Leds.greenOn();
call Leds.redOn();
call Dio5.setparam(DIG_OUTPUT);
call Dio5.high_cceken(opcode->param.custom_data.value);
break;
        }
default:
    break;
    }
    return SUCCESS;
}
```

## EK–B MİKRODENETLEYİCİ İÇERİSİNDEKİ KODLAR

```
#include <16F877.h>
#device ADC=8
#FUSES NOWDT                    //No Watch Dog Timer
#FUSES XT                       //Crystal osc <= 4mhz for PCM/PCH ,
3mhz to 10 mhz for PCD
#FUSES NOPUT                    //No Power Up Timer
#FUSES NOPROTECT                //Code not protected from reading
#FUSES NOBROWNOUT               //No brownout reset
#FUSES NOLVP                    //No low voltage prgming, B3(PIC16)
or B5(PIC18) used for I/O
#FUSES NOCPD                    //No EE protection
#FUSES NOWRT                    //Program memory not write protected
#FUSES NODEBUG                  //No Debug mode for ICD
#use delay(clock=4000000)
#use i2c(Slave,Slow,sda=PIN_C4,scl=PIN_C3,address=0x7E,force_hw)
#use fast_io(a)
#use fast_io(b)
int bilgi,data;
int state = 0;
#INT_SSP                      //Interrupt for I2C activity
void sspinterupt()
{
   state = i2c_isr_state();   //Reading the type of transmission

   if(state < 0x80)           //Master is sending data
   {
      data = i2c_read();      //An array will be needed to store
data if more than one byte is transferred
   }
  if(state == 0x80)          //Master is requesting data
   {
      i2c_write(0x55);
   }
}
void main()
{
   enable_interrupts(INT_SSP);
   enable_interrupts(GLOBAL);
   setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);
   setup_timer_1(T1_DISABLED);
   setup_timer_2(T2_DISABLED,0,1);
   set_tris_a(0x01); // RA0 Giriş
   set_tris_b(0x00);
   setup_adc(adc_clock_div_32);
   setup_adc_ports(AN0_AN1_VSS_VREF);
   set_adc_channel(0);
   delay_us(20);
   while(TRUE)
   {
    if(bolge==1 && bilgi>data-3)
       output_b(0xFF);
    if(bolge==1 && bilgi<=data-3)
    {
       output_b(0x00);
       bolge=0;
    }
    if(bolge==0 && bilgi<data+3)
```

```
        output_b(0x00);
     if(bolge==0 && bilgi>=data+3)
     {
        output_b(0xFF);
        bolge=1;
     }
    }
}
```

## KİŞİSEL YAYINLAR VE PROJELER

### A. Uluslararası Bilimsel Toplantılarda Sunulan ve Bildiri Kitabında Basılan Bildiriler

1. Aktaş, F., Çeken C., Erkan K., Yıldırım M., "Kablosuz Algılayıcı Ağlar Kullanılarak Birinci Dereceden Ölü Zamanlı Bir Sistemin Denetimi", ***6th International Advanced Technologies Symposium (IATS' 11)***, Elazığ, Türkiye, (2011).

### B. Ulusal Bilimsel Toplantılarda Sunulan ve Bildiri Kitaplarında Basılan Bildiriler

1. Turhan E., Erkan K., Aktaş F., Çeken C., "Matlab ile Kablosuz Algılayıcı Ağlar İçin Ortam İzleme Amaçlı Kullanıcı Arayüz Tasarımı", ***Elektrik-Elektronik Bilgisayar Sempozyumu (FEEB 2011)***, Elazığ, Türkiye, (2011).

## ÖZGEÇMİŞ

1983 yılında Bursa'da doğdu. İlk, orta ve lise öğrenimini Bursa'da tamamladı. 2004 yılında girdiği Kocaeli Üniversitesi Teknik Eğitim Fakültesi Elektronik Öğretmenliği bölümünden 2008 yılında mezun oldu. 2010 yılından itibaren Kocaeli Üniversitesi Teknik Eğitim Fakültesi'nde araştırma görevlisi olarak çalışmaktadır. 2009 yılında başladığı Kocaeli Üniversitesi Fen Bilimleri Enstitüsü Elektronik ve Bilgisayar Eğitimi Anabilim Dalı'ndaki Yüksek Lisans eğitimine devam etmektedir.