**PROJE ADI:**

**Perkarlo User Datagram Protocol Chat**

**ÖZET:**

Projenin özeti; “User Datagram Protocol” ile hızlı, güvenilir, şeffaf ve güvenli bir mesaj transferi yaparak kurum ya da bireylerin siber güvenliğini sağlamaktır. Yapılan araştırma, ön hazırlık ve uygulamadan sonra projeyi kullanılabilir hale getirip kullanıcılara test ile geri dönüşler alındı ardından projenin amacına ulaştığı görüldü.

**Anahtar Kelimeler:** User Datagram Protocol, Siber Güvenlik, Mesaj Transferi

**1.GİRİŞ**

Siber Güvenlik, tüm bilişim sistemlerini elektronik yapıları ve internette bağlı olarak çalışan bütün sistemleri ilgilendiren genel bir güvenlik kavramıdır. Birçok kez bilgi güvenliği ile karıştırılabilir ancak siber güvenlik bilgi güvenliğini bünyesine alan bir yapıdır.

Günlük hayatımızda neredeyse bütün kurumlar verimli üretken bir şekilde bilgi paylaşımı yapabilmek adına bilgisayar ağlarına güveniyorlar. Artık bilgisayar ağları çok geniş ve her yerde yer almaya başlamış durumdadır. Her personelin kendisine ait iş istasyonu olarak adlandırdığımız bilgisayara sahip olduğunu düşünürsek, 50 personelli ufak bir kurumda bile 50 adet sızıntıya açık hedef var diyebiliriz.

Ağ güvenliği yapısı bakımından OSI dediğimiz katman sisteminin bütün katmanlarında olması gerekmektedir. Güvenlik yönetimi uzun bir süreçtir ve açıklayacak olursak ağ üzerinde çalışan her şeyin güvenliğini içerir ve sürekliliğe bağlıdır.

Güvenli iletişim için gereken özellikler şunlardır:

* **Gizlilik**
* **Bütünlük**
* **Erişebilirlik**

**Perkarlo User Datagram Protocol** **Chat**’i diğer UDP Chat programlarından ayıran özellikler;

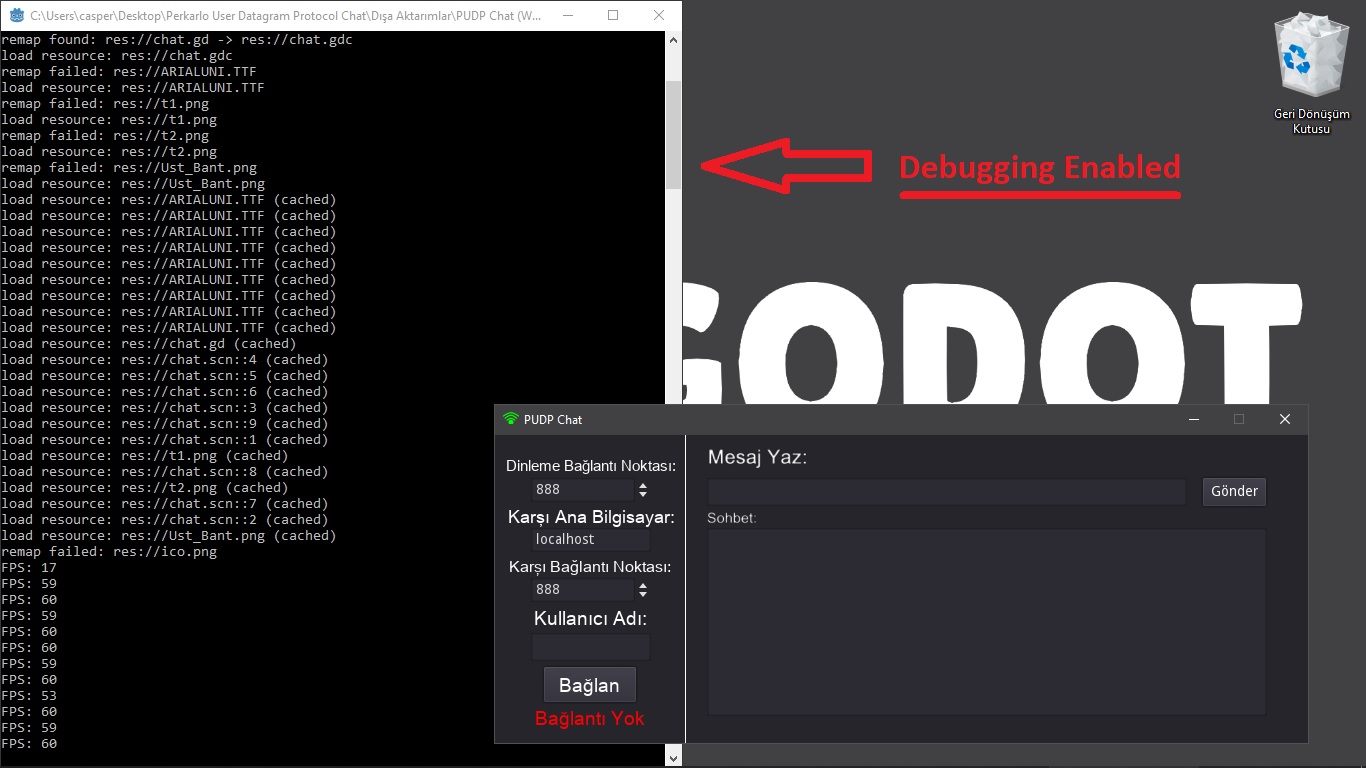
* **Açık Kaynak olması**,
* **Yaptığı işlemleri Debugging penceresinde sunması**,
* **Mesajları sadece RAM’e kaydetmesi**,
* **Bağ kesildikten sonra RAM’i temizlemesi**,
* **Bir server bulundurmaması**,
* **Çok hızlı olması**,
* **Gönderdiği paketlerin boyutunun çok küçük olması**,
* **Böylece veri kaybının neredeyse yaşanmaması**,
* **Windows dışında Linux için de olması**,
* **Port numaralarını bir şifre gibi kullanması** vb.’dir.

**1.1 PROBLEM**

Kurumlar, önemli mesajlarını “E-Posta” ile ilgili birimlere gönderirler ama bazen kullanılan E-Posta servisine, ilgili birimin bilgisayarına ya da kurumun bilgisayarına istenmeyen kişiler sızabiliyor. Bu gibi durumlarda kurum bilgileri çalınabiliyor ve kurumun istemediği şekilde kullanılabiliyor.

**1.2 PROJENİN AMACI**

Projem, kurumların ya da bireylerin mesajlarının istenmeyen kişilerden olabildiğince uzak tutmak için hayata geçirilmiştir. Projemin diğer aynı amacı güden projelerden farkı açık kaynak olup “Debugging Enabled”dır. Proje kullanılırken diğer diğer pencereden de yapılan işlemler ve anlık FPS değerleri Resim 1’deki gibi gözükür.



**Resim 1**:Debugging Enabled

**1.3 NEDEN GODOT ENGINE VE GDSCRIPT**

Öncelikle; neden Godot Engine? Godot Engine hızlı, anlaşılır ve açık kaynak bir “Oyun Motoru”dur. Godot Engine bir oyun motoru olmasına rağmen tamamen oyunlar üzerine odaklanmış bir motor değildir; açık kaynaklı olduğu için üzerinde dilenildiği gibi değişiklik yapılabilir. Neden GDScript? Projem çok az bir performans istediği için yüksek seviyeli bir dil kullanabilirdim. Bu yüzden önce araştırma yaptım ve sonunda 3 dil sonuçlandırdım; Ruby, Python ve GDScript. Önce Ruby diline baktığımda; Ruby dilinin oldukça esnek ve kolay olduğunu gördüm ama biraz fazla yavaş olduğunu gözlemledim. Python diline baktığımda; oldukça fazla kaynak olduğunu, hızlı ve yeterli olduğunu gözlemledim ama Kullanıcı Arayüzü için biraz uğraşacağımı anladım ve ne olur ne olmaz daha önceden de çalıştığım GDScript’e baktım ve kullanacağım User Datagram Protocol’ü için daha önce bir kılavuz olduğunu gördüm ve kullanacağım dili GDScript olarak seçtim.

**2. YÖNTEM**

Projemin yapım süreci 4 ana başlık altında toplandırıldı. Bunlar;

* **Araştırma ve Ön Hazırlık Yapma**,
* **Kullanıcı Arayüzü Oluşturma**,
* **Program Kodunu Yazma ve Kullanıcı Arayüzüne Bağlama**,
* **Programı Test Etme**’dir.

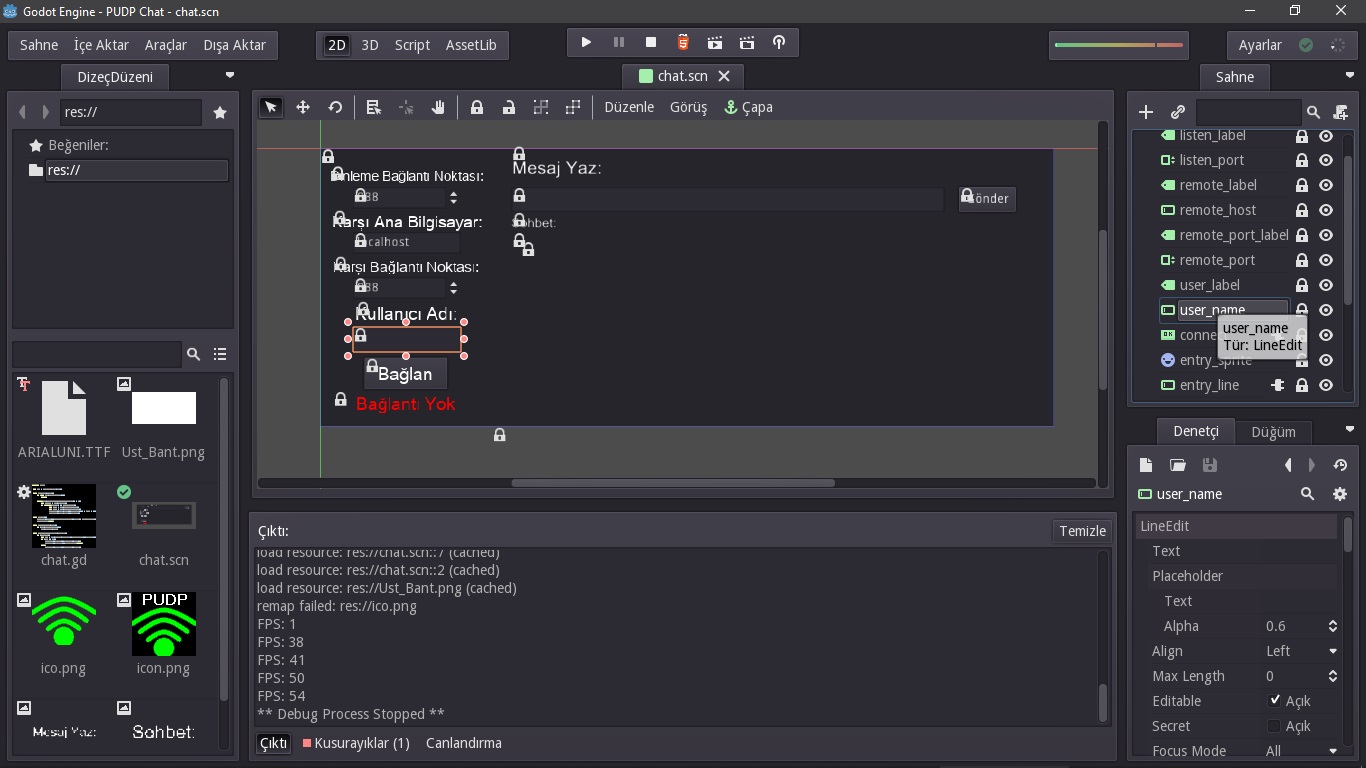
**3. BULGULAR**

**3.1 ARAŞTIRMA VE ÖN HAZIRLIK YAPMA**

Proje yapımında geri dönülmesi güç hatalardan kaçınmak için detaylı bir araştırma ve iyi bir ön hazırlık yapmam gerektiğini biliyordum. Bu yüzden internet ve kütüphaneden sıkı bir ön çalışma yaptım.

**3.2 KULLANICI ARAYÜZÜ OLUŞTURMA**

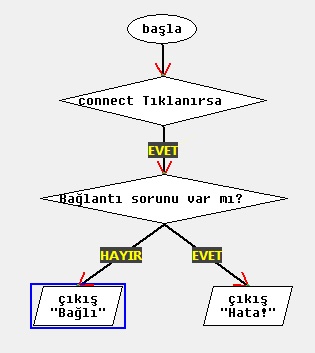
Bu aşamada Resim 2’de görüldüğü gibi Godot Engine programını kullanarak Kullanıcı Arayüzünü (UI) oluşturdum. “User Interface” tasarımında, tasarımın kullanıcı odaklı ve basit olmasına özen gösterilmiştir.



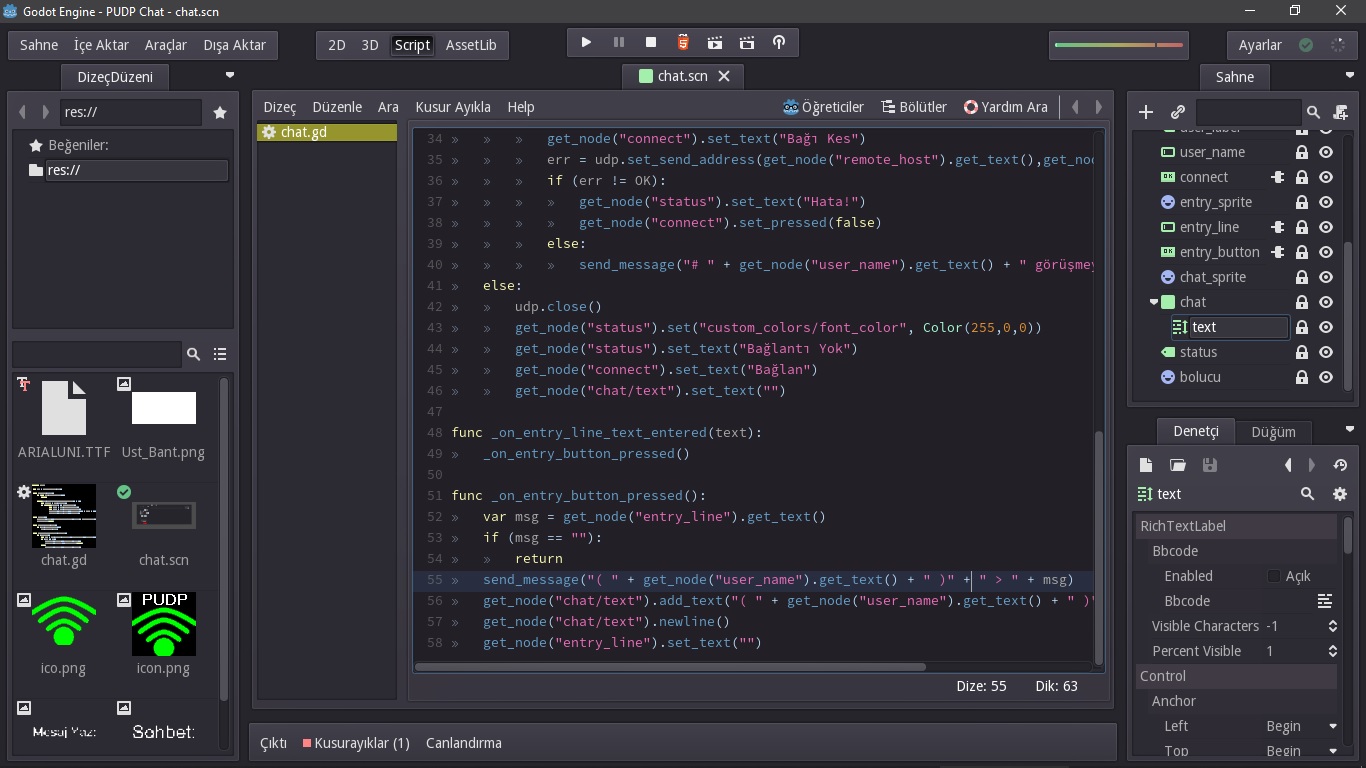
**Resim 2**: Kullanıcı Arayüzü Tasarımı

**3.3 PROGRAM KODUNU YAZMA VE KULLANICI ARAYÜZÜNE BAĞLAMA**

Araştırma ve Ön Hazırlık (bkz. 3.1 Araştırma ve Ön Hazırlık ) aşamasında yaptığım çalışmalar büyük çoğunlukla bu aşamada işime yaradı. Zaten programın Kullanıcı Arayüzü vardı (User Interface) ama bu aşamada asıl yapılmak istenilen amacı koda döküp Kullanıcı Arayüzüne bağladım (User Experience) ama önce kodu tasarladım (Resim 3) sonra kodu yazdım (Resim 4).



**Resim 3**: Görüşme bağlantısına ait ufak bir diyagram



**Resim 4**: Projenin GDScript kodu

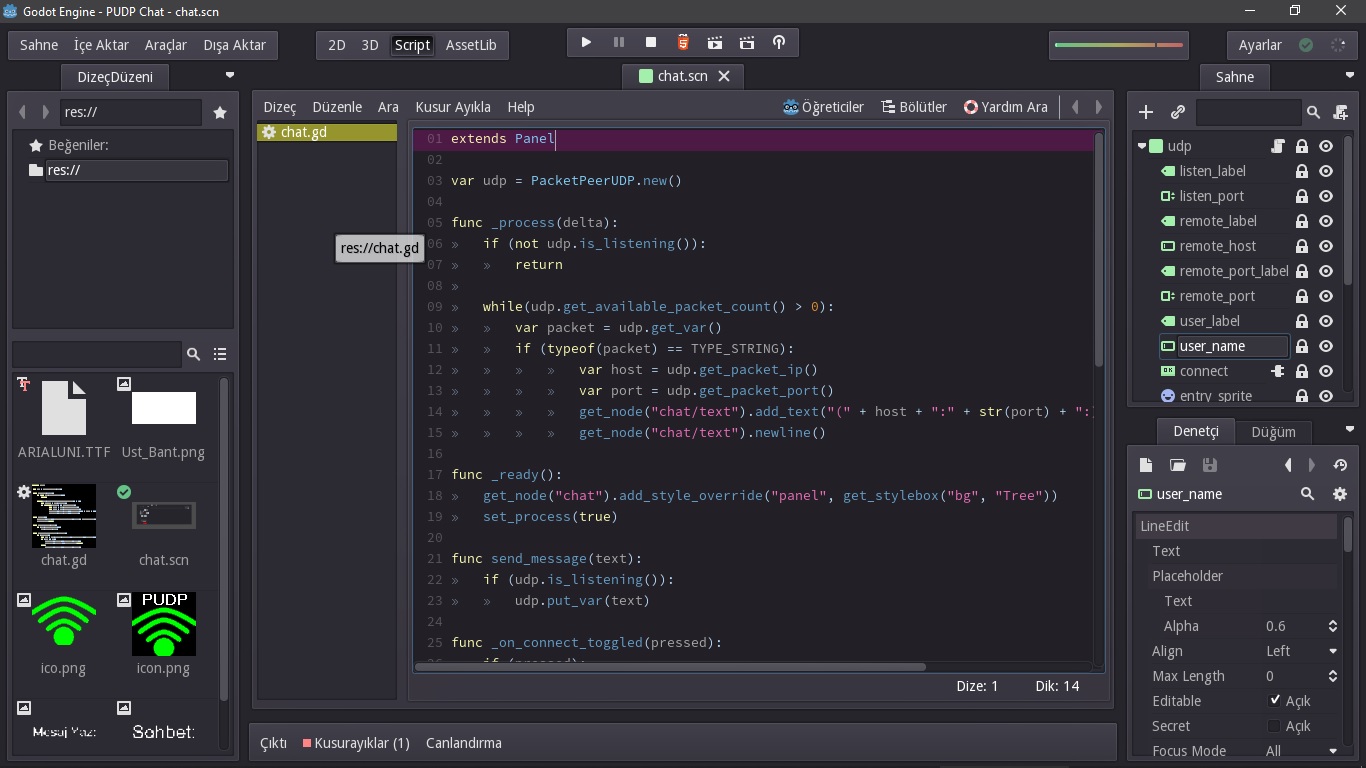
**3.4 PROGRAMI TEST ETME**

Ve son olarak evimde yaptığım bu projeyi okuluma götürüp sınıf arkadaşlarım ile birlikte test ederek çalışabilirliğini onayladık.

**3.5 PROJE KODLARININ AÇIKLANMASI**

Bu başlıkta projenin içindeki kodları ve işlevlerini kısaca anlatacağım. Programlama dili: GDScript; kod dosyası: chat.gd; Dosyadaki kod satırı sayısı: 58.

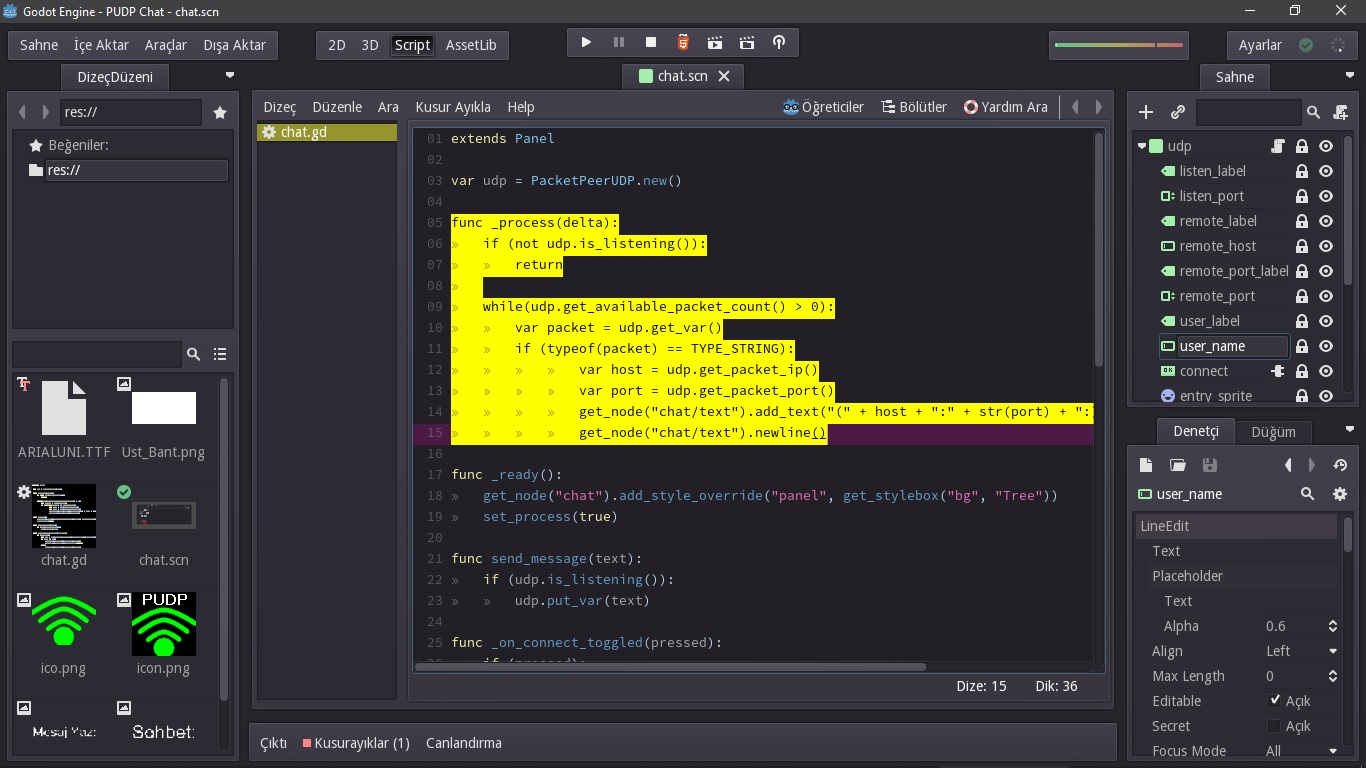
**3.5.1 RESİM 5**



**Resim 5**

Resim 5’te mor ile gösterilen kodda; dosyanın içeriğine yazılan tüm kodların bir Panel üzerinde çalışacağını belirtir.

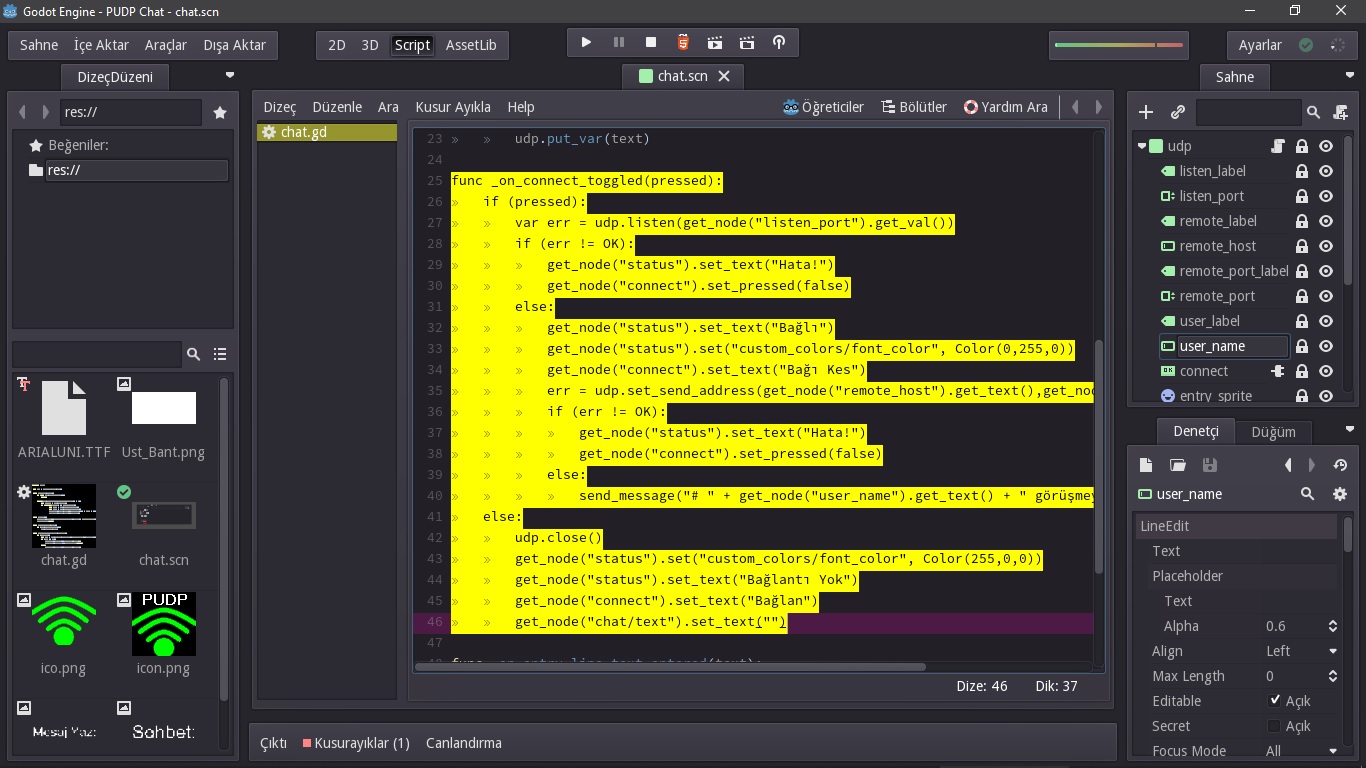
**3.5.2 RESİM 6**

****

**Resim 6**

Resim 6’da sarı ile gösterilen kodda; karşı bilgisayardan gelen veri paketlerini kontrol edip belli parametreler ile bize ulaştırıyor.

**3.5.3 RESİM 7**

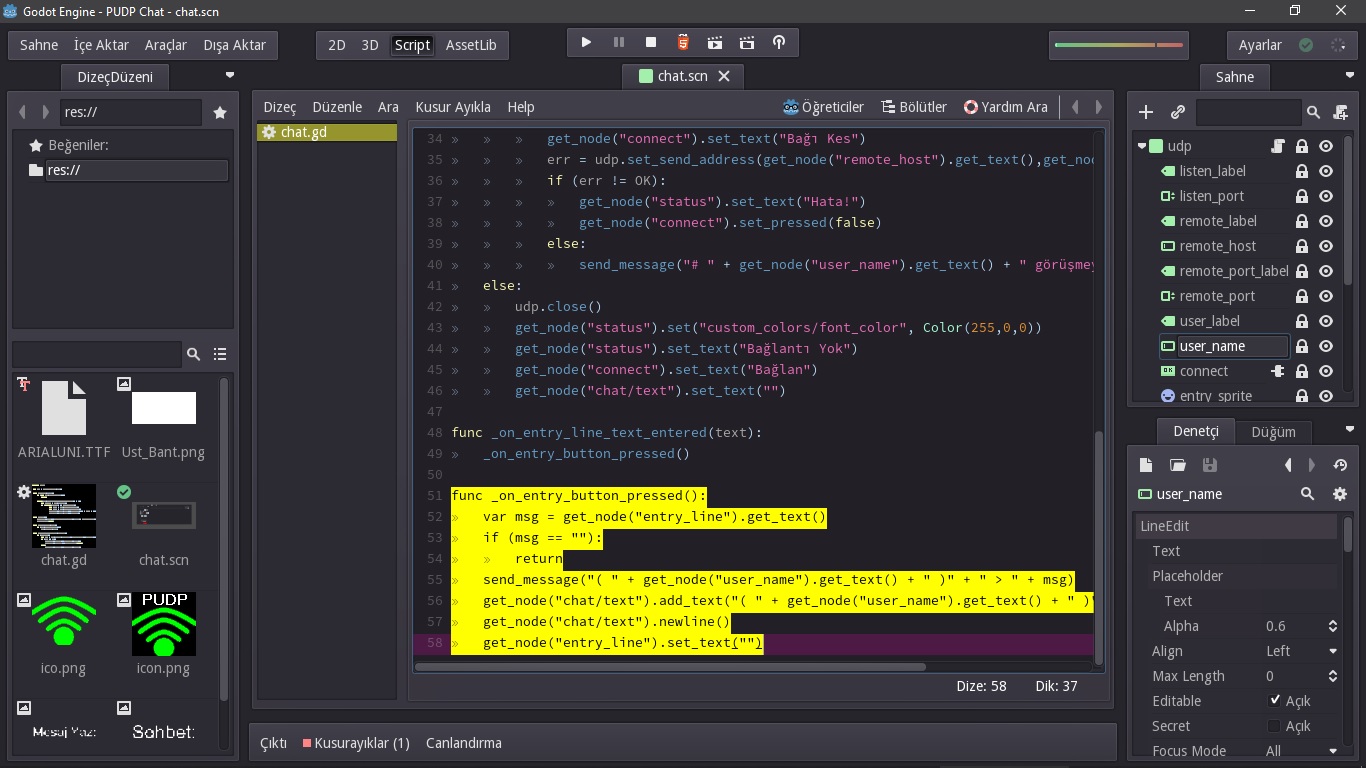
****

**Resim 7**

Resim 7’de sarı ile gösterilen kodda; Resim 3’ün geliştirilerek koda dökülmesini görüyoruz. Bu sarı ile vurgulu kodda, karşı bilgisayara bağlanma işlemini yapıyoruz. Bu kod dizininin üç çıktısı vardır. Bunlar;

1. Olumlu sonuç ile bağlanıp karşı tarafa bildirmek
2. Olumsuz sonuç ile bağlanamayıp hata vermek
3. El ile bağı kesip tüm görüşmeyi RAM’den silmek

**3.5.4 RESİM 8**

****

**Resim 8**

Resim 8’de sarı ile gösterilen kod karşı tarafa mesaj göndermemizi sağlar.

**3.5.5 TÜM KOD (chat.gd)**

**extends Panel**

**var udp = PacketPeerUDP.new()**

**func \_process(delta):**

**if (not udp.is\_listening()):**

**return**

**while(udp.get\_available\_packet\_count() > 0):**

**var packet = udp.get\_var()**

**if (typeof(packet) == TYPE\_STRING):**

**var host = udp.get\_packet\_ip()**

**var port = udp.get\_packet\_port()**

**get\_node("chat/text").add\_text("(" + host + ":" + str(port) + ":) " + packet)**

**get\_node("chat/text").newline()**

**func \_ready():**

**get\_node("chat").add\_style\_override("panel", get\_stylebox("bg", "Tree"))**

**set\_process(true)**

**func send\_message(text):**

**if (udp.is\_listening()):**

**udp.put\_var(text)**

**func \_on\_connect\_toggled(pressed):**

**if (pressed):**

**var err = udp.listen(get\_node("listen\_port").get\_val())**

**if (err != OK):**

**get\_node("status").set\_text("Hata!")**

**get\_node("connect").set\_pressed(false)**

**else:**

**get\_node("status").set\_text("Bağlı")**

**get\_node("status").set("custom\_colors/font\_color", Color(0,255,0))**

**get\_node("connect").set\_text("Bağı Kes")**

**err = udp.set\_send\_address(get\_node("remote\_host").get\_text(),get\_node("remote\_port").get\_val())**

**if (err != OK):**

**get\_node("status").set\_text("Hata!")**

**get\_node("connect").set\_pressed(false)**

**else:**

**send\_message("# " + get\_node("user\_name").get\_text() + " görüşmeye katıldı.")**

**else:**

**udp.close()**

**get\_node("status").set("custom\_colors/font\_color", Color(255,0,0))**

**get\_node("status").set\_text("Bağlantı Yok")**

**get\_node("connect").set\_text("Bağlan")**

**get\_node("chat/text").set\_text("")**

**func \_on\_entry\_line\_text\_entered(text):**

**\_on\_entry\_button\_pressed()**

**func \_on\_entry\_button\_pressed():**

**var msg = get\_node("entry\_line").get\_text()**

**if (msg == ""):**

**return**

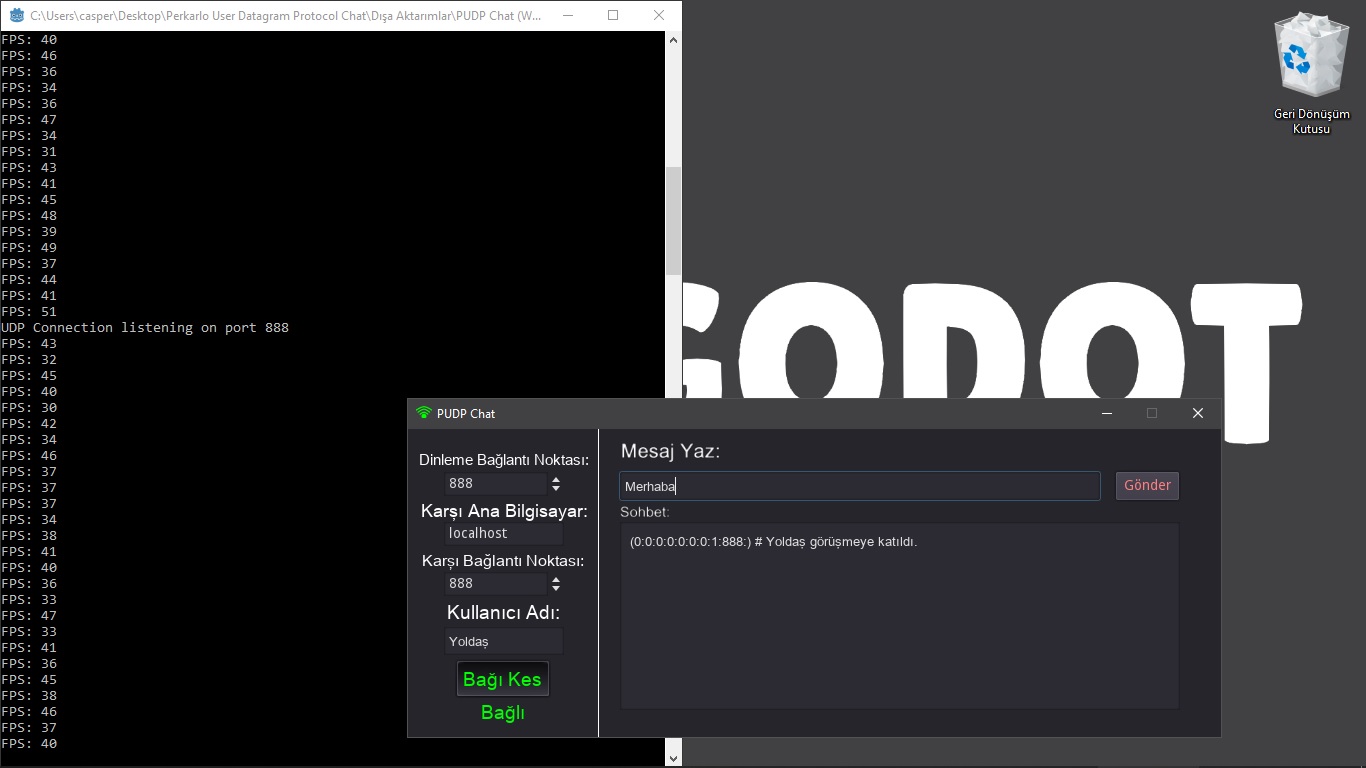
**send\_message("( " + get\_node("user\_name").get\_text() + " )" + " > " + msg)**

**get\_node("chat/text").add\_text("( " + get\_node("user\_name").get\_text() + " )" + " > " + msg)**

**get\_node("chat/text").newline()**

**get\_node("entry\_line").set\_text("")**

**3.5.6 PROJE EKRAN GÖRÜNTÜSÜ**

****

**4. SONUÇLAR VE TARTIŞMA**

Godot Engine programını kullanarak yaptığım ve GDScript ile kodladığım PUDP Chat programı ilk atılımında sonucuna ulaşmış ve test edilmiştir. 5 kişilik ve 2 kişilik ufak 2 ekip ile yaptığım test ve geri dönüşler bana programın şeffaf ve kararlı çalıştığını gösteriyor.

Bu ulaşımlar bana projenin amacına (bkz. 1.2 PROJENİN AMACI) ulaştığını da gösteriyor. Program “User Datagram Protocol”ü kullanmasına rağmen şu ana kadar yapılan testlerde veri kaybı yaşanmamıştır.

**5. ÖNERİLER**

Proje açık kaynaktır; yani geliştirmeye açıktır. Projeyi devam ettirmek isteyenlere önerilerim;

* Programa “Kişiler” sekmesi eklenebilir ve kişi kayıt edilebilir.
* Program bu evrede sadece metin gönderebiliyor; programı geliştirmek isteyenler resim, video vb. medya gönderim imkânı ekleyebilirler.

**KAYNAKÇA**

Bozkurtlar, B. (2017). Neden Ağ Güvenliği?. ANNE BEN HACKER OLUYORUM (1.Baskı) içinde (100-101). İstanbul: İKİNCİ ADAM YAYINLARI.

Bozkurtlar, B. (2017). 1.2 Temel Siber Güvenlik. ANNE BEN HACKER OLUYORUM (1.Baskı) içinde (17). İstanbul: İKİNCİ ADAM YAYINLARI.

PacketPeerUDP. (2014-2018). Godot Docs. http://docs.godotengine.org/en/2.1/classes/class\_packetpeerudp.html.