BAŞKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI
TEZLİ YÜKSEK LİSANS PROGRAMI

CAN BUS SİSTEMİ İÇİN FPGA TABANLI SALDIRI TESPİT SİSTEMİNİN GELİŞTİRİLMESİ

HAZIRLAYAN

ORHUN ARPACI

YÜKSEK LİSANS TEZİ

ANKARA – 2022

BAŞKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI
TEZLİ YÜKSEK LİSANS PROGRAMI

CAN BUS SİSTEMİ İÇİN FPGA TABANLI SALDIRI TESPİT SİSTEMİNİN GELİŞTİRİLMESİ

HAZIRLAYAN

ORHUN ARPACI

YÜKSEK LİSANS TEZİ

TEZ DANIŞMANI

PROF. DR. HAMİT ERDEM

ANKARA – 2022

BAŞKENT ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

Elektrik Elektronik Mühendisliği Anabilim Dalı Elektrik Elektronik Mühendisliği Tezli Yüksek Lisans Programı çerçevesinde Orhun ARPACI tarafından hazırlanan bu çalışma, aşağıdaki jüri tarafından Yüksek Lisans Tezi olarak kabul edilmiştir.

Tez Savunma Tarihi: 24/02/2022

Tez Adı: CAN Bus Sistemi için FPGA Tabanlı Saldırı Tespit Sisteminin Geliştirilmesi

Tez Jüri Üyeleri (Unvanı, Adı- Soyadı, Kurumu)	**İmza**
Dr. Öğr. Üyesi Atilla Özgür, Ankara Yıldırım Beyazıt Üniversitesi	..…….
Prof. Dr. Hamit Erdem, Başkent Üniversitesi	..…….
Dr. Öğr. Üyesi Selda Güney, Başkent Üniversitesi	..…….

ONAY

Prof. Dr. Ömer Faruk ELALDI
Fen Bilimleri Enstitüsü Müdürü

Tarih : … / … / ….……

BAŞKENT ÜNİVERSİTESİ
FEN BİLİMLER ENSTİTÜSÜ
YÜKSEK LİSANS TEZ ÇALIŞMASI ORİJİNALLİK RAPORU

Tarih: 24/02/2022

Öğrencinin Adı, Soyadı : Orhun ARPACI
Öğrencinin Numarası : 21710297
Anabilim Dalı : Elektrik – Elektronik Mühendisliği Anabilim Dalı
Programı : Tezli Yüksek Lisans Programı
Danışmanın Unvanı/Adı, Soyadı : Prof. Dr. Hamit Erdem
Tez Başlığı : CAN Bus Sistemi için FPGA Tabanlı Saldırı Tespit Sisteminin Geliştirilmesi

Yukarıda başlığı belirtilen Yüksek Lisans tez çalışmamın; Giriş, Ana Bölümler ve Sonuç Bölümünden oluşan, toplam 121 sayfalık kısmına ilişkin, 24/02/2022 tarihinde tez danışmanım tarafından Turnitin adlı intihal tespit programından aşağıda belirtilen filtrelemeler uygulanarak alınmış olan orijinallik raporuna göre, tezimin benzerlik oranı %3 tür. Uygulanan filtrelemeler:

1. Kaynakça hariç
2. Alıntılar hariç
3. Beş (5) kelimeden daha az örtüşme içeren metin kısımları hariç

"Başkent Üniversitesi Enstitüleri Tez Çalışması Orijinallik Raporu Alınması ve Kullanılması Usul ve Esaslarını" inceledim ve bu uygulama esaslarında belirtilen azami benzerlik oranlarına tez çalışmamın herhangi bir intihal içermediğini; aksinin tespit edileceği muhtemel durumda doğabilecek her türlü hukuki sorumluluğu kabul ettiğimi ve yukarıda vermiş olduğum bilgilerin doğru olduğunu beyan ederim.

Öğrenci İmzası:

ONAY
Tarih: 24/02/2022
Tez Danışmanı
Prof. Dr. Hamit Erdem

TEŞEKKÜR

Yüksek lisans öğrenim hayatım boyunca göstermiş olduğu tüm anlayış, sabır ve hoşgörüsü olmak üzere, çalışmanın sonuca ulaştırılmasında her zaman yardımcı ve yol gösterici olduğu için tez danışmanım Prof. Dr. Hamit ERDEM'e,

Bu çalışmanın sonuca ulaşmasında her türlü desteğini eksik etmeyen arkadaşlarım Buğra KABASAKAL, Erdoğan Berkay TEKİNCAN'a ve Anılcan ÖZÇELİK'e,

Eğitim hayatım boyunca bana inanıp her zaman destek olan annem Dilek ARPACI'ya, babam Fikret ARPACI'ya ve ağabeyim Burak ARPACI'ya,

Bu süreçte desteğini hep yanımda hissettiğim eşim Fidan Didem ARPACI'ya

Teşekkürü borç bilirim.

Orhun ARPACI

ÖZET

Orhun ARPACI

CAN BUS SİSTEMİ İÇİN FPGA TABANLI SALDIRI TESPİT SİSTEMİNİN GELİŞTİRİLMESİ

Başkent Üniversitesi Fen Bilimleri Enstitüsü

Elektrik – Elektronik Mühendisliği Anabilim Dalı

2022

Günümüz araç sistemleri teknolojik açıdan hızla gelişmektedir. Gelişen araç sistemlerindeki elektronik ünitelerin sayısı da git gide artmaktadır. Araç içi elektronik birimlerin farklı iletişim ağları ile dış dünya iletişimime açılması ile bu birimlerin veri iletişiminin güvenliği araştırma konusu olmuştur. Günümüz araçlarda araç içi iletişimde CAN veri yolu kullanılmaktadır. CAN veri yolu bit hatası açısından güvenilir bir veri yolu olsa da birçok güvenlik açığını da barındırmaktadır. CAN veri yolunun güvenlik açıklarını kapatmak için birçok saldırı tespit sistemi geliştirilmektedir. Saldırı tespit sistemleri ile CAN veri yoluna yapılabilecek saldırıların tespiti ve karşı önlemlerin alınması hedeflenmektedir. Bu tez kapsamında araç içerisindeki CAN veri yoluna fiziksel olarak yapılacak bir saldırıyı tespit edebilen bir saldırı tespit sistemi gerçeklenmiştir. Bu gerçeklemede elektronik birimlerin veri transferi sırasında, CAN veri yolunda oluşturdukları sinyaller incelenmiştir. İncelenen CAN sinyallerinden, sinyallerin parmak izlerinin çıkarılabileceği kanıtlanmıştır. Sinyallerin parmak izlerinin çıkarılması için FPGA üzerinde bir CAN IP çekirdeği geliştirilmiştir. Çıkarılan sinyal özellikleri MLP ve LSTM olmak üzere iki farklı tür yapay sinir ağlarının eğitiminde kullanılmıştır. Eğitilen yapay sinir ağları ile CAN veri yoluna fiziksel olarak saldırı gerçekleştiren bir elektronik cihazın bıraktığı sinyalin parmak izlerinden saldırı tespiti yapılmıştır.

ANAHTAR KELİMELER: Saldırı Tespit Sistemleri, CAN, FPGA, MLP, LSTM, Sinyal Parmak İzi, Yapay Sinir Ağı, Elektronik Kontrol Birimi

ABSTRACT

Orhun ARPACI

DEVELOPMENT OF FPGA BASED INTRUSION DETECTION SYSTEM FOR CAN BUS

Baskent University Institute of Science

Department of Electrical and Electronics Engineering

2022

Today's vehicle systems have been developing rapidly. The number of electrical units in the vehicle systems are increasing. With the opening of the in-vehicle electronic units to the outside world with different communication networks, the security of the data communication of these units has been the subject of research. In vehicle systems CAN bus is used to communicate between electrical control units (ECU). CAN bus is a very reliable communication network according to bit error rate. However, CAN bus has numerous vulnerabilities in terms of security. In order to seal these security vulnerabilities many intrusion detection systems (IDS) have been developed. With intrusion detection systems, it is aimed to detect attacks on the CAN bus and take countermeasures. In this thesis, an intrusion detection system which can detect attacks in the physical layer of the CAN bus. For his purpose, CAN signals, which are produced by ECUs, were analyzed. It is proven that ECU footprints can be extracted from CAN signals. In this thesis, to extract footprint of signals, a CAN IP core was implemented on FPGA. Extracted footprints were used to train different neural network architectures which are MLP and LSTM. Trained neural networks were used to detect intrusions by using signal footprints of intruder ECUs.

KEYWORDS: Intrusion Detection System, IDS, Neural Network, FPGA, MLP, LSTM, CAN Bus, Electronic Control Unit, ECU

İÇİNDEKİLER

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

SİMGELER VE KISALTMALAR LİSTESİ

ACK	Acknowlegment
ANN	Artificial Neural Network
BDT	Bagged Decision Tree
BRAM	Block Random Access Memory
CAN	Controller Area Network
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CSMA\CD	Carrier Sense Multiple Access\Collusion Detection
DBN	Deep Belief Network
DNN	Deep Neural Network
DoS	Denial of Service
ECU	Electronic Control Unit
EOF	End Of Frame
FIFO	First In First Out
FPGA	Field Programmable Gate Array
FSA	Finite State Automation
GND	Ground
GUI	Graphical User Interface
GPU	Graphical Processing Unit
HTM	Hierarchical Temporal Memory
IC	Integrated Circuit
ID	Identifier Field
IDE	Identifier Extention Bit
IDS	Intrusion Detection System
IP	Intellecutal Property
ISO	International Standardization Organization
Kbps	Kilobit Per Second
KSPS	Kilo Sample Per Second
LIN	Local Interconect Networks
LLC	Logical Link Control
LSB	Least Significant Bit
LSTM	Long-Short Term Memory
MAC	Medium Access Control

Mbps	Megabit Per Second
MLP	Multi-Layer Perceptron
MMCM	Mixed-Mode Clock Manager
MOST	Media Oriented System Transport
MSB	Most Significant Bit
MSE	Mean Squared Error
MSPS	Mega Sample Per Second
PHASE_SEG1	Phase Buffer Segment 1
PHASE_SEG2	Phase Buffer Segment 2
PMOD	Peripheral Module Interface
PROP_SEG	Propagation Segment
REC	Receive Error Count
ROC	Receiver Operating Characteristic
RTR	Remote Transmission Request
SoC	System On Chip
SOF	Start Of Frame
SPI	Serial Peripheral Interface
SRR	Substitute Remote Request
SVM	Support Vector Machine
SYNC_SEG	Synchronization Segment
TEC	Transmit Error Count
USB	Universal Serial Bus
VHDL	Very High Speed Integrated-Circuit Hardware Description Language
WiFi	Wireless Fidelity

1. GİRİŞ

Günümüzde, kara ve hava araçlarında veya uydu gibi yüksek teknoloji barındıran ürünlerde birçok elektronik alt birimler bulunmaktadır. Bu alt birimler farklı kablolu iletişim protokolleri aracılığı ile haberleşmektedir. Günümüzde bu haberleşme protokollerinden en yaygın kullanılanlardan biri CAN (Controller Area Network) protokolüdür. CAN protokolü günümüzde özellikle otomotiv sektörü başta olmak üzere aviyonik ve uzay sistemlerinde de yaygın olarak kullanılmaktadır. CAN protokolü özellikle hata oranı kritik sistemlerde tercih edilmektedir. Tercih edilmesinin en önemli sebeplerinden bazıları bu protokolün düşük hata yapma oranına sahip olması ve mesajların birden çok noktaya gönderilebilir olmasından kaynaklıdır.

CAN protokolü Bosch olarak bilinen Robert Bosch GmbH firması tarafından 1983 yılında geliştirilmeye başlanmıştır. 1986 yılında resmi olarak yayınlamıştır. 1991 yılında CAN 2.0 [1] protokolünü yayınlamıştır. Bu protokol part A ve part B olmak üzere iki kısımdan oluşmaktadır. Günümüzde en yaygın olarak kullanılan standart CAN 2.0B'dir. Bosh 2012 yılında CAN FD (Flexible Datarate) adını verdikleri protokolü yayınlandı. Bu protokolde CAN 2.0'dan farklı olarak paket yapısında değişikliğe gidilmiş ve veri hızında artış yapılmıştır; ancak bu protokol geriye dönük olarak CAN 2.0 protokolünü de desteklemektedir. Bu tez de CAN 2.0B protokolü incelenmiş ve geliştirmeler bu protokol üzerinde yapılmıştır.

Teknolojinin ilerlemesiyle otomobiller birçok sensör ve cihazın kablolu ve kablosuz iletişimini sağlayan kompleks yapılar haline gelmiştir. Günümüzde hemen her araçta, araç fonksiyonlarını yerine getiren Elektronik Konrol Birimleri (ECU) bulunmaktadır. Bu birimlerde kullanıcılar için bulunan, Hücresel 3G/4G(Cellular), WiFi, Bluetooth, USB gibi kablolu ve kablosuz iletişim teknolojileri bulunmaktadır. Buna ek olarak ECU'ların birbirleriyle iletişimlerinde kullanılan, CAN, LIN (Local Interconnect Network), MOST (Media Oriented System Transport), FlexRay gibi iletişim protokolleri de bulunmaktadır. Elektronik Kontrol Birimlerinin WiFi, Bluetooth gibi protokoller ile dış dünya birimleri ile iletişime geçmesi sistemin güvenlik olarak zafiyetini arttırmaktadır. Elektronik kontrol birimlerinin kontrollünün dışarıdan yetkisiz ele geçirilmesi durumunda (Hacklenmesi), aracın hareket ve diğer güvenlik fonksiyonlarına da erişimini mümkün kılmaktadır.

Yapılan son araştırmalarda, araçlarda güvenlik açıkları ve zafiyetleri keşfedilmiştir. Araştırmacılar, araçların motor kontrol ünitesi, ısıtma/soğutma, kapı kilitleri, frenler vb.

birçok güvenlik kritik fonksiyonlarına dışarıdan kontrol sağlamışlarıdır [2]. Aynı araştırmacılar, araç telemetrilerinden tersine mühendislik yapılarak uzaktan erişimine olanak sağlayan saldırılar yapmayı başarmışlardır [3]. Valasek ve Miller [4] Jeep Cheroke marka aracın eğlence ve navigasyon sistemi üzerinden sızarak aracın hareket, fren, radyo, ısıtma/soğutma istemini uzaktan kontrol edebildiklerini göstermişlerdir.

Araçların Elektronik Kontrol Birimlerinin güvenlik açıklarının ve zafiyetlerinin ortaya konmasından sonra CAN protokolünün önemi daha da artmıştır. Bunun sebebi ise araçların iç haberleşmesinin büyük bir çoğunluğunun bu protokol üzerinden gerçekleşmesidir. Otonom sürüş teknolojilerinin yaygınlaşması da araçların ve dolayısıyla CAN protokolünün dışarıdan gerçekleştirilebilecek saldırılara karşı güvenirliğinin sorgulanmasına sebep olmuştur.

CAN protokolü yapısı sebebiyle tercih sebebi olmakla beraber yine yapısı sebebiyle bazı güvenlik açıklarını da beraberinde getirmiştir. Modern araçlarda yaklaşık olarak 20 - 100 arasında Elektronik Kontrol Birimi (ECU) bulunmaktadır [5]. Bu ECU'lar araçların motor kontrolünden frenleme sistemine, müzik sisteminden havalandırma sistemine kadar bütün araç fonksiyonlarını kontrol etmektedirler. ECU'lar birbirleriyle CAN protokolü ile haberleşirler. CAN protokolü yakalanamayan bit hata oranı (BER) [1] açısından güvenilirdir; ancak dışarından gelebilecek saldırılara açık bir protokoldür. Bu nedenle CAN protokolünün güvenlik açısından incelenmesi bir araştırma konusu haline gelmiştir.

CAN protokolünün yapısı gereği sahip olduğu güvenlik açıklarının araştırılması sonucunda, bu açıkların kapatılması için birçok yöntem ortaya konulmuştur. Bu yöntemlerden biri IDS (Intrusion Detection System) olarak tanımlanan yetkisiz girişleri algılayan sistemlerdir.

1.1. IDS Teknolojileri

Genel olarak, IDS veri yolundaki veri trafiğini gözlemleyen ve sistemde beklenmeyen bir veri veya durum oluştuğunda uyarı veren sistemlerdir [6]. Bu beklenmeyen durumlar, dışardan veri yoluna izinsiz giriş yapmaya çalışan saldırganın sistemde ürettiği durumlar veya ürettiği sahte verilerdir. Veri yoluna yapılan ihlaller sistemin içerisinden olabileceği gibi dış dünyaya bir veri yolu ile bağlı olan sisteme uzaktan da olabilmektedir. IDS'ler pasif veya aktif olabilirler. Pasif IDS'ler sadece saldırıları algılayıp alarm veren sistemlerdir. Aktif IDS'ler ise saldırı tespiti yapmasından sonra saldırıyı durdurma aksiyonları alan sistemlerdir. Temel olarak bir IDS mimarisi sensörlerden, algılama

biriminden ve raporlama biriminden oluşur. Sensörler veri yolu üzerinde bulunabileceği gibi veri yolu üzerindeki düğümlerde de bulunabilir. IDS metodolojileri imza tabanlı, anomali tabanlı veya spesifikasyon tabanlı olarak kategorize edilmektedir.

IDS'in otomotiv sektörüne yönelik uygulaması ilk defa Hope et al. tarafından ortaya çıkarılmıştır [7]. Literatürde farklı metotlarla tasarlanmış IDS sistemleri bulunmaktadır. Bu metotlar, frekans tabanlı, makine öğrenmesi tabanlı, istatiksel/olasılık tabanlı veya bu metotların farklı kombinasyonları olacak şekilde hibrit metotlar olarak özetlenebilir [8]. IDS metotları sayesinde CAN arayüzüne yetkisiz veri girişi yapılması durumunda bu verilerin filtrelenip sistemin normal şekilde çalışmasına devam etmesi veya sistemin uyarı vererek sistemin doğru şekilde çalışmasını sağlayacak karşı önlemlerin devreye alınması hedeflenir.

1.2. Tezin Amacı

IDS'ler basit yapıları ve efektif olarak saldırıları tespit etme kabiliyetleri sebebiyle önemli bir araştırma konusu haline gelmiştir. Buna ek olarak, IDS'lerin şifreleme, yetkilendirme gibi yetkisiz girişleri engelleyen sistemler ile beraber çalışabilmektedir. IDS'ler için kritik olan iki temel performans kriteri bulunmaktadır. Bunlar saldırıların eksiksiz olarak tespit edilmesi bir diğeri ise sistemdeki normal bir durumu hata olarak algılanmamasıdır. Literatürde farklı metotlar ile gerçekleştirilen IDS'ler bu iki performans kriterine göre karşılaştırılmaktadır.

Bu tez kapsamında CAN veri yolu için bir IDS sistemi geliştirilmiştir. Bu sistemin gerçekçi olması açısından CAN veri yolunun fiziksel olarak benzetimi yapılmıştır. Geliştirilecek IDS, ECU'ların veri yolunda ürettiği sinyallerin farklılıkları üzerine geliştirilmiştir. Bu farklılıklar literatürde sinyal parmak izi olarak tanımlanmıştır. ECU'ların sinyal parmak izlerinin alınması için FPGA'de bir CAN 2.0B kontrolcüsü (kontrolcü gerçeklemesi dahil) içeren bir ECU gerçeklenmiştir. Bu ECU CAN sinyalinin dijital yapısına ait özellikleri çıkarmakla beraber diferansiyel olarak gönderilen CAN-H ve CAN-L sinyalinin, tez kapsamında belirlenmiş ilgili kısımlarının örneklemesini yapmaktadır. Geliştirilen sistem ile ECU'nun aldığı veriler otomatik olarak yazılım ortamına aktarılmıştır. Bu veriler tez kapsamında geliştirilen IDS sisteminin algılama birimini gerçeklemek için kullanılmıştır. IDS'in algılama birimi MLP ve LSTM tabanlı yapay siniri ağı algoritmaları kullanılarak gerçeklenmiştir. Bu algoritmaların yapısı, sistemin çalışma performansına olan etkisi değerlendirilecektir.

1.3. Tezin Yapısı

Bu tez giriş bölümünü takip eden 5 bölümden oluşmaktadır

İkinci bölümde IDS mimarileri ile ilgili literatürde bulunan çalışmalar açıklanmıştır. CAN protokolünün güvenlik açıklarının literatürdeki IDS teknolojileri kullanılarak kapatılmasına olanak sağlayan metotlar açıklanmıştır.

Üçüncü bölümde CAN protokolünün mesaj yapısı, bit zamanlaması, hata durumlarının yönetimi gibi özellikleri ve CAN protokolünün güvenlik açıkları anlatılmıştır.

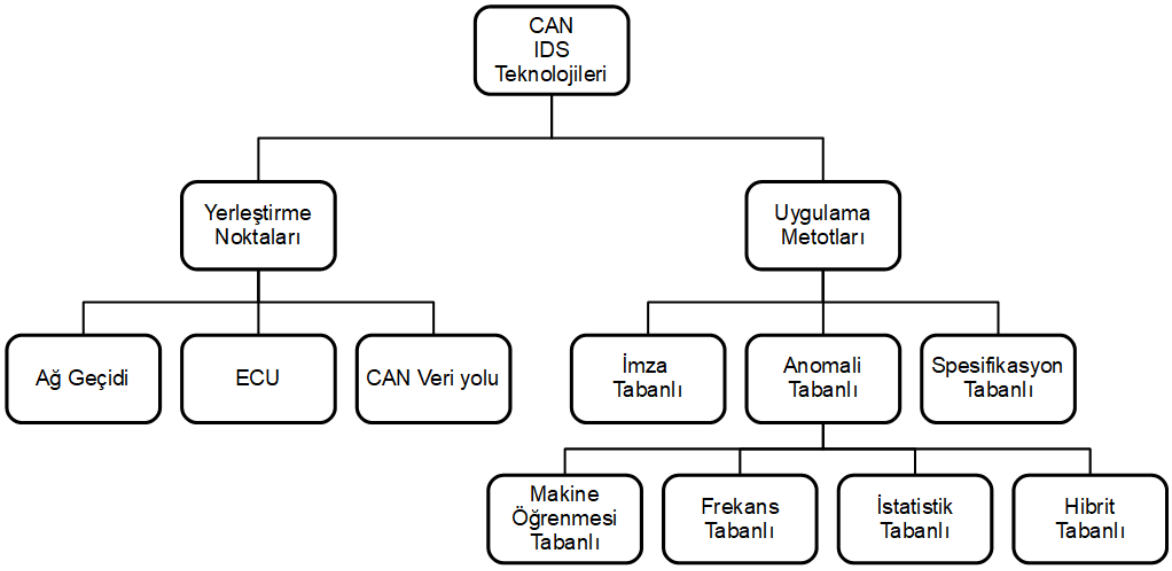
Dördüncü bölümde tez kapsamında geliştirilen ECU'nun tasarımı, ayrıca yazılım ortamında gerçeklenmiş yapay sinir ağlarının tasarımı anlatılmıştır. Bu tasarımda kullanılmış veya tasarlanmış donanımlar ve yazılımlar verilmiştir. Sisteme ait simülasyonlar ve gerçek sistem uygulamaları anlatılmıştır.

Beşinci bölümde IDS sisteminin algılama birimi için geliştirilen yapay sinir ağlarının sistem performansına etkisi açıklanmıştır. Ayrıca, farklı parametrelere göre tasarlanmış yapay sinir ağlarının sistem üzerindeki performans bulguları analiz edilmiştir. Bu parametrelerin sistem üzerindeki etkisi açıklanmıştır.

Altıncı bölümde geliştirilmiş IDS metodunun elde edilmiş en iyi performans sonuçları verilmiştir. Buna ek olarak tez kapsamında düşünülmüş; ancak tez kapsamına dahil edilmemiş ve gelecekte yapılmasının uygun olduğu düşünülen çalışmalar anlatılmıştır.

2. LİTERATÜR TARAMASI

IDS uygulama konsepti günümüz bilgi teknolojilerinde ve bilgisayar ağlarında yetkisiz yapılan girişleri, girişimleri tespit etmek için kullanılmaktadır. Otomotiv sektöründe ise IDS uygulaması ilk defa Hope et. al. tarafından ortaya atılmıştır [7]. Bu bölümde, literatürde kullanılan algılama metodolojilerine göre IDS teknolojileri özetlenmiştir. Genel olarak IDS yöntemleri, işaret (signature), anomali, spesifikasyon ve hibrit tabanlı olmak üzere dört başlıkta incelenmiştir [8]. CAN veri yolu IDS genel taksonomisi Şekil 2.1'de verilmiştir.



Şekil 2.1 CAN Veri Yolu IDS Taksonomisi [8]

IDS araç sistemlerinde farklı noktalara yerleştirilebilmektedir. IDS'lerin araç içerisindeki her bir ECU'ya yerleştirilmesi araç içerisindeki tüm haberleşmenin izlenebilmesine olanak sağlanmaktadır. Bu sayede sisteme gerçekleştirilen izinsiz bir girişin çalışma esnasında rahatlıkla algılanabilmesini sağlamaktadır. Ayrıca IDS sisteminin CAN veri yoluna veya ağ geçidi (gateway) birimlerine yerleştirilmesi araç içerisinde olan iletişimin incelenmesine olanak sağlamaktadır. Bu tip IDS'ler ağ tabanlı (network-based) IDS olarak tanımlanmıştır.

Otomotiv sektöründeki IDS teknolojileri, uygulamada kullanılan algılama metotlarına göre sınıflandırılmıştır. Bu algılama metotları, işaret (signature), anomali, spesifikasyon ve hibrit tabanlı olmak üzere 4 farklı ana başlıkta incelenebilir. Bu metotlardan

en yaygın olanı ise anomali tabanlı IDS sistemleridir. Bu bölümde literatürdeki farklı metotlar ile öne sürülmüş veya gerçeklenmiş IDS sistemleri ve çalışmalar anlatılmıştır.

Müter et al. çalışmasında CAN veri yolunda oluşabilecek anormallikleri algılayan farklı sensör tipleri sunmuşlarıdır [9]. Çalışmaya göre anomali algılama sistemlerindeki en büyük zorluğun çok fazla yanlış pozitif uyarı üretmeden algılama işleminin gerçekleştirilmesidir. Müter et al. çalışmasında farklı tip anomali algılayan sensörlerin, diğer anomali tabanlı IDS sistemlerine göre daha avantajlı olduğunu belirtmiştir. Bu avantaj sistemin hiç yanlış pozitif uyarı üretmemesidir. Sistemin hatalı pozitif veri üretmemesinin sebebi ise, sensörlerin kullandığı verilerin kesin ve güvenilir veriler olmasından kaynaklı olduğu belirtilmiştir. Bu güvenilir verilerin CAN veri yolunun spesifikasyonundan, cihazların önceden belirlenmiş ortaklaşa çalışma davranışlarından (örn. ECU birimlerinin mesaj duplikasyon tabloları), kendini tekrar eden veya gereksiz verilerden (redundant) veya bunların kombinasyonlarından çıkarıldığı anlatılmıştır. Bu nedenle sistemde bir anomali algılandığında, sistemde gerçek anlamda bir anormallik durumunun olduğu anlatılmıştır ancak bu sistemin de tüm saldırıları algılayamama ihtimalinin bulunduğu belirtilmiştir. Sistemin algıladığı bir anomalinin, bir donanım hatası veya bir saldırı kaynaklı olup olmadığı anlaşılamamaktadır fakat bu problemin diğer tüm IDS'ler için geçerli olduğu ve sistemin uygulanabilirliğini etkilemediği belirtilmiştir. Çalışmada tanımlanmış sensörleri, CAN veri yolu için tanımlandığı ancak sistemin diğer veri yolları için de uygulanabilir olduğu belirtilmiştir. Çalışmada verilmiş olan sensörler şu şekildedir:

Sensör-1 Formalite Sensörü: Bu sensör CAN protokolünde gönderilen paketlerin temel yapısını kontrol etmektedir. Bu kısımlar, mesaj uzunluğu, paket başlığı, sabit sonlayıcı noktaları, sağlama toplamı (checksum) vb. gibi kısımlardır.

Sensör-2 Lokasyon Sensörü: Otomotiv veri yolu genel olarak alt veri yollarına (sub-network) ayrılmıştır. Bir alt veri yolunda gönderilen bir paketin, diğer bir alt veri yoluna kabul edilmemektedir. Bu nedenle yapısal olarak doğru olan bir CAN paketinin olmaması gereken bir alt veri yolunda bulunması bir anomalidir. Bu sensör farklı alt veri yollarındaki izin verilmiş CAN paketlerini inceler, izinsiz bir paket var ise uyarı verir.

Sensör-3 Aralık Sensörü: Bu sensör veri yolunda gönderilen, verilerin doğru sınırlar içerisinde olduğunu kontrol eder. Eğer ilgili veri sayısal olarak doğru ama aracın fiziksel durumuna göre hatalı ise anomali durumu algılanır. Örneğin 200 km/s en fazla hız yapabilen bir aracın hızının 300 km/s olarak gönderilmesi bir anomalidir.

Sensör-4 Frekans Sensörü: Otomotiv CAN veri yolunda genel olarak paketler sabit zaman aralıklarında gönderilir. Bu sensör ise paketlerin gönderilme sıkılığını kontrol eder. Paketlerin gelme sıkılığı önceden tanımlanmış alt ve üst zaman sınırları dışında ise anomali olarak kabul edilir.

Sensör-5 Korelasyon Sensörü: Birden fazla alt veri yolları içeren bir araç veri yolunda bazı paketler, birden fazla alt veri yoluna iletilirler. Hangi verilerin hangi alt veri yollarına iletileceği araç üreticileri tarafından önceden belirlenmiştir. Korelasyon sensörü ise bu paketlerin, iletildiği alt veri yollarını da kontrol ederek verinin doğru alt veri yollarında da olduğunu doğrular. Ayrıca bu sensör bir anomali algıladığında da anomalinin hangi alt veri yolunda olduğunu da gösterebilmektedir.

Sensör-6 Protokol Sensörü: Araç içerisindeki ECU'lar genel olarak meydan okuma-karşılık verme (challange-response) benzeri iletişim protokollerine göre çalışırlar. Bu çalışma prensibi aracın başlangıçta hata teşhisi veya elektronik anahtar paylaşımı gibi amaçlar için kullanılmaktadır. Protokol sensörü ise bu iletişim protokollerindeki anomalileri algılamak için kullanılacaktır. Örneğin, sistemdeki mesajlaşma yapısının sırasında oluşan bir değişiklik sistemde anomali olarak algılanacaktır.

S-7 Olasılık(Plausibility) Sensörü: Bu sensör araç içerisinde gönderilen verilerin makul değerler arasında olup olmadığını kontrol eder. Örneğin, 20 km/h hızla bir aracın bir anda 200 km/h hıza çıkması imkansızdır.

Sensör-8 Tutarlılık Sensörü: Bu sensör birden fazla araç içi sensörden alınan veririnin birbiri ile tutarlılığını kontrol eden sensör birimidir. Örneğin, araçlardaki teker rotasyon sensörünün aracın hareket ettiğine işaret ederken, GPS'ten alınan verinin aracın durduğuna işaret etmesi gibi bir durumda bu sensör bu durumu anomali olarak algılayacaktır.

Mütter et al. çalışmada sensörlerin uygulanabilirliği konusunda altı farklı kriter belirlemişlerdir [9]. Her bir sensörün çalışabildiği koşullar farklı olmaktadır. Örneğin, bazı sensörlerin sadece bir paket ile çalışabilirken, bazı sensörlerin birden fazla paket ile çalışabilmektedir. Çalışma kapsamında uygulanabilirlik kriterleri şu şekilde verilmiştir.

Kriter-1 Spesifikasyon Tabanlı: Araç içerisindeki CAN veri yolları önceden ayarlanmış protokollere veya parametrelere göre çalışmaktadır. Önceden ayarlanmış bu parametrelere uymayan hatalar, spesifikasyona göre anormal durumlardır. Çalışmada verilen bazı sensörler bu durumları algılamaya yöneliktir.

Kriter-2 Paket Sayısı: Bu kriter, sensörün çalışabilmesi için gereken paket sayısını ifade eder. Formalite sensörü için bu değer bir iken, frekans sensörü için en az 2 en fazla *N* olacak şekilde değişiklik gösterebilir.

Kriter-3 Veri yolu sayısı: Bu kriter, sensörün çalışabilmesi için üzerinde çalışması gereken farklı veri yolu sayısını belirtmektedir. Örneğin, tutarlılık sensörü birden fazla farklı sensörden gelen verilerinin birbiri ile uyumlu olduğunu kontrol edebilmesi için araç sistemindeki birden fazla veri yoluna erişimi bulunması gerekmektedir.

Kriter-4 Farklı Mesaj Tipleri: Bu kriter, sensörün çalışabilmesi için birden fazla farklı mesaj tipine ihtiyaç duyulup duyulmadığını tanımlar.

Kriter-5 Veri Kontrolü: Bu kriter, sensörün çalışabilmesi için paket içeriğinin incelenmesinin gerekli olup olmadığını gösterir. Bu kriterin uygulanabilir olması için sistemdeki verilerin şifrelenmemiş olması gerekmektedir.

Kriter-6 Semantik Tabanlı: Bu kriter, sensörün çalışabilmesi için gönderilen verilerin anlamsal olarak incelenmesi gerektiğini belirtmektedir. Bu kriterin uygulanabilir olması için öncelikle veri kontrolünün yapılması gerekmektedir; ancak bu iki kriter birbirinden ayrı olarak tanımlanmıştır. Veri kontrolü kriterinden farklı olarak verinin anlamlı ve diğer veriler ile tutarlı olmasına bakılmaktadır.

Studnia et al. [10] tarafından yapılan çalışmada günümüz araçlarında ECU'ların sayılarının ve fonksiyonlarının artışına bağlı olarak araç yazılımlarının da geliştiğinden bahsedilmiştir. Bu artışa bağlı olarak araçların sahip olduğu USB, Bluetooth ve WiFi gibi dış bağlantı arayüzleri de arttığı ve bu durumun araçların siber saldırılar için daha uygun hale gelmesine sebep olduğu söylenmiştir. Araçlara yapılabilecek bir saldırı sonucunda navigasyon geçmişi gibi kişisel gizliliğin ihlali, bir fonksiyonun çalışmaması problemi ve araç içindeki insanların can güvenliğini tehdit edebildiğinden bahsedilmiştir. Geçmiş araçların sistemlerinin dışarıya kapalı olduğu için araç sistemlerinin güvenliğine daha az önem verilmiş fakat otomotiv endüstrisinin otonom sürüşe doğru ilerlemesi sebebiyle ECU'lerin ve araç içi haberleşmenin güvenliğine ekstra önem verilmesi gerektiğini öne sürmüştür. Ortalama bir araç ömrünün 20 sene olduğu düşünüldüğü takdirde güvenlik duvarlarının ve tanımlanan kuralların zamanla geçerliliğini yitirebileceği belirtilmiştir. Bu sebeple saldırı tespit sistemlerinin araçlara entegre edilmesi gerektiği öne sürülmüştür.

Yapılan çalışma kapsamında önerilen IDS sisteminin amacı, yapılmakta olan saldırıların ağın trafiğinden toplanan veriler ile tespiti amaçlanmıştır [10]. Bu çalışmada saldıranın sisteme mesaj gönderme yeteneğine ulaştığı varsayılmıştır. Başka bir deyişle, ihlal gerçekleştikten sonra tespit edilmesi üzerine yapılmış ve bu ihlalin sebepleri üzerinde durulmamıştır. Bu çalışma kapsamında tespit edilmek istenilen saldırılar şu şekilde gruplandırılmıştır; spesifikasyonlara uymayan çerçeveler (bilinmeyen tanımlayıcı, yanlış boyutta veri alanı, yanlış CRC gibi), sistem tarafından periyodik olarak tanımlı çerçeveler gönderilirken çelişkili mesajların gönderimi (ışıkların açılması periyodik olarak gönderilirken kapatılması için gelen bir mesaj gibi), tanımlı çerçevelerin değiştirilerek zararlı olanlar ile güncellenmesi ve bir çerçevenin periyodik olarak güncellenmiş veri ile gönderilmesi gerekirken senkron olmayan bir şekilde çerçevelerin gönderilmesidir. Bu saldırılardan sepsifikasyonlara uymayan çerçevelerin tespiti ve sistem tarafından periyodik olarak tanımlı çerçeveler gönderilirken çelişkili mesajların gönderiminin tek bir çerçevenin analizi ile çözülebildiği fakat diğer iki saldırı için tek çerçevenin yeterli olmayacağı ve bu saldırıları tespit etmenin anahtarı, izlenen çerçeveler hakkındaki bağlamsal bilgilerin korelasyonuna dayandığı öne sürülmüştür. Bu çalışmada saldırıların tespiti için kullanılan metot modellerden türetilen yasaklar dizisine dayandırılmasıdır. Bu metodun avantajı ilk senkronizasyondan sonra analizin çalışmaya devam edebilmesi olarak iddia edilmiştir. İzlenmekte olan çerçeveler için bir önceki mesajlardan aracın durumu ile ilgili bilgilerin toplanması gerektiği ve bu bilgiler ile gelen çerçevenin güncel durum ile kıyaslanarak uygunluğunun değerlendirilebileceği söylenmiştir. Bu kapsamda iki seviyeli bir tespit yapılması gerektiği ve ilk seviyede gelen çerçevenin tanımlı spesifikasyonlar ile uyumunun kontrol edilmesi gerektiği, ikinci seviyede ise sistemin güncel durumu ile gelen çerçevenin arasındaki uyumun kontrol edilmesi gerektiği söylenmiştir. Yapılan çalışmada IDS sisteminin izlenmek istenen tüm ağlara bağlı olması gerektiği söylenmiştir. Araçlarda iki veya daha fazla ECU'nun birbirlerine bağlandığı ve aralarında başka bir ECU'nun ağ geçidi görevi gördüğü sistemlerde, ağ geçidinin olası bir saldırıda ana hedef olabileceği söylenmiştir. Bu sebeple, IDS sisteminin hali hazırdaki bir ağ geçidine eklenebileceği önerilmiştir. IDS sisteminin konumu için lokasyonlar önerilmiştir.

Bu çalışmada her bir ECU için "formal language theory" kullanılmıştır [10]. Her bir ECU için tanımlanan dilde(language) semboller, incelenen sistemle ilgili olanlarla sınırlı, ağ üzerine yayılan veya ağdan alınan farklı taleplerden ve sistemin normal bir durumda çalışırken içermesi gereken tüm sembollerden oluşmaktadır. Daha sonra bu dili açıklayan

bir FSA (finite state automation) tanımlanmış ve bu tanımlama ağdan aldığı şekliyle ECU'nun davranışını temsil etmektedir.

Bu çalışma kapsamında dilde tanımlı olmayan bir diziye bağlı bir çerçevenin tespiti amaçlanmaktadır. Tanımlı olmayan bu dizi belirli bir noktaya kadar tanımlı ve devamında beklenmeyen bir sembolün görüldüğü bir dizi olarak tanımlanabilir. Bu duruma örnek olarak ışıkların kontrolü için otomatik mod aktifken, ışık sensörünün çevreyi gece olarak algılarken ışık kontrolcüsünün ışıkları kapalı tuttuğu durum verilmiştir.

Bu çalışmada tanımlanan yaklaşımın doğrulanabilmesi için A ve B olarak iki sistem tanımlanmıştır. İlk sistem 21 durum içerirken, ikinci sistem 108 durum içermektedir. Sistemlerin detayları gizlilik sebebiyle verilmemiştir. Çalışmanın denendiği CAN veri yolunda ortalama her saniye 1514 çerçeve bulunmaktadır. Bu algoritmanın sonuçlarına bakıldığı zaman sistemin tanımlanan dilde olmayan bir mesaj yapısını başarıyla tespit ettiği görülmüştür.

Bu çalışmada kullanılan otomat teorisi ağın ve ECU'ların spesifikasyonlarına dayanmaktadır; fakat gerçek sistemlerin bazı kısımları bu spesifikasyonlara uymak zorunda değildir ve bazı farklılıklar bulunabilir. Farklı durumların gerçekleşmesi halinde tasarlanan bu sistem bir saldırı olduğunu düşünebilecektir. Bu istenmeyen durumun önüne geçebilmek için bir makine öğrenmesi tanımlanabileceği önerilmektedir.

Bu duruma ek olarak, başlangıç çerçevelerinin unutulduğu durumda bazı saldıların asla tespit edilemeyeceği söylenmiştir. Bu durumda, saldırgan IDS sistemini düzenli olarak yeniden başlatarak saldırısını gizleyebilecektir.

Son olarak, bu çalışmada sadece basit FSA'lar düşünülmüştür. Örneğin, zaman bilgisi tanımlanan otomat teorisinde hesaba katılmamıştır. Bunun yerine tespitin ilk adımını oluşturan tanımlanan kurallar ile periyodik çerçeveler arasındaki zamanlama dahil edilmiştir. Çalışmada, saldırıların çerçevelerin sırasının değil çerçeveler arası zamanın önemli olduğu vurgulanmıştır.

Larson et al. tarafından yapılan çalışmada araç içi ağlar içinde saldırı tespitinin uygulanabilirliği değerlendirilmiştir [11]. Bu kapsamda CAN 2.0 ve iletişimle ilgili bilgileri çıkarıp ECU'ların protokollerini ve iletişim davranışlarını tanımlamak için CANopen uygulama katmanı taslak standardı 3.01 değerlendirilmiştir. Bu çalışmada tasarlanan saldırı tespit sisteminin uygulanabilirliğinin değerlendirilmesi için bir dizi saldırı eylemi

kullanılmıştır. Bu çalışmada saldırı tespiti için spesifikasyon tabanlı bir yaklaşım kullanılmıştır [11]. Yapılan çalışmada spesifikasyon tabanlı sistemlerin araç içi ağda kullanımının CAN ve CANopen protokollerinin yaygın kullanımı sebebiyle modifikasyon ve eklemelerin nadir olması; araç içi ortamının yüksek oranda spesifik görevlere atanması ve dinamik bir davranışının olmaması sebebiyle spesifikasyon tabanlı tespitin başarılı olması beklenmiştir.

Bu çalışmada saldırı tespit için geliştirilen tasarım 5 farklı ECU'dan oluşmaktadır. Bu ECU'lar, kaynak ECU, hedef ECU, ağ geçidi ECU, ECU1 ve ECU2 olarak tanımlanmıştır. Tanımlı mesajlar tüm ağ üzerinde hareket ederek kaynaktan hedefe ulaşmaktadır. Ağ geçidi ECU'ları bir röle gibi çalışmakta ve mesajları sadece ileri yönde iletmektedir. Yapılan çalışmada ağ geçidi ECU'larının kritik öneme sahip olduğu ve bunların deşifre olması durumunda saldırı ihtimallerinin ve çeşitlerinin artabileceği söylenmiştir. Bu sebeple, ağ geçidi ECU'larının korumasına ekstra önem verilmesi gerektiğinden bahsedilmiştir.

Larson et. al. yapılan çalışmada saldırganın gerçekleştirebileceği farklı tip saldırılara değinmiştir [11]. Bu saldırı tipleri paket düşürme, paket seli (flood), paketin değiştirilmesi, paketin okunması, paket tekrarı ve kandırmadır (spoof). Bu ataklar ele geçirilen birimlere göre değişiklik göstermektedir.

Çalışmada kullanılan modelde algılayıcı birimiler bulunmaktadır. Bu sisteme göre algılayıcıların, farklı kaynaklardan gerçekleştirilen saldırıları algılama kabiliyeti bulunmaktadır. Çalışmada algılayıcı performansları verilmiştir. Verilen sonuçlara göre algılayıcıların tekil olarak algılayabildikleri saldırılar bulunmakla beraber, algılayıcıların birlikte çalışarak da algılayabildiği hatalar gösterilmiştir.

Song et al. çalışmasında düşük karmaşıklığa sahip olduğunu öne sürdükleri bir IDS geliştirmişlerdir [12]. Bu sistemin CAN veri yolundaki işaret ve anomalileri algılayan hibrit bir sistem olduğu vurgulanmıştır. Araçlarda bilinen saldırı işaretlerinin fazla miktarda olmadığı bu nedenle bu sistemin yüksek hesaplama gücüne ihtiyaç duymadığı belirtilmiştir. Sistem genel olarak, CAN veri yolundaki paket enjekte etme saldırılarını algılaması için tasarlanmıştır. Bu saldırıları algılamak için veri yolundaki trafik paketlerin frekansına göre analiz edilmiştir. Çalışmada normal ve paket enjekte edilmiş veri yolu arasındaki fark gösterilmiştir. Genel olarak ECU'ların mesajları periyodik olarak ürettikleri belirtilmiştir. Saldırıyı gerçekleştiren tarafın sistemi kontrol etmek için araya paket enjekte etmesi

durumunda bu periyodikliğin bozulacağı anlatılmıştır. Bu paketlerin frekansının incelenerek sisteme paket enjekte edildiğinin anlaşabileceği vurgulanmıştır. Bu metodun efektif bir yöntem olduğu anlatılmış; ancak düşük frekanslı paketler için enjekte edilen paket miktarının hesaplanmasının yavaş olacağı da belirtilmiştir. Ayrıca, bu istatiksel metodun düşük verilerde hatalı sonuçlar verebileceğini bunun yanında yüksek miktarda veri toplamak için yavaş sonuç verme durumunun gözlenebileceği anlatılmıştır. Song et al. Çalışmada saldırı algılama performansını düşürmeden, sonuç verme hızının olabildiğince yüksek olması için işlemleri basitleştirdiklerini söylemişlerdir [12].

Song et al. Geliştirdikleri sistemi test etmek için gerçek bir araçtan 40 dakika normal bir sürüş esnasında veri toplamışlarıdır [12]. Bu verilere rastgele zamanlarda 5-10 saniye süresince paket enjekte etme saldırısı gerçekleştirmişlerdir. Sisteme iki, beş, on kat daha hızlı saldırlar gerçekleştirmişlerdir. Sistemlerinin bir veya birden fazla CAN ID'leri için hata algılama performansını ölçmüşlerdir.

Song et al. sonuç olarak geliştirdikleri sistemin gerçeklenmesi basit ve hızlı tepki veren bir sistem olduğunu vurgulamıştır [12]. Sistemin milisaniyeler mertebesinde saldırıları algılayabildiği ve hiç yanlış pozitif bildirim olamadan saldırı tespit oranının %100 olduğunu açıklamıştır.

Chris Valasek ve Charlie Miller tarafından yapılan çalışmada benzer bir frekans tabanlı IDS sistemi geliştirmişlerdir [13]. Çalışmalarında Ford marka bir araçtan alınan veriler ile araç içerisindeki bir CAN veri yolundan gönderilen ve alınan paketlerin frekans olarak öngörülebilir olduğunu gözlemlemişlerdir. Bu nedenle yapılan saldırılarda paketlerin frekanslarının belirgin bir ölçüde değiştiğini ve bu değişimin algılanabildiğini vurgulamışlardır.

Gmiden et al. frekans tabanlı bir IDS geliştirmişlerdir [14]. Diğer frekans tabanlı sistemlerden farklı olarak, paketleri toplu olarak değil aynı CAN ID olan paketlerin bir öncekine göre geliş süresi hesaplanmıştır. Tasarlanan bu IDS sisteminin şifreleme ve yetkilendirme gibi veri yolunu saldırılara karşı koruyan diğer metotlar ile uyumlu olduğunu belirtmişlerdir. Ayrıca, sistemde herhangi bir değişiklik yapılmasına gerek olmadığından ucuz ve etkili biri yöntem olduğunu vurgulamışlardır.

Moore et al. da çalışmasında frekans tabanlı bir IDS önermiştir. Bu IDS sistemin diğerlerinden farklı olarak; paketin bir önce gelen paketin süresine göre Markov zinciri metodu ile istatiksel bir yöntem kullanılmıştır [15]. Sistemin doğru pozitif algılama oranının

0.9998, yanlış pozitif oranının 0.00294 ve toplam hata oranının ise 0.00298 olduğu belirtilmiştir.

Safa Boumiza ve Rafik Braham tarafından yapılan çalışmanın amacı CAN veri yolu için mevcut sistemlerin sınırlamalarını aşan iyi bir algılama performansı ile frekans-görünüm ve veri içerik değişikliklerini tespit edebilen kapsamlı bir saldırı sistemi geliştirilmesidir [16]. Bu çalışma kapsamında tasarlanan modelde saldırgan, araç içi ağa erişmeyi başardığı ve iletilen tüm mesajları görebildiği varsayılmıştır. Tasarlanan sistemdeki saldırı detektörü CAN veri yolundaki, ID ve veri alanı verilerine dayanmaktadır. Verilerden, veri içeriğindeki değişikliği karakterize edebilen bir mod özelliği çıkarılmıştır. Frekans değişiklikleri için, tanımlanmış bir zaman penceresinde her ID'nin oluşum zamanı hesaplanmıştır. Her bir ID için, sistem davranışını karakterize etmek için mod ve frekans özellikleri kullanılmıştır.

Bu sistemde "Data Pretreatment" aşamasında veriler, K-means kümeleme algoritması kullanarak ID alanına göre alt kümelere bölünmektedir. "Features extraction" aşamasında her bir ID için, belirli bir ID'nin işlevini karakterize eden mod ve frekans özellikleri çıkarılmaktadır. "Classification Phase" aşamasında modeli eğitmek ve test etmek için Çok Katmanlı Algılayıcı (MLP) sinir ağı kullanılmıştır. Geri Yayılım (Back Propagation), öğrenme algoritması ile eğitilmiş üç katmanlı bir ileri beslemeli sinir ağıdır. Her ID için sistem çok sayıda paketi işlemektedir. Her bir detektördeki sinir ağı girdileri, daha önce çıkarılan iki özellik ve bir saldırının varlığını veya olmadığını gösteren ikili bir çıktıdır. Her bir ID için sistem tarafından verilen kararı sunan genel bir puan elde edilmektedir. Son aşamada ise önceki ID kararlarının birleştirilmesiyle oluşturulan "threshold" değeri aşıldığında bir alarm üretilmektedir.

Yapılan bu çalışmada geri yayılım algoritması ile eğitilmiş ve üç katmandan oluşan bir ağ kullanılmıştır. Giriş sinyalleri, mod ve frekans özellikleri olan her bir ID veri paketi için çıkarılan özelliklerdir. Her katman bir önceki katmana tamamen bağlıdır.

Bu çalışmada önerilen IDS sistemi her bir ID için ayrı ayrı çalışmakta ve her ID için alınan bireysel kararı birleştirerek nihai bir karar oluşturmaktadır. Alınan nihai karar eşik değerini aştığında bir alarm yaratmaktadır. Bu süreç, her bir veri penceresinde tekrarlanmaktadır.

Yapılan çalışmada tasarlanan izinsiz giriş tespit yönteminin avantajı olarak aynı anda frekans-görünüm değişikliğini ve veri içeriği değişikliğini tespit edebilmesi belirtilmiştir. Bu

durumun, daha sonra veri yolunun güvenliğini sağlamaktan sorumlu tek bir ECU'ya uygulanabileceği; önerilen IDS sisteminin birkaç ayrı IDS kullanımını ortadan kaldırabileceği ve böylece karmaşıklığı ve uygulama maliyetini azaltabileceği iddia edilmiştir.

Çalışmada kullanılan veriler 2012 üretimi Subaru Impreza otomobilinden elde edilen verilere eklenen bir saldırı çerçevesinden oluşmaktadır. Bu veri paketinde birkaç aylık süre için 24 saatlik CAN trafiği bulunmaktadır. Bu veri tabanında, yüksek hızlı CAN veri yolunun 20 farklı ID'si ve yaklaşık 30 farklı modu vardır. Veriler ön işleme gerçekleştirildikten sonra K-means algoritması kullanılarak, ID veri paketleri ayrı ayrı alınarak frekans görünümleri ve her ID'ye karşılık gelen modlar çıkarılmıştır. Daha sonra MLP ağı ayrı ayrı eğitilerek her bir ID için karar verilmiştir. Her ID için girilen veri dizisi normal veya şüphelidir. Son olarak, önceki uyarının kombinasyonundan bir eşik değeri belirlenerek nihai bir karar verilmiştir.

Çalışma kapsamındaki saldırı senaryosunun gerçekleştirilebilmesi için çeşitli varsayımlarda bulunulmuştur. Bu varsayımlar, saldırganları veri paketlerinin yapısını bildiği, ağa mesaj gönderip alabildiğidir. Saldırgan, saldırı yapabilmek için geçerli ECU'ların ID'lerini kullanarak hatalı paketler göndermek için veri alanlarının içeriğini veya paketlerin görünme sıklığını değiştirmektedir. Yapılan simülasyonda, her bir ID veri paketi, kendi eğitim, doğrulama ve test veri setleri ile ayrı ayrı gerçekleştirilmiştir. Kümeleme aşamasında elde edilen her bir ID veri paketi için veri seti %70 eğitim, %10 model doğrulama ve %20 son test olacak şekilde bölünmüştür. Yapılan çalışmanın performansının test edilmesi için ROC eğri analizi kullanılmıştır.

Armin Wasicek ve Andre Weimerskirch, çalışmasında makine öğrenmesi tabanlı bir IDS geliştirmişlerdir [17]. Bu çalışmada MLP tabanlı bir yapay sinir ağı (ANN) kullanılmıştır. Bu yapay sinir ağı sistemde algılayıcı modüldür. Algılayıcı modül olarak kullanılan MLP sistemin normal çalışma şekline göre eğitilir. Eğitim işlemi için sensör verilerinden çıkarılan özellik vektörleri kullanılır. Eğitildikten sonra sensörlerden gelen verileri, algılayıcı modül devamlı olarak kontrol eder. Yapay sinir ağının çıktısına göre bir anomali skoru hesaplanır. Bu skor ortalama karekök sapması (Root-Mean-Square Error) fonksiyonuna göre hesaplanır. Sistemde kullanılan 5 farklı çeşit özellik vektörü bulunur. Bunlar, ortalama değer, standart sapma, varyans, kayıklık (skewness), basıklıktır (kurtosis). Özellik vektörleri çıkarıldıktan sonra MLP'ye giriş olarak verilmeden normalize edilir.

Son olarak Armin Wasicek ve Andre Weimerskirch, çalışmasında sonuçların umut verici ve uygulanabilir olduğunu ancak yapay sinir ağı tabanlı IDS sistemlerinin daha da geliştirilmesi gerektiğini vurgulamıştır.

Min-Joo Kang ve Je-Won Kang çalışmasında Deep Neural Network (DNN) tabanlı bir IDS geliştirmişlerdir [18]. DNN eğitimi için özellik vektörleri doğrudan CAN paketinin bit verisinden oluşmaktadır. Herhangi bir kod çözme veya özellik çıkarma işlemi olmaması nedeniyle hesap yapma verimliliğinin arttırıldığı öne sürülmüştür. DNN katsayılarının başlangıç değerlerine atanması için (initializing) Deep Belief Networks (DBN) gözetimsiz (unsupervised) ön-eğitim metodu kullanılmıştır. Önerilen metotta mod verilerinin ve verilerin sayısal değerinin mantıksal olarak kullanıldığı belirtilmiştir. Min-Joo Kang ve Je-Won Kang sistemlerini birkaç ECU ile test etmişlerdir. Sistem için üretilen CAN paketleri OCTANE (Open Car Test-bed and Network Experiments) adı verilen paket üreteci ile üretilmiştir. Bu süreçte 200 bin paket üretilmiş bu paketlerin %70'i DNN eğitimi %30'u ise testi için kullanılmıştır. Sonuçlarla önerilen metodun %99 saldırı tespiti oranı olduğunu, ayrıca yanlış pozitif hatalarının ise %1-2 aralığında bulunduğunu belirtmişlerdir. ANN ve Support Vector Machine (SVM) metotları ile kendi metotlarını kıyaslamışlar ve kendi metotlarının bu iki yöntemden daha iyi performansa sahip olduğunu belirtmişlerdir.

Taylor et al. çalışmasında istatiksel ve gözetimsiz (unsupervised) yapay sinir ağı tabanlı bir IDS geliştirmişlerdir [19]. Metot genel olarak, çalışan bir CAN veri yolunu istatiksel olarak hesaplamalar yapıp, bu hesaplamaları önceki değerler ile kıyaslamaktadır. Bu sisteme akış tabanlı anomali algılama yöntemi adı verilmiştir. Bu sistemde belirli bir süre zarfında gönderilen paketlerin, ID numarası, kaç paket gönderildiği, ardışık paket arasındaki ortalama Hamming uzaklığı, ardışık paket arasındaki Hamming uzaklığın varyansı, ardışık paket arasındaki zaman ortalaması, ardışık paket arasındaki zaman varyansıdır. Sistemden çıkarılan bu önceki değerler, eğitim için kullanılmıştır. Sistemde özelliklerin çıkarılabilmesi için öncelikle 1 saniyelik pencereler seçilmiştir. Ancak bir CAN veri yolunda 1 saniye periyotlarla gelen paketler de bulunduğundan, 1 saniyelik istatistiklerin bu tip yavaş gelen paketler için kullanılamayacağı açıklanmıştır. Bu nedenle sistemin daha basit test edilmesi açısından 50 ms veya daha uzun periyotlarla gönderilen paketler istatistikten çıkarılmıştır. Sistemin verileri önceki veriler ile kıyaslama yapabilmesi için, OCSVM (One-Class Support Vector Machine) kullanılmıştır. Test için CAN veri yolundan toplanan veriler ile eğitilen OCSVM daha sonraki verilerin istatistiklerinden anomali algılaması yapmaktadır. Sistemin testi için 2011 üretimi Ford Explorer aracından alınan veriler kullanılmıştır. Veriler eğitim

için kullanıldıktan sonra, saldırı simülasyonu için veriler içerisine saldırı paketleri enjekte edilmiştir. 100 ms aralıklarla paket enjekte edilmiştir. Paketler için ortalama gönderilme sıklığının 1 katı, 5 katı veya 10 katı oranında paket enjekte edilmiştir. Ayrıca, sistem performansını ölçmek için farklı çerçeve uzunlukları için testler de yapılmıştır.

Weber et al. çalışmasında spesifikasyon ve makine öğrenmesi tabanlı hibrit bir IDS öne sürmüşlerdir [20]. Çalışmalarında Mütter et. al. çalışmasında tanımlanan sensör yapısını kullanmıştır. Bu sensörlerin ilk altısını (S1-S6) statik kontrol olarak tanımlamışlardır. Bu statik algılama, araç içerisinde haberleşme matrisinde tanımlanmış spesifikasyonlara göre algıma yapmaktadır; ancak S6 ve S7 olarak tanımlanmış sensörlerin haberleşme matrisine göre çalışamayacağı belirtilmiştir. Bu sensörler için farklı tip anomalileri algılamaya yarayan makine öğrenmesi algoritmalarının kullanılması önerilmiştir. Çalışmada tasarlanan IDS sisteminin statik algılamanın yanında paralel olarak birden fazla makine öğrenmesi algoritmasının da anomali kontrolü yapması önerilmiştir. Önerilen makine öğrenmesi algoritmaları, Replicator Neural Network (RNN), OCSVM, ve Lightweight On-line Dector of Anomalies (LODA) algoritmalarıdır. Bu çalışmaya göre her bir makine öğrenmesi algoritması kendi çıkışında bir anomali skoru üretir. Bu skor, paketin anormal olma ihtimalini belirtir. Skor daha önceden tanımlanmış eşik değerleri ile kıyaslanır. Bu değerin üzerinde ise anomali olarak kayıt edilir. Ancak çoklu algılama algoritması kullanılan sistemlerde üretilen skorların bir analizör ile oylanması ve olayın anomali olarak kayıt edilmesi veya edilmemesine karar verilmesi gerektiği belirtilmiştir. Statik algılama kısmında ise herhangi bir çıkış analizörüne gerek olmadığı, bu sensörlerin doğrudan haberleşeme matrisine göre çalıştığı belirtilmiştir. Bu nedenle statik algılamanın sonuçları doğrudan kayıt edilmiştir.

Taylor et al. çalışmasında makine öğrenmesi tabanlı bir IDS geliştirmişlerdir. Sistemlerinde Long-Short Term Memory (LSTM) yapay sinir ağı kullanmışlardır [21]. Öncelikle 2012 Subaru Impreza model araçtan alınan verileri incelemişlerdir. Bu verileri bit değerlerine göre incelemişlerdir. 20 farklı ID ve 64 bit veri içeren paketler için analiz yapılmıştır. Aynı ID'ye sahip paketlerin bazı noktalarında verilerin değişmediğini belirlemişlerdir. Farklı ID'ye sahip paketler için ortalama bit değerleri hesaplanmıştır. Taylor et al. çalışmada LSTM yapısını bir sonraki paketin verilerini tahmin etmek için kullanılmıştır. Bu tahmin sonucu ve gelen pakete göre bir hata oranı çıkarılmaktadır. Çıkarılan hata oranına göre durumun bir anomali olup olmadığının kararı verilmektedir. Kullanılan LSTM'in bir başka yararı ise veri yolu için fazladan herhangi bir bilgi

gerekmemesidir. Bu sebeple, CAN verilerinin şifreleme, kod çözme gibi işlemlerden geçirmeden doğrudan LSTM ağına verilebilmesidir.

Wang et al. bir arabada meydana gelen saldırıları oldukları sırada tespit etmektedir [22]. Bir diğer deyişle, araç sürüş halindeyken CAN veri yoluna yapılan saldırıları tespit etmeyi amaçlamaktadır. Bu çalışma sırasında algılayıcıların eğitilmesi sırasında büyük bir veri setine sahip olunduğu varsayılmıştır. Çalışmanın tanımladığı problem yarı denetimli bir anomali algılama problemidir. Bu çalışma sırasında karşılaşılan temel zorluklardan biri, gerçek veriler ile kötü amaçlı verilerin, yani saldırıların arasındaki farklılıkların tespit edilmesidir. Başka bir zorluk ise otomotiv endüstrisindeki algılayıcıların çevrimiçi olarak çalışma zorunluluğudur. Bu çalışmada tasarlanan sistem ilk olarak, tek bir ID'ye sahip paketler veri tanımlama modellerinden geçirilerek işlenmektedir. Algılayıcı girişleri paketin veri alanındaki bitlerdir. Ardından, HTM (hierarchical temporal memory) algoritması kullanılarak, sistemin bir sonraki veri alanlarını tahmin etmesi eğitilir. Ek olarak, veri alanlarındaki her bir bitin puanından sonra logaritmik kayıp fonksiyonu ile tek puana indirilir. Son olarak, bir zaman penceresi içinde tek bir ID'den oluşan giriş dizisi için genel bir skor alınmaktadır. Tasarlanan bu sistem, tek bir kimlik için çıkan anomali skoru belirlenmiş bir eşik değerinin altına düştüğü durumda uyarı vermektedir. Sistemin genel değerlendirmesi sonucunda, Markov ve RNN modelleri gibi diğer mevcut CAN veri alanı anomalisi tespit sistemleri ile karşılaştırıldığında daha güvenilir algılama elde edilebileceği söylenmiştir.

Tomlinson et al. zaman tanımlı pencere yaklaşımı ile iki denetimsiz yöntem olan Z skoru ve Otoregresif Entegre Hareketli Ortalama birleştirmiştir [23]. Aynı zamanda denetimli karşılaştırma ile de birleştirilerek sonuçlar karşılaştırılmıştır. Bu çalışma kapsamında saldırılar, CAN kayıtlarının güncellenmesi aracılığıyla her CAN ID için sistematik olarak simüle edilmiştir. Değiştirilen bu kayıtlar çalışma kapsamında tasarlanan algoritma ile işlenmiştir. Çalışma kapsamında üç farklı tespit yöntemi karşılaştırılmıştır. Bu yöntemler; ortalama kare için kullanılan pencerenin ARIMA modeli ile hata tabanlı yayın aralığı anormalliğini algılama, her yayın aralığı için Z-puanı ile karşılaştırıldığında pencere ortalaması, pencereye karşı yayın-zaman aralığının ortalamasının karşılaştırılmasıdır. Üç yöntem kullanılarak CAN veri yolundaki her yayın aralığı karşılaştırılarak analiz edilmiştir. Ortalamalar için hesaplamaların azaltılması amacıyla bir penceredeki her paketin aynı ID'ye sahip olduğu düşünülmüştür. Tüm CAN ID'leri eklendiğinde tespit başarısının azaldığı

gözlemlenmiştir. İki denetimsiz yöntemin öncelikleri düşük olduğunda sistemin daha çok etkilendiği görülmüştür.

Lee et al. çalışmalarında uzak çerçeve kullanarak genel performansı ve doğruluğu arttırmak için bir saldırı tespit yöntemi önermiştir [24]. Uzak çerçeve kullanılarak araç içi düğümlerin değiştirilmesi için özel bir tanımlayıcı içeren bir algoritma önerilmiştir. Bu algoritma sayesinde düşük hesaplama gücü ile tehlikeli olan düğüm saldırısı tespit edilmiştir. Bu çalışmadaki algoritma ile CAN protokolü değiştirilmeden izinsiz giriş tespiti gerçekleştirilmesi amaçlanmıştır.

Bu çalışmada araç içi CAN düğümünü değiştirebilen ve veri yoluna kimliği doğrulanmamış mesajlar enjekte edebilen iki tip saldırgan varsayılmıştır. İlk saldırganın fiziksel değişiklikler yapabildiği, hedefin kontrolünü ele geçirebildiği, mesajların iletimini durdurabildiği düşünülmüştür. İkinci saldırgan tipi olarak ise düğümlere erişimi olamayan ama veri yoluna kötü amaçlı yazılım enjekte ederek arızalara neden olabilecek bir yapı düşünülmüştür. Başka bir deyişle, DoS ve bulanık(fuzzy) saldırı türleri değerlendirilmiştir. Bu saldırganlara bağlı olarak bir saldırı tespit sistemi tasarlanmıştır. Bir aracın saldırı altında olup olmadığının tespiti için tasarlanan bu sistemde düğümlerin yanıt performansları analiz edilmiştir. Tasarlanan bu sistem ile mesaj enjeksiyonu sırasında ne tür mesajlar enjekte edildiği, saldırı ve kimliğe bürünme sırasında hangi düğümün güvenliğinin ihlal edildiği gibi bilgiler öğrenilebilmektedir.

Choi et al. çalışmada her bir ECU'nun CAN veri yolu üzerinde oluşturduğu sinyal karakteristiklerini kullanan bir IDS geliştirmiştir [25]. Bu sistem her bir ECU'nun veri yolu üzerinde oluşturduğu sinyaller örneklenmiştir. Her bir örneğin hem zaman hem de frekans alanındaki özellikleri çıkarılmıştır. Çıkarılan bu özellikler ile ECU'lar, yapay sinir ağı, SVM ve BDT (Bagged Decision Tree) metotları ile sınıflandırılmıştır. Sistemin testi için 12 ECU CAN düğümü kullanılmıştır. Veri yolu 2.5 GSPS olan bir osiloskop ile örneklenmiştir. Sistemde aynı mesajı gönderen her bir düğüm için 900 örnek alınmıştır. Veri transferi 500 Kbps hızında yapılmıştır. Choi et al. yüksek CAN hızlarında sinyalin karakteristiklerinin çıkarılmasının daha zor olacağını belirtmiştir. Sinyal karakteristikleri ile yapılan test sonuçlarında %85 - %95 oranında doğru sınıflandırma yapıldığı gösterilmiştir.

Yang et al. da çalışmada ECU'ların CAN veri yolunda oluşturduğu farklı sinyalleri kullanan bir IDS geliştirmiştir [26]. Bu sistem, diğer sistemlerden farklı olarak sinyallerden özellik vektörleri çıkarılmadan verilerin doğrudan LSTM yapay sinir ağına gönderilmesidir.

Yang et al. LSTM yapay sinir ağının zaman serisi olan verileri sınıflandırmada yüksek başarı sağladığını belirtmiştir. Bu nedenle fiziksel katmandan alınan veriler doğrudan LSTM yapısının eğitiminde kullanılmıştır. Çalışma kapsamında CAN fiziksel veri yolu modellenmiştir. Modellenen bu fiziksel model ve gerçek fiziksel katman kıyaslanmıştır. Modellenmiş fiziksel model üzerinden 50 farklı ECU verisi üretilmiştir. Her bir ECU için 5 farklı kayıt alınmıştır. Ayrıca Her bir ECU'nun 2 CAN-H ve CAN-L için iki farklı zaman serisi bulunmaktadır. Bu verilerin uzunluğu ise 6300 örnektir. Toplamda 3 milyon örnek kullanılmıştır. Sistem performansının %98 üzerinde başarı gösterdiği ve ayrıca %0.36 ortalama hatalı sınıflandırma oranına sahip olduğu vurgulanmıştır.

3. MATERYAL

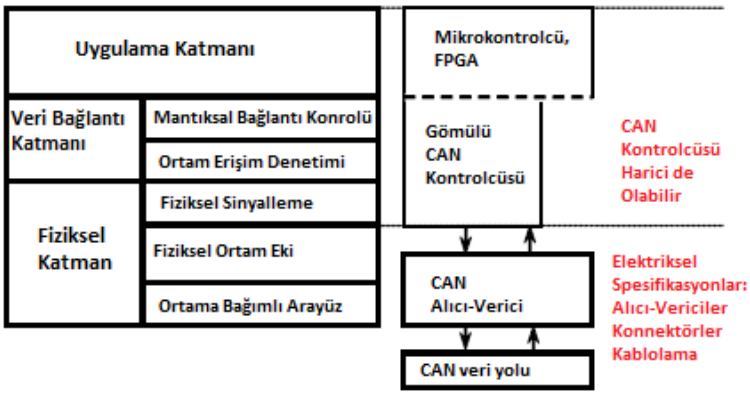
3.1. CAN PROTOKOLÜ

CAN 2.0 protokolü BOSH firması tarafından geliştirilmiş olan, çoklu erişim (CSMA\CD) yapısına sahip mesaj yayın sistemidir. Maksimum veri transfer hızı 1 Mbps'dir. CAN protokolü, USB, Ethernet gibi diğer geleneksel protokoller gibi bir noktadan diğer noktaya yüksek miktarda veri göndermez. Ayrıca, diğer protokoller gibi veri transferi merkezi bir veri yöneticisi (Bus Controller) altında gerçekleşmez. Tipik bir CAN ağında, düğümler sıcaklık, RPM, telemetri verileri gibi çok sayıda kısa mesajı tüm ağdaki diğer düğümlere yayın yaparlar. Bu sayede tüm düğümler diğer düğümlerin durumundan haberdar olurlar.

3.1.1. CAN Standardı

CAN protokolü ISO (International Standardization Organization) tarafından tanımlanmış olan seri haberleşme protokoldür. Otomotiv endüstrisi için tasarlanmış olup, araç içerisindeki çok sayıda olan haberleşme kablo sisteminin yerine sadece iki kablo kullanarak yerine getirmesi için tasarlanmıştır. CAN standardı elektriksel karışıma karşı dayanıklılık sağlamakla beraber, veri hatalarını kendi kendine algılamak ve ağı tamir edebilme kabiliyetine sahiptir. Bu özellikler CAN protokolünün otomotiv, medikal, savunma, havacılık ve uzay alanlarında popülerliğini arttırmıştır.

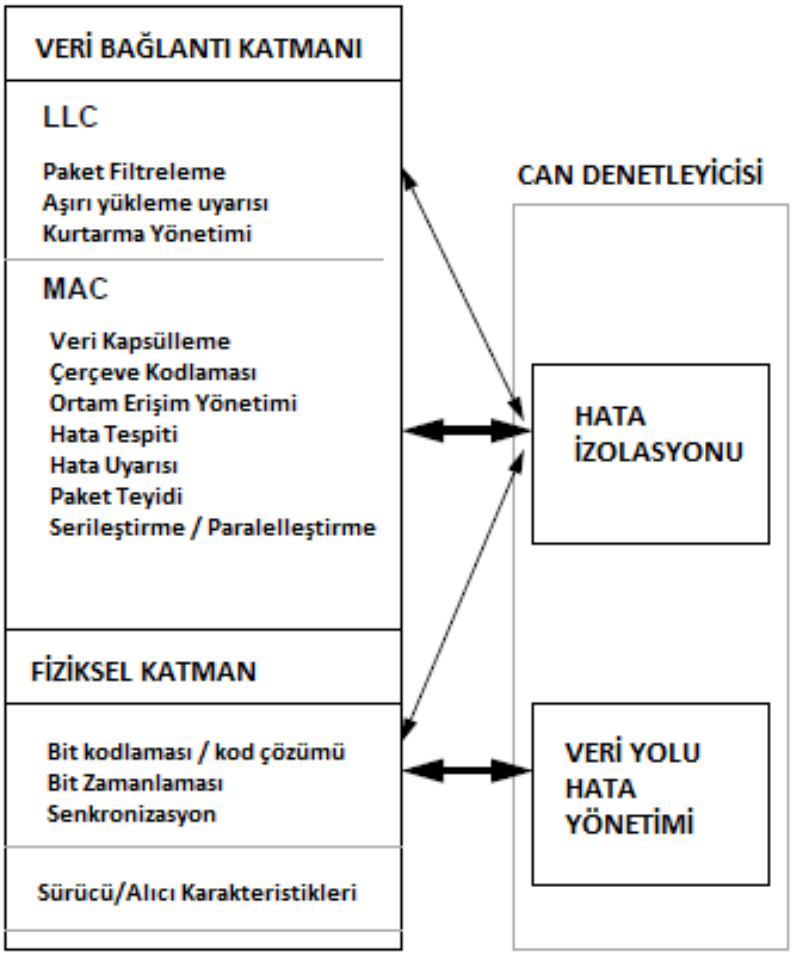
ISO-11898:2003, standardında CAN protokolünün haberleşme yapısı ve veri transfer mantığı tanımlanmıştır [27]. Standartta CAN katmanları OSI (Open Systems Interconnection) modeline göre tanımlanmıştır. Bu modele göre katmanlar Şekil 3.1'de verilmiştir.



Şekil 3.1. CAN protokol Katmanları [27]

3.1.2. CAN katmanları

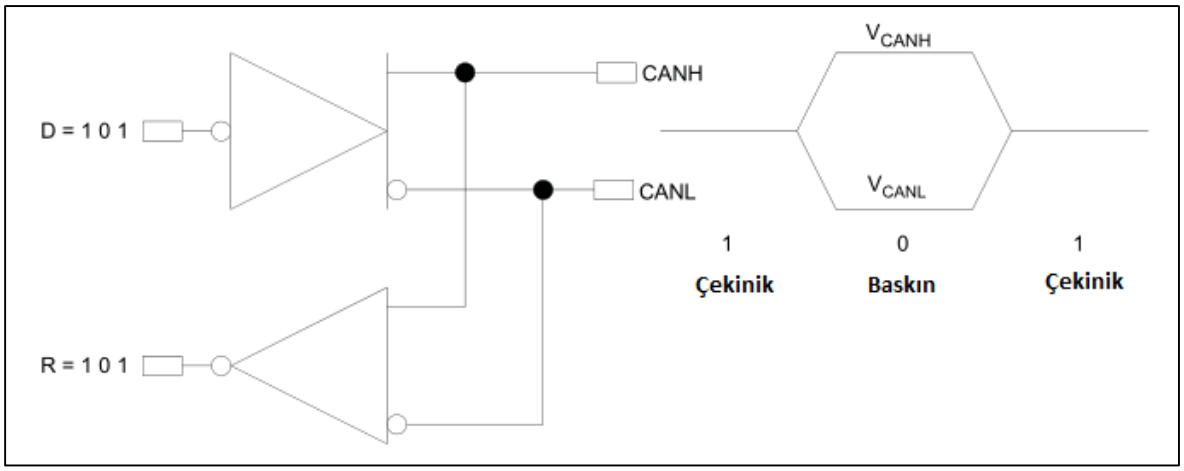
CAN protokolü OSI modeline göre iki katmandan oluşur. Fiziksel katman (Physical Layer) ve Veri Bağlantı katmanıdır (Data Data Link Layer). Bu katmanların genel olarak işlevleri Şekil 3.2'de verilmiştir.



Şekil 3.2. CAN OSI modeline göre katmanları [1]

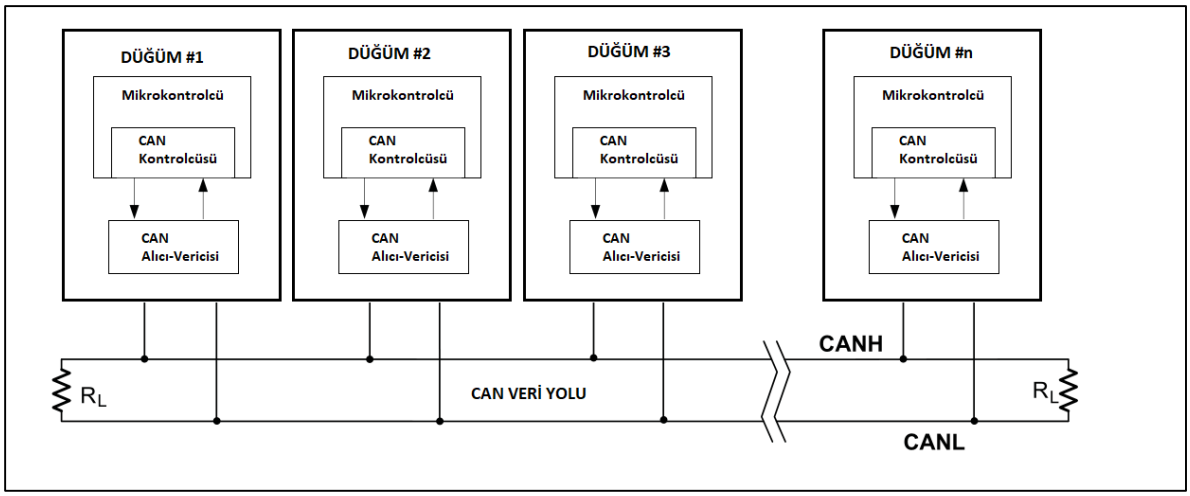
3.1.3. CAN Fiziksel Seviyesi

Bir CAN düğümünün fiziksel katmanında aynı hatta bağlı diferansiyel giriş ve çıkış tamponları (buffer) bulunur. Bu tamponlar CAN kontrolcüsünden çıkan tek uçlu (Single-ended) sinyalin CAN protokolünde tanımlanmış olan baskın ve çekinik sinyallere çevirir. CAN kontrolcüsünden gelen '0' lojik değeri tampon devreler ile baskın, '1' değeri ise çekinik bite çevrilir. CAN fiziksel bit seviyesi Şekil 3.3'te verilmiştir.



Şekil 3.3 CAN Giriş – Çıkış Tampon Devresi ve CAN sinyali [27]

CAN veriyolu düğümlerin CAN-H ve CAN-L olarak isimlendirilmiş hatlar üzerinden birbirlerine bağlanmasıyla oluşturulur. CAN-H ve CAN-L sinyalleri çekinik durumda iken yaklaşık 2.5 Volt değerinde olurlar. Baskın durumda ise CAN-H 3.5 volt değerine kadar çıkabilir, CAN-L ise 1.5 Volt değerlerine inerek ortalama olarak 2 Volt farklı bir diferansiyel sinyal oluştururlar. Bu veri yolu elektriksel olarak 120 Ohm empedans olacak şekilde tasarlanmıştır. Bu nedenle veri yolu 120 Ohm sonlandırma dirençleri ile sonlandırılır. Bu sayede sistemdeki sinyal geri yansıması engellenir. Klasik bir CAN veri yolu Şekil 3.4'te verilmiştir.



Şekil 3.4 CAN Veri Yolu Bağlantı Şeması [27]

3.1.4. Çerçeve Formatları

CAN protokolü iki farklı çerçeve yapısı bulunmaktadır. Bu formatlar 11 bitlik ID içeren standart çerçeve ve 29 bitlik ID alan içeren genişletilmiş çerçevedir.

3.1.5. Çerçeve Tipleri

Can protokolünde mesaj transferi 4 farklı çerçeve tipi ile gerçekleşir. Bu çerçeve tipleri şu şekilde verilmiştir:

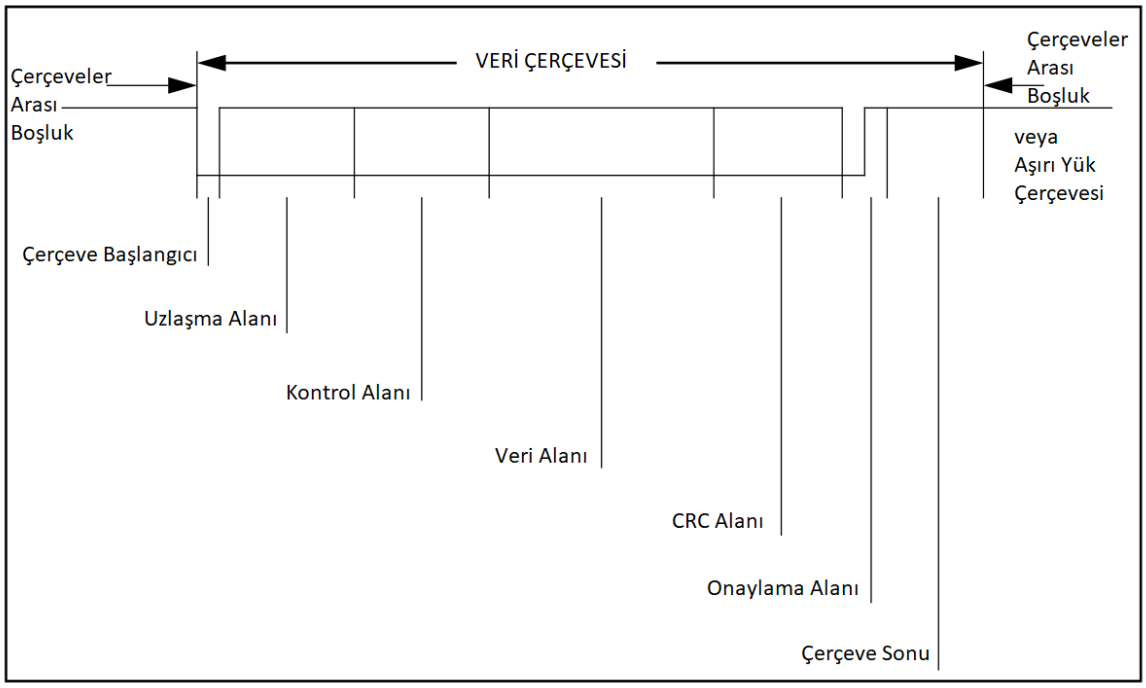
- Veri Çerçevesi
- Uzak Çerçeve
- Hata Çerçevesi
- Aşırı Yük Çerçevesi

3.1.5.1. Veri Çerçevesi

Veri çerçevesi yedi farklı bit alanından oluşmaktadır. Bu bit alanları şu şekilde verilmiştir:

- Çerçeve başlangıcı (SOF)
- Uzlaşma (Arbitration) Alanı
- Kontrol Alanı
- Veri Alanı
- CRC Alanı
- Onaylama (ACK) Alanı
- Çerçeve sonu (EOF)

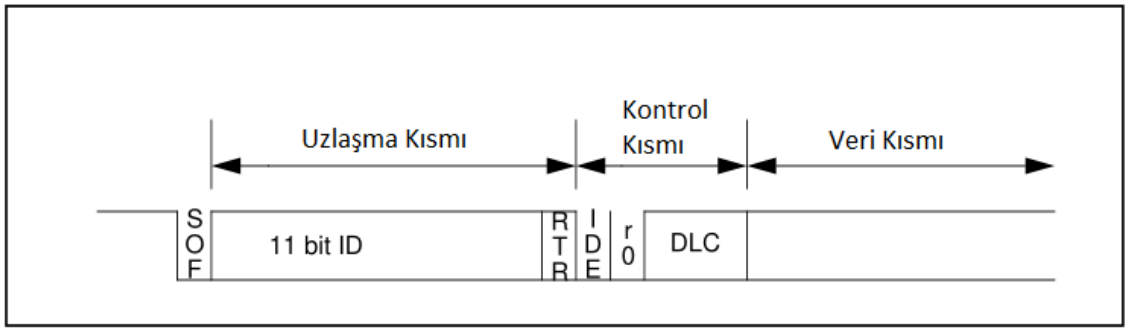
Veri çerçeve yapısı Şekil 3.5'te verilmiştir.



Şekil 3.5 CAN Veri Çerçevesi Yapısı [1]

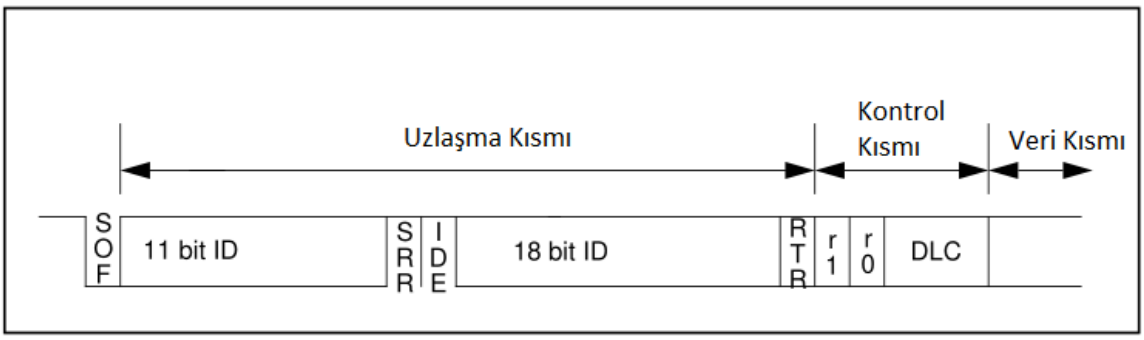
SOF kısmı, veri çerçevesinin başlandığını işaret eder. Veri çerçevesi ve uzak çerçeve için bu kısım ortaktır. SOF kısmı bir baskın bitten oluşmaktadır.

Uzlaşma kısmı Standart Format ve Genişletilmiş Format için farklılık göstermektedir. Standart formatta, uzlaşma kısmı 11 bit ID ve RTR bitinden oluşmaktadır. Standart format yapısı Şekil 3.6'da verilmiştir.



Şekil 3.6 CAN Standard Format Yapısı [1]

Genişletilmiş Formatta ise uzlaşma kısmı 29 bit ID, SRR biti, IDE biti ve RTR bitinden oluşmaktadır. Genişletilmiş Format Yapısı Şekil 3.7'de verilmiştir.



Şekil 3.7 CAN Genişletilmiş Format yapısı [1]

Genişletilmiş formatta uzlaşma kısmında bulunan ID kısmı gönderilen mesajın kimlik bilgilerinin bulundurur. ID kısmı 29 bitten oluşmaktadır ve [ID-28:ID-0] şeklinde gösterilmiştir. Standard Formatta ID kısmının sadece MSB 11 biti gönderilir [ID-28:ID-18]. Bu 11 bitlik ID kısmına temel (Base) ID denilmiştir. Genişletilmiş Formatta ise ID kısmının geri kalanı da [ID-17:ID-0] gönderilir. Bu 18 bitlik kısma ise genişletilmiş (extented) ID olarak tanımlanmıştır. Genişletilmiş formatta 29 bitlik ID kısmının tamamı gönderilir. Ancak CAN protokolüne göre ID kısmının MSB 7 bitinin [ID-28:ID-22] tamamı çekinik bit olmamalıdır.

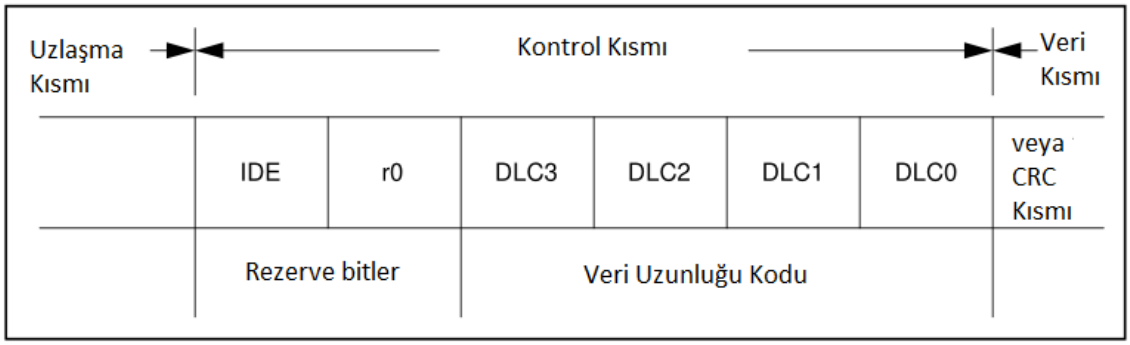
RTR (Remote Transmission Request) biti hem standart formatta hem de genişletilmiş formatta bulunmaktadır ancak bu bitin konumu her iki formata göre değişiklik gösterir. Bu bit bir CAN paketinin veri çerçevesi ya da uzak çerçeve olduğunu belirler. RTR biti baskın ise çerçeve veri çerçevesi, çekinik ise uzak çerçeve olduğuna işaret eder.

SRR (Substitute Remote Request) biti sadece genişletilmiş formatta gönderilir. Bu bit standart formatta RTR bitinin konumunda gönderilir. Değeri daima çekiniktir. SRR biti sayesinde bir CAN veri yolunda aynı anda gönderilen ve aynı temel ID'ye sahip iki paketten öncelik standart çerçeveye verilir.

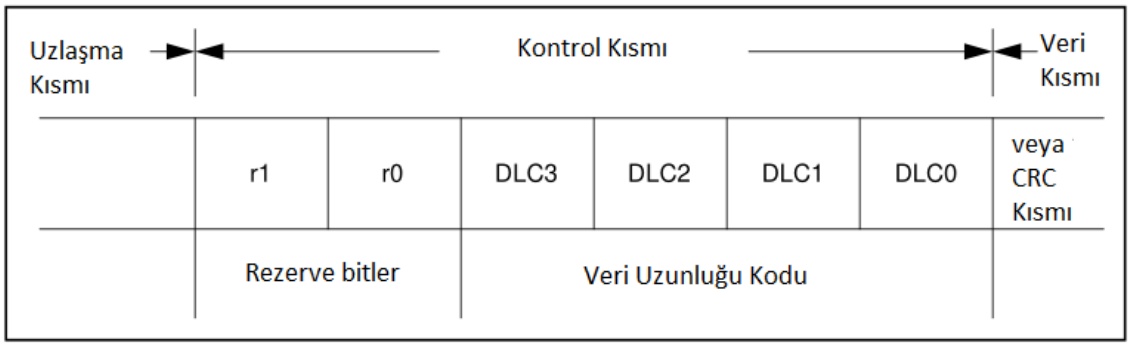
IDE (Identifier Extention Bit) biti aldığı değere göre paketin standart çerçeve veya genişletilmiş çerçeve olduğunu belirler. IDE biti standart formatta kontrol kısmında bulunmaktadır. Genişletilmiş formatta ise bu bit uzlaşma kısmına aittir. IDE biti standart formatta baskın olarak gönderilir, genişletilmiş formatta ise bu bit çekinik olarak gönderilmektedir.

Kontrol kısmı 6 bitten oluşmaktadır. Kontrol kısmının LSB 4 bit hem standart formatta hem de genişletilmiş formatta ortaktır ve bu kısım DLC (Data Length Code) olarak

tanımlanmıştır. Standart formatta MSB 2 bit IDE biti ve r0 bitinden oluşmaktadır. Genişletilmiş formatta ise bu kısım r1 ve r0 olarak tanımlanmış rezerve bitlerden oluşur. CAN 2.0B protokolünde rezerve bitler (r1, r0) dominant olarak gönderilir. Kontrol kısmı yapısı standart format için Şekil 3.8'de genişletilmiş format için ise Şekil 3.9'da verilmiştir.



Şekil 3.8 Standart Formatta Kontrol Kısmı [1]



Şekil 3.9 Genişletilmiş Formatta Kontrol Kısmı [1]

DLC kısmı veri kısmındaki verinin byte cinsinden sayısını belirtir. Bir CAN paketinde en fazla 8 byte veri bulunmaktadır. Ayrıca bir veri çerçevesinde DLC 0 olarak ayarlanabilmektedir ve hiç veri bulundurmayan veri çerçeveleri protokol için geçerli paketlerdir. DLC içeriği Şekil 3.10'da verilmiştir. DLC değerinin Şekil 3.10'da gösterilmemiş olan varyasyonları protokolde kullanılmamaktadır.

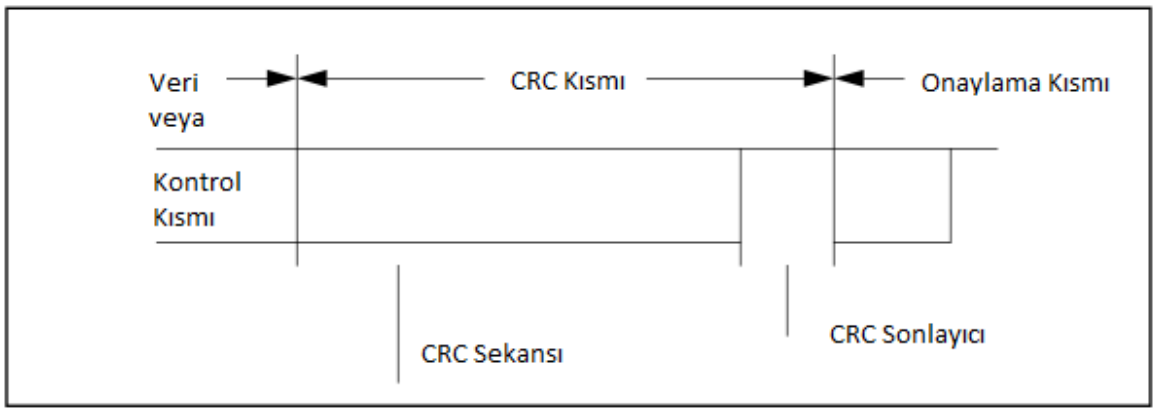
ç : Çekinik
b : Baskın

Veri Uzunluğu (byte)	Veri Uzunluğu Kodu			
	DLC3	DLC2	DLC1	DLC0
0	b	b	b	b
1	b	b	b	ç
2	b	b	ç	b
3	b	b	ç	ç
4	b	ç	b	b
5	b	ç	b	ç
6	b	ç	ç	b
7	b	ç	ç	ç
8	ç	b	b	b

Şekil 3.10 DLC Veri Sayısı Kodlaması [1]

DLC kısmı bir veri çerçevesinde gönderilecek verilerin sayısını bulundurur. Bu kısım en az 0 en fazla 8 değerini alabilir. Veri sayısı DLC kısmında belirtilir. Veriler MSB en önce olacak şekilde gönderilir. En sonda ise LSB gönderilir.

CRC kısmı 16 bitten oluşmaktadır. Bu kısım 15 bitlik CRC verisini ve 1 bitlik CRC sonlayıcı olarak tanımlanmış iki bölümden oluşur. CRC sonlayıcı kısmı çekinik olarak gönderilir. CRC kısmının yapısı Şekil 3.11'de gösterilmiştir.



Şekil 3.11 CRC Kısmı [1]

CRC verisi ise SOF, uzlaşma kısmı, kontrol kısmı ve varsa veri kısmına göre hesaplanır. CRC hesaplaması için kullanılan polinom aşağıda verilmiştir.

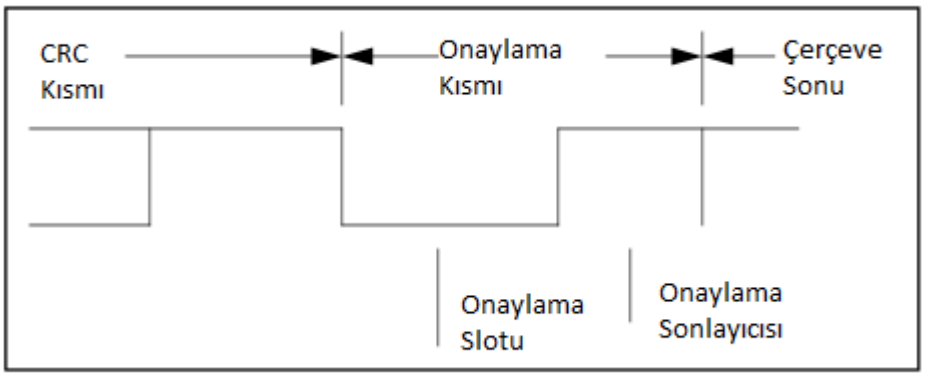
$$P(x) = X^{15} + X^{14} + X^{10} + X^8 + X^7 + X^4 + X^3 + 1$$

CRC algoritmasının sözde kod (pseudocode) olarak yazılmış hali Şekil 3.12'de verilmiştir.

```
function crc(bit array input[1..len], int len) {

    crc_reg = 0b0000000000000000
    crc_nxt = 0b0

    for i from 1 to len {

        crc_nxt = input[i] xor crc_reg[14]
        crc_reg = (crc_reg leftShift 1)
        crc_reg[0] = 0b0

        if crc_nxt == 0b1 {
            crc_reg = crc_reg xor 0b1000010110011001
        }

    }
```

Şekil 3.12 CAN CRC-15 Pseudocode

Onaylama kısmı onaylama slotu ve onaylama sonlayıcısı adı verilen her biri 1 bitlik iki kısımdan oluşur. Onaylama slotunda alıcı geçerli bir paket aldığını bu kısımda baskın bit göndererek vericiye bilgi verir. Bu işlemi veri yolundaki tüm alıcılar gerçekleştirir. Onaylama sonlayıcısı kısmı sabit 1 çekinik bitten oluşmaktadır. Onaylama kısmı Şekil 3.13'te gösterilmiştir.

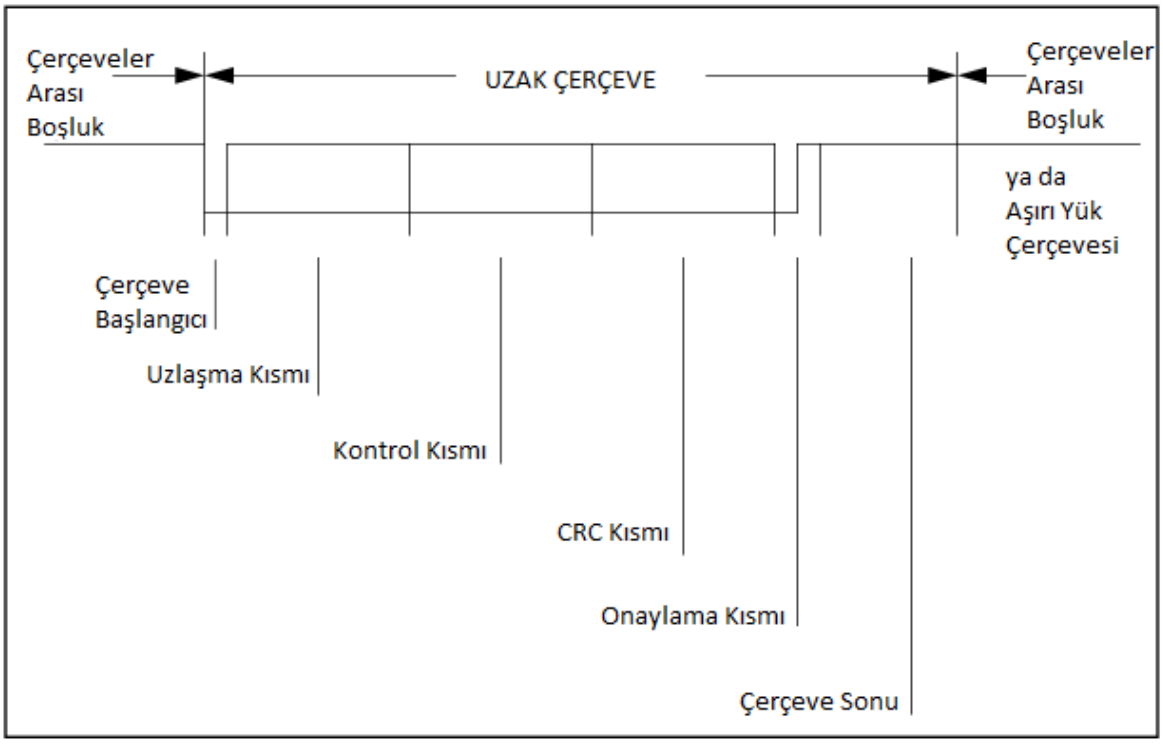


Şekil 3.13 CAN Çerçevesi Onaylama (ACK) Kısmı [1]

3.1.5.2. Uzak Çerçeve

Uzak çerçeve, CAN ağındaki bir düğümün diğer bir düğümden istenilen ID'ye ait verinin transfer edilmesini talep etmek için kullanılır. Hem Standart format hem de

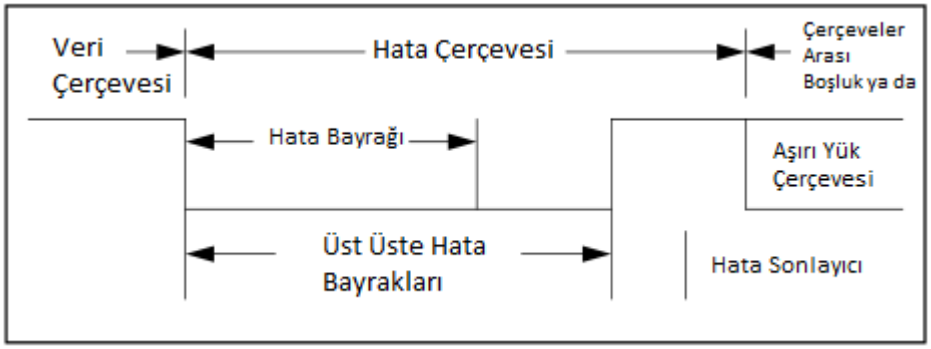
genişletilmiş format için uzak çerçeve bulunur. Uzak çerçeve, veri çerçevesinden farklı olarak RTR bit kısmı çekinik olarak gönderilir. Ayrıca uzak çerçevede veri kısmı bulunmamaktadır. Uzak çerçevede DLC kısmı veri kısmı olmadığı halde bu kısım '0' dan farklı bir değer alabilmektedir fakat DLC 0 dan farklı bir değer olsa bile uzak çerçeve de veri kısmı bulunmamaktadır. Uzak çerçeve yapısı Şekil 3.14'te gösterilmiştir.



Şekil 3.14 CAN Uzak Çerçeve Yapısı [1]

3.1.5.3. Hata Çerçevesi

Hata çerçevesi, hata bayrağı ve hata sonlayıcısı olarak tanımlanmış iki kısımdan oluşmaktadır. Hata bayrağı kısmı birden çok düğümün gönderdiği üst üste binmiş hata bayraklarından oluşmaktadır. CAN protokolünde iki farklı hata bayrağı tanımlanmıştır. Bunlar aktif hata bayrağı ve pasif hata bayrağıdır. Aktif hata bayrağı 6 ardışık baskın bitten oluşur. Pasif hata 6 ardışık çekinik bitten oluşmaktadır. Hata sonlayıcı kısmı ise 8 çekinik bitten oluşur.



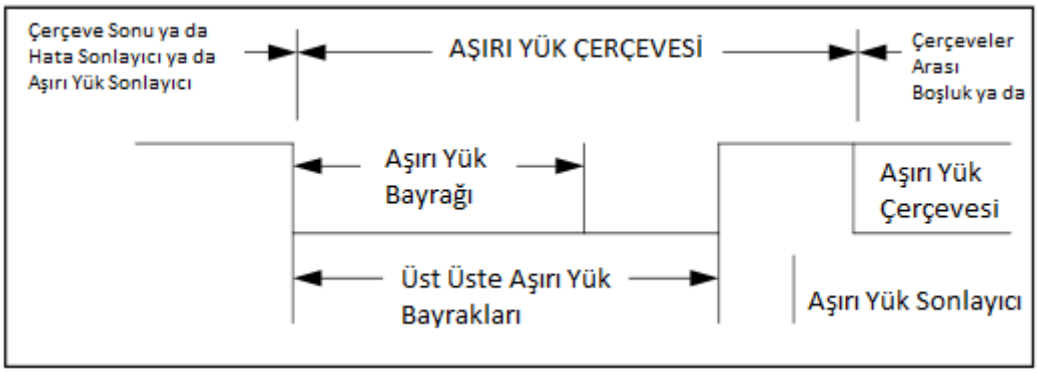
Şekil 3.15 CAN Hata Çerçeve Yapısı [1]

Hata çerçevesi, bir düğümün herhangi bir hata durumuyla karşılaşmasıyla gönderilen CAN protokolü açısından özel bir pakettir. Hata çerçevesi CAN protokolünü veri dolgulaması (stuffing) olarak tanımlanmış olan formatlama yapısını ihlal eder. Bu sebeple bir düğümden gönderilmiş hata çerçevesi tüm düğümlerde hata olarak algılanmasına ve diğer düğümlerinde hata çerçevesi göndermesine neden olur. Bundan dolayı hata çerçeveleri CAN veri yolunda üst üste binmiş olarak gözlemlenirler. Düğümlerde tutulan hata sayaçları sayesinde tekrar tekrar gönderilen hata çerçeveleri nedeniyle oluşabilecek veri yolu kilitlenmeleri protokolde önlenmiştir.

3.1.5.4. Aşırı Yük Çerçevesi

Aşırı yük çerçevesi, aşırı yük bayrağı ve aşırı yük sonlayıcısı olarak tanımlanmış iki kısımdan oluşmaktadır. Yapısal olarak hata çerçevesine benzemektedir. Aşırı yük bayrağı 6 ardışık baskın bitten oluşmaktadır. Aşırı yük sonlayıcısı ise 8 ardışık çekinik bitten oluşmaktadır. Aşırı yük çerçevesi gönderilmesinin genel amacı CAN veri ağının aşırı fazla veri gönderilmesini azaltmaktır. Aşırı yük çerçevesi CAN veri ağında 3 ana sebep nedeniyle gönderilmektedir. Bu sebepler şu şekilde açıklanabilir:

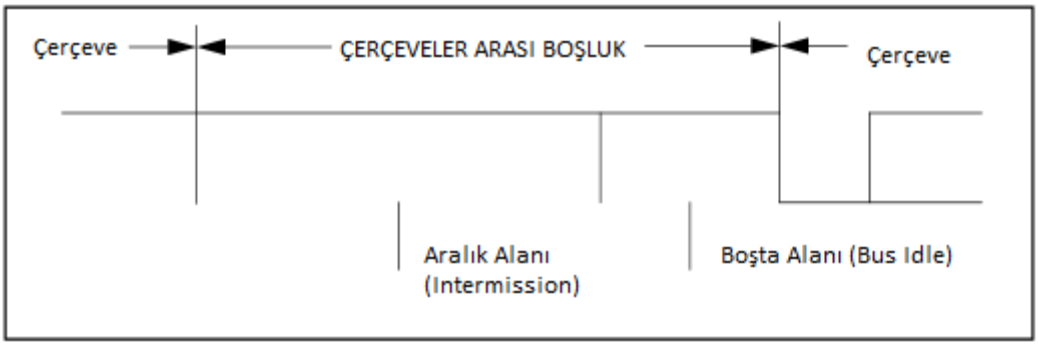
1. Alıcı biriminin iç gecikmeleri sebebiyle alıcı olan bir düğüm aşırı yük çerçevesi gönderebilir.
2. Aralık alanının ilk iki bitinde baskın bit algılanmış ise aşırı yük çerçevesi gönderilebilir.
3. Bir CAN düğümü hata sonlayıcısı veya aşırı yük sonlayıcısı kısmının 8. bitinde baskın bit algılarsa, o düğüm hata çerçevesi yerine aşırı yük çerçevesi gönderir ve hata sayaçları arttırılmaz.



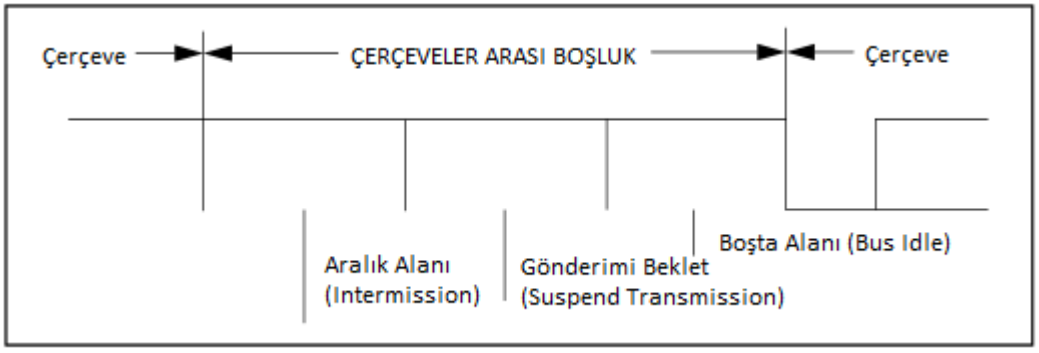
Şekil 3.16 CAN Aşırı Yük Çerçeve Yapısı [1]

3.1.5.5. Çerçeveler Arası Boşluk

Çerçeveler arası boşluk arka arkaya gönderilen iki veri çerçevesi veya uzak çerçeve arasında olan bit alanlarına verilmiş isimdir. Bu alan aralık alanı (intermission) ve boşta (Bus idle) alan olmak üzere iki alandan oluşur. Aralık alanı 3 ardışık çekinik bitten oluşur. Bu kısımda sadece aşırı yük çerçevesi gönderilmesine izin verilir. Boşta alanı, aralık alanından hemen sonra gelir ve sabit bir uzunluğa sahip değildir. Bu kısımda veri yolu boştadır ve herhangi bir düğüm veri transferi başlatabilir. Bu kısımda algılanan baskın bit yeni bir çerçevenin SOF kısmı olarak algılanır. Hata pasif durumda olan bir düğüm aralık alanı ve boşta alanları arasında ek olarak gönderimi beklet (suspend transmission) adı verilen bir kısım daha bulunmaktadır. Hata pasif durumundaki düğüm bu alanda 8 ardışık çekinik bit gönderir. Hata aktif düğümün çerçeveler arası boşluktaki yapısı Şekil 3.17, Hata pasif düğümün çerçeveler arası boşluk yapısı Şekil 3.18'de verilmiştir.



Şekil 3.17 CAN Hata Aktif Çerçeveler Arası Boşluk Yapısı [1]



Şekil 3.18 CAN Hata Pasif Çerçeveler Arası Boşluk Yapısı [1]

3.1.6. CAN Hata kontrolü ve İzolasyon

CAN protokolünün güvenilir olmasının en büyük sebeplerinden biri protokolün hata tespit etme ve bu hatanın izole edilerek bir düğümdeki hatanın diğer düğümleri etkilemesini engelleyen sistemlerinin olmasıdır. CAN protokolünde 5 farklı türde hata tespit yöntemi bulunmaktadır. Bu yöntemlerin ikisi çerçeve düzeyinde kontrol yaparken diğer üçü ise bit düzeyinde kontrol yapmaktadır.

Alınan hatalar düğümler içerisinde TEC (Transmit Error Count) ve REC (Receive Error Count) değerlerinin artmasına sebep olur. Hata değerleri düğümler içerisinde belirli değeri aşması durumunda o düğüm ya veri göndermeyi bırakır ya da tamamen hat üzerinde ne gönderme nede veri alma işlemlerine katılır. Bir düğümün hata sayaçlarının değiştirme algoritması şu şekilde CAN spesifikasyonunda tanımlanmıştır:

1. Alıcı durumunda olan bir düğüm hata algılaması durumunda REC değerini 1 arttırır.
2. Alıcı durumunda olan bir düğüm hata bayrağı göndermesinden bir bit sonra baskın bit algılarsa REC değerini 8 arttırır.
3. Verici durumunda olan bir düğüm hata bayrağı gönderirse TEC değerini 8 arttırır.
4. Verici durumunda olan bir düğüm aktif hata bayrağı veya aşrı yük bayrağı gönderirken bit hatası algılarsa TEC değerini 8 arttırır.
5. Alıcı durumunda olan bir düğüm aktif hata bayrağı veya aşırı yük bayrağı gönderirken bit hatası algılarsa REC değerini 8 arttırır.
6. Her CAN düğümü aktif hata bayrağı, pasif hata bayrağı veya aşırı yük bayrağı gönderdikten sonra en fazla 7 baskın biti tolere edebilir. 8 bit ardışık olarak algılanan baskın bit o durumda olan her verici durumda olan düğüm için TEC

değerinin 8 ve her alıcı durumunda olan düğüm için REC değerinin 8 artmasına neden olur.

7. Başarılı olarak gönderilen her bir çerçeve sonrası TEC değeri 1 azaltılır. (Paketin başarılı gönderilmesi durumu, çerçeve sonuna kadar hiç hata olmaması durumudur.)
8. Başarılı alınan bir paket sonunda REC değeri 1 azaltılır. Bu değer 127'den fazla ise 127 ile 119 arasında bir değere atanır.
9. Bir CAN düğümü TEC veya REC değerlerinden herhangi biri 128'e eşit veya daha büyük ise o düğüm hata pasif durumuna geçer.
10. Bir CAN düğümü TEC veya REC değerlerinden herhangi biri 256'a eşit veya daha büyük ise o düğüm veri yolu kapalı (bus off) durumuna geçer.
11. Hata pasif durumunda olan bir CAN düğümü TEC ve REC değerleri 127'ye eşit veya daha düşük ise o düğüm hata aktif durumuna geçer.
12. Veri yolu kapalı durumunda olan bir CAN düğümü 128 kere ardışık 11 bit çekinik bit monitör etmesi halinde hata aktif durumuna geçer.

3.1.6.1. Bit Hatası

Bit hatası bir CAN düğümünün gönderdiği bitin algıladığı bitten farklı olması durumudur. Bu durumun istisnai durumları vardır. Uzlaşma kısmında bir CAN düğümü gönderdiği çekinik biti baskın olarak algılaması durumu sistem için hata olarak algılanmaz bu durum CAN düğümünün veri yolu üzerindeki kullanım hakkını kayıp ettiğini ve başka bir düğümün o sırada daha öncelikli bir paket gönderdiği anlamına gelir. Ayrıca bir verici durumunda olan düğüm pasif hata bayrağı gönderirken baskın bit algılaması da sistemde hata olarak kabul edilmemektedir.

3.1.6.2. Veri Dolgulama Hatası (Stuff Error)

Veri Dolgulama hatası bir CAN sisteminde ardışık gönderilen en az 6 bitin aynı olması durumunda oluşur. Örneğin arka arkaya gönderilen en az 6 ardışık baskın bit veya en az 6 çekinik bit sistemde veri dolgulama hatasına sebep olur.

3.1.6.3. CRC Hatası

CAN paketlerindeki veriler 15 bitlik CRC verisi ile gönderilir. Paket içerisindeki veriler (de-stuffed) CRC algoritmasına sokulur. Çıkan sonuç ile paket içerisinde gönderilen CRC verisi birbirini tutmadığı zaman CRC hatası verilir.

3.1.6.4. Form hatası

Form hatası paket içerisinde sabit olarak gönderilen bitlerin olması gereken değerden farklı olarak algılanmasında olur. SOF, EOF, onaylama sonlayıcısı ve CRC sonlayıcısı bitleri bu kontrole tabidir.

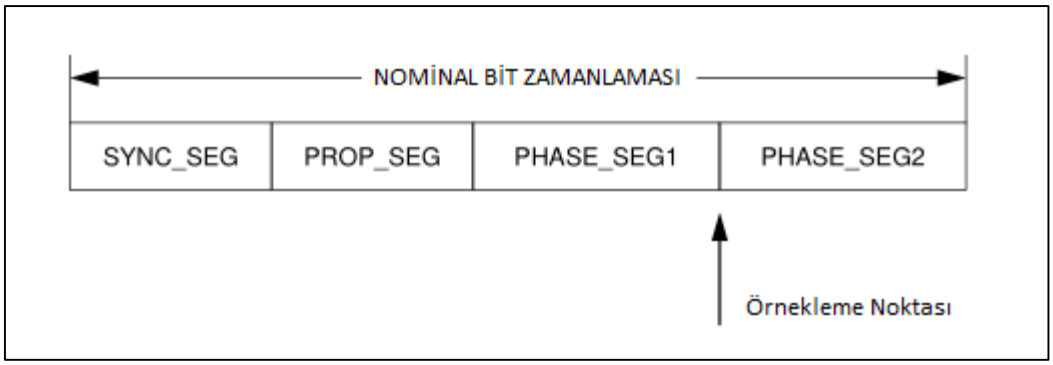
3.1.6.5. Onaylama (Acknowledgement) Hatası

Bu hata verici durumunda olan bir düğüm gönderdiği paket için onaylama slot kısmında baskın bit algılamaması durumudur. Bu durumda paket gönderilmiş sayılmaz ve düğümün durumuna göre daha sonra tekrar gönderilmeye çalışılır.

3.1.7. Bit Zamanlaması

CAN protokolünde düğümlerin zamanlaması çok önemlidir. Düğümlerin zamanlamasındaki kaymalar sistemin çalışma performansını kötü olarak etkileyebilir. CAN protokolünde bir bit zamanlaması 4 ayrı parçaya bölünmüştür. Bu parçalar zamanlama olarak ayarlanabilmektedir. Bu zamanlamalar kontrolcüye bağlı olan saat frekansına bağlı olarak değişkenlik gösterebilir. Farklı frekanslardaki osilatörler için CAN kontrolcülerinde saat bölücü (Clock Divider) adı verilen yapılar bulunur.

Bir bit zamanlaması SYNC_SEG (Synchronızation Segment), PROP_SEG (Propagation Segment), PHASE_SEG1 (Phase Buffer 1 Segment) ve PHASE_SEG2 (Phase Buffer 2 Segment) bölümlerinden oluşur. PHASE_SEG1 ve PHASE_SEG2 bölümleri arasında örnekleme noktası (Sample Point) adı verilen kısım bulunur. Örnekleme noktasında veri yolundaki voltaj değeri örneklenir ve lojik seviyesine dönüştürülür. CAN protokolünde normal bir bit zamanlaması Şekil 3.19'da verilmiştir.

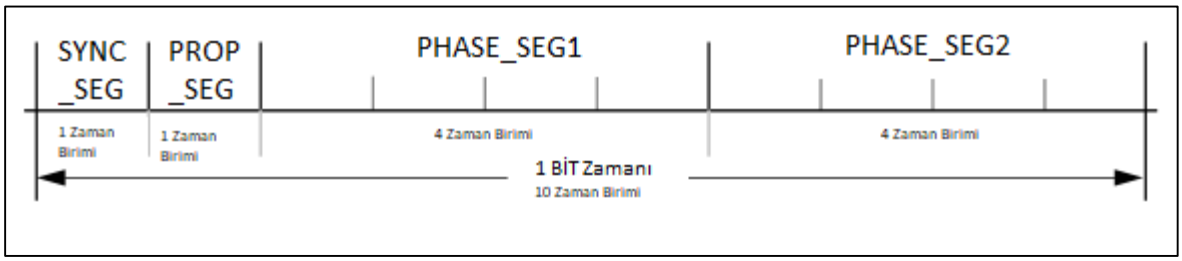


Şekil 3.19 CAN Bit Zamanlaması [1]

SYNC_SEG'te düğümler arası senkronizasyon sağlanır. Bu aralıkta sinyal değişimi beklenir. Bu kısım 1 zaman birimi (Time Quantum) uzunluğundadır.

PROP_SEG'te, sinyalin diğer düğümlere gidiş süresi telafi edilir. Bu kısım 1 – 8 zaman birimi uzunluğunda olabilir.

PHASE_SEG1, PHASE_SEG2 kısımları bitler arası oluşan faz farklını düzenlemek için kullanılır. Bu kısımlar faz hatasına göre uzatılır veya kısaltılır. Ayrıca, bu iki kısım arasında hat örneklenir ve bu değer kontrolcü tarafından işlenir. Kontrolcüler bu kısımda anlık oluşabilecek gürültü kaynaklı hatalı örnekleme olmaması için birbirlerine yakın aralıklar ile hattı 3 kere örnekler ve çoğunluk oylaması (majority voting) ile hattın değerini belirler. PHASE_SEG1, PHASE_SEG2 1-8 zaman birimi değerinde olabilir. Örnek bir CAN bit zamanlaması Şekil 3.20'de verilmiştir. Bu örneğe göre 1 zaman birimi değerinin 100 ns olması halinde CAN düğümünün veri hızı 1 Mbps olacaktır.



Şekil 3.20 Örnek CAN Bit Zamanlaması [1]

3.2. CAN PROTOKOLÜNDE GÜVENLİK

3.2.1. CAN Güvenlik Zafiyetleri

CAN veri yolu yapısı gereği birden fazla güvenlik zafiyeti bulunmaktadır. Carsten et. al. çalışmasında açıkladığı gibi CAN paketleri içerisinde gönderici veya alıcı düğümlere ait adresleme veya benzeri bir bilgi bulundurmamaktadır [28]. Bu nedenle alıcı birimler paketleri sadece CAN ID bilgisine göre işler. Veri yolu üzerindeki tüm alıcı birimleri için bu geçerlidir. Alıcı düğümler gönderilen paketin geçerli bir göndericiden geldiğini anlayamamaktadır. Bu durum CAN veri yolunun açıklarından biridir. Düğümler diğer düğümlerin geçerli bir düğüm olduğunu doğrulayamamaktadır. Bu nedenle CAN düğümleri yetkisiz gönderilen paketlerin ayrımını yapamamaktadır [28].

Tipik bir CAN veri yolunda düğümler için kimlik doğrulama işlemi yapılmamaktadır. Düğümler gelen verilerin geçerli bir düğümden geldiğini doğrulayamamaktadır. Bu durum dışardan yapılmış izinsiz bir bağlantının başka bir düğüm gibi davranmasına olanak sağlar. Düğümler gelen paketin izinsiz bağlantıdan veya geçerli düğümden geldiğini anlayamayacaktır [29].

CAN veri yolunda gönderilen paketler düğümler tarafından şifreleme yapılmadan gönderilmektedir. Veri yolunda şifreleme olmaması da yetkisiz paket gönderilmesine olanak sağlar [29]. Ayrıca yetkisiz bir paket gönderimi olmasa bile sistemin izinsiz bir şekilde gözlemlenmesi ve incelemesine olanak sağlamaktadır.

CAN veri yolunda düğümler birbirlerine bir çift diferansiyel kablo ile birbirlerine bağlıdır. Bu durum, CAN veri yoluna dışardan izinsiz olarak herhangi bir noktadan bağlanmış başka bir düğümün, diğer tüm düğümler ile iletişim sağlamasına olanak sağlar. Günümüz araçlarında hemen hemen tüm ECU'lar tek bir ağ geçidi (gateway) üzerinden birbirlerine bağlıdır. Bu durum araçlar içerinde standart olarak gelen OBD-II portu üzerinden diğer tüm ECU'lara ulaşılmasını sağlar [29]. OBD-II araçlar içerinde hata veya arıza tespiti için kullanılsa da bu port üzerinden izinsiz ve yetkisiz giriş yapılmasına engel olacak bir sistem bulunmamaktadır.

CAN veri yolu, özellikle CAN 2.0 protokolü, sınırlı bant genişliğine sahiptir. En fazla 1 Mbps veri hızına ulaşabilen CAN veri yolu sistemleri bu düşük veri hızlarında etkin bir kimlik doğrulama sistemi geliştirilmesini zorlaştırmaktadır [29]. Ayrıca protokolde bir paket içerisinde en fazla 64 bit (8 byte) gönderilebilmesi de paketlerin şifrelenmesi işlemini

sınırlandırmaktadır. Bunun sebebi ise şifreleme için ekstra (redundant) veri gönderilmesidir. Bir paketin veri alanının şifrelenmesi CAN veri yolunun bant genişliğinin azalmasına sebep olmaktadır.

3.2.2. CAN Veri Yoluna Gerçekleştirilen Ataklar

Boyes et. al. araçlardaki bağlantı arayüzlerinin artması nedeniyle oluşabilecek güvenlik zafiyetlerine dikkat çekmiştir. Araçlarda bulunan multimedia eğlence sistemlerinin USB, WiFi, Bluetooth gibi bağlantı protokollerini bulundurması ayrıca bu sistemlerin araç içerisindeki CAN veri ağına da bağlı olmasından dolayı birden fazla atak gerçekleştirilebilecek potansiyel bağlantı yüzeylerinin olduğunu belirtmiştir [30] [8].

3.2.2.1. Veri Çalma

CAN veri ağından veri çalma birçok farklı şekilde yapılabilmektedir. Örneğin, OBD-II portuna bağlanabilecek kablosuz bir CAN paket okuyucu üzerinden herhangi yetkisiz bir giriş sağlanabilmekte ve aracın ürettiği veriler çalınabilmektedir. Ayrıca, otomotiv sigorta firması Progressive "Snapshot" adında bir cihaz sağlamakta ve bu cihaz OBD-II portu üzerinden sürücülerin araç kullanım bilgilerine erişebilmektedir [28].

3.2.2.2. Kontrol Geçersiz Kılma

CAN veri yolundaki potansiyel açıkların bulunması için birçok çalışma yapılmıştır. Kosher et. al. araçların güvenlik açısından kritik olan ekipmanlara, fiziksel, fiziksel olmayan, kısa ve uzun menzil yetkisiz girişler ile verilerin manipüle edilebildiğini ortaya koymuşlarıdır [2] [8]. OBD-II portu üzerinden yapılmış olan ataklar ile hız sensörlerini manipüle etmişlerdir. Ayrıca araç kapılarının ve motorun kontrolünü bu port üzerinden ele geçirmeyi başarmışlardır.

Bunlara ek olarak 2015 yılında Valsaek et. al. Jeep Cherokee marka bir aracın uzaktan yetkisiz bir bağlantı ile CAN veri sistemine gönderilen üretilmiş paketler ile aracın kontrolüne etki edilebileceğini kanıtlamışlardır. Bu denemede Valsaek et. al. aracın frenleme sistemleri gibi kritik ekipmanlarına erişim sağlamışlardır [4].

3.2.2.3. Veri Saptırma

Bu tür ataklarda, CAN veri yolu üzerinden sürücüye iletilen araç parametreleri değiştirilerek iletilir. Bunun sonucunda sürücünün aracı hatalı kullanması sağlanır. Buna örnek olarak, Hope et. al. araçlarda üretilen uyarı mesajlarını baskılamayı başarmışlardır [31]. Araçlarda bulunan hava yastığı kontrol sistemleri, araç içerisinde hava yastığının olduğunu ve sorunsuz olduğunu kontrol eder. Hava yastığının bulunmaması durumunda sistem sürücüye uyarı bildirimleri ile bilgilendirme yapmaktadır. Hope et. al. aracın hava yastığı kontrol sisteminin hata kontrolü zamanında ürettiği paketleri toplayarak bu sistemin sürücüye hatalı uyarı bildirimi yapmasını sağlamışlardır.

3.2.2.4. Harici Sensör Atakları

Bu tür ataklarda, araçlarda bulunan sensörlerin CAN veri yolu üzerinden gönderdiği paketlerin verileri değiştirilerek aracın otomatik olarak veya sürücünün gerçekleştirdiği hatalı kullanımlardır. Örnek olarak, uzaklık sensörünün verileri değiştirilerek aracın ani frenleme sisteminin devreye sokulması veya sürücünün olmadığı halde fazla yakınlık uyarısından ani frene basması problemlere sebep olabilir [28].

4. TASARIM ÇALIŞMASI

Bu tezde, birbirine CAN veri yolu ile bağlı olan düğümlerin fiziksel seviyedeki sinyalleri kullanılarak bir IDS metodu geliştirilmesi amaçlanmıştır. Metodun geliştirilmesi için CAN veri yolunun sinyalleri incelenmiştir. Sinyallerin elektriksel olarak farklılık gösterdiği veya göstereceği tahmin edilen kısımlar incelenmiştir.

CAN veri yolunda, Bölüm 3.1.3'te anlatıldığı gibi, düğümler birbirlerine diferansiyel bir hat üzerinden bağlanmaktadır. Bu sistemde her bir düğüm veri yolu üzerinde CAN protokolüne uygun olarak veri gönderebilirler. Sistemde herhangi bir düğüm veri göndermediği sürece hat daimi olarak çekinik olarak tanımlanmış sinyal durumunda beklemektedir. Bu durum elektriksel olarak CAN-H ve CAN-L olarak isimlendirilmiş hatların voltaj seviyesini belirlemektedir. CAN veri yolunda hattın çekinik veya baskın olma durumumu CAN-H ve CAN-L hatlarının arasındaki farka göre hesaplanır. Çekinik durumunda CAN-H ve CAN-L sinyalleri yaklaşık olarak 2.5V seviyesindedir. Hattın diferansiyel olması sebebiyle CAN-H ve CAN-L sinyallerinin farkı 0 volt olmaktadır. Baskın sinyal durumunda ise CAN-H hattı yaklaşık 1 Voltu yükselerek 3.5 Volt seviyelerine ulaşır. CAN-L hattı ise yaklaşık 1 Volt düşerek 1.5 Volt seviyelerine bu durumda voltaj farkı 2 Volt değerlerine yaklaşır. Sistemde hatlar arası voltaj farkı 0.5 Volttan daha az ise hat çekinik durumdadır ve CAN alıcı-verici birimlerinin çıkışı lojik '1' olur. Hatlar arası voltaj farkı 0.9 Volt veya üzeri ise sistem baskın durumdadır ve CAN alıcı-verici birimlerinin çıkışı lojik '0' olur.

CAN veri yolunda elektriksel sinyaller genel hatlarıyla belirlenmiş olsa da bu sinyal durumları kullanılan alıcı-verici (transciever) entegrelerinin iç yapısından, bu entegrelerin çıkış empedansından, çıkış tamponlarının(buffer) yükselme ve alçalma zamanlamalarından, CAN kontrolcülerinin kullandığı saat frekanslarından, hattaki gürültü vb. gibi birçok nedenden dolayı sinyaller farklılık göstermektedir. Bu farklılıklar çoğunlukla donanımsal farklılıklardan kaynaklıdır ve sinyallerin farklılıkları deterministiktir. Bu tezin konusu, donanımsal kaynaklı sinyallerde oluşan farklılıkların yakalanıp bir CAN veri yolundaki düğümlerin sinyal karakteristiklerinin yapay sinir ağına öğretilmesi ve saldırı amaçlı dışarıdan bağlanabilecek farklı bir cihazın sistemde oluşturabileceği farklı sinyal karakteristiğinin fark edilmesi üzerinedir. Metodun ilk aşamasında farklı alıcı-verici birimlerinin sinyal karakteristiği incelenmiştir. İnceleme sonucunda sinyallerin farklılık gösteren kısımlarının yakalanıp kayıt edilmesi amaçlanmıştır. Kayıt edilen veriler üzerinden

yapay sinir ağının eğitilmesi ve eğitilmiş yapay sinir ağı ile dışardan sisteme sonradan bağlanmış saldırgan bir CAN düğümünün tespit edilmesini içermektedir. Metottun gerçeklenmesinde FPGA teknolojisinden faydalanılmıştır. Bu sayede sinyallerdeki yüksek hızlı değişimlerin yakalanabileceği öngörülmüştür.

Metottun uygulanması kapsamında FPGA üzerinde VHDL donanım betimleme dili ile CAN 2.0B standardını destekleyen bir IP çekirdek ürünü geliştirilmiştir. Bu IP çekirdeği standart bir CAN protokolü ile çalışabilmenin yanında, sinyalin özellikle zamanlama karakteristiğini de çıkarabilme kabiliyetine sahiptir. Bu IP çekirdeğine ek olarak, CAN veri yoluna harici veya FPGA'ler içerisinde dahili olabilen bir Analog-To-Digital (ADC) dönüştürücü bağlanmıştır. Bu ADC CAN sinyallerin, yükselme-açalma zamanlamaları, çekinik ve baskın durumdaki voltaj seviyeleri gibi düğümden düğüme az da olsa farklılık gösterebilecek özelliklerini yakalama için kullanılmıştır. Çıkarılan özellikler geliştirme için FPGA (SoC) cihazından, UART arayüzü gibi veri yolu ile bilgisayara aktarılması sağlanmış olup, geliştirmeler bilgisayar ortamında MATLAB [32] programı kullanılarak yapılmıştır. CAN veri yolu simülasyonu için farklı CAN kontrolcüleri ve farklı CAN alıcı-verici entegreleri kullanılmıştır. Sistemin gerçeklemesi için kullanılan cihaz ve ekipmanlar bölüm 4.1'de verilmiştir. CAN sinyalinin incelenmesi bölüm 4.2'de anlatılmıştır. Tez kapsamında geliştirilen CAN IP çekirdeği bölüm 4.3'te anlatılmıştır. Sistemin benzetim çalışmaları bölüm 4.4'te verilmiştir. CAN veri yolunun entegrasyonu ve IDS sistemi için veri sentezlemesi çalışmaları bölüm 4.5'te açıklanmıştır. Son olarak IDS sistemi için çalışılan yapay sinir ağı algoritmaları ve mimarileri bölüm 4.6'da anlatılmıştır.

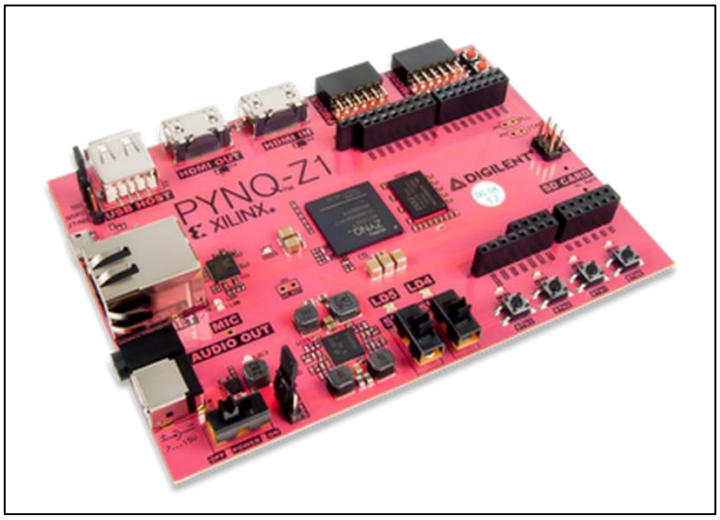
4.1. Kullanılan Ekipmanlar

Bu tezin uygulaması kapsamında birden farklı ekipman kullanılmıştır. Farklı ekipman ve entegrelerin kullanılması nedeni ise farklı parçaların sistemin performansını etkileyebileceği öngörüsüdür.

4.1.1. PYNQ Geliştirme Kartı (ZYNQ 7020 SoC)

Metodun gerçeklemesi için Digilent firması tarafından geliştirilmiş olan PYNQ geliştirme kartı kullanılmıştır. Bu kart üzerinde Xilinx firması tarafından üretilen ZYNQ 7020 SoC entegresi bulunmaktadır. Bu entegre çift çekirdek Arm Cortex A9 işlemcisini bulundurur. Ayrıca entegrede orta sınıf bir FPGA lojik kısmı bulunmaktadır. Hem FPGA hem de işlemci kısmı ayrı ayrı programlanabilmektedir. FPGA ve işlemciler arasında AXI-4 (AMBA) veri yolu tümleşik olarak çipin içerisinde yer almaktadır. FPGA kısmı istendiği

takdirde bağımsız olarak kullanılabilmektedir veya FPGA kısmı işlemcinin özelliklerini arttırma amacıyla da kullanılabilmektedir. Örneğin, bu çip içerisindeki işlemci birimine ayrılmış 2 adet Ethernet arayüzü bulunmaktadır; ancak FPGA lojik içerisinde Ethernet kontrolcüsü tanımlanarak, işlemciye AMBA veri yolu üzerinden bağlanabilir. Bu sayede işlemcinin fonksiyonelitesi arttırılabilmektedir.



Şekil 4.1 PYNQ Geliştirme Kartı

PYNQ geliştirme kartınındaki SoC çipinin içerisinde bulunan işlemcinin özellikleri Tablo 4.1'de FPGA'nın özellikleri ise Tablo 4.2 verilmiştir.

Tablo 4.1 ZYNQ XC7Z020-1CLG400C SoC İşlemci Özellikleri [33]

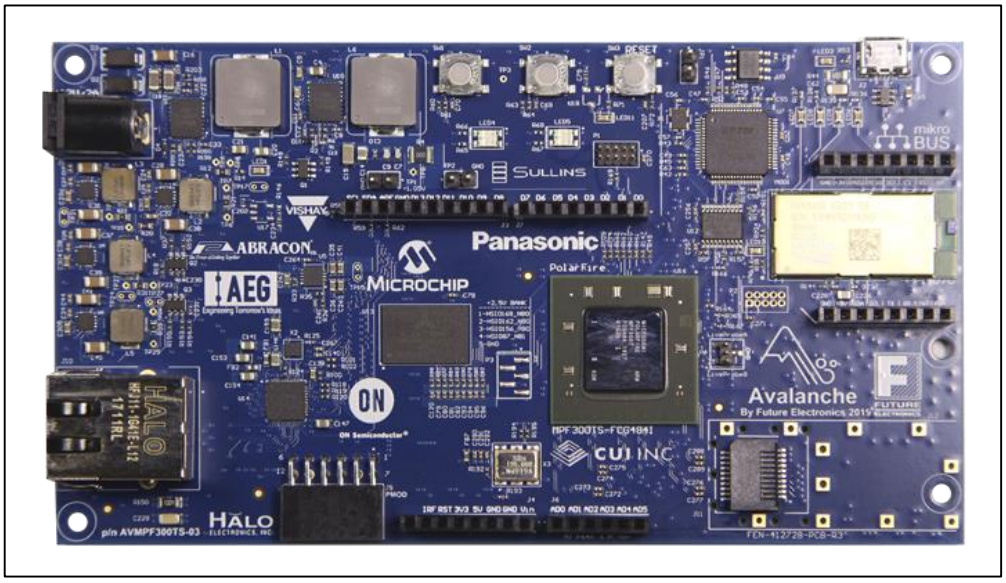
ARM Özellik	Açıklama
İşlemci	Dual Core Arm Cortex A9
L1	32 KB Data – 32 KB Instruction
L2	512 KB
Dahili Hafıza	256 KB
Harici Hafıza	512 MB DDR3
DMA Kanalı	8
Düşük Hızlı Çevresel Arayüzler	2x UART 2x CAN 2.0B 2x I2C 2x SPI 4x 32b GPIO
Yüksek Hızlı Çevresel Arayüzler	2x USB 2.0 (OTG) 2x Tri-mode Gigabit Ethernet 2x SD/SDIO
İşlemci FPGA Arası Bağlantı Arayüzleri	2x AXI 32b Master, 2x AXI 32b Slave 4x AXI 64b/32b Memory AXI 64b ACP 16 Interrupts

Tablo 4.2 ZYNQ XC7Z020-1CLG400C SoC FPGA Özellikleri [33]

FPGA Özellik	Açıklama
FPGA	Artix-7
Logic Cell	85K
LUT	53200
Flip-Flop	106400
Block RAM	4.9 Mb
DSP Slice	220
ADC	2x 12 bit 1 MSPS

4.1.2. Avalanche Geliştirme Kartı (AVMPF300TS)

Metodun gerçeklemesi kısmında CAN düğümü sayısını arttırmak adına farklı FPGA geliştirme kartları kullanılmıştır. Bu FPGA kartı Future Electronics firması tarafından geliştirilmiş olup üzerinde Microsemi firmasının PolarFire MPF300TS çipini bulundurmaktadır. ZYNQ FPGA'den farklı olarak bu çipte sadece FPGA lojik bulunmaktadır. Bu çip flash tabanlı bir FPGA olması ve iç yapısının farklı olması sebebiyle tercih edilmiştir. Tez kapsamında tasarlanan CAN IP çekirdeğinin farklı bir FPGA'ya entegre edilerek fazladan bir CAN düğümü olarak kullanılmıştır. Avalanche geliştirme kartının görseli Şekil 4.2'de verilmiştir.



Şekil 4.2 Avalanche Geliştirme Kartı [34]

Avalanche geliştirme kartında bulunan PolarFire FPGA özellikleri Tablo 4.3'te verilmiştir.

Tablo 4.3 PolarFire MPF300TS Özellikleri [35]

Özellik	Açıklama
FPGA	PolarFire MPF300TS
Logic Element (4-LUT DFF)	300K
LSRAM (20Kb)	952
Math Block	924
uSRAM (64x12)	2772
PLL/DLL	8
PCIe Gen2	2
SERDES (lane)	16

4.1.3. Digilent PMOD CAN Modülü

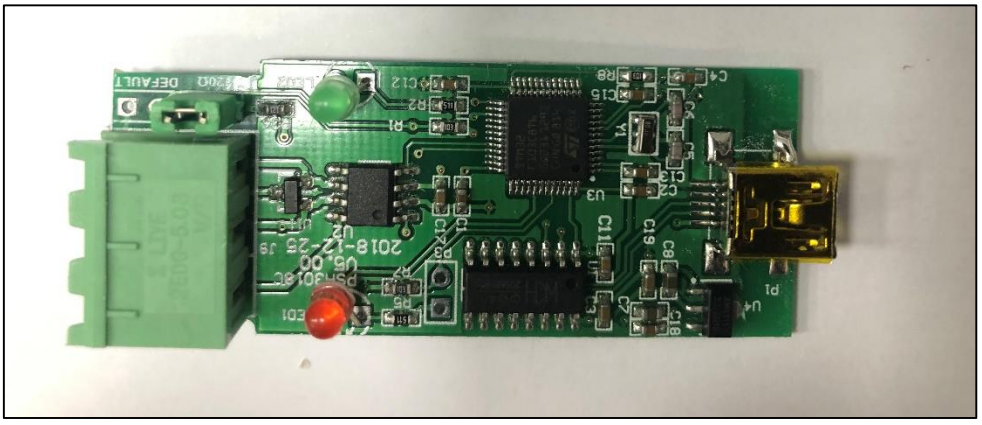
Araçta olabilecek bir CAN veri yolunun simüle edilmesi için sisteme farklı tasarımlarda CAN düğümleri eklenmesi gerekliliği ortaya çıkmıştır. Bu nedenle hazır bir CAN kontrolcüsü içeren Digilent firmasının geliştirdiği 3 adet PMOD CAN kontrolcüsü kullanılmıştır. Bu ekipman üzerinde Microchip firmasının ürettiği MCP25625 CAN kontrolcüsü ve tümleşik alıcı-verici entegresi bulunmaktadır. CAN protokolü çip içerisinde gerçeklenmiştir. Cihazın kullanılabilmesi için kartın PMOD (peripheral module interface) adı verilen bir bağlantı arayüzünden geliştirme kartına bağlanması gerekmektedir. Geliştirme kartı, MCP25625 entegresi ile PMOD konnektörü vasıtasıyla SPI protokolü ile haberleşmektedir [36]. CAN hattı üzerinden alınan paketler yine SPI üzerinden geliştirme kartına iletilir. Gönderilecek CAN paketleri ise geliştirme kartı ile SPI üzerinden çipin ilgili register'larına yazılarak gönderimi sağlanır. PMOD CAN cihazı Şekil 4.3'te gösterilmiştir.

Şekil 4.3 Digilent PMOD CAN

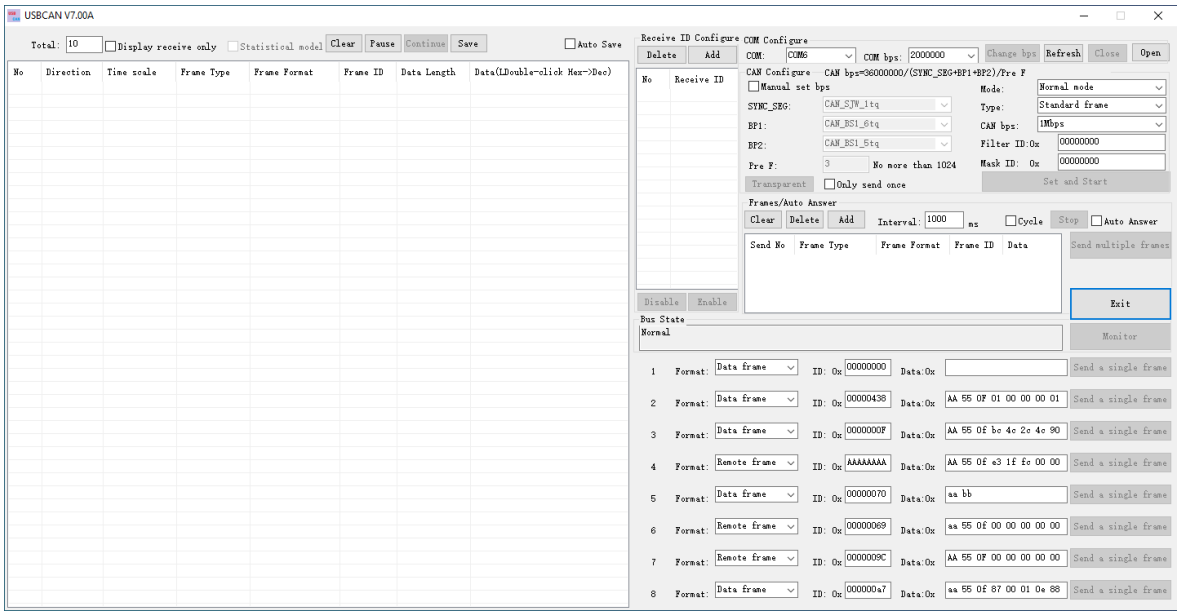
4.1.4. USB CAN Çevirici Modül

CAN IP çekirdeği geliştirmesinde hızlı prototipleme için USB CAN çeviriciler kullanılmıştır. Bu çevirici vasıtasıyla, hazırlanmış CAN kontrolcüsünün paket gönderme ve alma fonksiyonları test edilmiştir. Basit bir arayüze sahip olan bu cihaz bilgisayar ile UART arayüzü ile haberleşmektedir. Cihaz içerisinde STMicroelectronics firmasına ait STM32F103C8T6 ARM tabanlı işlemcisi bulunmaktadır. Bu işlemcinin içerisinde dahili CAN kontrolcüsü bulunmaktadır. Bu işlemciye bağlı MCP2551 CAN alıcı-verici entegresi bulunmaktadır [37]. İşlemcinin yazılımı üretici firma tarafından yüklü olarak gönderilmektedir. Cihazın bilgisayar ile haberleşmesi için grafiksel kullanıcı arayüzü (GUI) programı bulunmaktadır. Fakat tez kapsamında paket gönderim-alım işleminin otomatik olarak sümüle edilebilmesi için Python dilinde scriptler de hazırlanmıştır. USB CAN dönüştürücünün görseli Şekil 4.4'te, iç yapısı Şekil 4.5'te verilmiştir. Cihazın GUI programı ise Şekil 4.6'da gösterilmiştir.

Şekil 4.4 USB CAN Dönüştürücü Modülü [38]



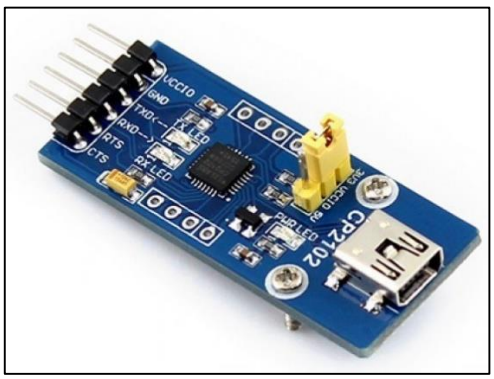
Şekil 4.5 USB CAN Dönüştürücü İç Yapısı [38]



Şekil 4.6 USB CAN Grafiksel Kullanıcı Arayüzü

4.1.5. USB UART Çevirici Modül

Tez kapsamında CAN veri yolundaki örneklenen verileri bilgisayar üzerindeki MATLAB programına aktarmak için USB UART çevirici kullanılmıştır [39]. UART protokolünü destekleyen bu modül FPGA'dan alınan verileri Python programına aktar aktarılmasını sağlar. Alınan veriler daha sonra MATLAB programı için .mat dosyasına çevirilmiştir. Kullanılan USB UART çeviricinin görseli Şekil 4.7'de gösterilmiştir.

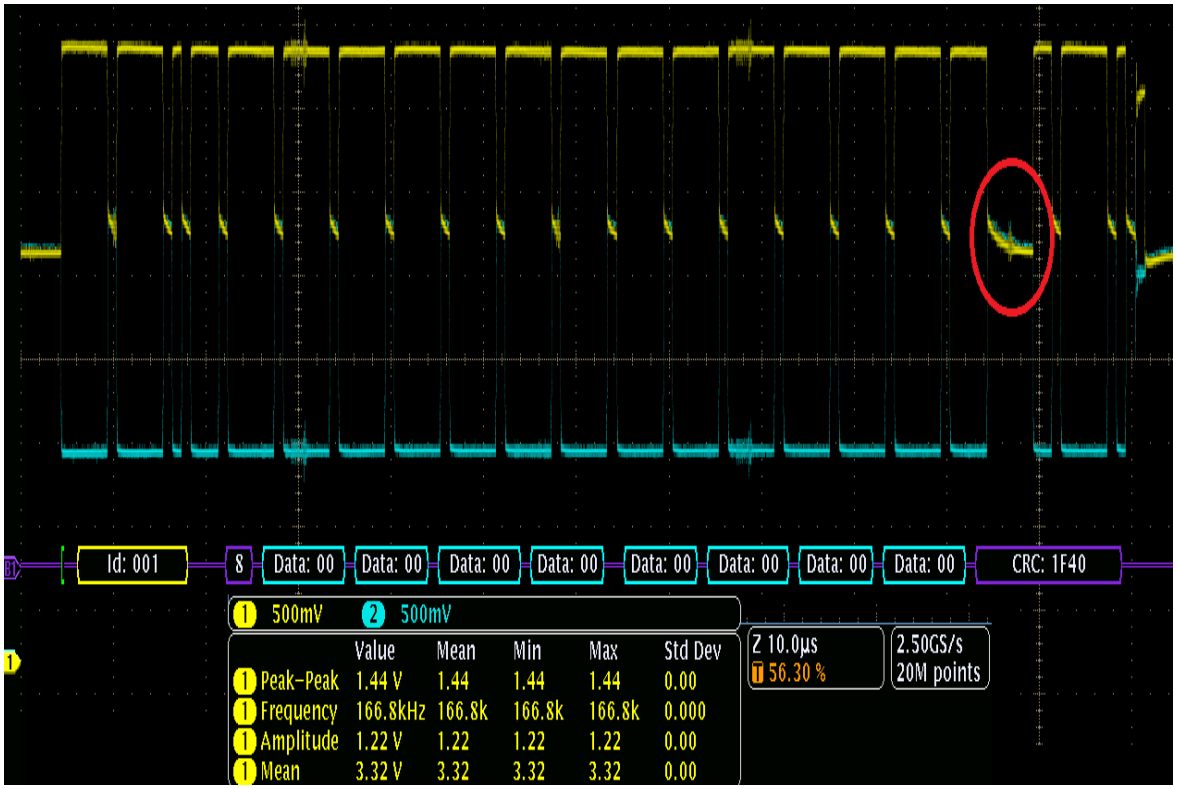


Şekil 4.7 USB UART Çevirici Modül [39]

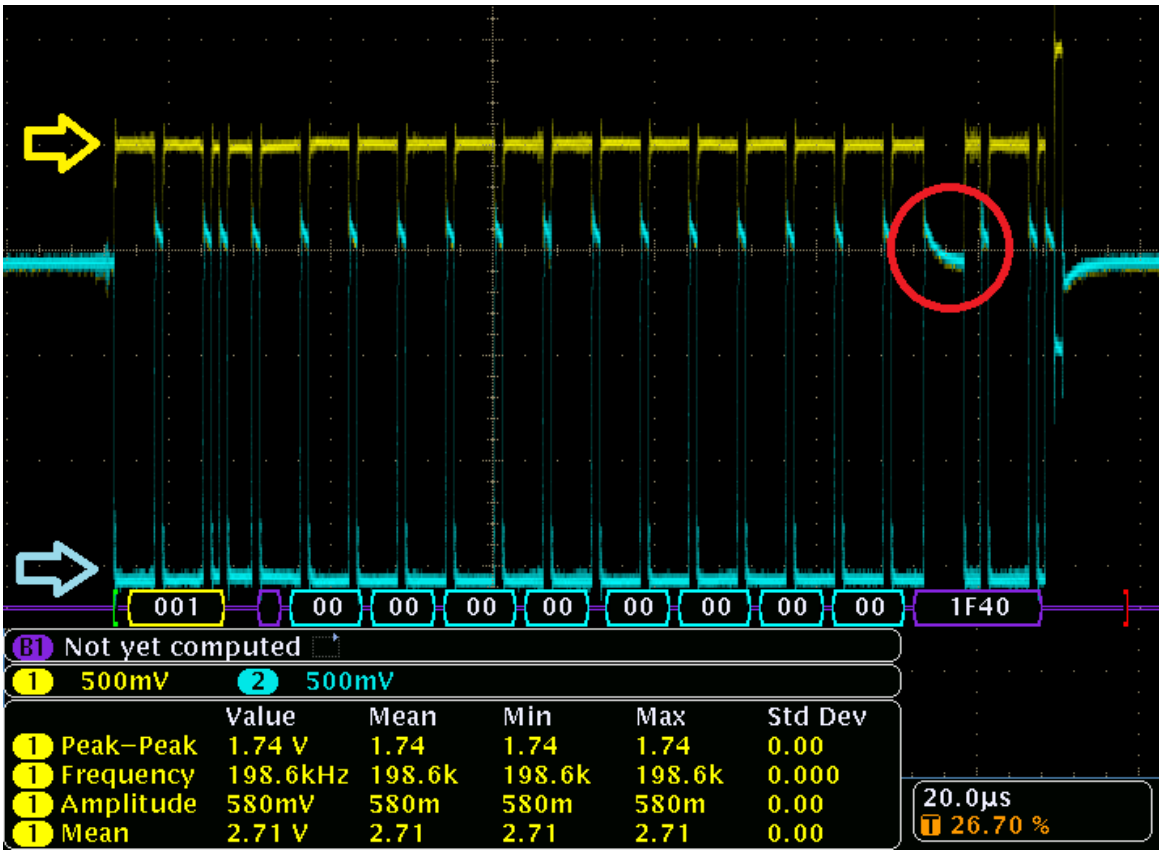
4.2. CAN Sinyalinin İncelenmesi ve Sinyal Parmak İzi

Tezin uygulaması bölümünden önce, CAN sinyallerinin farkının algılanabilmesi için CAN veri yolu üzerindeki düğümlerin oluşturdukları elektriksel sinyaller incelenmiştir. Bu incelemenin amacı, kontrolcü ve alıcı-vericilerin farklılıklarından kaynaklı sinyallerde

oluşmuş olan farklılıkların ortaya çıkarılması ve yapay sinir ağının eğitimi için kullanılacak özellik vektörlerinin çıkarılmasıdır. Şekil 4.8'de geliştirilmiş olan CAN IP çekirdeğinin PYNQ FPGA kartı üzerinde çalıştırılması ile elde edilen CAN sinyali gösterilmiştir. Bu kontrolcü FPGA üzerinde özel yapım bir deneme kartı üzerinde bulunan SNV65HVD CAN alıcı-verici çipine bağlıdır. Kontrolcü, alıcı-verici entegresine CAN verisini tek uçlu (single-ended) olarak iletir. CAN alıcı-verici çip bu sinyali diferansiyel sinyale çevirir. Ölçüm yapılan kısım SVN65HVD alıcı-verici entegresinin CAN-H ve CAN-L çıkışlarıdır. Yapılan ilk gözlemlerde, veri yolunun baskın durumdan çekinik sinyal durumuna geçişlerden sonra CAN-H ve CAN-L hatlarının, sinyalin orta voltaj durumundan daha üst bir voltaj durumunda olması ve bu voltaj değerinden veri yolunun orta değerine üstel fonksiyon ($e^{-}\alpha t$) benzeri bir düşüş gerçekleştirmesidir (Şekil 4.8 Kırmızı ile işaretlenmiş kısım). Bu durumun alıcı-verici entegresinin iç yapısıyla ilgili olduğu düşünülmüş ve bu gibi farklılık gösterebilecek kısımların sinyallerin tanınmasında kullanılabileceği düşünülmüştür.

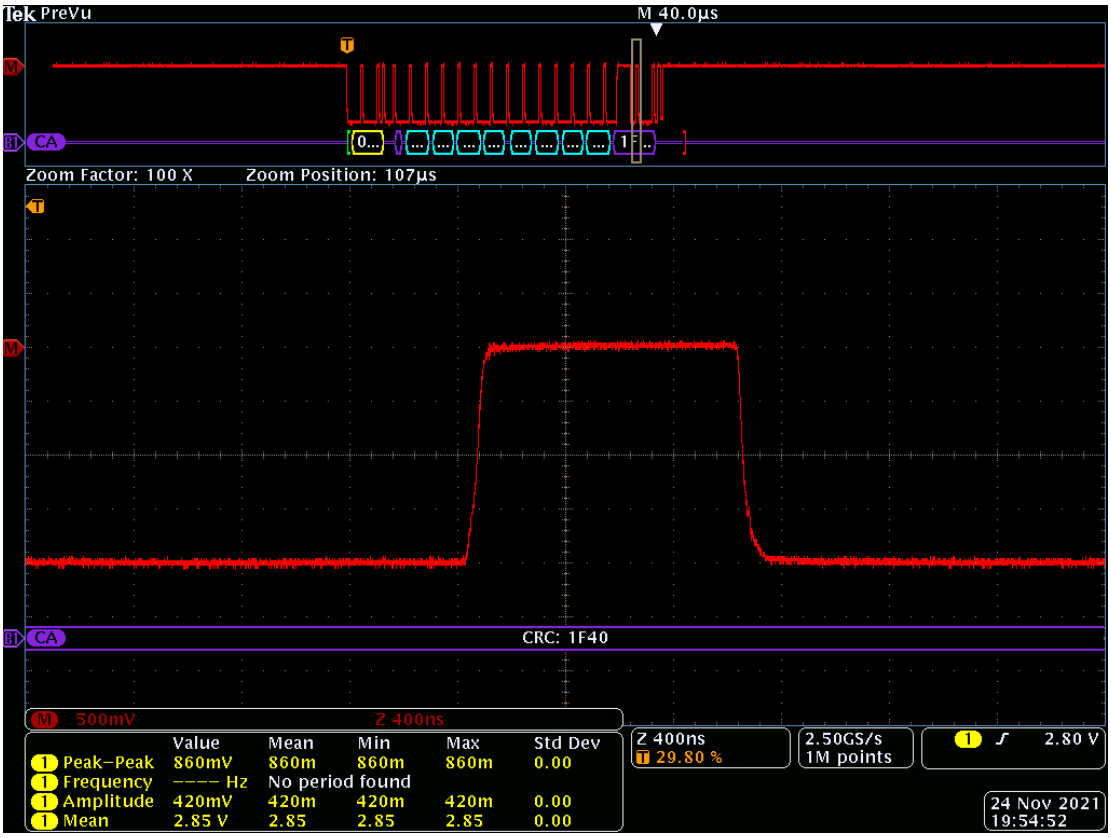


Şekil 4.8 Geliştirilen CAN IP Sinyali

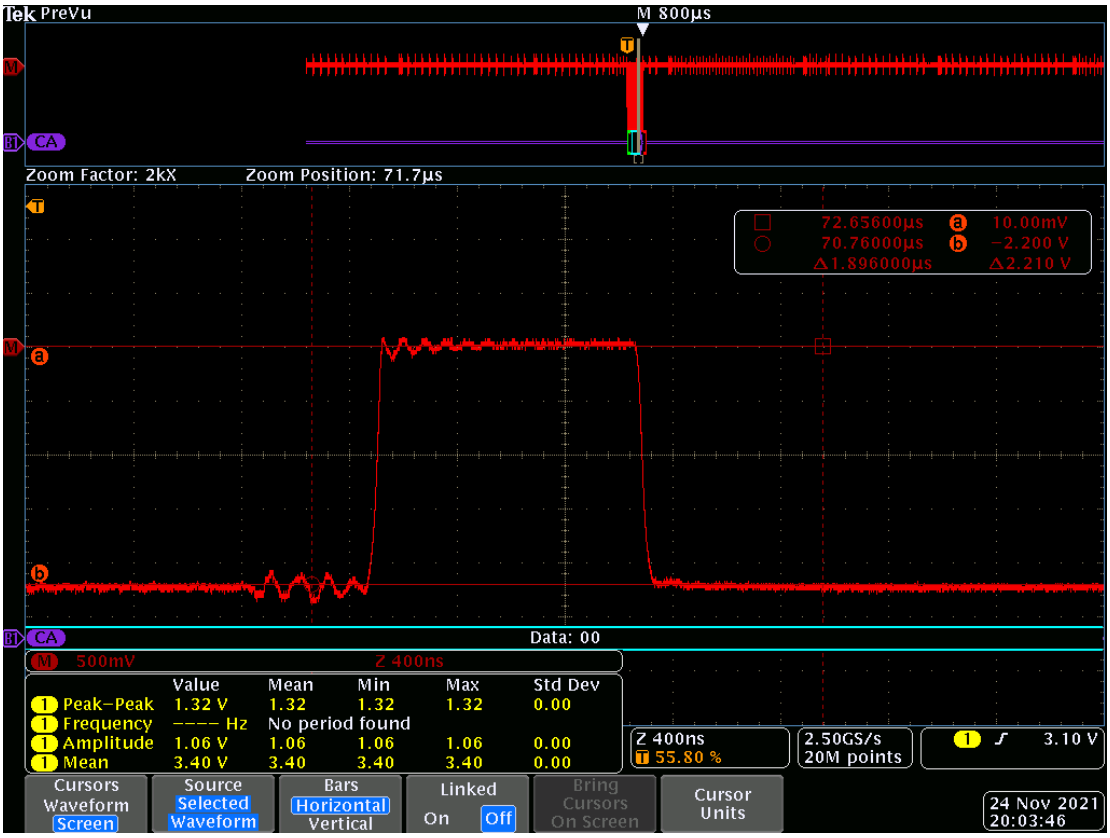


Şekil 4.9 USB CAN Dönüştürücü CAN Sinyali

USB CAN dönüştürücü ekipmanından alınan sinyal ise daha farklı bir yapıdadır. CAN protokolünün gerekliliklerini yerine getirmektedir ancak baskın durumdan çekinik duruma geçişten sonra sinyalin orta voltaj değerine yavaş bir şekilde düşüş bu alıcı-vericide de gözlemlenmiştir. Bu düşüşünde MCP2551 alıcı-vericisinde daha hızlı olduğu gözlemlenmiştir. Buna ek olarak MCP2551 tümleşik devresinin (IC) CAN-H ve CAN-L hatlarının baskın durumda daha farklı voltaj seviyelerinde olduğu gözlemlenmiştir. CAN-H voltaj değeri SNV65HVD tümleşik devresinde baskın sinyal durumunda yaklaşık 3.5 Volt iken MCP2551 tümleşik devresinde bu değer 3 Volt değerindedir. CAN-L sinyali ise SVN65HVD tümleşik devresinde 1.5 Volt, MCP2551 tümleşik devresinde ise 1 Volt değerindedir. Her iki tümleşik devre için diferansiyel voltaj farkı baskın durumda yaklaşık 2 Volttur bu nedenle kontrolcü kısmına bu durumun bir etkisi yoktur. Fakat sinyallerin farklılıkları düğümlerin ayırt edilmesinde kullanılabileceği düşünülmüştür. Şekil 4.10'da SVN65HVD tümleşik devresinin, Şekil 4.11 MCP2551 tümleşik devresinin diferansiyel çıktısı verilmiştir.



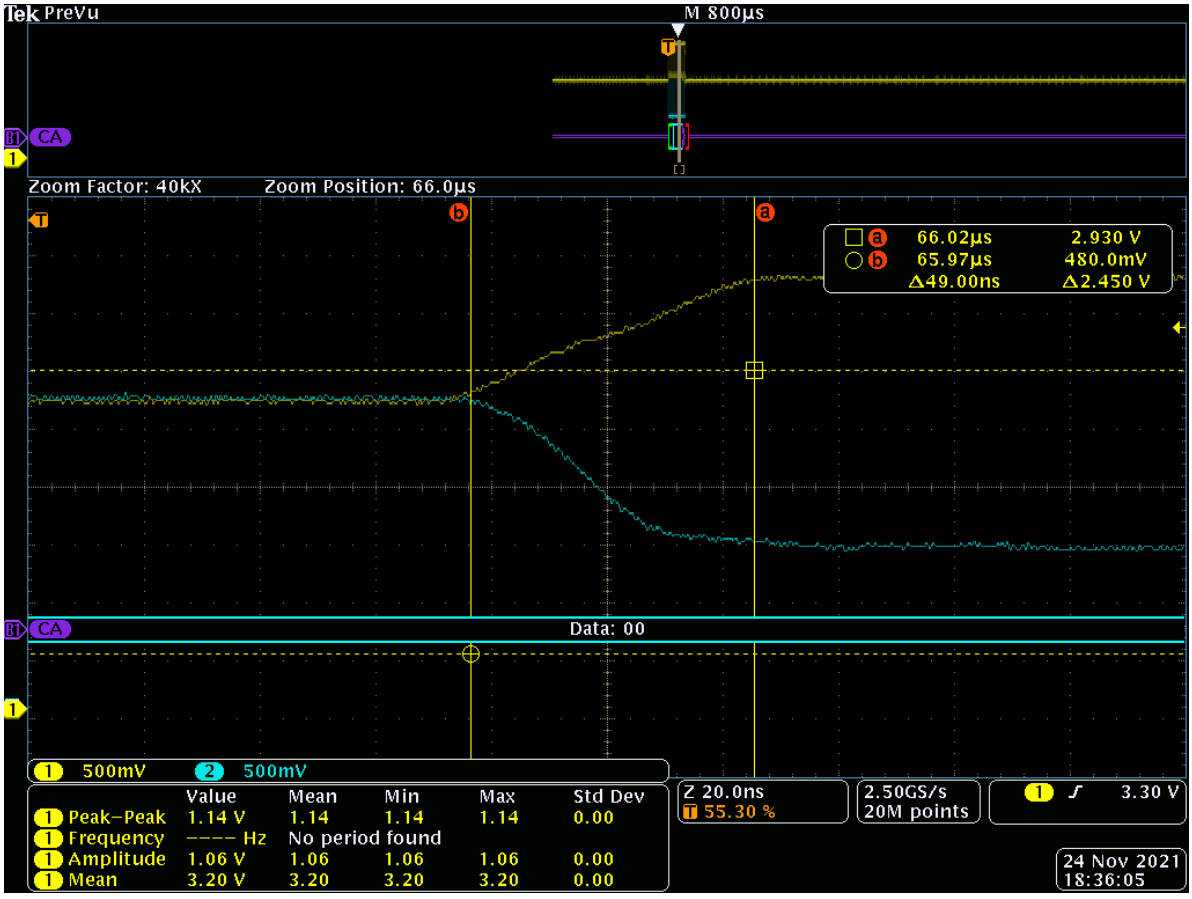
Şekil 4.10 SVN65HVD CAN Alıcı-Verici Diferansiyel Sinyali



Şekil 4.11 MCP2551 CAN Alıcı-Verici Diferansiyel Sinyali

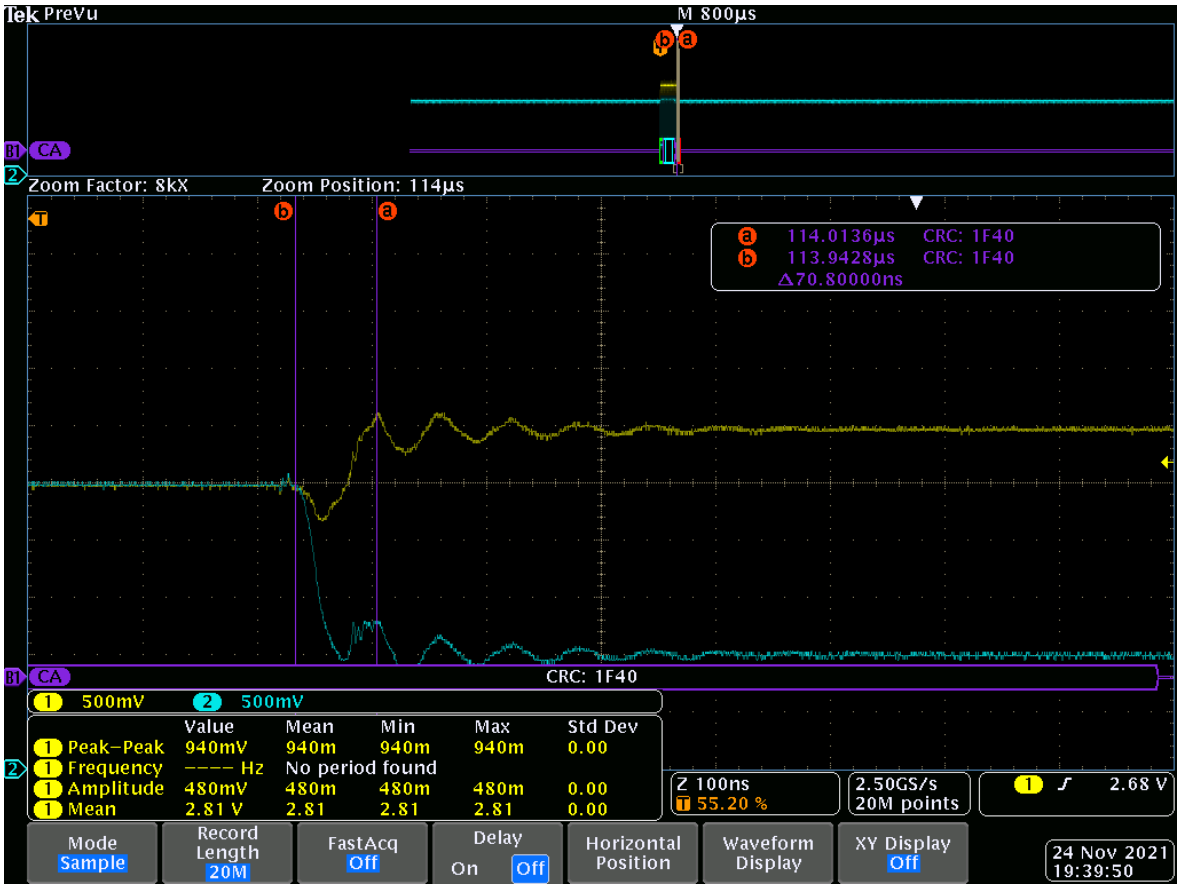
MCP2551 tümleşik devresinin, SVN65HVD tümleşik devresine göre daha sinyalin değişim kısmında daha değişken bir yapıda olduğu gözlemlenmiştir. Bu değişimin tez kapsamında kullanılabilirlik durumu bölüm 5'te ele alınmıştır.

Alıcı-verici tümleşik devrelerinden kaynaklı oluşabilecek farklılıklardan biri de sinyalin değişim süresidir. Sinyalin bir voltaj değerinde başka bir voltaj değerine geçişi kullanılan alıcı-vericinin iç yapısına bağladır. Ayrıca, bazı CAN alıcı-verici tümleşik devrelerinde sinyalin kalkış ve iniş süresi değiştirilebilmektedir. Bu ayar sinyalin kalkış ve iniş eğimini değiştirmekte ve sistemin elektromanyetik girişim performansını da etkilemektedir [37] [40]. Kalkış ve iniş süreleri sistemin veri hızını da etkilemektedir. Sinyalin çok yavaş değişmesi veri hızını düşürmektedir.



Şekil 4.12 SVN65HVD CAN Alıcı-Verici Sinyal Kalkış İniş Süresi

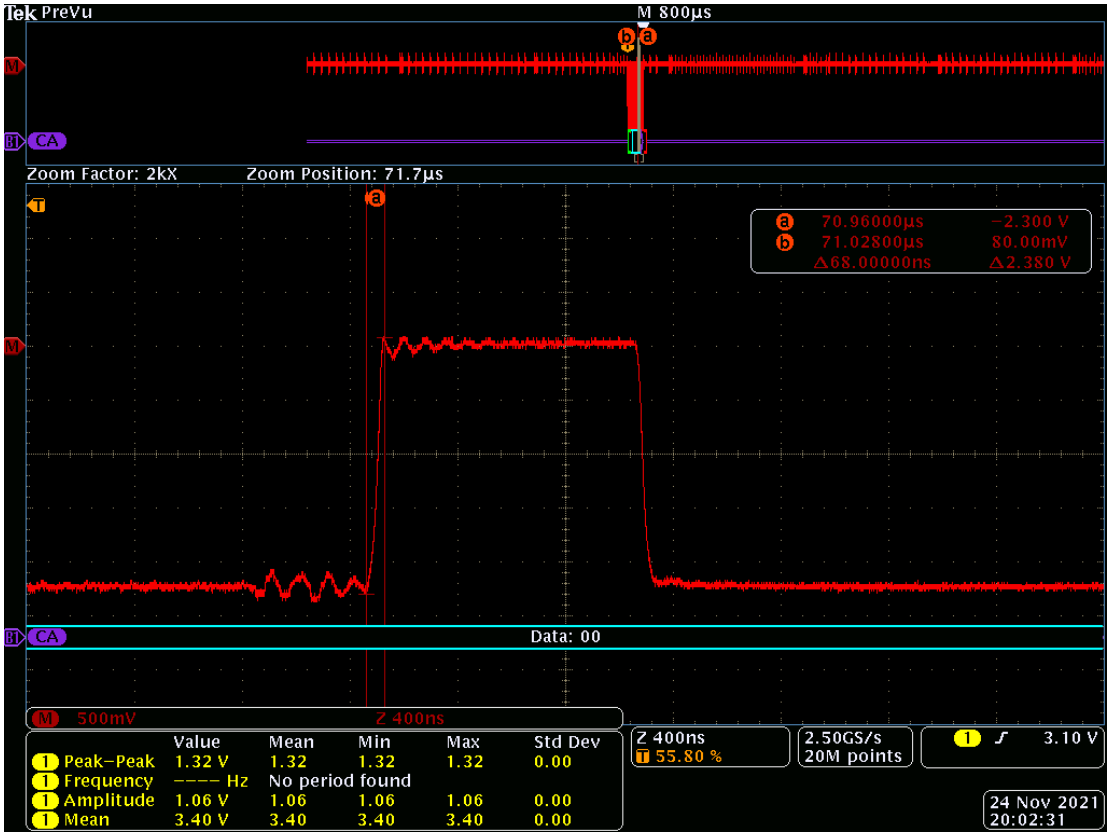
Eğim ayarı yapılmamış (yüksek hız) SVN65HVD tümleşik devresinin CAN-H ve CAN-L sinyallerinin kalkış-iniş süreleri Şekil 4.12'de verilmiştir. CAN-H sinyalinin kalkış süresi yaklaşık 50 (49) ns olarak gözlemlenmiştir. Buna ek olarak CAN-L sinyalinin iniş süresi ise CAN-H sinyalinin kalkış süresine oranla fark edilir şekilde daha küçüktür.



Şekil 4.13 MCP2551 CAN Alıcı-Verici Sinyal Kalkış İniş Süresi

USB CAN dönüştürücü içerisinde bulunan ve eğim ayarı yapılmamış (yüksek hız) MCP2551 CAN alıcı-verici tümleşik devresinin oluşturduğu sinyalin kalkış ve iniş süresi Şekil 4.13'te verilmiştir (Eğim ayarı yapılmadığı USB CAN dönüştürücüsü içerisindeki MCP2551 devresinin Rs pininin GND ile bağlantısı ölçülerek teyit edilmiştir). Buna göre bu devrenin kalkış ve iniş yapısı SN65HVD tümleşik devresine göre daha dalgalı yapıdadır ve bu durum her sinyal değişiminde gözlenmektedir. Ayrıca kalkış ve iniş süresinin diğer tümleşik devreye göre daha yavaş olduğu anlaşılmıştır. Dikkat çeken bir durum ise CAN-H sinyalinin, sinyal değişiminin ilk safhalarında voltajının düşmüş olmasıdır. Bu durumdan kaynaklı sinyalin kalkış süresinin ölçümünün doğru noktadan yapılmamış olması ihtimali düşünülmüştür. Sinyalin kalkış süresinin ölçümü için, osiloskop girişlerinin farkı osiloskop üzerinde alınarak tekrar çizdirilmiştir. Bu fark sinyali üzerinden yapılmış ölçüm Şekil 4.14'te gösterilmiştir. Daha önce CAN-H ve CAN-L sinyalleri üzerinden yapılmış kalkış süresi ölçümü, yapılmış olan diferansiyel ölçümü ile tutarlıdır. Buna göre MCP2551 alıcı-verici tümleşik devresinin kalkış süresi yaklaşık 70 ns ölçülmüştür. Sinyallerin değişim süresinin bu derece kısa olması nedeniyle 1 MSPS bir ADC tümleşik devresi ile değişimlerin

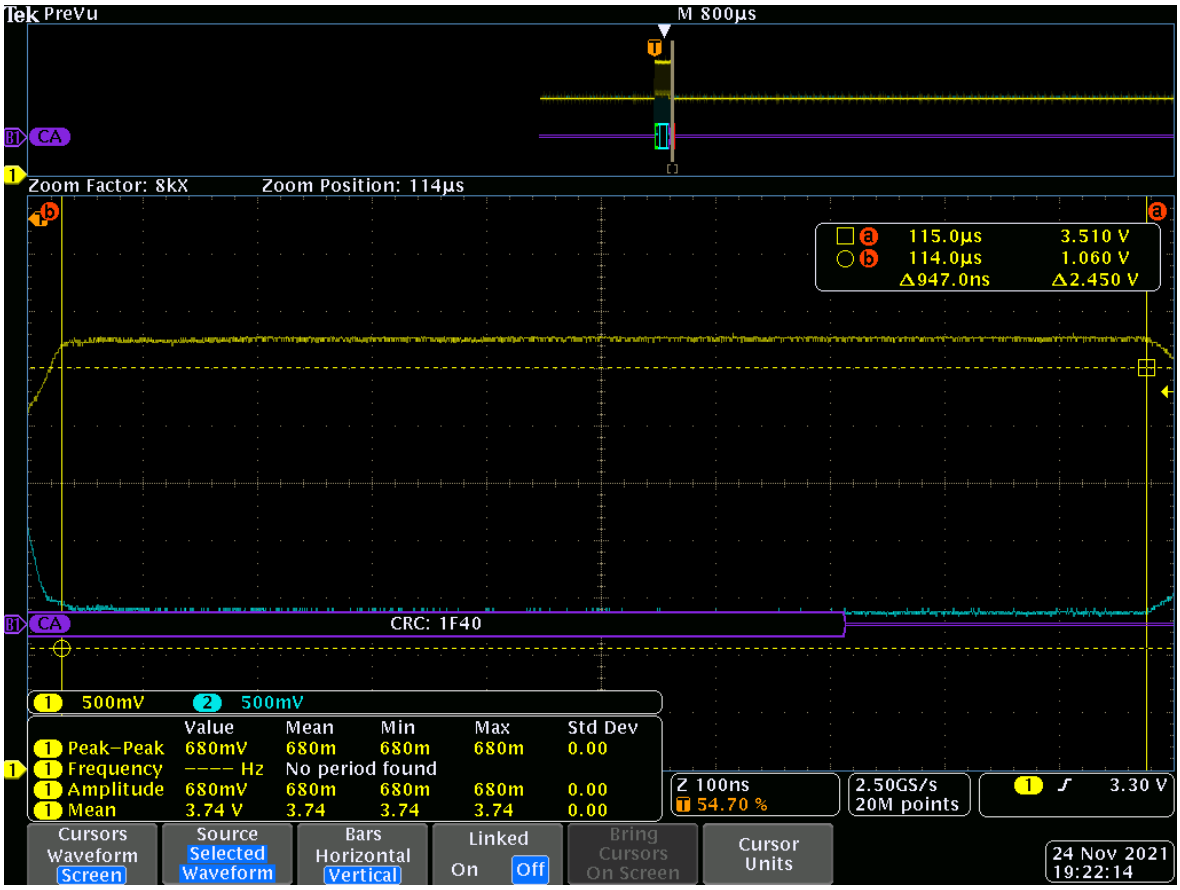
yakalanmasının uygulanabilirliği zor olduğundan bu alanlardan alınan verilerin sinyallerin özellik vektörlerine dahil edilip edilmeyeceği bölüm 5'te ele alınmıştır.



Şekil 4.14 MCP2551 CAN Alıcı-Verici Diferansiyel Sinyal Kalkış İniş Süresi

CAN veri yolunda alıcı-verici birimlerinin dışında CAN kontrolcülerinin de etkisi bulunmaktadır. CAN kontrolcüleri protokolün mantık kısmını gerçekleştiren birimlerdir. Kontrolcüler CAN protokolünün bit zamanlamasını ayarlar. Bit zamanlamaları CAN kontrolcüsünün iç yapısı ve kullanılan saat frekansına göre değişiklik gösterir. Cihazlarda kullanılan saat devrelerinin tolerans ve hata payları bulunmaktadır. Bu frekans hataları kullanılan saat devresinin üretici firmaya, parça numarasına, üretim zamanı gibi parametrelere göre değişkenlik gösterir. Ayrıca kontrolcülerin içerisinde bulunan saat bölücü yapıları da frekansı ayarlamaktadır. CAN kontrolcüsü tümleşik devrelerinde kullanılan saat bölücüler, saat devresinin deterministik olan hatasını arttırmaktadır. Kontrolcü devresindeki bu hatalar CAN protokolünün izin verdiği sınırlar içerisinde olduğu sürece protokolün çalışması etkilenmemektedir. Protokolde bu gibi zamanlama kayması durumuna karşı önlemler de alınmıştır. Protokolde alıcı tarafında donanımsal senkronizasyon ve tekrar senkronizasyon (re-synchronization) gibi prosedürler [1] sistemin zamanlama kaymasından oluşacak hataları gidermektedir. Bu tezde bu zamanlama

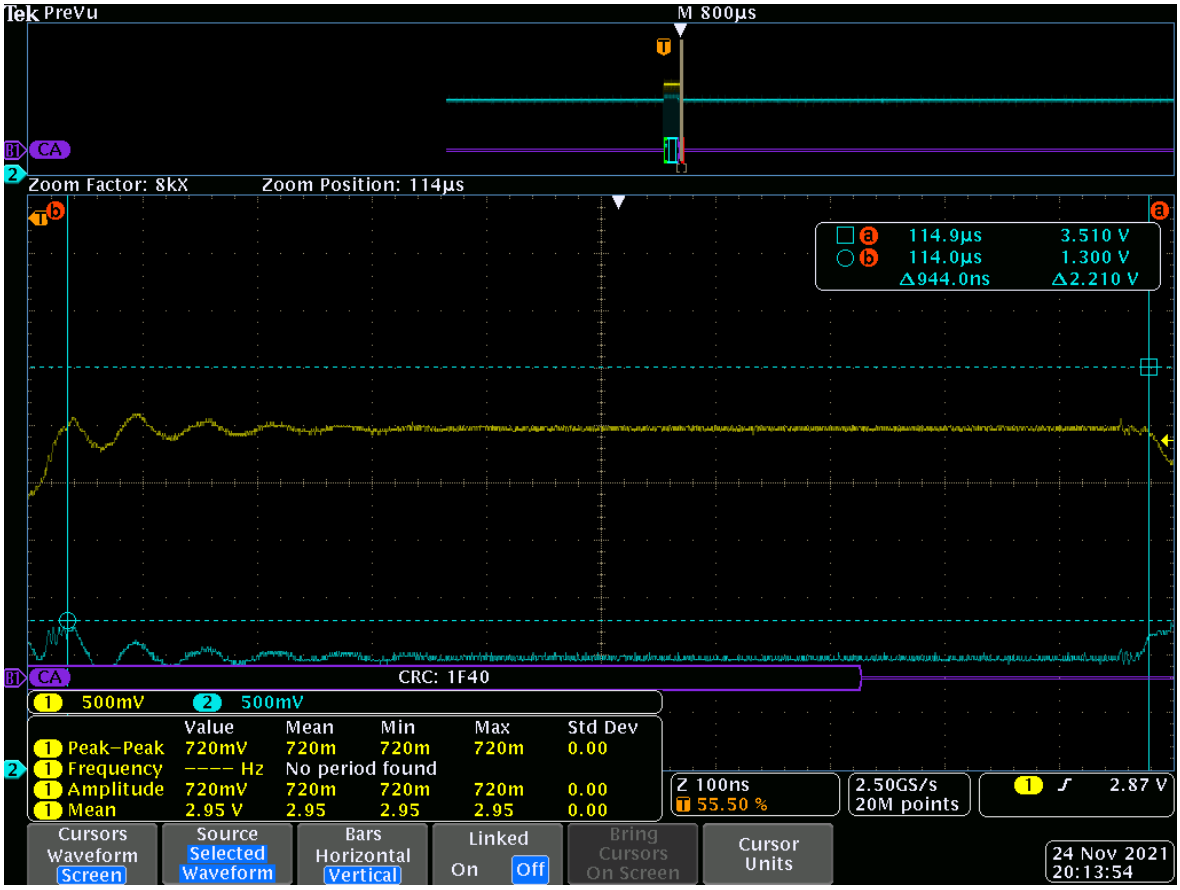
kaymalarının sinyalin ayırt edilmesi açısından önemli olabileceği düşünülmüştür. Sistemin deterministik yaptığı hatalar, sistemin karakteristiği hakkında daha fazla bilgi verecektir. Tez kapsamında geliştirilmiş CAN IP çekirdeğinin CAN veri yolundaki zamanlama karakteristiği Şekil 4.15'te verilmiştir.



Şekil 4.15 Geliştirilen CAN IP Çekirdeğinin Bit Zamanlaması

Geliştirilen CAN IP çekirdeğinin sinyalin sabit olduğu (steady state) kısımlar için zamanlama ölçümü yapılmıştır. Bu ölçüme göre sinyal 947 ns durumunu korumaktadır. Hazır olarak alınmış USB CAN dönüştücü cihazının bit zamanlama ölçüm grafiği Şekil 4.16'da verilmiştir. Bu grafiğe göre USB CAN dönüştürücünün CAN-H ve CAN-L sinyalinin bir bit için sabit durumda olduğu süre 944 ns durumunu korumaktadır. Yapılan ölçümlerde CAN IP çekirdeği ile USB CAN dönüştürücü arasındaki sinyalin sabit kaldığı kısım için süre farkı 3 ns bir değerdir. Bu değerin tez kapsamında kullanılan FPGA'ler ile ölçülebilmesinin zorlayıcı olabileceği ön görüşmüştür. Ayrıca, sistemde ölçüm yapılan noktalar CAN-H ve CAN-L olduğu için alıcı-verici tümleşik devrelerinin kalkış-iniş sürelerinin de bu ölçüme etkisi olduğu aşikardır. Kontrolcünün tekil olarak zamanlamalarının analiz edilebilmesi için kontrolcü ile alıcı-verici arasındaki sinyal hattının

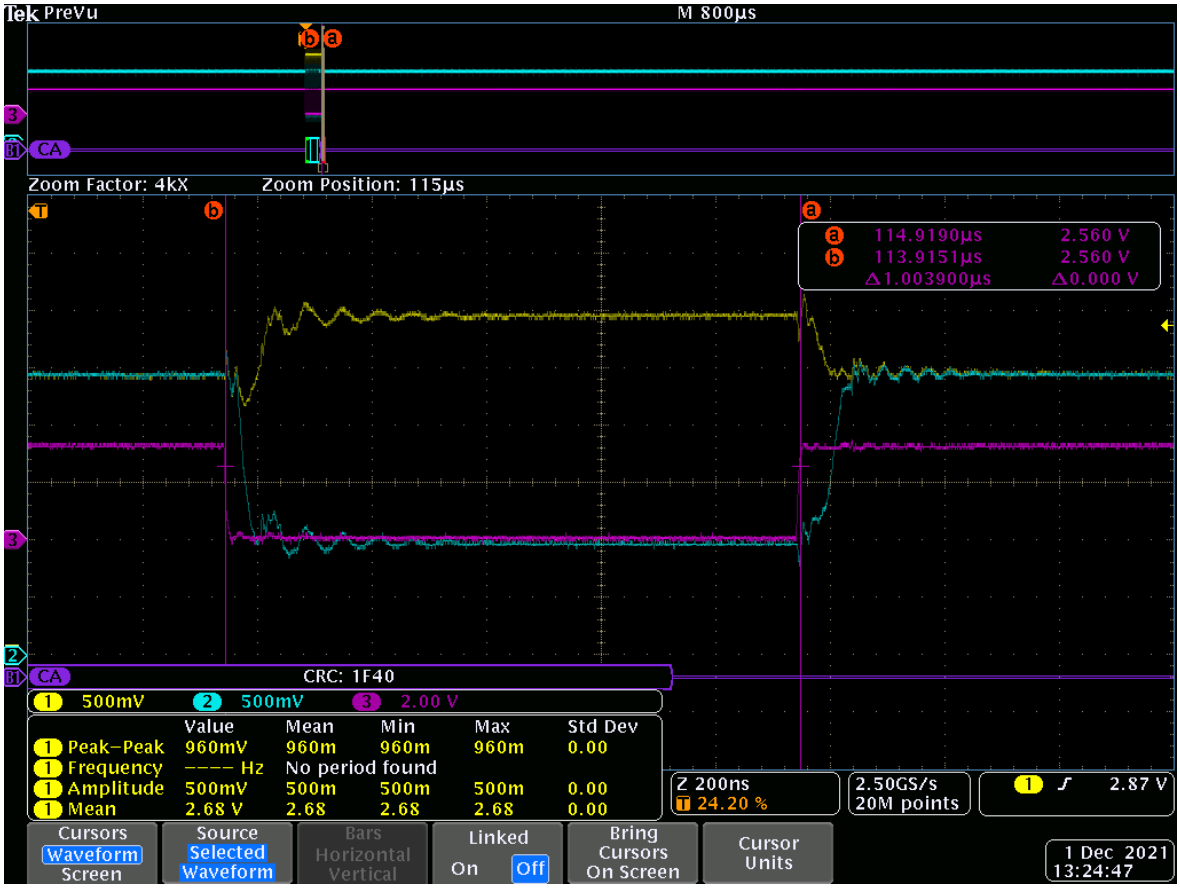
ölçülmesi gerekliliği ortaya çıkmıştır. Bu hatların ölçülmesi kontrolcünün tekil olarak sahip olduğu zamanlama hatalarının gösterecektir.



Şekil 4.16 USB CAN Dönüştürücü Bit Zamanlaması

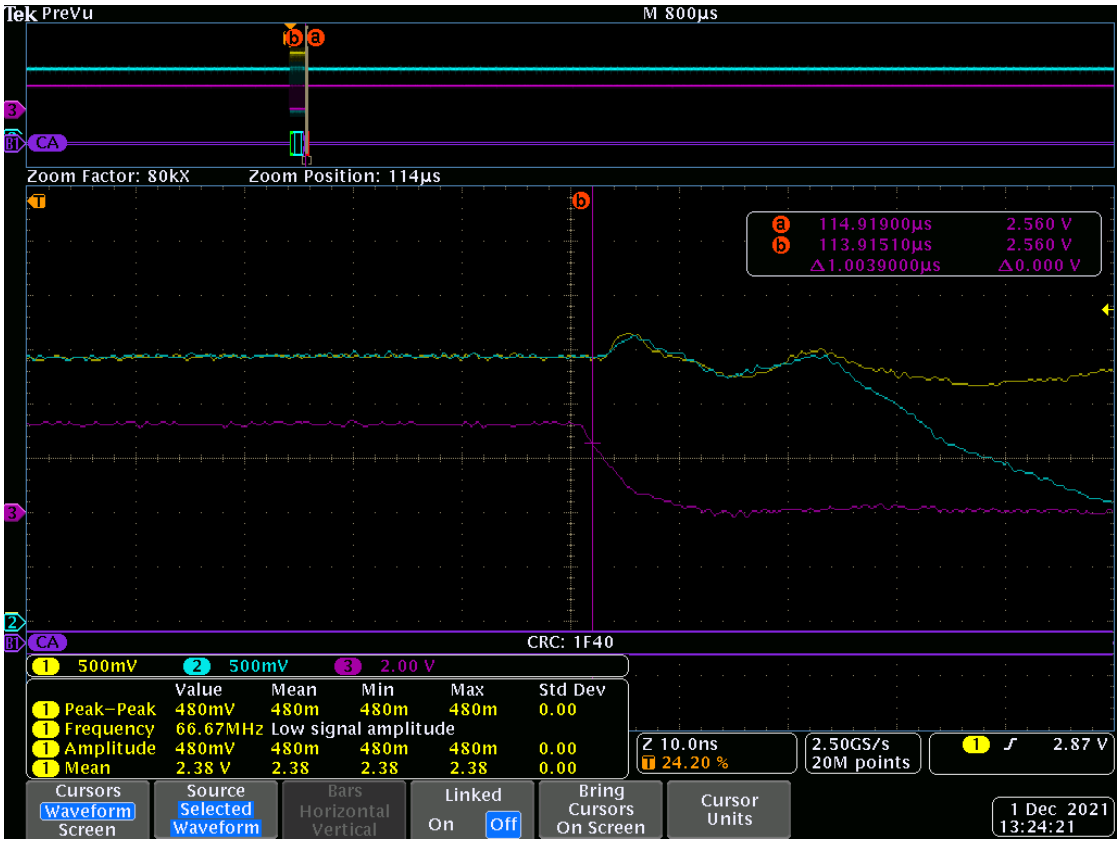
PYNQ FPGA'e bağlı SN65HVD CAN alıcı-verici tümleşik devresine doğrudan bağlantısı olan CAN Tx hattının ölçümü yapılmış olup, ölçüm grafiği Şekil 4.17'de verilmiştir. Bu grafikte mor renkte gösterilen sinyal CAN IP Çekirdeğinden çıkan Tx sinyalidir. Sinyalinin analizinin daha doğru yapılabilmesi için osiloskop örnekleme sayısı 20 milyon olarak ayarlanmıştır. Ayrıca osiloskopun sahip olduğu 2.5 GSPS'lik örnekleme miktarının ölçümün doğru yapılabilmesi açısından çok yeterli olduğu düşünülmüş ve bunun için ek bir çalışma yapılmamıştır. Grafikten de anlaşıldığı üzere kontrolcü sinyali ile alıcı-verici arasında faz farkı bulunmaktadır. Bu faz farkının alıcı-verici tümleşik devresinin iç yapısından kaynaklı ve normal bir bulgu olduğu düşünülmüştür. PYNQ FPGA kartı üzerinde bulunan 125 MHz osilatör devresi bulunmaktadır. Bu 125 MHz'lik saat kullanılarak, FPGA içersinde bulunan MMCM (Mixed-Mode Clock Manager) devresi ile 50 MHz'lik bir sistem saati üretilmiştir. Yapılan ölçümlerde CAN IP Çekirdeğinde 50 MHz'lik saat sinyali kullanılmıştır. CAN IP çekirdeğinin içerisinde CAN protokol veri hızına göre bir saat bölücü

devresi bulunmaktadır. Bu devre 50 MHz'lik sinyalden 1 MHz'lik bir saat üretmekte bu sinyal ise CAN protokolünün zamanlama mantığının gerçeklenmesi için kullanılmıştır.

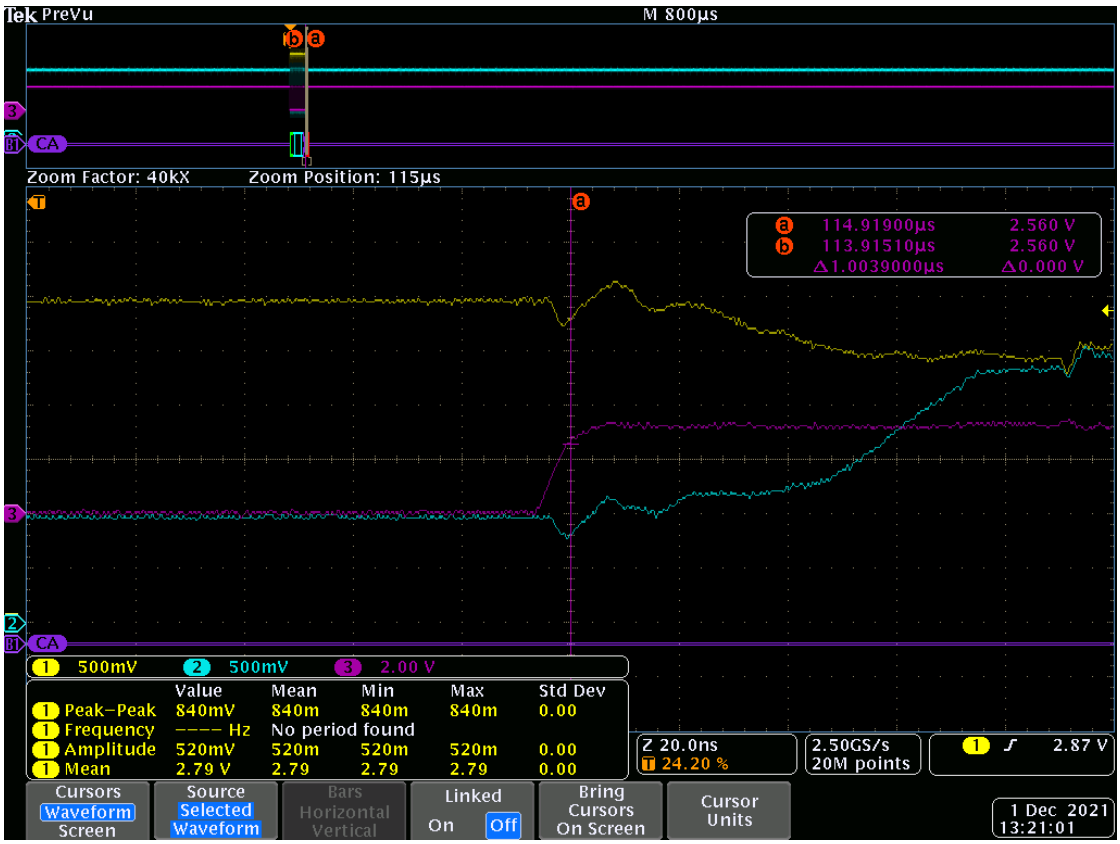


Şekil 4.17 Geliştirilen CAN IP Çekirdeği Kontrolcü Tekil Bit Zamanlaması

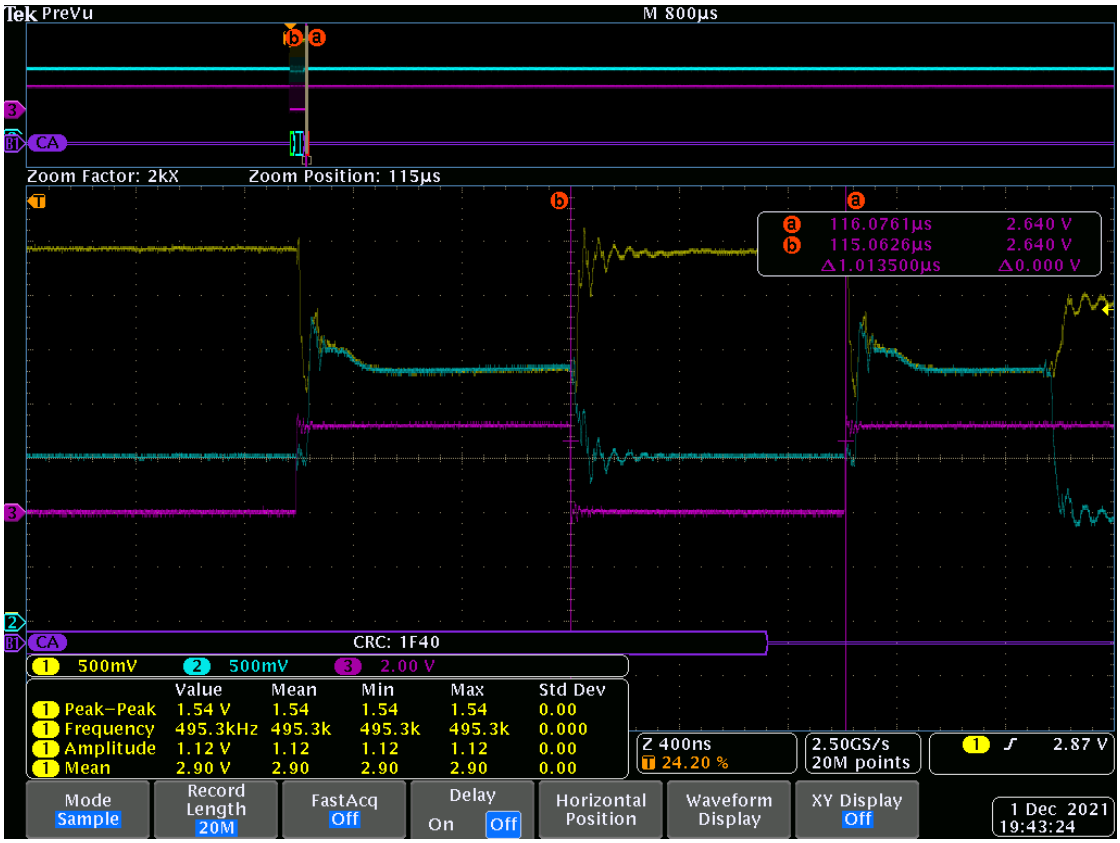
Şekil 4.17'de gösterilmiş olan ölçüm grafiğinde sinyalin uzunluğu 1003.9 ns olarak ölçülmüştür. Ölçüm sinyalin iniş ve kalkış noktalarında sinyalin yaklaşık %75(%±5) voltaj seviyesinde ölçüm yapılmıştır. Her iki noktanın voltajının aynı olması karşılaştırma yapılmasına kolaylık sağlamaktadır. Bu ölçüme göre CAN IP çekirdeği 3.9 ns'lik bir hataya sahiptir. USB CAN dönüştürücünü tekil bit zamanlama grafiği ise Şekil 4.20'de gösterilmiştir. Şekil 4.21 ve Şekil 4.22'de USB CAN dönüştürücüsünün kontrolcüsü için ölçüm noktaları verilmiştir. Bu ölçümlere göre USB CAN dönüştürücü 13.5 ns'lik hataya sahiptir. Arada bulunan yaklaşık 10 ns'lik farkın sinyal tanıma için kullanılabileceği düşünülmüştür. Bu hatalar CAN protololünün çalışmasına engel olmamakla beraber düğümler hakkında önemli bir ayırt edici bilgi sağlamaktadır. Ayrıca, daha düşük veri hızlarında bu farkın daha ayırt edici olabileceği düşünülmüş ve tezin gerçekleme safhasında bu denemenin yapılması uygun bulunmuştur.



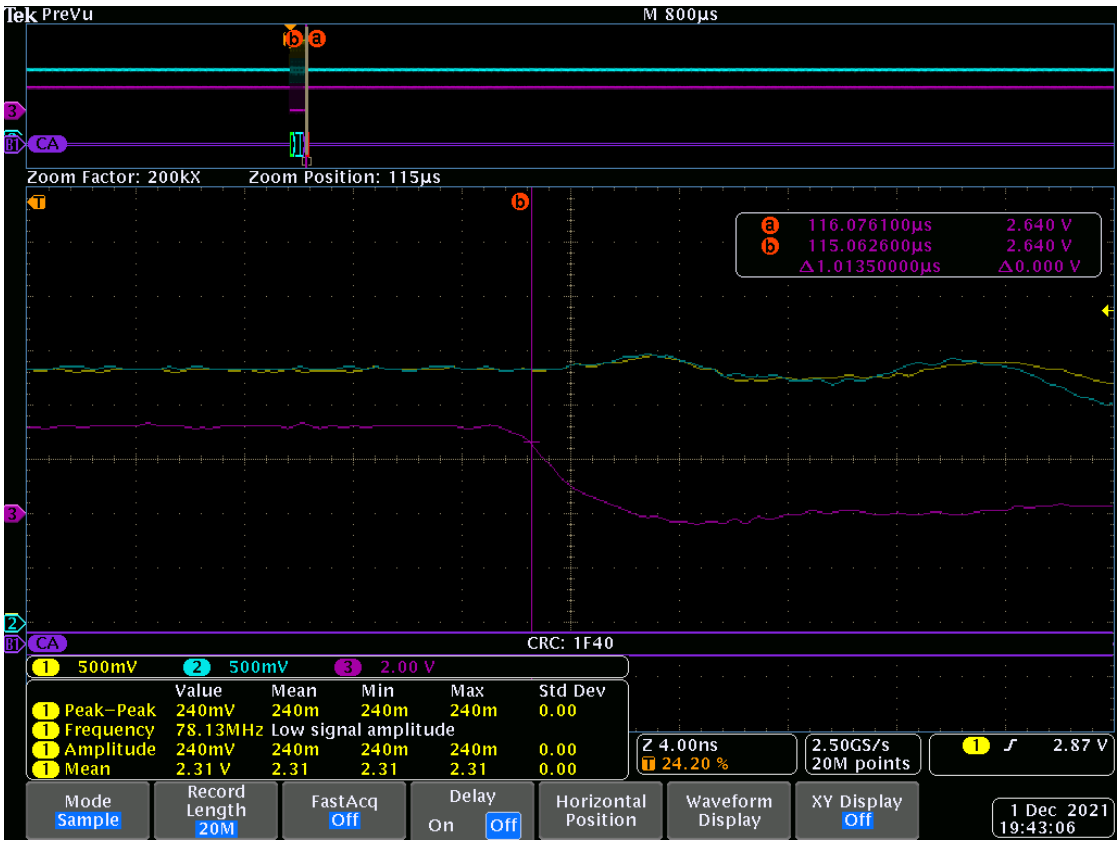
Şekil 4.18 Geliştirilen CAN IP Çekirdeği Düşen Kenar Ölçüm Noktası



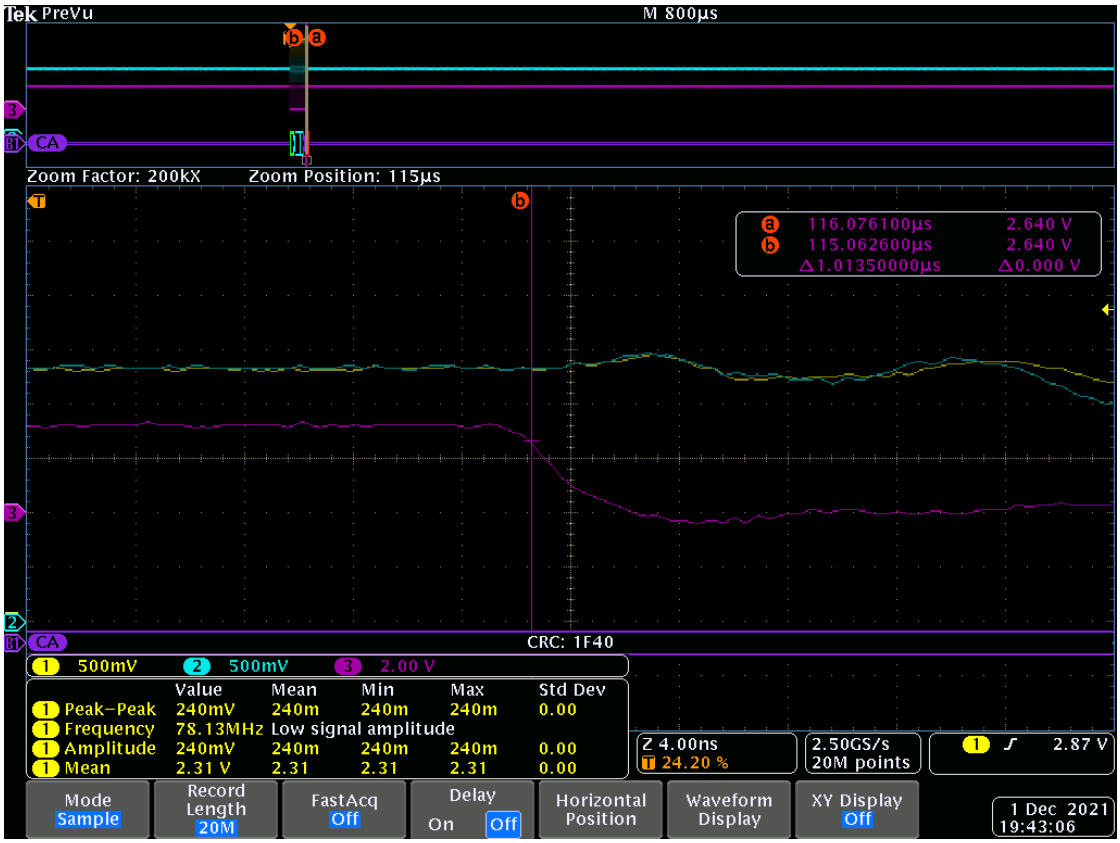
Şekil 4.19 Geliştirilen CAN IP Çekirdeği Yükselen Kenar Ölçüm Noktası



Şekil 4.20 USB CAN Dönüştürücü Kontrolcü Tekil Bit Zamanlaması



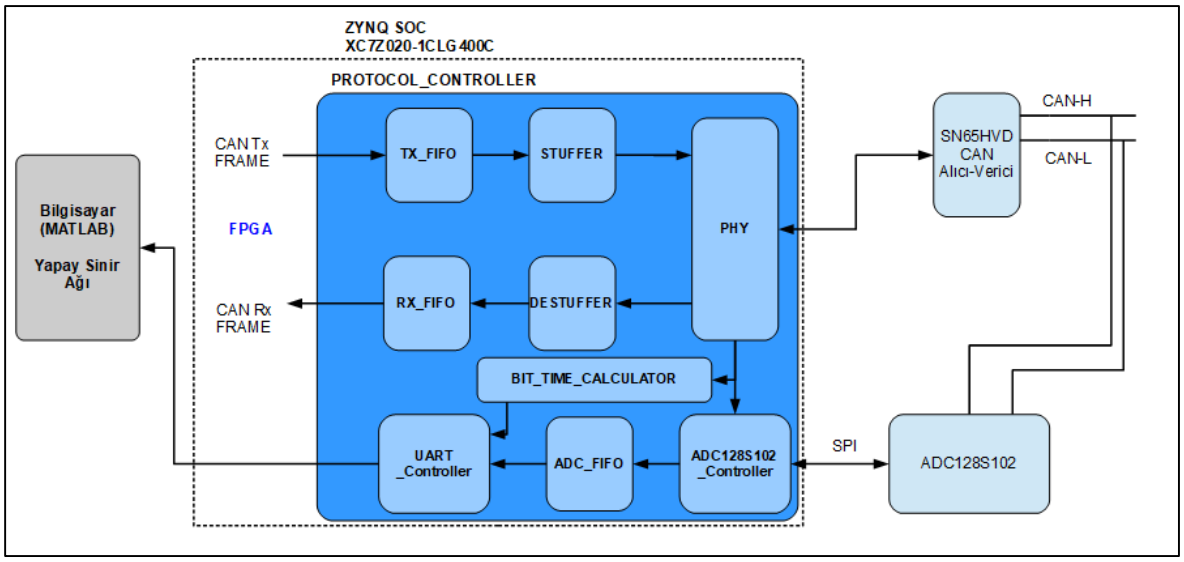
Şekil 4.21 USB CAN Dönüştürücü Kontrolcü Düşen Kenar Ölçüm Noktası



Şekil 4.22 USB CAN Dönüştürücü Kontrolcü Düşen Kenar Ölçüm Noktası

4.3. Geliştirilen CAN IP Çekirdeği

Tez kapsamında CAN sinyallerin örneklenmesi ve bu sinyallerin park izlerinin çıkarılması için CAN 2.0B protokolü Vivado 2019.2 programı ve VHDL donanım betimleme dili kullanılarak FPGA'da gerçeklenmiştir [41]. Bu IP çekirdeği CAN 2.0B standardında herhangi bir değişiklik yapılmadan kodlanmıştır. Bu sayede bir CAN arayüzünde normal bir CAN düğümü olarak çalışabilmektedir. Ayrıca, bu IP çekirdeğin içerisinde ticari bir ADC olan ADC128S102 tümleşik devresini süren bir kod parçası entegre edilmiştir. ADC ve CAN IP çekirdeği koordine olarak çalışmaktadır. Üretilen CAN IP çekirdeği sinyalin parmak izinin alınabileceği uygun olan noktaları işaretlemektedir. İşaretlenen kısımlar ADC ile örneklenip yine FPGA içerisinde bulunan geçici hafıza bellekleri olan BRAM'lerde tutulmaktadır. Bu BRAM'ler FIFO mantığında kullanılarak gerçeklenmiştir. Kayıt edilen veriler FPGA içerinde gerçeklenmiş olan UART alıcı-verici birimi ile bilgisayara aktarılır. Aktarılan veriler daha sonra makine öğrenmesi tabanlı algoritmaların eğitimi için kullanılmaktadır. CAN IP çekirdeğinin mimarisi Şekil 4.23'te verilmiştir.



Şekil 4.23 Geliştirilen CAN IP Mimarisi

4.3.1. PHY Lojik Bloğu

PHY kod bloğu CAN 2.0B protokolüne göre alıcı-verici biriminden gelen verileri protokole uygun olarak örneklemektedir. Örneklenen veriler DESTUFFER lojik kısmına gönderilir. Aynı zamanda STUFFER lojik bloğundan gelen, dolgulanmış (stuffed) veriyi CAN alıcı-vericisine gönderir.

Alınan ve gönderilen veriler Şekil 3.19'da gösterilmiş CAN protokolü bit zamanlamasına göre yapılır. PHY lojik bloğunun içerisinde alıcı ve verici kısımlar için ayrı olarak gerçeklenmiş BUS_TIMER adı verilen lojik bloklar bulunmaktadır. Bu bloklar örneklenen veya gönderilen bir bitin hangi zaman kısımda (segmentte) olduğunu gösterir. Bu zaman kısımları protokole uygun olarak ayarlanabilir olarak tasarlanmıştır.

4.3.2. DESTUFFER Lojik Bloğu

Bu lojik bloğu alıcı durumdaki PHY lojik bloğundan alınan verileri CAN 2.0B protokolüne göre veri ayıklama (destuffing) işlemi gerçekleştirir. Ayıklanana veriler blok içerisinde gerçeklenmiş olan CRC15 lojik bloğuna da gönderilir. Bu blok gelen bitlerin doğru noktalarda protokole uygunluğunu kontrol eder. Bu lojik bloğu ayrıca, PHY lojik bloğuna gelmekte olan CAN sinyalinin hangi noktada olduğunu bildirir (Örn. Uzlaşma kısmı, veri kısmı vb.). Bu bilgi PHY lojik bloğunun protokole uygun çalışması için kullanılmakla beraber çıkarılması istenen sinyal parmak izinin CAN sinyalinin istenilen bölgesinden örnek alınabilmesi için de kullanılmıştır.

4.3.3. STUFFER Lojik Bloğu

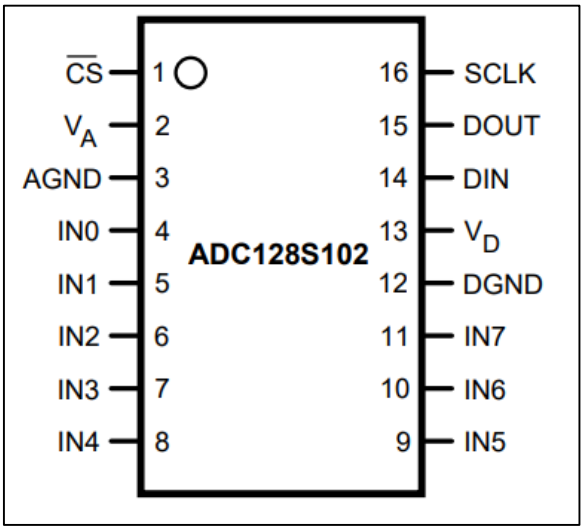
Bu lojik bloğu gönderilmek istenen verileri CAN 2.0B protokolüne göre veri dolgulama işlemini gerçekleştirir. STUFFER lojik bloğu TX_FIFO lojik bloğunda kayıtlı olan gönderilmek istenen CAN paketini bu bloktan alır. Alınan bitler tek tek veri dolgulama ve CRC hesabı yapılarak PHY lojik bloğuna gönderilir. Ayrıca gönderilen veriler ile birlikte bu verilerin CAN paketinin hangi kısmına ait olduğu bilgisi de gönderilir. Bu bilgi PHY bloğu tarafından uzlaşma gibi protokolün işlevlerinde kullanılır.

4.3.4. TX_FIFO ve RX_FIFO

Bu lojik blokları alınan veya gönderilecek olan CAN paketlerini barındırır. FPGA içerisinde BRAM'ler kullanılarak gerçeklenmiştir. 99 bit genişliğe ve 512 derinliğe sahip bu FIFO'lar FPGA içerisinde 1 adet 18 Kb ve 1 adet 36 Kb BRAM kullanmaktadır.

4.3.5. ADC128S102_CONTROLLER Lojik Bloğu

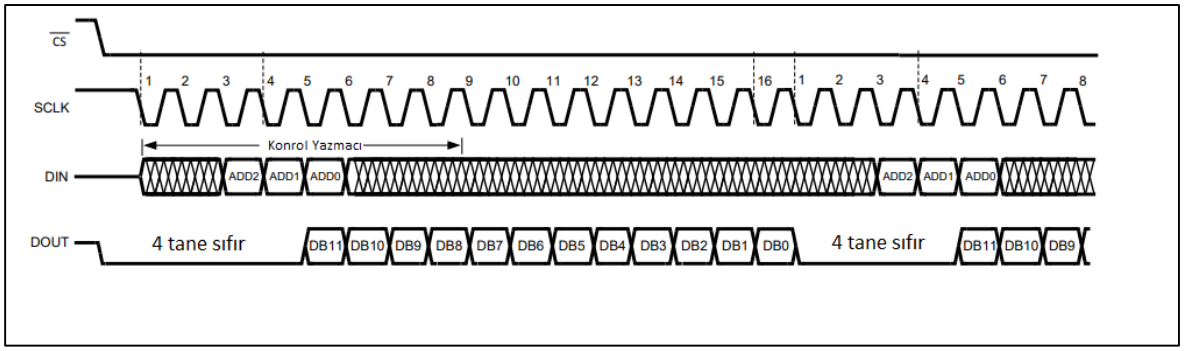
Bu lojik bloğu ADC128S012 tümleşik devresini sürmek için gerçeklenmiştir. ADC128S102 tümleşik devresi en fazla 1 MSPS örnekleme hızına sahip 8 kanallı bir ADC'dir. ADC128S102 tarafından alınan her bir örnek 12 bit genişliğe sahiptir. ADC128S102 pin tanımlamaları Şekil 4.24'te gösterilmiştir.



Şekil 4.24 ADC128S102 Giriş-Çıkışları [42]

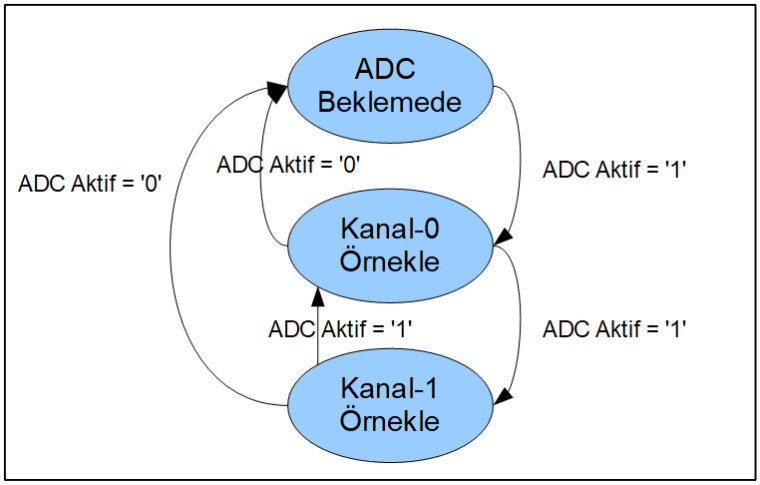
ADC128S102 SPI arayüzü ile sürülmektedir. ADC proje kapsamında maksimum örnekleme hızında çalıştırılmıştır. Bu ADC'de maksimum örnekleme hızının elde edilebilmesi için ADC'nin 16 MHz saat SCLK ile sürülmesi gerekmektedir. Geliştirilen CAN kontrolcüsünde 64 MHz saat frekansı kullanılmıştır. Bunun sebebi ise ADC biriminin

16 MHz SCLK ile sürmek içindir. ADC128S102_CONTROLLER lojik bloğu 64 MHz'lik saat frekansını içerisinde bulunan saat bölücü devre ile 4'e bölerek 16 MHz'lik saat frekansını elde eder.



Şekil 4.25 ADC128S102 SPI Zamanlama Grafiği [42]

Şekil 4.25'te ADC'nin SPI zamanlama grafiği verilmiştir. Buna göre ADC'ye bir sonraki çevirmenin hangi kanal üzerinden yapılacağı verisi DIN üzerinden gönderilir. AD2-0 bitleri kanal seçilimi için kullanılır. Tez kapsamında sadece kanal-0 ve kanal-1 kullanılmıştır. Kanal-0 CAN veri yolunun CAN-H kısmına, kanal-1 ise CAN-L kısmına bağlıdır. ADC bir örnek için başta 4 bit sıfır sonra 12 bit örnekleme değerini MSB'den LSB'ye olacak şekilde toplamda 16 bit gönderir. Bu lojik bloğu aktif olduğu süre zarfında sırasıyla önce kanal-0 ve sonra kanal-1'i örnekler ve blok aktif olduğu sürece sırasıyla bu şekilde devam eder. Bloğun çalışma akış diyagramı Şekil 4.26'da verilmiştir.



Şekil 4.26 ADC Çalışma Akış Diyagramı

Teknik dokümanına göre ADC'nin maksimum sürülebilecek saat frekansı 16 MHz'dir. Buna göre tez kapsamında geliştirilen kod elde edilebilecek maksimum örnekleme

frekansı aşağıdaki formüle göre hesaplanır. Bu hesaplamaya göre her bir kanal başına düşen örnekleme sayısı 500 KSPS'dir.

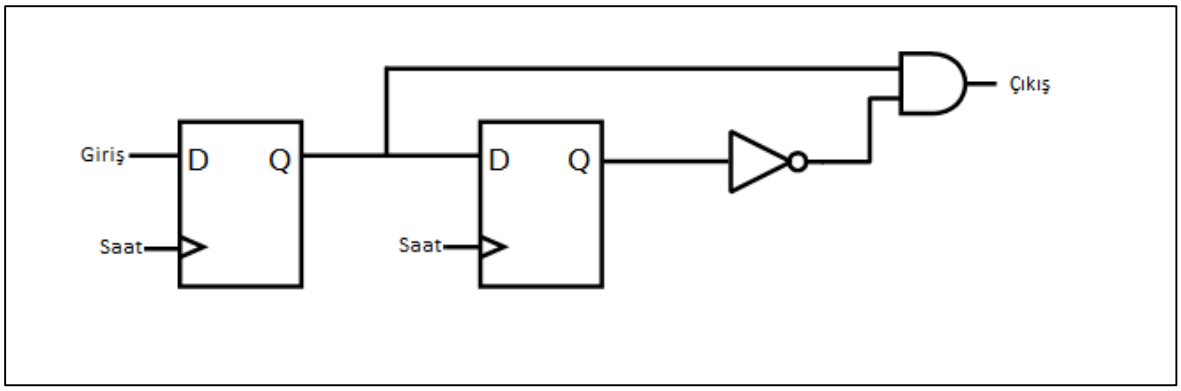
$$Örnekleme\ Frekansı = \frac{Maksimum\ SCLK\ saat\ frekansı}{16\ x\ Kanal\ sayısı}$$

4.3.6. ADC_FIFO Lojik Bloğu

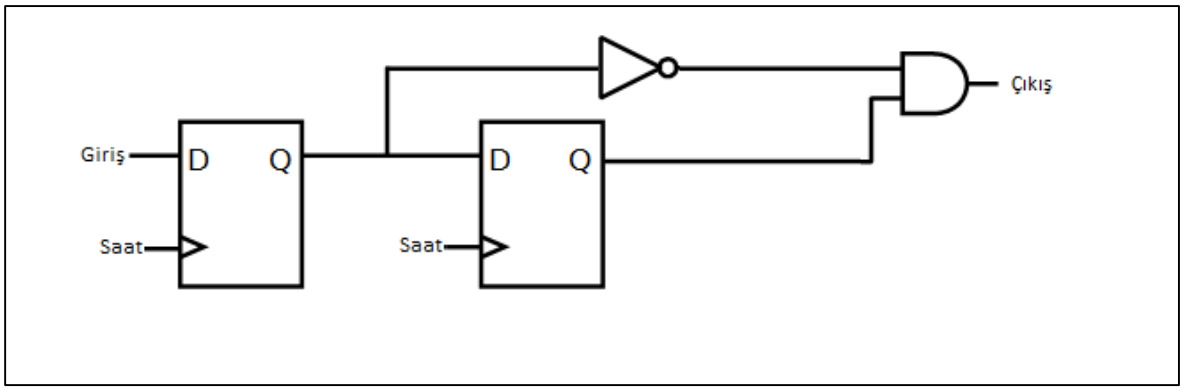
Örneklenen veriler FPGA içerisinde BRAM'ler ile gerçeklenmiş ADC_FIFO'ya yazılmaktadır. Bu FIFO 16 bit genişlik 2048 derinliğe sahiptir. Her iki kanal örnekleri bu geçici hafızaya yazılır. 16 bitlik veri kısmının LSB 12 biti ADC'den alınan örnekleme değeridir. MSB 2 biti ise verinin hangi kanaldan alındığını belirten işaret bulunur. Verinin 13. ve 12. Biti ise rezerve bitlerdir. ADC bloğu kanal-0 ve kanal-1'i sırasıyla okuduğu için bir kanal için en fazla örnek tutma sayısı 1024'tür. Tek bir kanal örnekleme periyodu ise 2 mikrosaniyedir. Bu durumda ADC_FIFO geçici hafızası tek bir kanal için sinyalin en fazla 2,048 milisaniyelik kısmını içerisinde saklayabilmektedir. ADC_FIFO'nun saklama derinliği arttırılabilmektedir ancak tez kapsamında CAN sinyalinin parmak izinin çıkarılması için kullanılan kısım için bu boyut hesaplanarak seçilmiştir.

4.3.7. BIT_TIME_CALCULATOR Lojik Bloğu

Bu blok Bölüm 4.2'de anlatılmış olan CAN sinyalinin incelenmesinden sonra gerçeklenmesi düşünülmüştür. Bu lojik bloğu gönderilen CAN sinyalinin bit uzunluğunu ölçen bir bloktur. Bu blok 512 MHz saat frekansında CAN alıcı-vericinin kontrolcüye ilettiği Rx sinyalini örnekler. Örneklenen sinyaller D flip-flop ve AND kapısı kullanılarak gerçeklenmiş olan yükselen ve düşen kenar sinyal algılayıcı (edge detector) devresi ile sinyalin yükselen ve düşen kenarları çıkarılır. FPGA içerisinde gerçeklenmiş yükselen ve düşen kenar algılayıcı devreler sırasıyla Şekil 4.27 ve Şekil 4.28'de gösterilmiştir.



Şekil 4.27 Yükselen Kenar Algılayıcı Lojik Devresi



Şekil 4.28 Düşen Kenar Algılayıcı Lojik Devresi

512 MHz saat frekansı ile çalışmasından kaynaklı bu devrenin 2 nanosaniyenin altında bir çözünürlükte sinyalin bit uzunluğunun ölçülebilmesine olanak sağlamaktadır. Genel olarak bu blok ardışık olan yükselen ve düşen kenar arasındaki (veya tam tersi) sürede her bir saat çeviriminde bir artacak şekilde bir sayaç bulunur. İki farklı sinyal kenarı arasındaki süreyi sayan bu sayacın değerinin 512 MHz'lik saat periyodu ile çarpılması ile CAN sinyalinin alıcı-verici çıkışındaki bit uzunluğunun değeri bulunur. Her bir alınan CAN paketi için ölçülen bit değeri örneklenen ADC verileri ile beraber UART arayüzü ile birlikte yazılım ortamına aktarılır. Ölçülen bu bit uzunluğu makine öğrenmesi algoritmalarının eğitilmesinde kullanılması planlanmıştır.

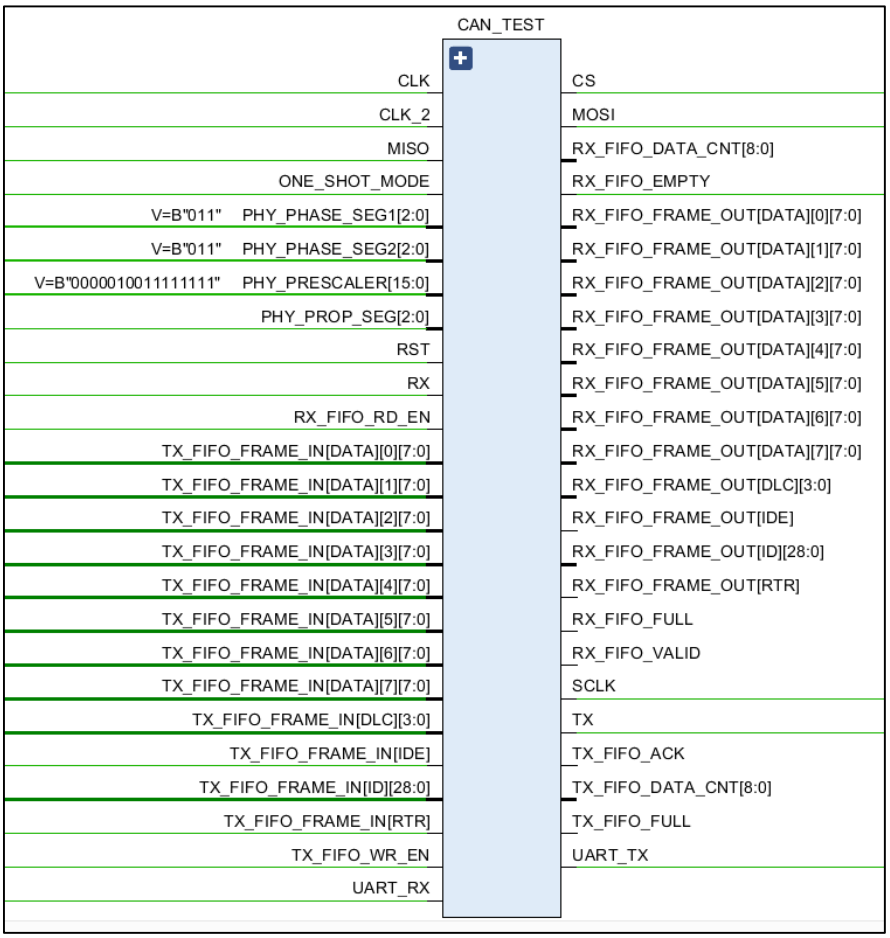
4.3.8. UART Lojik Bloğu

Bu lojik bloğu tez kapsamında gerçekleştirilen CAN IP çekirdeğinin ve ona bağlı ADC tümleşik devresinin örneklediği verileri bilgisayar yani yazılım ortamına aktarılması için kullanılmıştır. UART lojik bloğu örneklenen CAN-H ve CAN-L sinyallerini ve bunun yanında hesaplanan göndericinin bit uzunluğunu hesaplayan birimden alınan ham verilerin bilgisayar aktarılması için kullanılır. Bu blok 921600 bps veri hızına sahiptir. Veri 8 bit protokole uygun olarak LSB'den MSB'ye olacak şekilde sırasıyla gönderilir. UART protokolünün 1 bitlik parite hesabı desteği gerek duyulmadığı için tez kapsamında kullanılmamıştır.

4.3.9. PROTOCOL_CONTROLLER Lojik Bloğu

Tez kapsamında gerçeklenmiş olan CAN IP çekirdeğine ait diğer lojik blokları barındıran ana lojik bloğudur. Bu blok birimlerin birbirleri ile olan bağlantılarını gerçekleştirir. Buna ek olarak, CAN 2.0B protokolünün uygulama katmanında yapılması gerek işlevleri gerçekleştir. Protokol işlevleri dışında diğeri ECU'ların gönderdiği sinyal

parmak izlerini çıkaran ADC bloğu, bit uzunluğunu hesaplayan blok ve verileri yazılım ortamına aktaran UART arayüz bloğu da bu lojik bloğunun altında bulunmaktadır. Sistemin uygulanabilirliğini sağlayan bu kod bloğunun Vivado geliştirme ortamından üretilmiş olan giriş-çıkışlarını gösteren blok şeması Şekil 4.29'da verilmiştir.

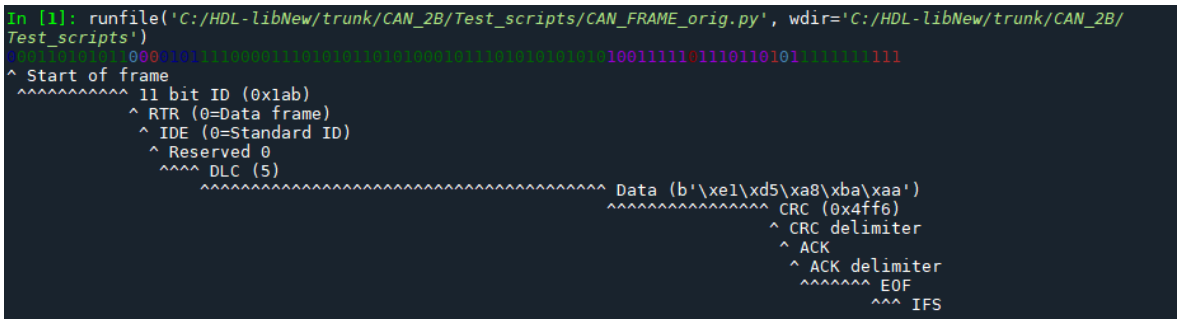


Şekil 4.29 Geliştirilmiş CAN IP Çekirdeği Blok Şeması

4.4. Benzetim Çalışmaları

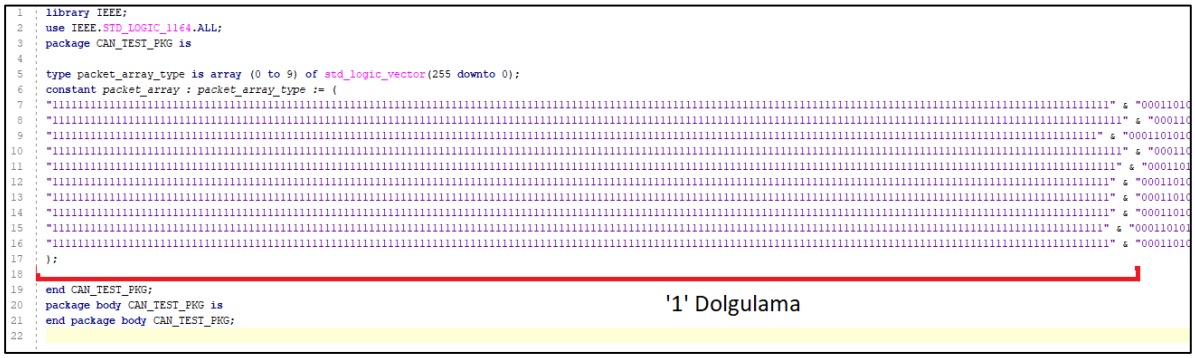
Tez kapsamında geliştirilen CAN IP çekirdeğinin makine öğrenmesi tabanlı bir IDS geliştirilmesinde kullanılması için sistemin çalışma mantığının doğrulanması gerektiği düşünülmüştür. Ayrıca, CAN sinyallerinin parmak izlerinin doğru noktalardan alındığı ve alınan verilerin mantıksal olarak doğru olduğunun testleri hem yazılım hem de donanım ortamında yapılan simülasyonlar ile test edilmiştir. Öncelikle CAN IP çekirdeğinin sinyalin belirlenen noktalarını doğru bir şekilde yakalayabilmesinin doğrulanması gerektiği düşünülmüş ve bu kapsamda doğrulama simülasyonları yapılmıştır.

CAN IP çekirdeğinin simülasyon çalışmaları kapsamında Ken Tindell tarafından Pyhon ortamında [43]geliştirilmiş açık kaynak kodlu CAN paket üreteci kullanılmıştır [44]. Bu paket üreteci verilen ID ve paket verilerine göre veri dolgulaması yapılmış CAN paketini üretir. CAN paket üretecinin çıktısı Şekil 4.30'da verilmiştir.

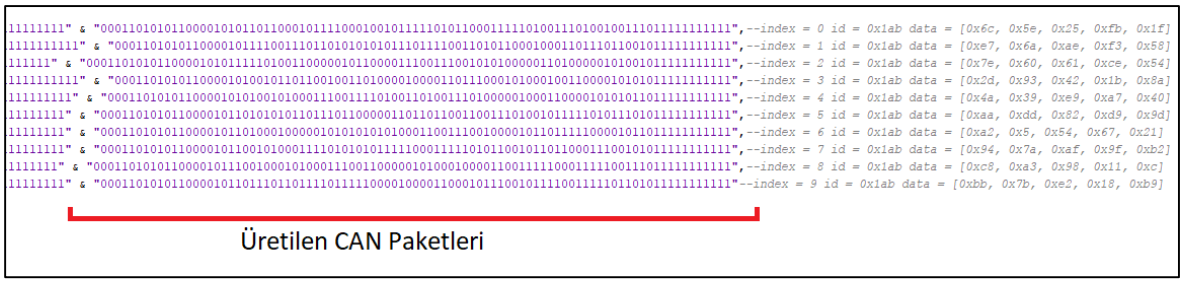


Şekil 4.30 CAN Paket Üreteci Çıktısı

Yüksek sayıda test yapılabilmesi için açık kaynak kodlu CAN üreteci tez kapsamında değiştirilmiştir. Paket üreteci istenilen sayıda veri paketinin bir .vhd dosyası yaratarak her bir paketin boyutu ayarlanan bir dizi (array) içerisine yazılması sağlanmıştır. Üretilen paketin bit uzunlukları veri dolgulaması nedeniyle değişken boyuttadır; fakat bir CAN paketinde tüm bitler aynı bile olsa CAN paketinin bit uzunluğunun 256 bit uzunluğuna erişmesi mümkün değildir. Bu nedenle üretilen dizinin bit uzunluğu 256 olarak seçilmiştir. Bir CAN paketi 256 bitlik kısmı tamamen dolduramamaktadır. 256 bitlik kısmın CAN paketini içermeyen kısmına ise '1' doldurulmuştur (padding). Bu bitler her iki paket arasında olmasından kaynaklı ve doldurulan paketlerin boyutlarının da farklı olması nedeniyle de, paketler arası değişken bir bekleme oluşturulmasını da sağlamıştır. Bu sayede sistemin daha gerçekçi bir benzetimi yapılması amaçlanmıştır. Üretilen CAN paket dizisinin bir örneği parçalar halinde Şekil 4.31 ve Şekil 4.32'de gösterilmiştir.



Şekil 4.31 Üretilen Örnek CAN Paket Dizisi '1' Doldurulmuş Kısım



Şekil 4.32 Üretilen Örnek CAN Paket Dizisi Paket Kısımı

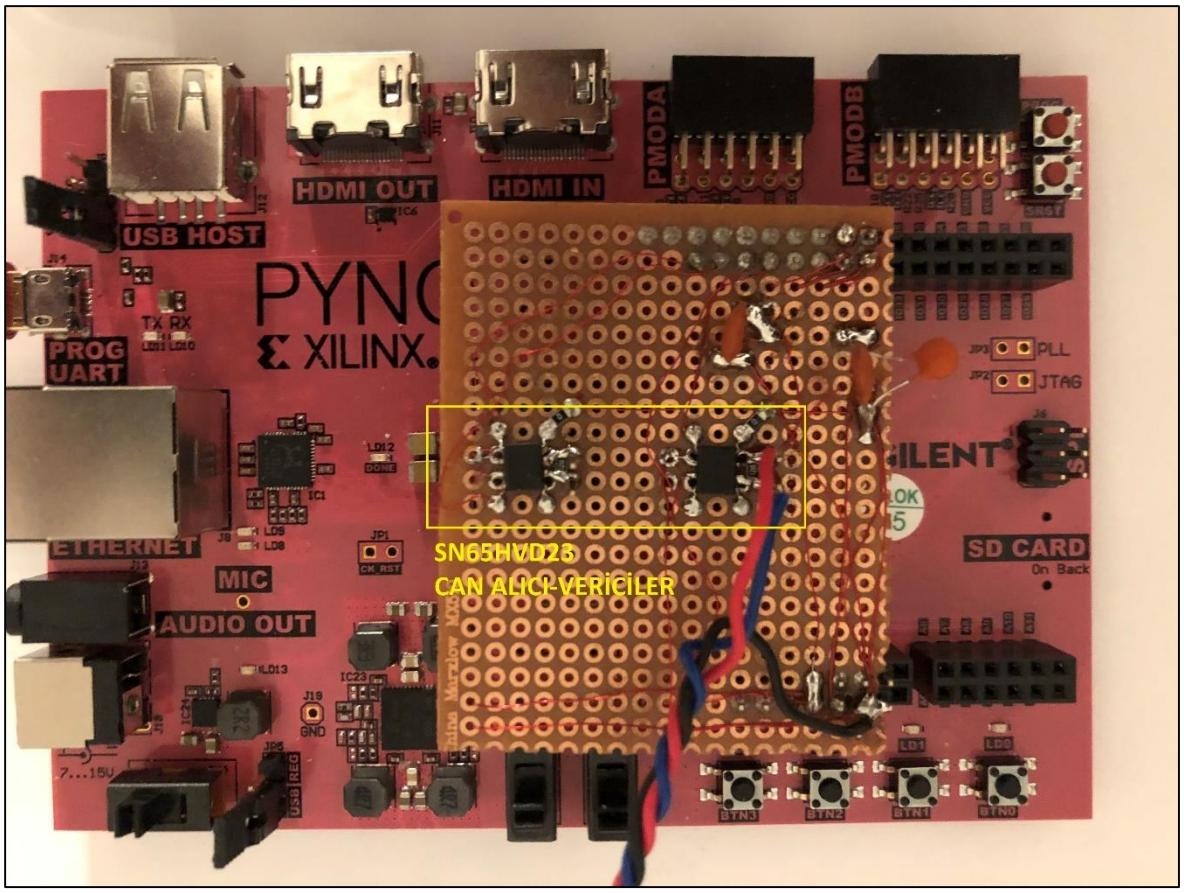
Örnek CAN paketleri üretildikten CAN IP çekirdeği yazılım ortamında test edilmiştir. Sistemin doğru çalışması için CAN IP çekirdeğinin verileri doğru zamanlarda yakalaması önemlidir. Bu nedeni ise sistemin istenilen özel bir lokasyonda sinyali örneklenmesinin istenmesidir. ADC lojik bloğu örneklenecek sinyalin lokasyonunu CAN IP çekirdeğinden almaktadır. Bu nedenle CAN IP çekirdeğinin yapacağı zamanlama ve senkronizasyon hatası ADC lojik bloğunun da CAN-H ve CAN-L sinyalinin yanlış noktalarını örneklemesine neden olacaktır. Bu da sistemin IDS sistemi üretilmesi için kullanılması planlanan makine öğrenmesi algoritmalarının doğru şekilde eğitilmesine engel olacaktır. Bu nedenle CAN IP çekirdeğinin doğru çalıştığının garantilenmesi, tez kapsamında geliştirilecek olan kalan kısımlar için bir ön şarttır.

CAN IP çekirdeğinin benzetimi üretilen örnek CAN paketleriyle Vivado Xsim simülatöründe yapılmıştır. Birkaç farklı paketle yapılan testlerde kodlarda bulunan hatalar düzeltilmiştir ve çalışır hale getirilmiştir. Şekil 4.32'de gösterilmiş olan üretilmiş CAN paketleri ile yapılan simülasyon çalışması Şekil 4.33'te gösterilmiştir. Şekilden de anlaşılacağı gibi üretilen CAN paket dizisinin ilki, RX hattı üzerinden gönderilmiştir. Gönderilen paket, alıcı tarafından doğru olarak alınmış ve RX_FIFO'ya yazılmıştır. Paketin RX_FIFO'dan çıkarılmış hali benzetimde gösterilmiştir.

Şekil 4.33 CAN IP Benzetim Çalışması

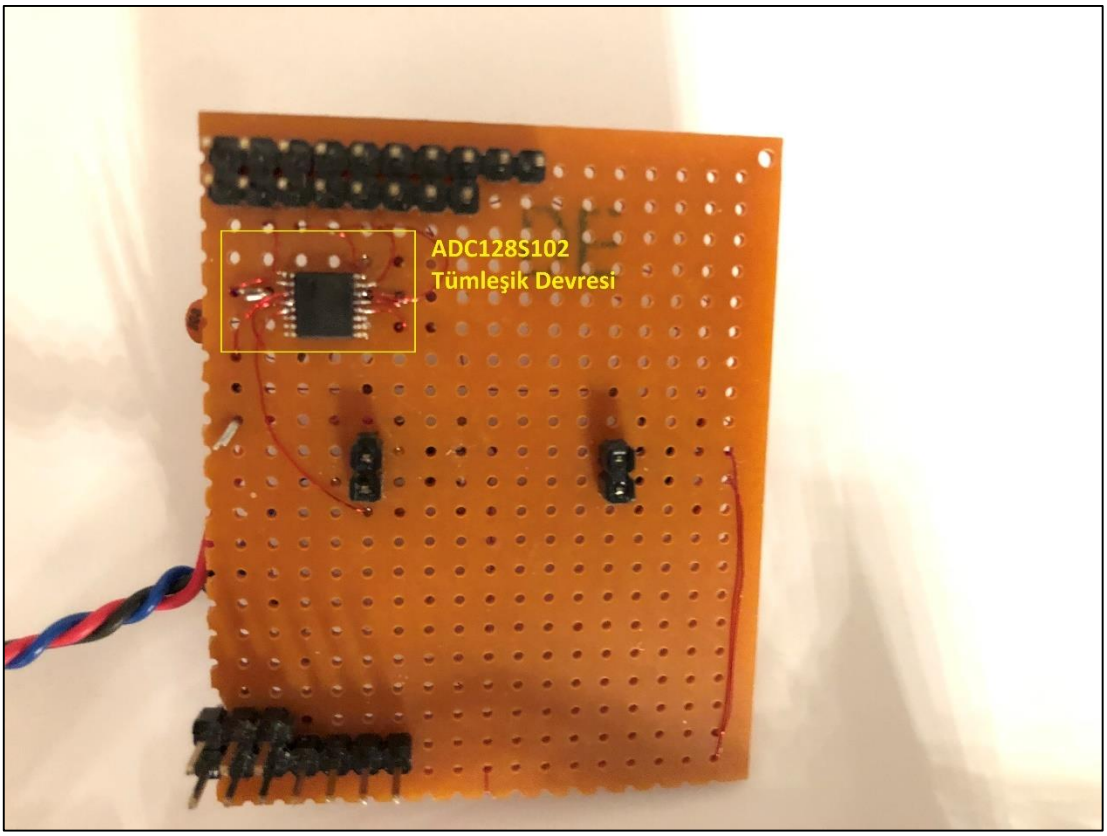
CAN IP çekirdeğinin birkaç farklı paket ile test edilmesinden sonra sistemin doğru çalıştığının garantilenmesi adına çok daha fazla paket ile simülasyonunun yapılması planlanmıştır. Bu kapsamda 100 bin paket içeren test verisi üretilmiştir. Bu paketlerin ID, DLC ve veri kısmı rastgele (random) üretilmiştir. Simülasyon süresinin kısaltılması için CAN veri yolunun benzetimdeki hızı 1 Mbps olarak ayarlanmıştır. Üretilen bu paketler ile test yapılmış ve tüm testlerde CAN IP çekirdeği verileri doğru olarak yakalamıştır. Simülasyon süresinin çok uzun olmasından dolayı, bu aşamadan sonra CAN IP çekirdeğinin donanım üzerinde test edilmesine karar verilmiştir.

CAN IP çekirdeğinin donanım üzerinde test edilmesi için Şekil 4.23'te gösterilen mimari, PYNQ geliştirme kartına uygun olarak pertinaks üzerine el ile lehimlenerek üretilmiştir. Bu başlık üzerine 2 adet SN65HVD23 alıcı-verici entegresi, ADC128S102 entegresi ve gürültü azaltma amacıyla voltaj çıkışlarına 3 adet kapasitör lehimlenmiştir. Şekil 4.34'te gösterilmiş pertinaksın ön yüzünde SN65HVD23 CAN alıcı-verici entegreleri bulunmaktadır. Ayrıca bu yüzde 120 ohm sonlandırma dirençleri, ve baypas kapasitörleri bulunmaktadır.



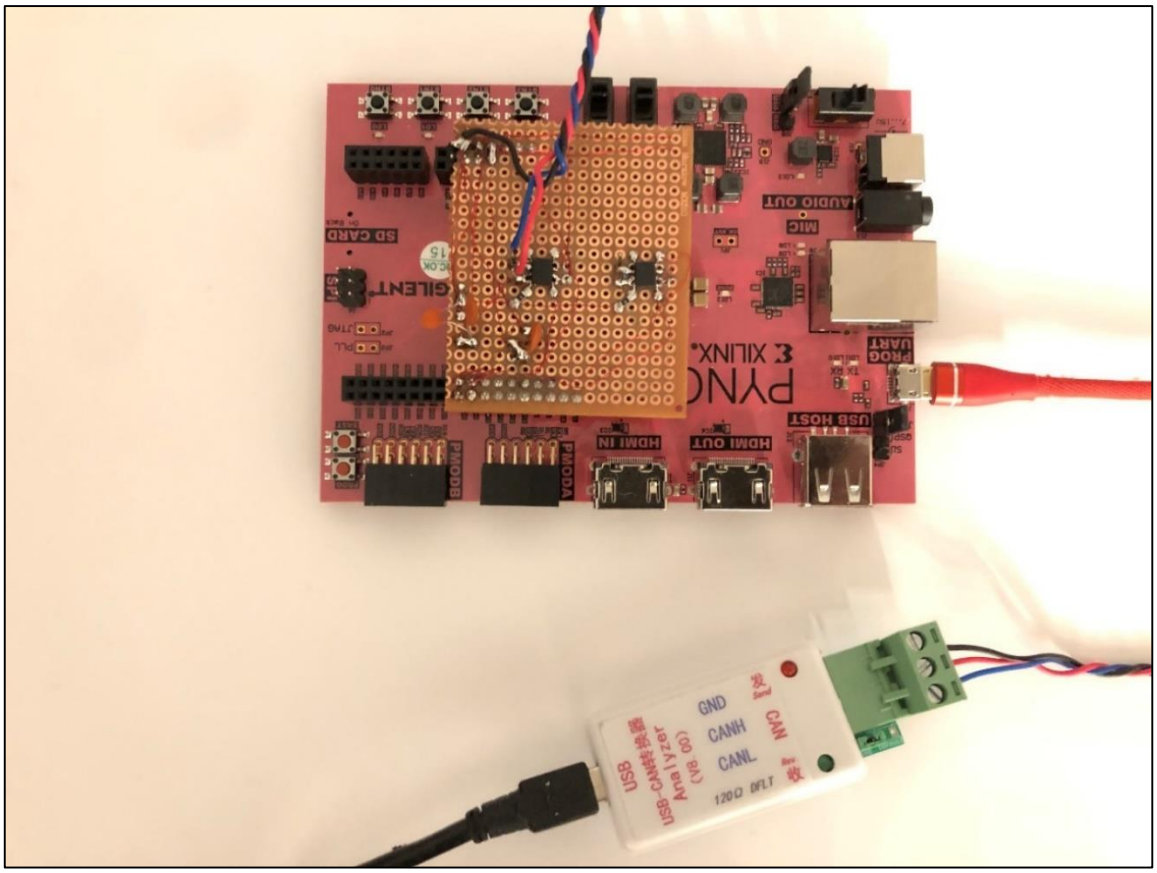
Şekil 4.34 Pertinaks Ön Yüz

Şekil 4.35'te gösterilmiş olan pertinaksın arka yüzünde ise ADC128S102 tümleşik devresi bulunmaktadır.



Şekil 4.35 Pertinaks Arka Yüz

CAN IP çekirdeğinin test edilmesi için gerekli devre oluşturulduktan sonra geliştirilen bu CAN düğümü Bölüm 4.1.4'te özellikleri verilmiş olan USB CAN çevirici modüle noktadan noktaya (point-to-point) olacak şekilde her bir düğüm için 1 metrelik kablo toplamda 2 metrelik kablo ile birbirine bağlanmıştır. Donanımlar arası bağlantı Şekil 4.36'da gösterilmiştir. Donanımsal bağlantı gerçekleştirildikten sonra Şekil 4.6'da gösterilmiş olan USB CAN dönüştürücü programı ile paketler gönderilerek test edilmiştir, fakat bu programın paketlerin gönderimi için çok yavaş olduğu ve yüksek sayıda test paketinin gönderilmesinin çok uzun süreceği gözlemlenmiştir. Buna ek olarak program üzerinden rastgele paket üretilemediği için bu program üzerinden yapılacak testlerin CAN IP çekirdeğinin doğrulanması için kullanılamayacağı anlaşılmıştır. Bu sebeple Python ortamında programın yaptığı işlevi daha hızlı ve rastgele gerçekleştirecek bir kod parçası yazılmıştır. Bu kod parçası USB CAN dönüştürücüye seri arayüz üzerinden bağlanarak hızlı bir şekilde veri gönderilmesini sağlamıştır. Seri haberleşme için Python için geliştirilmiş olan pyserial kütüphanesi kullanılmıştır [45].

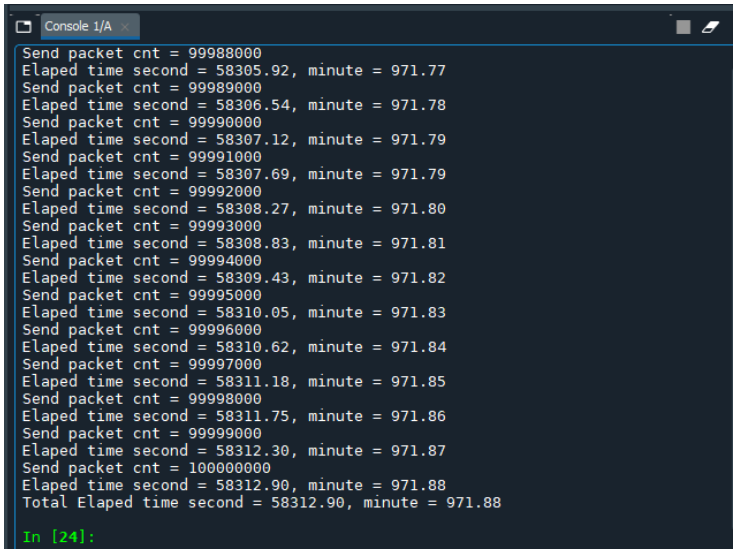


Şekil 4.36 USB CAN ve PYNQ Geliştirme Kart(CAN IP) Bağlantısı

```
testCntTh = 100000000


startTime = time.time()
while(True):
    try:
        testID = random.randint(0,2031)
        testDlc = random.randint(0, 8)
        testFrameType = stdData
        testFtypeDlc = (testFrameType << 4) | testDlc
        testData = random.sample(range(255), testDlc)
        testPacket = [dataHeader] + [testFtypeDlc] + hexToList(testID, 2, 'little') + testData + [dataFo
        sendData(ser, testPacket)
        packetCnt += 1


        if (packetCnt % 1000) == 0:
            print("Send packet cnt = %d"%(packetCnt))
            stopTime = time.time()
            elapsed = stopTime - startTime
            print("Elaped time second = %.2f, minute = %.2f"%(elapsed,(elapsed/60)))

        if packetCnt == testCntTh:
            stopTime = time.time()
            elapsed = stopTime - startTime
            print("Total Elaped time second = %.2f, minute = %.2f"%(elapsed,(elapsed/60)))
            break

    except KeyboardInterrupt:
        stopTime = time.time()
        elapsed = stopTime - startTime
        print("Total Elaped time second = %.2f, minute = %.2f"%(elapsed,(elapsed/60)))
        print("Send packet cnt = %d"%(packetCnt))
        ser.close()

ser.close()
```

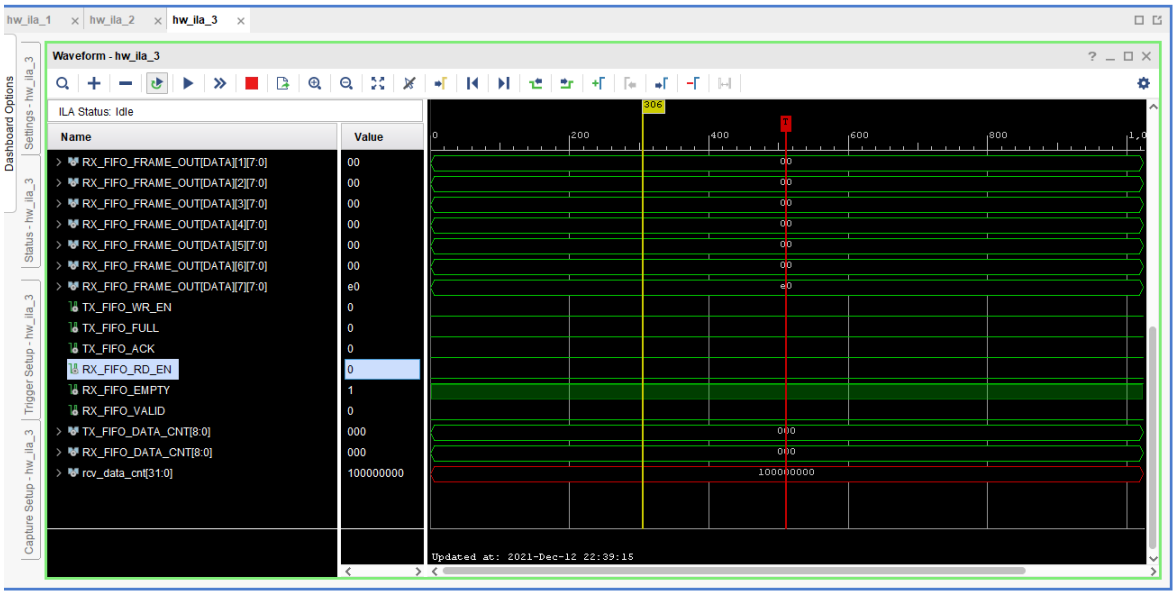
Şekil 4.37 USB CAN Paket Üreteci Python Kod Parçası

Rastgele üretilen CAN paketleri ile CAN IP test edilmiştir. Test için toplam 100 milyon paket gönderilmiş ve FPGA tarafından toplamda 100 milyon başarılı paket alımı gerçekleştirilmiştir. Gönderilen paket sayısı Python üzerinden ayarlanmıştır. Alınan doğru paket sayısı ise FPGA içerisine yerleştirilebilen tümleşik lojik analizör ile incelenmiştir. Gönderilen ve alınan paket içeriklerinin kıyaslaması yapılmamıştır. Bunun nedeni ise protokolde hali hazırda CRC algoritmasının bulunmasıdır. Gönderilen veride bir hata olması durumda paket CRC hesaplamasından geçmeyecek ve paket uygun olarak işaretlenmeyecektir. Ayrıca lojik analizör vasıtası ile oluşacak hata bayrakları da incelenmiş ve herhangi bir hata ile karşılaşılmamıştır. Tüm testin süresi 971 dakika (16,18 saat) sürmüştür. Test sonucunda gönderilen tüm paketler başarılı bir şekilde alınmıştır. Testin Python programındaki çıktısı Şekil 4.38'de verilmiştir.

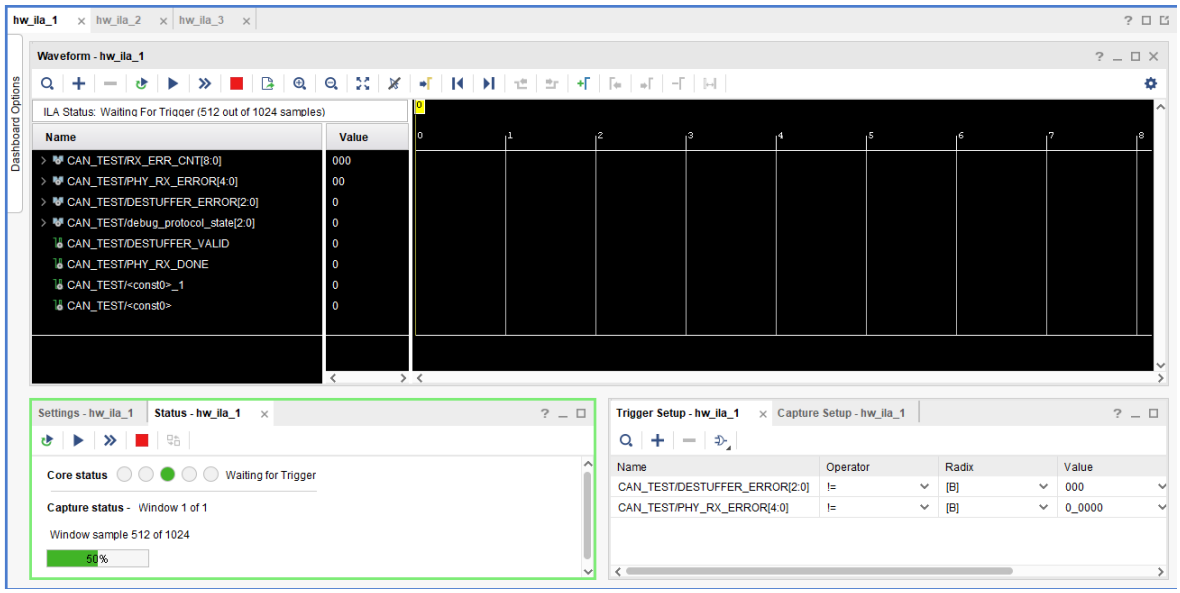


Şekil 4.38 Python Donanımsal Test Çıktısı

FPGA tarafında ise uygun olarak alınan paket sayısı bir sayaca bağlanmıştır. Bu sayacın gönderilen paket sayısı ile birebir aynı olması beklenmiştir. Test sonucunda 100 milyon paket başarılı olarak alınmıştır. Alınan doğru paket sayısı tümleşik lojik analizörden gözlemlenmiştir. Bu gözlemin çıktısı Şekil 4.39'da verilmiştir. Görselde kırmızı ile gösterilmiş olan "rcv_data_cnt" parametresi alınan doğru paket sayısını göstermektedir. Buna ek olarak Şekil 4.40'ta gösterilmiş olan başka bir tümleşik lojik analizör çıktısında CAN IP çekirdeğinin içerisinde gerçeklenmiş olan hata bayrakları takip edilmiştir. Bir hata olması durumunda lojik analizör tetiklenecek (trigger) şekilde ayarlanmıştır. Tüm test boyunca hiçbir hata algılanmadığı için lojik analizör tetiklenmemiştir. Bu test sonucunda CAN IP çekirdeğinin doğru şekilde çalıştığı doğrulanmıştır.



Şekil 4.39 Tümleşik Lojik Analizör Alınan Doğru Paket Sayısı

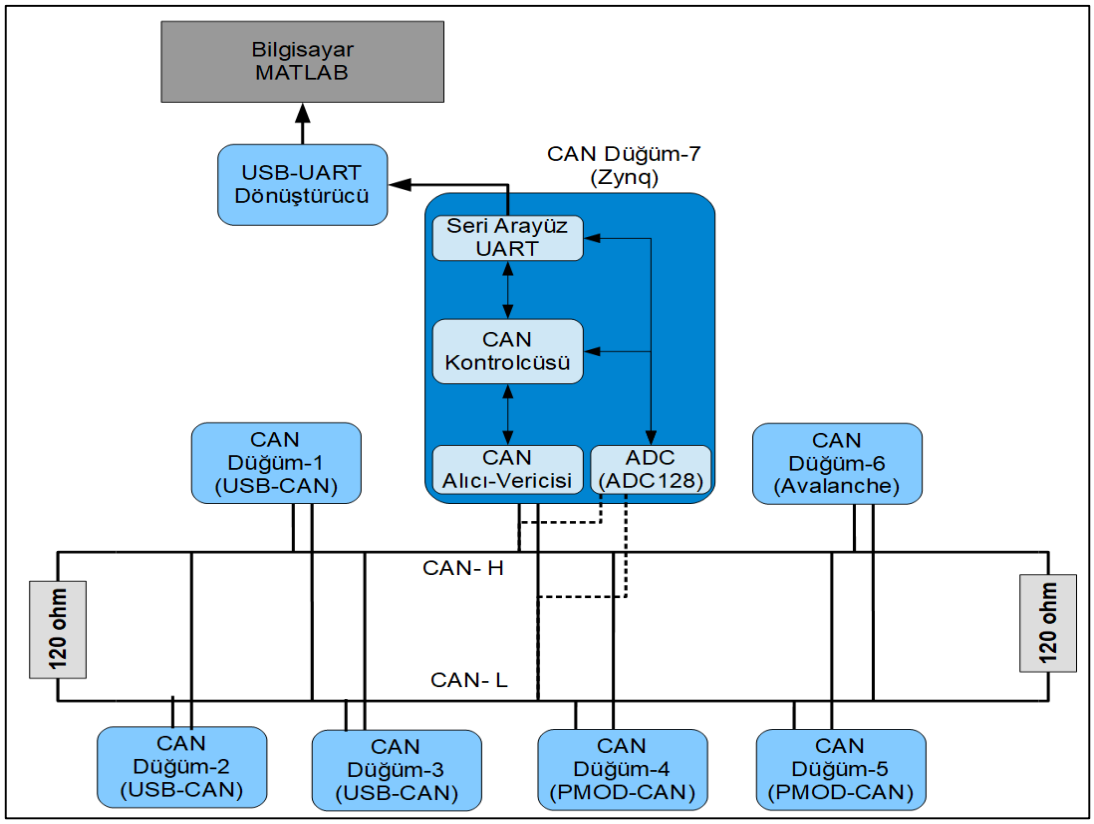


Şekil 4.40 Tümleşik Lojik Analizör Hata Çıktı Takibi

4.5. Entegrasyon ve Veri Sentezlenmesi

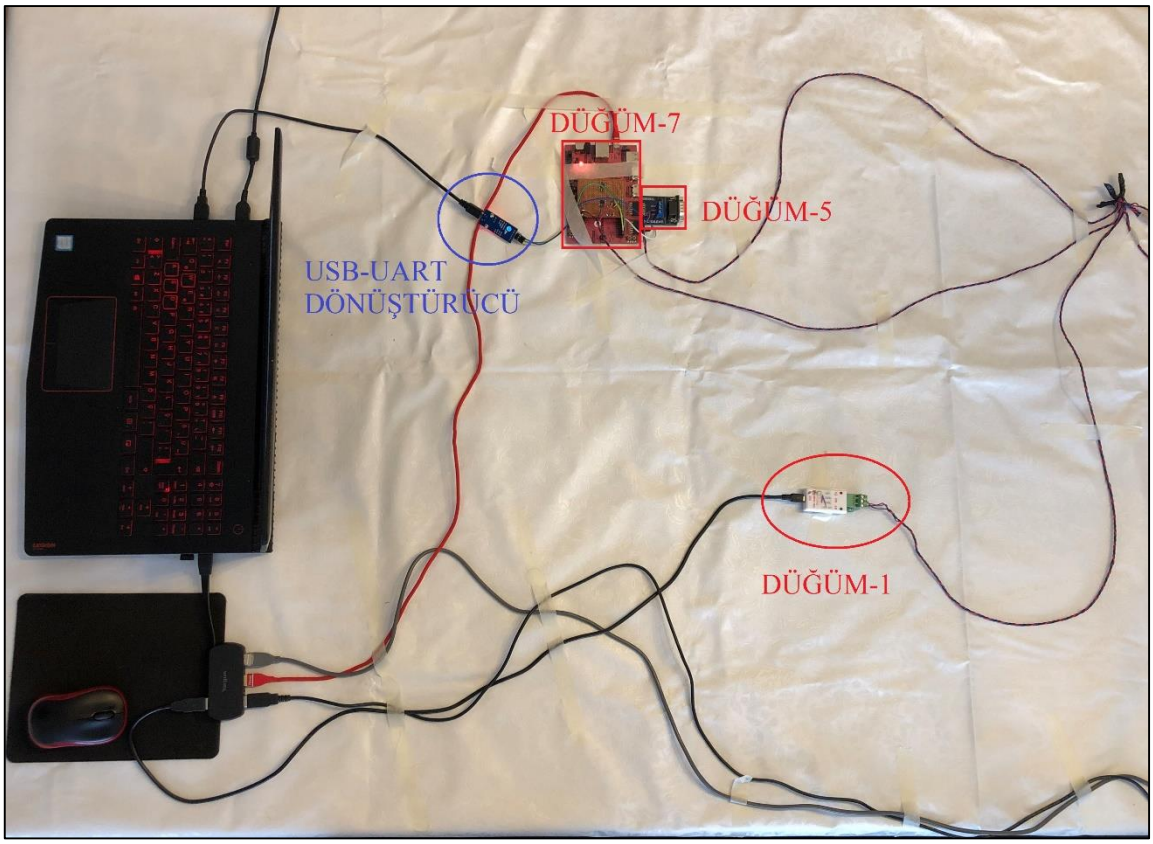
Tez kapsamında CAN sinyallerinin parmak izlerinin çıkarılması için toplamda 7 düğüm içeren bir CAN veri yolu üretilmiştir. Bu 7 düğümden biri tez kapsamında geliştirilmiş olan CAN IP çekirdeği ve buna bağlı ADC içeren PYNQ geliştirme kartıdır. Avalanche geliştirme kartı üzerinde ise geliştirilmiş CAN IP gönderici olarak kullanılmıştır. Geri kalan 5 düğüm ise standart olarak üretilen Bölüm 4.1.3 ve Bölüm 4.1.4'te verilmiş olan USB CAN çevirici ve PMOD CAN modülüdür. Birleştirilen CAN veri yolunun blok şeması Şekil 4.41'de gösterilmiştir. Birbirinin aynı CAN düğümlerinin kullanılması sistemin

çalışma performansını olumsuz etkileyeceği düşünülmüştür. Bunun sebebi ise aynı üretim cihazların birbirine çok yakın sinyaller üreteceğidir ancak sistemin bu koşullarda test edilmesi daha gerçekçi olacaktır. Bunun sebebi ise aynı üretim cihazların ürettiği sinyallerin parmak izlerinden düğümlerin sistemin kendi parçası veya dışarıdan bir saldırgan olduğu anlaşılabilir ise farklı cihazların ürettiği sinyaller çok daha farklı olacağından daha kolay tespit edileceği ön görülmüştür.

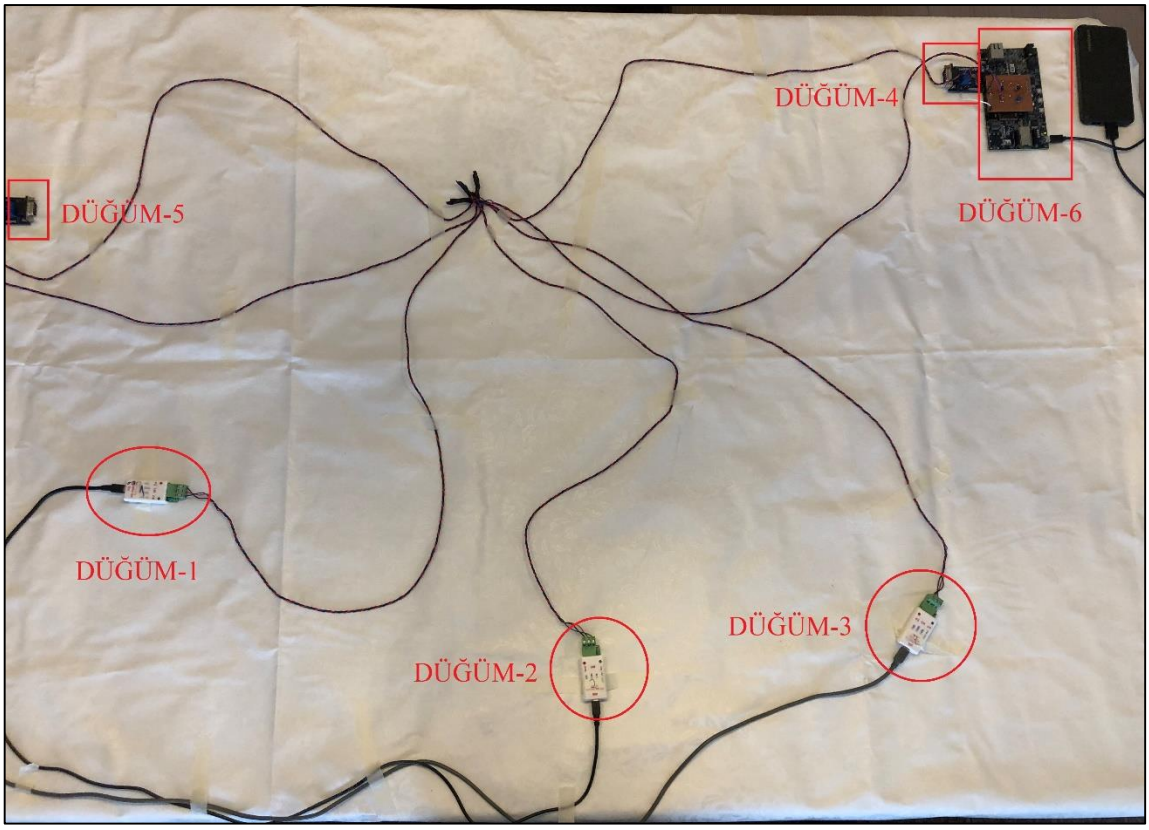


Şekil 4.41 Oluşturulan CAN Veri Yolu Blok Şeması

Sistemin test edilmesi için 3 adet USB CAN dönüştürücü, 2 Adet PMOD CAN modülü, Avalanche ve ZYNQ geliştirme kartları üzerinde kullanılmış olan CAN IP olmak üzere toplam 7 ECU'luk bir sistemin benzetimi yapılmıştır. Gerçek bir araç sisteminde çok daha fazla ECU bulunmaktadır. Üretilen bu sistem ise kavramın ispatı (proof of concept) için yeterli olduğu düşünülmüştür. Üretilen CAN veri yolunda sistemin benzetiminin doğru yapılması açısından her bir düğüm için 1 metrelik kablolar kullanılmıştır. Her bir düğümden CAN-H, CAN-L ve GND kabloları çıkarılmıştır ve bu kablolar yıldız topolojisine göre bir noktada birleştirilmiştir. Bu kablolar gürültünün sisteme etkisinin azaltılması için birbirlerine sarılı hale getirilmiştir. Sistemin görseli parçalar halinde Şekil 4.42 ve Şekil 4.43'te verilmiştir.



Şekil 4.42 Üretilen CAN Veri Yolu-1



Şekil 4.43 Üretilen CAN Veri Yolu-2

Sistemin donanımsal birleştirilmesinden sonra IDS sisteminin algılama birimi için kullanılacak makine öğrenmesi algoritmaları için veriler üretilmiştir. Veri sentezlenmesi için her düğüm için tek bir ID gönderilmesi düşünülmüştür. Bunun sebebi ise sisteme saldırı gerçekleştiren saldırganın başka bir cihazın gönderdiği ID'ye ait paketi taklit edeceği bu nedenle herhangi bir düğüme ait bir paketin kopyalanıp verinin içeriği değiştirilerek gönderileceği varsayımıdır. Bu nedenle her düğümün gönderdiği aynı ID'ye ait paketlerin hangi düğüme ait olduğu sadece sinyalin parmak izinden algılanıp algılanamayacağı sorusunun cevabı araştırılmıştır. Bu kapsamda sinyallerin parmak izinin çıkarılması açısından ID'si 0x1AB DLC'si 5 olan standart formatta olan paket seçilmiştir. Bu paket aynı zamanda simülasyonlarda kullanılmıştır. Paket içerisindeki 5 byte veri rastgele seçilmiştir.

Paketler USB CAN dönüştürücüler için Şekil 4.6'da gösterilmiş olan cihazın kendi GUI programından gönderilmiştir. PMOD CAN modülleri için ise FPGA'de modülleri süren kod parçaları hazırlanmıştır. Sistemde kullanılan ADC'nin örnekleme frekansının 500 KSPS olmasından kaynaklı CAN veri yolunun veri hızının 5, 50 ve 100 Kbps olacak şekilde 3 ayrı veri tabanı oluşturulmuştur. Veri hızının daha düşük olarak ayarlanması tez kapsamında veri yoluna sinyallerin özelliklerinin daha iyi çıkarılmasına olanak sağlamaktır. CAN veri yolunun daha hızlı çalıştırılıp efektif olarak örneklenmesi için ADC örnekleme hızının en az en az 5 kat daha fazla olması gerektiği düşünülmüştür. Nyquist frekansına göre bir sinyalin doğru örneklenebilmesi için en az 2 katı frekansta örneklenmesi gerekmektedir fakat bu hızlarda sinyalin özellikleri yüksek oranda kayıp olmaktadır. Bu nedenle en düşük 5 kat örnekleme frekansında çalışılmıştır. Farklı örnekleme oranları ile ADC'nin örnekleme hızının sistem üzerindeki etkisi de araştırılmıştır. Bu araştırma sonucunda sistemin saldırı tespit performansını etkilemeyecek en düşük örnekleme hızının belirlenmesi amaçlanmıştır.

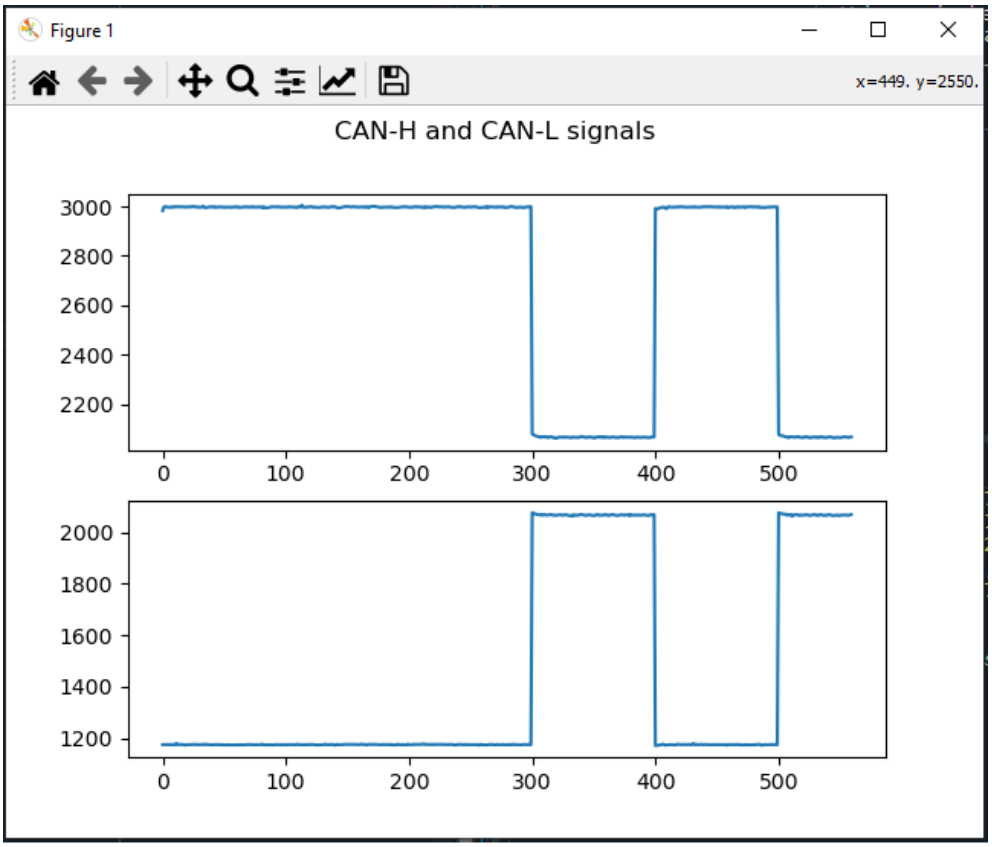
Paketler düğümlerden her iki paket arası 50 milisaniye beklemeler olacak şekilde gönderilmiştir. Paketlerin arasındaki bu aralığın sebebi örneklenmiş verileri gönderen UART arayüzünün hızından kaynaklanmaktadır. ADC bu hızda toplamda 2240 byte veri örneklemektedir. Bu boyuttaki verinin gönderilme süresi 921600 baudrate sahip UART arayüzü ile yaklaşık 24 milisaniye sürmektedir. Ekstra 26 milisaniye FIFO'nun taşmasını ve kayıt programının kilitlenmesini önlemek açısından eklenmiştir. Gönderilen verileri kayıt etmek için bir Python kod parçası yazılmıştır. Bu kod parçası gelen verileri bir .txt formatında bir dosyaya kayıt eder ayrıca veride bir problem olmadığını gözlemlemek amacıyla grafiğini de çizer. Bu kod parçasının geliştirilmesinde açık kaynak olan pyserial [45]ve Matplotlib [46]kütüphaneleri kullanılmıştır. Her bir düğüm için 50 bin örnek

alınmıştır. Buna ek olarak CAN veri hızının 5, 50, 100 Kbps hızları için veri örneklenmesi ayrı ayrı yapılmış olup toplamda 900 bin örnek veri sentezlenmiştir. Tüm kayıtların alınması yaklaşık 15 saat sürmüştür. Kayıtlar tamamlandıktan sonra MATLAB ortamına aktarılmıştır. Kayıt yapan kod parçasının ilgili kısmı Şekil 4.44'te verilmiştir.

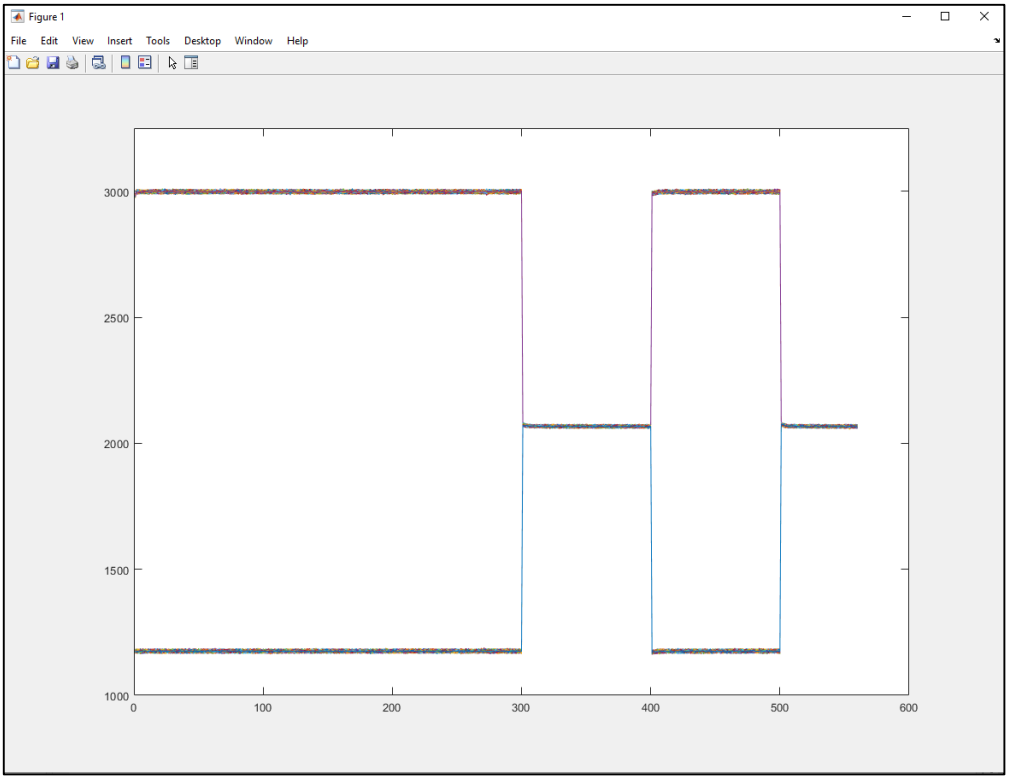
```
if dataValid:
    dataValid = False
    if len(channel_0_sample) == len(channel_1_sample):
        for i in channel_0_sample:
            f1.write("%d,"%(i))
        f1.write("\n")
        for i in channel_1_sample:
            f2.write("%d,"%(i))
        f2.write("\n")
        recv_cnt += 1
        print(recv_cnt)
        if recv_cnt == 50000:
            f1.close()
            f2.close()
            ser.close()
            break;
```

Şekil 4.44 Python Veri Kayıt Yapan Kod Parçası

Şekil 4.44'ten de anlaşılabileceği gibi her bir düğümden CAN-H, CAN-L özelliklerini barındıran toplamda 50 bin örnek toplanmıştır. Bu veriler her bir CAN paketinin kontrol kısmından alınmıştır. Bu kısımdan alınmasının sebebi ise uzlaşma kısmında aynı anda her iki düğümün veri gönderiyor olma ihtimalidir. Veri kısmından alınmamasının sebebi ise verinin değişken olabilme ihtimalidir. Kayıt etme sisteminin doğru çalıştığını garanti etmek adına Python programı sinyallerin grafiğini de çizmektedir. Sinyallerin gözlemle doğrulanmasından sonra kayıt işlemi otomatik olarak devam eder. Şekil 4.45'te Python üzerinden alınan sinyalin grafiği gösterilmiştir. Txt dosyasına kayıt edilen veriler MATLAB programına aktarılarak .mat dosyasına çevrilir. Aktarılan verilerin tamamının grafiği çizdirilerek alınan tüm verilerin doğru bir şekilde örneklenip kayıt edildiği gözlemle doğrulanır. Aynı düğümden alınan örneklerin birbirinden gözle görülür derecede farklı olmayacağı düşünülmüştür. Çizdirilen grafikte verilerin birbirinin üstüne denk gelecek şekilde olduğu gözlemlenmiştir. 50 bin örneğin çizdirildiği CAN düğüm grafiği Şekil 4.46'da gösterilmiştir. Bu aşamadan sonra tüm çalışmalar MATLAB ortamında gerçekleştirilmiştir.



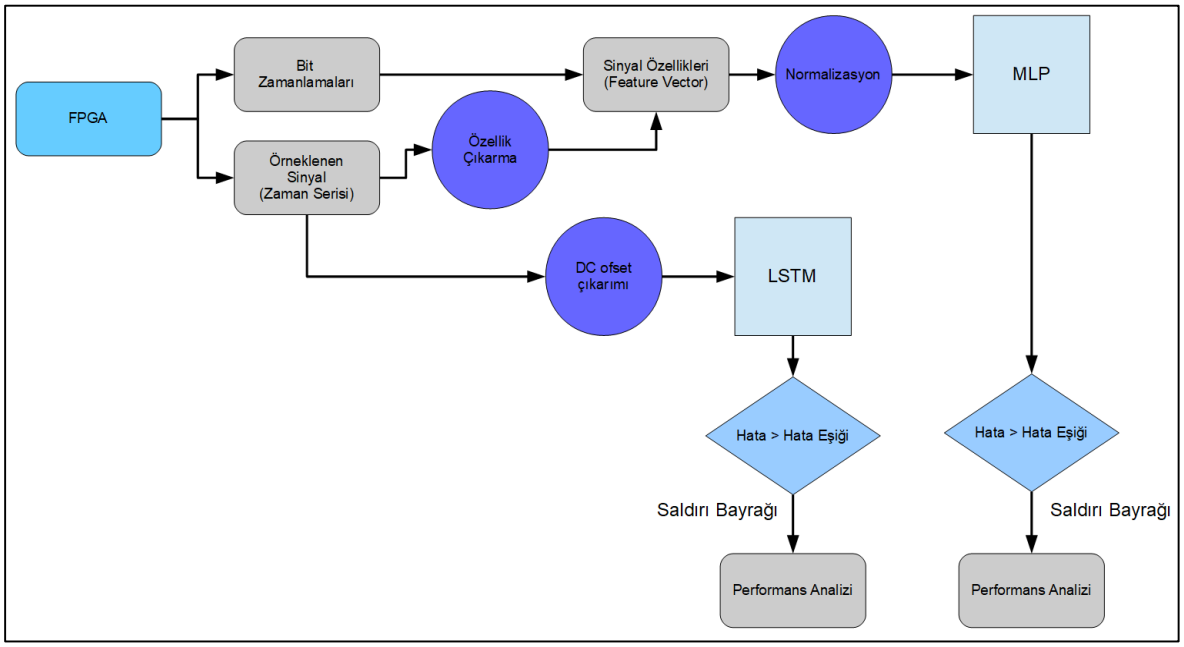
Şekil 4.45 CAN Düğümünden Python ile Alınan Veri Örneği



Şekil 4.46 CAN düğüm-1 50 bin Örnek Grafiği

4.6. Yapay Sinir Ağı Gerçeklemeleri

Yapay sinir ağı gerçeklemeleri MATLAB ortamında gerçekleştirilmiştir. Kayıt edilen her bir düğüm için 50 bin örnek, toplamda 300 bin örnek veri üzerinden yapay sinir ağları eğitilmiştir. Bu eğitim 3 farklı örnekleme frekansı için ayrı ayrı yapılmıştır. Eğitim için iki farklı yapay sinir ağı kullanılmıştır. Bu sinir ağlarından ilki MLP (Multi-Layer Perceoptron) diğeri ise LSTM'dir (Long-Short Term Memory). MLP sınıflandırma, örüntü algılama, fonksiyon yakınsama gibi işlemler için sıkça kullanılan bir yapay sinir ağı mimarisidir. LSTM ise zaman serisi tahminleri ve zaman serilerinin algılama için kullanılabilen bir yapay sinir ağıdır. Bu sebeple tez kapsamında Şekil 4.47 gösterilmiş IDS algılama birimi tasarlanmıştır. Bu mimaride MLP sinyalden çıkarılan özellik vektörleri ile örüntü tanıma işlemi için kullanılmıştır. LSTM için ise doğrudan zaman serisi tanımlama için kullanılmıştır. Zaman serisi değerlerinden CAN sinyalinin sahip olduğu 2.5V DC ofset karşılığı ADC değeri olan 2048 sayısı çıkarılarak eğitim yapılmıştır.



Şekil 4.47 IDS Yapay Sinir Ağı Tabanlı Tespit Birimi

FPGA'de gerçeklenmiş olan CAN IP birimi tarafından alınan zaman serisi örnekleri ve FPGA'de çıkarılan bit zamanları MATLAB ortamına aktarılmıştır. CAN-H ve CAN-L sinyallerinin kontrol kısmından alınan zaman serisi verilerinden özellik vektörleri çıkarılmıştır. Zaman sinyallerinden çıkarılan özellikler şunlardır:

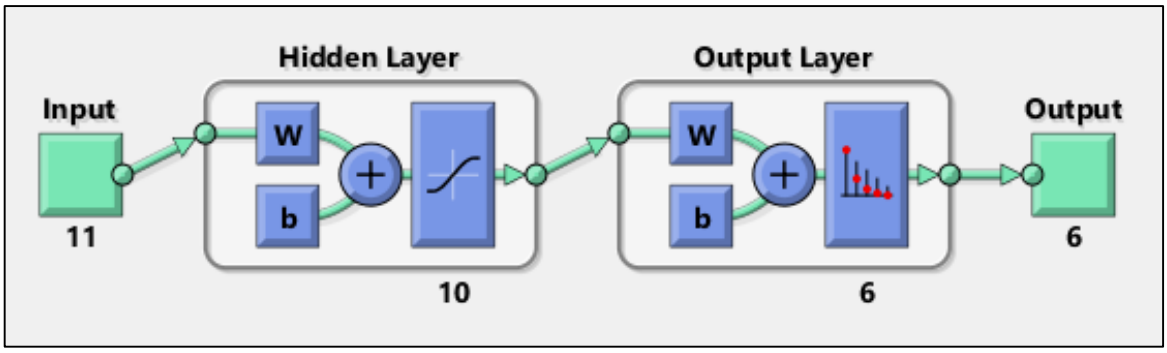
1. Ortalama
2. Standart Sapma

3. Ortalama Mutlak Sapma
4. Maksimum
5. Minimum
6. Bit Süresi

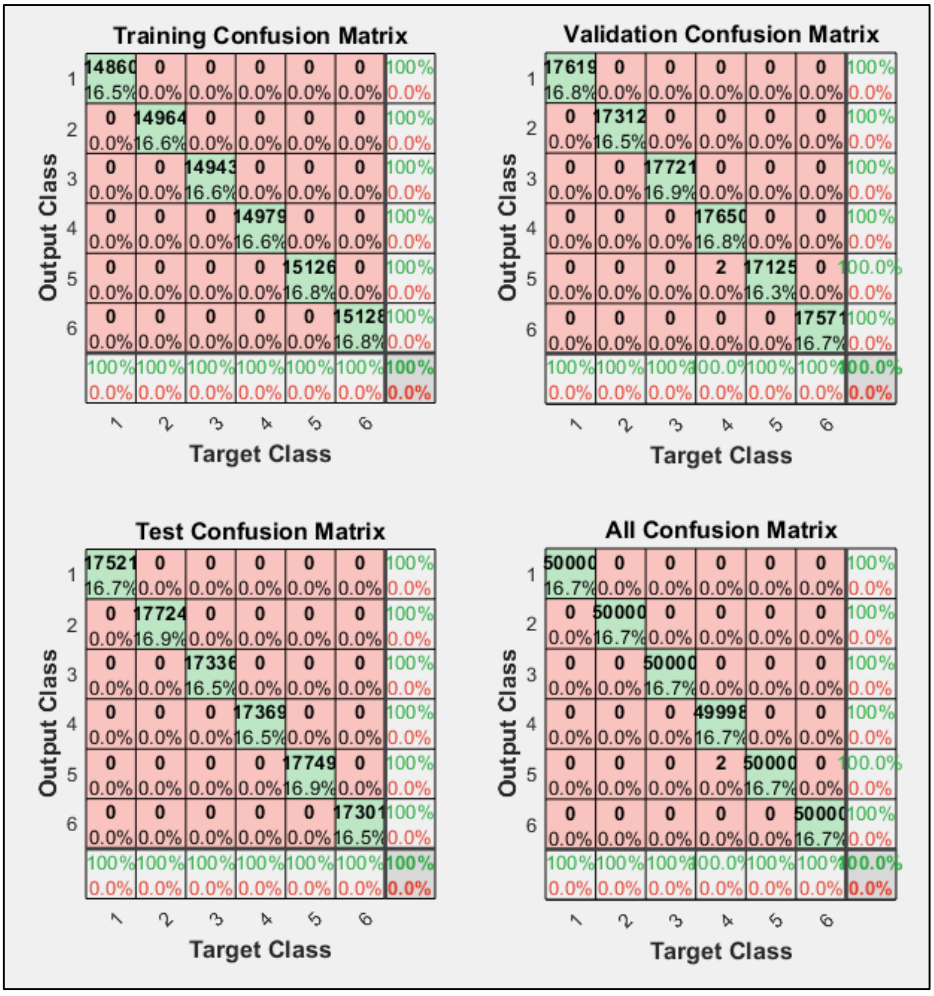
İlk beş özellik hem CAN-H hem de CAN-L için çıkarılmıştır ancak bit süresi her bir paket için tektir. Ayrıca sistemin eğitimi için sinyal yükselme, alçalma süreleri, sinyalin çarpıklığı (skewness) ve yağılması (kurtosis) gibi özelliklerinin de kullanılması düşünülmüştür ancak bu özellikler kullanılmamıştır. Bu özelliklerin kullanılmamasının detayları bölüm 5'te açıklanmıştır. Sinyallerden çıkarılan özelliklerin değerleri incelenmiştir. Sistem verilerinin MLP ile IDS sistemi gerçekleştirilmeden önce veriler ile sınıflandırma denemesi yapılmıştır. Bunun sebebi ise düğümlerden alınan verilerin birbirlerine çok yakın olması ve bu sistemin tamamen oluşturulmadan önce bu verilerin kullanılabilir olup olmadığının anlaşılmasının istenmesidir. Bu kapsamda MATLAB yapay sinir ağı geliştirme aracı (nnstart) [47] kullanılmıştır. Giriş katmanı, gizli katmanı ve çıkış katmanı olan yapay sinir ağı kullanılmıştır. Gizli katmanında 5 nöron bulunan bir ağ ile başlanmış ve hatasız sınıflandırma yapılabilmesi için kaç gizli katmanda kaç nöron gerektiği çıkarılmıştır. Gizli katmandaki nöron sayısı 5'in altında olan sistemde istenilen performans sağlanamamıştır. Nöron sayısı 10'a çıkarılınca sınıflandırma performansı artmıştır. MATLAB aracından alınan verilere göre 6 düğümden alınan 300 bin veri 2 hata ile sınıflandırılmıştır. MLP'de eğitim için "Levenberg-Marquardt backpropagation" algoritması kullanılmıştır. Performans analizi için 5-25 sayıları arasında değişken gizli nöron katmanları bulunduran farklı yapay sinir ağları ile eğitim yapılmıştır. Gizli katmanda hiperbolik tanjant sigmoid transfer fonksiyonu kullanılmıştır. Çıkış katmanında ise lineer transfer fonksiyonu kullanılmıştır. Ağırlıkların başlangıç değerleri ise Nguyen-Widrow algoritmasına göre ilklendirilmiştir. Kullanılan yapay sinir ağının blok şeması Şekil 4.48'de, özellikleri ise Tablo 4.4'te verilmiştir. Sınıflandırma sonuçları ise Şekil 4.49'da verilmiştir.

Tablo 4.4 MLP Özellikleri

MLP Özelliği	Değer
Giriş Sayısı	11
Çıkış Sayısı	5
Gizli Katman Sayısı	1
Gizli Katman Nöron Sayısı	5,10,15,20,25
Eğitim Algoritması	Levenberg-Marquardt
Aktivasyon fonksiyonu	Hiperbolik tanjant sigmoid
Ağırlık ilklendirme algoritması	Nguyen-Widrow



Şekil 4.48 Sınıflandırma MLP Yapay Sinir Ağı Özellikleri



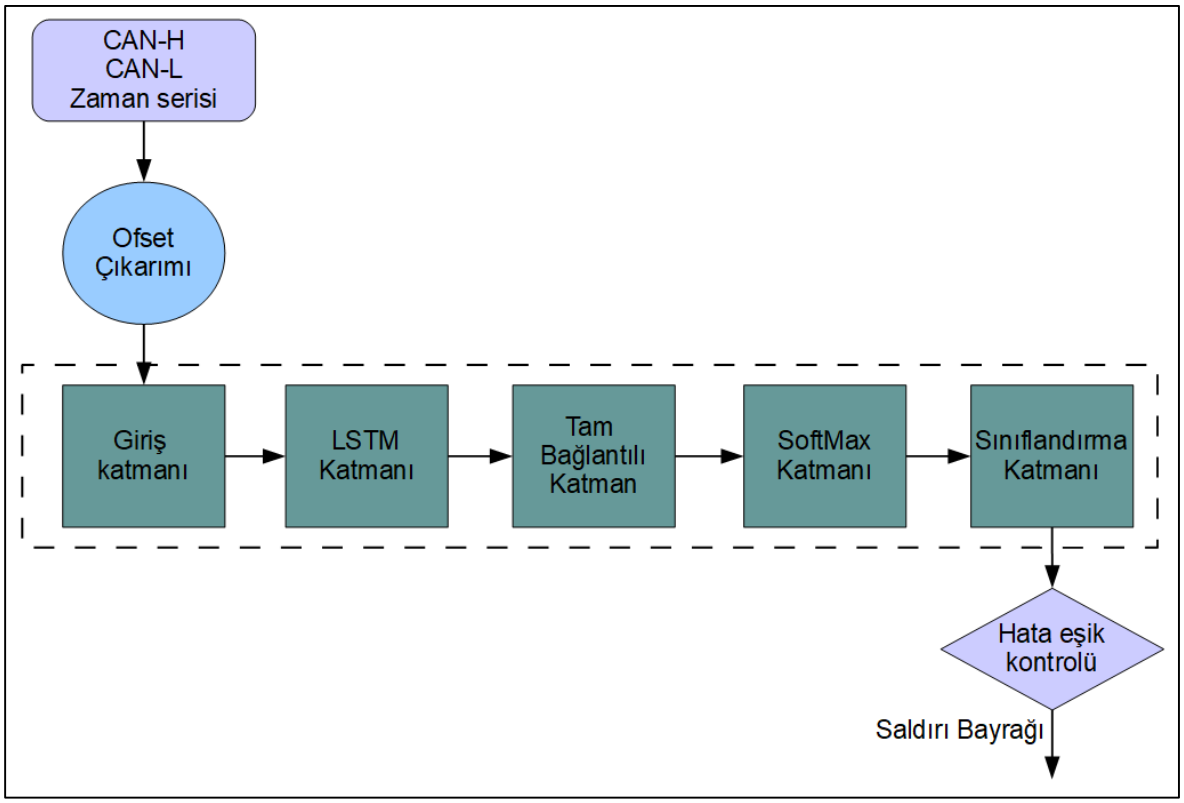
Şekil 4.49 Düğüm Verileri Sınıflandırma Sonuçları

Sınıflandırma işleminin başarılı bir şekilde yapılması gerçekleştirilecek sistemin performansı açısından umut verici bir sonuç olmuştur. Bu aşamadan sonra sistem için farklı bir yapay sinir ağı gerçeklemesi yapılmıştır. Bu sistemde ise bir düğümün değerleri eğitim setinden çıkarılmıştır. Çıkarılan bu düğüm, sisteme saldırı gerçekleştiren saldırgan olarak kullanılmıştır. Saldırgan düğüm olarak üç farklı düğüm kullanılmıştır ve bu düğümlere göre

ayrı performans sonuçları paylaşılmıştır. Önceki düğümlere göre eğitilen sisteme saldırgan verileri gönderildiğinde ise sistem verinin diğer düğüm verilerine olan benzerliğinin sonucunu vermektedir. Buna bağlı olarak da hata oranı çıkarılmaktadır. Hata değeri belirli bir eşiğin üzerinde ise sistem bunu saldırı olarak işaretlemektedir. Bu eşik değere göre hem eğitime katılmamış düğüm verileri ve eğitime katılmış düğüm verileri test edilmiştir. MLP sisteminin sonuçları bölüm 5'te verilmiştir.

LSTM yapay sinir ağı zaman serisi tahmini, zaman serilerini sınıflandırılması gibi problemlerde sıklıkla kullanılan bir yapay sinir ağı mimarisidir. Tez kapsamında CAN-H ve CAN-L hatları için örneklenen sinyalleri DC ofseti çıkarılarak 5 katmandan oluşan LSTM yapay sinir ağına verilir. İlk katman olan giriş katmanı verileri alan ve istendiği durumda normalizasyon işlemini gerçekleştirebilen bir katmandır. Tez kapsamında yapılan çalışmada CAN-H ve CAN-L sinyalleri için 2 giriş bulunmaktadır ve sinyal normalize edilmemektedir. İkinci katman LSTM katmanından oluşmaktadır. Bu katman yinelemeli bir yapay sinir ağıdır. İleri beslemeli (feed forward) yapay sinir ağlarından farklı olarak geri beslemeli bir yapıya sahiptir. Çalışmada kullanılan LSTM yapısı 2 girişlidir. Katmanda farklı sayılarda gizli hücre kullanılmıştır. Uygun sayıda gizli hücre sayısı çalışma kapsamında performans sonuçlarına göre belirlenmiştir. LSTM hücresi durum aktivasyonu için hiperbolik tanjant fonksiyonu, kapı aktivasyon fonksiyonu içinse sigmoid kullanılmıştır. Üçüncü katman tam bağlantılı katmandır. Bu katman girişleri ağırlık değerleri ile çarpar ve bias değeri ekleyerek çıkışa verir. Bu katmanın girişi LSTM katmanının çıkışı boyuttadır. Katmanın çıkış sayısı ise sınıflandırma sayısı kadardır. Saldırgan düğüm, sınıflandırmaya katılmadığı için tez kapsamında çıkış sayısı beştir. Tam bağlantılı katmamanın çıktısı softmax katmanına bağlanır. Softmax katmanı, tam bağlantı katmanından alınan ölçeklenmemiş değerleri 0-1 aralığındaki sınıflandırma skoru olarak ölçeklendirir. Sınıflandırma katmanı ise skor değerlerini cross-entropy hatasını hesaplar ve sınıflandırma işlemini gerçekleştir. Sınıflandırma katmanı çıktısına göre eğitim ve test sınıflandırması yapılır. Sınıflandırma sonuçlarına göre eğitimin uygun veya uygun olmadığı sonucu elde edilir. Eğitimi uygun bulunan yapay sinir ağının en kötü sonuç veren eğitim değerine göre bir hata eşiği belirlenir. Saldırgan düğümün verileri yapay sinir ağına verilir ve bu değerlere göre sınıflandırma skoru alınır. Alınan sınıflandırma skoru hata eşiğinden geçirilir. Hata eşiğinden yüksek olanlar saldırı olarak hesaplanır. Eşikten düşük olanlar ise normal veri olarak sınıflandırılır. LSTM yapay sinir ağının blok şeması Şekil 4.50'de gösterilmiştir. Ağın genel özelleri ise Tablo

4.5'te verilmiştir. LSTM yapay sinir ağı eğitimi için MALTAB programının derin öğrenme aracı (deep learning toolbox) [48] kullanılmıştır.



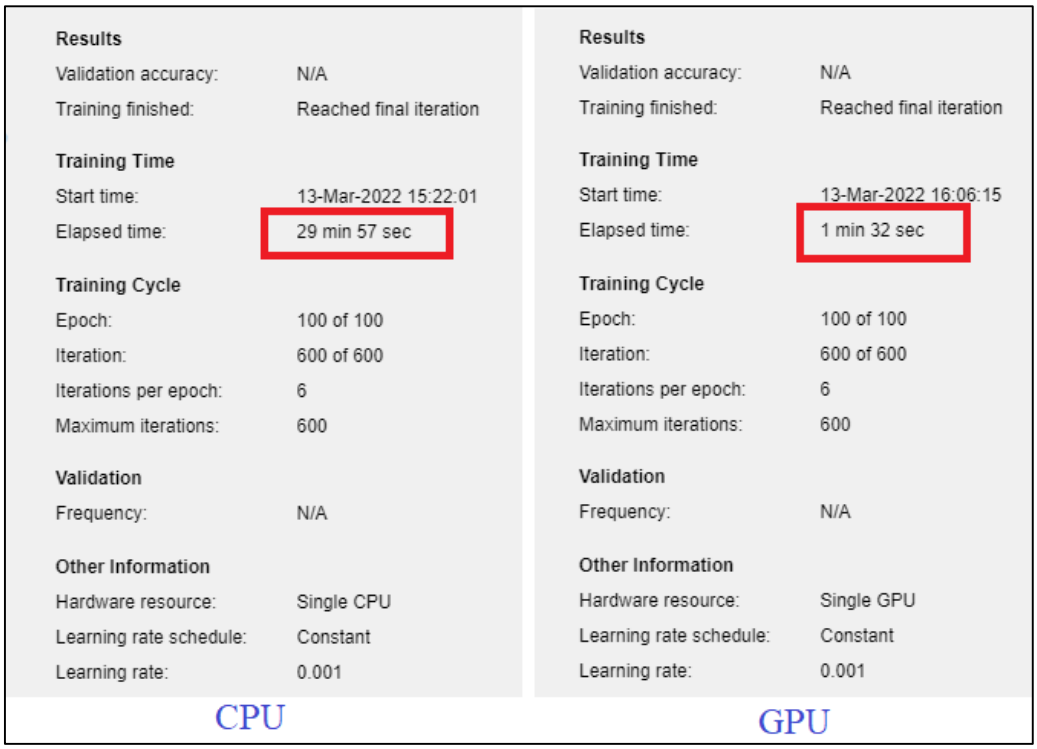
Şekil 4.50 LSTM Yapay Sinir Ağı

Tablo 4.5 LSTM Özellikleri

LSTM Özelliği	Değer
Giriş Sayısı	2
Çıkış Sayısı	5
Normalizasyon	Yok
LSTM ağırlık ilklendirme algoritması	Glorot
LSTM katman sayısı	1
LSTM gizli hücre sayısı	25,50,100,150,200
Durum Aktivasyon Fonksiyonu	Hiperbolik Tanjant
Kapı Aktivasyon Fonksiyonu	Sigmoid
Tam Bağlantılı Katman Ağırlık İlklendirme Algoriması	Glorot
Kayıp hesaplama fonksiyonu	Cross-Entropy

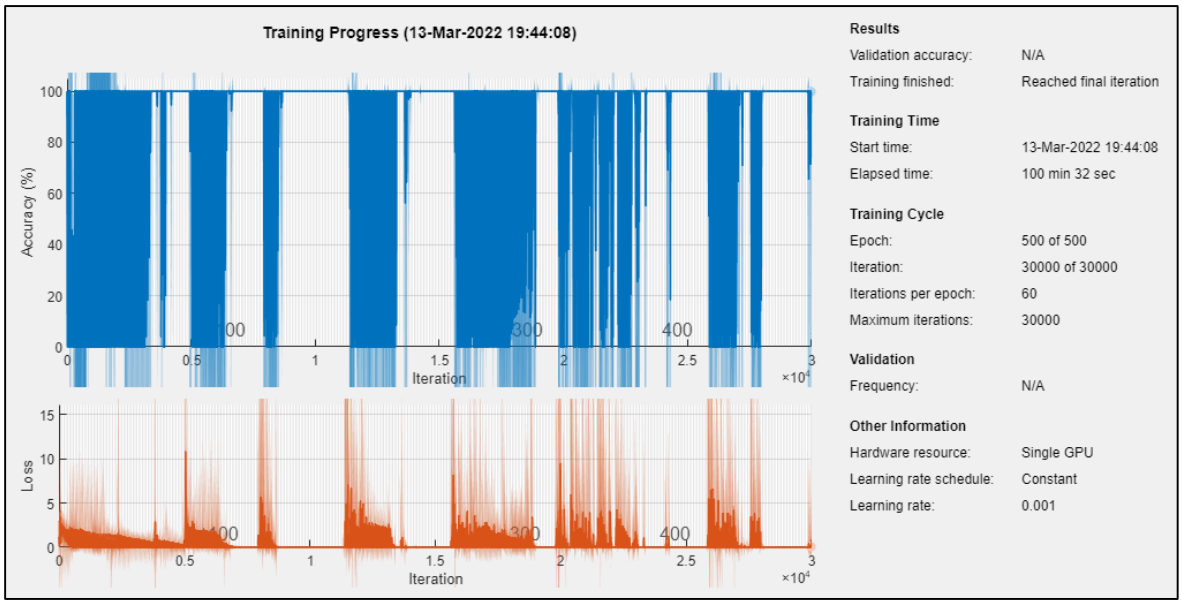
LSTM saldırı performans sonuçları hesaplanmadan önce, yapay sinir ağının sınıflandırma performansı test edilmiştir. LSTM yapısı her bir düğüm için 2x100 matris toplamda 2x600'lük bir matris ile eğitimi yapılmıştır. Eğitimde miniBatchSize parametresi 100, maxEpoch parametresi ise 100 olarak ayarlanmıştır. LSTM eğitim süresi MLP eğitim

süresine oranla çok daha uzun sürdüğü gözlemlenmiştir. LSTM yapay sinir ağının eğitilmesi yaklaşık olarak 30 dakika sürmüştür. Düşük boyutta bile eğitim süresinin çok fazla olması sebebiyle sistemin paralel olarak eğitim yapılması gerekliliği ortaya çıkmıştır. Bu sebeple eğitim için GPU kullanılması düşünülmüştür. MATLAB programının NVIDA ekran kartları için CUDA desteği bulunmaktadır. NVIDA CUDA toolkit kullanılarak MATLAB entegrasyonu yapılmıştır [49]. Bu sayede eğitim süreleri yaklaşık 20 kat kısaltılmıştır. Aynı parametreler ile eğitimi yapılmış LSTM yapay sinir ağının eğitim süreleri Şekil 4.51'de gösterilmiştir.

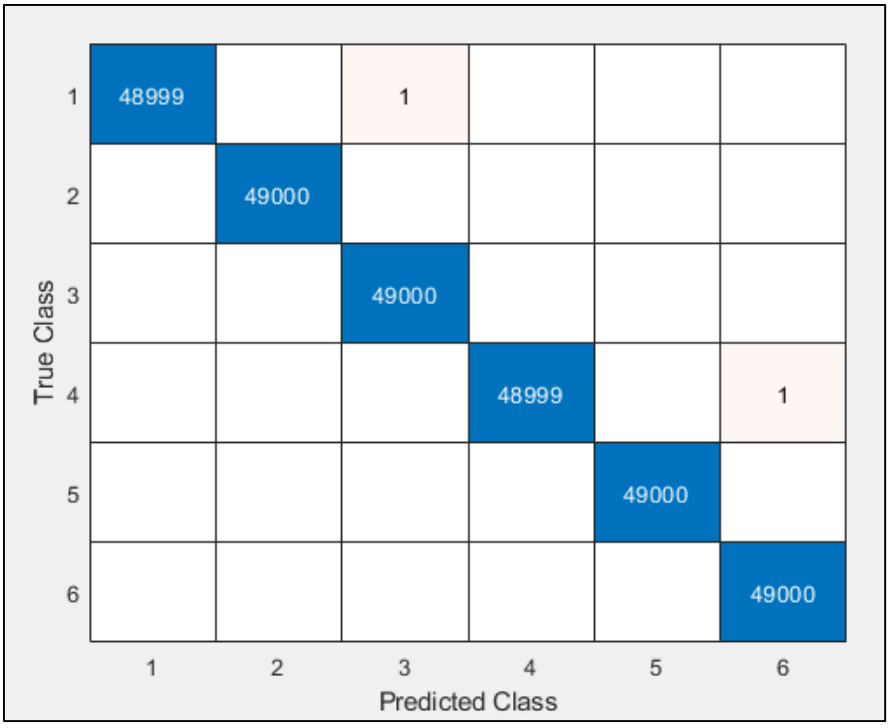


Şekil 4.51 LSTM CPU ve GPU Eğitim Süresi Kıyaslaması

Saldırı tespit performans analizi yapılmadan önce LSTM yapay sinir ağının sınıflandırma performansı incelenmiştir. Her bir düğüm için 1000 örnek alınarak toplamda 6000 örnekle sınıflandırma eğitimi yapılmıştır. Düğümlerden alınan her bir örnek 2x560 bir matristir. Eğitimde miniBatchSize parametresi 100, maxEpoch parametresi 500, öğrenme oranı ise 0.001 olarak ayarlanmıştır. Eğitim GPU üzerinde 100 dakika sürmüştür. Eğitimin görseli Şekil 4.52'de gösterilmiştir. Eğitilen yapay sinir ağı her bir düğüm için eğitimde kullanılmamış 49 bin örnek ile test edilmiştir. Test sonuçları Şekil 4.53'te gösterilmiştir. LSTM sınıflandırması MLP'ye benzer bir şekilde sadece 2 hata ile gerçekleştirilmiştir. LSTM eğitim süresi daha uzun sürmüştür fakat MLP'ye oranla daha düşük eğitim verisi ile aynı performans yakalanmıştır.



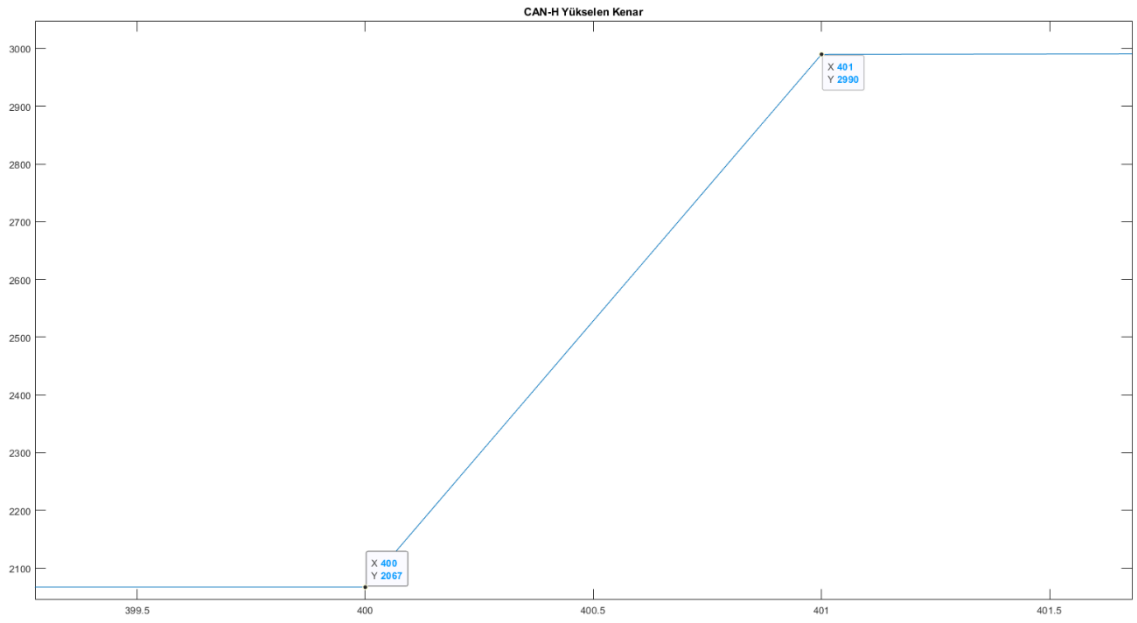
Şekil 4.52 LSTM Sınıflandırma Eğitimi



Şekil 4.53 LSTM Düğüm Sınıflandırma Performansı

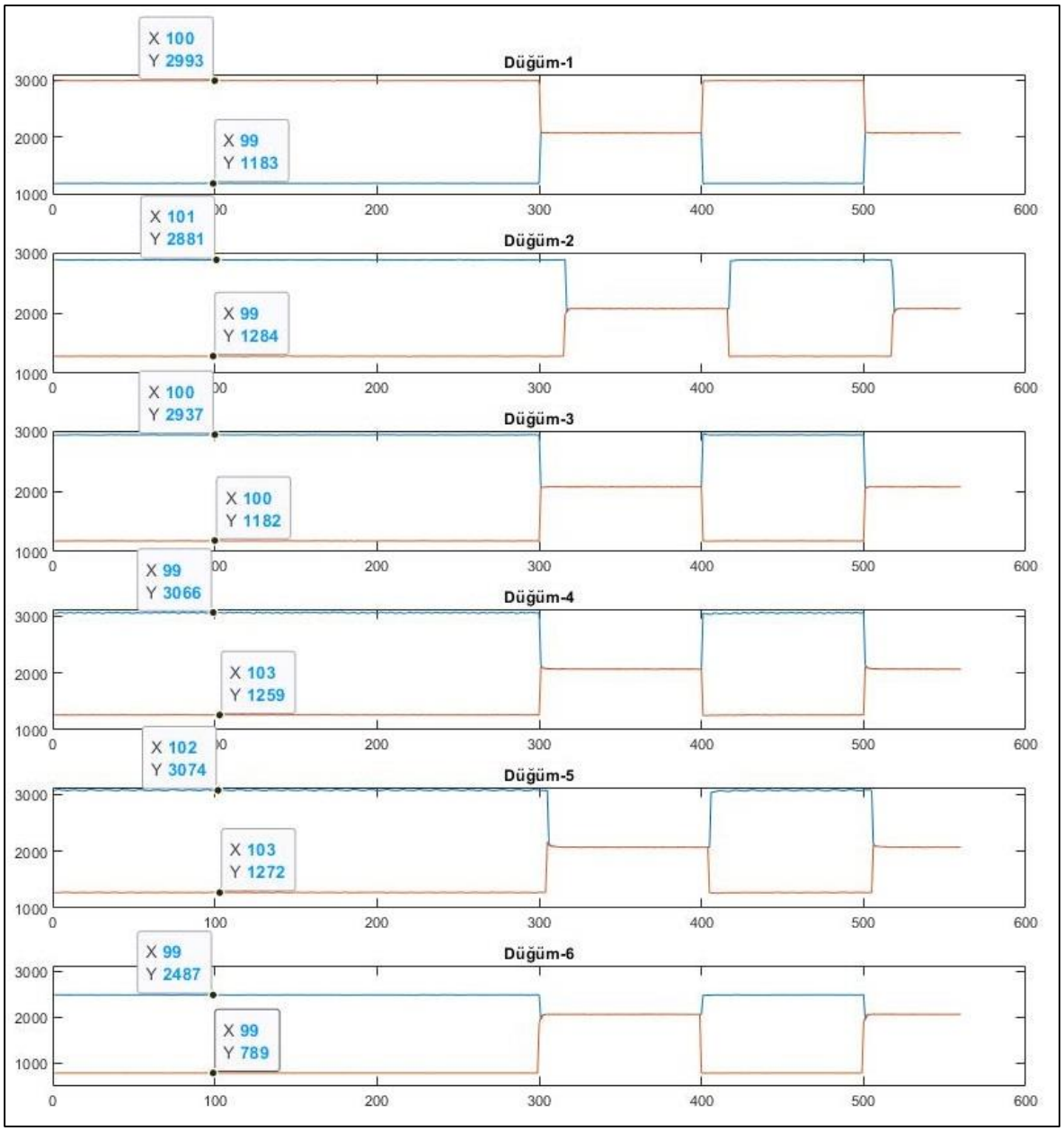
5. BULGULAR VE TARTIŞMA

CAN sinyallerinin kurulan CAN veri yolu sistemi üzerinden her bir düğüm için 50 bin veri toplanmıştır. Bu veriler üzerinden sinyallerden çıkarılabilecek sinyal parmak izleri analiz edilmiştir. Alınan örneklerin ilk incelemesinde ortaya çıkan en önemli bulgulardan bir tanesi örneklenen verilerden sinyalin yükselen ve düşen kenarlarında gözlemlenen farklılıkların gözlemlenememesidir. Bunun sebebi veri yolunun hızı düşürülse de sinyalin kalkış ve inişi kullanılan alıcı-verici tümleşik devresinden kaynaklanmaktadır. Bölüm 4.2'de gözlemlenmiş olan veriler 2,5 GSPS örnekleme hızındaki bir osiloskop ile alınmıştır ancak kullanılan ADC'nin veri örnekleme hızı 500 KSPS'dir ve osiloskopa göre çok yavaştır. Bu nedenle yükselen ve düşen kenarlarda gözlemlenen farklı sinyal durumları yapay sinir ağı eğitimlerinde kullanılamamıştır. ADC üzerinden alınan sinyalin yükselen kenarının görseli Şekil 5.1'de verilmiştir.



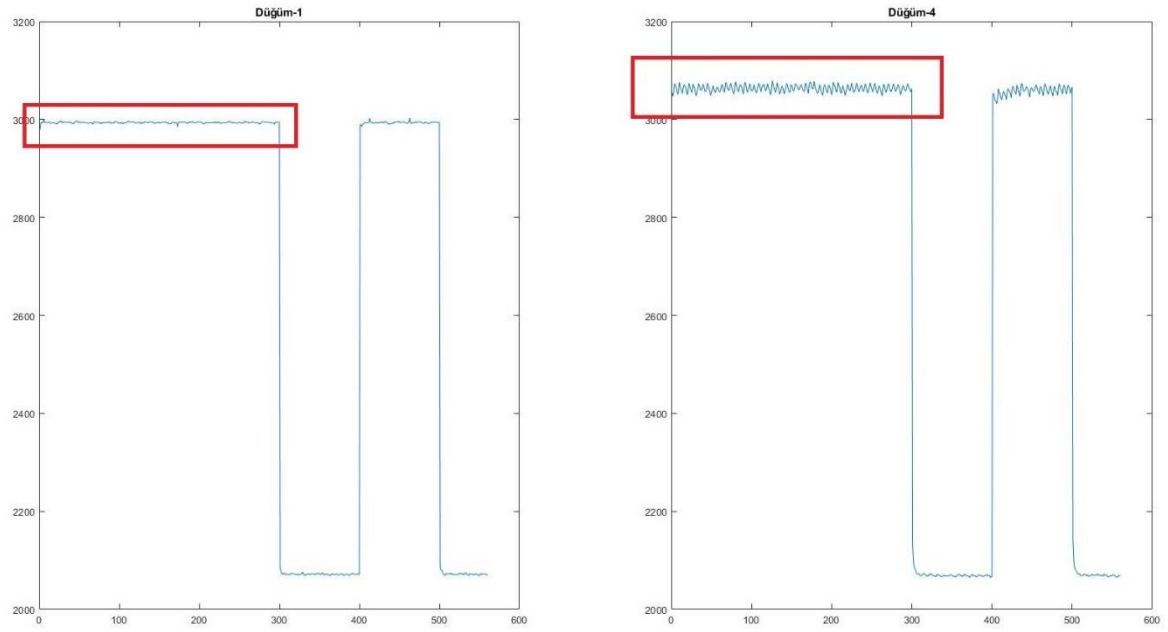
Şekil 5.1 Düğüm-1 CAN-H Yükselen Kenar

Kayıt edilen sinyallerin incelenmesi ile ortaya çıkan bir diğer bulgu ise, farklı düğümlerden alınan sinyallerin en yüksek ve en düşük noktalarının değişkenlik göstermesidir. Bu değişkenliğin düğümlerin sınıflandırılmasında kullanılabileceği düşünülmüştür bu nedenle özellik vektörlerine maksimum ve minimum değerleri eklenmiştir. Ayrıca, sabit durumdaki sinyal değerlerinin farklı olması sebebiyle sinyallerin ortalama değerlerinin de farklı olacağı düşünülerek ortalama değer de özellik vektörüne dahil edilmiştir. Düğümlerin maksimum ve minimum değerleri Şekil 5.2'de gösterilmiştir.

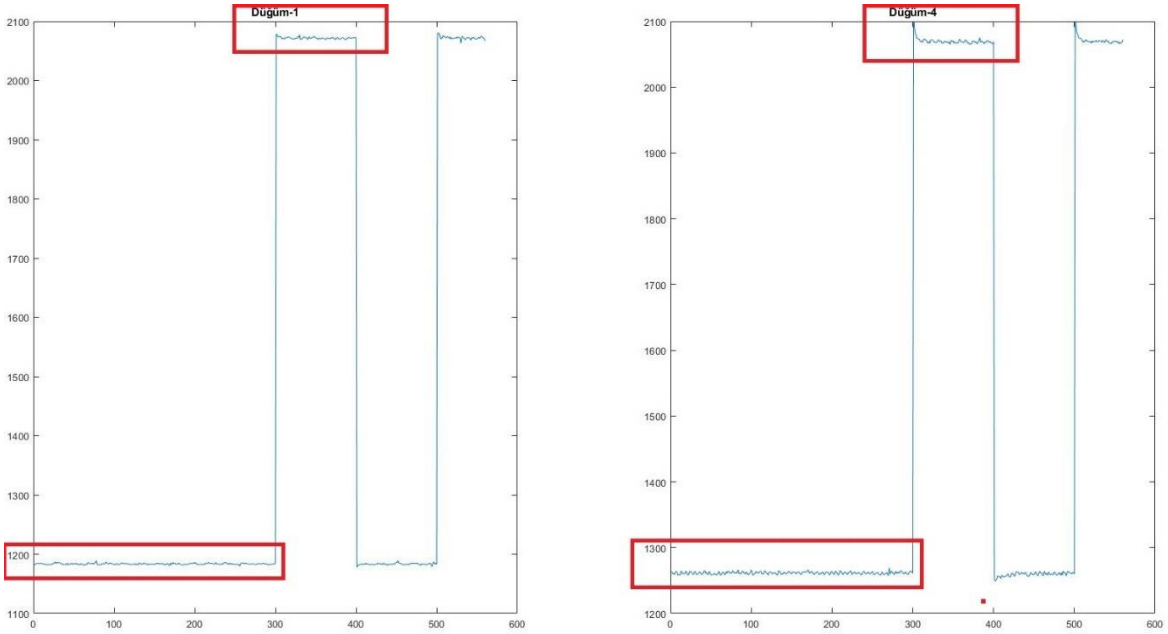


Şekil 5.2 CAN Düğümleri Sinyal Maksimum ve Minimum Değerleri

Farklı yapıdaki CAN düğümlerinin ürettiği sinyallerde cihazların iç yapılarına bağlı olarak sinyallerde sapmalar fark edilmiştir. Bu sapmaların sinyalleri birinden ayırt edebilmek için kullanılabileceği düşünülmüştür. Bu nedenle MLP eğitimi için çıkarılan özellik vektörüne standart sapma ve ortalama mutlak sapma da eklenmiştir. Düğümlerde gözlemlenen sinyal sapmalarının örnekleri Şekil 5.3 ve Şekil 5.4'te gösterilmiştir.



Şekil 5.3 5 Kbps Veri Hızı İçin Düğüm-1 ve Düğüm-4 CAN-H Sinyal Sapması



Şekil 5.4 5 Kbps Veri Hızı İçin Düğüm-1 ve Düğüm-4 CAN-L Sinyal Sapması

CAN sinyallerinden alınan zaman serisi verilerinden çıkarılan özellik vektörlerine göre yapay sinir ağları eğitilmiştir. Sistemler eğitilmeden önce çıkarılan özelliklerin değerleri düğümlere göre incelenmiştir. Bu sonuçlara göre sinyalin çarpıklık ve yığılma değerlerinin ortalamaları düğümler için çok yakın olmaktadır. Bu veriler birbirlerinden bazı düğümler için on binde bir farklılık göstermektedir. Verilerin bu kadar yakın olması nedeniyle bu verilerin sistem performansına etkisinin olmayacağı düşünülmüştür. Ayrıca,

sistemin gereksiz hesaplama yapmasının önüne geçilmek istenmiştir. Kalan sinyal özelliklerin sinyalin tanımlama için yeterli olduğu düşünülerek bu veriler eğitime katılmamıştır. CAN sinyallerinden alınan çarpıklık ve yığılma değerleri Tablo 5.1'de verilmiştir.

Tablo 5.1 CAN Sinyali Çıkarılan Özellikler

Düğüm No	Veri Yolu	Çarpıklık	Yığılma
Düğüm # 1	CAN-H	-0.948668	1.900082
	CAN-L	0.948676	1.900068
Düğüm #2	CAN-H	-1.118374	2.253467
	CAN-L	1.1125557	2.239506
Düğüm #3	CAN-H	-0.948628	1.900097
	CAN-L	0.948672	1.900070
Düğüm #4	CAN-H	-0.948321	1.900838
	CAN-L	0.948830	1.900823
Düğüm #5	CAN-H	-0.997249	1.996194
	CAN-L	0.998317	1.997895
Düğüm #6	CAN-H	-0.955018	1.925210
	CAN-L	0.940668	1.886812

MLP yapay sinir ağının eğitimi için kullanılan verilerin ortalama değerleri 5 Kbps, 50 Kbps ve 100 Kbps CAN veri hızları için sırasıyla Tablo 5.2, Tablo 5.3 ve Tablo 5.4'te verilmiştir. Bu değerler ADC'den alınan ham veridir. Değerler voltaja çevrilmemiştir. Değerlerin MLP'de kullanılmasının kolaylaştırılması için normalize edilmiştir.

Tablo 5.2 5 Kbps CAN Sinyali MLP Eğitimi Özellik Değer Ortalaması

Düğüm No	Veri Yolu	Ortalama	Std. Sapma	Ort. Mutlak Sapma	Mak. (ort.)	Min (ort.)	Bit Süresi (ort. Saat çevrimi)
Düğüm # 1	CAN-H	2730.322	416.836	376.274	3001.169	2066.547	102407.629
	CAN-L	1437.535	401.520	362.449	2080.363	1176.525	
Düğüm #2	CAN-H	2672.880	353.445	308.176	2889.635	2008.936	103404.644
	CAN-L	1485.567	343.991	300.356	2076.525	1275.157	
Düğüm #3	CAN-H	2688.434	390.899	352.858	2954.739	2053.652	102406.242
	CAN-L	1430.623	405.450	404.367	2076.151	1166.119	
Düğüm #4	CAN-H	2778.314	448.025	402.019	3077.783	2063.860	102431.372
	CAN-L	1492.793	365.535	329.951	2112.306	1250.986	
Düğüm #5	CAN-H	2797.600	448.023	400.472	3092.549	2067.235	102564.675
	CAN-L	1495.895	320.209	327.539	2162.269	1262.339	
Düğüm #6	CAN-H	2363.387	189.551	170.961	2486.705	1943.118	102378
	CAN-L	1155.011	577.178	521.847	2070.219	786.082	

Tablo 5.3 50 Kbps CAN Sinyali MLP Eğitimi Özellik Değer Ortalaması

Düğüm No	Veri Yolu	Ortalama	Std. Sapma	Ort. Mutlak Sapma	Mak. (ort.)	Min (ort.)	Bit Süresi (ort. Saat çevrimi)
Düğüm # 1	CAN-H	2730.390	419.578	375.681	2997.548	2068.698	10227.338
	CAN-L	1437.672	405.130	362.750	2079.998	1179.161	
Düğüm #2	CAN-H	2678.092	353.485	303.375	2887.541	2066.821	10338.630
	CAN-L	1480.458	344.415	295.594	2074.510	1279.290	
Düğüm #3	CAN-H	2687.927	394.924	353.591	2954.700	2053.652	10226.659
	CAN-L	1430.257	408.162	365.463	2074.034	1170.632	
Düğüm #4	CAN-H	2779.806	448.510	401.397	3075.051	2066.041	10244.121
	CAN-L	1493.836	370.412	331.595	2112.302	1251.796	
Düğüm #5	CAN-H	2804.961	446.575	391.954	3082.851	2069.351	10267.598
	CAN-L	1488.528	356.875	313.236	2082.100	1266.794	
Düğüm #6	CAN-H	2359.610	198.065	176.258	2485.864	1943.118	10217
	CAN-L	1151.246	578.439	517.803	2068.841	786.748	

Tablo 5.4 100 Kbps CAN Sinyali MLP Eğitimi Özellik Değer Ortalaması

Düğüm No	Veri Yolu	Ortalama	Std. Sapma	Ort. Mutlak Sapma	Mak. (ort.)	Min (ort.)	Bit Süresi (ort. Saat çevrimi)
Düğüm # 1	CAN-H	2730.498	422.587	374.912	2996.754	2069.389	5105.991
	CAN-L	1437.864	409.328	363.160	2079.995	1179.468	
Düğüm #2	CAN-H	2678.049	356.895	303.503	2887.371	2066.903	5168.295
	CAN-L	1480.399	347.575	295.581	2074.004	1279.964	
Düğüm #3	CAN-H	2687.391	399.638	354.524	2954.700	2053.652	5105.427
	CAN-L	1429.809	411.212	364.827	2073.185	1171.482	
Düğüm #4	CAN-H	2781.821	448.587	397.680	3073.806	2066.976	5123.568
	CAN-L	1495.203	376.384	333.816	2112.302	1251.844	
Düğüm #5	CAN-H	2822.586	441.309	375.280	3081.729	2069.755	5133.753
	CAN-L	1473.988	351.642	299.039	2074.439	1270.517	
Düğüm #6	CAN-H	2355.155	207.685	182.464	2485.494	1943.118	5097
	CAN-L	1148.605	579.685	514.083	2068.289	786.804	

İncelenen veriler sonucunda düğümlerin özellik vektörlerinin düğümleri saldırgan düğümden ayırt etmek için uygun olduğu düşünülmüştür. Buna ek olarak, ADC örnekleme hızının özellik vektörü değerlerine negatif etkisi gözlemlenmemiştir. Her bir düğüme ait CAN-H ve CAN-L sinyalleri için beş farklı özellik bunlara ek olarak bit süresi özelliği de eklenerek toplamda 11 özellik bulunduran vektörler ile MLP eğitimi yapılmıştır. MLP eğitiminde her bir düğüm için 10,15,20,25 ve 30 bin eğitim vektörü kullanılmıştır. Kalan veriler ise test için kullanılmıştır. Eğitimler 5 gizli nöron sayısı ile başlatılmıştır. İstenilen

performans elde edilememesi durumunda gizli nöron sayısı 5 arttırılarak tekrar eğitim yapılmıştır. MLP eğitimi sonucunda elde edilen saldırı tespit performans verileri EK 1'de tablo olarak verilmiştir. Tabloda aynı senaryo numarasına sahip veriler, aynı eğitime sahip yapay sinir ağı ile alınmıştır.

Farklı düğümlere göre saldırı tespit performansı incelenebilmesi, 2. 4. ve 6. düğümler sırasıyla saldırgan olarak seçilmiştir. Sistemde aynı anda sadece bir saldırgan düğüm olduğu varsayılmıştır. Saldırgan düğümün verileri, eğitim verilerine dahil edilmez. Özetle, MLP yapay sinir ağının saldırgan düğüme ait hiçbir bilgiye sahip değildir.

Eşik değeri, eğitim verileri içerisindeki en kötü hata oranından fazla olacak şekilde seçilmiştir. Bu sayede eğitim verileri her zaman saldırgan olmayan düğümler olarak sınıflandırılmıştır.

10 bin verilik eğitim setleri istenilen performansa ulaşılmıştır. Eğitim veri setinin 10 bin üzerinde olması sistemin performansına pozitif bir etki sağlamamıştır. Ayrıca, eğitim veri sayısının artması ile benzer performans elde etmek için daha fazla gizli nöron kullanılması gerekmiştir. Hiçbir senaryoda saldırgan olmayan (yanlış pozitif) düğüm hata oranı sıfır olmamıştır. Elde edilen en düşük hata oranı % 0.0005'tir. 4. Düğümün saldırgan olarak seçildiği senaryolarda, istenilen performansın sağlanabilmesi için daha yüksek miktarda gizli nöron sayısı kullanılmıştır. Bunun sebebinin ise 4. Düğümün verilerinin 5. Düğüm verilerine benzer olmasıdır. 6. Düğümün saldırgan olduğu senaryolarda diğer senaryolara göre daha düşük sayıda gizli nöron kullanılmıştır. Bunun sebebi ise sistemde 6. Düğüme benzer başka bir düğümün olmamasıdır. Yüksek veri hızlarında istenilen performans, düşük veri hızlarına göre daha fazla miktarda gizli nöron sayısı kullanılarak erişilmiştir.

LSTM yapay sinir ağı için bir özellik vektörleri oluşturulmamıştır. Örneklenen CAN-H ve CAN-L zaman serileri üzerinden eğitilmiştir. LSTM yapay sinir ağının girişi CAN-H ve CAN-L girişlerini alacak şekilde iki girişlidir. Her bir düğüm örneği 5 Kbps CAN veri hızı için 2x560'lık bir matris, 50 Kbps için 2x112'lik bir matris ve 100 Kbps veri hızı için 2x56'lık bir matristir. 5 Kbps veri hızı için daha fazla örnek toplandığı için bu veri tabanı için eğitim diğerlerine göre daha uzun sürmüştür.

İlk aşamada CAN sinyalleri üzerinde herhangi bir ön işlem yapılmadan eğitilmeye çalışılmıştır. Bu denemeler sonucunda sistemin eğitim performansının istenilen değere çıkmadığı veya çok uzun eğitimler sonucu istenilen değere ulaşıldığı gözlemlenmiştir.

Örneklenen sinyallerin pozitif yüksek değerler olduğu ver normalizasyon işlemi gerçekleştirilmediği de göz önünde bulundurularak, sinyallerden CAN sinyaline ait 2.5V değere sahip DC ofset değeri çıkarılarak tekrar eğitim yapılmıştır. DC ofseti çıkarılmış eğitimlerde, eğitim performansı istenen değere DC ofset çıkarılmamış olan sinyallere göre çok daha hızlı ulaşmıştır. Bu bilgi ışığında tüm eğitimler sinyallerden DC ofset çıkarılarak yapılmıştır. LSTM saldırı performans sonuçları EK 2'de tablo olarak verilmiştir. Alınan sonuçlara göre LSTM yapısı MLP'ye göre çok daha düşük eğitim sayısı ile istenilen performansa ulaşabilmektedir. LSTM performansı eğitimden eğitime büyük oranda değişmektedir. İstenilen performansa erişmek için aynı parametrelerle LSTM yapay sinir ağı tekrar tekrar eğitilmiştir. Eğitimlerde MaxEpoch parametresi ile eğitim iterasyon sayısı arttırılarak test edilmiştir. Bin örnekli eğitimlerde, eğitim süresi çok uzun sürmesi nedeniyle daha yüksek eğitim setleri kullanılmamıştır fakat düşük eğitim setleriyle bile yüksek saldırı tespit performansı elde edilebilmiştir. Performans testleri 10 gizli hücre sayısı ile başlatılmıştır. 10 gizli hücre sayısı ile 100 örnekli senaryolarda eğitim yeterli başarıya ulaşsa da 500 ve 1000 örnekli eğitimlerde yapay sinir ağı eğitilememiştir. Eğitimi yapılamamış ağlarda MaxEpoch parametresi veya gizli hücre sayısı arttırılarak tekrar test edilmiştir. Eğitim süresi ve performansı göz önünde bulundurularak 100 gizli hücrenin sistem için uygun olduğu deneme-yanılma yöntemi ile belirlenmiştir. 5 Kbps CAN veri hızı ile oluşturulmuş veri tabanında 4. Düğüm 100 eğitim verisi ile yapılmış eğitimlerde istenilen saldırı tespit oranına erişilememiştir. Bunun sebebinin eğitimin yeteri kadar tekrar edilmemesinden olduğu düşünülmüştür. Daha yüksek iterasyonlu eğitimin yapılması gerektiği ancak eğitim süresinin günler mertebesinde sürmesinden dolayı denenememiştir. Senaryolarda saldırgan olmayan düğüm hata oranlarında sıfır hata oranına ulaşılan durumlarda, saldırgan düğüm hata oranı olarak sıfır yakalanamamıştır. Özetle hem saldırgan hem de saldırgan olmayan hata oranlarının aynı anda sıfır olduğu bir senaryo bulunmamaktadır. Sıfır hata oranına çok yakın olan LSTM yapıları eğitilmiştir. Düğüm 6 için yapılan eğitimler diğer düğümlere göre yüksek performansa daha hızlı erişmiştir. Bunun sebebinin oluşturulan CAN veri yolunda düğüm 6'nın iç yapısına benzer başka bir düğüm bulunmamasıdır. Buna zıt olarak ise düğüm 4 eğitimlerinin istenilen performansa erişebilmesi için daha çok iterasyon ve daha çok tekrar gerekmiştir. Bunun sebebinin ise düğüm-4 ile düğüm-5'in değerlerinin diğer düğümlere göre göreceli olarak birbirine yakın olmasıdır. 50 Kbps ve 100 Kbps CAN veri hızlarında veri tabanı veri sayısı daha az olduğundan eğitimler daha kısa sürmüştür. Düğümler için alınan düşük veri boyutları saldırı tespit performansını etkilemediği sonucuna ulaşılmıştır.

6. SONUÇ VE ÖNERİLER

CAN veri yolu günümüz otomobillerin içerisindeki elektronik kontrol birimlerinin iletişimini sağlayan en temel veri yoludur. Gelişen otomotiv sektörü ve teknoloji ile araç içerisindeki ECU'ların sayısı yüksek oranda artmıştır. Bununla beraber günümüzdeki araçlarda kullanılan iletişim ağlarının dış dünya ile iletişimi bulunmaktadır. Yapılan çalışmalar sonucu CAN veri yoluna yetkisiz girişler yapılabildiği kanıtlar ile ortaya konulmuştur. CAN veri yoluna yapılan yetkisiz girişleri engellemek için kullanılan yöntemlerden biri de IDS sistemleridir. IDS sistemleri veri yoluna yapılan girişleri tespit etmek ve hatta karşı önlemler için günümüzde hem otomotiv hem de bilişim sektöründe yaygın olarak kullanılmaktadır.

Bu tezde, otomotiv sektöründe kullanılan CAN 2.0B veri yolu için kullanılabilecek bir saldırı tespit sistemi tasarlanmış ve gerçeklenmiştir. Tasarlanan IDS sistemi CAN veri yoluna kolaylıkla yapılabilecek fiziksel saldırıları, ECU'ların veri yolunda oluşturdukları sinyal karakteristiklerine göre tespit edebilmesi araştırılmıştır. Geliştirilen IDS sisteminin test edilmesi için 7 CAN düğümü içeren bir CAN veri yolu oluşturulmuştur. Bu düğümlerden, örnekleyen düğüm hariç, 5, 50 ve 100 Kbps veri hızları için ayrı ve her bir düğüm için ayrı ayrı 50 bin örnek, toplamda 900 bin örnek toplanmıştır. Bu düğümlerden 3 tanesi sırasıyla saldırıyı gerçekleştirecek düğüm olacak şekilde farklı senaryolarda test edilmiştir. Saldırı tespit birimi için MLP ve LSTM yapay sinir ağları kullanılmıştır. Bu mimarileri saldırı tespit performansları analiz edilmiştir.

Bu tez kapsamında geliştirilmiş olan tasarımın FPGA üzerinde gerçeklemesi için Xilinx Vivado 2019.2 programı kullanılmıştır. LSTM ve MLP eğitimi ve analizi için MATLAB programı kullanılmıştır. LSTM eğitimleri NVDIA toolkit ile GPU kullanılarak hızlandırılmıştır. Veri kayıt ve örneklenen verileri gerçek zamanlı görüntüleme için Python programı kullanılmıştır.

Tez kapsamında, MLP ve LSTM yapay sinir ağları kullanılarak iki farklı saldırı tespit birimi tasarlanmıştır. MLP yapay sinir ağı, CAN veri yolunun örneklemesi ile elde edilen zaman serisi verilerinden çıkarılan özellik vektörleri kullanılmıştır. MLP için üretilen özellik vektörleri normalize edilerek eğitim yapılmış ve test edilmiştir. LSTM yapay sinir ağı ise, doğrudan zaman serisi verileri kullanılmıştır. LSTM için kullanılan zaman serisi verilerinden CAN veri yolunun sahip olduğu 2.5 volt DC ofset değeri çıkarılmış ve eğitimi yapılmıştır. Her iki yapay sinir ağı için farklı miktarlarda eğitim verisi ve gizli katman birim

sayısı ile test edilmiştir. Buna ek olarak farklı hızlardaki CAN veri yolları için aynı testler tekrar edilmiş ve ADC örnekleme hızının sistem performansına olan etkisi araştırılmıştır.

Elde edilen sonuçlar ışığında, MLP ve LSTM yapay sinir ağlarının her ikisi de hata tespiti açısından yüksek performans göstermiştir. MLP yapay sinir ağının eğitiminin LSTM'e göre çok daha hızlı yapılabildiği ortaya çıkmıştır. Bu nedenle MLP yapay sinir ağı daha fazla eğitim verisi ile eğitilebilmiştir. LSTM yapay sinir ağının eğitimi çok uzun sürmesi nedeniyle daha düşük miktarda eğitim verileri kullanılmıştır fakat daha düşük eğitim verileri ile bile MLP'nin göstermiş olduğu performansa benzer bir performans elde edilebilmiştir. LSTM ağının eğitimi için GPU kullanılmıştır.

MLP yapay sinir ağı için saldırgan düğümün hata oranının sıfır olduğu durumlarda saldırgan düğüm 2 için % , 0.0008 (Senaryo 5), saldırgan düğüm 4 için % 0.001 (Senaryo 8,9), saldırgan düğüm 6 için % 0.0005 (Senaryo 38) yanlış pozitif saldırı tespiti oranı elde edilmiştir. Bu oranlar elde edilen en iyi performans sonuçlarıdır.

LSTM yapay sinir ağı için alınan sonuçlar daha farklıdır. Bu sonuçlara göre düğüm 2 için %0 saldırgan düğüm hata oran ile (yanlış negatif), % 0.0004 (Senaryo 50) yanlış pozitif sonucu edilmiştir. Aynı düğüm için bir başka senaryoda % 0.1420 yanlış negatif hata oranı ile % 0 (Senaryo 29) yanlış pozitif oranı elde edilmiştir. Saldırgan düğüm 4 için % 0 yanlış negatif hata oranına karşılık, % 0.0004 (Senaryo 61) yanlış pozitif hata oranı elde edilmiştir. Saldırgan düğüm 6 için de % 0 yanlış negatif hata oranına karşılık, % 0.0004 (Senaryo 49) yanlış pozitif hata oranı elde edilmiştir. Genel olarak her bir saldırgan düğüm için %0 yanlış negatif, % 0.0004 yanlış pozitif sonuç elde edilebilmiştir. Bu değerler MLP performansları ile tutarlıdır.

MLP ve LSTM yapay sinir ağlarının performans sonuçlarının kıyaslanabilmesi için her bir saldırgan düğüme ait alınan en iyi sonuçlar için doğruluk (accuracy), kesinlik (precision), hassasiyet (recall), F-Skoru (F-Score) hesaplanmıştır. Bu parametrelerin hesaplanma formülasyonları aşağıda verilmiştir.

$$Doğruluk\ (Accuracy) = \frac{YN + DP}{DN + DP + YN + YP}$$

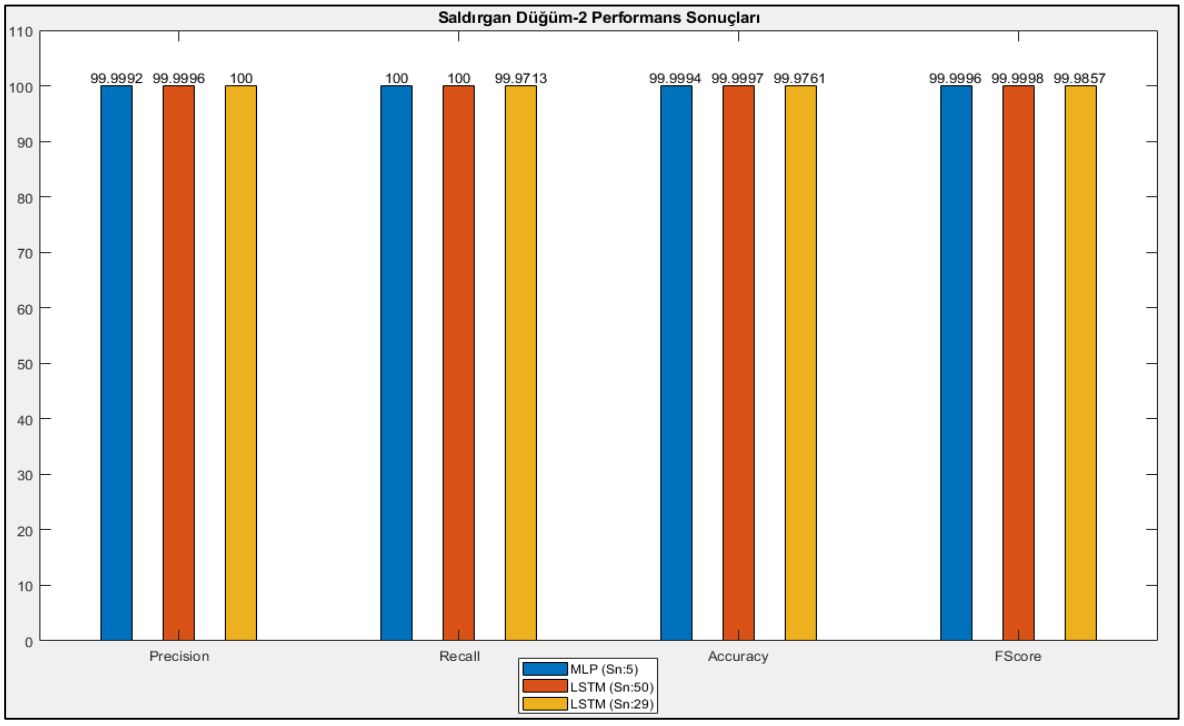
$$Kesinlik\ (Precision) = \frac{DP}{DP + YP}$$

$$Hassasiyet\ (Recall) = \frac{DP}{DP + YN} \qquad YN = Yanlış\ Negatif\ \ DP = Doğru\ Pozitif$$

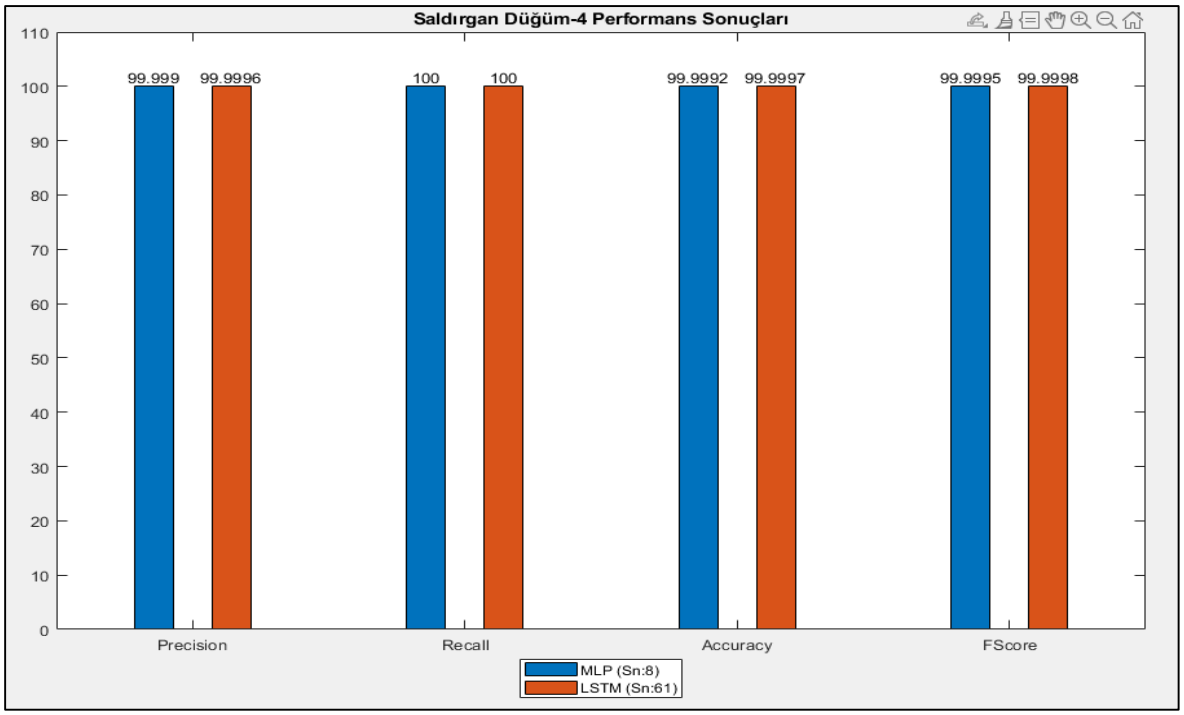
$$F - Skoru\ (F - Score) = 2 \times \frac{Kesinlik \times Hassasiyet}{Kesinlik + Hassasiyet}$$

Sonuçların kıyaslamaları sütun grafikleri üzerinden verilmiştir. Sonuçlar saldırgan düğüm olarak seçilen düğüm 2, düğüm 4 ve düğüm 6 için grafikler sırasıyla Şekil 6.1, Şekil 6.2 ve Şekil 6.3'te verilmiştir. Grafiklerden de anlaşıldığı gibi MLP ve LSTM saldırı tespit performansları birbirlerine çok yakındır.

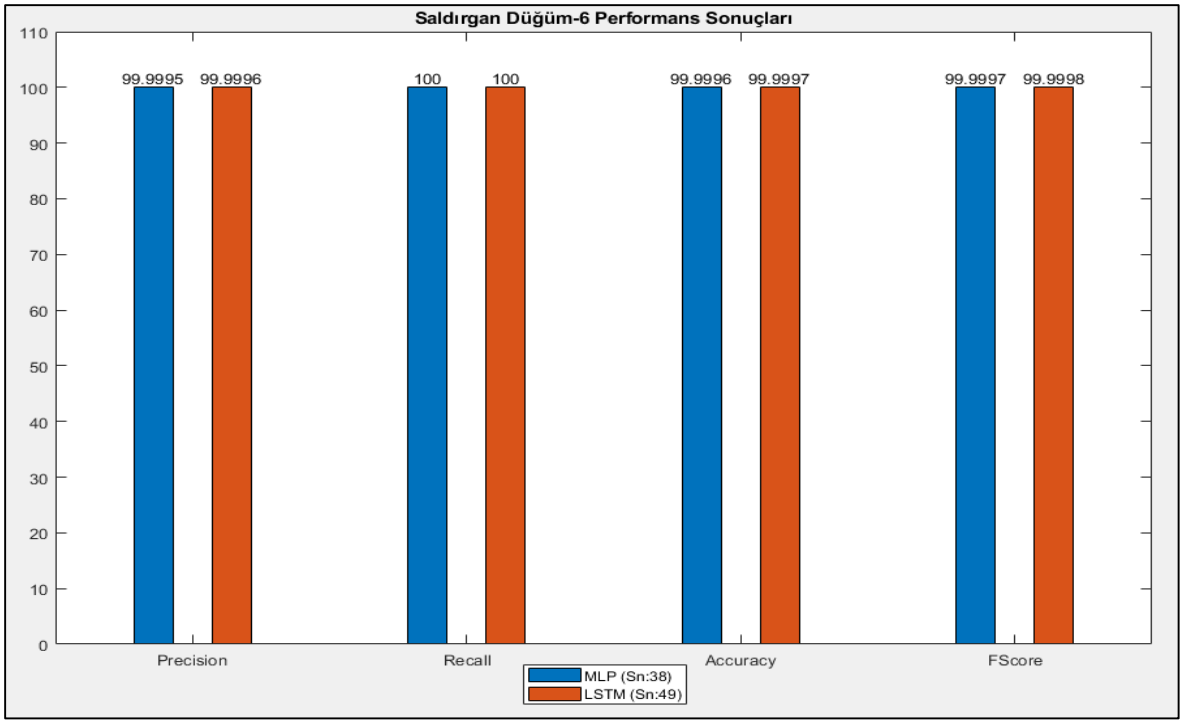
Sonuç olarak MLP yapay sinir ağı düşük hesaplama gücüne sahip sistemler için daha uygun olduğu görülmüş; ancak bu sistemler için yüksek sayıda eğitim verisi ihtiyacı olduğu anlaşılmıştır. LSTM yapay sinir ağı ise düşük veri setinde bile yüksek saldırı performansı sağladığı görülmüş; ancak eğitim için yüksek hesaplama gücü barındıran bilgisayar sistemlerinin gerekli olduğu ortaya çıkmıştır.



Şekil 6.1 Saldırgan Düğüm-2 İçin En iyi Performans Sonuçları



Şekil 6.2 Saldırgan Düğüm-4 İçin En iyi Performans Sonuçları



Şekil 6.3 Saldırgan Düğüm-6 İçin En iyi Performans Sonuçları

İleriki çalışmalarda sistemin güncel bir araç sisteminde denenmesi düşünülmüştür. Çok sayıda ECU'nun olduğu gerçek bir sistemde ECU'lardan kaynaklı oluşan ve beyaz Gauss gürültüsü olmayan gürültülerin sistemin hata performansı üzerindeki etkisi

araştırılmalıdır. Ayrıca, sisteme aynı anda birden fazla saldırgan düğüm bağlanarak test edilmesi denenmelidir. LSTM yapay sinir ağı için daha yüksek miktarda eğitim verisi kullanılarak eğitimler yapılmalı ve bu eğitimlerin yapılması için süper bilgisayarlar veya bilgisayar kümesi kullanılmalıdır. Eğitilen sistemler gömülü bir sisteme entegre edilerek araç kullanılırken test edilmeli ve performans sonuçları analiz edilmelidir.

KAYNAKLAR

[1] C. A. N. Specification, "Version 2.0," Robert Bosch GmbH, vol. 27, 1991.

[2] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham and others, "Experimental security analysis of a modern automobile," in 2010 IEEE symposium on security and privacy, 2010.

[3] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, T. Kohno and others, "Comprehensive experimental analyses of automotive attack surfaces.," in USENIX Security Symposium, 2011.

[4] C. Miller and C. Valasek, "Remote exploitation of an unaltered passenger vehicle," Black Hat USA, vol. 2015, 2015.

[5] C. Miller and C. Valasek, "A survey of remote automotive attack surfaces," black hat USA, vol. 2014, p. 94, 2014.

[6] R. A. Kemmerer and G. Vigna, "Intrusion detection: a brief history and overview," Computer, vol. 35, no. 4, pp. 127-130, 2002.

[7] T. Hoppe, S. Kiltz and J. Dittmann, "Applying intrusion detection to automotive it-early insights and remaining challenges," Journal of Information Assurance and Security (JIAS), vol. 4, p. 226–235, 2009.

[8] S.-F. Lokman, A. T. Othman and M.-H. Abu-Bakar, "Intrusion detection system for automotive Controller Area Network (CAN) bus system: a review," EURASIP Journal on Wireless Communications and Networking, vol. 2019, p. 1–17, 2019.

[9] M. Müter, A. Groll and F. C. Freiling, "A structured approach to anomaly detection for in-vehicle networks," in 2010 Sixth International Conference on Information Assurance and Security, 2010.

[10] I. Studnia, E. Alata, V. Nicomette, M. Kaâniche and Y. Laarouchi, "A language-based intrusion detection approach for automotive embedded networks," International Journal of Embedded Systems, vol. 10, p. 1–12, 2018.

[11] U. E. Larson, D. K. Nilsson and E. Jonsson, "An approach to specification-based attack detection for in-vehicle networks," in 2008 IEEE Intelligent Vehicles Symposium, 2008.

[12] H. M. Song, H. R. Kim and H. K. Kim, "Intrusion detection system based on the analysis of time intervals of CAN messages for in-vehicle network," in 2016 international conference on information networking (ICOIN), 2016.

[13] C. Miller and C. Valasek, "Adventures in automotive networks and control units," Def Con, vol. 21, p. 15–31, 2013.

[14] M. Gmiden, M. H. Gmiden and H. Trabelsi, "An intrusion detection method for securing in-vehicle CAN bus," in 2016 17th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA), 2016.

[15] M. R. Moore, R. A. Bridges, F. L. Combs, M. S. Starr and S. J. Powell, "Modeling inter-signal arrival times for accurate detection of," in 12th Annual Conference on Cyber and Information Security Research, 2017.

[16] S. Boumiza and R. Braham, "An Anomaly Detector for CAN Bus Networks in Autonomous Cars based on Neural Networks," in 2019 International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), 2019.

[17] A. Wasicek and A. Weimerskirch, "Recognizing manipulated electronic control units," 2015.

[18] M.-J. Kang and J.-W. Kang, "Intrusion detection system using deep neural network for in-vehicle network security," PloS one, vol. 11, p. e0155781, 2016.

[19] A. Taylor, N. Japkowicz and S. Leblanc, "Frequency-based anomaly detection for the automotive CAN bus," in World Congress on Industrial Control Systems Security (WCICSS), 2015.

[20] M. Weber, S. Klug, E. Sax and Z. Bastian, "Embedded Hybrid Anomaly Detection for Automotive," in 9th European Congress on Embedded Real Time Software and Systems (ERTS 2018), 2018.

[21] A. Taylor, S. Leblanc and N. Japkowicz, "Anomaly detection in automobile control network data with long short-term memory networks," in IEEE International Conference on Data Science and Advanced Analytics (DSAA), 2016.

[22] C. Wang, Z. Zhao, L. Gong, L. Zhu, Z. Liu and X. Cheng, "A distributed anomaly detection system for in-vehicle network using HTM," IEEE, vol. 6, pp. 9091-9098, 2018.

[23] A. Tomlinson, J. Bryans, S. A. Shaikh and H. K. Kalutarage, "Detection of automotive CAN cyber-attacks by identifying packet timing anomalies in time windows," in 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W), 2018.

[24] H. Lee, S. H. Jeong and H. K. Kim, "OTIDS: A novel intrusion detection system for in-vehicle network by using remote frame," in 15th Annual Conference on Privacy, Security and Trust (PST), 57-5709.

[25] W. Choi, H. J. Jo, S. Woo, J. Y. Chun, J. Park and D. H. Lee, "Identifying ecus using inimitable characteristics of signals in controller area networks," IEEE Transactions on Vehicular Technology, vol. 67, no. 6, pp. 4757-4770, 2018.

[26] Y. Yang, Z. Duan and M. Tehranipoor, "Identify a Spoofing Attack on an In-Vehicle CAN Bus Based on the Deep Features of an ECU Fingerprint Signal," Smart Cities, vol. 3, no. 1, pp. 17-30, 2020.

[27] S. Corrigan, "Introduction to the Controller Area Network (CAN)," Texas Instruments, 2016.

[28] P. Carsten, T. R. Andel, M. Yampolskiy and J. T. McDonald, "In-vehicle networks: Attacks, vulnerabilities, and proposed solutions," in Proceedings of the 10th Annual Cyber and Information Security Research Conference, 2015.

[29] S. Hartzell and C. Stubel, "Automobile CAN bus network security and vulnerabilities," Seattle, Washington, 2017.

[30] H. A. Boyes and A. E. A. Luck, "A security-minded approach to vehicle automation, road infrastructure technology, and connectivity," 2015.

[31] T. Hoppe, S. Kiltz and J. Dittmann, "Security threats to automotive CAN networks– practical examples and selected short-term countermeasures," in International Conference on Computer Safety, Reliability, and Security, 2008.

[32] "MATLAB," MathWorks, [Online]. Available: mathworks.com/products/matlab.html. [Accessed 15 03 2022].

[33] Xilinx, "zynq®-7000q defense-grade all programmable socs," [Online]. Available: xilinx.com/content/dam/xilinx/support/documentation/selection-guides/zynq-7000q-product-table.pdf. [Accessed 15 11 2021].

[34] Future Electronics, "future electronics - avalanche development board," [Online]. Available: futureelectronics.com/avalanche. [Accessed 11 11 2021].

[35] Microsemi, "PolarFire fpgas," [Online]. Available: microsemi.com/product-directory/fpgas/3854-polarfire-fpgas#product-table. [Accessed 17 11 2021].

[36] Microchip, "MCP25625 - CAN Controller with Integrated Transceiver," 2019. [Online]. Available: microchip.com/downloads/en/DeviceDoc/MCP25625-CAN-Controller-Data-Sheet-20005282C.pdf. [Accessed 1 12 2021].

[37] Microchip, "MCP2551 - High-Speed CAN Transciever," 2016. [Online]. Available: microchip.com/downloads/en/devicedoc/20001667g.pdf.

[38] robitshop.com, "USB-can Dönüştürücü v7.10," [Online]. Available: robitshop.com/urun/usb-can-donusturucu-v7-10. [Accessed 3 12 2021].

[39] "CP2102 Usb Uart Dönüştürücü/Programlayıcı," [Online]. Available: robotistan.com/cp2102-usb-uart-donusturucuprogramlayici. [Accessed 15 03 2022].

[40] Texas Instruments, "SN65HVD23x 3.3-V CAN Bus Transceivers," [Online]. Available: ti.com/lit/gpn/sn65hvd230.

[41] "Vivado," Xilinx, [Online]. Available: xilinx.com/products/design-tools/vivado/vivado-ml.html. [Accessed 11 09 2021].

[42] Texas Instruments, "ADC128S102 Eight-channel, 50-kSPS to 1-MSPS, 12-bit analog-to-digital converter (ADC)," 2015. [Online]. Available: ti.com/lit/ds/symlink/adc128s102.pdf. [Accessed 29 12 2021].

[43] "Pyhon," Python, [Online]. Available: python.org/. [Accessed 10 03 2022].

[44] K. Tindell, "The canframe.py tool," [Online]. Available: kentindell.github.io/2020/01/03/canframe_py_tool/. [Accessed 17 01 2022].

[45] "pyserial," [Online]. Available: pypi.org/project/pyserial/. [Accessed 10 03 2022].

[46] "matplotlib," [Online]. Available: matplotlib.org/. [Accessed 07 03 2022].

[47] "nnstart," MathWorks, [Online]. Available: mathworks.com/help/deeplearning/ref/nnstart.html. [Accessed 16 03 2022].

[48] "Deep Learning Toolbox," MathWorks, [Online]. Available: mathworks.com/help/deeplearning/. [Accessed 12 03 2022].

[49] "CUDA toolkit," NVIDIA, [Online]. Available: developer.nvidia.com/cuda-toolkit. [Accessed 12 03 2022].

[50] Texas Instruments, "What do can bus signals look like?," [Online]. Available: e2e.ti.com/blogs_/b/industrial_strength/posts/what-do-can-bus-signals-look-like. [Accessed 9 11 2021].

EKLER

EK 1: MLP Saldırı Tespit Performans Sonuçları

Eğitim Veri Sayısı	Test Veri Sayısı	Maksimum Eğitim Hatası	Hata Eşiği	Saldırgan Düğüm	Saldırgan Düğüm Hata Oranı (Yanlış Negatif)	Saldırgan Olmayan Düğüm Hata Oranı (Yanlış Pozitif)	Gizli Katman Nöron Sayısı	CAN Veri Hızı	Senaryo
10000	40000	0.036	0.0399	2	0.0079	0.000005	5	5 Kbps	1
15000	35000	0.032071	0.035278	2	1	0.000000	5	5 Kbps	2
15000	35000	0.032072	0.032392	2	1	0.000000	5	5 Kbps	2
15000	35000	0.003645	0.004009	2	0	0.000017	10	5 Kbps	3
20000	30000	0.022739	0.025013	2	0	0.000020	10	5 Kbps	4
25000	25000	0.010852	0.011937	2	0	0.000008	10	5 Kbps	5
30000	20000	0.021037	0.023141	2	0	0.000010	10	5 Kbps	6
10000	40000	0.018560	0.020416	4	1	0.000000	5	5 Kbps	7
10000	40000	0.018560	0.019488	4	0.999980	0.000010	5	5 Kbps	7
10000	40000	0.018560	0.018746	4	0.999980	0.000010	5	5 Kbps	7
10000	40000	0.046855	0.051540	4	0	0.000010	10	5 Kbps	8
10000	40000	0.046855	0.049197	4	0	0.000010	10	5 Kbps	8
10000	40000	0.046855	0.047323	4	0	0.000010	10	5 Kbps	8
15000	35000	0.008224	0.009046	4	0	0.000010	10	5 Kbps	9
20000	30000	0.014216	0.015638	4	0	0.000013	10	5 Kbps	10
25000	25000	0.006954	0.007649	4	0.000040	0.000008	10	5 Kbps	11
30000	20000	0.022179	0.024397	4	0.000620	0.000010	10	5 Kbps	12
10000	40000	0.020765	0.022841	6	0	0.000015	5	5 Kbps	13
15000	35000	0.012822	0.014105	6	1	0.000023	5	5 Kbps	14
15000	35000	0.012822	0.013464	6	1	0.000023	5	5 Kbps	14

Eğitim Veri Sayısı	Test Veri Sayısı	Maksimum Eğitim Hatası	Hata Eşiği	Saldırgan Düğüm	Saldırgan Düğüm Hata Oranı (Yanlış Negatif)	Saldırgan Olmayan Düğüm Hata Oranı (Yanlış Pozitif)	Gizli Katman Nöron Sayısı	CAN Veri Hızı	Senaryo
15000	35000	0.012822	0.012951	6	1	0.000023	5	5 Kbps	14
15000	35000	0.007063	0.007769	6	0	0.000017	10	5 Kbps	15
20000	30000	0.010067	0.011073	6	0	0.000027	10	5 Kbps	16
25000	25000	0.023189	0.025508	6	0	0.000032	10	5 Kbps	17
30000	20000	0.018611	0.020472	6	0	0.000020	10	5 Kbps	18
10000	40000	0.000019	0.000020	2	0	0.000010	5	50 Kbps	19
15000	35000	0.027073	0.029780	2	0	0.000011	5	50 Kbps	20
20000	30000	0.030594	0.033653	2	0	0.000007	5	50 Kbps	21
25000	25000	0.058152	0.063967	2	1	1	5	50 Kbps	22
25000	25000	0.058152	0.058734	2	1	0.000008	5	50 Kbps	22
25000	25000	0.017065	0.018771	2	0.658200	0.000040	10	50 Kbps	23
25000	25000	0.017065	0.017235	2	0.472940	0.000040	10	50 Kbps	23
25000	25000	0.010835	0.011918	2	0	0.000040	15	50 Kbps	24
30000	20000	0.051230	0.056353	2	0.738400	0.000030	15	50 Kbps	25
30000	20000	0.051230	0.051742	2	0.694320	0.000030	15	50 Kbps	25
30000	20000	0.032908	0.036199	2	0	0.000050	15	50 Kbps	26
10000	40000	0.022305	0.024536	4	1	0.000035	5	50 Kbps	27
10000	40000	0.022305	0.022528	4	1	0.000045	5	50 Kbps	27
10000	40000	0.010430	0.011473	4	0.858040	0.000015	10	50 Kbps	28
10000	40000	0.010430	0.010534	4	0.234520	0.000015	10	50 Kbps	28
10000	40000	0.010430	0.010440	4	0.187420	0.000015	10	50 Kbps	28
10000	40000	0.000391	0.000430	4	0	0.000085	15	50 Kbps	29
15000	35000	0.010469	0.010574	4	0	0.000023	15	50 Kbps	30
20000	30000	0.008547	0.009402	4	0	0.000020	15	50 Kbps	31

Eğitim Veri Sayısı	Test Veri Sayısı	Maksimum Eğitim Hatası	Hata Eşiği	Saldırgan Düğüm	Saldırgan Düğüm Hata Oranı (Yanlış Negatif)	Saldırgan Olmayan Düğüm Hata Oranı (Yanlış Pozitif)	Gizli Katman Nöron Sayısı	CAN Veri Hızı	Senaryo
25000	25000	0.012944	0.014238	4	0.999980	0.000032	15	50 Kbps	32
25000	25000	0.012944	0.013073	4	0.999880	0.000040	15	50 Kbps	32
25000	25000	0.010286	0.010389	4	0.676360	0.000072	15	50 Kbps	33
25000	25000	0.022833	0.023061	4	0.965360	0.000016	20	50 Kbps	34
25000	25000	0.003219	0.003541	4	0.002760	0.000072	25	50 Kbps	35
25000	25000	0.003219	0.003251	4	0.002220	0.000080	25	50 Kbps	35
30000	20000	0.020154	0.022170	4	0.965600	0.000030	25	50 Kbps	36
30000	20000	0.020154	0.020356	4	0.934360	0.000040	25	50 Kbps	36
30000	20000	0.056007	0.061608	4	0.000060	0.000010	30	50 Kbps	37
30000	20000	0.056007	0.056568	4	0	0.000020	30	50 Kbps	37
10000	40000	0.019146	0.021061	6	0	0.000005	5	50 Kbps	38
15000	35000	0.012503	0.013753	6	1	0.000017	5	50 Kbps	39
15000	35000	0.012503	0.012628	6	1	0.000017	5	50 Kbps	39
15000	35000	0.042573	0.046830	6	1	0.000023	10	50 Kbps	40
15000	35000	0.042573	0.042998	6	1	0.000023	10	50 Kbps	40
15000	35000	0.000041	0.000045	6	0	0.000011	15	50 Kbps	41
20000	30000	0.030702	0.033772	6	0	0.000013	15	50 Kbps	42
25000	25000	0.018393	0.020233	6	0	0.000016	15	50 Kbps	43
30000	20000	0.000018	0.000020	6	0.973880	0.000020	15	50 Kbps	44
30000	20000	0.000018	0.000018	6	0.966780	0.000020	15	50 Kbps	44
30000	20000	0.016679	0.016845	6	0	0.000020	15	50 Kbps	45
10000	40000	0.024270	0.026697	2	0	0.000010	5	100 Kbps	46
15000	35000	0.018499	0.020349	2	1	0.000006	5	100 Kbps	47
15000	35000	0.018499	0.018684	2	1	0.000006	5	100 Kbps	47
15000	35000	0.009050	0.009955	2	0.878480	0.000017	10	100 Kbps	48

Eğitim Veri Sayısı	Test Veri Sayısı	Maksimum Eğitim Hatası	Hata Eşiği	Saldırgan Düğüm	Saldırgan Düğüm Hata Oranı (Yanlış Negatif)	Saldırgan Olmayan Düğüm Hata Oranı (Yanlış Pozitif)	Gizli Katman Nöron Sayısı	CAN Veri Hızı	Senaryo
15000	35000	0.009050	0.009141	2	0.771060	0.000017	10	100 Kbps	48
15000	35000	0.013390	0.014729	2	1	0.000023	15	100 Kbps	49
15000	35000	0.011713	0.012884	2	0.667560	0.000017	20	100 Kbps	50
15000	35000	0.011713	0.011830	2	0.666880	0.000017	20	100 Kbps	50
15000	35000	0.000038	0.000039	2	0	0.000023	25	100 Kbps	51
20000	30000	0.000031	0.000034	2	0	0.000013	25	100 Kbps	52
25000	25000	0.012469	0.013716	2	0	0.000024	25	100 Kbps	53
30000	20000	0.009094	0.010003	2	0	0.000030	25	100 Kbps	54
10000	40000	0.045057	0.049563	4	1	0.000005	5	100 Kbps	55
10000	40000	0.011358	0.012493	4	0	0.000010	10	100 Kbps	56
15000	35000	0.014688	0.016156	4	0	0.000011	10	100 Kbps	57
20000	30000	0.025289	0.027818	4	0	0.000013	10	100 Kbps	58
25000	250000	0.011277	0.012405	4	0	0.000008	10	100 Kbps	59
30000	20000	0.000144	0.000159	4	0.000380	0.000030	10	100 Kbps	60
30000	20000	0.088974	0.097872	4	0.000020	0.000010	10	100 Kbps	61
10000	40000	0.023650	0.026015	6	0.000020	0.000005	5	100 Kbps	62
15000	35000	0.009542	0.010497	6	0	0.000006	5	100 Kbps	63
20000	30000	0.028786	0.031665	6	1	0.000020	5	100 Kbps	64
20000	30000	0.028786	0.029074	6	1	0.000027	5	100 Kbps	64
20000	30000	0.001316	0.001447	6	0	0.000007	10	100 Kbps	65
25000	25000	0.000106	0.000116	6	1	0.000008	10	100 Kbps	66
25000	25000	0.000015	0.000017	6	0	0.000024	15	100 Kbps	67
30000	20000	0.024053	0.026459	6	0	0.000020	15	100 Kbps	68

EK 2: LSTM Saldırı Tespit Performans Sonuçları

Eğitim Veri Sayısı	Test Veri Sayısı	Maksimum Eğitim Hatası	Hata Eşiği	Saldırgan Düğüm	Saldırgan Düğüm Hata Oranı (Yanlış Negatif)	Saldırgan Olmayan Düğüm Hata Oranı (Yanlış Pozitif)	Gizli Katman Nöron Sayısı	CAN Veri Hızı	Senaryo
100	49900	0.047757	0.057279	2	0.073420	0.002689	10	5 Kbps	1
500	49500	0.019261	0.029068	2	0.107120	0.000162	25	5 Kbps	2
1000	49000	0.030149	0.039848	2	0.000020	0.000269	100	5 Kbps	3
2000	48000	0.702213	0.705191	2	1	0.000950	250	5 Kbps	4
100	49900	0.000222	0.010220	4	0.999980	0	100	5 Kbps	5
100	49900	0.000735	0.010727	4	0.260220	0	100	5 Kbps	6
100	49900	0.000576	0.010570	4	0.999920	0.000004	100	5 Kbps	7
100	49900	0.008234	0.018152	4	0.999980	0	100	5 Kbps	8
100	49900	0.003256	0.013223	4	0.000660	0	100	5 Kbps	9
500	49500	0.000416	0.010412	4	1	0	100	5 Kbps	10
500	49500	0.000602	0.010596	4	1	0	100	5 Kbps	11
500	49500	0.001723	0.011706	4	0.999980	0	100	5 Kbps	12
500	49500	0.061823	0.071205	4	0.999980	0	100	5 Kbps	13
1000	49000	0.000706	0.010699	4	0.999980	0.000029	100	5 Kbps	14
1000	49000	0.001697	0.011680	4	0.999980	0.000004	100	5 Kbps	15
100	49900	0.000432	0.010428	6	1	0.000008	100	5 Kbps	16
100	49900	0.000522	0.010517	6	0	0.000012	100	5 Kbps	17
500	49500	0.004443	0.014399	6	0.993680	0.000004	100	5 Kbps	18
1000	49000	0.032426	0.042102	6	1	0.000082	100	5 Kbps	19
500	49500	0.002176	0.012154	6	0.017160	0.000012	100	5 Kbps	20
1000	49000	0.012541	0.022416	6	1	0.000151	100	5 Kbps	21
1000	49000	0.125142	0.133890	6	1	0.000098	100	5 Kbps	23

Eğitim Veri Sayısı	Test Veri Sayısı	Maksimum Eğitim Hatası	Hata Eşiği	Saldırgan Düğüm	Saldırgan Düğüm Hata Oranı (Yanlış Negatif)	Saldırgan Olmayan Düğüm Hata Oranı (Yanlış Pozitif)	Gizli Katman Nöron Sayısı	CAN Veri Hızı	Senaryo
100	49900	0.000834	0.010825	2	0.334080	0.000024	100	50 Kbps	25
100	49900	0.000763	0.010755	2	0.057180	0.000036	100	50 Kbps	26
500	49500	0.000294	0.010291	2	0.044420	0.000053	100	50 Kbps	27
500	49500	0.000291	0.010288	2	0.598600	0.000061	100	50 Kbps	28
500	49500	0.000606	0.010600	2	0.001420	0	100	50 Kbps	29
1000	49000	0.017496	0.027321	2	0.179340	0.000008	100	50 Kbps	30
1000	49000	0.000580	0.010574	2	0.011180	0	100	50 Kbps	31
100	49900	0.075948	0.085188	4	0.999920	0.001146	100	50 Kbps	32
100	49900	0.003356	0.013322	4	0.000380	0.000072	100	50 Kbps	33
100	49900	0.006923	0.016853	4	1	0.000132	100	50 Kbps	34
100	49900	0.001466	0.011451	4	0.998640	0.000024	100	50 Kbps	35
100	49900	0.004946	0.014896	4	0.647800	0.000986	100	50 Kbps	36
500	49500	0.000830	0.010822	4	0.127500	0.000178	100	50 Kbps	37
500	49500	0.011716	0.021599	4	0.999980	0.000614	100	50 Kbps	38
500	49500	0.121565	0.130349	4	0.999920	0.000154	100	50 Kbps	39
1000	49000	0.072351	0.081628	4	1	0.000322	100	50 Kbps	40
1000	49000	0.043897	0.053458	4	1	0.000318	100	50 Kbps	41
1000	49000	0.007818	0.017740	4	1	0.000147	100	50 Kbps	42
1000	49000	0.014343	0.024199	4	0.727940	0	100	50 Kbps	43
1000	49000	0.991484	0.991569	4	1	0.003114	200	50 Kbps	44
1000	49000	0.000721	0.010713	4	0.999980	0	250	50 Kbps	45
100	49900	0.005593	0.015537	6	0.000020	0.000064	100	50 Kbps	46
500	49500	0.001172	0.011160	6	0.266860	0.000513	100	50 Kbps	47
1000	49000	0.385230	0.391378	6	0.000060	0.000518	100	50 Kbps	48

Eğitim Veri Sayısı	Test Veri Sayısı	Maksimum Eğitim Hatası	Hata Eşiği	Saldırgan Düğüm	Saldırgan Düğüm Hata Oranı (Yanlış Negatif)	Saldırgan Olmayan Düğüm Hata Oranı (Yanlış Pozitif)	Gizli Katman Nöron Sayısı	CAN Veri Hızı	Senaryo
1000	49000	0.006032	0.015972	6	0	0.000004	250	50 Kbps	49
100	49900	0.001328	0.011314	2	0	0.000004	100	100 Kbps	50
500	49500	0.001632	0.011615	2	0.003440	0.000081	100	100 Kbps	51
1000	49000	0.002168	0.012146	2	0.999900	0.000016	100	100 Kbps	52
1000	49000	0.010648	0.020542	2	1	0.000069	100	100 Kbps	53
1000	49000	0.001251	0.011239	2	0.999960	0.000024	100	100 Kbps	54
1000	49000	0.002996	0.012966	2	1	0.000012	100	100 Kbps	55
100	49900	0.010766	0.020658	4	0	0.000261	100	100 Kbps	56
100	49900	0.002015	0.011995	4	0.000040	0.000637	100	100 Kbps	57
500	49500	0.005202	0.015150	4	0	0.000085	100	100 Kbps	59
1000	49000	0.985763	0.985906	4	1	0.000759	150	100 Kbps	60
1000	49000	0.002051	0.012030	4	0	0.000004	150	100 Kbps	61
100	49900	0.006161	0.016099	6	0	0.001074	100	100 Kbps	62
500	49500	0.016194	0.026032	6	0	0.000158	100	100 Kbps	63
1000	49000	0.021172	0.030961	6	0	0.000082	100	100 Kbps	64