YILDIZ TEKNİK ÜNİVERSİTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

VERİ İLETİŞİMİ VE BİLGİSAYAR AĞLARI PROJESİ

Şeyma BAŞARAN-Ç19052100
İclal ERTÜRK-21011037
Merve ÇAKIR-23011603
Ayşe Sena ERKMEN-22011649

İÇİNDEKİLER

1.Giriş

Bu proje, OSI modelinin 2. katmanı olan Veri Bağlantı Katmanı (Data Link Layer)'nın temel işlevlerini simüle eden bir program geliştirmeyi amaçlamaktadır. Geliştirilen sistem, bir gönderici (sender) ile bir alıcı (receiver) arasında gerçekleşen veri iletim sürecinde; veri çerçevelerinin oluşturulması, hata kontrolü (CRC ve checksum), akış kontrolü (Stop-and-Wait) ve iletim sırasında karşılaşılabilecek kayıp/bozulma gibi problemleri simüle etmektedir. Tüm bu süreç, oluşturulan grafiksel kullanıcı arayüzü (GUI) üzerinden adım adım görselleştirilmiştir.

1.2 Görev Dağılımı

- Şeyma BAŞARAN – (GUI, Fonksiyonların kodlanması)
- İclal ERTÜRK – (GUI, Fonksiyonların kodlanması)
- Merve ÇAKIR – (Fonksiyonların kodlanması, Raporlama)
- Ayşe Sena ERKMEN – ((Fonksiyonların kodlanması, Raporlama)

2. Kullanılan Yöntem ve Teknolojiler

2.1 CRC

CRC (Cyclic Redundancy Check), veri iletimi sırasında hata kontrolü yapmak için yaygın olarak kullanılan bir yöntemdir. CRC, gönderilen verinin hatasız bir şekilde alıcıya ulaşıp ulaşmadığını kontrol etmek amacıyla kullanılır. Gönderen taraf, veriye bir CRC kodu ekler ve alıcı taraf bu kodu kullanarak verinin doğruluğunu kontrol eder.

Çalışma prensibi olarak CRC ile verinin polinomla işlenmesi ve XOR işlemiyle CRC kodunun hesaplanması sağlanır. Bu işlem seçilmiş bir polinom ve bu polinomun derece sayısı kadar sıfırın verinin sonuna basamak olarak eklenmesiyle uygulanır. Gönderen taraf, veriyi belirli bir polinomla böler ve kalan değeri CRC kodu olarak ekler. Alıcı taraf aynı işlemi yaparak CRC kodunu kontrol eder. Eğer hesaplanan CRC değeri alıcı tarafın gönderdiği CRC koduyla eşleşiyorsa, veri doğru bir şekilde iletilmiş demektir. Eğer eşleşme yoksa, veri hatalı kabul edilir.

```
19      std::string generator = "10001000000100001";
20      for (int i = 0; i < frames.size(); ++i) {
21          std::string data = frames[i] + std::string(16, '0');
22          for (int step = 0; step <= data.size() - generator.size(); ++step) {
23              if (data[step] == '1') {
24                  for (int j = 0; j < generator.size(); ++j) {
25                      data[step + j] = (data[step + j] == generator[j]) ? '0' : '1';
26                  }
27              }
28          }
```

2.2 Checksum

Checksum, verinin doğruluğunu kontrol etmek için yaygın bir hata kontrolü yöntemidir. Bu yöntem, veri iletimi sırasında veri paketlerine eklenen ve verinin doğru iletilip iletilmediğini kontrol eden bir değerdir.

Gönderici taraf, her frame'in sonunda yer alan 16 bitlik CRC kodunu iki adet 8 bitlik parçaya ayırarak toplar. Tüm frame'lere ait bu 8 bitlik parçalar sırayla toplanarak toplam bir değer elde edilir. Ardından bu toplam değerine bir eklenerek checksum değeri elde edilir. Hesaplanan checksum değeri, veri ile birlikte alıcıya gönderilir. Bu checksum değeri, iletilen verinin doğruluğunu kontrol etmek için kullanılır. Alıcı taraf, kendisine gelen verinin checksum değerini kontrol eder. Eğer alıcı tarafın hesapladığı checksum değeri, gönderilen checksum ile eşleşirse,

```
    std::bitset<8> checksumBits;
    for (const std::string& crcStr : crcList) {
        std::bitset<16> b(crcStr);
        std::bitset<8> highByte(b.to_ulong() >> 8);
        std::bitset<8> lowByte(b.to_ulong() & 0xFF);
        checksumBits = std::bitset<8>((checksumBits.to_ulong() + highByte.to_ulong() + lowByte.to_ulong()) & 0xFF);
    }
```

veri doğru kabul edilir. Eşleşme olmazsa, veri hatalı kabul edilir.

Bu projede, yolda bozulmuş CRC değerlerini kontrol etmek amacıyla Checksum kullanılmıştır. Veri iletimi sırasında CRC kodu ile veriler alıcıya gönderilirken, bu verilerin yolda bozulup bozulmadığı kontrol edilmiştir. Gönderilen CRC kodunun doğruluğunu kontrol etmek için Checksum kullanılarak, veri iletimindeki olası bozulmalar tespit edilmiştir.

2.3 Rastgelelik

Proje kapsamında, veri iletimi sırasında oluşabilecek hata senaryoları belirli olasılıklarla ve rastgele şekilde uygulanmaktadır. Bu sayede, veri kaybı, bozulma, ACK kaybı ve checksum hatası gibi durumlar simüle edilerek gerçek dünyadaki belirsizlikler modellenmiştir.

Rastgelelik, C++'ta Qt'nin QRandomGenerator sınıfı kullanılarak sağlanmış; her çalıştırmada farklı sonuçlar elde edilmesi mümkün kılınmıştır. Bu yöntem, uygulamanın farklı senaryolara karşı esnek ve sağlam bir algoritmaya sahip olmasını gerektirmiştir.

```
void FrameEkrani::kontrolEt() {
    ackTimer->stop();
    int rastgele = QRandomGenerator::global()->bounded(100);

    if (rastgele < 10) {
        durumEtiketi->setText("❌ Frame yolda kayboldu! Yeniden gönderiliyor...");
        QTimer::singleShot(3000, this, &FrameEkrani::gonderFrame);
        return;
    } else if (rastgele < 30) {
        durumEtiketi->setText("⚠️ Frame bozuldu!");
        dataLabel->setText("📦 ! ");
        QTimer::singleShot(3000, this, &FrameEkrani::gonderFrame);
        return;
    } else if (rastgele < 45) {
        durumEtiketi->setText("🔁 ACK kayıp, gönderici tekrar bekliyor...");
        QTimer::singleShot(3000, this, &FrameEkrani::gonderFrame);
        return;
    }
```

2.4 Stop and Wait

Stop & Wait, veri iletiminde kullanılan basit bir akış kontrolü protokolüdür. Bu protokolde, gönderen taraf her veri çerçevesini gönderdikten sonra, alıcıdan bir onay (ACK) alana kadar bir sonraki veriyi göndermez. Bu, her bir veri çerçevesi için bir geri besleme (feedback) mekanizması sağlar.

Stop & Wait protokolü, iletim sırasında oluşabilecek frame kayıpları ve bozulmalarını yönetmek için timeout mekanizması kullanır. Gönderen taraf, her çerçeve için bir ACK mesajı aldıktan sonra bir sonraki çerçeveyi gönderir. Eğer ACK mesajı zamanında alınmazsa, timeout beklenir ve gönderim tekrar edilir. Bu sayede, frame kaybolma veya bozulma durumlarında veri iletimi yeniden başlatılır.

2.5 Data Transparency

Veri iletimi sırasında, gönderici tarafın veriyi gönderirken özel karakterler (frame başlangıcı, frame bitişi veya escape karakterleri gibi) kullanması, alıcı tarafın bu karakterleri yanlış anlamasına ve verinin hatalı bir şekilde işlenmesine yol açabilir. Özellikle, verinin içinde 0x7E (frame başlangıcı), 0x7D (escape karakteri) gibi özel baytlar bulunduğunda, alıcı bu baytları çerçevenin sonu veya özel bir kontrol karakteri olarak algılayabilir. Bu durumu engellemek için gönderici, bu özel karakterleri veri içinde kaçış karakterleri ile maskeler.

Bu işlem, byte stuffing adı verilen bir tekniktir. Gönderici, verinin içinde bu özel baytlar bulunduğunda, önce 0x7D karakterini ekler ve ardından XOR 0x20 işlemi ile karakteri maskeler. Bu şekilde, alıcı bu karakterleri doğru şekilde anlamak için veriyi çözmeden önce byte unstuffing işlemi yapar. Alıcı, 0x7D gördüğünde, hemen bir sonraki baytı XOR 0x20 ile eski haline getirir ve böylece orijinal veriyi geri alır.

Bu iki işlem arasındaki fark, göndericinin veri içindeki özel karakterleri gizlemesi (escape ekleyerek ve XOR yaparak) ve alıcının bu gizlemeyi geri çözmesidir. Byte stuffing ve byte unstuffing süreçleri, özel karakterlerin yanlışlıkla veri içeriği olarak kabul edilmesini engelleyerek veri bütünlüğünün korunmasını sağlar. Bu yöntem, göndericinin ve alıcının veri iletimindeki güvenliğini artırır ve doğru iletişim sağlanır.

```
128  // Byte Stuffing
129  std::string FrameEkrani::applyByteStuffing(const std::string& rawData) {
130      std::string stuffed;
131      for (unsigned char ch : rawData) {
132          if (ch == 0x7E || ch == 0x7D) {
133              stuffed += 0x7D;
134              stuffed += ch ^ 0x20;
135          } else {
136              stuffed += ch;
137          }
138      }
139      return stuffed;
140  }
141
142  // Byte Unstuffing
143  std::string FrameEkrani::removeByteStuffing(const std::string& stuffedData) {
144      std::string unstuffed;
145      for (size_t i = 0; i < stuffedData.size(); ++i) {
146          if (stuffedData[i] == 0x7D && i + 1 < stuffedData.size()) {
147              unstuffed += stuffedData[i + 1] ^ 0x20;
148              i++;
149          } else {
150              unstuffed += stuffedData[i];
151          }
152      }
153      return unstuffed;
```

2.6 Qt-GUI

Projenin kullanıcı arayüzü, C++ ile uyumlu olan Qt kütüphanesi kullanılarak geliştirilmiştir. Qt, sinyal-slot yapısı sayesinde olay tabanlı programlama için güçlü bir altyapı sunmaktadır. Bu projede, frame gönderimi, alıcıdan gelen ACK'ler, CRC kontrolü ve checksum işlemleri, görselleştirilmiş öğeler (etiketler, animasyonlar, butonlar) aracılığıyla sunulmuştur. Bu sayede kullanıcılar, veri iletim sürecini adım adım gözlemleyebilmekte ve olası hata durumlarını (frame kaybı, bozulma, ACK kaybı) sezgisel olarak deneyimleyebilmektedir.

2.7 GUI'nin Çalıştırılması

Oluşturulan GUI' yi açmak için ödev dosyasında bulunan 'viba-20242-6-uygulama' isimli klasörü açıp içerisindeki uygulama.exe dosyasını çalıştırmalısınız. (.exe uzantılı dosya .txt uzantısına ödev dosyasında denildiği gibi elle dönüştürülmüştür) Sadece .exe uzantılı dosya gönderildiği zaman uygulama çalışmamaktadır bunun için dll dosyaları ile beraber gönderilmiştir.

2.8 C++

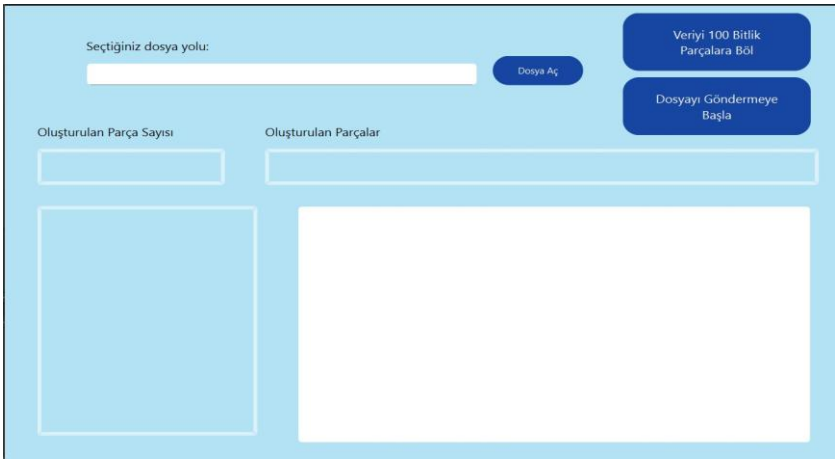
Projenin arka plan mantığı ve simülasyon işlemleri C++ programlama dili ile gerçekleştirilmiştir.

3.Uygulamanın İşleyişi

3.1 Dosya Yükleme, Frame Yapısı Oluşturma

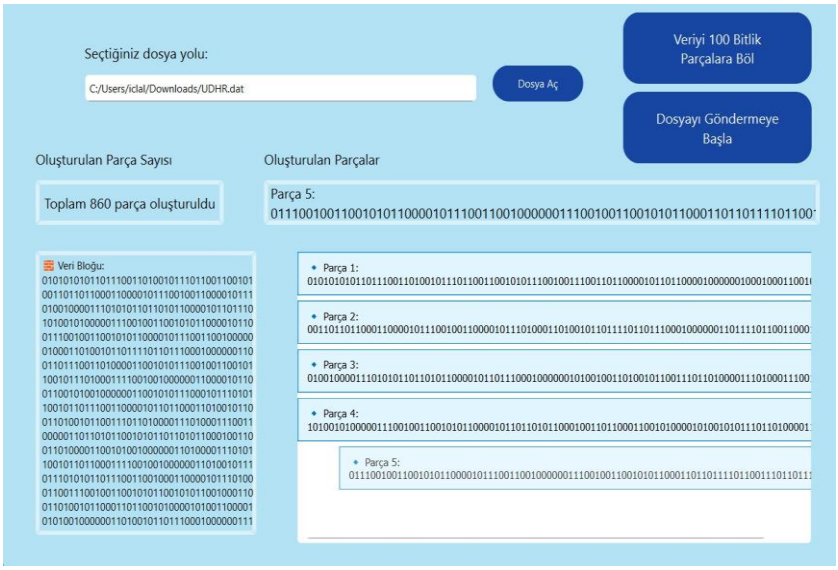
Uygulamanın başlangıç noktasıdır. "Başla" butonu, kullanıcının uygulamanın diğer adımlarına geçmesini sağlar. Başlaya basıldıktan sonra dosya yükleme ve frame bölme işlemlerinin yapıldığı sayfaya geçiş yapılır.

(Giriş Sayfası)



(Dosya Yükleme)

"Dosya Aç" butonu ile yüklenmek istenen dosya seçildikten sonra dosya binary formata çevrilerek uygulamaya yüklenir. Ardından "Veriyi Böl" butonu ile veri yüz bitlik alt parçalara ayrılır. Burada ayrılan veri parçaları ve parçaya ayrıldığı gibi bilgilerin takibi yapılır.



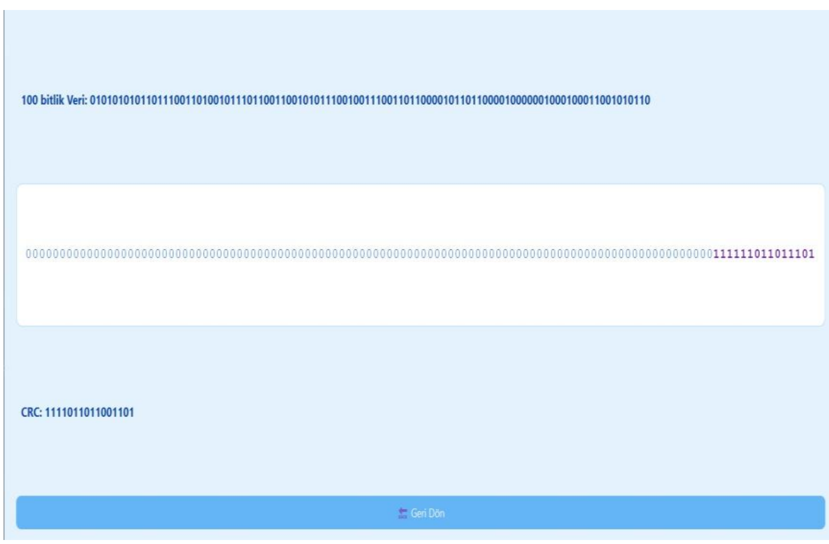
(Veri Bölme)

Bu sayfada, kullanıcının seçtiği dosyadan veriyi alarak 100 bitlik parçalara böldüğü bir arayüzü göstermektedir. "Veri Bloğu" kısmında, dosyanın içeriği bit bazında sıralanırken, "Oluşturulan Parçalar" bölümünde her 100 bitlik parça gösterilmektedir. Kullanıcı, toplamda kaç parça oluşturulduğunu ve her bir parçanın içeriğini bu bölümde görebilir. Bu işlem, verinin doğru bir şekilde bölünmesini ve her bir çerçevenin gönderilmeye hazır hale gelmesini sağlar. "Dosyayı Göndermeye Başla" butonuna basıldıktan sonrasında ise bir menü sayfası gelmektedir.

Kullanıcının uygulama içerisinde hangi işlemi görmek istediğini seçtiği bir ara ekrandır. Sayfada yer alan dört butondan herhangi biri tıklanarak, ilgili işleme ait sayfaya yönlendirme yapılır.

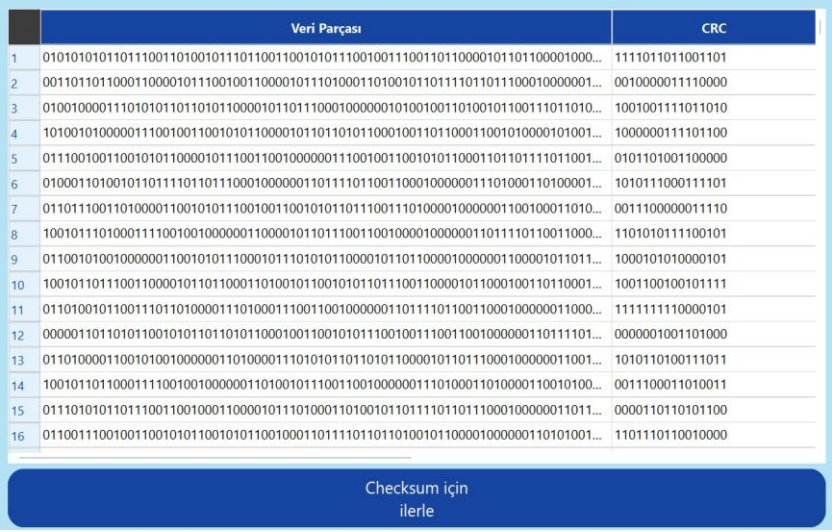
(Menü)

3.2 CRC Kodu Hesaplama



Menüden "CRC Oluşumunu Gör" butonuna tıklandığında bu sayfaya geçiş yapılır. Bu sayfada, CRC hesaplama işlemi animasyonla gösterilmektedir. Bu animasyon, verinin bit dizisi üzerinde yapılan CRC hesaplamasını adım adım gösterir. CRC hesaplama, her bir bitin XOR işlemiyle işlenmesi ve polinomla bölünmesi işlemini içerir. Bu görselde, CRC işleminin nasıl çalıştığı ve veri üzerinde nasıl işlendiği adım adım açıklanmaktadır. Animasyon sayesinde, CRC hesaplamasının her adımı görsel olarak takip edilebilir. Bu, kullanıcıların CRC hesaplama yöntemini daha iyi anlamalarına yardımcı olur.

(CRC Oluşumunu Gör)



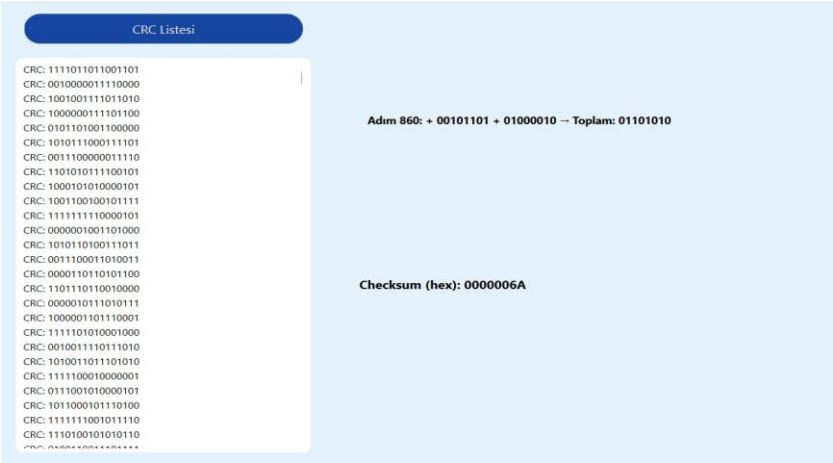
	Veri Parçası	CRC
1	0101010101101110011010010111011001010111001001110011011000010110110000010...	1111011011001101
2	0011011011000110000101110010011000010111010001101001011011110110111000010000001...	0010000011110000
3	0100100001110101011011010110000101101110001000000101001001101001011001111011010...	1001001111011010
4	1010010100000011100100110010101100001011011010110001001101100011001010000101001...	1000000111101100
5	0111001001100101011000010111001100100000011100100110010101100011011011111011001...	0101101001100000
6	0100011010010110111101110001000000110111101100110001000000111010001101000010...	1010111000111101
7	0110111001101000011001010111001001100101011011001110100001000000110010001101 0...	0011100000011110
8	1001011101000111100100100000011000010110111001100100001000000110111101100110000...	1101010111100101
9	0110010100100000011001010111000101110101011000010110110000100000011000010110 11...	1000101010000101
10	1001011011100110000101101100011010010110010101101110011000010110001001101100010...	1001100100101111
11	0110100101100111011010000111010001110011001000000110111101100110001000000110000...	1111111110000101
12	0000011011010110010101101101100010011001010111001001110011001000000110111101...	0000001001101000
13	0110100001100101001000000110100001110101011011010110000101101110001000000110011...	1010110100111011
14	1001011011000111100100100000011010010111001100100000011101000110100001100101000...	0011100011010011
15	0111010101101110011001000110000101110100011010010110111101110110000100000011011...	0000110101101100
16	0110011100100110010101100101011001000110111101101101001011000010000001101010010...	1101110110010000

Checksum için ilerle

Menüden “CRC Tablosunu Gör” butonuna tıklandığında bu sayfaya geçiş yapılır. Bu sayfada ise veri parçalarına karşılık gelen CRC değerlerinin bir tablosu gösterilmektedir. Bu tablo, her bir veri parçası için hesaplanan CRC değerlerini sırasıyla listeler. Bu CRC değerleri, her bir veri parçasının doğruluğunu kontrol etmek için kullanılır ve veri iletimi sırasında alıcı tarafı tarafından doğrulama amacıyla kullanılır. Her bir satırda, veri parçasının yanı sıra o parça için hesaplanan CRC değeri de gösterilmektedir.

(CRC Tablosunu Gör)

3.3 Checksum Hesaplama

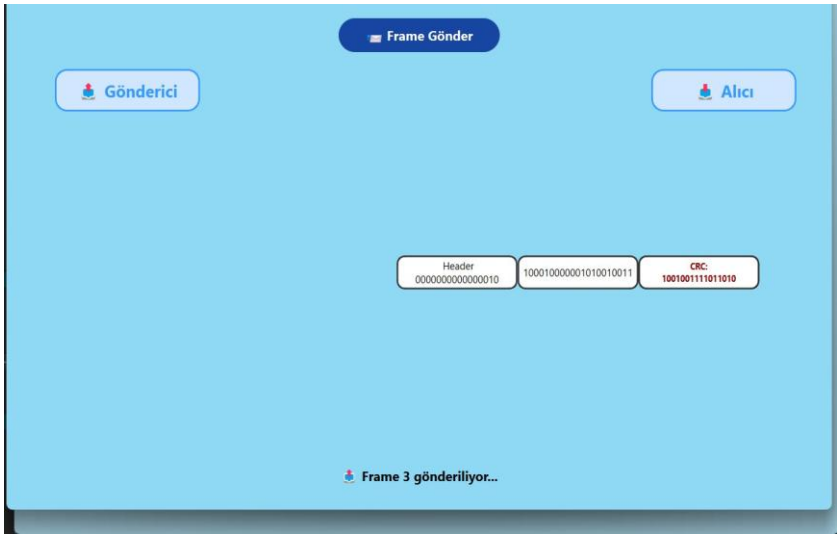


Menüden “Checksum Oluşumunu Gör” butonuna tıklandığında, hesaplama sürecinin animasyonlarla adım adım gösterildiği sayfaya geçiş yapılır. Sayfanın sol tarafında her frame’e ait CRC değerleri liste halinde görüntülenirken, bu değerlerin nasıl toplandığı görsel olarak takip edilebilir. Adım adım gerçekleştirilen toplama işlemi sayesinde, checksum hesaplaması kullanıcı açısından kolay ve anlaşılır bir şekilde sunulmaktadır. Bu sayfada, süreci sadeleştiren ve görselleştiren etkili bir animasyon kullanılmıştır.

(Checksum Oluşumunu Gör)

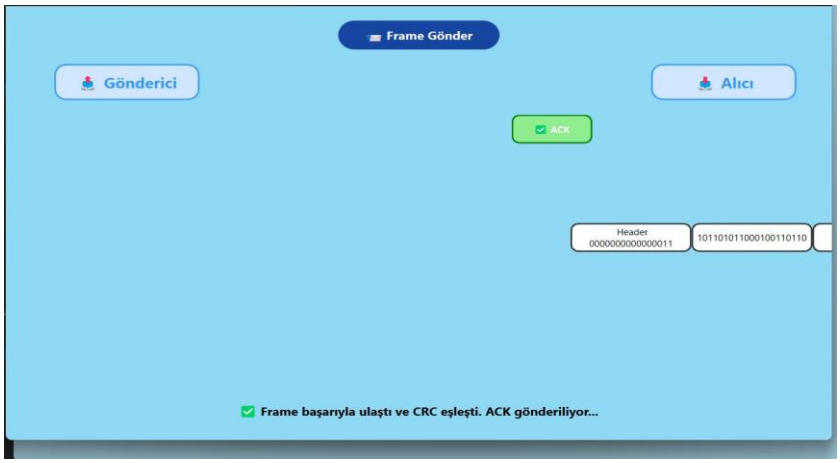
3.4 Simülasyon Akışı

Gönderici ve alıcı olmak üzere ekranın sol ve sağ tarafında konumlandırılan kutular arasında, veri akışı yatay düzlemde animasyonlarla simüle edilmiştir. “Frame Gönder” butonuna basıldığında, frame’lerin iletimi Stop & Wait protokolüne uygun şekilde başlatılır. Her frame gönderilmeden önce, verinin başına header, sonuna trailer ve ayrıca hata kontrolü amacıyla CRC bilgisi eklenerek tam bir frame yapısı oluşturulur. Bu yapı, animasyonlarla birlikte gönderici tarafından alıcıya iletilir. Her frame’in başarılı ya da başarısız şekilde alıcıya ulaşıp ulaşmadığı animasyonlarla görsel olarak gösterilir. Bu sayede, veri iletim süreci ve olası iletişim hataları kullanıcı tarafından anlık ve sezgisel olarak takip edilebilir.

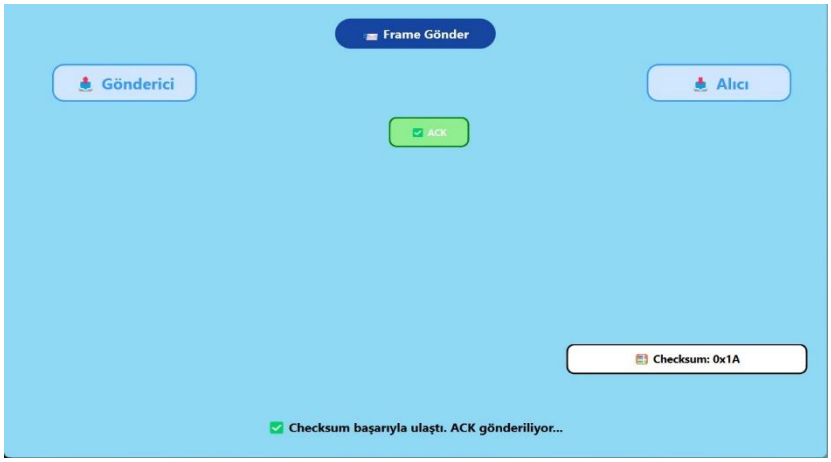


(Veri Gönderimini Gör)

Frame’in yolda herhangi bir problem yaşamadan, başarılı bir şekilde alıcı tarafa ulaşması, veri iletiminin ilk aşamasıdır. Alıcıya ulaşan frame’in veri alanı için CRC hesaplaması yapılır. Bu hesaplanan CRC ile gönderici tarafından iletilen CRC değeri karşılaştırılır. Eğer iki değer de uyumlu ise, bu durum başarılı bir iletimin ikinci aşamasının da sorunsuz tamamlandığını gösterir. Bu iki adımın eksiksiz gerçekleşmesi, birçok olası iletişim hatasına rağmen verinin doğru ve sağlam bir şekilde iletildiğini ifade eder.

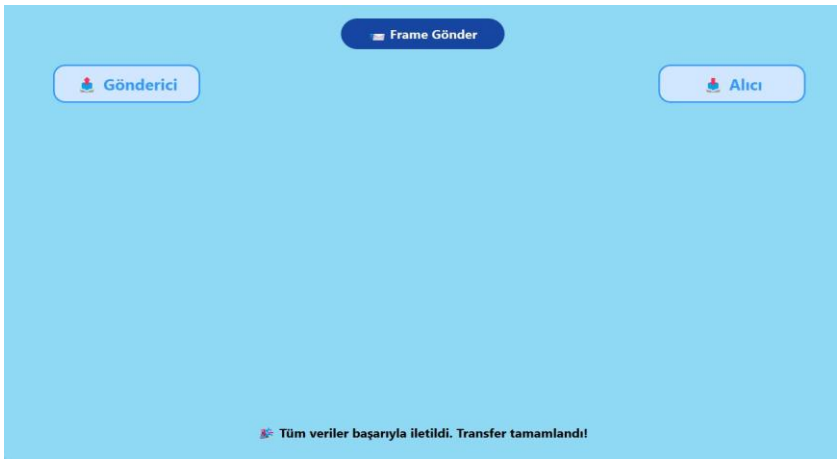


(Frame gönderimi başarılı, CRC başarılı)



Frame'lerin tamamının başarılı bir şekilde iletilmesinin ardından, veri bütünlüğünü doğrulamak amacıyla checksum içeren özel bir frame gönderilir. Alıcı taraf, daha önce aldığı tüm frame'lerden elde ettiği checksum değeri ile göndericiden gelen checksum değerini karşılaştırır. Bu iki değerin uyumlu olması durumunda, veri iletimi sorunsuz ve başarılı bir şekilde tamamlanmış kabul edilir.

(Checksum başarılı)



Checksum değerinin doğrulanmasının ardından, alıcı taraf son bir ACK sinyali gönderir. Bu ACK'in göndericiye ulaşmasıyla birlikte, gönderici taraf tüm veri iletiminin başarılı bir şekilde tamamlandığını teyit eder. Böylece iletişim süreci, hem içerik hem de bütünlük açısından sorunsuz olarak sonlanmış olur.

(Veri iletimi başarılı)

3.5 Hata Simülasyonu

3.5.1 Frame Kaybolması Durumu

Veri gönderimi sürecinde yaşanabilecek olası problemlerden biri, çerçevenin (frame) yolda kaybolmasıdır. Proje kurallarına göre bu durumun gerçekleşme ihtimali %10 olarak belirlenmiştir.

Gönderici, çerçeveyi oluşturup gönderdikten sonra, belirlenen timeout süresi boyunca alıcıdan bir ACK (onay) yanıtı bekler. Ancak eğer çerçeve herhangi bir nedenle alıcıya hiç ulaşmazsa, alıcı bu durumu fark etmez ve doğal olarak hiçbir yanıt göndermez. Timeout süresinin dolmasıyla birlikte, gönderici taraf çerçevenin kaybolduğunu varsayar ve ilgili frame'i tekrar göndermeye karar verir. Bu mekanizma, Stop-and-Wait protokolünün temel davranışını yansıtmaktadır.

Bu senaryo, ağ ortamında paket kayıplarının yaşanabileceğini ve güvenilir iletim için göndericinin yanıt almadığı durumlarda tekrar deneme yapması gerektiğini simüle eder.

(Frame Yolda Kayboldu)

3.5.2 Frame Bozulması Durumu

Bu durumda, çerçeve fiziksel olarak alıcıya ulaşmasına rağmen içerdiği veri iletim sırasında bozulmuş olabilir. Alıcı, gelen veriye ait CRC (Cyclic Redundancy Check) hesaplamasını gerçekleştirir ve bu değeri, gönderici tarafından iletilen CRC değeriyle karşılaştırır. Eğer iki değer arasında uyuşmazlık varsa, bu durum veri üzerinde bir bozulma meydana geldiğini gösterir.

Veri bozulması, genellikle çevresel faktörler, gürültü, donanım sorunları veya ağ üzerindeki parazitlerden kaynaklanabilir. Projede bu senaryonun %20 olasılıkla gerçekleşeceği kabul edilmiştir.

CRC uyuşmazlığı tespit eden alıcı, ACK (onay) mesajı göndermez. Bu durumda, çerçevenin iletiminden sonra ACK bekleyen gönderici, timeout süresinin dolmasıyla birlikte bir sorun olduğunu anlar ve ilgili frame'i tekrar göndermek üzere süreci yeniden başlatır.

Gönderilen frame'in veri alanı, proje kuralları gereği 100 bitten oluşmaktadır. Veri bozulmasını simüle edebilmek için bu 100 bitlik alan içerisinden rastgele bir bit seçilerek XOR

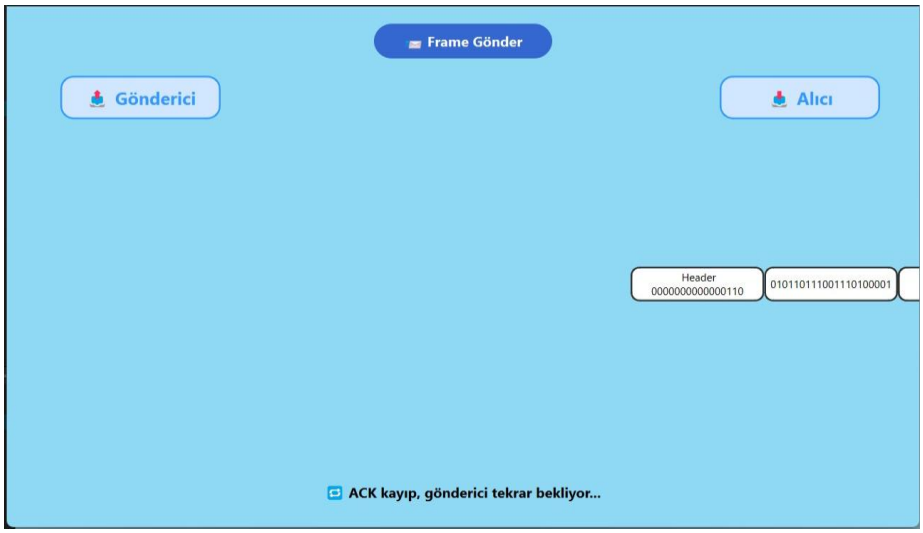
işlemiyle terslenir. Yani ilgili bitin değeri '0' ise '1', '1' ise '0' olarak değiştirilir. Bu küçük değişiklik, CRC hesaplamasının farklı sonuç vermesine neden olur ve alıcı tarafında veri bozulması senaryosunun gerçekçi bir şekilde test edilmesini sağlar.

(Frame Bozuldu)

3.5.3 ACK Kaybı Durumu

Bu durumda, çerçeve alıcıya başarılı bir şekilde ulaşır ve CRC kontrolü hatasız tamamlanır. Alıcı taraf, veri çerçevesinin doğru bir şekilde alındığını bildirmek amacıyla göndericiye ACK (Acknowledgment) mesajı gönderir.

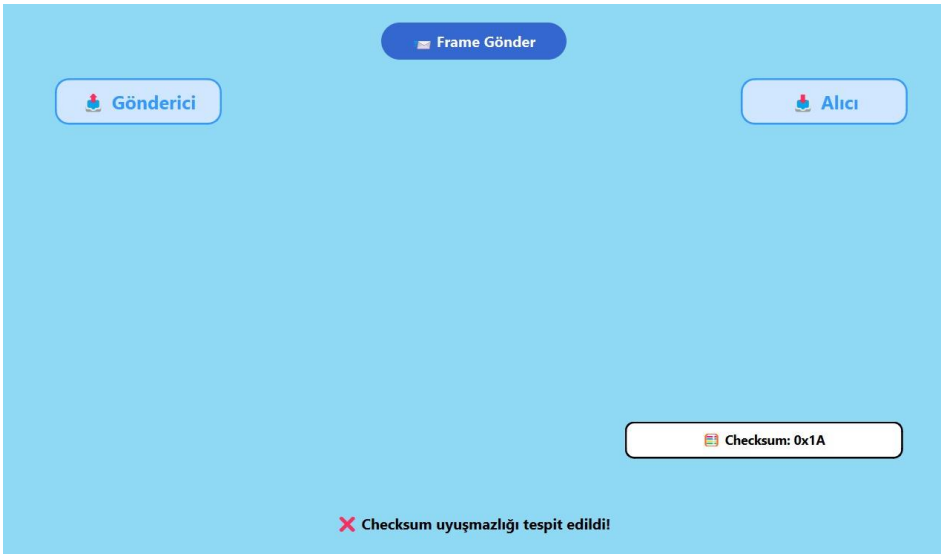
Ancak proje kapsamında, bu ACK mesajının %15 olasılıkla iletim sırasında kaybolabileceği varsayılmıştır. Böyle bir durumda gönderici, çerçeveyi gönderdikten sonra belirlenen timeout süresi boyunca ACK bekler. ACK ulaşmazsa, gönderici veri iletiminde bir sorun olduğunu varsayarak aynı çerçeveyi tekrar gönderir. Bu senaryo, geri bildirim (feedback) kaybı gibi gerçek dünya iletişim sorunlarını simüle etmiştir.



(ACK Kayıp)

3.5.4 Checksum Bozulması Durumu

Tüm veri çerçeveleri başarıyla iletildikten sonra, gönderici tarafından checksum değeri hesaplanır ve bu değer ayrı bir çerçeve olarak alıcıya gönderilir. Alıcı taraf, kendi aldığı veri çerçeveleri üzerinden aynı işlemi yaparak checksum'ı yeniden hesaplar ve gelen değerle karşılaştırır.



Proje kurallarına göre, bu özel checksum çerçevesinin iletimi sırasında %5 oranında hata oluşabileceği kabul edilmiştir. Eğer alıcı taraf iletilen checksum ile kendi hesapladığı değeri eşleştiremezse, veri bütünlüğünün bozulduğu anlaşılır ve iletim süreci geçersiz sayılır.

(Checksum Bozuldu)

4.Sonuç

Bu projede, OSI modelinin Veri Bağlantı Katmanı'na ait temel işlevler başarıyla simüle edilmiştir. Gönderici ile alıcı arasındaki veri aktarımı; çerçeve kaybı, veri bozulması, ACK (onay) kaybı gibi olasılıklar dikkate alınarak modellenmiş, Stop & Wait protokolü temel alınarak güvenli iletişim süreci oluşturulmuştur. Veri bütünlüğünü sağlamak amacıyla CRC ve checksum gibi hata kontrol teknikleri uygulanmış ve iletimin doğruluğu test edilmiştir.

Qt framework'ü kullanılarak geliştirilen grafiksel kullanıcı arayüzü sayesinde, kullanıcılar veri iletim sürecini hem görsel olarak takip edebilmekte hem de gerçekleşen tüm işlemleri sezgisel olarak gözlemleyebilmektedir. Animasyon desteği ile desteklenen bu arayüz, özellikle hata senaryolarının anlaşılmasını kolaylaştırmıştır.

Bu proje, teorik bilgilerin somut örneklerle pekiştirilmesini sağlamış ve kullanıcıyla etkileşimli bir simülasyon ortamı sunarak uygulamalı öğrenmeyi desteklemiştir. Gelecekte farklı akış kontrol protokollerinin ya da ileri düzey hata düzeltme algoritmalarının entegrasyonu ile sistem daha kapsamlı hale getirilebilir.