**TCP/IP Server & Client Demo**

TCP/IP Server demosu NUCLEO-F429ZI kartı üzerinde kurulmuştur. Amaç TCP/IP Server cihaz üzerinden Client’a sürekli olarak belli aralıklarla dummy veri göndermektir.

1. **Ethernet Konfigürasyonu**

Bahsedilen konfigürasyonlar TCP/IP Server ve Client cihazlar için de geçerlidir.

* 1. **Projenin Oluşturulması**

Graphical user interface, application, Word

Description automatically generatedÖnce yeni bir proje oluşturulmalıdır. Sol üst köşedeki ***“File”*** kısmından ***“New”*** ve ***“STM32 Project”*** seçeneği seçilmelidir. Çıkan ekranda ***“Board Selector”*** kısmına gelinmeli ve ***“Commercial Part Number”*** kısmına ***“NUCLEO-F429ZI”*** yazılmalıdır (Figure 1).

Figure 1 - Proje Kurulumu

***“Boards List”*** kısmından kart seçilir, projeye isim verilir ve proje oluşturulur. Tercihen proje default ayarlarda oluşturulmamalıdır. Proje oluşturulduktan sonra projenin ioc dosyası açılır. Yine tercihen ioc dosyası açıldıktan sonra ***“Pinout”*** kısmından ***“Clear Pinout”*** seçeneği seçilerek mevcut seçili pinoutlar temizlenmelidir.

* 1. **Clock Konfigürasyonu**

Kartın clock ayarının yapılması için ***“System Core”*** kısmından ***“RCC”*** ekranı açılır. ***“High Speed Clock (HSE)”*** seçeneği ***“Crystal/Ceramic Resonator”*** olarak seçilmiştir. ***“Low Speed Clock (LSE)”*** seçeneği kapalı tutulmuştur (Figure 2). Demo oluşturulurken clock ayarları bu şekilde tercih edilmiştir. Farklı clock ayarları ile de proje çalışabilir.

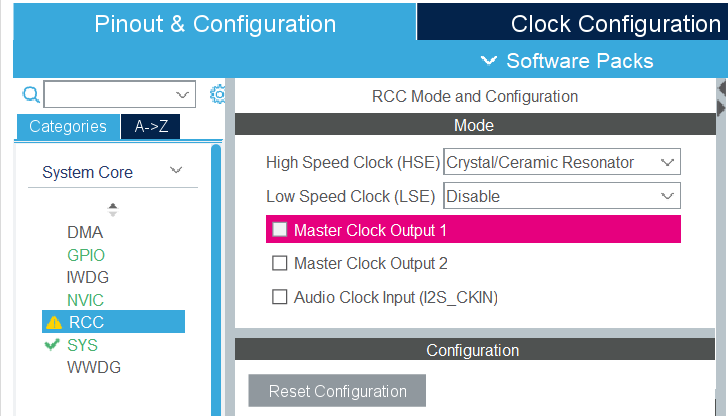


Figure 2 - Clock Seçimi

***Graphical user interface, diagram

Description automatically generated“Clock Configuration”*** kısmına gelinir. ***“HCLK”*** seçeneği maksimumda çalıştırılması tercih edilmiştir. 180 MHz olarak ayarlanmıştır. ***“Clock Configuration”*** kısmı sorunsuz bir şekilde ayarlandıktan sonra clock ayarları tamamlanmıştır.

Figure 3 - Clock Konfigürasyonu

* 1. **Timer Konfigürasyonu**

***“Timer”*** kategorisine gelinir ve ***“TIM1”*** seçilir. Tercihen ***“TIM1”*** kullanılmıştır. İstenirse farklı timer kullanılabilir. Bu noktada önemli olan seçilen timer’ın hangi APB hattına bağlı olduğudur. ***“TIM1”*** APB2 hattına bağlıdır. Eğer farklı bir timer kullanılacaksa STM32F429ZI Reference Manuel incelenerek seçilen timer’ın hangi APB hattına bağlı olduğu bulunabilir. Bunun için Reference Manuel’de ***“2.3 Memory Map”*** kısmına gelinir. Tabloda ***“Peripheral”*** kısmında TIM1 görüldüğü noktada yazan ***“Bus”*** seçeneği hangi APB hattına bağlı olduğunu belirtmektedir.

Table

Description automatically generated

Figure 4 - TIM1 BUS

***“TIM1”*** seçeneği seçildikten sonra ***“Mode”*** kısmından ***“Clock Source”*** seçeneği ***“Internal Clock”*** olarak seçilir.

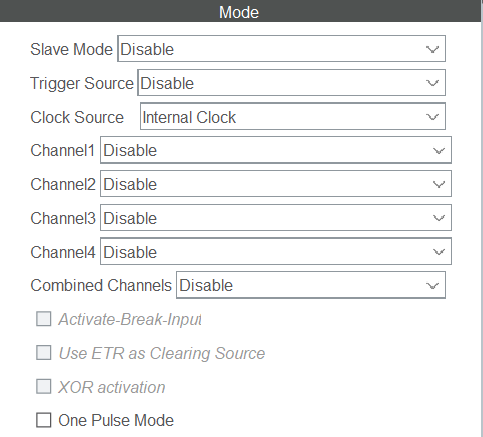
Timer’ın 1 saniye aralıklarla çalışması istenmektedir. ***“Configuration”*** kısmına gelinerek ***“Parameter Settings”*** kısmında ***“Prescaler”*** ve ***“Counter Period”*** 1 saniye için ayarlanmalıdır. Timer’ın çalışacağı süre aralığı, ilgili APB timer clock değeri ***“Prescaler”*** değerine bölünür. Çıkan sonuç da ***“Counter Period”*** değerine bölünür. Çıkan sonuç eğer 1 olursa timer süresi 1 saniye olacaktır. Bu yüzden ***“Prescaler”*** değeri ***“18000-1”*** ve ***“Counter Period”*** değeri ***“10000-1”*** olarak girilmiştir.

Figure 5 - TIM1 Mode

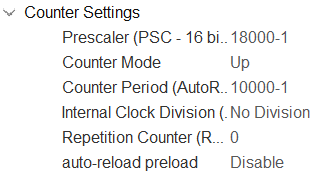


Figure 6 - Timer PSC ve CNT Konfigürasyonu

***“Configuration”*** kısmında ***“NVIC Settings”*** içinde ***“TIM1 update interrupt and TIM10 global interrupt”*** seçeneği active edilmelidir. Bu ayarın tamamlanması ardından timer konfigürasyonu tamamlanacaktır.

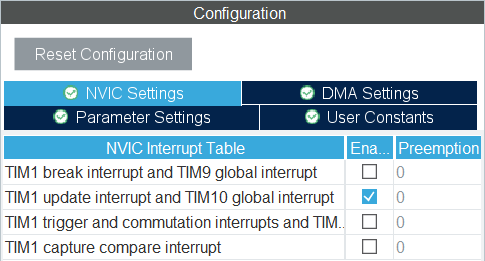


Figure 7 - Timer NVIC Konfigürasyonu

* 1. **Ethernet Konfigürasyonu**

**Graphical user interface, text, application, email

Description automatically generated“Categories”** kısmından **“Connectivity”** altındaki **“ETH”** aktif edilmelidir. Bu noktada kullanılacak olan kartın MII’mı yoksa RMII’mı desteklediği bilinmelidir. Kullanılan kart örneğin yazılımsal olarak RMII destekliyor ama donanımsal olarak desteklemiyor olabilir. Bu bilginin kontrolü için ilgili kartın şematik dosyası incelenebilir. NUCLEO-F429ZI kartı için şematik dosyasına ulaşmak adına şu adımlar izlenebilir. Tarayıcı üzerinde **“NUCLEO-F429ZI schematic”** şeklinde arama yapılır. **“NUCLEO-F429ZI – STMicroelectronics”** isimli arama sonucuna girilir. **“CAD Resources”** sekmesinde **“Schematic Pack”** başlığı altında **“STM Nucleo (144 Pins) schematics”** isimli dosya indirilir.

Figure 8 - NUCLEO-F29ZI Şematik Dosyalarına Erişim

Diagram, schematic

Description automatically generatedİndirilen zip’in içinde hem tek tek Altium dosyaları hem de tüm şematiğin olduğu **“MB1137”** bir pdf dosyası bulunmaktadır. Pdf dosyasında sayfa 5’te **“LAN8742A-CZ-TR”** çipi ve çevre birimleri görülebilir.

Figure 9 - LAN8742 Bağlantıları

Figür 9’da çipin bağlantılarının RMII ile yapıldığı görülebilir. Bu yüzden **“ETH”** konfigürasyonu sayfasında **“Mode”** seçeneği **“RMII”** seçilebilir.

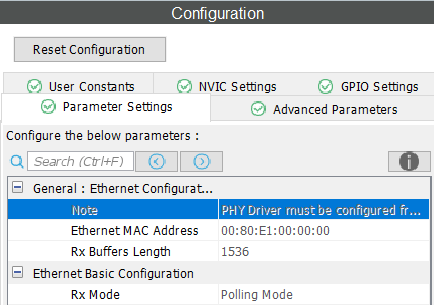
**“Parameter Settings”** kısmında bir not düşüldüğü görülebilir. **“PHY Driver must be configured from the LwIP ‘Platform Settings’ top right tab”** şeklinde bir not düşülmüştür. İlgili ayar, raporun **“LwIP Konfigürasyonu”** kısmında yapılmaktadır.

Figure 10 - Ethernet Parametre Konfigürasyonu

**Graphical user interface, text, application, email

Description automatically generated“Advanced Parameters”** sekmesinde **“PHY”** ayarının **“LAN8742A\_PHY\_ADDRESS”** olarak seçili olduğundan emin olunması gerekmektedir.

Figure 11 - Ethernet PHY Konfigürasyonu

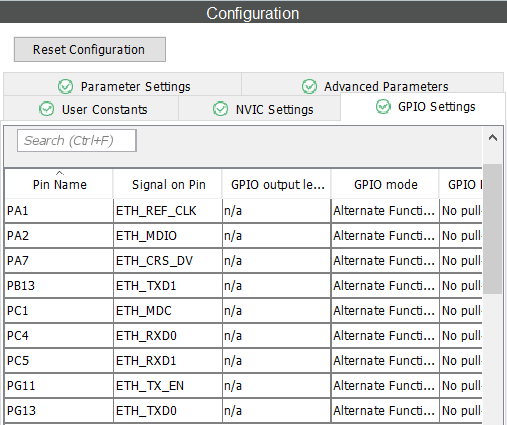
**“GPIO Settings”** sekmesinde ilgili sinyallerin hangi pinlere bağlı olduğu görülmektedir. CubeIde default olarak bu pinleri bazen yanlış konfigüre edebilmektedir. Pinlerin doğru olduğundan emin olunması için kontrol edilmesi gerekmektedir. Figür 9’da pin konfigürasyonun nasıl olması gerektiği gösterilmiştir. Eğer düzeltimesi gereken bir pin olması durumunda **“Pinout View”** sekmesinde çipin ilgili pinine sağ tıklayarak, o pine atanabilecek olan sinyaller görülebilir. Pinoutla ilgili bir hata olması durumunda bu şekilde ilgili sinyaller ilgili pinlere atanabilir.

Figure 12 - Ethernet Pin Konfigürasyonu

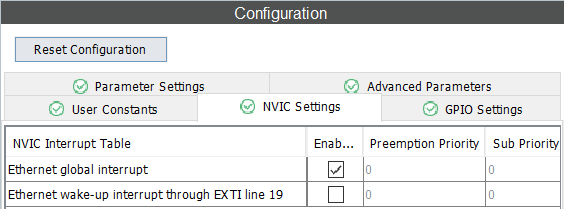
**“NVIC Settings”** sekmesinde **“Ethernet Global Interrupt”** ayarının aktif edilmesi gerekmektedir.

Figure 13 - NVIC Konfigürasyonu

* 1. **LWIP Konfigürasyonu**

***“Categories”*** kısmından ***“Middleware”*** altında ***“LWIP”*** seçeneği active edilmelidir. ***“LWIP”*** seçeneği altındaki ***“Mode”*** kısmından ***“Enabled”*** kutucuğu aktive edilmelidir. Bu kutucuk aktive edildikten sonra ***“Configuration”*** kısmı aktif olacaktır.

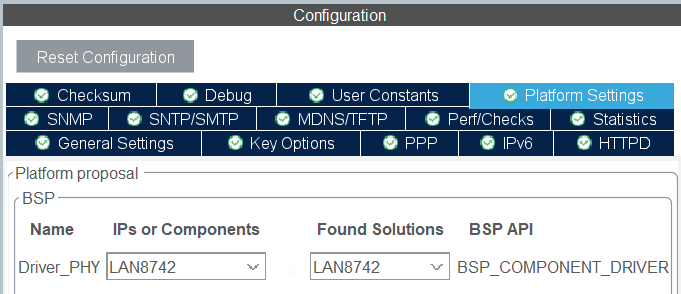
***“Platform Settings”*** kısmında ***“Driver\_PHY”*** seçeneği olarak ***“LAN8742”*** seçilmelidir. Bu seçenek kart üzerindeki ethernet çipini seçmektedir. Eğer seçenek seçilmezse ethernet çalışmayacaktır. Aynı zamanda kod derlendiğinde de hata alınmaz. O yüzden ayarı gözden kaçırmamak gerekmektedir.

Figure 14 - Platform Settings

***“Checksum”*** kısmında ***“CHECKSUM\_BY\_HARDWARE”*** seçeneği ***“Enabled”*** olmalıdır.

Graphical user interface

Description automatically generated

Figure 15 - Checksum Konfigürasyonu

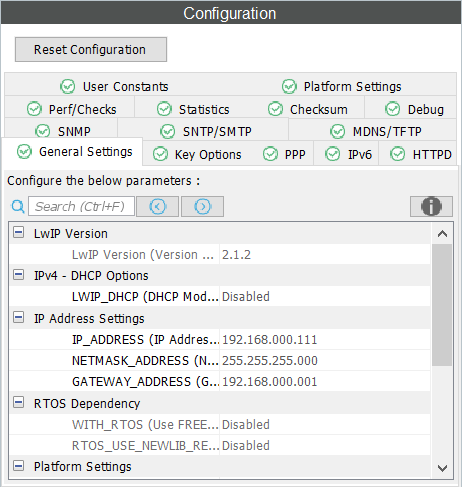
**“*General Settings”*** kısmında ***“LWIP\_DHCP”*** ayarı ***“Disabled”*** olmalıdır. Bu ayarın ***“Disabled”*** olması ethernet server’ı için manuel ip verebilmemizi sağlamaktadır. ***“LWIP\_DHCP”*** ayarı ***“Disabled”*** edildikten sonra ***“IP\_ADDRESS”***, ***“NETMASK\_ADDRESS”*** ve ***“GATEWAY\_ADDRESS”*** ayarları yapılmalıdır. Buradaki ayarlar istendiği gibi konfigüre edilebilir. Fakat ***“IP\_ADDRESS”*** değeri sonrasında ethernet bağlantılarında kullanılacağı için unutulmamalıdır.

Figure 16 - IP Konfigürasyonu

***Graphical user interface, text, application

Description automatically generated“Key Options”*** kısmında ***“MEM\_SIZE”*** default olarak bırakılabilir. Örnek projede *10\*1024* *Byte* olarak tanımlanmıştır.

Figure 17 - Heap Memory Size Konfigürasyonu

Kalan ayarlar default olarak bırakılmıştır.

1. **Yazılım Konfigürasyonu**
   1. **TCP/IP Server**

Örnek proje dosyaları içerisinde ***“tcpServerRAW.c”*** ve ***“tcpServerRAW.h”*** dosyaları bulunmaktadır. Bu dosyalar TCP ethernet server için kullanılacak olan temel fonksiyonları içermektedir. Fonksiyonlar ST tarafından yazılmış olup, dosyalar ControllersTech tarafından düzenlenmiştir. Koda istenilen bazı eklemeler yapılarak örnek projedeki son halini almıştır.

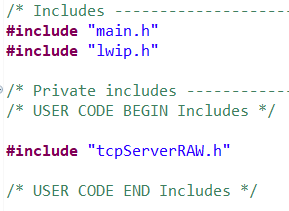
Main kodda ***“Private Includes”*** kısmında ***“tcpServerRAW.h”*** eklenmelidir.

Figure 18 - tcpServerRAW Headerının Eklenmesi

***“Private Define”*** kısmına ***“extern struct netif gnetif”*** kodu eklenmelidir. Ethernet ile ilgili konfigürasyonları barıdındıran struct yapısı bu şekilde main kodda çağrılabilir olmaktadır.

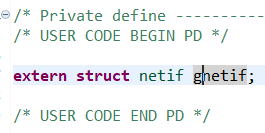


Figure 19 - netif Structının Çağrılması

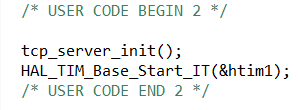
Main fonksiyonu içerisinde TCP server ve TIM1 init edilmelidir. Timerın TCP server’ın init edilmesinden sonra init edilmesi daha sağlıklı olacaktır.

Figure 20 - TCP Server ve Timer Init Edilmesi

Text

Description automatically generatedMain fonksiyonu içerisindeki while döngüsü içinde ***“ethernetif\_input(&gnetif)”*** ve ***“sys\_check\_timeouts”*** fonksiyonları çağrılmalıdır. ***“ethernetif\_input(&gnetif)”*** fonksiyonu ***“gnetif”*** ile ethernet konfigürasyonlarını girdi olarak alarak ethernet portunu dinlemeye almaktadır. ***“sys\_check\_timeouts”*** fonksiyonu ise timeout expire olduğunda timeout handlerını çağırmaktadır.

Figure 21 - While(1) döngüsü

Main kod içerisindeki düzenlemeler bu noktada tamamlanmıştır. Fonskiyonları daha iyi anlamak için ***“tcpServerRAW.c”*** dosyası incelenebilir. Giriş kısmında includelar ve yapılar tanımlanmıştır. ***“tcp\_server\_states”*** enum yapısı ethernet protokolünün mevcut durum tanımlamalarının tutulduğu yapıdır. ***“tcp\_server\_struct”*** yapısı direkt olarak mevcut bağlantı bilgilerinin tutulduğu yapıdır.

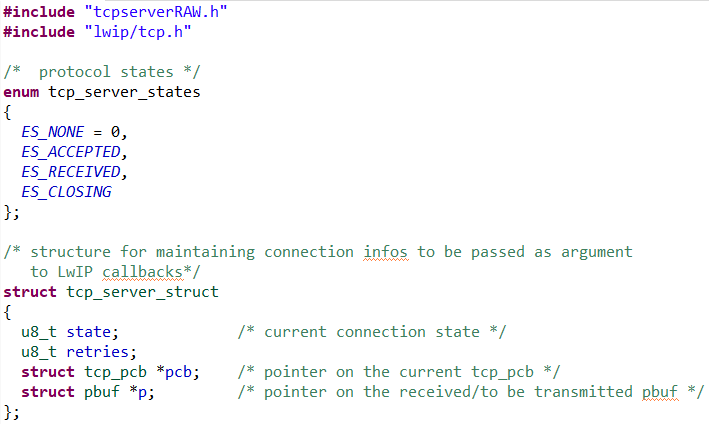


Figure 22 - tcpServerRAW.c Includelar ve Yapılar

Kodun devamında tanımlanan fonksiyonların prototipleri bulunmaktadır.

Text

Description automatically generated

Figure 23 - tcpServerRAW.c Fonksiyon Prototipleri

Timer callback fonksiyonun tanımlaması ***“tcpServerRAW.c”*** dosyası içerisinde yapılmıştır. Fonksiyon erişimleri dosya içerisinde tutulduğu için böyle bir yöntem izlenmiştir. Timer saniyede 1 callback fonksiyonuna girmektedir. Ethernet portuna veri basma callback fonksiyonu içerisinde yapılacaktır. Fonksiyonun başında 100’lük bir char array tanımlanmıştır. Sprintf kullanılarak tanımlanan char array’e string atanmıştır ve string uzunluğu ***“len”*** isimli değişkene return edilmiştir.

Counter değerinin 0 olmadığı durumlarda pBuf string uzunluğu kadar bir boyutta allocate edilmiştir. ***“pbuf\_take”*** fonksiyonu ile allocate edilen bölgeye stringi içeren buffer atanmıştır. Sonrasında ***“tcp\_server\_send”*** fonksiyon çağıralarak veri ethernet portuna gönderilmiş ve sonrasında pbuf ***“pbuf\_free”*** fonksiyonu ile pBuf serbest bırakılmıştır.

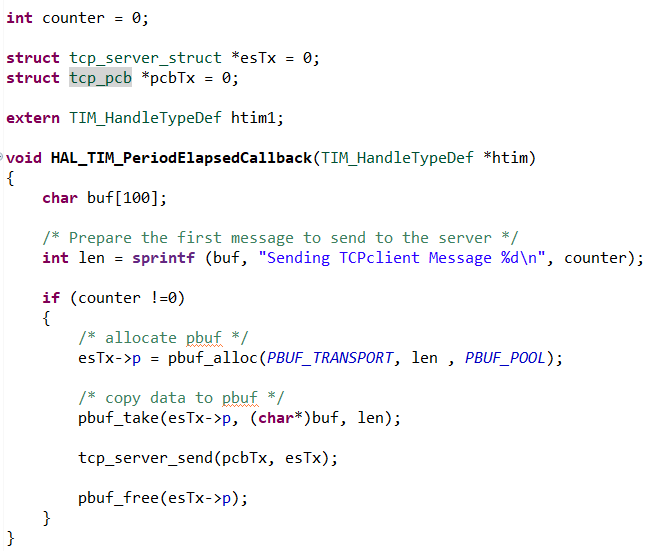


Figure 24 - TIM Callback Fonksiyonu

***“tcp\_server\_init”*** fonksiyonu incelenebilir. Fonksiyonun başlangıcında ***“tcp\_pcb”*** yapısında ***“tpcb”*** isminde bir pointer obje oluşturulmuştur. Bu obje ***“tcp\_new”*** fonksiyonu ile init edilmitşir. ***“err\_t”*** tipi error kodları saklamaktadır. Bu tipte bir değişken oluşturulmasının sebebi budur. Typedef ile tanımlanmıştır. ***“ip\_addr\_t”*** tipinde ***“myIPADDR”*** isimli bir obje tanımlanmıştır. IP adres bilgilerini bu obje tutmaktadır. Sonraki satırda ***“IP\_ADDR4”*** fonksiyonu ile istenilen IP adres değerleri objeye tanımlanmıştır. Burada verilen IP adres değeri ile ethernet konfigürasyonu kısmında ayarlanan IP değeri aynı olmalıdır. ***“tcp\_bind”*** fonksiyonu birlikte verilen port ve IP bilgileri ile bağlantı kurulmuştur. Return edilen error kodu ***“err”*** değişkenine atanmıştır. Eğer return edilen error kodu ***“ERR\_OK”*** ise pcb için TCP dinlemeye sokulmuş ve ***“tcp\_accept”*** fonksiyonu çağırılarak LwIP initialize edilmiştir. Eğer error kodu ***“ERR\_OK”*** değilse allocate edilen memory serbest bırakılmıştır.

Text

Description automatically generated

Figure 25 - tcp\_server\_init Fonksiyonu

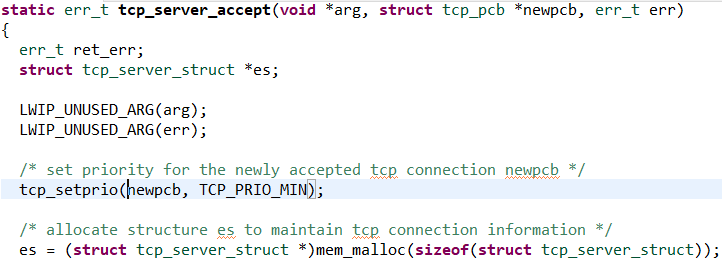
***“tcp\_server\_accept”*** fonksiyonu incelenebilir. ***“err\_t”*** tipinde, error durumunu tutan ***“ret\_err”*** objesi ve ***“tcp\_server\_struct”*** tipinde ethernet konfigürasyonlarını tutan ***“es”*** pointerı tanımlanmıştır. Kullanılmayan objelerin derleme sırasında sorun çıkarmaması adına ***“LWIP\_UNUSED\_ARG”*** fonksiyonu kullanılmıştır. Ethernet konfigürasyonlarının init edilmesi ve prioritylerin atanması için ***“tcp\_setprio”*** fonksiyonu çağrılmıştır. RAM’de ***“tcp\_server\_struct”*** kadar yerin ayrılması adına ***“mem\_malloc”*** fonksiyonu kullanılmıştır.

Figure 26 - tcp\_server\_accept Fonksiyonu Tanımlamalar

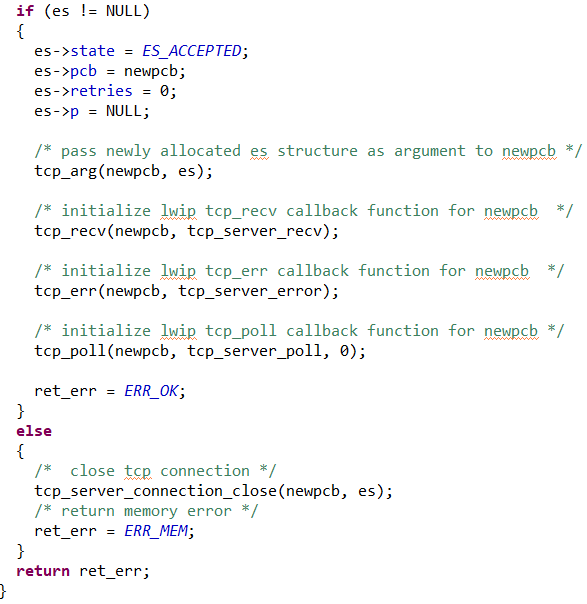
Eğer ***“es”*** objesi ***“NULL”*** değilse ve gerekli yer RAM’de ayrılabildiyse kod if bloğunun içine girer. Diğer durumda else bloğunun içine girecektir. If bloğunun içine girmesi durumunda ***“es”*** objesine gerekli atamalar yapılır ve sırasıyla ***“tcp\_arg”*,** ***“tcp\_recv”***, ***“tcp\_err”*** ve ***“tcp\_poll”*** fonksiyonları çağrılır. Fonksiyonların amaçlarının ne olduğu raporun ilgili yerlerinde anlatılmıştır. Bloğun sonunda herhangi bir sorunla karşılaşılmadığına dair ***“ret\_err”*** değişkenine atama yapılır. Else bloğuna girilmesi durumunda tcp bağlantısı ***“tcp\_server\_connection\_close”*** fonksiyonu çağrılarak sonlandırılır. Son olarak ***“ret\_err”*** değişkenine ***“ERR\_MEM”*** ataması yapılır. Bunun anlamı memory allocation konusunda bir sıkıntı olduğu ve bağlantının sonlandırıldığıdır.

Figure 27 - tcp\_server\_accept Fonksiyonu If Else Bloğu

***“tcp\_server\_recv”*** fonksiyonu incelenebilir. Önceki fonksiyonlarda olduğu gibi ***“es”*** ve ***“ret\_err”*** objeleri oluşturulur. Eğer ***“arg != NULL”*** ise ***“LWIP\_ASSERT”*** fonksiyonu kullanılarak ***“arg != NULL”*** yazısı ***“printf”*** ile yazdırılır.

Text

Description automatically generated

Figure 28 - tcp\_server\_recv Fonksiyonu Tanımlamalar

Yorum satırında da bahsedildiği gibi if bloğu incelendiğinde, eğer ***“p”*** pointer’ı ***“NULL”*** ise boş frame alınmıştır. Bu durumda tcp durumu ***“ES\_CLOSING”*** yapılır. Eğer ***“es -> p”*** değeri de ***“NULL”*** ise ***“tcp\_server\_connection\_close”*** fonksiyonu çağrılarak TCP bağlantısı sonlandırılır. Diğer durumda paket alınmış demektir ve kalan paket gönderilir. ***“ret\_err”*** objesine ***“ERR\_OK”*** ataması yapılır.

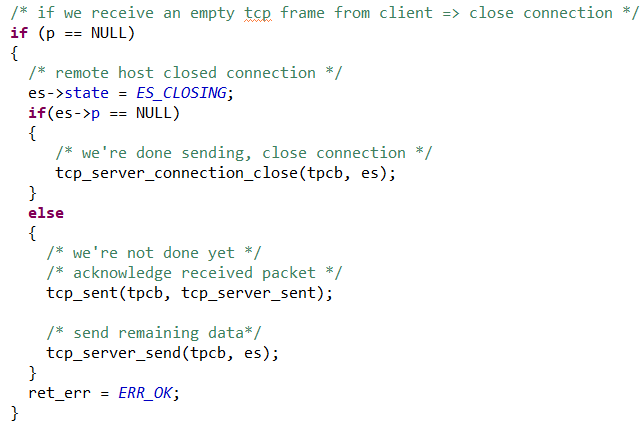


Figure 29 - tcp\_server\_recv Fonksiyonu If Bloğu

Yorum satırında da bahsedildiği gibi eğer gelen frame boş değilse ama bir şekilde ***“err”*** ***“ERR\_OK”*** olarak gelmediyse bu bloğa girilir. ***“p” “NULL”*** değilse if bloğuna girer ve ***“NULL”*** ataması ile birlikte ***“pbuf”*** serbest bırakılır. **“err”** objesinin mevcut durumu **“ret\_err”** objesine de atanır.

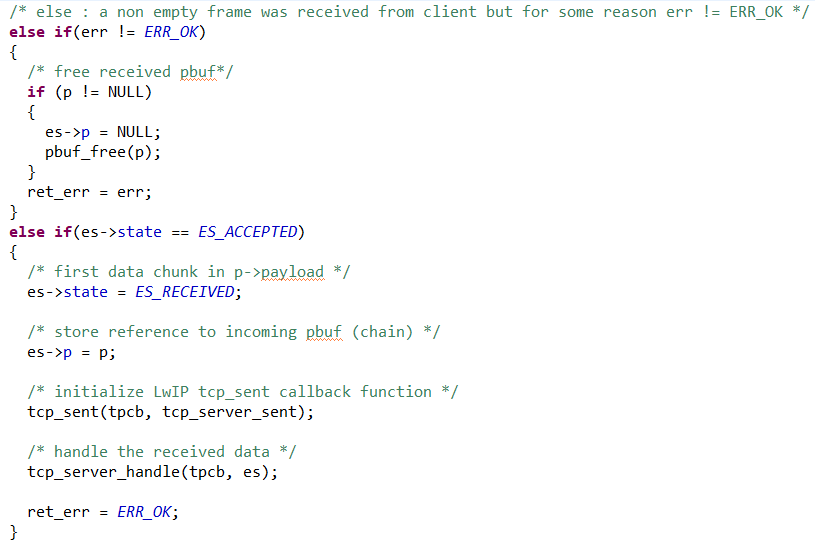
***“es”*** objesinin mevcut durumu ***“ES\_ACCEPTED”*** olduğunda data alınmıştır. ***“es”*** objesinin durumu ***“ES\_RECEIVED”*** olarak güncellenir. ***“p”*** objesine gelen veri ***“es -> p”*** objesine atanır. ***“tcp\_set”*** ve ***“tcp\_server\_handle”*** fonksiyonları sırası ile çağrılır. ***“ret\_err”*** objesi ***“ERR\_OK”*** olarak güncellenir.

Figure 30 - tcp\_server\_recv Fonksiyonu ERR\_OK ve ES\_ACCEPTED Blokları

***“ES\_RECEIVED”*** bloğu incelenebilir. Yorum satırında da anlatıldığı gibi eğer client tarafından data gelmeye devam ediyor ve önceki data gönderilmişse if bloğuna girer. Bu durumda ***“es -> p”*** objesine ***“p”*** ataması yapılır. Yani mevcut veri struct yapısına alınır ve ***“tcp\_server\_handle”*** fonksiyonu çağrılır. Diğer durumda **“pbuf”** tipinde bir pointer tanımlanır ve bu pointera ***“es -> p”*** atanır. ***“pbuf\_chain”*** fonksiyonu kullanılarak gelen veri ***“pbuf”*** zincirine eklenir. Son durumda ***“ret\_err”*** objesine ***“ERR\_OK”*** ataması yapılır.

**Graphical user interface, text, application

Description automatically generated*“ES\_CLOSING”*** bloğu incelenebilir. ***“tcp\_recved”*** fonksiyonu çağrılarak daha büyük bir frame ihtiyacı olduğu belirlenir. ***“es -> p”*** objesine ***“NULL”*** ataması yapılır ve ***“p”*** objesi serbest bırakılır. Son olarak ***“ret\_err”*** objesine ***“ERR\_OK”***ataması yapılır.

Figure 31 - tcp\_server\_recv Fonksiyonu ES\_RECEIVED ve ES\_CLOSING Blokları

Text

Description automatically generatedSon olarak else bloğunda ***“es -> state”*** bilinmemektedir. Bu yüzden ***“tcp\_recved”*** fonksiyonu daha önce olduğu gibi çağrılır. ***“es -> p”*** objesine ***“NULL”*** ataması yapılır ve ***“p”*** objesi ***“pbuf\_free”*** fonksiyonu ile serbest bırakılır. Son olarak ***“ret\_err”*** objesine ***“ERR\_OK”*** ataması yapılır. Fonksiyonun sonunda ***“ret\_err”*** değeri return edilir.

Figure 32 - tcp\_server\_recv Fonksiyonu Else Bloğu

Text

Description automatically generated ***“tcp\_server\_error”*** fonksiyonu incelenebilir. Önceki fonksiyonlarda olduğu gibi ***“es”*** objesi oluşturulmaktadır. Derleme sırasında sorun olmaması için kullanılmayan argumanlar ***“LWIP\_UNUSED\_ARG”*** fonkisyonu ile gönderilmiştir. ***“tcp\_server\_error”*** fonksiyonuna verilen ***“arg”***, ***“es”*** objesine atanmıştır. Eğer ***“es != NULL”*** ise if bloğuna girilmektedir. Bu durumda memory serbest bırakılmaktadır.

Figure 33 - tcp\_server\_error Fonksiyonu

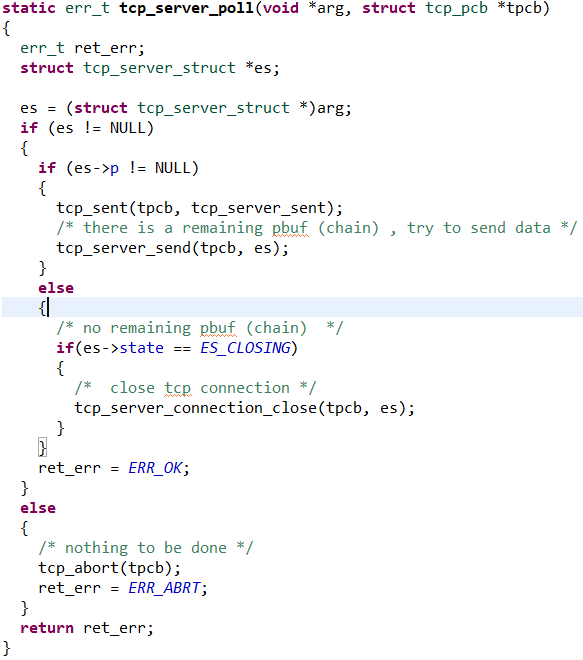
***“tcp\_server\_poll”*** fonksiyonu incelenebilir. Daha önceki fonksiyonlarda olduğu gibi ***“ret\_err”*** ve ***“es”*** objeleri tanımlanmıştır. Fonkisyona gönderilen ***“arg”*** cast edilerek ***“es”*** objesine atanmıştır. Eğer ***“es”*** objesi boş değilse if bloğuna girilir. Eğer ***“es -> p”*** objesi boş değilse gönderilecek veri vardır. ***“tcp\_sent”*** ve ***“tcp\_server\_send”*** fonksiyonları sırası ile çağrılır. Eğer **“es -> p” “NULL”** ise ve ***“es -> state”*** objesine mevcut durumda ***“ES\_CLOSING”*** atanmışsa daha fazla ***“pbuf”*** zinciri kalmamıştır ve ***“tcp\_server\_connection\_close”*** fonksiyonu çağrılır. Son olarak ***“ret\_err”*** objesine ***“ERR\_OK”*** ataması yapılır. Eğer ***“es”*** objesi ***“NULL”*** ise, yapılacak bir şey kalmamıştır ve ***“tcp\_abort”*** fonksiyonu çağrılır. Son olarak ***“ret\_err”*** objesine ***“ERR\_ABRT”*** ataması yapılır ve ***“ret\_err”*** return edilir.

Figure 34 - tcp\_server\_poll Fonksiyonu

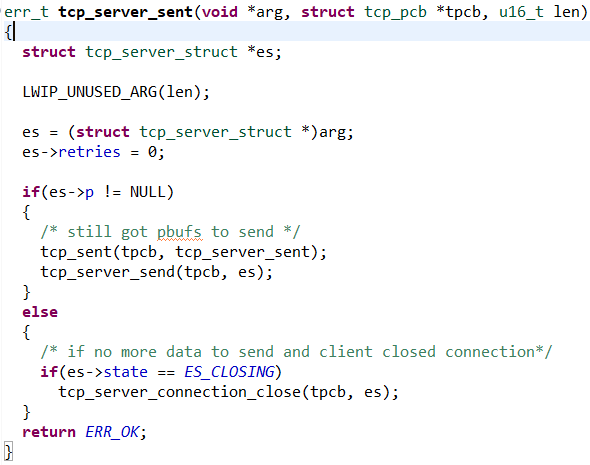
***“tcp\_server\_sent”*** fonksiyonu incelenebilir. Önceki fonksiyonlarda olduğu gibi ***“es”*** objesi tanımlanmaktadır. Derleme sırasında sorun olmaması için kullanılmayan ***“len”*** değişkeni ***“LWIP\_UNUSED\_ARG”*** fonksiyonuna gönderilmiştir. Fonksiyona gönderilen ***“arg”*** objesi ***“es”*** objesine atanmıştır. Ayrıca ***“es -> retries”*** objesi de sıfırlanmıştır. Eğer ***“es -> p”*** objesi ***“NULL”*** değilse if bloğuna girmektedir. Bu gönderilecek ***“pbuf”*** olduğunu belirtmektedir. Sırası ile ***“tcp\_sent”*** ve ***“tcp\_server\_send”*** fonksiyonları çağırılmıştır. Eğer değilse ve ***“es -> state”*** objesi ***“ES\_CLOSING”*** olarak atalıysa ***“tcp\_server\_connection\_close”*** fonksiyonu çağrılarak bağlantı sonlandırılır. Son olarak ***“ERR\_OK”*** değeri return edilir.

Figure 35 - tcp\_server\_sent Fonksiyonu

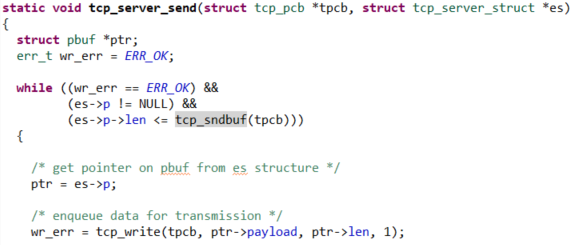
***“tcp\_server\_send”*** fonksiyonu incelenebilir. Başlangıç olarak ***“pbuf”*** tipinde ***“ptr”*** isminde bir pointer struct tanımlanmıştır. Aynı zamanda ***“err\_t”*** tipinde ***“wr\_err”*** isminde bir değişken tanımlanmış ve ***“ERR\_OK”*** atanmıştır. ***“tcp\_sndbuf”*** fonksiyonu uygun olan buffer size’ını döndürmektedir. Kod while döngüsüne sokulmuştur. Koşul olarak, ***“wr\_err”*** ***“ERR\_OK”*** olduğu, ***“es -> p” “NULL”*** olmadığı ve ***“es -> p -> len”*** objesi uygun olan buffer size’ından küçük olduğu sürece while dönmektedir. ***“es -> p”*** pointer’ı başlangıçta tanımlanan ***“ptr”*** pointerına atanmıştır. Mevcut data ***“tcp\_write”*** ile yazıldıktan sonra return değeri ***“wr\_err”*** değişkenine atanmıştır.

Figure 36 - tcp\_server\_send Fonksiyonu

Graphical user interface, text, application

Description automatically generated***“wr\_err”*** değişkeni ***“ERR\_OK”*** olduğu durumda if bloğuna girilmiştir. ***“plen”*** ve ***“freed”*** değişkenleri tanımlandıktan sonra, ***“plen”*** değişkenine ***“ptr -> len”*** ataması yapılmıştır. ***“es -> p”*** pointerına ***“ptr -> next”*** ataması yapılarak sıradaki ***“pbuf”*** atanır. Eğer ***“es -> p” “NULL”*** değilse ***“pbuf\_ref”*** fonksiyonu çağrılarak referans counterı artırılır. Do while bloğunun do bloğu içerisinde ***“pbuf\_free”*** fonksiyonu çağırılarak, return değeri ***“freed”*** değişkenine atanmıştır. ***“freed”*** değişkeni 0’a eşit olana kadar kod while içerisinde boş dönmüştür. Son durumda daha fazla data okunulabilir olmuştur ve ***“tcp\_recved”*** fonksiyonu çağırılmıştır.

Figure 37 - tcp\_server\_send Fonksiyonu If Bloğu

Eğer ***“wr\_err”*** değişkeni ***“ERR\_MEM”*** değerine eşitse else if bloğuna girer ve ***“ptr”*** pointerı ***“es -> p”*** pointerına atanır. Bu durumda memory azalmıştır ve sonra denemek üzere devam edilir. Farklı problemlerde yapılacakların yazılması için else bloğu boş bırakılmıştır.

Graphical user interface, text

Description automatically generated with medium confidence

Figure 38 - tcp\_server\_send Fonksiyonu Else If ve Else Bloğu

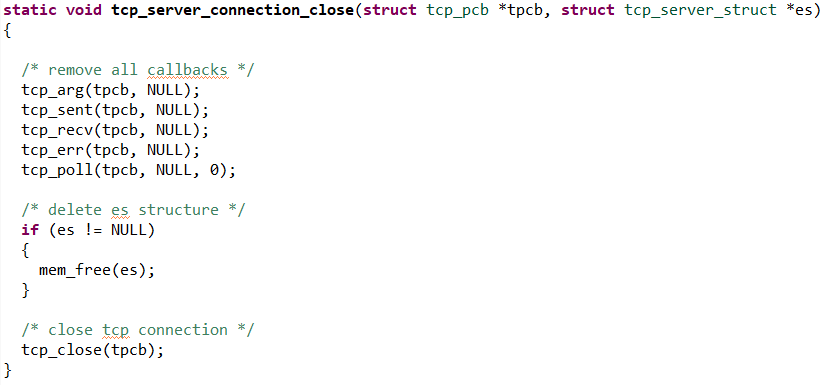
***“tcp\_server\_connection\_close”*** fonksiyonu incelenebilir. Sırasıyla ***“tcp\_arg”***, ***“tcp\_sent”***, ***“tcp\_recv”***, ***“tcp\_err”*** ve ***“tcp\_poll”*** fonksiyonları çağırılarak ***“tpcb”*** objesi boşaltılır. Fonksiyonlar çağırılırken argüman olarak ***“tpcb”*** ve ***“NULL”*** değeleri verilmiştir. ***“es”*** structure yapısının da memoryde tuttuğu alan serbest bırakılır. Son olarak TCP bağlantısı ***“tcp\_close”*** fonksiyonu kullanılarak sonlandırılır.

Figure 39 - tcp\_server\_connection\_close Fonksiyonu

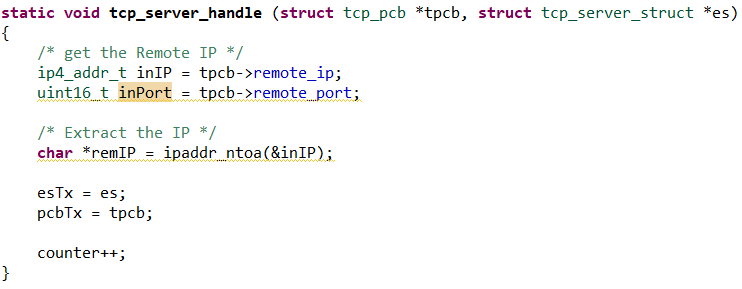
***“tcp\_server\_handle”*** fonksiyonu incelenebilir. Client cihazının IP adresi ve port bilgisi, tanımlanan ***“inIP”*** ve ***“inPort”*** değişkenlerine atanmıştır lakin bu değişkenler kullanılmayacaktır. Aynı şekilde ***“ipaddr\_ntoa”*** fonksiyonu kullanılarak char pointer olan ***“remIP”*** pointerına IP bilgisi çıkartılmıştır. Aynı şekilde bu değişken de kullanılmamıştır. Mevcut olan ***“es”*** structure içindeki bilgiler ve ethernet konfigürasyonu bilgilerini saklayan ***“tpcb”***, ***“esTx”*** ve ***“pcbTx”*** yapılarına çıkarılmıştır. Bu değişkenler ***“HAL\_TIM\_PeriodElapsedCallback”*** fonksiyonunda kullanılmıştır. Son olarak ***“counter”*** değeri artırılmıştır. Bu durum, client tarafından her veri geldiğinde ***“counter”*** değeri artırılacaktır anlamına gelmektedir.

Figure 40 - tcp\_server\_handle Fonksiyonu

* 1. **TCP/IP Client**

Örnek proje dosyaları içerisinde ***“tcpClientRAW.c”*** ve ***“tcpClientRAW.h”*** dosyaları bulunmaktadır. Bu dosyalar TCP ethernet client için kullanılacak olan temel fonksiyonları içermektedir. Fonksiyonlar ST tarafından yazılmış olup, dosyalar ControllersTech tarafından düzenlenmiştir. Koda istenilen bazı eklemeler yapılarak örnek projedeki son halini almıştır.

***“TCP/IP Server”*** kısmında yazılım konfigürasyonu konusunda çok büyük bir kısmı benzerdir. ***Bu kısımda sadece farklı olan kısımlar anlatılacaktır.*** Eğer client bağlantısı yapılmak isteniyorsa önce ***“TCP/IP Server”*** kısmı incelenip sonra bu kısıma geçilebilir.

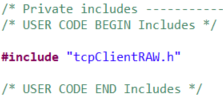
Main kodda ***“Private Includes”*** kısmında ***“tcpClientRAW.h”*** eklenmelidir.

Figure 41 - tcpClientRaw Headerın Eklenmesi

Main fonksiyonu içerisinde TCP client ve TIM1 init edilmelidir. Timerın TCP client’ın init edilmesinden sonra init edilmesi daha sağlıklı olacaktır.

Text

Description automatically generated

Figure 42 - TCP Server ve Timer Init Edilmesi

Main kod içerisindeki düzenlemeler bu noktada tamamlanmıştır. Fonskiyonları daha iyi anlamak için ***“tcpClientRAW.c”*** dosyası incelenebilir. “tcpClientRAW.c” dosyası ile “tcpServerRAW.c” dosyaları birbirine neredeyse tamamen benzemektedir. ***O yüzden bu kısımda “tcpClientRAW.c” dosyası incelenmeyecektir***. Eğer client bağlantısı yapılmak isteniyorsa önce ***“TCP/IP Server”*** sonra devam edilebilir.

1. **Ethernet Bağlantısı**
   1. **TCP/IP Server**

***“tcp\_server\_init”*** fonksiyonu içerisinde server cihazının IP adresi ve portu belirlenmişti. Raporun önceki kısımlarında görülebilir. ***“ioc”*** dosyası içerisinde de aynı şekilde tanımlı ise bağlantı yapılabilir. Bağlantı için ***“Hercules”*** program kullanılacaktır. Verilen [link](https://www.hw-group.com/software/hercules-setup-utility) üzerinden uygulama indirilebilir. Program direkt olarak exe üzerinden çalışmaktadır ve kuruluma ihtiyaç yoktur. Program kurulduktan sonra ***“TCP Client”*** sekmesi açılır. Bu sekmede program TCP bağlantısında client gibi davranacaktır. ***“TCP”*** isimli bloğun içerisinde ***“Module IP”*** kısmına server cihazının IP adresi yazılır. ***“Port”*** kısmına haberleşilen port bilgisi yazılır.

Graphical user interface, application

Description automatically generated

Figure 43 - Hercules TCP Client Konfigürasyonu

Ethernet için kablo bağlantısı yapıldıysa ve NUCLEO kartına program yüklenip, karta güç verildiyse bağlantı hazır demektir. ***“TCP”*** isimli bloğun altındaki ***“Ping”*** tuşuna basılırsa kart pinglenir. Eğer bağlantı başarılı ise ekranın ***“Received/Sent data”*** kısmında ***“Received ICMP ECHO REPLY”*** yazısının görülmesi gerekmektedir. Eğer ping başarılı ise ekrandaki ***“TCP”*** isimli bloğun altındaki ***“Connect”*** tuşuna basılarak bağlantı kurulabilir. Eğer bağlantı başarılı oldu ise ekranda ***“Connected to 192.168.0.111”*** yazısı görülmelidir. Yazı içindeki IP adresi server IP adresidir. Eğer server IP adresi farklı bir şekilde tanımlandıysa yazı içerisinde o adres görülecektir. Aynı zamanda **“Module IP”** kısmına, o IP adresi girilmedilir. Aksi takdirde zaten bağlantı başarılı olmayacaktır.

Graphical user interface, text, application

Description automatically generated

Figure 44 - TCP Server ve Client Arasındaki Bağlantının Başarılı Olması Durumu

Graphical user interface, text

Description automatically generatedBağlantı başarılı bir şekilde kurulduysa, artık veri gönderilip alınabilir. NUCLEO kartına yüklenen mevcut yazılım konfigürasyonunda, server tarafından veri alınabilmesi için önce client tarafından veri gönderilmelidir. Herhangi, rastgele bir veri gönderilenebilir.

Figure 45 - Server ve Client Tarafından Gönderilen Veriler

Figür 43’te görülen ***“Received/Sent data”*** ekranın altındaki mesajlarda pembe renkli olan mesajlar client tarafından yani Hercules üzerinden bizim gönderdiğimiz mesajlar, siyah renkli olanlar ise server tarafından client’a gönderilen mesajlardır. Görüldüğü üzere client tarafından ilk veri gönderildikten sonra server 1 saniye aralıklarla veri göndermeye başlamıştır. Bir saniye aralıklarla olmasının sebebi ***“Clock Konfigürasyonu”*** kısmında anlatıldığı gibi timer süresinin 1 saniyeye göre ayarlanmış olmasıdır. Client tarafından her veri geldiğinde, server tarafından gönderilen mesajın sonundaki counter değeri artmaktadır. Bir nevi client tarafından gelen mesajları sayan bir system gibi çalışmaktadır kod.

* 1. **TCP/IP Client**

***“tcp\_client\_init”*** fonksiyonu içerisinde client cihazının IP adresi ve portu belirlenmişti. Raporun önceki kısımlarında görülebilir. ***“ioc”*** dosyası içerisinde de aynı şekilde tanımlı ise bağlantı yapılabilir. Bağlantı için ***“Hercules”*** program kullanılacaktır. Verilen [link](https://www.hw-group.com/software/hercules-setup-utility) üzerinden uygulama indirilebilir. Program direkt olarak exe üzerinden çalışmaktadır ve kuruluma ihtiyaç yoktur. Program kurulduktan sonra ***“TCP Server”*** sekmesi açılır. Bu sekmede program TCP bağlantısında server gibi davranacaktır. ***“Server Status”*** isimli bloğun içerisinde ***“Port”*** kısmına haberleşme için kullanılan ethernet portu yazılır.

Graphical user interface, application

Description automatically generated

Figure 46 - Hercules Programında TCP Server Konfigürasyonu

Graphical user interface, text, application

Description automatically generatedEğer port değeri doğru girildiyse ***“Server Status”*** bloğunun altındaki ***“Listen”*** tuşuna basıldığı durumda, ***“Client connection status”*** bloğunun altındaki ekranda ***“Client Connected”*** yazısı görülecektir. Bu NUCLEO kartının Hercules program üzerinde oluşturulan TCP Server’a bağlandığını göstermektedir. Bağlandıktan hemen sonra mevcut yazılım konfigürasyonuna göre ***“Received data”*** kısmında client tarafından gönderilen veriler gözükecektir. Server yazılımında yapılan konfigürasyonda olduğu gibi client yazılımında da karşı taraftan, burada karşı taraf server oluyor, veri geldikçe client tarafından gelen verinin sonundaki counter artacaktır. Figür 47’de görüldüğü gibi ***“Sent data”*** kısmında iki kez ***“30”*** verisi gönderilmiştir. ***“Received data”*** kısmında mesajın sonunda counter değerinin 2 kez arttığı görülebilir.

Figure 47 - TCP Server ve Client Tarafından Gönderilen Veriler

Client dinleme konusunda client’a bağlanırken sorun yaşanabilir. Bu durumda ***“Listen”*** tuşuna basıldıktan sonra NUCLEO kartına reset atılabilir.