İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

GÖMÜLÜ LINUX İLE DAĞITIK VERİ TOPLAMA SİSTEMİ

YÜKSEK LİSANS TEZİ

Müh. A. Gökhan GÜLTEKİN
(504021533)

Tezin Enstitüye Verildiği Tarih : 09 Mayıs 2005
Tezin Savunulduğu Tarih : 02 Haziran 2005

Tez Danışmanı : Y.Doç.Dr. Berk ÜSTÜNDAĞ
Diğer Jüri Üyeleri Prof.Dr. Tamer ÖLMEZ
Y.Doç.Dr. Turgay ALTILAR

HAZİRAN 2005

ÖNSÖZ

Tez çalışması boyunca benden yardımlarını esirgemeyen değerli danışmanım Y.Doç.Dr. B. Berk Üstündağ'a teşekkürlerimi sunarım. Ayrıca, öğrenim hayatım boyunca benden maddi ve manevi desteklerini hiçbir zaman esirgemeyen aileme çok teşekkür ederim.

Haziran 2005 A. Gökhan GÜLTEKİN

İÇİNDEKİLER

KISALTMALAR

ADC	: Analog Sayısal Dönüştürücü - Analog to Digital Converter
BIOS	: Basit Giriş Çıkış Sistemi - Basic Input Output System
CF	: Kompakt Flaş - Compact Flash
CRT	: Katod Işın Tüpü - Cathode Ray Tube
DHCP	: Dinamik Ana Bilgisayar Konfigürasyon Protokolü -
	: Dynamic Host Configuration Protocol
GLiVET	: Gömülü Linux ile Veri Toplama
GNU	: GNU's Not UNIX
GPL	: Genel Halk Lisansı - General Public Licence
GPRS	: Genel Paket Radyo Servisi - General Packet Radio Service
GPS	: Küresel Konum Bulma Sistemi - Global Positioning System
FTP	: Dosya Transfer Protokolü - File Transfer Protocol
ISO	: Uluslararası Standardizasyon Organizasyonu -
	: International Organization for Standardization
LCD	: Likid Kristal Ekran - Liquid Crystal Display
MBR	: Ana Önyükleme Kaydı - Master Boot Record
MPPT	: Maksimum Güç Noktası İzleyici - Maximum Power Point Tracker
MODEM	: Modülatör Demodülatör
PC	: Kişisel Bilgisayar - Personel Computer
PPP	: Noktadan Noktaya Protokol - Point to Point Protocol
SCSI	: Küçük Bilgisayar Seri Arabirimi -
	: Small Computer System Interface
SMPS	: Anahtarlamalı Güç Kaynağı - Switch Mode Power Supply
TCP/IP	: Haberleşme Kontrol Protokolü / İnternet Protokolü -
	: Transmission Control Protocol/Internet Protocol
TKB	: Tek Kart Bilgisayar
UART	: Üniversal Asenkron Alıcı Verici -
	: Universal Asynchronous Receiver Transmitter
UTP	: Ekranlanmamış Çift Büklümlü - Unshielded Twisted Pair
VTK	: Veri Toplama Kartı
WD	: Beyaz Cüce - White Dwarf

TABLO LİSTESİ

Sayfa No

ŞEKİL LİSTESİ

Sayfa No

GÖMÜLÜ LINUX İLE DAĞITIK VERİ TOPLAMA SİSTEMİ

ÖZET

Günümüzde kullanılan dağıtık veri toplama sistemleri coğrafi olarak farklı yerlerde bulunan verileri tek bir merkezde toplayarak burada depolanmasını ve analizlerin yapılmasını sağlamaktadır. Konularına göre ayrım yapıldığında bu sistemlere örnek olarak meteorolojik, çevresel, depremsel, yapısal ve endüstriyel veri toplama sistemleri verilebilir. Farklı sektörlerde kullanılan sistemler aynı işlevleri gerçekleştirmelerine rağmen ortak bir ağ yapısında, algılayıcı tanımlarında veya haberleşme protokollerinde buluşamamaktadırlar.

Ortak çalışan bir veri toplama yapısı olmaması, özel tasarlanmış sistemlerin hepsinin farklı yöntemler kullanmasına neden olmuştur. Farklı yöntemler ve farklı tasarımlar ise bir çok sorunu beraberinde getirmiştir. Bu sorunlardan öne çıkanlar algılayıcı bağımlılığı, yazılım bağımlılığı, uzaktan güncellenememe, elektrik şebekesi bağımlılığı (fazla güç tüketimi) ve fiziksel olarak büyük boyutlardır.

Gömülü linux ile dağıtılmış veri toplama, GLiVET, sistemi yukarıda sayılan dezavantajları tek bir sistemde avantaja çevirmek, bunun yanında kabul edilebilir maliyet değerlerine ulaşmak amacı ile tasarlanmıştır. Sistem temelinde algılayıcı bağımsızlığını (analog, lojik sinyalleri ve protokolle çalışan algılayıcıları destekleyebilen) ve dağıtık veri toplamayı barındırmaktadır. Az güç harcayan ve açık kaynak kod ile çalışan bir yapı tasarlanmış ve sonuçta ortaya istenildiği gibi konfigüre edilebilen ve uygun maliyetli bir veri toplama platformu çıkartılmıştır.

Ortaya çıkartılan kolayca bir çok son ürün elde etmeye yarayan bir alt yapı olduğundan sistem bir veri toplama platformu olarak adlandırılmıştır. Sistemde algılayıcı bağımsızlığını tam anlamıyla sağlamak için buna özel donanım katmanı oluşturulmuştur. Bu katman, standart algılayıcılardan farklı bir çıkışı olan bir algılayıcı kullanılması gerektiğinde buna özel tasarıma izin verecek şekilde tasarlanmıştır. Dağıtık veri toplama, istasyon tabanlı bir yapı ile oluşturulmuştur. Coğrafi olarak dağıtılmış olan istasyonlar, lokal yöneticiler, bulundukları alandaki verileri toplayarak sunucuya iletmektedirler.

GLiVET sistemi, sunum için özelleştirilmiş ve bir uygulama örneği hazırlanmıştır. Gerçeklenen sistem üzerinde hem geliştirmeler yapılmış hem de testler uygulanmış ve çalışma performansı gözlenmiştir.

DISTRIBUTED DATA ACQUISITION SYSTEM WITH EMBEDDED LINUX

SUMMARY

Distributed data acquisition systems used today gather data from geographically distributed locations to one main server for storing and analyzing. Meteorological, environmental, earthquake, construction and industrial data acquisition systems can be some examples to those. Although these different data acquisition systems make the same processes, they do not have a common network, sensor definitions or communication protocols.

Since there are no common data acquisition definitions, specialized systems for different sectors used different methods. Different methods and different designs caused lots of disadvantages to occur. Some of these disadvantages are sensor dependency, software dependency, no remote updates, power grid dependency (high power consumption) and physically big measures.

Distributed data acquisition with embedded linux system is designed to turn all of the disadvantages told above to advantages in one system with affordable costs. The basis of the design covers sensor independency (supports analog and logic signals and sensors that work with a protocol) and distributed data acquisition. This core is supported with a low power consuming and open source structure resulting in a very cost effective reconfigurable data acquisition platform.

The result is not a consumer product but a basic structure, a platform, which helps developing a data acquisition system in a very short time. To support sensor independency and to cover all sensors, a reconfigurable hardware layer is added to the design. To support a sensor with non-standard output, this hardware layer is redesigned specially for that. To perform distributed data acquisition, a station based structure is designed. Geographically distributed stations, local managers, acquire data from their environment and send that to the server.

GLiVET system is specialized as an application and built for the presentation. The system built is not only used for presentation but also used for designing, testing and performance monitoring.

1. GİRİŞ

Veri toplama, gerçek hayattaki olayların örneklenerek makinelerin anlayabileceği bir dile çevrilmesi işlemidir. "DAQ (Data Acquisition)" diye de isimlendirilen bu işlem, algılayıcılar tarafından analog veya sayısal elektriksel işaretlere dönüştürülen verilerin belli bir düzen ile sayısallaştırılması, toplanması ve depolanmasıdır [1].

Bir veri toplama sistemi birden çok birim ve alt sistemden oluşmaktadır. Bunlar sırasıyla aşağıdaki işlemleri gerçekleştirmektedir [1].

- Fiziksel değişkenlerin algılanması: Basınç, sıcaklık, akış, hareket...
- Sinyal düzenleme ile analog girişlere uygun değerler elde edilmesi.
- Analog sayısal dönüştürücüler ile verilerin sayısallaştırılması.
- Verilerin depolama ünitesine gönderilmesi.
- Verilerin burada işlenmesi, analiz edilmesi, görüntülenmesi ve depolanması.

Toplanacak veri algılayıcılar tarafından elektriksel işaretlere dönüştürülen verilerdir ve bunlar temel olarak analog veya sayısal veri şeklindedirler. Sayısal işaretler genellikle anahtarlardan, röle kontaklarından, TTL uyumlu çıkış verebilen elektronik elemanlardan elde edilmektedir. Uygun bir arabirim ile bu işaretler başka işleme tabi tutulmadan depolama için transfer edilebilmektedir. Analog işaretler ise cihazlardan veya basınç, sıcaklık, akış vb.. fiziksel verilerden dönüşüm yapabilen algılayıcılardan elde edilmektedir. Bu işaretler sayısal işaretler kadar kolay örneklenememektedir ve bazı işlemlere tabi tutulmak zorundadırlar. Bu işlemler, kuvvetlendirme, sinyal düzenleme, filtreleme ve sayısallaştırma aşamalarıdır [1],[4]. Her analog işaret bütün bu işlemlere tabi tutulmak zorunda değildir ve işaretler arasında bu işlemler standart değildir. Yani uygulanacak bu işlemler işarete özel tasarlanmalıdır.

Veri toplama işleminin tamamlanması için sayısallaştırılan verilerin depolama ünitesine transfer edilmesi gerekmektedir. Bu transfer işlemi de uygulamaya özeldir. Veri toplama hızına yani iletişim hızına ve toplanan veriler ile depolama aygıtı arasındaki mesafeye bağlı olarak uygun bir transfer yönteminin kullanılması

gerekmektedir. Bu iki birim -algılayıcılar ve depolama aygıtı- aynı elektronik kartta da olabilmekte, birbirlerinden kilometrelerce uzakta da olabilmektedir.

Dağıtılmış verinin toplanması, birden çok lokal olmayan verinin tek bir noktada toplanmasına denmektedir. Bu, farklı noktalardaki farklı verilerin tek bir depolama aygıtında toplanarak, tek noktadan gözlem ve sonuç çıkarılmasına olanak sağlayan bir sistemdir. Dağıtılmış veri toplama sisteminde veri transferi en önemli sistem bileşenidir. Veri transferinin yöntemi, dağıtıklığı ve lokalizasyonu etkileyecek en önemli unsurdur. Yöntemin kapsadığı alan ve fiziksel katmanı dağıtıklığı kısıtlayan öğelerdir. Örneğin mobil bir alandan, bir arabadan veri toplanması isteniyorsa kablosuz haberleşme yapılması şarttır. Genellikle dağıtık veri toplama sisteminde yaygınlığı nedeni ile internet alt yapısı kullanılmakta ve veriler internet protokolleri ile depolama ünitesine taşınmaktadır. Kablosuz haberleşmenin gerektirdiği noktalarda ise internete ulaşmak için mobil telefonlar ve GPRS devreye girmektedir. Tüm bu haberleşme alt yapısı kullanılarak dağıtılmış veri toplama uç birimlerden depolama ünitesine kadar gerçeklenmektedir.

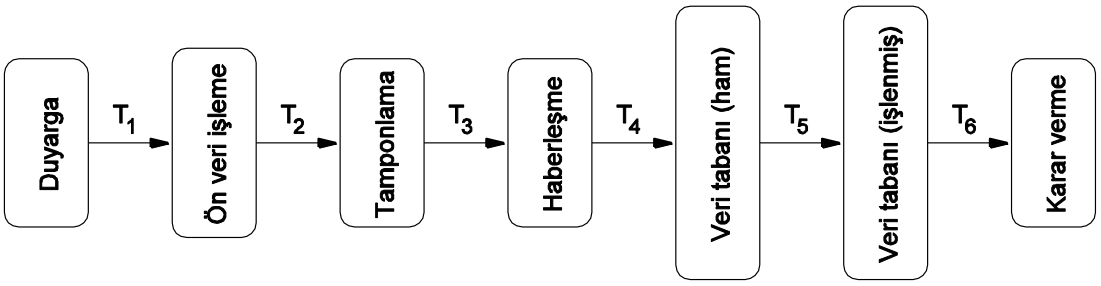
Veri transferi söz konusu olunca, sistemde oluşabilecek gecikmeler ve trafik hesapları da göz önüne alınmalıdır. Sistemde oluşabilecek gecikmeler temel olarak ikiye ayrılabilir:

1. Örnekleme gecikmesi
2. Veri transferi gecikmesi

Nyquist örnekleme teoremine göre, bir sinyalin kayıpsız örneklenebilmesi için örnekleme periyodunun minimum sinyalin bant genişliğinin iki katı olması gerekmektedir. Aksi takdirde örneklenen sinyalde kayıplar meydana gelmektedir ve sayısallaştırma başarısızlıkla sonuçlanmaktadır. Daha yüksek frekans ile yapılan örneklemeler veri büyüklüklerini etkileyecektir ve veri transferlerinde bunlara bağlı gecikmeler oluşacaktır. Örnekleme sırasında karşılaşılabilecek bir diğer gecikme ise elektronik elemanların gecikmesidir. Örneğin analog sayısal dönüştürücülerin çevrim zamanı burada direkt rol oynayacaktır ve bunun gibi diğer elemanların gecikmeleri de eklenince ortaya yine göz önüne alınması gereken zamanlamalar çıkacaktır.

Bir üst düzeyde bakıldığında, dağıtık veri toplama işleminde aslında bir ağ oluşturulduğu görülmektedir. Yıldız topoloji şeklinde ortada bir sunucu çevresinde ise birden çok dağıtılmış veri toplama noktası. Oluşturulan ağ üzerinde, bant

genişliği veya sunucunun işlem hızı nedeni ile oluşabilecek kuyruklar ve gecikmeler de bir veri toplama sistemi tasarımında göz önüne alınması gereken gecikmelerdir. Kuyrukların oluşmaması için hız sınırlayıcı etkenlerin göz önünde bulundurulup trafik analizlerinin yapılması gerekmektedir. Bir verinin algılayıcıdan depolama ünitesine kadar maruz kaldığı gecikmeleri şekil 1.1'de görülmektedir.



Şekil 1.1 Bir veri toplama sistemindeki gecikmeler ve beklemeler

- T_1: Örnekleme gecikmesi
- T_2: Veri ön işleme gecikmesi
- T_3: Tamponlama beklemesi
- T_4: Haberleşme gecikmesi
- T_5: Veri işleme gecikmesi
- T_6: Karar verme gecikmesi

Günümüzde, çalışan çok sayıda dağıtık veri toplama sistemleri bulunmaktadır. Coğrafi veri toplama sistemleri bu tip veri toplamaya verilebilecek önemli bir örnektir [2], [3]. Coğrafi bilgi sistemleri adı altında toplanan, veri toplama, depolama ve işleme sistemleri, belli coğrafyalar üzerinden farklı algılayıcılardan verileri toplayarak belli sonuçlar çıkarmak üzere geliştirilmiş sistemlerdir [3]. Genel bakıldığında bu sistemler uydu fotoğraflarından basit algılayıcılardan edinilen bilgilere kadar bir çok veri tipini incelemektedirler [2], [3]. Algılayıcı tabanlı ve sektörel bazda bakıldığında bu sistemlere örnek olarak aşağıdakiler gösterilebilir:

- Meteorolojik veri toplama sistemleri (www.meteor.gov.tr)
- Çevresel veri toplama sistemleri (http://www.usgs.gov)
- Deprem verileri toplama sistemi (http://www.koeri.boun.edu.tr, http://deprem.itu.edu.tr)

Coğrafi veri toplamanın yanı sıra başla sektörlerde de dağıtık veri toplama sistemlerinin kullanıldığı gözlenmektedir. Bunlara örnek aşağıdakiler verilebilir:

- Yapısal hareket verileri toplama sistemleri
- Endüstriyel veri toplama sistemleri

Meteorolojik veri toplama sistemleri, coğrafi bölgeler üzerinde dağıtılmış olan istasyonlardan meteorolojik veriler toplamaktadır. Bu istasyonlar, sıcaklık, nem oranı, yağmur miktarı gibi verileri merkeze gönderebilen sistemlere sahiptirler. Çevresel veri toplama sistemleri de çevre verilerini örnekleyebilecek algılayıcılara sahip istasyonlardan oluşmaktadır. Bu istasyonlar, atmosferdeki gaz oranları, kirlilik oranları, sulardaki bakteri ve diğer büyüklüklerin oranları gibi verileri toplamaktadır. Deprem verileri toplama sistemleri sismolojik değişimleri, yapısal veri toplama sistemleri yapısal hareketleri, titreşimleri ve yapıya yönelik diğer bilgileri, endüstriyel veri toplama sistemleri ise makinelerden, cihazlardan, üretim bantlarından ürünlerle, cihazlarla veya tesis ile ilgili verilerin toplanmasını kapsamaktadırlar.

Tüm bu sistemlerin ortak bir çalışma mantığı olmasına rağmen hepsi farklı farklı yöntemler kullanmaktadırlar. Kullanılan istasyonlar, sistemde kullanılan algılayıcılara özel satın alınmaktadır. Bazı sistemlerde bu her algılayıcıya yeni bir veri toplama ünitesi satın almaya kadar gidebilmektedir. Bu hem tüketilen enerjiyi arttırmakta hem de fiziksel boyutu büyütmektedir (Şekil 1.2). Ayrıca sistemin esnekliğinden bahsetmek olanaksızdır. Algılayıcı değişikliğine gidildiğinde bazen halihazırda kurulu sistemin tümünün değişmesi gerekebilmektedir. İstasyonlar dışında bakıldığında kullanılan üniteler farklı olduğundan, ortak bir veri toplama ağı dahi oluşturulamayabilmektedir. Veri toplama üniteleri ile verilen yazılımlara bağımlı kalındığından, bazı sistemlerde tüm veriler ayrı ayrı toplanım sonradan birleştirilmektedir. Sonuçta özetlemek gerekirse veri toplama sistemlerinin temel sorunları:

- Algılayıcı bağımlılığı
- Yazılım bağımlılığı
- Dağıtık olamama
- Yazılımların uzaktan güncellenememesi

Şekil 1.2 Boğaziçi Üniversitesi Kandilli Rasathanesi Bodrum istasyonu

- Elektrik şebekesi bağımlılığı
- Fiziksel olarak büyük boyutlar

Gömülü linux ile veri toplama sistemi, kısaca GLiVET, dağıtık verinin toplanmasını sağlayan bir alt yapı oluşturmak ve yukarıda tanımlanan sorunları çözebilmek amacı ile tasarlanmış bir veri toplama sistemidir. Bağımsızlığı ve açıklığı benimseyen linux işletim sistemini isminde bulunduran bu sistemin tasarımında da ön plana çıkan bu özelliklerdir. GLiVET, tamamen bağımsız, her türlü veri toplama sisteminin gerçeklenebileceği, açık kaynak kodlu bir veri toplama alt yapısı sunmaktadır.

Tasarımda çıkış noktası olan algılayıcı bağımsızlığı tüm sistemin tasarımına etki etmiş ve şekillendirmiştir. Algılayıcı bağımsızlığı, filtreleme, sinyal düzenleme, kuvvetlendirme ve sayısallaştırma katlarında da bağımsızlığı getirmiştir. Bunun sonucunda, tüm bu bağımsız öğelerin toplandığı noktaların yani uç birimlerin algılayıcıya özel tasarlanması gerekliliği oluşmuştur. Standart olacak kısım ile uç birimlerin ortak bir dilde depolama ünitesi ile yapacağı haberleşmesidir. Uç birim kartları çok basit ve alt düzey bir tasarıma sahip olacaktır ki bu basitlik özelleştirme

için en büyük gereksinimdir. Uç birimler üst düzey bir iletişimi ve üst düzey bir yönetimi sağlayacak yeteneklere sahip olamayacaklardır. Bu nedenle bu uç birimlerle haberleşebilecek, onları yönetebilecek ve veriyi lokal olarak toplayıp üst düzey yazılımlarla depolama ünitesine aktaracak istasyonlar ortaya çıkmıştır. Üç seviyeli bir ağaç şeklinde dallanan sistem alt yapısı bu sayede şekillenmiştir ve GLiVET alt yapısı ortaya çıkmıştır (Şekil 1.3).

Şekil 1.3 GLiVET sisteminin genel katmanlı yapısı

Genel olarak GLiVET:

- Dağıtık veri toplama yeteneğine sahip
- Her hangi analog veya sayısal algılayıcıdan veri toplayabilen
- Kablolu ve/veya kablosuz haberleşme yapabilen
- Az enerji harcayan - şehir şebekesine ihtiyaç duymayan ve aküden beslenebilen
- Az yer kaplayan - küçük bir kutu içine koyulabilen ve hava şartlarına dayanıklı arazi koşullarında veri toplayabilen

bir sistemdir.

GLiVET getirdiği alt yapı ile deprem verilerinin toplanması, meteoroloji verilerinin toplanması gibi coğrafyaya yayılmış büyük sistemlere uyarlanabileceği gibi bir bina içinde veya bir fabrika içinde veri toplayabilen bir yapıya uyarlanabilmektedir. Tamamen esnek bir veri toplama alt yapısı sunan GLiVET buna rağmen yetmediği yerlerde de açık kaynak kodunun getirdiği avantaj sayesinde uygulayıcıya özel hale getirilebilmektedir.

Birinci bölümde veri toplama hakkında genel bilgi verilmiştir. İkinci bölümde, gömülü linux ile veri toplama sisteminin genel yapısı anlatılarak temel elemanları tanıtılacaktır. Üçüncü bölümde GLiVET'in temel elemanlarından olan istasyonlar, yani ana düğümler anlatılacaktır. Bu bölümde, istasyonların donanımlarına değinildikten sonra, işletim sistemi, kurulumu ve konfigürasyonu anlatılacaktır. En son olarak da uygulama katmanı yazılımları tanıtılacaktır. Dördüncü bölümde lokal düğümler olan uç birimler anlatılacaktır. Bu bölümde uç birimlerin donanımlarına ve uygulama katmanı yazılımlarında değinilecektir. Beşinci bölümde istasyonlar ile uç birimler arasındaki haberleşmeyi sağlaması için tasarlanmış haberleşme protokolü anlatılacaktır. Protokolü gerçeklemek için gereken donanım ve yazılım bu bölümde tanıtılacaktır. Altıncı bölümde sunum için özelleştirilmiş GLiVET sistemi tanıtılacaktır. İstasyon ve uç birim yazılımları, tasarlanan donanımlar anlatıldıktan sonra sistem üzerinde yapılan testler ve yorumlar aktarılacaktır. Yedinci ve son bölümde varılan sonuçlar ve GLiVET sisteminin geleceğine ilişkin fikirler ve öngörüler anlatılacaktır.

2. GLİVET

2.1 Genel Yapısı

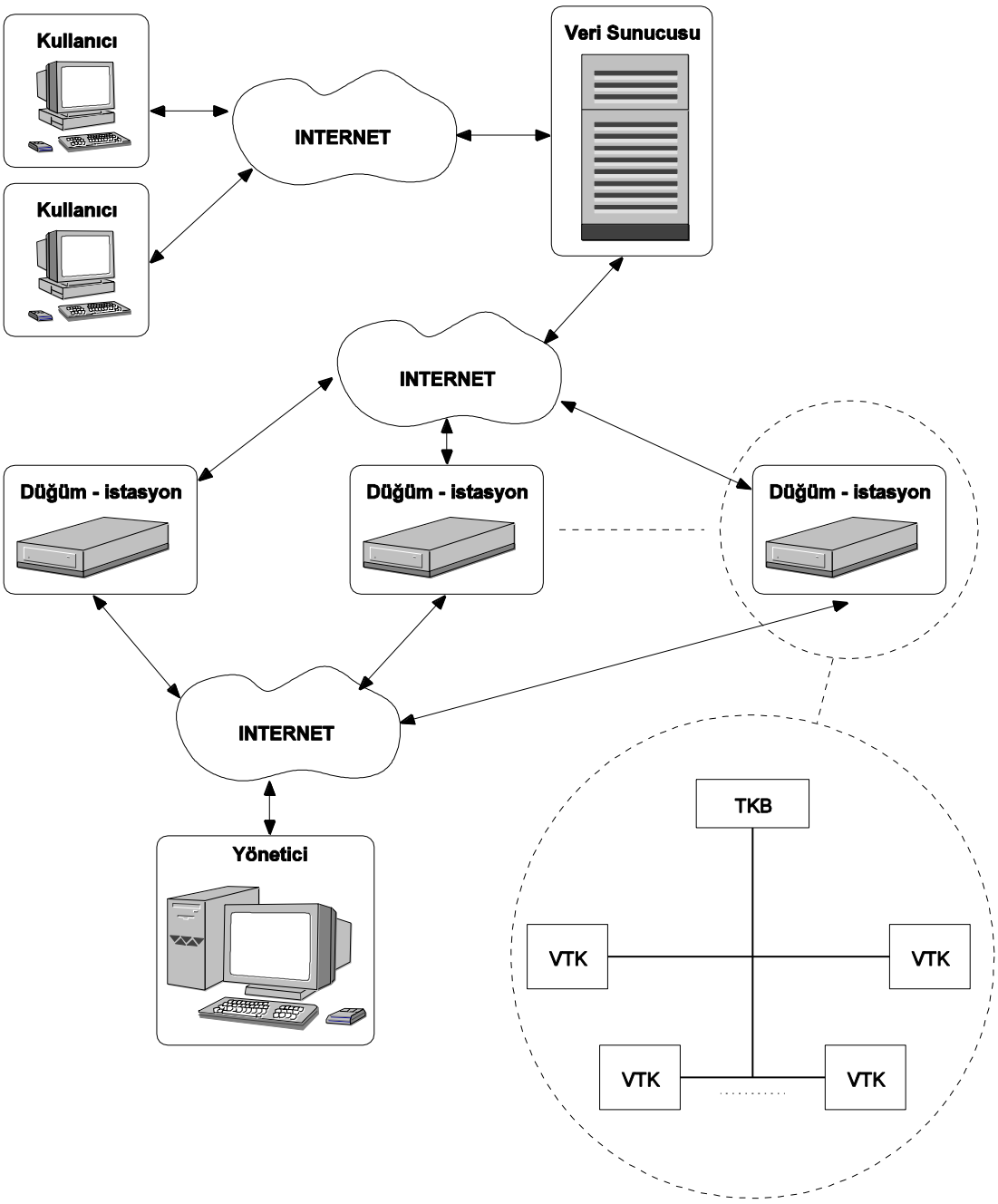
GLiVET birbirinden bağımsız tasarlanabilen üç ana katmandan oluşan bir veri toplama sistemidir (Şekil 2.1). Bu katmanlar kullanılarak değişik kombinasyonlarda veri toplama sistemi oluşturulabilmektedir. Bu katmanlar:

1. Yönetim merkezi - Sunucu
2. İstasyonlar - Düğümler
3. Uç birimler - Lokal düğümler

Oluşturulacak veri toplama sistemi bu katmanların hepsinin veya bir kaçının birleşiminden ortaya çıkmaktadır. GLiVET yapısı gereği farklı kombinasyonlarda kullanılabilmektedir. Sistemde bir tane sunucu, sunucuya bağlı birden çok istasyon ve bu istasyonlara bağlı birden çok lokal düğüm bulunabilmektedir. Dağıtılmış veya dağıtılmamış veri toplama özelliğine göre iki ana kombinasyon oluşmaktadır.

Dağıtık veri toplama için kesinlikle üç katmanında kullanılması gerekmektedir. Lokal düğümlerden alınan veriler istasyonlarda, dağıtılmış istasyonda toplanan veriler de sunucuda toplanmaktadır. Dağıtık olmayan daha özel bir yapısında lokal düğümler direkt olarak sunucuya (veya daha az özellikli normal bir PC'ye) bağlanabilir. Bu sayede bir merkezdeki lokal düğümlerden veri toplanabilmektedir. Bu konfigürasyona örnek olarak bir bina içindeki verilerin toplanması gösterilebilir. Toplanan veriler, örneğin odaların sıcaklıkları, kapıların ve aydınlatmanın açıklık kapalılık durumu, PC üzerinden gözlenebilir ve raporlanabilir.

Bir başka lokal konfigürasyon da sunucunun sistemde bulunmamasıdır. Çok fazla veri depolamanın veya raporlamanın gerekmediği daha basit uygulamalarda bu konfigürasyon seçilebilir. Lokal düğümler tüm sistemde bir tane olan istasyon düğümüne bağlanabilir ve veriler burada toplanabilir. Örnek olarak bir fabrikadaki birden çok makinenin verimliliğinin günlük raporlarını almak verilebilir. Fabrika içindeki makineler üzerinde bulunacak lokal düğümler bir istasyon modülüne bağlanabilir ve buradan her gün rapor alınabilir. Bu uygulamada tabi ki verilerin



Şekil 2.1 GLiVET sistemi

sadece o gün için önemli olduğu kabul edilmiştir çünkü sunucunun olmayışı veri depolamasını olanaksız kılmaktadır.

Geliştirilecek haberleşme protokolleri ve tasarlanacak donanımlar sistemin birden fazla konfigürasyonda kurulabilmesini olanaklı kılmaktadır. Fakat her konfigürasyon için katmanların içlerine yüklenecek yazılımlar farklılık gösterecektir. Bu farklılıklar bu tezin konusu değildir ve tez içinde sadece üç katmanlı yapı anlatılacak ve gerçeklenecektir.

2.2 Yönetim Merkezi

Yönetim merkezi temel olarak bir sunucudur. Bir alt katmandan gelen verilerin depolanması ve bu veriler üzerinde işlem yapma, grafik çizdirme, raporlama gibi görevlerden sorumludur. İnternet bağlantısı ile istasyonlar sunucuya bağlantı kurmaktadır ve verilerini göndermektedir.

İstasyon yönetimi de bu birimden yapılmaktadır. İnternet üzerinden her bir istasyona erişilip, burada değişiklikler yapılabilmektedir.

2.3 İstasyonlar – Düğümler

İstasyonlar, veri toplayan uç birimler ile yönetim merkezi arasındaki iletişim katmanıdır. Bulunduğu bölgeyi kontrol eder ve topladığı verileri internet üzerinden kablolu veya kablosuz sunucuya aktarır. Yönetim merkezinden aldığı görevleri yerine getirir, kendi kendine karar verme yetkisi yoktur. Bu sayede tek noktadan tüm sistemin kontrolü sağlanabilmektedir.

İstasyonlar temel olarak internet bağlantısı bulunan ve seri haberleşme olanağı bulunan üzerlerinde işletim sistemi çalışan küçük bilgisayarlardır. İşletim sisteminin imkanları doğrultusunda, internet haberleşmesi (http,ftp,xml,soap), seri haberleşme, sınırlı veri depolama, veri sıkıştırma ve arşivleme, vb.. yetenekleri bulunmaktadır. Az güç tüketen ve fiziksel olarak az yer kaplayan birimlerdir.

2.4 Uç Birimler - Lokal Düğümler

Uç birimler, GLiVET sisteminin algılayıcılara temas ettiği noktalarıdır. Analog veya sayısal veriyi toplayabilen ve bunları seri haberleşme protokolünü kullanarak bir üst katmana gönderebilen yapılardır. Bu kartların geliştirilme amacı, sistemi

algılayıcılardan da bağımsızlığını sağlamaktır. Kartlar genel amaçlı analog veya sayısal veri örneklemek için kullanılabileceği gibi, algılayıcılara özel de tasarlanabilmektedirler. Bu sayede istenilen tüm algılayıcılar küçük bir tasarım değişikliğiyle sisteme eklenebilecektir.

Uç birimler GLiVET sisteminde "Veri Toplama Kartları (VTK)" olarak adlandırılmaktadırlar. Bir mikrokontrolör tarafından yönetilen bu kartlar, algılayıcıya özel donanım bulundururlar. Kuvvetlendirme, sinyal düzenleme ve örnekleme katlarını üzerlerinde bulundururlar. Kullanılacak olan mikrokontrolörün seri haberleşme modülünü (UART) içermesi gerekmektedir. Mikrokontrolörün diğer yetenekleri de kartın yeteneklerini belirlemektedir.

3. İSTASYONLAR

İstasyonlar, GLiVET sisteminin, coğrafi olarak dağıtılmış parçalarıdır. Her bir istasyon, bir şekilde elektrikle beslenmek ve internete bağlı olmak zorundadır. Tüm koşullarda çalışmasını sürdürecek bir istasyon tasarlayabilmek için az enerji harcayan, fiziksel olarak küçük boyutlarda ve beklenilen tüm yetenekleri sağlayacak (uzaktan erişim, uzaktan güncelleme, uzun süreli güvenilir çalışma, güvenlik) bir çözüm bulunmuştur (Şekil 3.2).

Tek kart bilgisayara bir işletim sisteminin yüklenmesi, yazılım ve çevre donanım desteğini sağlayacağı gibi, küçük ve az enerji harcayan bir yapı da elde edilmiş olacaktır. Bir işletim sistemi, internet bağlantısını yapabilecek, güvenliği sağlayabilecek, güvenilir ve uzun zamanlı bir çalışma sağlayabilecek ve uzaktan erişime imkan vererek yazılım güncellemesine olanak sağlayacaktır.

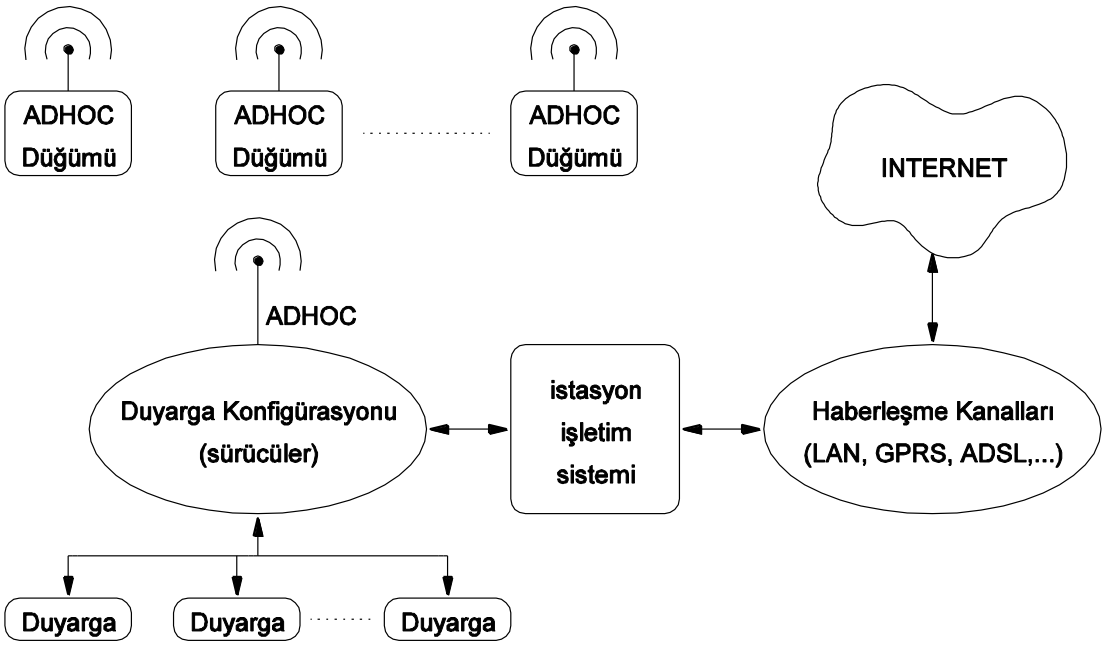
İstasyonun az enerji harcamasını gerektiren bölgeler şehir şebekesinin ulaşmadığı bölgelerdir. Elektrik şebekesinin bulunmadığı bölgelerde sistem akü ile beslenecek ve akü, güneş panelleriyle gün içinde şarj edilecektir. Şehir şebekesinin bulunduğu bölgelerde ise tasarlanabilecek bir anahtarlamalı güç kaynağı ile sistem beslenebilecektir. Bu donanımlar tez kapsamında bulunmamaktadır fakat alt yapı bu özellikleri destekleyebilecek şekilde tasarlanmıştır.

Bir bina içine yayılmış lokal bir çok düğümü yöneten bir istasyon da tasarlanabileceği gibi, arazi koşullarına uygun, su, nem, hava sıcaklığı gibi dış etkilerden etkilenmeyecek küçük bir kutu içine toplanmış (güç ünitesi, aküler, TKB, VTK'lar, algılayıcılar) bir istasyon da gerçeklenebilmektedir. Yine arazide kablosuz internet bağlantısının GPRS ile yapılabileceği gibi, bir bina içinde MODEM veya lokal ağ ile de internete çıkılabilmektedir.

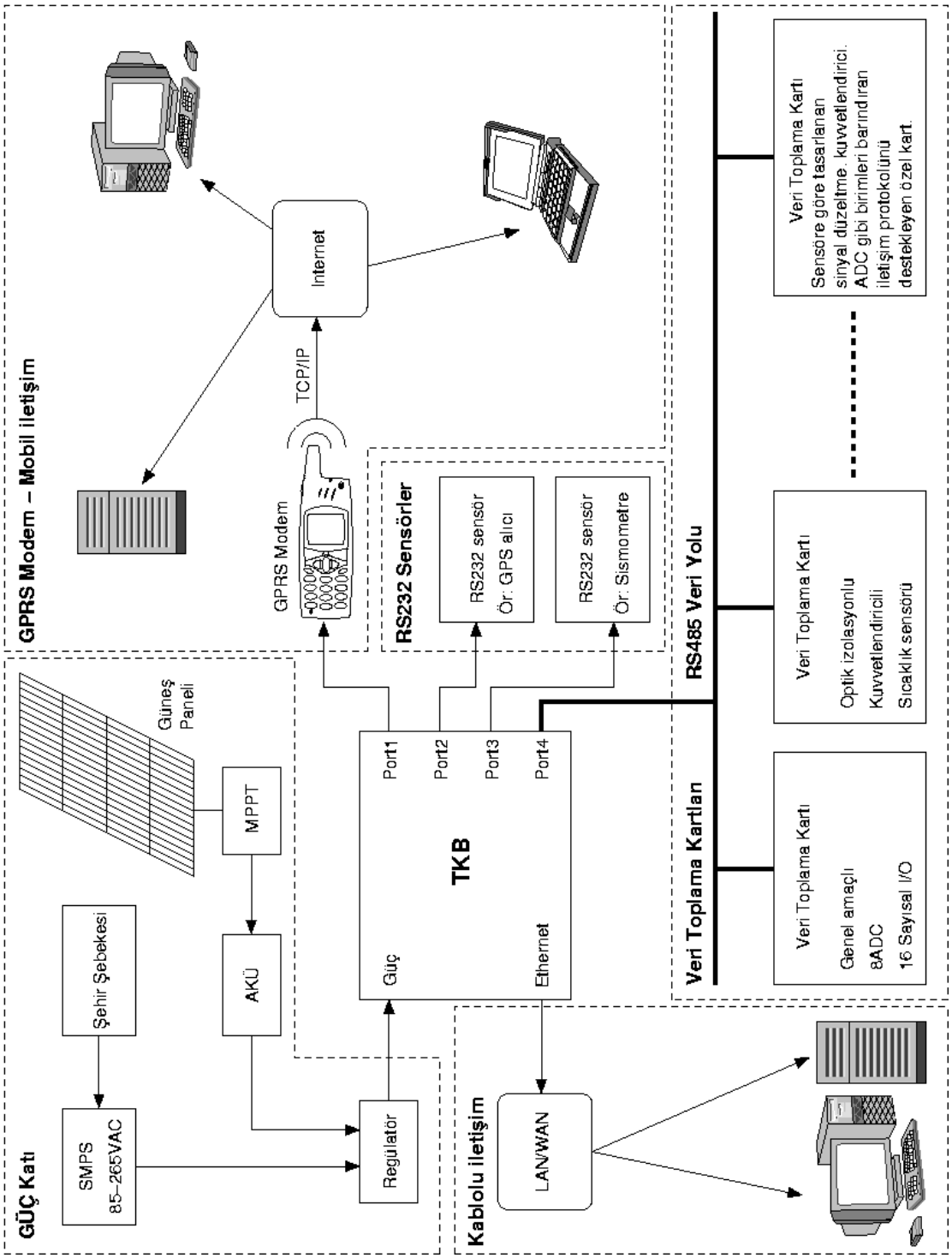
VTK’lara kablolu ulaşımın yanı sıra lokal alanda kablosuz haberleşmeyi destekleyen alt yapıya da istasyonlar sahiptir. ADHOC düğümleri ile kablosuz haberleşerek lokal alandan veri toplanabilecektir. ADHOC alt yapısı kullanılması sayesinde kapsama alanı düğümlerin pozisyonlarına göre genişletilebilmektedir, bu da düğüm sayısı ile artan bir kapsama alanı meydana getirmektedir.

Temel olarak istasyonların sorumlulukları aşağıda sıralanmıştır (Şekil 3.1):

- Kablolu haberleşme ile çevre verilerin tanımlanan protokollerle toplanması.
- ADHOC kablosuz haberleşme ağı ile (yürütülen başka bir proje) kablosuz veri toplanması.
- Toplanan verilerin tamponlanması, gerekli filtrelerden geçirilmesi, gerekiyorsa sıkıştırılması.
- Tanımlanan periyotlarda, verinin belirtilen haberleşme alt yapısı ile verinin sunucuya gönderilmesi.



Şekil 3.1 GLiVET sistemi, istasyon temel fonksiyonları



Şekil 3.2 GLiVET sistemi, istasyon ve çevre birimleri blok diyagramı

3.1 Tek Kart Bilgisayar

İstasyonu yönetecek ve üzerinde işletim sisteminin koşacağı tek kart bilgisayar, satın alınacak ve gerçeklenmeyecek bir modüldür. Bu nedenle tek kart bilgisayarın üretici bir firma kataloğundan seçilmesi gerekmektedir. TKB'nin sağlaması gereken özellikleri:

1. Yeterli işlem gücüne sahip bir mikroişlemciye sahip olması.
2. Mikroişlemcinin X86 tabanlı olması: Kaynak kodların tekrardan başka bir işlemci çekirdeğine göre derlenmesinden kurtulmak ve derlenmiş programları da bu makinada kullanabilmek için.
3. İki veya daha fazla seri haberleşme iskelesine sahip olması: Bir iskele RS485 veriyolu için bir diğeri de modem ile internet erişimini sağlamak için gerekmektedir. Daha fazla olduğu takdirde bunlara direkt seri haberleşme iskelesine bağlanabilen algılayıcılar da takılabilecektir. Ör: GPS alıcı.
4. Ethernet: Lokal alan ağından internete bağlanabilmek için gerekmektedir.
5. CF kart okuyucu: İşletim sisteminin, yazılımın ve verilerin bulunduğu ortamın compact flash bellek şeklinde kullanılabilmesi için.

TKB'nin kesinlikle sağlaması gerekmeyen fakat geliştirmede kolaylık sağlayacak özellikleri de bulunmaktadır. Bunlar:

1. IDE arabirimi: Geliştirme amaçlı CD-ROM veya hard disk bağlanabilme olanağı bulunması.
2. Ekran kartı: CRT veya LCD monitör sürebilen bir ekran kartının bulunması. Bu özellik geliştirme dışında istasyonun lokal monitörüzasyonu içinde kullanılabilir.
3. Klavye ve fare arabirimi: Geliştirme amaçlı olmasının yanında bu özellik de istasyona lokal müdahaleyi olanaklı kılmaktadır.

Yukarıda verilen özelliklere uygun bir TKB, Aaeon firması kataloğundan seçilmiş ve temin edilmiştir. Temin edilen PCM-5335 modelinin teknik özellikleri tablo 3.1'de verilmiştir [11].

Tek kart bilgisayara takılacak olan depolama ünitesi de satın alınarak (32MB CF kart), istasyon merkez yönetimi için tüm donanım tamamlanmış oldu. Güç kaynağı

olarak örnek sistem gerçeklemesinde bir PC güç kaynağı kullanılmaktadır. Bu güç kaynağının 5V çıkışı TKB'de 12V çıkışı ise RS232-RS485 dönüştürücü karta takılmaktadır. Bu ayrıca hat üzerinden VTK'lara iletilen besleme gerilimidir.

Tablo 3.1 AAEON PCM 5335 Tek Kart Bilgisayarın Teknik Özellikleri

CPU	Gömülü düşük güçlü 2.0V NS GX1-300 MHz
Sistem belleği	SDRAM SODIMM x 1, 32MB
Tümdevre	NS CS5530A
I/O tümdevresi	Winbond W83977F
BIOS	Award 256KB FLASH BIOS
Ethernet	Realtek 8139DL, 10/100 Mbps Ethernet
CF Soketi	SSD Type I Compact Flash soketi
Bekçi Köpeği	1.6 saniye aralıklı bekçi köpeği zamanlayıcı
Batarya	Lithium
Güç yönetimi	APM 1.2
Güç gereksinimi	2.0A@ +5V
Boyutlar / Ağırlık	3.55" x 3.775" (90mm x 96mm), 0.25 kg
Çalışma sıcaklığı	0 - 60 derece
Çalışma nemi	0% - 90%
Giriş çıkışlar	MIO IDE x 1, FDD x 1, KB+Fare x 1, RS-232 x 1
	RS-232/422/485 x 1, Paralel 1
	IrDA 1 x 115 Kbps SIR, IrDA 1.0 uyumlu
	USB: Bir 5*2 uçlu kablo kafası 2 USB 1.1 iskeleyi destekler
Görüntü	Çip: NS CS5530A LCD/CRT Kontrolör
	Bellek: Paylaşılan bellek 4MB
	Çözünürlük (CRT): 1024 x 768@ 16bpp
	Çözünürlük (TFT LCD): 1024 x 768@ 18 bpp
	LCD bağlantısı: 18 bit TFT LCD panel

3.2 İşletim Sistemi

İstasyonda koşacak olan işletim sisteminin, her şeyi kontrol edeceğinden ve her şeyden sorumlu olacağından bir çok yeteneğe ve özelliğe sahip olması gerekmektedir. En önemli iki kriter olarak istasyon, aylar veya yıllar boyunca çalışmak zorunda kalabilecek kadar güvenilir veya kötü niyetli insanların ulaşamayacağı kadar güvenli olmak zorundadır. Bunlar gibi işletim sistemi seçiminde kullanılan kriterler aşağıda sıralanmıştır:

1. Güvenlik.
2. Güvenilirlik.
3. Donanım bağımsızlığı, farklı işlemcilerde çalışabilme.

4. Düşük donanım özelliklerinde performanslı çalışabilme özelliği.
5. Lisans ücretinin olmaması.
6. Kaynak kodunun açık ve değişiklik imkanı sunuyor olması.
7. Yeni sürücülerin veya sistem güncellemelerinin sağlanması ve yeterli dokümantasyonun bulunması.

Yukarıdaki kriterler göz önüne alınarak yapılan incelemelerde, tüm kriterleri sağlayan Linux işletim sisteminin istasyon işletim sistemi olarak kullanılmasına karar verildi. Linux işletim sistemi zaten projenin çıkış noktası olmasına rağmen yine de bunun doğruluğunu kanıtlamak amacıyla böyle bir araştırma yapılmıştır. Linux işletim sisteminin en büyük avantajları [10]:

- Lisans ücretinin olmaması: Linux işletim sisteminin GNU-GPL (Genel Halk Lisansı) lisansına sahip olması ve bunun bir ücrete tabi olmaması birim maliyetler açısından çok büyük avantaj getirmektedir. Ayrıca işletim sisteminin yanında, diğer paketlerin de (internet uygulamaları, internet sunucuları, derleyiciler, kütüphaneler vb..) aynı lisansa sahip olması ve ücretsiz olması lisanslı işletim sistemleri ile karşılaştırılamayacak bir avantaj sağlamaktadır.
- Açık kaynak kodlu olması: Çekirdek ve destekleyici uygulamaların açık kaynak kodlu olması, koda direkt erişimi mümkün kılmaktadır. Bu sayede geliştiriciler, uygulamaların ve çekirdeğin davranışlarını gözlemleyip değiştirebilme imkanını elde etmektedirler ve performans ve güvenlik gibi önemli noktalara hakim olmaktadırlar. Diğer işletim sistemlerinde yapılması imkansız olan bu işlemler, özellikle medikal veya savunma sanayinde tüm işleyişin kontrol edilebilmesini sağlamaktadır. Bu da sadece kaynak kodun açıklığıyla gerçeklenebilmektedir.
- Güvenilirlik: Linux'ün kıskanılacak bir ayakta kalma şöhreti bulunmaktadır. Aylarca, yıllarca sistem tekrar başlatılmadan çalışabilmektedir.
- Ölçeklenirlik: Linux 2.4 çekirdeği ile daha modüler ve daha ölçeklenebilir bir yapı almıştır. Bu Linux için kaynakların sınırlı olduğu ortamlarda çalışabilmesini sağlayan çok büyük bir özelliğidir.

- Büyük programcı tabanı: Linux, internet üzerinden haberleşen oldukça fazla yazılımcı tarafından geliştirilmektedir. Linux'te şu anda bulunan oldukça fazla sayıda işlevsel program bunun bir yansımasıdır. Bu oluşum hem işlevselliği çok fazla arttırmıştır hem de yeni sürücülerin, yamaların ve bunun gibi gerekli her türlü programın yazılma zamanını kısaltmıştır.
- Destek: Linux hakkında oldukça fazla dokümantasyon bulunmaktadır. Konfigürasyonundan programcık yazılmasına, kurulumdan her komutun açıklandığı yardımcı dokümanlara kadar çok fazla kitap, makale ve internet sitesi bulunmaktadır. Bunların toplandığı en büyük kaynak ise "The Linux Documentation Project" adındaki http://www.tldp.org internet sitesidir.
- Standartlar: Linux'ün dağıtık ortamda geliştirilmesi, müthiş bir standartlaşmayı beraberinde getirmiştir. Yazılan tüm kodlar, düzenlenen tüm dokümanlar bir standart dahilindedir.
- Taşınabilirlik: Linux, herhangi bir linux sisteminde hedef makinaya göre derlenip, sonrada hedef makinaya kopyalanması yapılabilmektedir. Hedef makinaları çok fazla olması ve böyle bir özelliğe sahip olması taşınabilirliği arttırdığı gibi hata bulma ve düzeltme süreçlerini de kısaltmaktadır.

3.2.1 Linux Dağıtımı

Linux işletim sisteminin kurulacağı kesinleştikten sonra hangi dağıtımın kurulacağına karar verilmesi gerekmektedir. Bir linux dağıtımı, linux çekirdeğinin yanı sıra uygulama programlarının, yardımcı programları, program dili kütüphanelerinin, kaynak kodları vb.. bir çok şeyin paketlenerek kullanıcıya sunulan halidir [6],[8]. Genellikle dağıtımlar ticarileşmiştir ve bu dağıtımlar, dağıtımı ücretsiz vererek, verdikleri desteklerden ücret almaktadırlar. Sunucu veya iş istasyonları için oldukça fazla dağıtımı olan linux işletim sistemi (Red Hat, Suse, Mandrake, Turbo Linux, Debian, Caldera...) gömülü ortamlar için de çok yaygınlaşmıştır [6],[8]. Gömülü sistemlerde de oldukça fazla alternatifi olan linux dağıtımları, ticarileşmenin getirdiği avantajla bu yönde de kendini çok geliştirmiştir. Örneğin Sharp şirketinin Zaurus modeli ile öncülük ettiği PDA pazarında linux kurulu makineler oldukça artmıştır [12]. Bir başka örnek ise Montavista şirketinin öncülüğünde, Linux işletim sistemi ile çalışan daha akıllı mobil telefonlar geliştirmek için kurulan konsorsiyum gösterilebilir [13].

Bir linux kurulumu yapmak için temel olarak üç yol bulunmaktadır: Ticarileşmiş bir ürünü satın almak, ücretsiz bir dağıtım temin etmek veya sıfırdan linux'ü oluşturmak. Ticarileşmiş bir ürün satın almak bu proje için gereksiz bir maliyet açacağından bu yola başvurulmamıştır, zaten oluşturulan bilgi birikimi nedeniyle de böyle bir şeye gerek olmamıştır. Sıfırdan linux'ü kurmak, çok fazla zaman alabileceğinden tercih edilmemiştir. "Linux From Scratch" projesinde de anlatıldığı gibi, bu işlem için çekirdeğin derlenmesi, dosya sisteminin oluşturulması, gerekli programların derlenip kopyalanması ve ön yükleyici programın kurulması gerekmektedir [9]. Bunun yerine ilk önce ücretsiz ve gereksinimleri sağlayabilecek bir dağıtım arayışına girilmiştir. Aranılan Linux dağıtımının temel gereksinimleri:

- Lisans ücretinin olmaması.
- Ethernet desteğinin olması.
- Temel internet ve ağ programlarını bulundurması: ssh, ftp, http...
- Basit kurulumunun olması.
- 32MB CF diske sığacak büyüklükte olması.
- 32MB ram ile performanslı çalışabilmesi.
- X86 çekirdeğine göre derlenmiş olması.
- X'in (grafik ekranın) bulunmaması.

Bu gereksinimlere göre yapılan araştırmalar sonucunda "White Dwarf Linux" adındaki Slackware tabanlı bir gömülü sistem Linux dağıtımı bulunmuştur [14]. Bu dağıtım ilk bulunduğunda http://www.whitedwarflinux.com internet adresinde ticarileşmemiş bir proje iken, yaklaşık 2 sene süren tez süresince bir şirket tarafından satın alınmış ve ticarileştirilmiştir. Blast Internet Sevices adındaki bu şirket eğitim, kurulum gibi destekler üzerinden ücretlendirme yapmaktadır fakat dağıtımı hala ücretsiz olarak internet sitesinden sağlamaktadır.

- Linux çekirdeği 2.4.24.
- Tamamen grafik tabanlı paket-bazlı kurulum.
- glibc 2.2.5 standart.
- TCP/IP araçları - ftp, telnet, ssh, ping, pure-ftpd, sshd, netfilter ve ip route.

- Apache 1.3.x internet sunucusu.
- glibc 2.2.5 için gcc 2.3.2
- PPP sürücüleri ve programcıkları.

3.2.2 Kurulum

Seçilen Linux dağıtımı, http://www.blast.com/wd_install.html internet sitesindeki kurulum aşamaları izlenerek 32MB'lik CF karta kurulmuştur. Yapılan işlemler sırasıyla aşağıda verilmiştir.

Kurulum aşamaları:

1. BIOS ayarları saat dahil olmak üzere yapılmıştır. CDROM'dan ilk yükleme aktif edilmiştir.
2. Kullanılan TKB'nin CDROM desteği olduğundan CDROM'dan kurulum tercih edilmiştir. White Dwarf Linux ISO kurulum CD görüntüsü internet sitesinden indirilerek bir CD yazma programıyla boş bir CD'ye yazılmıştır.
3. WD kurulum CD'si takılarak TKB yeniden başlatılmıştır. CD'den ilk yükleme yapılarak, grafik özelliği kazandırılmış kurulum ekranındaki menüler takip edilerek kurulum tamamlanmıştır.
4. CD çıkartılarak TKB yeniden başlatıldığında white dwarf linux yüklenir ve ilk giriş ekranı görüntülenir.

Basit bir kurulumu olmasına rağmen, kurulum sonrasında sistem sorunsuz olmamaktadır. Başka bir TKB için özelleştirildiğinden, karşılaşılan ilk ve tek sorun işletim sisteminin ethernet kartını tanımamasıdır. Bunun nedeni ise çekirdeğin kartı desteklememesidir. Kurulumun tamamlanması için çekirdeğin yeniden gerekli destek ile derlenmesi, bunun CF karta kopyalanması ve ön yükleyiciye bu çekirdeğin tanıtılarak sistemin bu çekirdek üzerinden yüklenmesi gerekmektedir.

3.2.3 Çekirdek Derleme

Çekirdek derlenmesi için ilk önce çekirdeğin kaynak kodlarının temin edilmesi gerekmektedir [15]. Bunun için kernel.org internet sitesinin çekirdek arşivlerinden, ftp://ftp.kernel.org/pub/linux/kernel/v2.4/ adresinden 2.4.22 versiyonlu çekirdek kaynak kodları indirilmiştir.

Derleme öncesinde ise çekirdekte bulunması istenen modüller seçilerek konfigürasyon dosyası oluşturulmalıdır. Bu işlem için dört ayrı seçenek bulunmaktadır [6], [15]:

- config: En az kullanıcı yardımcısı konfigürasyon arayüzüdür. Sırayla tek tek sorular sorarak konfigürasyonun yapılmasını sağlar.
- oldconfig: Eski konfigürasyon dosyasından okuyup küçük değişikliklere izin veren bir arayüzdür.
- menuconfig: Karakter temelli bir menü sistemi ile bir arayüz sunmaktadır.
- xconfig: X pencere sistemi temelli, grafik ekran bir arayüz sunmaktadır.

Derleme için "make menuconfig" komutu kullanılarak, konfigürasyon programı çalıştırılmaktadır [15]. Burada yapılması gereken en önemli ayar, TKB'nin ethernet kartının, "RealTEK", konfigürasyona eklenmesidir. TKB tarafından desteklenmeyen modüller de (ör: SCSI) konfigürasyondan çıkarılmalıdır ki gereksiz yer tüketimi engellensin [15]. Konfigürasyon kaydedildikten sonra derleme işlemine geçilmektedir.

Modüllerin birbirleriyle bağımlılıklarını çıkartmak için "make dep" komutu çalıştırılmalıdır. Bu işlem sonucunda bağımlılık dosya oluşturulmuş olacaktır. "Bundan sonra make clean" komutu ile genel obje dosyaları silinmektedir. En sonunda çekirdek derlemesi "make bzImage" komutu ile gerçekleştirilir. Herşey sorunsuz gerçekleştiyse yeni çekirdek "/usr/linux/linux-2.4.22/arch/i386/boot/" altında oluşturulmuş olacaktır [15].

3.2.4 Ön Yükleyicinin Ayarlanması

Çekirdek derlendikten sonra çekirdeğin hedef sisteme yani TKB'ye kopyalanması gerekmektedir. Depolama ünitesinin CF kart olarak tercih edilmesinin en büyük nedeni sökülüp başka bir bilgisayarda kolayca okunup yazılabilmesidir. Ethernet bağlantısı çalışmadığından dosya kopyalama ya disketle, ya CD ile ya da CF'karta direkt yazma ile olacaktır. En kolay yol olan CF karta yazmak için, CF kart TKB üzerinden sökülerek ana sisteme bağlı olan bir CF kart okuyucusuna takılır. Bu andan itibaren tüm dosya sistemi ana bilgisayar üzerinde görülebilmekte, okunup yazılabilmektedir.

CF kart "mount /dev/sda1 /mnt/flash" komutu ile sisteme bağlandıktan sonra "cp bzImage /mnt/flash/boot/bzImage-2.4.22" komutu ile hedef işletim sisteminin /boot klasörüne kopyalanır. Eski çekirdeğin üstüne yazılmamasına dikkat edilmektedir. "umount /dev/sda1" komutu ile CF kartın bağlantısı çözülerek, kart tekrar TKB üzerinde takılır [18].

TKB açılarak, eski çekirdekten yüklenmesi sağlanır. Bu aşamadan sonra ön yükleyici (bootloader) programına yeni çekirdeğin tanıtılması gerekmektedir. WD linux'ta ön yükleyici programı olarak LILO kullanılmıştır. LILO konfigürasyon dosyasına (/etc/LILO.conf) yeni çekirdeğin ismi eklenerek ön yükleyicinin yeni çekirdeği tanıması sağlanır [6], [16], [17]. Konfigürasyon dosyası aşağıdaki gibidir:

```
# LILO configuration file
#
# Start LILO global section
# append="mem=128M"
# reserve=0x280,32 ether=10,0x280,eth0"
vga = normal    # force sane state
# ramdisk = 0     # paranoia setting
# End LILO global section

prompt
timeout=0
default=NewKernel
boot = /dev/hdc
map=/boot/map
install=/boot/boot.b
message=/boot/boot_message.txt
linear

  # Linux bootable partition config begins
   image = /wdboot
   root = /dev/hdc1
   label = linux
   read-only # Non-UMSDOS filesystems should be mounted read-only
             # for checking
  # Linux bootable partition config ends

  # Linux bootable partition config begins
   image = /wdboot.old
   root = /dev/hdc1
   label = linux.old
   read-only # Non-UMSDOS filesystems should be mounted read-only
             # for checking
  # Linux bootable partition config ends

  # New kernel
   image = /boot/bzImage
   root = /dev/hdc1
   label = NewKernel
   read-only # Non-UMSDOS filesystems should be mounted read-only
             # for checking
  # New kernel ends
```

Bu işlemden sonra yapılması gereken konfigürasyonda yazılan bu bilginin Ana İlkyükleme Kaydına (Master Boot Record - MBR) yazılmasıdır [16]. MBR, Bir sistem açıldığında ilk okunan bölümdür. Buraya yazılı olan kod ilk deva çalıştırılır ve bu işletim sisteminin yüklenmesini sağlar. LILO TKB'de ilk çalışacak koddur ve Linux çekirdeğinin yüklenmesini sağlamaktadır. "LILO" komutu ile yapılan değişikliklerin MBR'ye yüklenmesi sağlanır. Bir sonraki açılışta Linux yeni derlenen çekirdek ile açılacaktır ve ethernet kartının sürücüleri de yüklenmiş olacaktır.

3.3 İşletim Sisteminin Konfigürasyonu

İşletim sistemi kurulduğunda bir çok özelliği ayarlanmış ve çalışmaya hazır bir halde kurulmaktadır. Makine ismi, komut satırı yazısı, açılış mesajı gibi basit değişikliklerdin dışında herhangi bir değişikliğe gerek duymamaktadır. Yapılması gereken tek ve en önemli ayar ağ ayarlarıdır. TKB'nin ip adresini belirleyip, lokal ağa ve internete ulaşmasını sağlayacak ayarların yapılması gerekmektedir.

3.3.1 Ağ Ayarları

WDLinux'te bu ayarlar açılış sırasında çalıştırılan rc.inet1 programcığında bulunmaktadır. Aslında ifconfig ve route komutlarını belirlenen parametrelere göre çalıştıran programcık, TKB'nin IP adresini (IPADDR), alt ağ maskesini (NETMASK), genel yayın adresini (BROADCAST), ve ağ geçidini (GATEWAY) belirler [6],[8]. Statik IP kullanmak istenilmiyorsa, DHCP sunucusundan sistemin IP alması da sağlanabilmektedir. Bunun için rc.inet1 programcığındaki DCHP sunucusu ile ilgili bölümleri aktif etmek gerekmektedir.

```
#!/bin/sh
#
# rc.inet1      This shell script boots up TCP/ip

#load some functions
. /etc/rc.d/rc

# Attach the loopback device.
run "/sbin/ifconfig lo 127.0.0.1";
run "/sbin/route add -net 127.0.0.0 netmask 255.0.0.0 lo";

IPADDR="192.168.0.80"
NETMASK="255.255.255.0"
BROADCAST="192.168.0.255"
GATEWAY="192.168.0.1"

run "/sbin/ifconfig eth0 $IPADDR broadcast $BROADCAST"\
     "netmask $netmasK";
run "/sbin/route add default gw $GATEWAY netmask 0.0.0.0"\
```

```
     "metric 1";

#Echo "Looking for DHCP server(will stop trying after 60"\
       "seconds)..."#if [ -x /sbin/dhcpcd ]; then
#  run "rm -f /etc/dhcpc/dhcpcd-eth0.pid";
#  run "/sbin/dhcpcd -t 60 eth0";

#  if [ $? -gt 0 ]; then
#    echo not found, DHCP disabled
#    echo "no" >$TMP/DHCP
#  else
#    echo found DHCP server, getting ip info...
#    echo "yes" >$TMP/DHCP
#    route;
#    ifconfig eth0;
#  fi
#fi
```

Bu ayarları değiştirmek için rc.inet1 dosyasındaki ilgili yerleri değiştirip dosyayı kaydettikten sonra, bu programcığın çalıştırılması gerekmektedir. rc.inet1 programcığı çalıştırıldıktan sonra yeni ağ konfigürasyonu etkin olacaktır.

3.3.2 İnternet Bağlantısı

Tek kart bilgisayarın internet bağlantısı çeşitli şekillerde yapılabilmektedir. Bunlar:

- Lokal alan ağı
- GPRS
- ADSL
- Çevirmeli bağlantı MODEM'i

Tek kart bilgisayar bu dört bağlantı şeklini gerçekleyebilecek donanıma bağlantı desteğine ve iskelelerine sahiptir. Hangi bağlantı şeklinde hangi iskelenin kullanılabileceği tablo 3.2'de verilmiştir.

Tablo 3.2 İnternet Bağlantısı İçin Kullanılabilecek İskeleler

Bağlantı Şekli	Ethernet	USB	RS232
Lokal alan ağı	x	-	-
GPRS	-	x	x
ADSL	x	x	-
Çevirmeli MODEM	-	-	X

GPRS ile gerçekleştirilen internet bağlantısı bağlantı programcıkları ile kurulmaktadır. Linux işletim sistemi için yazılmış telefon veya GPRS modem programcıkları kullanılmalıdır.

Çevirmeli MODEM veya ADSL MODEM kullanılacak ise ilgili MODEM'in sürücüsü bağlantı kurulması için kullanılmalıdır.

Lokal alan ağı üzerinden bağlantı, gerekli ayarlamaların yapılmasından sonra hemen kurulabilmektedir ve bir sürücüye ihtiyaç duymamaktadır.

Tüm bağlantılar için ağ ayarları bölümünde anlatılan ayarlar yapılmalıdır.

3.4 GLiVET İstasyon Yazılımları

GLiVET istasyon yazılımları, verilerin uç birimlerden toplanmasından, tamponlanmasından, gerekli filtrelerin uygulanmasından, gerekiyorsa tamponlanan verilerin sıkıştırmasından ve en sonunda sunucuya tamponlanan verilerin gönderilmesinden sorumludur. Tüm bu işlemlerden çalışmanın bütünlüğünü sağlamak ve Linux işletim sisteminin gücünden yararlanmak için kabuk programcıkları şeklinde yazılmıştır. Gerekli olan işlemler kabuk programcığına istenildiği şekilde eklenerek sisteme yaptırılabilmektedir.

Örneğin bir olay gözlemlenmek isteniyorsa ve olay anında sadece veri transferi yapılmak isteniyorsa aşağıdaki gibi bir kabuk programı yazılabilir:

```
- Veriyi uç birimden oku
- Tampon dosyaya kaydet
- Her 10 veride bir filtreyi çalıştır
- - Eğer filtre olay tespit ederse
- - - Tampon dosyayı gönder
- - Tespit etmezse
- - - Tampon dosyayı sil
- Başa dön
```

Bu sayede gereksiz veri transferinden kurtulmuş olunmaktadır. Kabuk programcığında istenildiği gibi dallanma ve döngü işlemleri yaptırılabildiğinden her türlü filtreleme, karar algoritmaları, tamponlama değerlerinin değiştirilmesi vb.. gibi ayarlar uzaktan bu programcığın modifiye edilmesi ile yapılabilmektedir.

Uç birimler ile haberleşmeyi sağlayan programlar C programlama dilinde yazılmıştır [7]. Kısa genel yardımcı programların yanında sunum için özelleştirilmiş sistem için de programlar yazılmıştır.

Yardımcı programlar:

- readADC
- scanbus

- changeADDR

Geliştirilen özelleştirilmiş sistem dosyaları (6. bölümde anlatılacaklar):

- sample2file
- sample
- glivet_config
- glivet.log

3.4.1 readADC.c

readADC programı, belirtilen uç birim adresindeki uç birimin belirtilen kanalındaki veriyi okur ve ekrana yazar. İki parametre kabul etmektedir: uç birim adresi ve kanal numarası. Uç birim adresi 1-15 arasında, kanal numarası ise 1-4 arasında verilmelidir.

Ör:

```
glivet ~# ./readADC 2 3
glivet ~# 2. uc birim, 3. kanal: 83
```

readADC.c:

```
/************************************************
/
/    GLiVET - Gömülü Linux İle Veri Toplama
/
/    readADC() - Tek Kart Bilgisayar uygulama katmanı
/    programı
/
/    Veri yolu üzerindeki uç birimlerden, istenilen adresteki
/    istenilen kanaldaki veriyi okur.
/
/    V1.6
/
/    Geliştiren: A. Gökhan Gültekin
/
/************************************************/

#include <stdio.h> /* Standard input/output definitions */
#include <unistd.h> /* UNIX standard function definitions */
#include "glivet.h"

//---------------------------------------------------------------
// readADC() - 8 bitlik ADC verisini okur.
// Parametreler: Uçbirim adresi, ADC kanal numarası (1-4)
//---------------------------------------------------------------

unsigned char readADC(int adres, int kanal, int fd)
{
     int bufindex;
     char c[2] = "";
```

```
     unsigned char s[20] = "";

     c[0] = 0x02;
     send_char(fd, c);  // Başla karakteri gönderildi

     c[0] = (char) adres;
     send_char(fd, c);  // Adres gönderildi

     s[0] = get_char(fd);     // Onay karakteri alındı
     if (s[0] != 0x04)
          {
          c[0] = 0x03;
          send_char(fd, c);   // Bitti gönderildi

          printf("Adresteki uc birim bulunamadi...\n");
          exit();
          }

     c[0] = (char) (kanal * 16);
     send_char(fd, c);        // Komut gönderildi

     c[0] = 0x04;
     send_char(fd, c);        // Tamam gönderildi

     s[0] = 0x00;
     bufindex = 0;

     // Tamam gelene kadar veri alınır. Max 100 byte.
     while (1)
          {
          s[bufindex] = get_char(fd);     // Veri alındı
          if (s[bufindex] == 0x04) break;
//        printf("%d",s[bufindex]);
          bufindex++;
          }

     c[0] = 0x03;
     send_char(fd, c);        // Bitti gönderildi

     return s[0];
}

//---------------------------------------------------------------------
// main()
//---------------------------------------------------------------------
int main(int argc, char *argv[])
{
     int fd, adres, kanal;
     unsigned char n;

     if (argc != 3)
     {
     printf("\nreadADC - Ucbirimdeki bir ADC kanalini okur.\n");
     printf("-------------------------------------------\n");
     printf("Kullanimi: readADC adres kanal\n");
     printf("adres: Uc birim adresi [1-16].\n");
     printf("kanal: Uc birimdeki ADC kanali [1-4].\n\n");
     printf("GLiVET V1.0 - A Gokhan Gultekin\n");
     exit();
     }
```

```
    fd = open_port();

    configure_port(fd);

    clear_RTS(fd);  // idle state.

    adres = str2int(argv[1]);
    kanal = str2int(argv[2]);

    if (((adres < 1) || (adres > 15)) || \
        ((kanal < 1) || (kanal > 4)))
    {
    printf("Kanal [1-4] adres [1-15] arasinda girilmelidir!\n");
    exit();
    }

    n = readADC(adres,kanal,fd);

    printf("\n%d. uc birim, %d. kanal: %d\n\n", \
        adres, kanal, n);

    close(fd);
}
```

3.4.2 scanbus.c

scanbus programı, veri yolu üzerindeki 1-15 adres alanını tarayarak, adreslerdeki uç birim varlığını sınamaktadır. Birden on beşe kadar listeyi alt alta oluşturarak ekrana yazmaktadır ve adreslerin yanlarında da uç birimlerin varlıklarını belirtmektedir.

Ör:

```
glivet ~# ./scanbus
1. Uc birim: VAR
2. Uc birim: ---
3. Uc birim: ---
4. Uc birim: ---
5. Uc birim: ---
6. Uc birim: VAR
7. Uc birim: ---
8. Uc birim: ---
9. Uc birim: ---
10. Uc birim: ---
11. Uc birim: ---
12. Uc birim: ---
13. Uc birim: ---
14. Uc birim: ---
15. Uc birim: ---
```

scanbus.c:

```
/************************************************
/
/    GLiVET - Gömülü Linux İle Veri Toplama
/
/    scanbus() - Tek Kart Bilgisayar uygulama katmanı
/    programı
/
/    Veri yolu üzerindeki adresleri tarayarak uç
```

```
/    birimlerin varlığını onaylar.
/
/    V1.6
/
/    Geliştiren: A. Gökhan Gültekin
/
/************************************************/

#include <stdio.h> /* Standard input/output definitions */
#include <unistd.h> /* UNIX standard function definitions */
#include "glivet.h"

//---------------------------------------------------------------------
// scanvtk() - Adresteki uç birim varlığını doğrular. Adreste uç
// birim varsa 1 yoksa 0 döndürür.
// Parametreler: Uçbirim adresi
//---------------------------------------------------------------------

int scanvtk(int adres, int fd)
{
     int bufindex;
     char c[2] = "";
     unsigned char s[20] = "";

     c[0] = 0x02;
     send_char(fd, c);  // Başla karakteri gönderildi

     c[0] = (char) adres;
     send_char(fd, c);  // Adres gönderildi

     s[0] = get_char(fd);     // Onay karakteri alındı
     if (s[0] != 0x04)
     {
          c[0] = 0x03;
          send_char(fd, c);   // Bitti gönderildi
          return 0;
     }

     else
     {
          c[0] = 0x03;
          send_char(fd, c);   // Bitti gönderildi
          return 1;
     }
}

//---------------------------------------------------------------------
// main()
//---------------------------------------------------------------------
int main()
{
     int fd, i;
     unsigned char n;

     printf("\nscabbus - 1-15 adresleri arasinda veriyolundaki \
     uc birimleri tarar ve varliklarini tesbit eder.\n");
     printf("-----------------------------------------\n");
     printf("GLiVET V1.0 - A Gokhan Gultekin\n\n");

     fd = open_port();
```

```
     configure_port(fd);

     clear_RTS(fd);  // idle state.

     for(i=1;i<16;i++)
     {
          if (scanvtk(i,fd))
               printf("%d. Uc birim: VAR\n", i);
          else
               printf("%d. Uc birim: ---\n", i);
     }
     close(fd);
}
```

3.4.3 changeADDR.c

changeADDR programı uç birim adresini değiştirmeye yaramaktadır. İki parametre kabul etmektedir: eski adres, yeni adres. Adresler 1-15 aralığında girilmelidir.

Ör:

```
glivet ~# ./changeADDR 3 6
glivet ~# 3 adresindeki uc birim adresi 6 olarak degistirildi...
```

changeADDR.c:

```
/************************************************
/
/    GLiVET - Gömülü Linux İle Veri Toplama
/
/    changeADDR() - Tek Kart Bilgisayar uygulama katmanı
/    programı
/
/    Veri yolu üzerindeki uç birimlerin adreslerini değiştirir.
/
/    V1.6
/
/    Geliştiren: A. Gökhan Gültekin
/
/************************************************/

#include <stdio.h> /* Standard input/output definitions */
#include <unistd.h> /* UNIX standard function definitions */
#include "glivet.h"

//---------------------------------------------------------------------
// changeADDR() - Uc birim adresini değiştirir.
// Parametreler: Uçbirim adresi, Yeni uç birim adresi
//---------------------------------------------------------------------

unsigned char changeADDR(int adres, int adres2, int fd)
{
     int bufindex;
     char c[2] = "";
     unsigned char s[20] = "";

     c[0] = 0x02;
     send_char(fd, c);  // Başla karakteri gönderildi

     c[0] = (char) adres;
```

```
     send_char(fd, c);  // Adres gönderildi

     s[0] = get_char(fd);     // Onay karakteri alındı
     if (s[0] != 0x04)
          {
          c[0] = 0x03;
          send_char(fd, c);   // Bitti gönderildi

          printf("Adresteki uc birim bulunamadi...\n");
          exit();
          }

     c[0] = (char) ((6 * 16) + adres2);
     send_char(fd, c);        // Komut gönderildi

     c[0] = 0x04;
     send_char(fd, c);        // Tamam gönderildi

     s[0] = 0x00;
     bufindex = 0;

     // Tamam gelene kadar veri alınır. Max 100 byte.
     while (1)
          {
          s[bufindex] = get_char(fd);     // Veri alındı
          if (s[bufindex] == 0x04) break;
          bufindex++;
          }

     c[0] = 0x03;
     send_char(fd, c);        // Bitti gönderildi

     return s[0];
}

//---------------------------------------------------------------------
// main()
//---------------------------------------------------------------------
int main(int argc, char *argv[])
{
     int fd, adres, adres2;
     unsigned char n;


     if (argc != 3)
     {
     printf("\nchangeADDR - Ucbirimin adresini değiştirir.\n");
     printf("-------------------------------------------\n");
     printf("Kullanimi: changeADDR adres adres2\n");
     printf("adres: Uc birim adresi [1-16].\n");
     printf("adres2: Uc birimin yeni adresi [1-16].\n\n");
     printf("GLiVET V1.0 - A Gokhan Gultekin\n");
     exit();
     }

     fd = open_port();

     configure_port(fd);

     clear_RTS(fd);  // idle state.
```

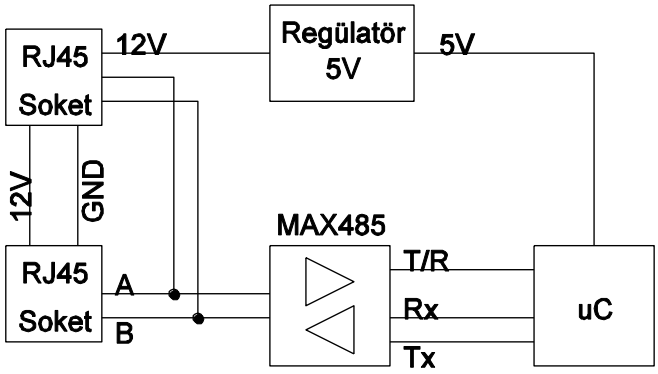
```
        adres = str2int(argv[1]);
        adres2 = str2int(argv[2]);

        if (((adres < 1) || (adres > 15)) || \
             ((adres2 < 1) || (adres2 > 15)))
        {
             printf("Adresler [1-15] arasinda girilmelidir!\n");
             exit();
        }

        n = changeADDR(adres,adres2,fd);

        printf("\n%d adresindeki uc birim adresi %d olarak"\
                  "degistirildi...\n\n", adres, n);

        close(fd);

}
```

4. UÇ BİRİMLER

Uç birimler, diğer adlarıyla veri toplama kartları, sistemim donanım bağımsızlığını oluşturmak için tasarlanmışlardır. Her algılayıcının kendine has çıkışı vardır. Bu çıkış analog veya sayısal olabilir, kuvvetlendirilmiş veya çok zayıf bir sinyal olabilir, gürültü içeriyor olabilir. Bu nedenle standart sistemlerde olan yapıya her algılayıcı takılamayabilmektedir. Sistemi algılayıcı çıkışlarından bağımsızlaştırmak ve tüm algılayıcıların bağlanmasına olanak sağlamak için VTK'lar geliştirilmiştir. VTK'lar, belirli bir kısmın standart tutulup geri kalan donanımın özgürce algılayıcıya özel tasarlanmasını sağlamaktadır.

4.1 Minimum Donanım

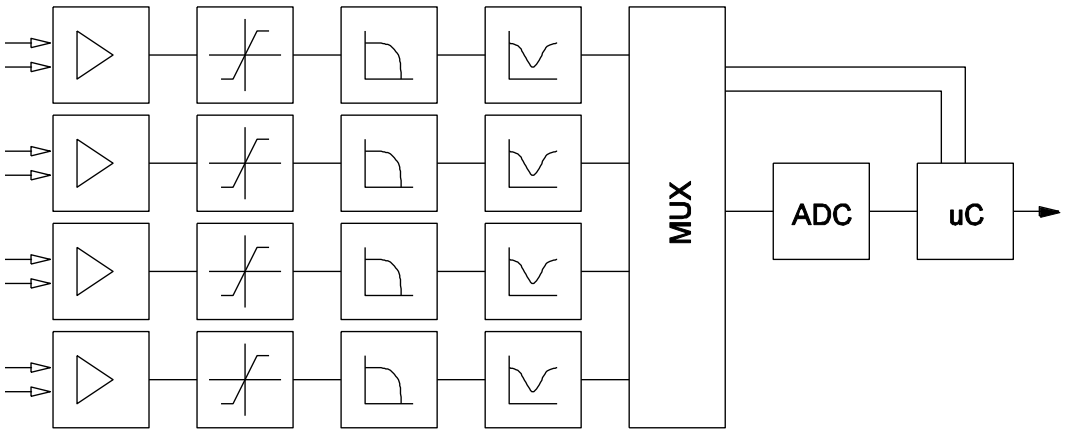
Bir veri toplama kartının minimum donanım gereksinimi veri yolu haberleşmesini gerçekleyecek kısmıdır. Kartın haberleşme protokolü ile TKB ile haberleşebilme yeteneği olduktan sonra, kart üzerindeki kuvvetlendiriciler, filtreler, ADC'ler istenildiği gibi özelleştirilebilmektedir. Haberleşme protokolünü gerçeklemek için ise hat sürücü devresi ve protokolü gerçekleyebilecek bir mikrokontrolör gerekmektedir (Şekil 4.1) [5], [25], [26].



Şekil 4.1 Veri toplama kartında olması gereken minimum donanım

4.2 Örnek Veri Toplama Kartları

Özelleştirilmiş veri toplama kartına bir örnek şekil 4.2 'de verilmiştir. Bu kartta basınç algılayıcısı gibi düşük gerilim farkı üretebilen algılayıcıların bağlanabildiği bir uç birim tasarlanmıştır. Bu tür algılayıcılar mikro kontrolöre bağlanamayacaklarından ek katlar gerekmektedir. Bu tasarımda bu katlar sırasıyla kuvvetlendirme, sinyal düzenleme, yüksek frekanslı gürültüleri bastırmak için alçak geçiren filtre, 50 Hz şebeke gürültüsünü bastırmak için çentik filtre, çoğullayıcı ve analog sayısal dönüştürücüdür. ADC tarafından sayısallaştırılan veri mikro kontrolör tarafından okunarak haberleşme protokolü sayesinde TKB'ye ulaştırılır.



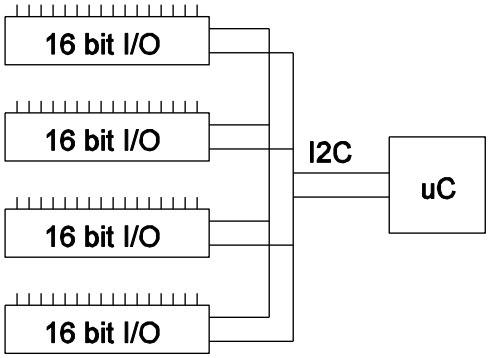
Şekil 4.2 Örnek VTK: 4 kanallı düşük gerilim farkı örnekleyebilen uç birim

Özelleştirilmiş ikinci örnek ise şekil 4.3'te verilmiştir. Bu örnekte sayısal veri toplamak amaçlanmıştır ve bu nedenle toplanan veri sayısını arttırmak için iskele çoğullayıcılar kullanılmıştır. İskele çoğullayıcılar sayesinde 64 bitlik veri tek bir kart ile toplanabilecektir. İskele çoğullayıcılar I2C protokolü ile mikro kontrolöre bağlanmıştır. Herhangi bir bitin değeri mikro kontrolör tarafından okunarak TKB'ye iletilebilmektedir. Bu kart ile manyetik alan algılayıcıları, kızıl ötesi algılayıcılar veya limit düğmeleri gibi sayısal çıkış üretebilen 64 adet algılayıcıdan veri toplanabilmektedir.

4.3 Yazılım

VTK yazılımı, haberleşme protokolünün uygulama katmanıdır. Protokol tarafından bağlantı kurulduktan sonra gelen komutları çözümleyerek bunlara cevap gönderecek kısımdır. Bu kısım VTK'ya özel yazılmaktadır. VTK'da bulunacak kanal sayısına, kanalların tiplerine (analog - sayısal) ayrıca verilerin uzunluklarına (8 bit, 16 bit)

göre değişmektedir. Örneğin bir VTK'da bir komuta cevap mikrokontrolörün sayısal iskelesi okunarak anında da gönderilebilir veya başka bir VTK'da SPI protokolü ile bir ADC'nin ilgili kanalından 16 bitlik veri okunduktan sonra bu veri gönderilebilir. Her türlü veriye ve yapılacak işleme göre bu kısım özelleştirilmektedir. Özelleştirilmiş GLiVET sisteminin anlatıldığı bölümde VTK uygulama katmanı yazılımına da örnek verilmiştir.



Şekil 4.3 Örnek veri toplama kartı: 64 bit sayısal veri toplayabilen uç birim

5. TKB - VTK HABERLEŞME PROTOKOLÜ

Tek kart bilgisayar ile veri toplama kartları arasındaki haberleşmeyi gerçeklemek için özgün bir haberleşme protokolü tasarlanmıştır. CANBUS, ProfiBUS vb.. haberleşme protokollerini kullanmaktansa özgün bir protokol tasarlamanın iki temel nedeni bulunmaktadır.

Birinci neden, hazır kullanım yerine zaman harcanarak tasarımın yapılması o tasarımın ileriki zamanlarda istenilen yönde geliştirilebilmesini sağlayacaktır. Tasarım sırasında edinilen bilgi birikimi, ileride standartlara uygun ve istenildiği gibi özelleştirilebilen bir haberleşme protokolünün tasarlanmasına olanak sağlayacaktır. Geliştirme için harcanan zamanın, geliştirme sonunda getireceği kazançtan daha önemsiz kalmasından dolayı, tasarımın yapılmasına karar verilmiştir ve özgün bir haberleşme protokolü gerçeklenmiştir.

İkinci neden ise, açık kaynak kod felsefesinden gelen bağımsızlığın ve özgürlüğün donanıma da yansıtılmak istenmesidir. Özgün protokol, uç birimlerde istenilen mikrokontrolörün kullanımına izin vermektedir. Donanım olarak çok küçük bir bloğun devreye eklenmesi ve yazılım olarak da verilen akış diyagramlarının birkaç satırlık program koduna çevrilerek işlemciye yüklenmesi ile protokol gerçeklenebilmektedir. İstenilen mikrokontrolörün (PIC, Intel, Motorola...) kullanılması, gerek tasarımı yapan kişinin bilgi birikimini en iyi şekilde kullanmasını gerekse maliyet açısından en uygun işlemcilerin seçilmesine olanak sağlamaktadır.

5.1 Katmanlar

Geliştirilen haberleşme protokolü katmanlı yapı dikkate alınarak tasarlanmıştır. Protokol, en alt üç katmanı -Fiziksel katman, veri katmanı, şebeke katmanı- ve uygulama katmanını kullanmaktadır.

5.1.1 Fiziksel Katman

Haberleşmede kullanılan protokol RS485 protokolüdür, bu nedenle fiziksel katman buna uygun tasarlanmıştır. Bitler, çift büklümlü kablo üzerinden farksal olarak gönderilmektedir. A ve B olarak tanımlanan çift büklümlü kablonun iki kablosunda

bitler, A>B durumunda 1, B>A durumunda 0 olarak algılanmaktadır. Çift büklümlü kablo üzerinden farksal iletim ile 1km'lik mesafeye veri gönderilebilmektedir.

Veri yolu topolojisi ile oluşturulan ağda, her uç birimde alıcı-vericiler bulunmaktadır. Usta - köle şeklinde veri yolunda haberleşildiğinden bir anda sadece bir birimin veri göndermesi sağlanmaktadır. Bu nedenle gönderen ve alan şeklinde iki veri yolu kullanmak yerine, bir veri yolu gönderilmiştir. Simplex haberleşme kullanılarak tek veri yolundan hem veri gönderilmekte hem de okunmaktadır. Akış kontrolü yazılım ile yapıldığından fiziksel olarak akış kontrolü protokole eklenmemiştir.

Veri yolunu oluşturan kablo 8 kablolu UTP (aynalanmamış çift büklümlü kablo) olarak seçilmiştir. Kabloda çift büklümlü iki kablo veri iletimi için geriye kalan altı kablo da güç iletimi için kullanılmıştır. Üç kablo besleme gerilimi, üç kablo ise toprak olarak kullanılmıştır. Üçer kablo kullanılmasının nedeni kabloların akım kapasitesini arttırmak içindir. Uzun mesafelere güç iletimi sağlanabilmesi için, kablo direnci nedeni ile kayıplar göz önüne alınarak besleme kablolarına 5V'luk gerilim uygulanmamıştır. Bunun yerine 12V'luk gerilim uygulanmış ve uç birimlerde 5V'luk regülatör kullanılmıştır. Bu sayede gerilim düşmesi olsa bile uç birimler regülatörler sayesinde bundan etkilenmeyeceklerdir.

5.1.2 Veri Katmanı

RS232 protokolünden dönüştürülerek RS485 sinyalleri elde edildiğinden, verilerin gönderilmesi RS232 protokolü standartlarında olmaktadır. Veriler 8 bitlik paketler halinde gönderilmektedir. Paketlerin başında başlangıç, sonunda bitiş bitleri bulunmaktadır. GLiVET'in kullandığı seri haberleşme konfigürasyonu 19.200 baud hızı, 8 bit veri, 1 dur bitidir. Parity biti kullanılmamaktadır (19200, 8, none, 1).

5.1.3 Şebeke Katmanı

Veri yolu topolojisi kullanıldığından dolayı tüm iletişim bütün uç birimler tarafından dinlenmektedir yani gönderilen veri bütün uç birimler tarafından alınmaktadır. Doğru uç birime ulaşım için adresleme ve adres çözümleme gerekmektedir.

Haberleşme protokolünün tasarımı usta - köle şeklinde yapılmıştır. Usta, tek kart bilgisayar veya bir PC'dir. Usta haberleşmeyi başlatır, veri alış verişini yapar ve bitirir. Uç birimlerin hepsi birer köledir ve veri yolu üzerinde tek bir adres taşımaktadırlar. Veri yolu üzerinde bu adres tektir, iki adet olması durumunda çakışma olacaktır.

Usta, haberleşme başlangıcı paketini gönderdikten sonra, ulaşmak istediği uç birimin adresini veri yoluna göndermektedir. Bu adres uç birimlerce çözümlenmekte ve adresi uyan bir uç birim onay gönderirken diğerleri haberleşmenin bitmesini beklemekte ve bu ana kadar sessiz kalmaktadırlar.

5.1.4 Uygulama Katmanı

Haberleşme başladıktan, ve adres çözümlemesi yapılarak bağlantı kurulduktan sonra usta tarafından komutlar gönderilmekte ve komutlara göre uç birimler cevaplar üretmektedirler. Uygulama katmanı hangi komutlara ne cevaplar üretileceğinin ve nasıl davranılacağının tanımlandığı katmandır.

5.2 VTK Yazılımı

Veri toplama kartları üzerinde geliştirilen yazılım, bu kartların tek kart bilgisayarla RS485 veri yolu üzerinden haberleşmesini yani adres çözümlemesini, isteklere cevap vermesini ve haberleşmeyi sonlandırmasını sağlamaktadır.

Yazılım, katmanlı yapıda şebeke ve uygulama katmanını gerçeklemektedir. Şebeke katmanında adres çözümlemesi ve haberleşmenin kontrolü sağlanmaktadır. Uygulama katmanında ise alınan veri işlenerek cevap geri gönderilmektedir.

Geliştirilen algoritma, haberleşmede kullanılan komut ve cevaplardan bağımsız çalışması amaçlanarak tasarlanmıştır. Gönderilen komutların ve verilerin içeriği ve uzunluğu değişebilmektedir. Bu şekilde uygulama katmanında yapılacak komut veya cevap değişiklikleri şebeke katmanını etkilemeyecektir ve burada değişiklik gerektirmeyecektir. Bu da işlevsellik olarak yenilikleri ve eklentileri kolayca kabul edebilecek bir yapı oluşturmaktadır.

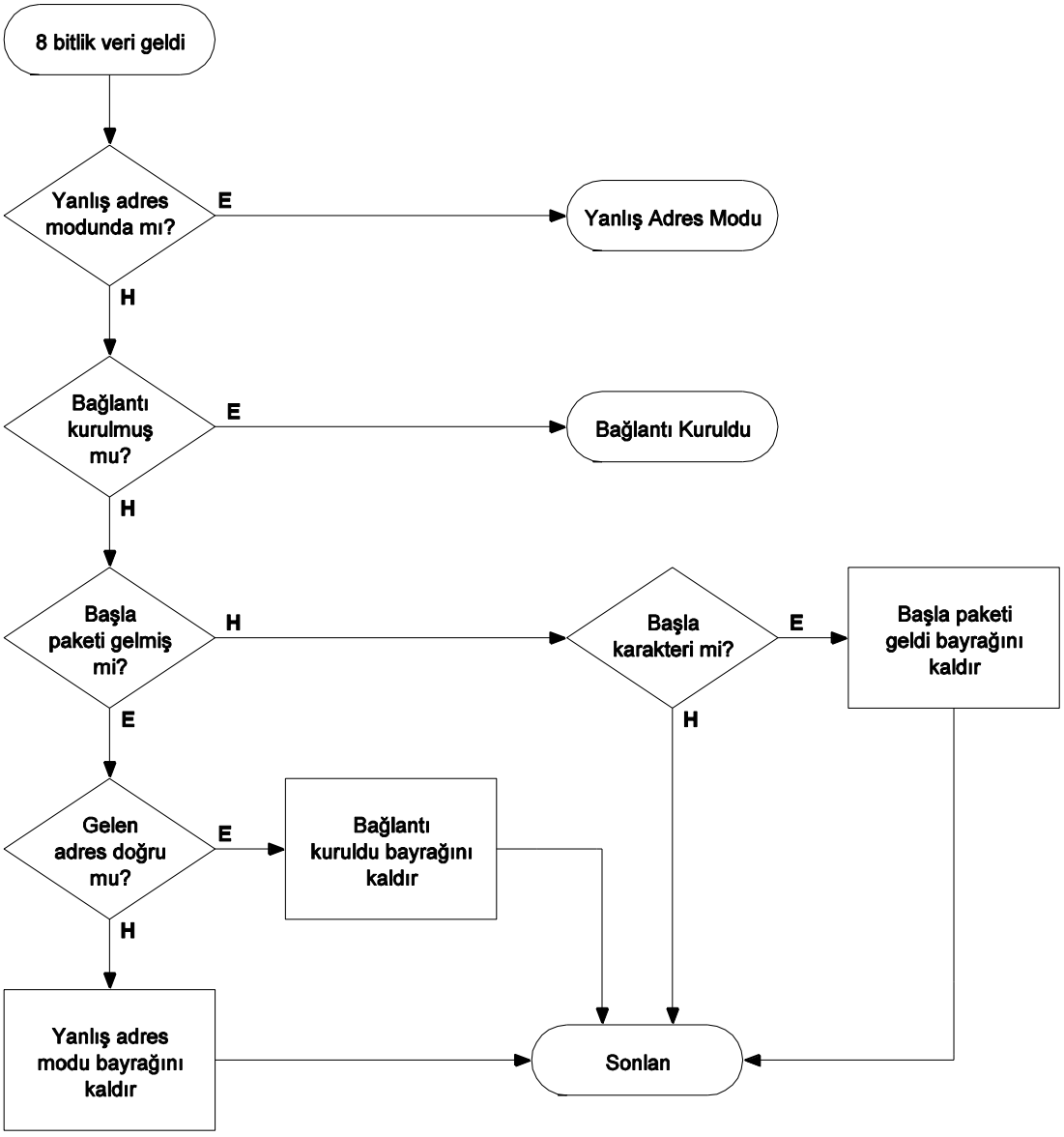
5.2.1 Haberleşme Başlangıcı

Haberleşme her zaman usta tarafından başlatılmakta ve bitirilmektedir. Usta istediği adresteki köle ile bağlantıyı kurar, komutu gönderir, varsa cevabı alır ve haberleşmeyi bitirir. Komut gönderme ve cevap alma bölümü istenirse usta tarafından tekrarlanabilmektedir. Bu sayede aynı adresteki kart ile yapılacak ardışıl haberleşmelerin her biri için tekrar bağlantı kurmak gerekmemektedir.

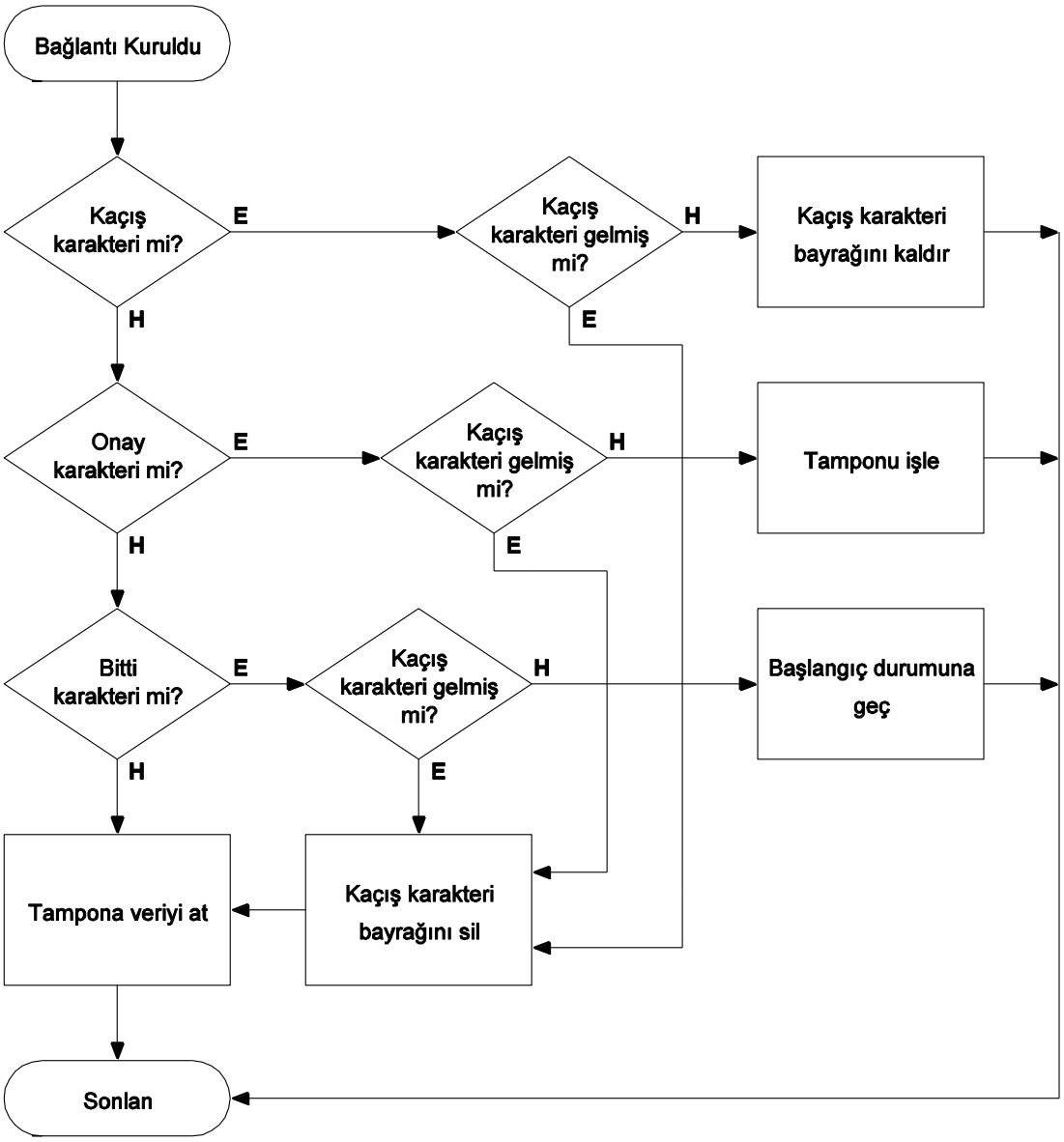
5.2.2 Bağlantı Kurulması

Haberleşme sırasında gönderilen her karakter şekil 5.1'deki akıştan geçerek işlenir. Programın bu kısmı haberleşme başlangıcından ve adres çözümlemesinden sorumludur. Haberleşme başlangıcında usta tarafından ilk önce başla karakteri veri yoluna gönderilmektedir. Bu karakteri alan bütün veri toplama kartları başla karakteri geldi bayrağını kaldırırlar. Usta tarafından gönderilen ikinci karakter adres bilgisini içermektedir. Bu karakterin gelmesiyle VTKlar iki ana moddan birinde geçiş yaparlar:

1. Yanlış adres modu
2. Bağlantı kuruldu modu



Şekil 5.1 Adres çözümleme ve bağlantı kurma akış diyagramı



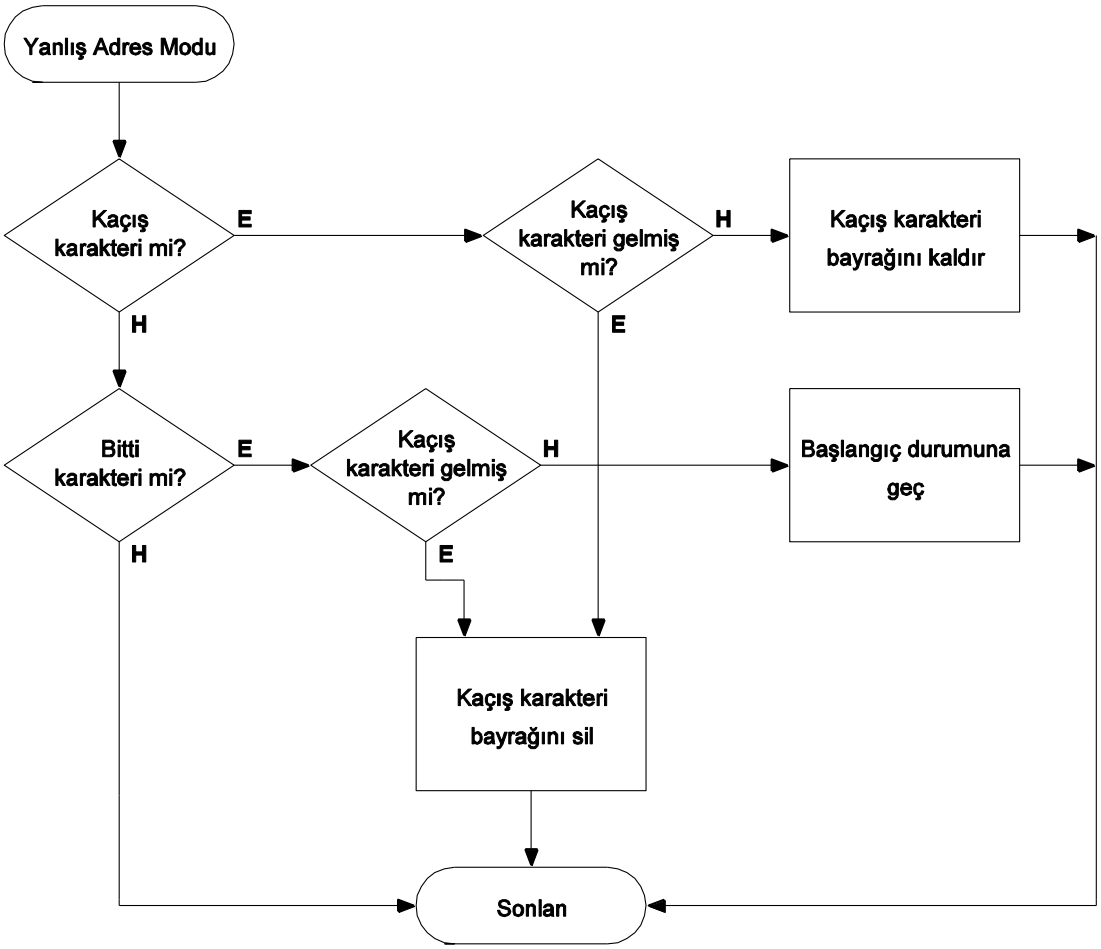
Şekil 5.2 Bağlantı kuruldu modu akış diyagramı

İkinci karakter geldiğinde adresi yanlış olan VTKlar yanlış adres moduna geçerler (Şekil 5.3) ve bu durumda başında kaçış karakteri olmayan bir dur karakteri gelene kadar kalırlar. Bu karakter geldiğinde haberleşme sona ermiş olur ve VTKlar başlangıç durumuna yani başla karakteri bekledikleri duruma geri dönerler.

Adresi doğru olan yani bağlantının kurulacağı VTK ise bağlantı kuruldu moduna geçer (Şekil 5.2). Bu moda geçtikten sonra usta ile veri haberleşmesi başlamış olur ve komutlara cevap verilmeye başlanır (Şekil 5.4).

5.2.3 Özel Karakterler

Haberleşmede kullanılan üç tane özel karakter vardır: Bitti, tamam, kaçış karakterleri.



Şekil 5.3 Yanlış adres modu akış diyagramı

Şekil 5.4 Haberleşme paketleri

- Bitti karakteri haberleşmeyi sonlandırmak için kullanılmaktadır. Bir önceki karakter kaçış karakteri olmayan bir bitti karakteri haberleşmeyi sonlandırır ve bütün VTKları başlangıç durumuna geri döndürür.
- Tamam karakteri gönderilen komutun veya verinin bittiğini ve işlenmesi gerektiğini belirtir. Tamam karakteri geldiğinde tampona atılmış olan komut işlenir ve uygun cevap geri gönderilir. Tamam karakterinin olmasının amacı değişken bit uzunluğunda komutların kullanılabilmesidir.
- Kaçış karakteri veri içinde bitti veya tamam karakterleri geçiyorsa, bunların özel karakterler olmadığını belirtmek için bu karakterlerden önce gönderilen karakterdir. Şekil 5.5'te kullanımına örnek verilmektedir. Veri veya komut içinde kaçış karakteri bulunuyorsa bunu belirtmek için ard arda iki kaçış karakteri gönderilmektedir.

Şekil 5.5 Kaçış karakteri kullanımı

5.3 TKB Haberleşme Yazılımı

TKB haberleşme yazılımı, haberleşme protokolünde sahibin yani tek kart bilgisayarın fiziksel katmanı sürebilmesi için geliştirilen fonksiyonları içermektedir. Linux üzerinde C dilinde yazılmış olan bu fonksiyonlar, seri haberleşme iskelesinin açılması ve konfigürasyonunu, veri gönderilmesini ve okunmasını sağlayan temel fonksiyonlardır. Fonksiyonlar glivet.h C kütüphane dosyasında bulunmaktadır ve

bütün diğer uygulama katmanını gerçekleyen TKB yazılımlarınca eklenmesi gerekmektedir.

5.3.1 Haberleşme İskelesinin Açılması

Linux işletim sisteminde /dev klasörü altında ttyS0 dosyası, 1. seri haberleşme iskelesinin sürücüsüdür (COM1). C dilinde bu dosya open komutu ile açılarak seri haberleşme iskelesine erişim sağlanmaktadır. Yazma-okuma modunda dosya açıldığından dosya izinlerinin çalıştıran kullanıcıya uygun olması gerekmektedir. TKB'de kullanıcı root olduğundan dolayı bunun önemi olmamaktadır [19], [20].

```
//---------------------------------------------------------------------
// open_port() - Seri haberleşme iskelesini açar.
// Dosya işaretçisini geri döndürür, hata durumunda -1 döndürür.
//---------------------------------------------------------------------
int open_port(void)
{
     int fd;
     fd = open("/dev/ttyS0", O_RDWR | O_NOCTTY | O_NDELAY);
     if (fd == -1)
     {
          perror("open_port: Unable to open /dev/ttyS0 - ");
     }
     else
          fcntl(fd, F_SETFL, 0);
     return (fd);
}
```

5.3.2 Haberleşme İskelesinin Konfigürasyonu

Seri haberleşme iskelesi açıldıktan sonra veri alıp vermek için konfigüre edilmelidir. Bu konfigürasyon iskelenin veri iletim hızını, veri bitlerini ve kontrol bitlerini, akış kontrolünün nasıl sağlanacağını belirtir. Linux işletim sisteminde bu ayarları yapabilmek için termios.h C kütüphane dosyasından yararlanılmaktadır [19], [22].

tcgetattr() fonksiyonu ile konfigürasyonu bulunduran yapı belleğe alındıktan sonra gerekli değişiklikler üzerinde yapıldıktan sonra tcsetattr() fonksiyonu ile geri yüklenir [19], [20]. cfsetispeed() ve cfsetosped() fonksiyonları ile veri gönderme ve alma hızları 19200 baud hızına ayarlanmaktadır. termios yapısında olan options değişkenindeki bayraklar ayarlanarak, 8 bit veri, parity biti yok, 1 dur biti ve donanım akış kontrolü yok şeklinde iskele konfigüre edilmektedir. Diğer önemli bir ayar ise okumadaki bekleme zamanıdır. Bu c_cc[VTIME] değişkeninin 1'e eşitlenmesi ile 10ms'ye ayarlanmış olur. Tüm bayraklar ayarlandıktan sonra termios yapısındaki options değişkeni sürücüye geri yüklenir [19], [20].

```
//---------------------------------------------------------------------
// configure_port() - Seri haberleşme iskelesini konfigüre eder.
// Baud: 19200
// Bit: 8
// Parity: None
// Stop Bits: 1
//---------------------------------------------------------------------
void configure_port(fd)
{
     struct termios options;

     tcgetattr(fd, &options);

     cfsetispeed(&options, B19200);
     cfsetospeed(&options, B19200);

     options.c_cflag          &= ~PARENB;
     options.c_cflag          &= ~CSTOPB;
     options.c_cflag          &= ~CSIZE;
     options.c_cflag          |= CS8;

     //Donanım akış kontrolünü iptal et
     options.c_cflag          &= ~CRTSCTS;

     options.c_cflag          |= (CLOCAL | CREAD);
     options.c_lflag          &= ~(ICANON | ECHO | ECHOE | ISIG);
     options.c_oflag          &= ~OPOST;
     options.c_cc[VMIN] = 0;
     options.c_cc[VTIME]      = 1;

     tcsetattr(fd, TCSANOW, &options);
}
```

5.3.3 Veri Akış Yönü Kontrolü

RS232 - RS485 dönüştürücü devresinde RTS ucu, veri akış yönünü kontrol eden uç olarak kullanılmıştır. Yani RS485 entegresinin alıcı veya verici olarak çalışmasını belirleyen uçtur. Bu ucun 1 yapılması devrenin verici, 0 yapılması alıcı olarak çalışmasını sağlamaktadır.

set_RTS() fonksiyonu, RTS ucunu 1 yaparak veri iletimine olanak sağlamaktadır. ioctl() fonksiyonunu kullanarak seri iskelenin durum (status) bitlerinden ilgili olanları ayarlanması ile bu gerçeklenmektedir. TIOCMGET komutu ile okunan durum bitleri değişiklikten sonra TIOCMSET komutu ile geri yazılmaktadır [19], [21].

clear_RTS() komutu, RTS ucunu 0 yaparak veri alımına olanak sağlamaktadır. Yine ioctl() fonksiyonu ile ilgili bitler değiştirilerek RTS ucu 0 yapılmaktadır [19], [21].

```
//---------------------------------------------------------------------
// set_RTS() - RTS'yi set eder.
//---------------------------------------------------------------------
void set_RTS(fd)
{
```

```
     int status;
     ioctl(fd, TIOCMGET, &status);
     status &= ~TIOCM_RTS;
     ioctl(fd, TIOCMSET, &status);
}

//-------------------------------------------------------------------
// clear_RTS() - RTS'yi clear eder.
//-------------------------------------------------------------------
void clear_RTS(fd)
{
     int status;
     ioctl(fd, TIOCMGET, &status);
     status |= TIOCM_RTS;
     ioctl(fd, TIOCMSET, &status);
}
```

5.3.4 Karakter Gönderme

Bir karakterin (8 bitlik bir verinin) veri yoluna yazılması için belirlenmiş bir yolun izlenmesi gerekmektedir ve bu yol send_char() fonksiyonu ile gerçeklenmiştir:

1. RS485 sürücüsü veri gönderme durumuna getirilir ve verici hattı sürmeye başlar.
2. 8 bitlik veri seri iskeleye yazılır ve veri gönderilmeye başlanır.
3. Seri iskelenin belirlenen hızda veriyi göndermesi beklenir [24].
4. Gönderme işlemi bittikten sonra RS485 sürücüsü veri alma durumuna getirilir.

Bekleme süresi 9600 baud hızında 1ms, 19200 baud hızında 0.5ms'dir. Beklemeler boş bir döngü oluşturarak gerçeklenmektedir ve bu nedenle kesin bekleme zamanları elde edilememektedir. 300MHz'lik bir işlemci için 1ms ve 0.5ms civarındaki bekleme döngü değerleri kaynak kod içinde verilmiştir. Bu değerler farklı bir işlemci kullanıldığı takdirde değiştirilmelidir aksi takdirde sistem çalışmayacaktır.

```
//-------------------------------------------------------------------
// send_char() - Seri porta bir karakter gönderir.
//-------------------------------------------------------------------
int send_char(int fd, char *c)
{
     int n,i;
     set_RTS(fd);           // Max485 Transmit moduna geçirilir.
     n = write(fd, c, 1);   // Karakter seri porta yazılır.

     // Bekleme - UART'ın bitleri göndermesi beklenir
     // ~ 1ms -> 9600baud
     //for (i=0;i<23000;i++);

     // Bekleme - UART'ın bitleri göndermesi beklenir
     // ~ 0.5ms -> 19200baud
     for (i=0;i<11500;i++);
```

```
    clear_RTS(fd);          // Max485 Receive moduna geçirilir.
    return n;
}
```

5.3.5 Karakter Okuma

Veri yolundan veri okumak istendiğinde get_char fonksiyonu çağırılmaktadır. RS485 hat sürücüsü zaten veri okuma modunda olduğundan dolayı RTS ucu ile ilgili bir işleme gerek duyulmamaktadır. Hatta gönderilen bir veri doğrudan seri haberleşme iskelesi kütüğüne yazılacaktır ve read() fonksiyonu ile bu veri okunabilecektir. Veri okunamaması, yani 10ms'lik zaman aşımı durumunda fonksiyon geriye 0 döndürmektedir.

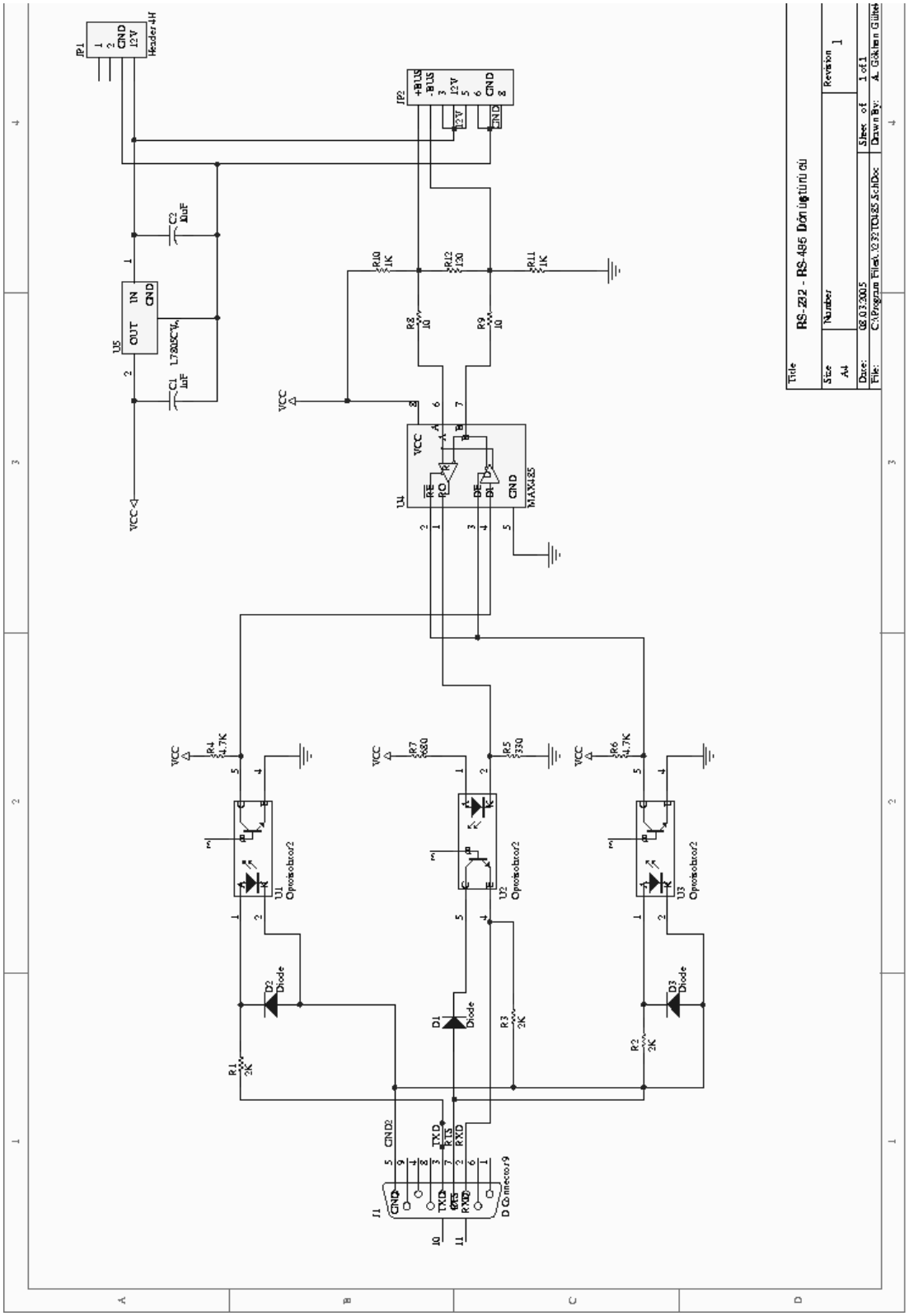
```
//---------------------------------------------------------------
// get_char() - Seri porttan bir karakter okur.
//---------------------------------------------------------------
char get_char(fd)
{
    int n;
    char s[2] = "";
    n = read(fd, &s, 1);
    return s[0];
}
```

5.4 Donanım

GLiVET'te, TKB ve VTK'lar arasındaki haberleşme protokolünün donanım tarafı, RS485 protokolü fiziksel katmanını gerçekleyebilecek şekilde tasarlanmıştır. VTK'lar tarafından TTL düzeyden hat sürme gerilimlerine, TKB tarafında ise RS232 protokolü gerilimlerinden RS485 düzeyine dönüşümler yapılmaktadır. Geliştirilen ve gerçeklenen tasarım RS485 protokolü fiziksel değerleri olsa da, tasarımın esnekliği sayesinde buradaki küçük bir kısım donanımın ve sürücülerin değiştirilmesiyle istenilen fiziksel iletim metodu kullanılabilecektir. Tasarımın, örneğin RS485 sürücülerinin kaldırılıp yerlerine RF modüller takılması ile kablosuz haberleşme gerçekleştirme yeteneği bulunmaktadır.

5.4.1 Veri Toplama Kartları Donanımı

Uç birimler bölümünde anlatılan minimum donanım kısmı daha önce de belirtildiği gibi haberleşme protokolünün gerçeklenmesi için gereken donanımdır. Haberleşme protokolünü gerçeklemek için bundan başka bir donanıma gerek duyulmamaktadır.



Şekil 5.6 RS232 - RS485 dönüştürücü devresi

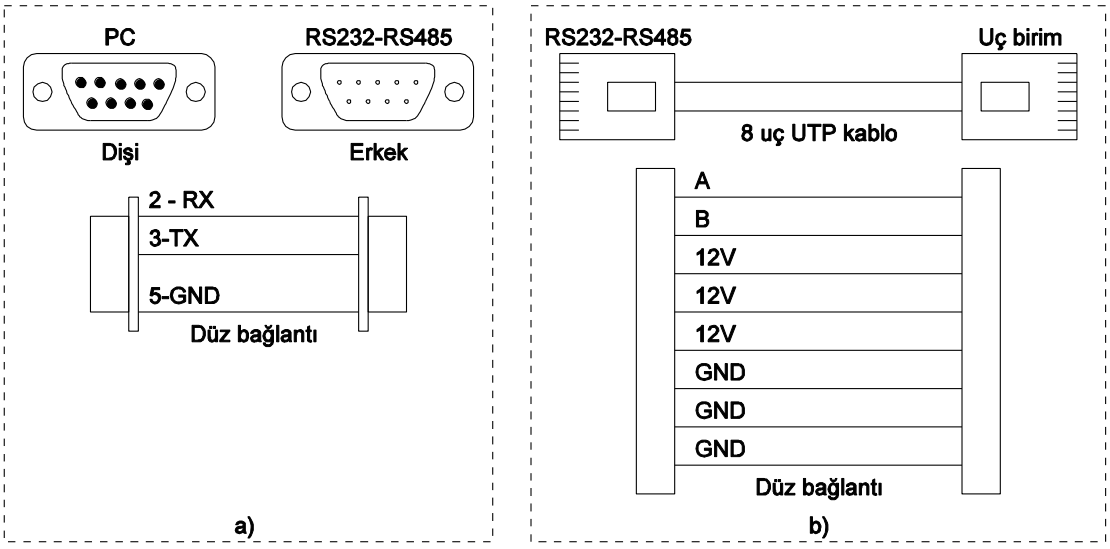
5.4.2 RS232 - RS485 Dönüştürücü

RS232 - RS485 dönüştürücü kartı (Şekil 5.6), bilgisayarın seri haberleşme iskelesindeki RS232 sinyallerini RS485 sinyallerine dönüştürmek için tasarlanmıştır [23], [24], [25]. Veri yolu kablosunda oluşabilecek yüksek gerilim yüklemelerinden TKB'yi korumak için optik izolasyon katı da kart tasarımına eklenmiştir. RS232 iskelesindeki 3 sinyal, alıcı uç (RX), verici uç (TX) ve transfere hazır ucu (RTS) üç adet optik kuplajlı entegreden geçirildikten sonra hat sürücü entegreye ulaşmaktadırlar. Optik izolasyonda dikkat edilmesi gereken en önemli nokta, transistor tarafında sinyali evirmeden çıkış almaktır.

Kart ayrıca veri yolundaki VTK'ların besleme gerilimini de sağlamaktadır ve bunun için dışarıdan beslenmektedir. 12V'luk besleme gerilimi 8 uçlu UTP kablonun üç kablosuna, toprak da diğer üç kabloya uygulanmaktadır. Besleme geriliminin üçer kablo ile aktarılmasının nedeni kablonun akım kapasitesini arttırmaktır.

5.4.3 Kablolar

RS232 - RS485 dönüştürücü ile RS232 iskelesi bulunan TKB veya bir PC arasında kullanılan kablo standart düz bağlantılı bir RS232 kablodur. Bu kablonun PC tarafı dişi, dönüştürücü tarafı erkek kablo kafası bulundurmaktadır. Şekil 5.7 (a)'da görüldüğü gibi bu kablonun sadece üç ucu kullanılmaktadır, diğer uçlarında bağlantı bulunmamaktadır. TKB'lerin kendilerine has bağlantı kabloları bulunduğundan satın alındığında içinden çıkan RS232 kablosu da RS232-RS485 dönüştürücü girişine uyacaktır ve kullanılabilecektir.

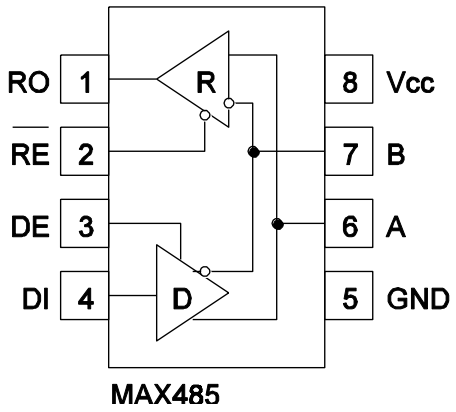


Şekil 5.7 GLiVET sisteminde kullanılan kablolar

RS485 veri yolu standart bir 8 uçlu UTP (aynalanmamış çift büklümlü) kablo ile yapılmıştır. Şekil 5.7 (b)'de de görüldüğü gibi her iki ucunda da RJ45 kablo kafası kullanılmıştır ve bunlar arasında düz bağlantı yapılmıştır. Yani iki kafada da renk sırası aynıdır. Kablolama sırasında dikkat edilmesi gereken bir nokta A ve B uçlarının çift büklümlü kablo çiftine bağlanmasıdır. Bu iletişimdeki gürültüyü azaltmak ve daha uzak mesafelere iletim yapabilmek için gerekmektedir.

5.4.4 RS485 Hat Sürücü Entegresi

RS485 veri yolunu sürmek için MAXIM firmasının MAX485 entegresi kullanılmıştır. Bu entegrenin uç diyagramı şekil 5.8'de görülebilmektedir [23].

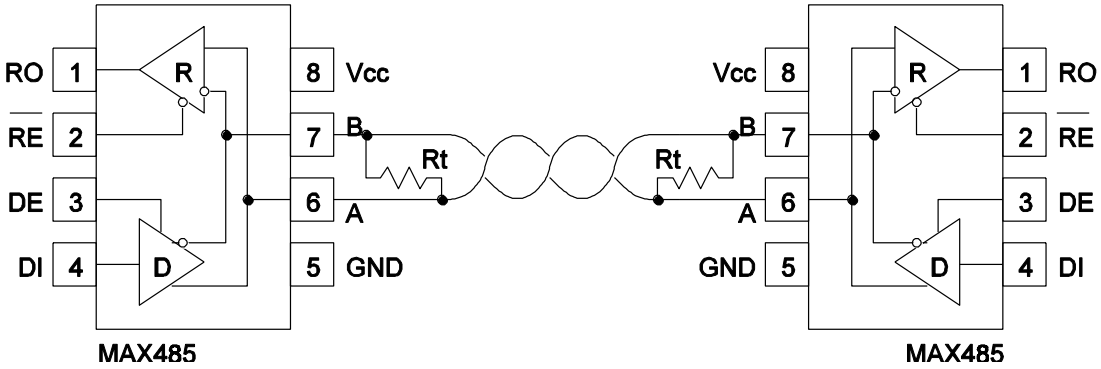


Şekil 5.8 MAX485 entegresi uç diyagramı

- Birinci uç RO (Receive Out), verinin TTL seviyede alındığı uçtur. Mikro kontrolörde ve seri haberleşme iskelesinde Rx ucuna bağlanmaktadır.
- İkinci uç RE (Receive Enable), alıcının aktif edildiği uçtur. Sıfırken aktiftir ve DE ucuna bağlıdır. Mikro kontrolör tarafında herhangi bir sayısal çıkış ucu buraya bağlanabilmektedir. Seri haberleşme iskelesinde ise RTS uç bu uca bağlanmaktadır.
- Üçüncü uç DE (Drive Enable), vericinin aktif edildiği uçtur. RE ucu ile kısa devredir.
- Dördüncü uç DI (Data In), TTL seviyedeki gönderilen verinin sürücüye girdiği uçtur. Mikro kontrolörde ve seri haberleşme iskelesinde Tx ucuna bağlanmaktadır.
- Beşinci uç GND, topraktır.

- Altıncı uç sürücü çıkışı A hattıdır. Çift büklümlü kablo ile hattaki diğer sürücülerin A hatlarına bağlıdır.
- Yedinci uç sürücü çıkışı B hattıdır. Çift büklümlü kablo ile hattaki diğer sürücülerin B hatlarına bağlıdır.
- Sekizinci uç VCC 5V besleme gerilimidir.

Sürücü entegrelerin birbirlerine bağlantı şeması şekil 5.9'da verilmiştir. Çift büklümlü kablo ile A ve B hatları her sürücünün A ve B hatlarına bağlanmaktadır. En uçtaki iki sürücünün A ve B hatları arasına hattaki yansımaları engellemesi için sonlandırma dirençleri bağlanması gerekmektedir.



Şekil 5.9 MAX485 bağlantı şekli

6. ÖZELLEŞTİRİLMİŞ GLİVET SİSTEMİ

Sunum için özelleştirilmiş bu sistem, uzun süreli dağıtık veri toplamayı amaçlayan bir sistemdir. Deprem tahmini veya meteoroloji verileri gibi coğrafi olarak dağıtılmış yani birden çok istasyondan oluşan ve uzun bir süre sürekli çalışacak ve yavaş veri toplayan bir sistem gerçeklenmiştir. Minimum örnekleme periyodu 1 saniyedir, sınırsız istasyon bağlanabilir ve her istasyon dört analog, bir 8-bit sayısal veri toplayabilmektedir.

6.1 İstasyon Yazılımı

Her istasyondan istenilen sayıda ve periyotlarda istenilen kanallardan toplanan veriler istenilen periyot ile sunucuya FTP yoluyla gönderilmektedir. Sistem yöneticisinin ayarlayabileceği tüm bu parametreler istasyonun içindeki glivet_config dosyasında bulunmaktadır. Sistem yöneticisi SSH (secure shell) programı ile istasyona bağlanarak sistemi konfigüre etmekte ve veri toplamayı başlatmaktadır. İstasyonda bulunan yardımcı programlar ile adresleri sınar ve değiştirebilir, veri alımını kontrol edebilir daha sonra da konfigürasyon dosyasını değiştirerek veri toplamayı başlatabilmektedir.

GLiVET konfigürasyon dosyası aşağıdaki ayarlamaları içermektedir:

- GLiVET_ID: İstasyonun sistemde tanımlanma numarası. Her istasyonun kendine özel bir tanımlama numarası vardır.
- SEND_IP: Verilerin gönderileceği internet adresi.
- FTP_PATH: Verilerin kopyalanacağı yol.
- TOT_SMP: Toplam alınacak örnek sayısı. Aşağıda sıralanan kanallardan her biri için alınacak toplam örnek sayısını belirtir. Örneğin üç kanal örneklenecekse ve buraya toplam örnek sayısı on girilmişse, kanal başına on örnek alınır, yani toplamda otuz örnek toplanır.

- FTP_INT: Saniye hassasiyetinde toplanan örneklerin sunucuya iletilme periyodu. Örnekleme periyodundan küçük olamaz, ama aynı olabilir (tamponlama yapılmaz).
- SMP_INT: Saniye hassasiyetinde örnekleme periyodu. Tüm kanallar aynı anda burada yazan periyodda örneklenmektedir, bu nedenle kanala özel örnekleme periyodu girilemez.
- KANAL: Veri yolu üzerindeki hangi uç birimin hangi kanalının örnekleneceğini tanımlar. KANAL:uç birim adres:kanal numarası şeklinde girilmektedir. Her yeni eklenecek kanal için yeni bir satır eklenmelidir.

```
#------------------------------------------------------------
#    GLiVET V1.0 Konfigurasyon Dosyasi
#------------------------------------------------------------

# Glivet ana dugum tanimlama numarasi (tum sistemde tektir)
GLiVET_ID:G0036

# Verilerin gonderilecegi internet adresi ve ftp yolu
SEND_IP:192.168.0.1
FTP_PATH:/glivet_data/

# Toplam alinacak ornek sayisi (kanal sayisindan bagimsizdir)
TOT_SMP:19

# Tamponlanan dosyanin gonderilme zaman araligi.
# Saniye cinsinden verilir.
# Or: ftp_int:3 -> 3 saniyede bir tamponlanan dosya
# ftp ile sunucuya aktarilir.
FTP_INT:7

# Ornekleme periyodu.
# Saniye cinsinden kanallarin orneklenip dosyaya yazilma periyodu.
SMP_INT:2

# Orneklenecek kanallar.
# Her yeni orneklenecek yeni kanal icin yeni bir satir eklenir.
# Kanal:adres:kanal
# or: kanal:1:2 -> 1.ucbirim adresinin 2. kanalini ornekle.
# or: kanal:14:4 -> 14.ucbirim adresinin 4. kanalini ornekle.
KANAL:1:1
KANAL:1:2
KANAL:1:3
KANAL:1:4
KANAL:2:1
KANAL:11:3

#------------------------------------------------------------
```

Özelleştirilmiş bu sistem için yazılan C programı "sample2file"'dır. Bu program, glivet_config dosyasından uç birim adreslerini ve kanal numaralarını okuyarak,

readADC alt programı ile verileri toplamaktadır. Topladığı verileri aralarına virgül koyarak standart çıkış olan monitöre göndermektedir.

```
int main()
{
     int fd, count, kanal, adres;
     unsigned char n;
     char s[100], buf[3];
     FILE *file;


     if ((file = fopen("glivet_config","r")) == NULL) // Geçici veri
depolama dosyası açılıyor.
          {
          printf("ID Dosyasi acilamadi...");
          exit();
          }

     fd = open_port();

     configure_port(fd);

     clear_RTS(fd);  // idle state.

     count=0;
     while (!feof(file))
     {
          fscanf(file, "%s", &s);
          if (strncmp(s,"KANAL",strlen("KANAL")) == 0)
          {
               buf[0] = s[strlen("KANAL")+1];
               if (s[strlen("KANAL")+2] != ':')
               {
                   buf[1] = s[strlen("KANAL")+2];
                   buf[2] = '\0';
                   adres = str2int(buf);
                   buf[0] = s[strlen("KANAL")+4];
                   buf[1] = '\0';
                   kanal = str2int(buf);
               }
               else
               {
                   buf[1] = '\0';
                   adres = str2int(buf);
                   buf[0] = s[strlen("KANAL")+3];
                   buf[1] = '\0';
                   kanal = str2int(buf);
               }

               n = readADC(adres,kanal,fd);

               if (count==0)
                   printf("%d",n);
               else
                   printf(",%d",n);
               count++;
          }
     }
     printf("\n");
```

```
    fclose(file);
    close(fd);
}
```

Veri toplamayı gerçekleştiren, bir kabuk programı olan sample dosyasıdır [7]. Bu program, glivet_config dosyasından parametreleri okuyarak veri toplamayı gerçekleştirmektedir.

```
#!/bin/sh
# Sample data

GLiVET_ID=`grep GLiVET_ID glivet_config | cut -d: -f2`
SEND_IP=`grep SEND_IP glivet_config | cut -d: -f2`
FTP_PATH=`grep FTP_PATH glivet_config | cut -d: -f2`
FTP_INT=`grep FTP_INT glivet_config | cut -d: -f2`
SMP_INT=`grep SMP_INT glivet_config | cut -d: -f2`
TOT_SMP=`grep TOT_SMP glivet_config | cut -d: -f2`

if [ ${SMP_INT} -gt ${FTP_INT} ]
then
  echo Ornekleme periodu dosya gonderme periodundan kucuk olmalidir!
  exit
fi

let i=1
let send=0
let send_id=1
while [ $i -le ${TOT_SMP} ]
do
  ./sample2file >> ${GLiVET_ID}.dat
  j=1
  while [ $j -le ${SMP_INT} ]
  do
    sleep 1
    let send=(${i}*${SMP_INT}+${j})%${FTP_INT}
    if [ ${send} -eq 0 ]
    then
      echo `date` ${GLiVET_ID}_${send_id}.dat " gonderildi. ->
ftp://"${SEND_IP}${FTP_PATH} >> glivet.log
      wput -qR ${GLiVET_ID}.dat
ftp://${SEND_IP}${FTP_PATH}${GLiVET_ID}_${send_id}_`date
+%Y%m%d_%T`.dat
    rm -f ${GLiVET_ID}.dat
    let send_id++
    fi
    let j++
  done
  let i++
done
if [ -f ${GLiVET_ID}.dat ]
then
  echo `date` ${GLiVET_ID}_${send_id}.dat " gonderildi. ->
ftp://"${SEND_IP}${FTP_PATH} >> glivet.log
  wput -qR ${GLiVET_ID}.dat
ftp://${SEND_IP}${FTP_PATH}${GLiVET_ID}_${send_id}_`date
+%Y%m%d_%T`.dat
fi
```

Program sırasıyla şu işlemleri yapmaktadır:

1. Tüm parametreler glivet_config dosyasından okunur.
2. sample2file programı çağırılarak veri toplama gerçekleştirilir. Program çıktısı tampon dosyaya yönlendirilir.
3. Bir saniye hassasiyet ile örnekleme periyodu kadar bekleme yapılır.
4. Bekleme süresince dosya gönderme periyodunun aşılıp aşılmadığı kontrol edilir. Eğer aşıldıysa "wput" ftp programı yardımı ile konfigürasyon dosyasındaki adrese tampon dosya yeniden adlandırılarak gönderilir.
5. Gönderme işleminden sonra glivet.log dosyasına yapılan işlem işlenir.
6. Bu döngü toplam veri toplama sayısına ulaşılıncaya kadar devam eder.
7. Toplam veri toplama sayısı ulaşıldığında hala tamponda gönderilmemiş veri varsa bu veri gönderilir ve log dosyasına işlenir.

Sunucuya gönderilen dosyalar 'istasyon tanımlama numarası'_'gönderme sıra no'_'Yıl"Ay"Gün'_'Saat':'Dakika'.dat şeklinde isimlendirilmektedir. Örneğin: 'G0036_6_20050210_19:45.dat'. Dosyaların her satırı ise konfigürasyon dosyasındaki sıra ile kanalların virgül ile ayrılmış değerlerinden oluşmaktadır. Yukarıdaki konfigürasyon dosyası için örneğin bir dosyanın içi aşağıdaki gibi olacaktır:

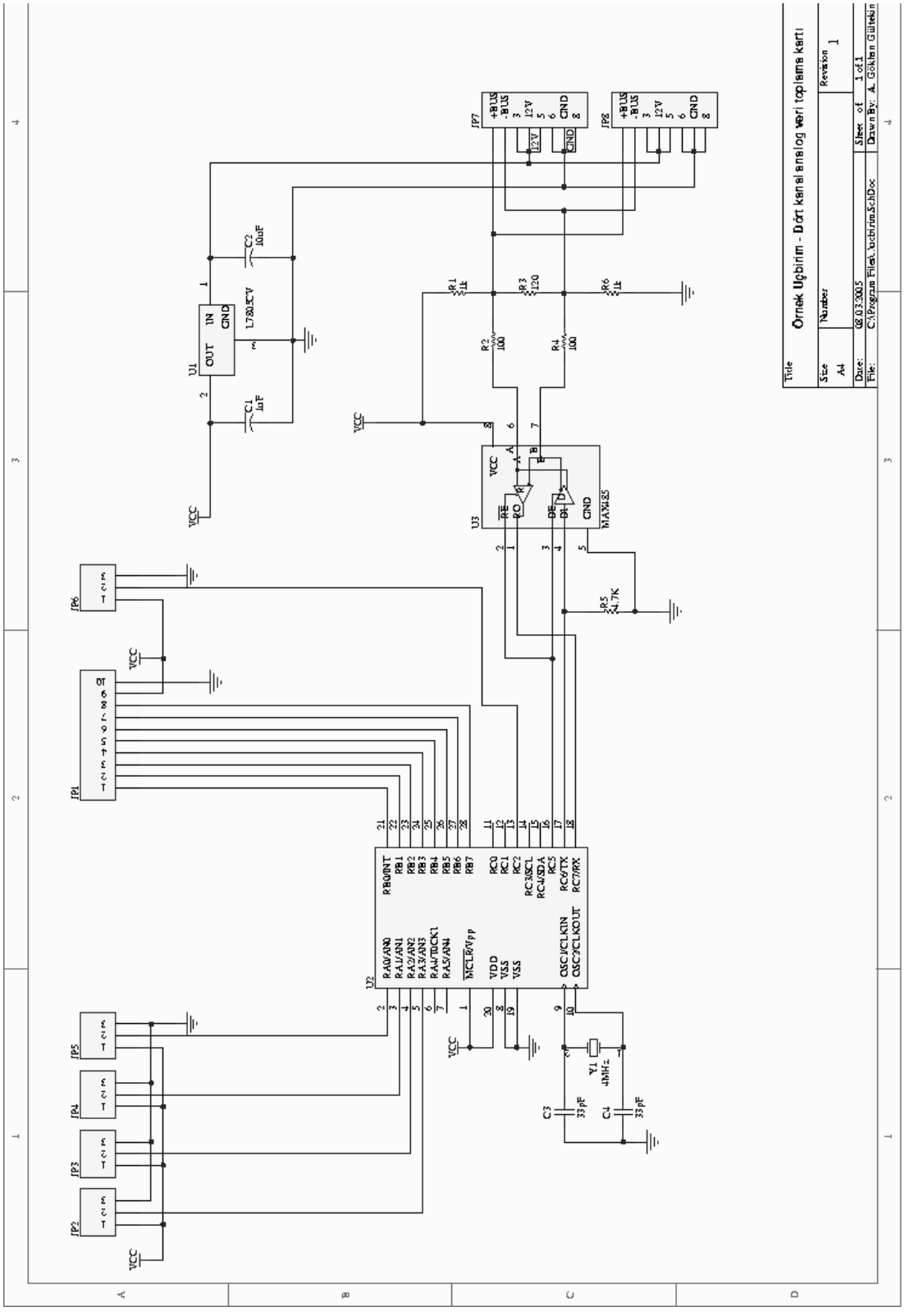
```
23,42,12,53,45,134
25,48,6,100,45,132
28,56,2,189,44,130
33,60,0,226,45,129
```

Sıra numarasına göre sıralanmış dosyalar sunucuda tek tek sıra ile okunabilir veya cat komutu ile birleştirilip tüm verileri içeren yeni bir dosya yaratılabilmektedir. Bu aşamadan sonra toplanan veri sunucuda veri depolama, analiz ve raporlama programlarınca işlenmektedir.

6.2 Gerçeklenen Veri Toplama Kartı Donanımı

Tez sunumu için hazırlanan özelleştirilmiş GLiVET sisteminde demostrasyon amaçlı genel bir veri toplama kartı tasarlanmıştır. Dört kanaldan analog veri toplamak amacıyla bünyesinde ADC modülünü barındıran bir mikro kontrolör, Microchip firmasının PIC16F873 [27] modeli seçilmiştir. Analog kanalların uçlarına

potansiyometreler bağlanarak buradaki analog seviyelerin el ile değiştirilebilmesi sağlanmıştır. İki adet eşdeğer devre gerçeklenmiştir ve veriyoluna bağlanmıştır. Tasarlanan kartın devre çizimi şekil 6.1'de verilmiştir.



Şekil 6.1 Gerçeklenen veri toplama kartının devre çizimi

6.3 Gerçeklenen Veri Toplama Kartı Yazılımı

Veri toplama kartlarında, haberleşme protokolü yazılımı protokolün standart program akışına göre yazılmakta, uygulama katmanı yazılımı ise karta özel yazılmaktadır.

Protokol yazılımı PIC serisi mikro kontrolörlerin kendi dilinde akış diyagramlarına göre yazılmıştır. Mikro kontrolörün USART modülünü [28] seri haberleşme için kullanan protokol yazılımı toplam yedi fonksiyon ile, genel fonksiyonlar hariç, kolaylıkla gerçeklenmiştir.

Uygulama katmanı yazılımı, TKB tarafından gönderilen komutları yorumlayarak gerekli işlemleri yapıp cevap gönderen fonksiyondur. Gerçeklenen veri toplama kartı yazılımı, dört kanal analog veriyi okumayı ve adres değişikliğini içerek toplam beş komut içermektedir.

Gerçeklenen VTK'da komutlar 8 bitlik tasarlanmıştır. Son dört bit komutu, ilk dört bit ise komut gerektiriyorsa veriyi içermektedir. Komut listesi tablo 6.1'de verilmiştir.

Tablo 6.1 Komut tablosu

Komut	İşlevi
0001xxxx	1. ADC kanalını oku
0010xxxx	2. ADC kanalını oku
0011xxxx	3. ADC kanalını oku
0100xxxx	4. ADC kanalını oku
0110aaaa	Adresi aaaa daki veri ile değiştir

Tablo 6.1'de verilen komutların işlenişini ve bir cevap gönderilmesini içeren mikrokontrolör yazılımı alt programı aşağıda verilmiştir. Gelen komut incelenerek ilgili komutlara dallanılmakta ve cevap geri gönderilmektedir.

```
;=================================================================
;    ProcessBuffer
;
;
;=================================================================

; Komutlar:

          ReadADC1 EQU   0x01
          ReadADC2 EQU   0x02
          ReadADC3 EQU   0x03
          ReadADC4 EQU   0x04
          ChangeADDR     EQU   0x06
```

```
ProcessBuffer
          MOVF Buffer, W
          MOVWF    TEMP
          SWAPF    TEMP, F                ; Komutun incelenmesi için
          MOVLW    0x0F                   ; 4 bit maskeleniyor ve
          ANDWF    TEMP, F                ; sağa kaydırılıyor.


          MOVF TEMP, W
          SUBLW    ReadADC1               ; ADC1'i oku komutu ise
          BTFSC    STATUS, Z              ; oku ve gönder
          GOTO SendADC1

          MOVF TEMP, W
          SUBLW    ReadADC2               ; ADC2'i oku komutu ise
          BTFSC    STATUS, Z              ; oku ve gönder
          GOTO SendADC2

          MOVF TEMP, W
          SUBLW    ReadADC3               ; ADC3'i oku komutu ise
          BTFSC    STATUS, Z              ; oku ve gönder
          GOTO SendADC3

          MOVF TEMP, W
          SUBLW    ReadADC4               ; ADC4'i oku komutu ise
          BTFSC    STATUS, Z              ; oku ve gönder
          GOTO SendADC4

          MOVF TEMP, W
          SUBLW    ChangeADDR             ; Adresi değiştir
          BTFSC    STATUS, Z              ; Yeni adresi geri gönder
          GOTO ChangeAddress

          MOVLW    'E'                    ; Komut algılanamadı.
          CALL SendChar
          MOVLW    0x04
          CALL SendChar

SendADC1
          BCF  ADCON0, 3
          BCF  ADCON0, 4
          CALL ADConvert
          MOVF ADRESH, W
          CALL SendChar
          MOVLW    0x04
          CALL SendChar
          RETURN

SendADC2
          BSF  ADCON0, 3
          BCF  ADCON0, 4
          CALL ADConvert
          MOVF ADRESH, W
          CALL SendChar
          MOVLW    0x04
          CALL SendChar
          RETURN

SendADC3
          BCF  ADCON0, 3
          BSF  ADCON0, 4
```

```
          CALL ADConvert
          MOVF ADRESH, W
          CALL SendChar
          MOVLW    0x04
          CALL SendChar
          RETURN

SendADC4
          BSF  ADCON0, 3
          BSF  ADCON0, 4
          CALL ADConvert
          MOVF ADRESH, W
          CALL SendChar
          MOVLW    0x04
          CALL SendChar
          RETURN

ChangeAddress
          MOVF Buffer, W
          MOVWF    TEMP2
          MOVLW    0x0F
          ANDWF    TEMP2, F          ; Yeni adres maskelenip okunuyor

          MOVLW    Adres_EEADDR
          MOVWF    DATA_EE_ADDR
          MOVF TEMP2, W
          MOVWF    DATA_EE_DATA
          CALL EEDataWrite

          CALL EEDataRead                 ; Onayla ve geri gönder
          MOVF DATA_EE_DATA, W
          MOVWF    ADRES
          CALL SendChar
          MOVLW    0x04
          CALL SendChar
          RETURN
```

6.4 Testler

Gerçeklenen özelleştirilmiş GLiVET sisteminde, sistemin güvenilirliğini test eden ve çalışma şeklini ortaya koyan üç adet test yapılmıştır. Bunlar:

1. Haberleşme protokolü testi.
2. Hatasız veri toplama testi.
3. Uzun süreli veri toplama testi.

6.4.1 Haberleşme Protokolü Testi

Haberleşme protokolü testleri, alınan ölçümlerle sinyallerin gözlenmesi ve bu yönde yazılımın ve donanımın geliştirilmesi amacıyla yapılmıştır. Yazılım ve donanımdaki düzeltmeler sonucunda en uygun sinyal düzeyleri elde edilmiştir ve hatasız çalışan protokol yapılanmıştır. Elde edilen bu son sinyal düzeyleri şekil 6.2, 6.3, 6.4, 6.5,

6.6, 6.7, 6.8, 6.9 ve 6.10'da verilmiştir. Bu ölçümlerin osiloskop ayarları ve bazı bilgileri de tablo 6.2'de verilmiştir. Bu sinyal düzeylerini gözlemlemek için bir test programı kullanılmıştır. Bu test programı, haberleşmeyi başlatan başla karakterini ve adres karakterini göndermekte, tamam karakterini okumakta ve en sonunda bitti karakterini göndermektedir. Bu haberleşme sırası osiloskopta gözlenebilmesi için bir döngü ile tekrarlatılmaktadır. Osiloskopta gözlenen karakterler sırasıyla bunlar olacaktır fakat uç birimin veri gönderme için beklemesi daha uzun olduğundan osiloskop tarafından bu bekleme sona atılmıştır ve tetikleme bitti karakterinden yapılmıştır. Yani Osiloskop ekranında gözlenen karakterler sırasıyla bitti, başla, adres ve tamam karakterleridir. Bu sinyal düzeylerini izlemek için C dilinde yazılan test programının main() fonksiyonu aşağıda verilmiştir.

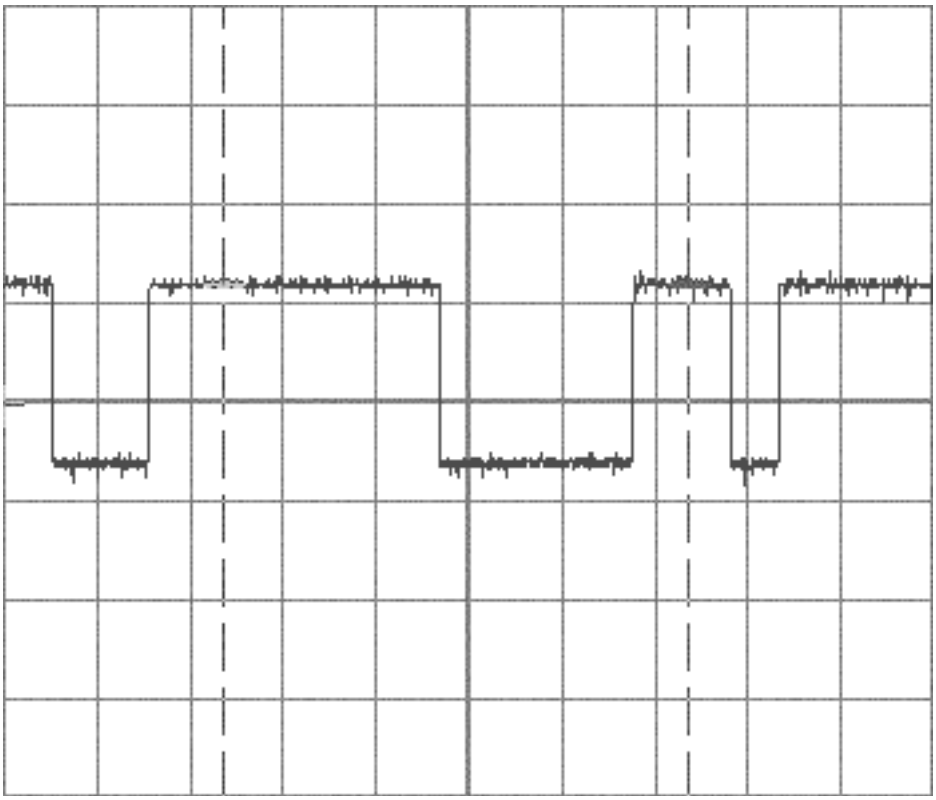
```
//---------------------------------------------------------------
// main()
//---------------------------------------------------------------
int main()
{
     int fd, i, adres;
     unsigned char n;
     unsigned char s[20] = "";
     char c[2] = "";

     printf("comm_test - Haberlesme dalgalari gozleme test"\
             "programi\n");
     printf("1 numarali adreste uc birim olması gerekmektedir.\n");
     printf("-----------------------------------------\n");
     printf("GLiVET V1.0 - A Gokhan Gultekin\n");

     fd = open_port();

     configure_port(fd);

     clear_RTS(fd);  // idle state.

     adres = 1;

     for (i=0;i<10000;i++)
     {

     c[0] = 0x02;
     send_char(fd, c);  // Başla karakteri gönderildi

     c[0] = (char) adres;
     send_char(fd, c);  // Adres gönderildi

     s[0] = get_char(fd);     // Onay karakteri alındı
     if (s[0] != 0x04)
          {
          c[0] = 0x03;
          send_char(fd, c);   // Bitti gönderildi

          printf("Adresteki uc birim bulunamadi...\n");
          exit();
```

```
        }
      c[0] = 0x03;
      send_char(fd, c);         // Bitti gönderildi
      }
      close(fd);
}
```

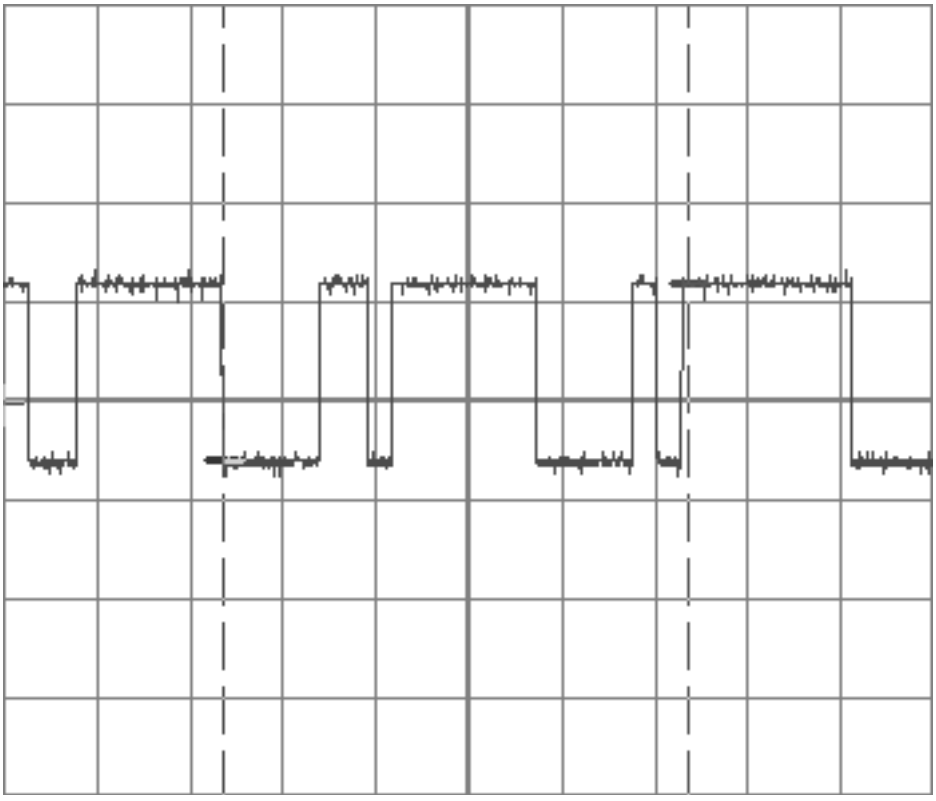
Tablo 6.2 RS232 - RS485 Dönüştürücü Kartında İzlenen Haberleşme Sinyalleri

Ölçüm	Zaman	*Kanal	Ö. Noktası	Gerilim	Vp+	Vp-	Vpp
1	100µs/böl	1	D1	1V/div	1.360V	-0.800V	2.160V
		2	-	-	-	-	-
2	200µs/böl	1	D1	1V/div	1.360V	-0.720V	2.080V
		2	-	-	-	-	-
3	500µs/böl	1	D1	1V/div	1.400V	-0.680V	2.080V
		2	D3	2V/div	1.360V	-0.640V	2.000V
4	2ms/böl	1	D1	1V/div	1.400V	-0.720V	2.120V
		2	D3	2V/div	1.360V	-0.720V	2.080V
5	100µs/böl	1	D1	1V/div	1.440V	-0.720V	2.160V
		2	R9	2V/div	3.680V	1.760V	1.920V
6	200µs/böl	1	D1	1V/div	1.400V	-0.680V	2.080V
		2	R9	2V/div	3.680V	1.760V	1.920V
7	500µs/böl	1	D1	1V/div	1.400V	-0.680V	2.080V
		2	R9	2V/div	3.680V	1.760V	1.920V
8	200µs/böl	1	R8	2V/div	3.600V	1.520V	2.080V
		2	R9	2V/div	3.680V	1.760V	1.920V
9	500µs/böl	1	R8	2V/div	3.600V	1.600V	2.000V
		2	R9	2V/div	3.680V	1.760V	1.920V
* Ekran görüntülerinde üstteki sinyal kanal 1, alttaki sinyal kanal 2'dir.							

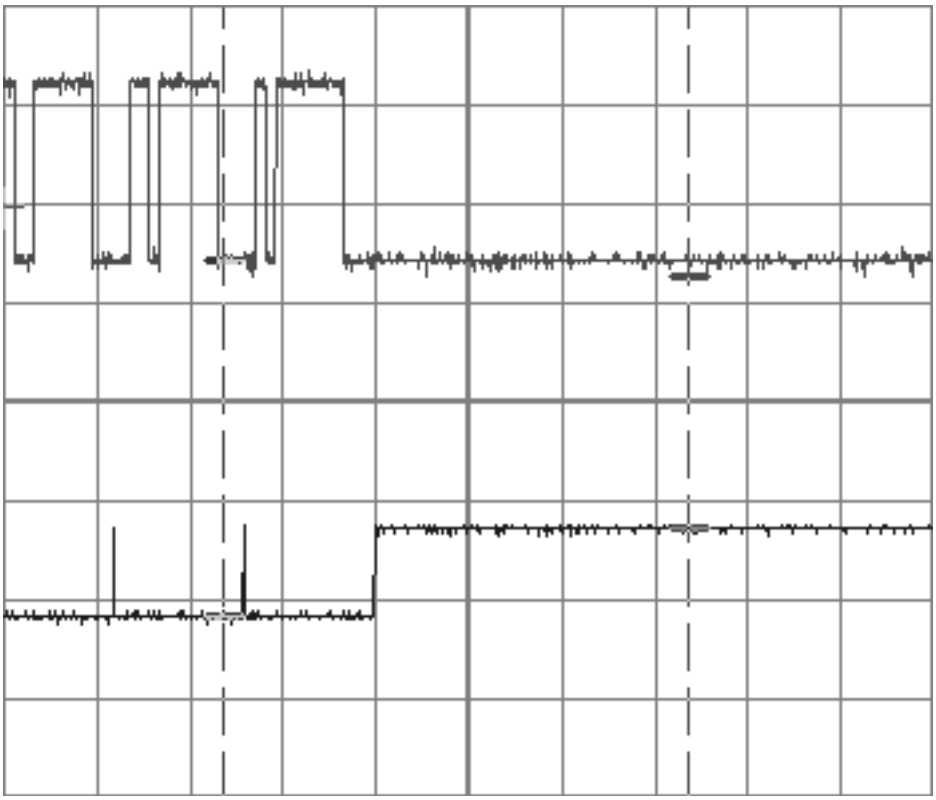
- **1. Ölçüm - Şekil 6.2:** Bu ölçümde TKB'nin RS232 çıkış sinyali yakından gözlemlenmektedir. Sinyalin sıfır düzeyi 1.360V, bir düzeyi ise -0.8V'tur. Osiloskop ekranında görünen ilk karakter bitti (0x03) karakteri olduğundan, bit düzeyleri sinyal seviyelerinden görülebilmektedir. Sağ taraftan sırasıyla başla biti (0) 1 1 0 0 0 0 0 0, toplam dokuz bit görülmektedir. En sonda dur biti gerilimi normal düzeye (1) çekmekte ve belli bir zaman sonra ikinci karakter olan başla karakterinin bitleri başlamaktadır.
- **2. Ölçüm - Şekil 6.3:** Bu ölçümde TKB'nin RS232 çıkışındaki üç karakter de sırasıyla gözlemlenmektedir. Bunlar sırasıyla bitti (0x03), başla (0x02) ve adres (0x01) karakterleridir. Gözlemlenen bitler sırasıyla: başla biti (0) 1 1 0 0 0 0 0 0 dur biti (1) - başla biti (0) 0 1 0 0 0 0 0 0 dur biti (1) - başla biti (0) 1 0 0 0 0 0 0 0 dur biti (1).



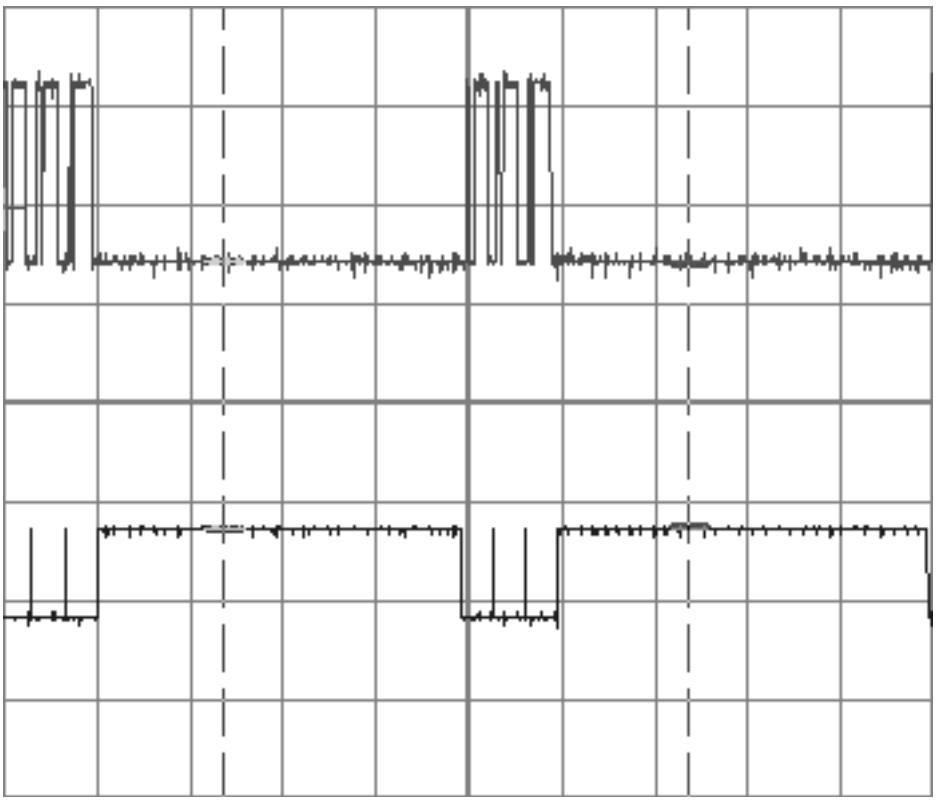
Şekil 6.2 TKB RS232 çıkış sinyali - bir karakterin ayrıntılı görünümü



Şekil 6.3 TKB RS232 çıkış sinyali - üç karakterin görünümü

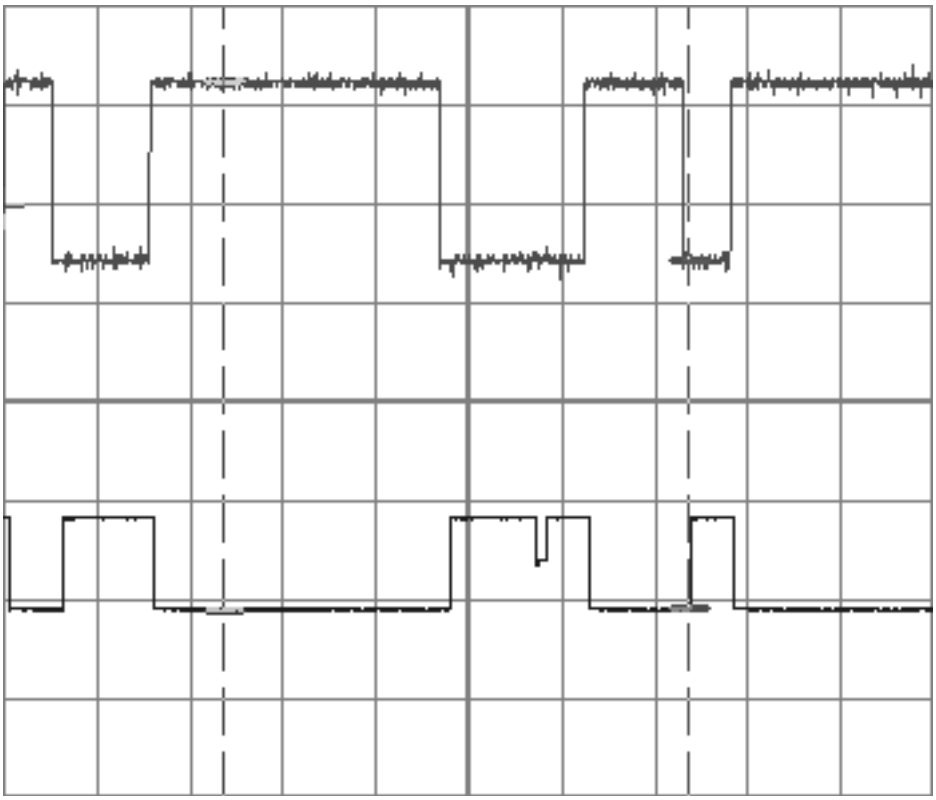


Şekil 6.4 RTS sinyali - RS232 çıkışı ile paralel görüntülenmesi

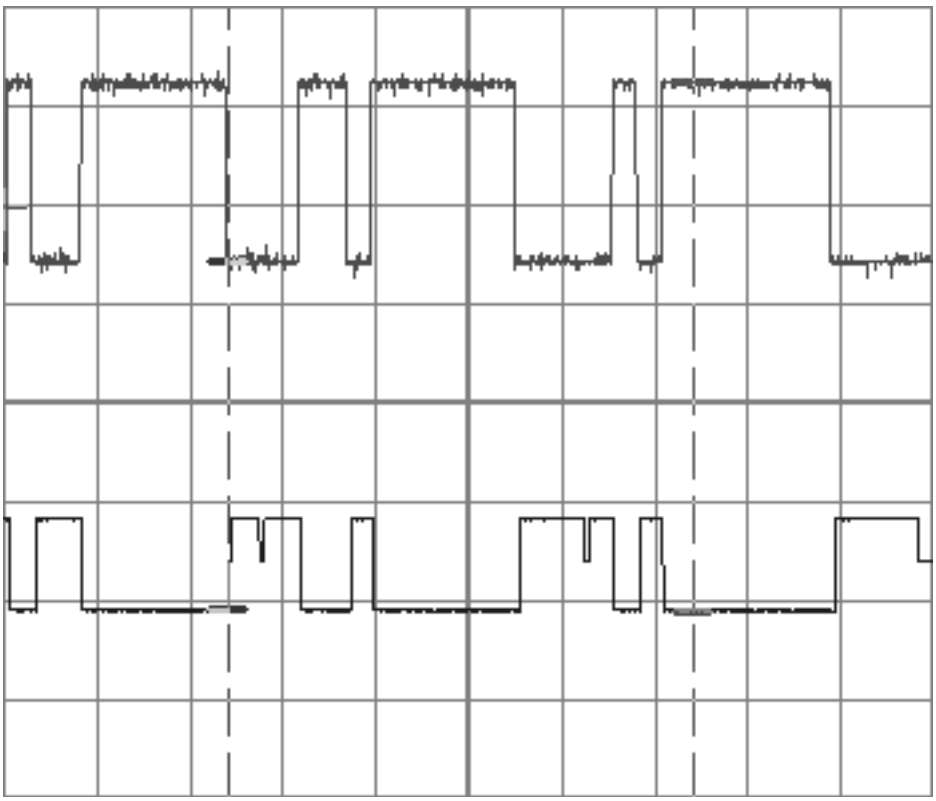


Şekil 6.5 RTS sinyali - İki ardışık haberleşme karakterleri ile görünümü

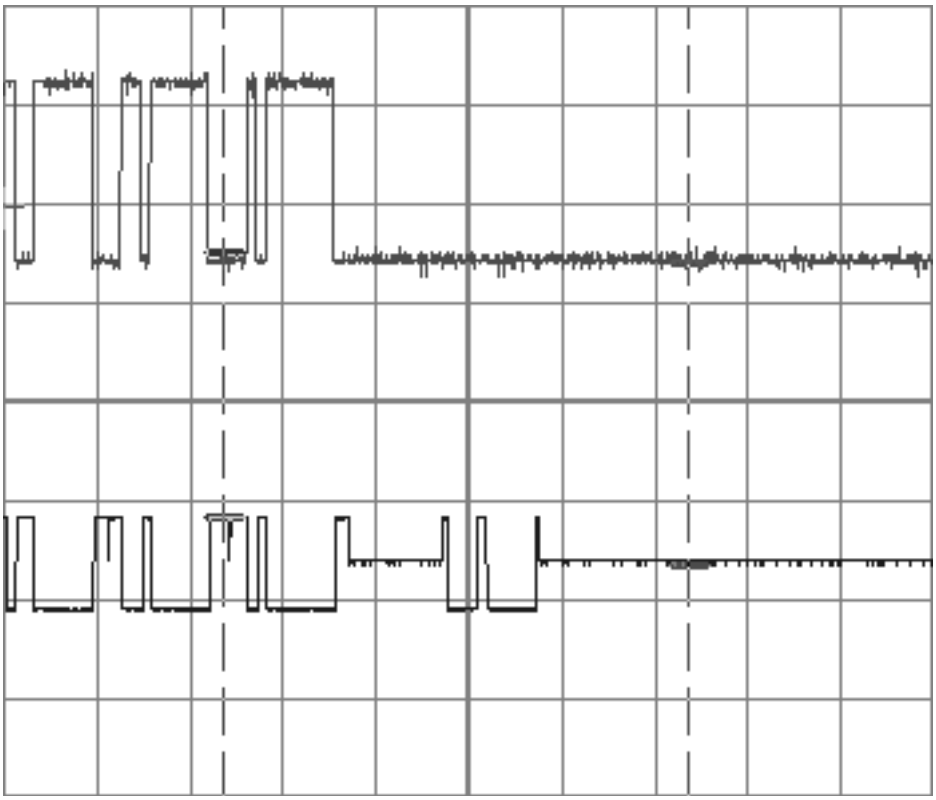
- **3. Ölçüm - Şekil 6.4:** Bu ölçümde TKB'nin RS232 çıkış sinyaline ek olarak RTS çıkış sinyali de görüntülenmektedir. RTS çıkışı -6.4V değerindeyken veri gönderme, 1.36V değerindeyken veri alma işlemleri yapılmaktadır. Ardışık iki karakter gönderme sırasında RTS sinyalinin yazılım nedeniyle veri alma konumuna geçip hemen veri gönderme konumuna geçmesi bir darbe şeklinde görülmektedir.
- **4. Ölçüm - Şekil 6.5:** Bu ölçümde ardışık üçer karakterden iki haberleşme karakterlerinin RTS sinyali ile birlikte gözlemlenebilmektedir.
- **5. Ölçüm - Şekil 6.6:** Bu ölçümde TKB'nin RS232 çıkış sinyaline ek olarak RS485 dönüşümü sonunda elde edilmiş RS485 hat sinyalleri de görüntülenmektedir. Bitlerin gözlemlenebildiği bu sinyalde hat sürücü entegre tarafından üretilen RS485 B sinyali, RS232 sinyalinin 1 olduğu durumda 3.68V, 0 olduğu durumda ise 1.76V olmaktadır.
- **6. Ölçüm - Şekil 6.7:** Bu ölçümde 5. ölçümdeki sinyal daha küçük zaman ölçeğinde gözlemlenmektedir. Bu ölçümde üç ardışık karakter görülmektedir. İki karakter arasındaki darbeler, sürücü entegrenin alma moduna geçmesi ve hemen ardından gönderme modu olarak değişmesinden oluşmaktadır.
- **7. Ölçüm - Şekil 6.8:** Bu ölçümde 6. ölçümdeki üç ardışık karakterden sonraki alınan karakter (tamam karakteri) gözlemlenmektedir. Tüm iletişim üç karakter veri gönderme ve bir karakter alma bu ekran görüntüsünde görülmektedir. Gönderilen karakter ile alınan karakter arasındaki gecikme, çakışma olmaması için mikro kontrolör tarafından yapılan gecikmedir.
- **8. Ölçüm - Şekil 6.9:** Bu ölçümde RS485 A ve B hat sinyalleri üç ardışık gönderilen karakteri görüntüleyebilecek şekilde gözlemlenmektedir. B sinyali 5. 6. ve 7. ölçümlerde de gözlenmişti. Buna ek olarak A sinyalinin de B sinyali ile birlikte değişimi bu ölçümde görülmektedir. RS485 haberleşme protokolünde olması gerektiği gibi B sinyali yüksek gerilimdeyken A sinyalinin düşük gerilimde olduğu görülmektedir. A sinyali hat 1 durumundayken 1.52V, 0 durumundayken 3.6V olmaktadır.
- **9. Ölçüm - Şekil 6.10:** Bu ölçümde RS485 A ve B hat sinyallerinin tüm iletişimi gözlemlenmektedir. Üç adet gönderilen karakterin ardından alınan tamam karakteri de aynı hat üzerinde çift yönlü haberleşmeyi göstermektedir.



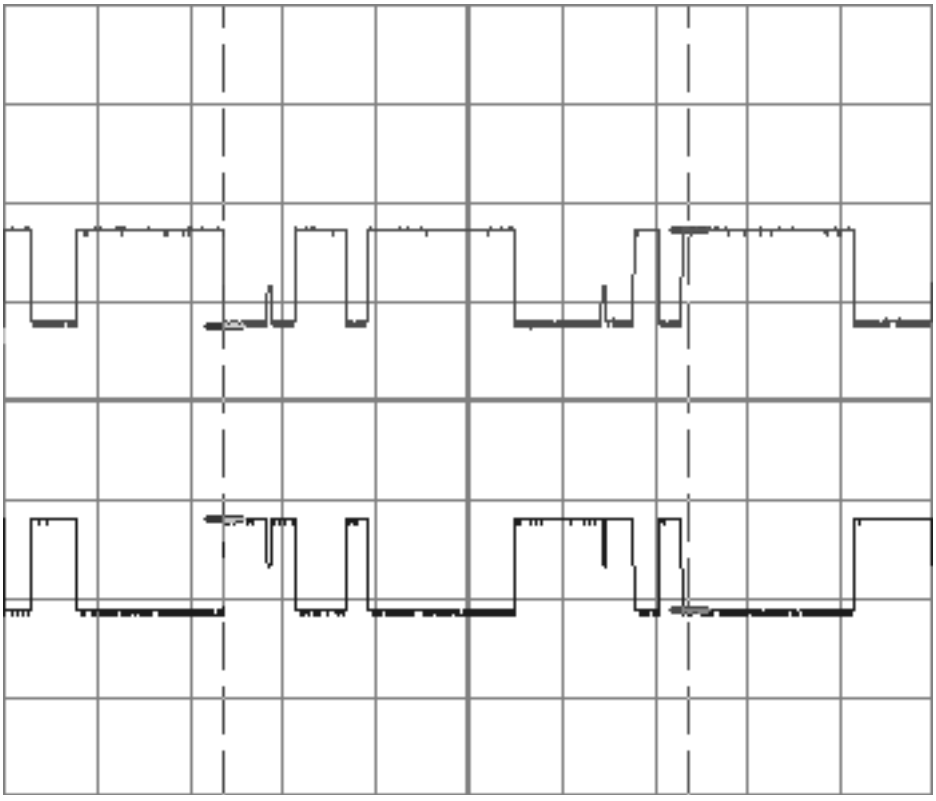
Şekil 6.6 RS232 - RS485 hat sinyalleri - Bir karakterin yakın görüntülenmesi



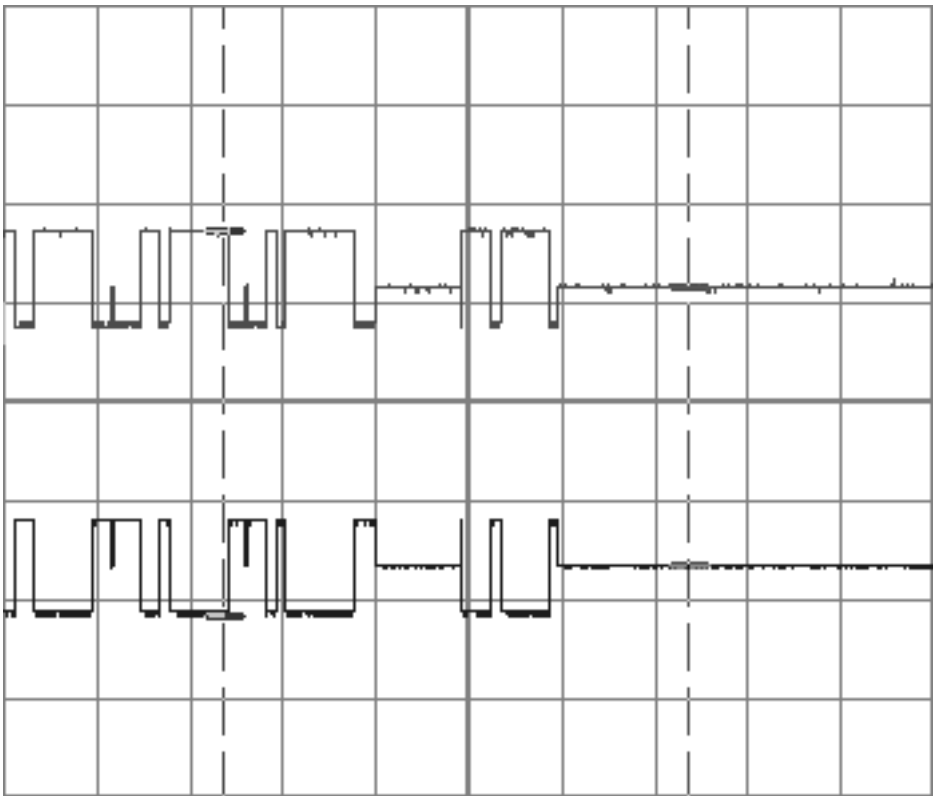
Şekil 6.7 RS232 - RS485 hat sinyalleri - Üç karakterin görüntülenmesi



Şekil 6.8 RS232 - RS485 hat sinyalleri - Alınan karakterin görüntülenmesi



Şekil 6.9 RS485 A ve B hat sinyalleri - Üç ardışık karakterin görünümü



Şekil 6.10 RS485 A ve B hat sinyalleri - Tüm haberleşmenin görünümü

6.4.2 Hatasız Veri Toplama Testi

Bu testte amaçlanan, gerçeklenen haberleşme protokolünün, donanımın ve yazılımın, toplanan veri sayısına göre hatalarını gözlemlemektir. Yani milyonlarca kez aynı veriyi toplayarak, örneğin değeri belli ve sabit olan bir analog kanalı, toplanan verideki tutarlılık gözlemlenecektir. Hata sayısı, verilerdeki tutarsızlıklar veya bağlantı hataları kaydedilerek veri toplama sisteminin çok fazla veri üzerindeki hataları incelenecektir.

Bu test için data_test.c programı yazılmıştır. readADC fonksiyonu kullanılarak birinci uç birimin birinci kanalından istenilen sayıda veri okumaktadır ve okunan değeri girilen karşılaştırma değeri ile karşılaştırarak farklı ise hatayı belirtmektedir. Program ayrıca istenilen periyotlarda da okunan veri sayısını ekrana yazmaktadır. Birinci parametre okuma sayısını, ikinci parametre kanalın bilinen değerini (karşılaştırılacak değeri), üçüncü parametre ise rapor aralığını belirtmektedir.

```
//---------------------------------------------------------------
// main()
//---------------------------------------------------------------
int main(int argc, char *argv[])
{
     int fd, adres, kanal, i, okuma, sayi, count, rapor;
```

```
    unsigned char n;

    if (argc != 4)
    {
    printf("\ndata_test - 1. ucbirimdeki 1. ADC kanalini okur.\n");
    printf("-----------------------------------------\n");
    printf("Kullanimi: data_test [okuma_sayisi] [ADC degeri]"\
           "[rapor araligi]\n");
    printf("GLiVET V1.0 - A Gokhan Gultekin\n");
    exit();
    }
    fd = open_port();

    configure_port(fd);

    clear_RTS(fd);  // idle state.

    okuma = atoi(argv[1]);
    sayi = atoi(argv[2]);
    rapor = atoi(argv[3]);

    adres = 1;
    kanal = 1;
    count = 0;

    for(i=1;i<okuma+1;i++)
    {
         n = readADC(adres,kanal,fd);
         if (n != sayi)
              {
              printf("%d. ornek hatali: %d\n", i, n);
              count++;
              }
         if (i%rapor == 0)
              printf("%d. okuma\n", i);
    }
    printf("Toplam hata sayisi: %d\n", count);
    close(fd);
}
```

Test programı bir milyon veri için çalıştırılmıştır ve oluşan ekran görüntüsü aşağıda verilmiştir. readADC komutu ile önceden okunan analog değer 97 olduğundan karşılaştırılacak değer olarak 97 girilmiştir. Onbin örnekte bir de rapor vermesi için üçüncü parametre onbin olarak girilmiştir. Toplam test süresi beş saat elli dakika sürmüştür. Test sonrasında okunan bir milyon veride hiç hata oluşmamıştır.

```
glivet:~/glivet# ./data_test 1000000 97 10000
10000. okuma
20000. okuma
30000. okuma
.
.
970000. okuma
980000. okuma
990000. okuma
1000000. okuma
Toplam hata sayisi: 0
glivet:~/glivet#
```

Test sonucu olarak bir milyon veride sıfır hata oluşması, haberleşme protokolüne hata belirleme bitlerinin koyulmamasını desteklemektedir. Band genişliğini düşürmemek için protokole hata belirleme bitleri koyulmamıştır. Bunun yerine çok düşük bir olasılıkla oluşabilecek hatalar göz ardı edilmektedir.

6.4.3 Uzun Süreli Veri Toplama Testi

Uzun süreli çalışmadaki hataların gözlenmesi için bu test gerçekleştirilmiştir. Bu testte toplanan veri sayısından çok sistemin çalışma zamanı önem kazanmaktadır. Bu test için özel bir program yazılmamıştır, özelleştirilen GLiVET sistemi kabuk programı "sample" çalıştırılarak sistemin beş gün süreyle veri toplanması sağlanmıştır. Konfigürasyon dosyası ayarları tablo 6.3'teki gibidir.

Tablo 6.3 Uzun Süreli Veri Toplama Testi Konfigürasyon Değerleri

Glivet istasyon tanımlama numarası	G0036
FTP adresi	192.168.0.1
FTP yolu	/glivet_data/
Toplanacak toplam örnek sayısı	86.400 (5 gün)
Veri dosyasını gönderme periyodu	600s (10dk.)
Örnekleme periyodu	5s
Örneklenecek kanallar	1. uç birimin 1. kanalı
	1. uç birimin 2. kanalı
	1. uç birimin 3. kanalı
	1. uç birimin 4. kanalı
	2. uç birimin 1. kanalı
	2. uç birimin 2. kanalı
	2. uç birimin 3. kanalı
	2. uç birimin 4. kanalı

Test sonucunda beş gün boyunca örneklenen veriler, belirtilen adrese gönderilmiştir ve burada depolanmıştır. glivet.log dosyasında ve dosya isimlerinde gönderildikleri zamanlar belirtilmiştir. Veriler sabit olduğundan her dosya paketi birbiri ile aynı boyutta ve içeriktedir. Bu aynı zamanda gönderilen verilerin doğruluklarını da ispatlamaktadır. Tüm dosyalar birleştirildiğinde konfigürasyon dosyasında yazan örnek sayısı kadar satır olduğu yani eksik veya hatalı bir okuma yapılmadığı gözlenmiştir. Gerçek sistemde batarya desteği olacağından bu test boyunca elektrik kesintisi sisteme uygulanmamıştır.

7. SONUÇ

Bir veri toplama sistemi alt yapısı olarak geliştirilen GLiVET, esnekliği ile, ölçeklendirilebilme özelliği ile, dağıtık yapılandırılabilmesi ile öne çıkmaktadır. GLiVET sistemi ile temelde bitmiş bir ürün tanımı yapılmasa da bitmiş bir ürünü çok kısa sürede elde etmeye yarayacak uygun maliyetlerde bir alt yapı sunulmaktadır.

Tasarlanacak sistemin büyüklüğüne göre ölçeklendirilebilmektedir. Bir coğrafya üzerine yayılmış algılayıcılardan veri toplamaktan, sadece bir oda içerisinde verileri toplamaya kadar ölçeklendirilebilmektedir. Her iki uç nokta arasındaki sistemlere destek verebilecek şekilde tasarlanmıştır.

Verilerin toplanacağı algılayıcıların bağımsızlığı sistemin esnekliğini sağlamaktadır. Bu bağımsızlığı sağlamak için küçük de olsa bir donanım tasarımı gerektirse de, bir kaç standart kartın tasarımı gerçekleştirildiğinde oldukça fazla algılayıcı sistem tarafından kapsanabilir duruma gelecektir. Sadece özel algılayıcılar yeni donanım tasarımı gereksinimi getirecektir.

Elde edilen sistemin avantajlarının yanında bazı dezavantajları da bulunmaktadır. Bunlardan en önemlisi ise veri toplama hızıdır. Tasarlanan sistemdeki veri toplama hızını sınırlayan en büyük etken VTK'lar ve TKB arasında kullanılan veri yolunun bant genişliğidir. RS232 temeline dayanılarak tasarım yapıldığından burada elde edilebilecek en büyük hız 115.200bps'dır. Eğer tek VTK ve tek kanal örnekleme yapılacak olsa 8'bitlik bir örnekleme için saniyede 14.400 örnek toplanabilecektir. Tabi ki kanal sayısı arttıkça bu sayı bölünecektir. Uzun süreli ve çok hızlı olmayan veri toplama işlemi gerçekleştirilmek üzere tasarlandığından sistem aslında çok başarısız sayılmamaktadır fakat bu tasarımla 14.400 örnek / saniyeden daha hızlı veri toplayamayacaktır.

Sistemin ileride başarılı olabilmesi ve kararlı ürünler ortaya çıkartabilmesi için ileride yapılması gereken önemli geliştirmeler bulunmaktadır. Gelecek için en önemli hamle, veri toplama kartlarının çoğaltılarak birçok standart algılayıcının kapsanması olacaktır. Algılayıcılar işlevlerine veya ürettikleri çıkışlara göre sınıflandırılabilirler ve bunlara göre yeni standart oluşturabilecek veri toplama kartları tasarlanmalıdır.

Örneğin basınç algılayıcılarına veya sıcaklık algılayıcılarına ya da +/-100uV veya 0-30mV analog çıkış üreten algılayıcılara göre standart sınıflara göre kartlar tasarlanabilir. Bu şekilde yeni bir sistem özelleştirmesine gidildiğinde özelleştirme süresi hazır tasarlanmış kartların kullanılmasından dolayı oldukça kısalacaktır.

Sistemin kullanılabilirliğini arttırmak ve yaygınlaşmasını sağlamak amacı ile kullanıcı arayüzlerinin tasarlanması gerekmektedir. İleride yapılabilecek bu iyileştirme, sistemin sadece deneyimli kullanıcılar tarafından tüm kullanıcılar tarafından kullanılabilmesini sağlayacaktır. Bu bir sektör için özelleştirilmiş bir sistemin o sektör içindeki bir kişinin rahatça kullanabilmesini, ayarlarının değiştirilmesini ve yöneticiliğini yapabilmesini sağlayacaktır. Böyle bir arayüzün varlığında, yönetici konfigürasyonunda değişiklik yapmak istediği noktaya ssh ile bağlanıp linux komutları yardımıyla değişiklikler yapmak yerine, kullanıcı arayüzü bulunan bir program tarafından menüler yardımı ile değişiklikler yapabilecektir. Tabi ki sistem tarafından sunulan ssh bağlantıları ve gerekli linux komutları alt tarafta çalışacaktır fakat kullanıcının bunlardan bir haberi olmayacaktır.

Bu arayüze bir örnek bir meteoroloji veri toplama sistemi verilebilir. Meteorolog bir coğrafi noktadan topladığı nem oranı verisinin örnekleme periyodunu arttırmak ve daha ayrıntılı inceleme yapmak istemektedir. Geliştirilebilecek arayüz sayesinde yapması gereken sadece birinci menüden noktayı seçmek, ikinci menüden nem oranı algılayıcısını seçmek, örnekleme periyodu seçim alanından istediği seçimi yapmak ve güncelleme düğmesine basmak olacaktır. Düğmeye basıldığı anda alt tarafta belirtilen noktanın adresi bulunarak ssh komutları ile bağlantı sağlanacak, konfigürasyonda değişiklikler yapılarak örnekleme programı tekrardan başlatılacaktır. Meteoroloğa görüntülenecek tek ekran ise başarılı bir değişiklik gerçekleştirildiği olacaktır. Bu örnek aslında geliştirilen GLiVET sisteminin de gücünü göstermektedir. Sistem sayesinde basit bir hamle ile en tepeden en alttaki noktaya müdahale gerçekleştirilebilmektedir, kullanıcı arayüzü sadece sıralı işlemleri belli parametreler ile uygulayacaktır.

GLiVET sisteminin kullanılabilirliğinin arttırılması için yapılan eklemelerin yanında gelecekte sistemin başka amaçlara da hizmet edebileceği köklü eklentiler de yapılabilir. Buna verilebilecek en önemli örnek, bir dağıtık veri toplama sistemi olan GLiVET'in, kontrol özelliği eklenerek dağıtık kontrol sistemine çevrilmesidir. Veri toplama noktası olarak kullanılan her bir nokta kontrol noktası olarak yeniden

tasarlanabilir ve bu noktalara uzaktan müdahale imkanı tanınabilir. Sayısal veya analog çıkış alınarak uzaktan kontrolün mümkün olabileceği noktalara ayrıca basit kontrol programcıkları yazılarak bu noktalar otomatik kontrol noktalarına da çevrilebilir. Bu sayede hem uzaktan kontrol imkanı bulunan hem de kontrol programcıklarına müdahale edebilme imkanı olan otomatik kontrol noktaları elde edilmiş olur. Programcıkları uzaktan yazma ve o noktanın uzaktan kontrolünü kolaylaştırma adına noktalara internet sunucular yerleştirilebilir. Bu sunucular üzerinde oluşturulacak internet sayfalarına uzaktan bağlanılarak noktalarda uzaktan kontrol daha kolay hale getirilebilir.

Bir veri toplama alt yapısı olan GLiVET, dağıtık veya lokal bir çok veri toplama yöntemini gerçekleyebilecek, gelişime açık, Linux tabanlı ve açık kaynak kodlu bir sistemdir. Uygun maliyetlerde son ürün oluşturmada büyük kolaylılar sağlayacak bu sistem ile elde edilecek ürünler ileride yüksek maliyetli muadillerine rakip olabilecek kapasitededir.

KAYNAKLAR

[1] **Taylor, H. R.,** 1997. Data acquisition for sensor systems, *Chapmann & Hall,* New York.

[2] **Aronoff, S.,** 1995. Geographic Information Systems: A Management Prespective, *WDL Publications*, Canada.

[3] **Mussio, L. and Forlani, G. and Crosilla F.,** 1996. Data acquisition for multimedia GIS, *Springer Wien,* New York.

[4] **Maxim,** 2000. Analog-Signal Data Acquisition in Industrial Automation Systems, *Maxim Integrated Products,* USA.

[5] **Mota, A. and Fonseca, J. A. and Santos, F.,** 1998. Low Cost Data Acquisition Systems based on Standard Interfaces, *IEEE,* 433-437.

[6] **Gagne, M.,** 2003. Linux System Administration A User's Guide, *Addison-Wesley*, USA.

[7] **Loukides, M. and Oran A.,** 1997. Programming with GNU Software, *O'Reilly & Associates Inc.,* USA.

[8] **Garrels, M.,** 2004. Introduction to Linux, A Hands on Guide, GNU Free Documentation License, The Linux Documentation Project, http://www.tldp.org.

[9] **Beekmans, G.,** 2003. Linux From Scratch Version 5.0, The Linux Documentation Project, http://www.tldp.org.

[10] **Lentnon, A.,** 2001. Embedding Linux, *IEE REVIEW*, **MAY 2001**, 33-37.

[11] **AAEON,** 2004. PCM-5335 Geode GX1-300 Data Sheet, *AAEON Electronics Inc.*, ftp://data.aaeon.com/DOWNLOAD/Data%20Sheet%20PDF/PCM-5335.pdf.

[12] **Linux Devices,** 2004. Linux PDA Quick Reference Guide, *Ziff Davis Publishing Holdings Inc.*, http://www.linuxdevices.com/articles/AT8728350077.html.

[13] **Linux Devices,** 2005. MontaVista aims embedded Linux ecosystem at mobile phones, *Ziff Davis Publishing Holdings Inc.*, http://www.linuxdevices.com/news/NS9628832666.html.

[14] **Blast,** 2004. About White Dwarf, *Blast Internet Services*, http://www.blast.com/wd_about.html.

[15] **Lowe, K.,**2003. Kernel Rebuild Guide, *Digital Hermit*, www.digitalhermit.com.

[16] **Skoric, M.,** 2004. LILO mini-HOWTO, GNU Free Documentation License, The Linux Documentation Project, http://www.tldp.org.

[17] **Erlich, S.,** 2003. Custom Linux: A Porting Guide, Porting LinuxPPC to a Custom SBC, GNU Free Documentation License, The Linux Documentation Project, http://www.tldp.org.

[18] **Horton, D.,** 2004. Pocket Linux Guide, GNU Free Documentation License, The Linux Documentation Project, http://www.tldp.org.

[19] **Sweet, M. R.,** 2003. The Serial Programming Guide for POSIX Operating Systems, GNU Free Documentation License.

[20] **Lawyer, S. D.,** 2004. Serial HOWTO, GNU Free Documentation License, The Linux Documentation Project, http://www.tldp.org.

[21] **Unix Manual Page for ioctl,** GNU Free Documentation License, The Linux Documentation Project, http://www.tldp.org.

[22] **Unix Manual Page for termio,** GNU Free Documentation License, The Linux Documentation Project, http://www.tldp.org.

[23] **Maxim,** 2003. Low-Power, Slew-Rate-Limited RS-485/RS-422 Transceivers, *Maxim Integrated Products*, USA.

[24] **Maxim,** 2000. Selecting and Using RS-232, RS-422, and RS-485 Serial Data Standards, Application Note, *Maxim Integrated Products*, USA.

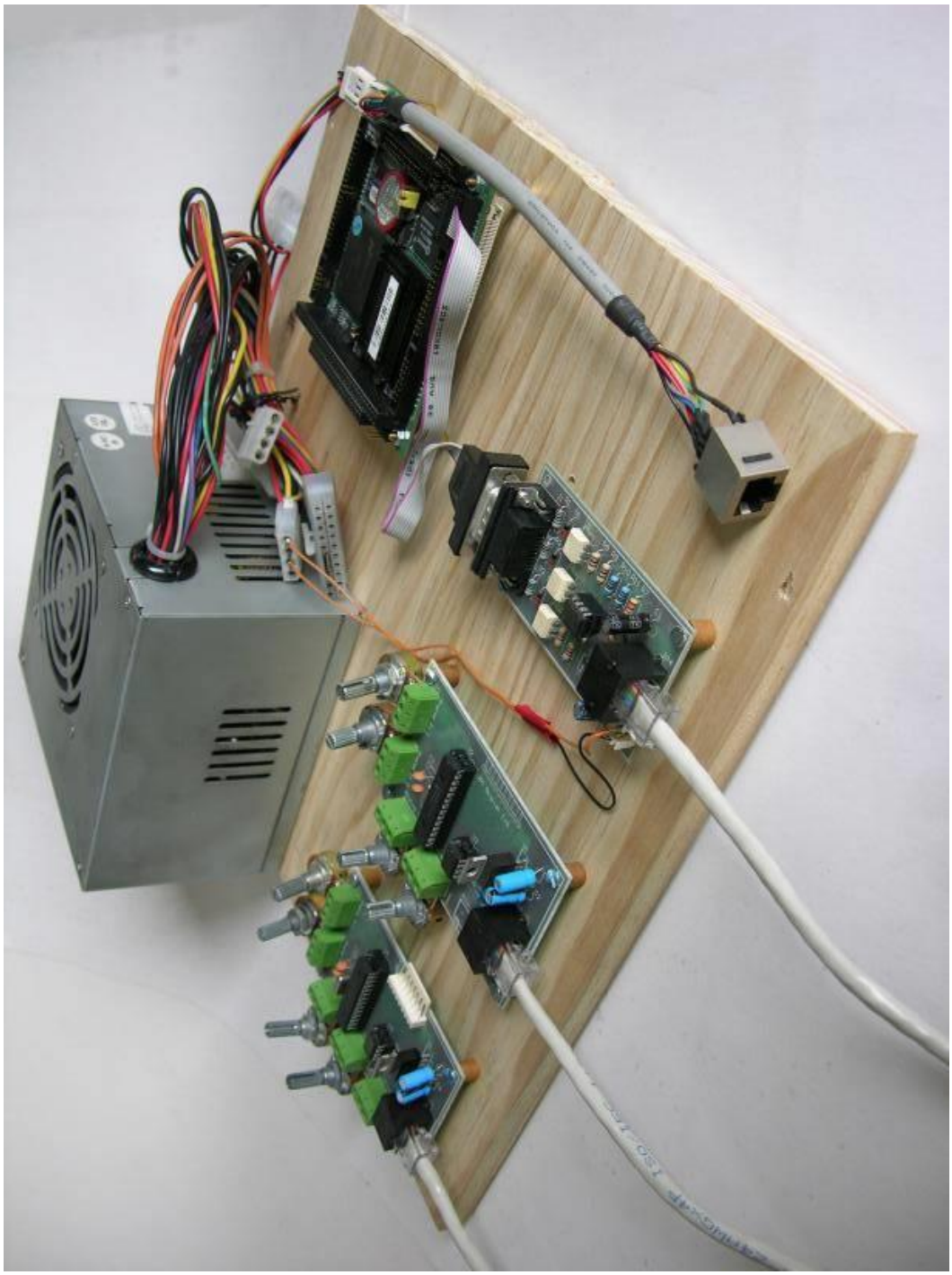
[25] **Maxim,** 2000. Explanation of Maxim RS-485 Features, Application Note, *Maxim Integrated Products*, USA.

[26] **Axelson, J.,** 1999. Designing RS-485 Circuits, CIRCUIT CELLAR, *The Computer Applications Journal*, **107**, 20-24.

[27] **Microchip,** 2001. PIC16F87X Data Sheet 28/40-Pin 8-Bit CMOS FLASH Microcontrollers, *Microchip Technology Inc*, USA.

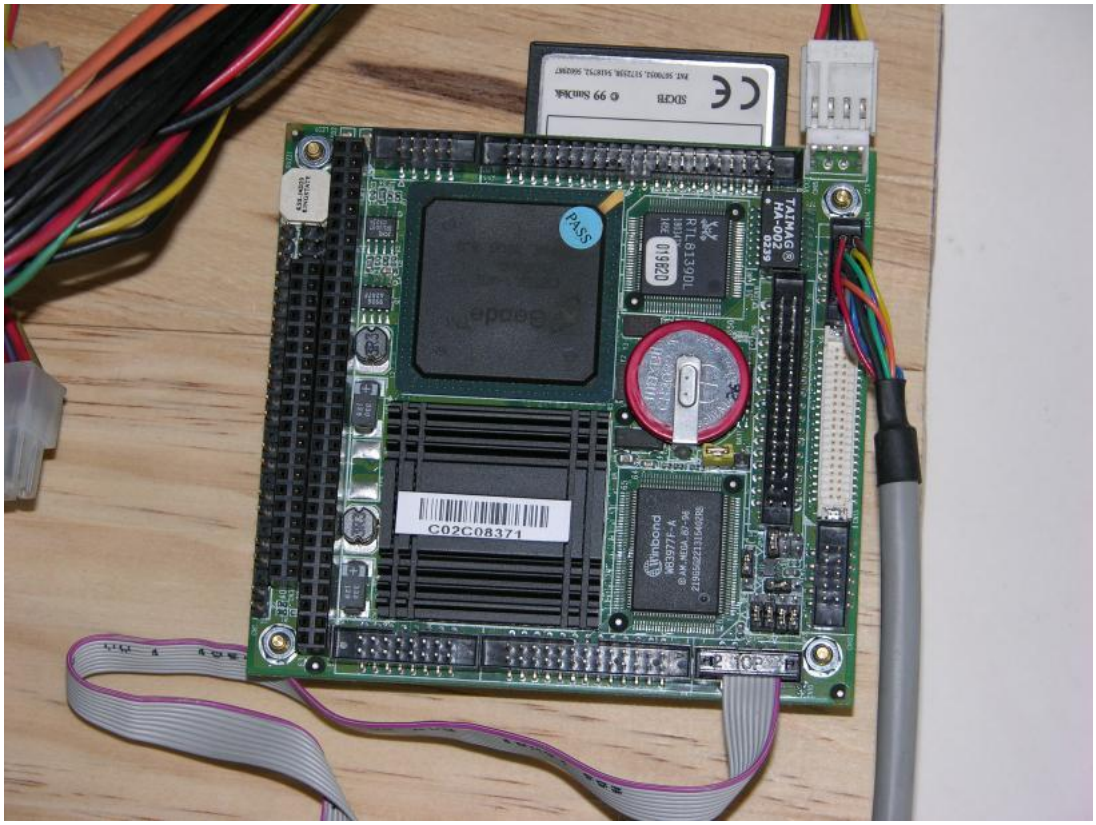
[28] **Microchip,** 2003. Asynchronous Communications with the PICmicro® USART, Application Note, *Microchip Technology Inc*, USA.

EK A: GERÇEKLENMİŞ SİSTEMİN FOTOĞRAFLARI



Şekil A.1 Gerçeklenmiş GLiVET sistemi, genel görünüm 1

Şekil A.2 Gerçeklenmiş GLiVET sistemi, genel görünüm 2



Şekil A.3 Gerçeklenmiş GLiVET sistemi, TKB yakından görünüm

EK B: CD – BİLGİSAYAR PROGRAM KODLARI

Ek CD'nin içeriği:

- C programları kaynak kodları
    - readADC.c
    - scanbus.c
    - changeADDR.c
    - sample2file.c
    - glivet.h
- Kabuk programcıkları ve konfigürasyon dosyaları
    - sample
    - glivet_config
    - glivet.log
- Mikrokontrolör yazılımı
- İşletim sistemi 32MB CF kart görüntüsü

ÖZGEÇMİŞ

A. Gökhan Gültekin 1980 yılında İstanbul'da doğdu. Orta ve lise öğrenimini Eyüboğlu Lisesinde tamamladıktan sonra 1997 yılında İstanbul Teknik Üniversitesi Elektrik ve Elektronik Fakültesi Elektronik ve Haberleşme Mühendisliğini kazandı. 2001 yılında mezun olduktan sonra aynı sene İ.T.Ü Bilişim Enstitüsü Bilgi Teknolojileri Yüksek Lisansı programını kazandı. 2002 yılında bu bölümden mezun olduktan sonra, yine aynı sene İ.T.Ü Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Yüksek Lisansı bölümünü kazandı. Üç senedir kurucusu olduğu ANKA Bilgi Teknolojileri Ar-Ge A.Ş.'de, medikal ve endüstriyel Ar-Ge projeleri yöneticiliği yapmaktadır.