T.C.

TRAKYA ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

ATMEL AVR MİKRODENETLEYİCİLERİ İÇİN YENİ BİR ÖNYÜKLEYİCİ

ERCAN ERSOY

YÜKSEK LİSANS TEZİ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

Tez Danışmanı: DOÇ. DR. GÜRKAN TUNA

EDİRNE-2018

ERCAN ERSOY'un hazırladığı "ATMEL AVR MİKRODENETLEYİCİLERİ İÇİN YENİ BİR ÖNYÜKLEYİCİ" başlıklı bu tez, tarafımızca okunmuş, kapsam ve niteliği açısından BİLGİSAYAR MÜHENDİSLİĞİ Anabilim Dalında bir Yüksek lisans tezi olarak kabul edilmiştir.

Jüri Üyeleri (Ünvan, Ad, Soyad): İmza

Doç. Dr. Gürkan TUNA

Doç. Dr. İlhan UMUT

Dr. Öğr. Üyesi Murat Olcay ÖZCAN

Tez Savunma Tarihi: 14/05/2018

Bu tezin Yüksek Lisans tezi olarak gerekli şartları sağladığını onaylarım.

İmza

Doç. Dr. Gürkan TUNA
Tez Danışmanı

Trakya Üniversitesi Fen Bilimleri Enstitüsü onayı

Prof. Dr. Murat YURTCAN
Fen Bilimleri Enstitüsü Müdürü

T.Ü.FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ YÜKSEK LİSANS PROGRAMI
DOĞRULUK BEYANI

Trakya Üniversitesi Fen Bilimleri Enstitüsü, tez yazım kurallarına uygun olarak hazırladığım bu tez çalışmasında, tüm verilerin bilimsel ve akademik kurallar çerçevesinde elde edildiğini, kullanılan verilerde tahrifat yapılmadığını, tezin akademik ve etik kurallara uygun olarak yazıldığını, kullanılan tüm literatür bilgilerinin bilimsel normlara uygun bir şekilde kaynak gösterilerek ilgili tezde yer aldığını ve bu tezin tamamı ya da herhangi bir bölümünün daha önceden Trakya Üniversitesi ya da farklı bir üniversitede tez çalışması olarak sunulmadığını beyan ederim.

14 / 05 / 2018
Ercan ERSOY
İmza

Yüksek Lisans Tezi
Atmel AVR Mikrodenetleyicileri İçin Yeni Bir Önyükleyici
T.Ü. Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği

ÖZET

Mikrodenetleyiciler için tasarlanmış olan önyükleyicilerin çoğunda bazı yetersizlikleri olan uygulama programı yükleme yöntemleri kullanılır. Ayrıca, mevcut önyükleyicilerin çoğu önyükleme sırasında uygulama programı yükleme isteğinin olup olmadığını denetlemek için bir süre bekler ve hızlı açılması istenilen sistemlerde zaman kaybına neden olmaktadır. Bu sorunların üstesinden gelmek için bu tez çalışmasında, ATmega328P mikrodenetleyicileri için "Micro Boot" adlı yeni bir önyükleyici tasarlanmıştır. Micro Boot önyükleyicisi, mikrodenetleyiciye bağlı bir Ethernet denetleyicisi ve aynı bilgisayar ağına bağlı başka bir bilgisayarda çalışan tarayıcı üzerinden bir uygulama programı yükler. Kesme kullanılarak Micro Boot önyükleyicisinin önyükleme süresi kısaltılmıştır. Micro Boot önyükleyicisinin çalışmasını sınamak amacıyla bir uygulama programı geliştirilmiştir. Uygulama programı bu önyükleyici kullanılarak bir Atmega328P mikrodenetleyicisi barındıran ve üzerinde Wiznet 5100 Ethernet denetleyicili bir kalkan olan bir Arduino Uno geliştirme ortamına aktarılmıştır. Uygulama programının ve önyükleyicinin kararlı olarak çalıştığı gözlemlenmiştir.

Yıl : 2018

Sayfa sayısı : 81

Anahtar Kelimeler : Gömülü Sistem, Mikrodenetleyici, Önyükleyici, Bellenim, Önyükleme, Atmel AVR, Arduino, Ethernet, Wiznet 5100, HTTP

Master's Thesis
A New Bootloader for Atmel AVR Microcontrollers
Trakya University Institute of Natural Sciences
Computer Engineering Department

ABSTRACT

Bootloaders designed for microcontrollers use application program installation methods which have some inadequacies in the majority. Also, most existing bootloaders wait a while to check if there is a request to load the application program during boot, and cause time loss in systems that require fast boot-up. To overcome these problems, in this thesis study, a new bootloader named "Micro Boot" was designed for the ATmega328P microcontrollers. The micro boot preloader loads an application program through a microcontroller connected to an Ethernet controller and a browser running on another computer connected to the same computer network. The boot time of the Micro Boot bootloader has been shortened using interrupt. An application program has been developed to test the operation of the Micro Boot bootloader. Using this bootloader, the application program has been transferred to an Arduino Uno development environment, which houses an Atmega328P microcontroller and has a Wiznet 5100 Ethernet supervisory shield on it. It has been observed that the application program and the bootloader operate stably.

Year : 2018

Number of Pages : 81

Keywords : Embedded System, Microcontroller, Bootloader, Firmware, Booting, Atmel AVR, Arduino, Ethernet, Wiznet 5100, HTTP

TEŞEKKÜR

Çalışmalarımda yardımcı olan Doç Dr. Gürkan TUNA'ya ve Dr. Öğr. Üyesi Süleyman Bahadır KARUV'a teşekkür ederim.

İÇİNDEKİLER

SİMGELER VE KISALTMALAR DİZİNİ

AC	Analog Comparators
ADC	Analog to Digital Converter
ALU	Arithmetic Logic Unit
ARM	Advanced RISC Machine
ARP	Address Resolution Protocol
AVR	Advanced Virtual RISC
C	Santigrat
CPU	Central Processing Unit
DAC	Digital to Analog Converter
EPROM	Erasable Read-Only Memory
EEPROM	Electirically Erasable Read-Only Memory
Gpbs	Gigabit Per Second
GPIO	General Purpose Input/Output
HTTP	Hypertext Transfer Protocol
Hz	Hertz
I^2C	Inter-Integrated Circuit
ICMP	Internet Control Message Protocol
ICSP	In-Circuit Serial Programming
IEEE	Institute of Electrical and Electronics Engineers
IGMP	Internet Group Management Protocol
IP	Internet Protocol
IPv4	Internet Protocol Version 4
KB	Kilobayt
LED	Light-Emitting Diode
mA	Miliamper

Mbps	Megabit Per Second
MLF	Micro Lead Frame
ms	Milisaniye
PDIP	Plastic Dual-In-line Package
PIC	Peripheral Interface Controller
PPPoE	Point-to-Point Protocol Over Ethernet
PWM	Pulse Width Modulation
QFN	Quad Flat No-lead package
RAM	Random Access Memory
ROM	Read-Only Memory
RISC	Reduced Instruction Set Computer
SD	Secure Digital
SPI	Serial Peripheral Interface
SRAM	Static Random Access Memory
TCP	Transmission Control Protocol
TFTP	Trivial File Transfer Protocol
TQFP	Thin Quad Flat Pack
UART	Universal Asynchronous Receiver/Transmitter
USART	Universal Synchronous/Asynchronous Receiver/Transmitter
UDP	User Datagram Protocol
USB	Universal Serial Bus
V	Volt
µA	Mikroamper

ÇİZELGELER VE ŞEKİLLER DİZİNİ

BÖLÜM 1

GİRİŞ

Önyükleyicilerin gömülü sistemlerde geniş kullanım alanı vardır (Kraleing ve McKay, 2013). Önyükleyiciler, kullanımları zorunlu olmamakla birlikte gömülü sistem olarak mikrodenetleyicilerde uygulama programı yüklemelerinde kullanılırlar. Mikrodenetleyicilerde kullanılan önyükleyiciler, mikrodenetleyicilere çeşitli yollarla uygulama programı yükler (Siegesmund, 2014).

Mikrodenetleyicilerde önyükleyici kullanılarak uygulama programı yükleme mikrodenetleyiciler için uygulama programı geliştirme ve mikrodenetleyiciler için bellenim güncelleme için önemlidir. Önyükleyici, mikrodenetleyicilerde özel programlayıcı devreler gerektirmeden mikrodenetleyiciye uygulama programı yükleme olanağı sağlamaktadır (Siegesmund, 2014).

Bir işlemci mimarisi olan Atmel AVR mimarisinin 8 bitlik türü için yazılmış mevcut önyükleyicilerin son kullanıcılar için kullanımı zor olabilir. Bundan dolayı son kullanıcının güncelleme işlemini kolaylıkla yapabileceği yeni bir önyükleyici tasarımı gerekir. Bu çalışmada, 8 bitlik Atmel AVR mikrodenetleyicilerinden ATmega328P mikrodenetleyicileri için bir önyükleyici tasarlanmıştır. Bu önyükleyici, bu mikrodenetleyiciler için yazılmış çoğu önyükleyiciden farklı olarak uygulama programını bir bilgisayar ağına bağlanan bir Ethernet denetleyicisi üzerinden ve bu ağa bağlı başka bir bilgisayar üzerinde çalışan bir ağ tarayıcısı yardımıyla HTTP kullanılarak yüklemektedir. Ağ tarayıcısı üzerinden uygulama programı yükleme yöntemi son kullanıcı için oldukça uygun bir yöntemdir. Ayrıca, bu önyükleyici, mikrodenetleyicinin bir GPIO bacağını kullanarak uygulama programı yükleme isteğini

algılar. Bu durum, mikrodenenetleyicinin açılış sırasında uygulama programı yükleme isteğinin algılanması için beklenilmesine gerek yoktur. Tasarlanan önyükleyicinin çalışmasını sınamak için bir uygulama programı yazılmıştır. Bu uygulama programı Atmega328P mikrodenetleyicisi barındıran Arduino Uno geliştirme ortamına bir programlayıcı kullanılarak önyükleyici ve bu önyükleyici kullanılarak uygulama programı yüklenmiştir. Tasarlanan önyükleyicinin ve uygulama programının kararlı olarak çalıştığı gözlemlenmiştir.

BÖLÜM 2

GENEL BİLGİLER

Bu bölümde tez çalışmasına temel teşkil eden ve/veya tez çalışmasında kullanılan teknolojilere ilişkin bilgiler sunulmaktadır.

2.1. Gömülü Sistem

Gömülü sistem, belirli bir işi yerine getirmek için tasarlanmış bir sistem olarak tanımlanabilir. Gömülü sistemler, birçok amaç için kullanılır. Bu amaçlara örnek olarak tüketici elektroniği, sanayi sistemleri, sağlık sistemleri ve askeri sistemler verilebilir (Fan, 2015). Gömülü sistemlerin tasarım amaçlarına göre birçok türü vardır (Heath, 2002; Parab vd., 2008; Toulson ve Wilmshurst, 2017). Gömülü sistemler, belli amaçlar için girdi olarak alınan verileri bu amaçlara göre çıktı olarak verilen verilere dönüştürür. Gömülü sistemlerin bazılarında kullanıcı arayüzü vardır. Kullanıcı arayüzü, gömülü sistemin hem girdi ve hem çıktı bileşenlerindendir ve bu arayüz gömülü sistem denetimini sağlar. Ayrıca, bir gömülü sistem, diğer sistemlere de bağlantı kurabilir (Toulson ve Wilmshurst, 2017).

Gömülü sistemlerde, diğer bilgisayar türlerinde olduğu gibi temel bilgisayar bileşenleri bulunur. Bu bileşenler, CPU, bellek, girdi bileşenleri ve çıktı bileşenleridir. Bellek olarak geçici bellek ve kalıcı bellek bulunur. Gömülü sistemlerde bulunan kalıcı bellek çoğunlukla ROM ya da flaş bellektir (Fan, 2015). Gömülü sistemlerde, kullanıcı arayüzü bileşenlerine örnek olarak tuş takımı, anahtarlar, ekran verilebilir (Jiménez vd., 2014). Gömülü sistemlerde ortam durumunu algılamak için sensörler bulunur (Jiménez vd., 2014). Gömülü sistemlerde, diğer sistemlerle iletişim kurmak için seri ve paralel iletişim birimleri bulunabilir (Jiménez vd., 2014). Ayrıca, gömülü sistemlerde, analog

verileri girdi olarak almak için ADC, analog verileri çıktı olarak almak için DAC bulunur (Jiménez vd., 2014).

Gömülü sistemlerde yazılım olarak bir uygulama programı bulunur (Sloss vd., 2014; Fan, 2015). Bu uygulama programına bellenim denir (Sloss vd., 2014). Gömülü sistemler için yazılımsız çözüm geliştirilebilir. Ancak, yazılımsız çözümler genellikle pahalıdır ve gömülü sistemin güncellenmesi için yeni donanım tasarımı gerekir. Ayrıca, gömülü sistemlerde yazılım kullanmak, gömülü sistemin hızlıca geliştirilmesini ve hızlıca güncellenmesini sağlar (Fan, 2015). Gömülü sistemlerde, açılış sırasında bellenimin çalışmasını sağlayacak bazı işlemler için bir önyükleyici yazılımı bulunabilir (Fan, 2015).

Gömülü sistemleri programlamak için genellikle Assembly, C, C++, Java ve Python programlama dilleri kullanılır. C programlama dili, birçok uygulamada öne çıkmaktadır (Lipovski, 2004). Gömülü sistemleri programlamak için gömülü sistem dışında olan bilgisayarlar kullanılması gereklidir. Programın yazılması ve programın derlenmesi gömülü sistem dışında olan bilgisayarlarda gerçekleşir. Aynı zamanda, programlama işleminin yapıldığı bilgisayarlarda derlenmiş programların sınama amaçlı ve hata ayıklama amaçlı benzetimi de yapılabilir. Derlenmiş programlar, gömülü sistemlere aktarılarak çalıştırılabilir. Gömülü sistemde sınama işlemi ve hata ayıklama işlemi de yapılabilir (Toulson ve Wilmshurst, 2017).

2.2. Bellenim

Bellenim, gömülü sistemlerde çalışan uygulama yazılımıdır ve bir kalıcı bellek tümdevresi içinde saklanır (Sloss vd., 2014). Bellenim temel sistem işlemlerinin yürütülmesini sağlar. Bir gömülü sistemde bellenim, bir işletim sistemi olabilir (Fan, 2015; Sloss vd., 2014). Mikrodenetleyicilerde, tek parça bellenim yazılımları kullanılır (Calcutt vd., 2004).

2.3. Önyükleyici

Önyükleyici, gömülü sistemlerde sistem açılışında ilk çalıştırılan yazılımdır (Fan, 2015; Sloss vd. 2014). Önyükleyiciler, mikrodenetleyicilerde bellenim yazılımını özel programlayıcı devreler gerektirmeden mikrodenetleyiciye yükleme olanağı sağlar

(Fan, 2015; Siegesmund, 2014). Örnek olarak, bellenim yazılımını önyükleyici yazılımıyla USB arabirimli flaş bellek birimi üzerinden mikrodenetleyiciye yüklenebilir (Fan, 2015).

2.4. Mikrodenetleyici

Mikrodenetleyiciler, bir bilgisayarda bulunan işlemci, ana bellek, giriş birimleri ve çıkış birimleri bileşenlerini bir tümdevre olarak barındıran bilgisayarlardır (Sickle, 2001). Mikrodenetleyiciler, aynı zamanda bir tür gömülü sistemdir (Toulson ve Wilmshurst, 2017).

Mikrodenetleyicilerin, hesaplama gücü çoğunlukla çok sınırlıdır. Mikrodenetleyicilerde kullanılan işlemci mimarilerine örnek olarak 8051 (Calcutt vd., 2004), ARM (Toulson ve Wilmshurst, 2017), Atmel AVR (Morton, 2002) ve PIC (Sloss vd., 2014) verilebilir. Mikrodenetleyiciler, bilgisayar mimarisi olarak Von Neumann mimarisi ve Harvard mimarisinde olabilirler. Von Neumann mimarisine dayalı olan mikrodenetleyiciler Harvard mimarisine dayalı olan mikrodenetleyicilerden daha yavaştırlar ve ek veri yolu bulundurmadıklarından büyüklük bakımından daha küçüktürler.

Ek veri yolu bulundurmadıklarından, Von Neumann mimarisine dayalı olan mikrodenenetleyiciler genellikle daha ucuzdur. Mikrodenetleyiciler, bellek türü olarak çoğunlukla hem geçici hem de kalıcı bellek barındırırlar. Geçici bellek olarak RAM, kalıcı bellek olarak ise EPROM, EEPROM ve flaş bellek barındırırlar (Sickle, 2001). Giriş birimleri ve çıkış birimleri olarak zamanlayıcılar, gerçek zaman saati, sayısal giriş ve çıkış birimleri, analog-sayısal çeviriciler, PWM üreteci, I^2C arabirimi, SPI arabirimi, USB arabirimi gibi birimleri barındırabilirler (Siegesmund, 2014).

Mikrodenetleyicilerde çalışan programlar, bir sonsuz döngüde çalışır (Park, 2003; Sickle, 2001). Bu sonsuz döngüde bazı durumlar gerçekleştiğinde başka program parçalarına dallanılır. Bu durumlara örnek olarak, istisnalar, dış kesmeler ve iç kesmeler verilebilir (Sickle, 2001). Mikrodenetleyiciler, genel olarak birleştirici dil, C ya da C++ dilleriyle programlanırlar (Lipovski, 2004). Mikrodenetleyicilere yazılım geliştirmek için çapraz geliştirme araçları kullanılır (Park, 2003; Calcutt vd., 2004). Geliştirilmiş olan yazılımları sınamak için ise benzetim yazılımları kullanılır (Park, 2003). Mikrodenetleyiciler için yazılmış yazılımları, mikrodenetleyicilere yüklemek için

programlayıcılar kullanılır (Park, 2003; Siegesmund, 2014). Bu programlayıcıların birçok türleri vardır (Siegesmund, 2014). Mikrodenetleyicilerin çalışmasını sınamak için programlayıcı ve çevre birimlerinden oluşan mikrodenetleyici geliştirme platformları kullanılır (Park, 2003; Sickle, 2001).

2.5. AVR

AVR, geliştirme sürecinin kısalığı için eniyilenmiş bir mikrodenetleyici türüdür (Morton, 2002). Gömülü sistemlerde geniş kullanım alanları için bir seçenek olarak sunulmuştur. Başarım, enerji verimliliği ve tasarım esnekliğinin bir arada olmasını sağlar. AVR mikrodenetleyicileri, 8 bitlik mikrodenetleyiciler olup, bir CPU, geçici bellek olarak RAM, program belleği olarak flaş bellek, EEPROM bellek ve iç dalga üreteci barındırırlar (Morton, 2002). AVR mikrodenetleyici modellerinin birçoğunda UART, I2C ve SPI arabirimleri bulunur (Morton, 2002). AVR mikrodenetleyicilerinde genellikle ADC bulunmakla birlikte bazı modellerinde DAC bulunur. Bu mikrodenetleyicilerde 8 ya da 16 bitlik zamanlayıcı bulunur.

AVR mikrodenetleyicilerinin tinyAVR, megaAVR ve AVR XMEGA olarak üç ailesi vardır. tinyAVR mikrodenetleyici ailesi, en küçük mikrodenetleyicilerin olduğu türdür. megaAVR mikrodenetleyici ailesi tinyAVR mikrodenetleyici ailesinden daha başarımlı ve daha çok geçici bellek ve kalıcı bellek barındıran mikrodenetleyici ailesidir. AVR XMEGA mikrodenetleyici türü en başarımlı AVR mikrodenetleyici ailesidir (https://www.microchip.com/design-centers/8-bit/microchip-avr-mcus).

2.6. ATmega328P

ATmega328P, AVR mikrodenetleyicilerin bir modeli olup, megaAVR ailesindendir (http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf). ATmega328P mikrodenetleyicilerinin özellikleri şunlardır:

- PDIP olarak, QFN olarak ya da MLF olarak olarak piyasaya sürülür.
- Bilgisayar mimarisi olarak Harvard mimarisindendir.
- ALU'ya doğrudan bağlanılan 32 genel amaçlı yazmaç, 64 giriş ve çıkış yazmacı ve 160 genişletilimş giriş ve çıkış yazmacı barıdırır.

- 2 KB SRAM barıdırır. Yazmaçlar ve SRAM aynı adres uzayını kullanır.
- Program belleği olarak 32 KB flaş bellek barındırır. Bu flaş bellek SPI arabirimi yoluyla programlanabilirdir. Bu flaş belleğin ömrü 10000 yazma ve silme çevrimidir.
- 1 KB EEPROM barıdırır. Bu EEPROM bellek 100000 yazma ve silme ömrüne sahiptir. Bu EEPROM bellek SPI ya da paralel olarak programlanabilirdir.
- 8 MHz iç dalga üreteci ve 128 KHz iç dalga üreteci barıdırır. Ayrıca, en yüksek 20 MHz dış dalga üreteci bağlanabilir. İç dalga üreteçleri ya da dış dalga üretecinin dışında dış dalga sinyaliyle de çalışabilir.
- ATmega328P mikrodenetleyici modelinin boşta kipi, ADC gürültü azaltma kipi, güç azaltma kipi, enerji tasarrufu kipi, bekleme kipi ve genişletilmiş bekleme kipi olmak üzere güç kipleri vardır.
- Atmega328P mikrodenetleyicileri, dört farklı yolla yeniden başlatılabilir:
    - Çalışma gelirimi, çalışma sınırının altına inmişse yeniden başlatılır.
    - Sıfırlama bacağına sıfır değeri verildiği zaman yeniden başlatılır.
    - Watchdog etkinse, watchdog yeniden başlatma kipine programlanmışsa, watchdog sayıcısı sıfırlanamamışsa ve watchdog sayıcısı belli bir seviyeye gelmişse yeniden başlatılır.
    - Eğer, voltaj azalması algılayıcısı etkinse ve çalışma gelirimi belli bir seviyenin altına inmişse yeniden başlatılır.
- İki 8 bitlik genel amaçlı zamanlayıcı ve bir 16 bitlik genel amaçlı zamanlayıcı barındırır.
- Port B, Port C ve Port D olmak üzere üç giriş ve çıkış portu vardır. Bu portlar GPIO görevini görür. Aynı zamanda bu portları belirttiği bazı bacaklar ADC, PWM, I^2C, UART, SPI, görevlerini de görür.
- İki giriş ve çıkış bacağında dış kesme isteği yapılabilir. Ayrıca, sıfırlama bacağında da dış kesme istekleri vardır.

- PDIP olarak dağıtılanları 6 tane ve QFN olarak dağıtılanları ve MLF paketli olarak dağıtılanları 8 tane 10 bitlik ADC kanalı barındırır. Bu ADC kanallarının referans olarak bir AREF kanalının giriş gerilimi olabilir.
- İki kanallı AC barındırır.
- Altı PWM kanalı barındırır. Bu PWM kanalları genel amaçlı zamanlayıcılarıyla ayarlanır.
- İki SPI kanalı barındırır. Bir bağımlı SPI kanalı vardır.
- Bir UART kanalı barındırır. Bu UART kanalını USART ya da SPI olarak kullanılma olanağı vardır.
- Bir I^2C kanalı barındırır.
- Bir watchdog zamanlayıcısı barındırır.
- Mikrodenetleyici üzerinde hata ayıklama olanağı vardır.
- Önyükleyici kullanma olanağı vardır. Önyükleyici için program belleği olan flaş belleğin adres uzayının bir bölümü ayrılmıştır. Bu ayrılan bölümün büyüklüğü ayarlanabilir.
- 1,8V-5,5V gerilim aralığında çalışabilir.
- Etkin çalışmada 0,2mA akım çeker. Güç azaltma kipinde 0,1µA akım çeker. Enerji tasarrufu kipinde 0,75µA akım çeker.
- -40°C'den 105°C'ye kadar sıcaklık aralığında çalışabilir.

2.7. Arduino

Arduino, açık kaynaklı bir donanım ve yazılım geliştirme platformu olup, Arduino kartları ve Arduino yazılımı olarak ikiye ayrılır (https://www.arduino.cc/). Arduino kartları, elektronik geliştirme ortamlarıdır. Bu kartların birçok türü bulunmaktadır. Bu kartlar, kalkan denilen genişleme kartlarıyla özellik eklenebilir (https://www.arduino.cc/en/Main/Products). Arduino yazılımı, Arduino kartlarını programlamak için kullanılır. Bu yazılımda Arduino programlamak için Arduino

programlama dili kullanılır (https://www.arduino.cc/en/Main/Software). Arduino platformu için geniş belgelendirme desteği vardır (https://www.arduino.cc/).

Arduino donanımında devre kartının üzerinde bir mikrodenetleyici, bir USB seri arabirimi, bir güç kaynağı ve genişleme bağlantıları yer almaktadır (Wheat, 2011). Arduino, kolay programlama olanağı sağlamaktadır (Pan ve Zhu, 2018). Arduino, çok fazla elektronik bilgisi gerektirmez (Trevennor, 2012).

2.8. Arduino Uno

Arduino Uno, Atmega328P tabanlı bir mikrodenetleyici kartıdır. Arduino Uno üzerinde bulunan bu mikrodenetleyici PDIP olarak bulunur. Arduino Uno kartında mikrodenetleyici için 16MHz'lik bir dalga üreteci, bir USB-UART çevirici, bir USB bağlantısı ve bir güç bağlantısı, ICSP bağlantısı ve sıfırlama düğmesi bulunur. Arduino Uno kartı USB üzerinde 5V'luk gerilimde güç alabilir ya da USB bağlantısı yapılmadığı zaman 7V-12V gerilim aralığında güç alabilir (https://store.arduino.cc/usa/arduino-uno-rev3).

2.9. Ethernet

Ethernet, sık kullanılan bir ağ standardıdır. Bu standartta, ağ yapısınn kabloyla olacağını belirtir. Ethernet standardında kullanılan kablo türleri koaksiyel kablo, çift bükümlü kablo ve fiber optik kablo türleridir. Ethernet standardında belirtilen iletim hızları 10Mbps, 100Mbps, 1Gpbs, 10Gbps, 40Gbps ve 100Gpbs'dir. Ethernet konusunda gelecek için çalışmalar sürmektedir (Robertazzi, 2017).

2.10. Wiznet 5100

Wiznet 5100 Ethernet denetleyicisi, gömülü sistemler için tasarlanmış Ethernet denetleyicisidir. Wiznet 5100, bir tümdevre olarak satışa sunulmaktadır. Wiznet 5100, işletim sistemi olmadan ağ bağlantısı için tasarlanmıştır. Wiznet 5100 Ethernet denetleyicisi, IEEE 802.3 10BASE-T ve IEEE 802.3u 100BASE-TX standartlarını desteklemekte olup, bu denetleyicide donanım tabanlı olarak TCP, UDP, IPv4, ICMP, ARP, IGMP ve PPPoE protokolleri desteklenmektedir (http://www.wiznet.io/wp-content/uploads/wiznethome/Chip/W5100/Document/W5100_Datasheet_v1.2.7.pdf).

Donanımsal olarak dört ağ soketini destekleyen Wiznet 5100 ethernet denetleyicisinde dört ağ soketi için 8 KB okuma önbelleği ve 8 KB yazma önbelleği bulunmaktadır.

2.11. HTTP

HTTP, İnternet sitelerini yayınlamak için kullanılan bir iletişim kuralıdır. Bu iletişim kuralı, istemci tarafından gelen komut tabanlı isteklere göre sunucu tarafından istemciye isteklerin yanıtlanması ve istek yapılan bilgilerin gönderilmesini tanımlar (Goralski, 2017). HTTP'nin HTTP 0.9, HTTP 1.0 ve HTTP 1.1 olmak üzere üç sürümü vardır (Goralski, 2017).

BÖLÜM 3

MEVCUT ÖNYÜKLEYİCİLER

Bu bölümde, Atmel AVR mimmarisinin 8 bitlik olanı için tasarlanmış mevcut önyükleyicilerin en bilinenleri incelenmiştir.

3.1. Arduino Bootloader

Arduino için tasarlanmış bu önyükleyici olup, açılışta uygulama programı denetimi için bir süre beklemesi gerekir. UART üzerinden yükleme için özel bir program gerektirmekte olup, bu özel programı edinmek, son kullanıcı için zor olabilir (https://www.arduino.cc/en/Hacking/Bootloader).

3.2. Optiboot

Arduino platformunun Atmel AVR mimarisi için standart önyükleyicisinden daha hafif ve daha hızlıdır (Marcin, Tomasz ve Piotr, 2017). Ancak UART üzerinden yükleme için özel bir program gerektirmektedir. Bundan dolayı, program güncelleme işlemi son kullanıcı için zor olabilir. Ayrıca, bu önyükleyicinin, açılışta uygulama

programı denetimi için bir süre beklemesi gerekmektedir (Marcin, Tomasz ve Piotr, 2017; https://github.com/Optiboot/optiboot).

3.3. AN_8429

Bu önyükleyici uygulama programını USB üzerinden yükler. Ancak, uygulama programını yüklemek için özel bir program gerekir. Bu, son kullanıcı için zor olabilir. Ayrıca, açılışta uygulama programı denetimi için bir süre beklemesi gerekmektedir (http://ww1.microchip.com/downloads/en/AppNotes/doc8429.pdf).

3.4. AN_42788

Bu önyükleyici uygulama programını SD kart üzerinden yükler. Açılışta SD kartta uygulama programı olup olmadığının denetiminde SD kartın okunması için bir süre beklemektedir (http://ww1.microchip.com/downloads/en/AppNotes/doc8457.pdf).

3.5. TFTP Bootloader

Ethernet arabirimi yoluyla TFTP sunucusu üzerinden uygulama programı yükleme yöntemini kullanan TFTP Bootloader mikrodenetleyiciyi bir ağ bağdaştırıcısı üzerinden bir bilgisayar ağına bağlamak ve bu ağa bağlı başka bir bilgisayar üzerinde çalışan bir TFTP sunucu yazılımı gerektirmektedir. Bu önyükleyiciler, TFTP sunucusu üzerinden belirli bir isimde olan bir dosyayı okurlar (http://playground.arduino.cc/Code/TFTPBootloader1). Ancak, TFTP sunucusu üzerinden uygulama programı yükleme yöntemi, son kullanıcı için zor olabilir. Ayrıca, bu önyükleyicinin açılışta uygulama programı denetimi ve Ethernet arabiriminin başlatılması için bir süre beklemesi gerekir (http://playground.arduino.cc/Code/TFTPBootloader1).

BÖLÜM 4

MICRO BOOT ÖNYÜKLEYİCİSİ

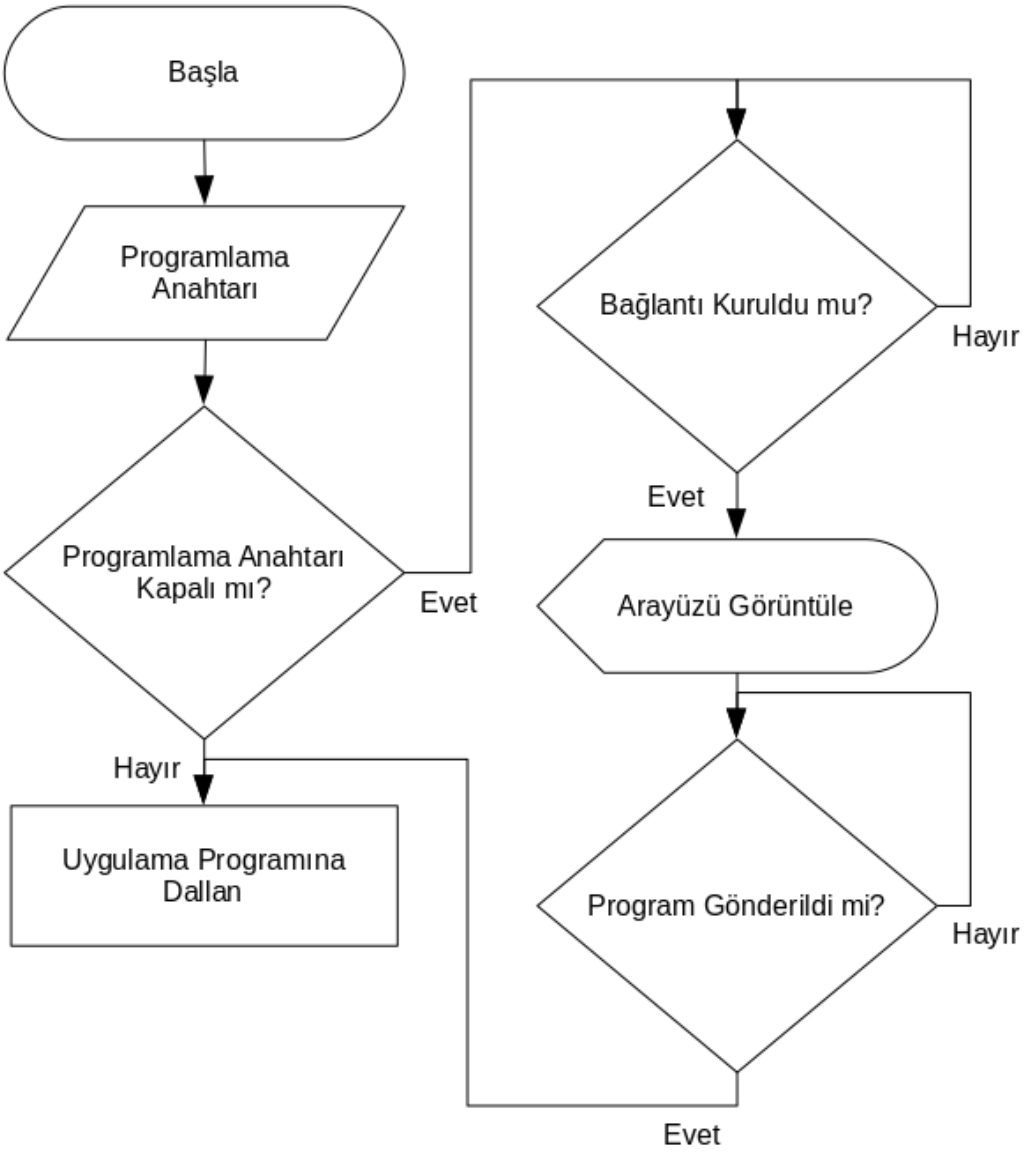
Bu bölümde tez çalışması kapsamında geliştirilen önyükleyicinin yazılım geliştirme süreci hakkında bilgiler verilmektedir.

4.1. Geliştirilen Yazılımla İlgili Genel Bilgiler

Bu çalışmada, Atmel AVR mikrodenetleyicileri için bir önyükleyici geliştirilmiştir. Geliştirilen önyükleyici yazılımına "Micro Boot" adı verilmiştir. Micro Boot önyükleyicisi, uygulama programını bir bilgisayar ağına bağlı Ethernet denetleyicisi ve TCP/IP iletişim kuralı yoluyla HTTP üzerinden yükler. Kullanıcının, ağ tarayıcısı üzerinden uygulama programı yüklemesi için bir ağ sayfası barındırır. Ağ sayfasını iletmek için sunucu görevi de görür.

Micro Boot önyükleyicisi, HTTP kullanıcı arayüzünü istek geldiği zamanda görüntülemek için mikrodenetleyicinin belirlenmiş bir GPIO bacağını kullanır. Bu bacak kablolar ve bir anahtarla gelirim kaynağının pozitif kutbuna bağlıdır. Bu anahtar programlama anahtarı görevini görür. Micro Boot önyükleyicisinin uygulama programı yükleme sayfası için programlama anahtarının açılması, bilgisayar ağına bağlı başka bir bilgisayarda ağ tarayıcısında IP adresinin ve port numarasının belirtilerek istek gönderilmesi gerekir. Uygulama programı yüklenmesi için ağ sayfasından dosya

yükleme denetimine dosya tanıttırılması ve gönderme denetimine basılması gerekir. Micro Boot önyükleyicisinin çalışma prensibi Şekil 4.1'de, önyükleyicinin HTTP kullanıcı arayüzü ise Şekil 4.2'de gösterilmektedir.

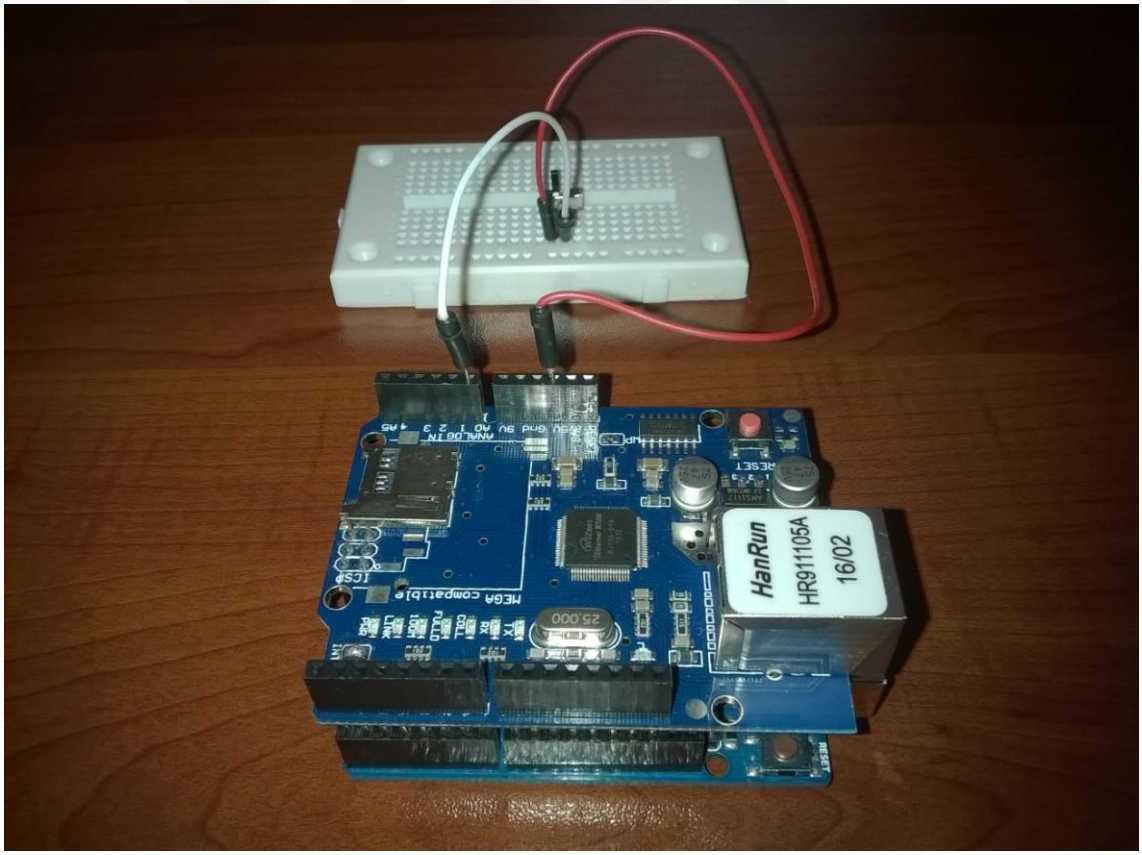


Şekil 4.1. Micro Boot Önyükleyicisinin Çalışmasını Gösteren Akış Şeması

Şekil 4.2. Micro Boot Önyükleyicisinin HTTP Kullanıcı Arayüzü

Önyükleyici için kullanılan geliştirme ortamında bir Arduino Uno, bit Wiznet 5100 Ethernet denetleyicisi barındıran bir Arduino kalkanı, bir breadboard, iki jumper kablo ve bir anahtar kullanılmıştır.

Önyükleyici için kurulan geliştirme ortamı Şekil 4.3'te gösterilmektedir.



Şekil 4.3. Micro Boot İçin Kurulan Geliştirme Ortamı

4.2. Geliştirilen Yazılımın Kısıtları

Micro Boot önyükleyicisi, yalnızca saf makine kodu biçiminde olan dosyaları desteklemektedir. Ayrıca, Micro Boot önyükleyicisinin görevini yapabilmesi için mikrodenetleyicinin GPIO bacaklarından biri yalnızca bu önyükleyici için ayrılmaktadır.

4.3. Geliştirilen Yazılımın İç Yapısı

Micro Boot önyükleyicisi C diliyle yazılmıştır. Micro Boot önyükleyicisinin barındırdığı ağ sayfası HTML diliyle yazılmıştır. Micro Boot önyükleyicisi bir ana gövde ve çeşitli bileşenlerden oluşmuştur. Bu bileşenler, aygıt sürücüsü bileşenleri, kesme isteği bileşeni ve kullanıcı arayüzü bileşeninden oluşmuştur. Micro Boot önyükleyicisinin program boyutu 3834 bayttır. Bundan dolayı, Micro Boot önyükleyicisi Atmega328p mikrodenetleyicisinin program belleğinin önyükleyici için ayrılmış bölümüne sığmaktadır.

Bu çalışmada geliştirilen bileşenler Atmel AVR mikrodenetleyicisinin megaAVR türü için geliştirilmiştir. Geliştirilen önyükleyicinin C diliyle yazılmış olması ve bileşenli yapısı, bu önyükleyicinin diğer bilgisayar platformları için geliştirilmesine ve bu önyükleyiciye başka bileşenlerin eklenilmesine olanak sağlar. Geliştirilen yazılımın dizinlerinin ve dosyalarının açıklamaları Tablo 4.1 ve Tablo 4.2'de sunulmuş olup, Micro Boot önyükleyicisinin kaynak kod dosyaları ise ek olarak verilmiştir.

Micro Boot önyükleyicisi ağ sayfasıda bulunabilecek İngilizce diline olan ileti ve Türkçe dilinde olan ileti için bir başlık dosyası tanımlanmıştır. Micro Boot önyükleyicisini derlenmeden önce yapılandırmak için bir başlık dosyası tanımlanmıştır.

Micro Boot önyükleyicisinde kullanılan bazı standart fonksiyonları içeren bir kaynak dosyası bulunmaktadır. Micro Boot önyükleyicisi için derleme işlemini gerçekleştirecek ve mikrodenetletyicinin fuse bitlerini ayarlamak için bir dosya bulunmaktadır. Micro Boot önyükleyicisi için İngilizce dilinde ve Türkçe dilinde belgelendirme yapılmıştır.

Tablo 4.1. Geliştirilen Yazılımın Dizin Yapısı

Dizin	Açıklama
binaries	Çalıştırılabilir dosyaların bulunduğu dizin
documentation	Yazılımla ilgili yardım belgelerin bulunduğu dizin
documentation/en-US	Yazılımla ilgili Amerikan İngilizcesi diliyle yazılmış belgenin bulunduğu dizin
documentation/tr-TR	Yazılımla ilgili Türkiye Türkçesi diliyle yazılmış belgenin bulunduğu dizin
platforms	Bilgisayar platformu bağımlı kaynak dosyalarının bulunduğu dizin
platforms/megaavr	megaAVR platformu için kaynak dosyalarının bulunduğu dizin
standard_library	Bu yazılım için oluşturulmuş standart kütüphanenin kaynak dosyasının bulunduğu dizin

Tablo 4.2. Geliştirilen Yazılımın Dosyaları

Dizin	Açıklama
README	Yazılımın yardım belgelerinin dizin konumunu belirten metin dosyası
configurations.h	Yazılımın derleme sırasında olan yapılandırmasını belirten başlık dosyaları
strings.h	Yazılımın sözcelerini belirten başlık dosyası
documentation/en-US/README	Yazılımın Amerikan İngilizcesi diliyle yazılmış belgesinin bulunduğu dizin
documentation/tr-TR/README	Yazılımın Türkçe yazılmış belgesinin bulunduğu dizin
Makefile	Yazılımın inşa dosyası
main.h	Yazılımın ana başlık dosyası
main.c	Yazılımın ana kaynak dosyası
standard_library/standard_library.c	Bu yazılım için oluşturulmuş standart kütüphanenin kaynak dosyasının
platforms/megaavr/uart.c	megaAVR platformu için UART sürücü dosyası
platforms/megaavr/wiznet_5100.c	megaAVR platformu için Wiznet 5100 Ethernet denetleyicisinin sürücü dosyası
platforms/megaavr/http_user_interface.c	megaAVR platformu için HTTP kullanıcı arayüzü

4.4. Geliştirilen Yazılımın Geliştirme Sürecinde Kullanılan Yazılımlar

Micro Boot önyükleyicisi için çapraz araç zincir sistemi kullanılmıştır. Bu çapraz araç zinciri Atmel AVR mimarisinin 8 bitlik olanı içindir. Bu çapraz araç zinciri, GNU Binutils (https://www.gnu.org/software/binutils/), GNU C Compiler

(https://gcc.gnu.org/), AVR Libc (http://www.nongnu.org/avr-libc/) yazılımlarından oluşur. Önyükleyiciyi derlemek için GNU Make (https://www.gnu.org/software/make/) yazılımından yararlanılmıştır. Önyükleyiciyi mikrodenetleyicinin program belleğine yazdırmak için AVRDUDE (http://www.nongnu.org/avrdude/) yazılımı kullanılmıştır. Önyükleyicide hata ayıklanması için UART arabirimi kullanılmıştır. UART arabiriminden veri almak için Microcom (https://git.pengutronix.de/cgit/tools/microcom) yazılımı kullanılmıştır. Bu yazılımların sürümleri Tablo 4.3'te belirtilmiştir.

Tablo 4.3. Micro Boot Yazılımını Geliştirmek İçin Kullanılan Yazılımlar

Yazılım	**Sürüm**
GNU Binutils	2.29.1
GNU C Compiler	7.3.0
AVR Libc	2.0.0
GNU Make	4.2.1
AVRDUDE	6.3
Microcom	2016.01.0

4.5. Geliştirilen Yazılımda kullanılan Derleme ve Bağlama Seçenekleri

Micro Boot önyükleyicisi için Makefile dosyasında belirtilen GCC derleyicisinin derleme seçenekleri ve bağlama seçenekleri Tablo 4.4'de sunulmaktadır.

Tablo 4.4. Micro Boot Önyükleyicisi İçin kullanılan GCC Derleyicisinin Derleme Seçenekleri ve Bağlama Seçenekleri

Seçenek	**Açıklama**
-O1	Program uzunluğu bakımından 1. en iyileme seviyesini belirtir. Bu seviye bütün en iyileme seviyelerinin uygulanacağını belirtir.

-DF_CPU=$(F_CPU)	Mikrodenetleyicinin bağlı olduğu saat üretecinin frekansını Hz cinsinden belirtir. $(F_CPU) değişkeni saat üretecinin frekansını belirtir.
-mmcu=$(DEVICE)	AVR mikrodenetleyicisinin türünü belirtir. $(DEVICE) değişkeni mikrodenetleyici türü belirtir.
-Ttext=$(START_BOOTLOADER_SECTION)	Programın, mikrodenetleyicinin program belleğinin hangi konumundan başlayarak yazılacağını belirtir.

4.6. Geliştirilen Yazılımla İlgili Sınama Verileri

Micro Boot önyükleyicisinin geliştirme ortamında kullanılan ortamın geliştirme sürecinde kullanılan yerel ağda olan yanıtlama süreleri ölçülmüştür. Bu süreler, ağ tarayıcısında isteklere karşı yanıt süreleridir. Bu sürelerin değerlendirilmesinde Wiznet 5100 Ethernet denetleyicisinin ICMP paketlerinin yanıtlama süreleri kullanılmıştır. Bu süre değerleri milisaniye cinsindendir.

Bu çalışmada, sınama amaçlı kurulan yerel ağla ilgili Wiznet 5100 Ethernet denetleyicisinin ICMP paketlerinin yanıtlama süreleri iputils (http://www.skbuff.net/iputils/) yazılımının ping bieşeniyle ölçülmüştür. Ölçümde 30 paket gönderilmiştir ve Wiznet 5100 Ethernet denetleyicisi tarafından yanıtlanmıştır. Ayrıca, bu sürelerin aritmetik ortalaması hesaplanmıştır. Bu süreler ve bu sürelerin aritmetik ortalaması Tablo 4.5'de listelenmektedir.

Micro Boot önyükleyicisinin sınama amaçlı kurulan yerel ağda istek geldiğinde ağ süreleri ölçülmüştür (Tablo 4.6). Bunun için curl (https://curl.haxx.se/) aracı kullanılmıştır. Aşağıda verilen kategorilerde süreler ölçülmüştür ve aşağıda curl aracının parantez içinde bu kategorilere karşılık gelen isimleri verilmiştir:

- URL çözümlenmesine kadar geçen süre (name_lookup)
- Bağlantı kurulduktan sonra olan geçen süre (time_connect)
- Veri aktarım öncesi işlemlere kadar olan süre (time_pretransfer)
- Veri aktarım başlangıcına kadar olan süre (time_starttransfer)
- Toplam süre (time_total)

Tablo 4.5. Wiznet 5100 Ethernet Denetleyicisinin Sınama Sürecinde Kullanılan Yerel Ağda Olan ICMP Paketlerinin Milisaniye Cinsinden Yanıtlama Süreleri

Paket Srası	**Yanıtlanma Süresi (ms)**
1	0,192
2	0,181
3	0,146
4	0,202
5	0,144
6	0,160
7	0,196
8	0,157
9	0,214
10	0,193
11	0,191
12	0,194
13	0,205
14	0,200
15	0,216
16	0,199
17	0,217
18	0,216
19	0,212
20	0,217
21	0,141
22	0,213
23	0,202
24	0,197
25	0,199
26	0,198
27	0,198
28	0,202
29	0,198
30	0,145
Aritmetik Ortalaması	0,1915

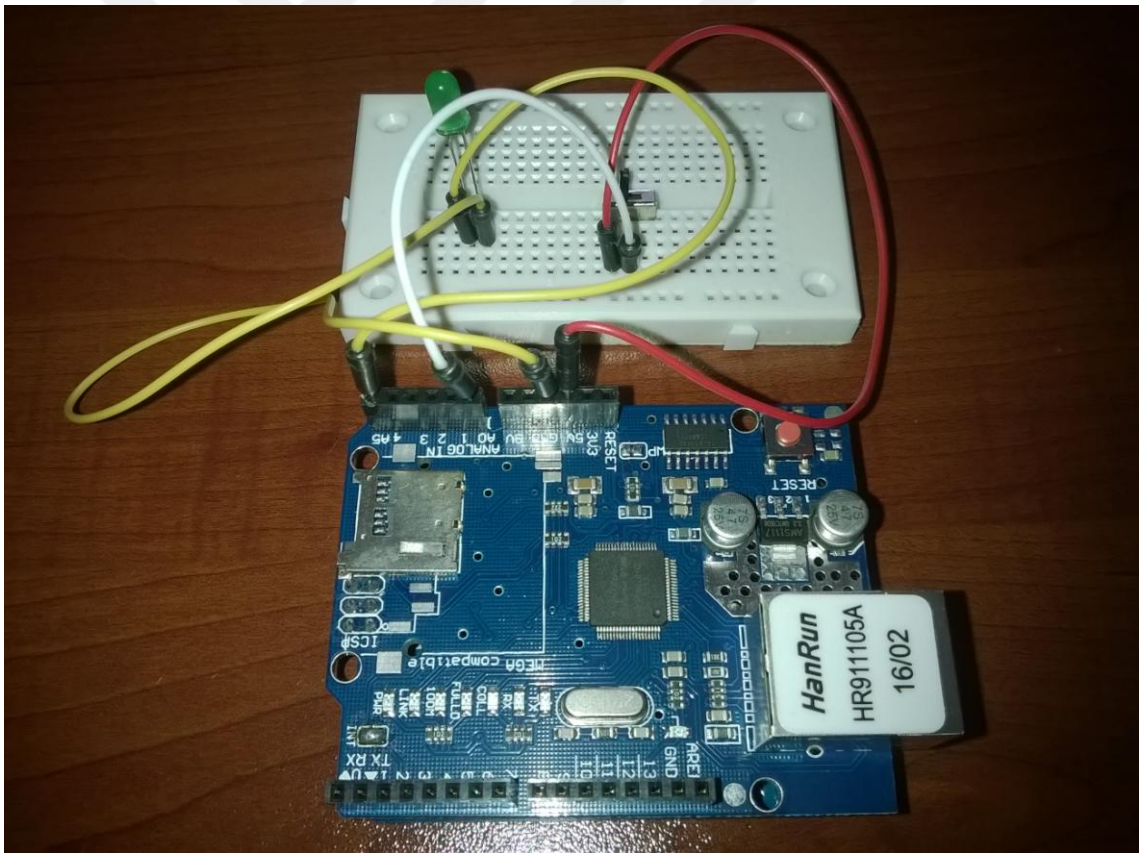
Tablo 4.6. Geliştirilen Yazılıma Sınama Sürecinde KullanılanYerel Ağdan İstek Gönderildiğinde Milisaniye Cinsinden Geçen Süreler

Kategori	**Süre (ms)**
name_lookup	0,117
time_connect	0,398
time_pretransfer	0,467
time_strattransfer	1,271

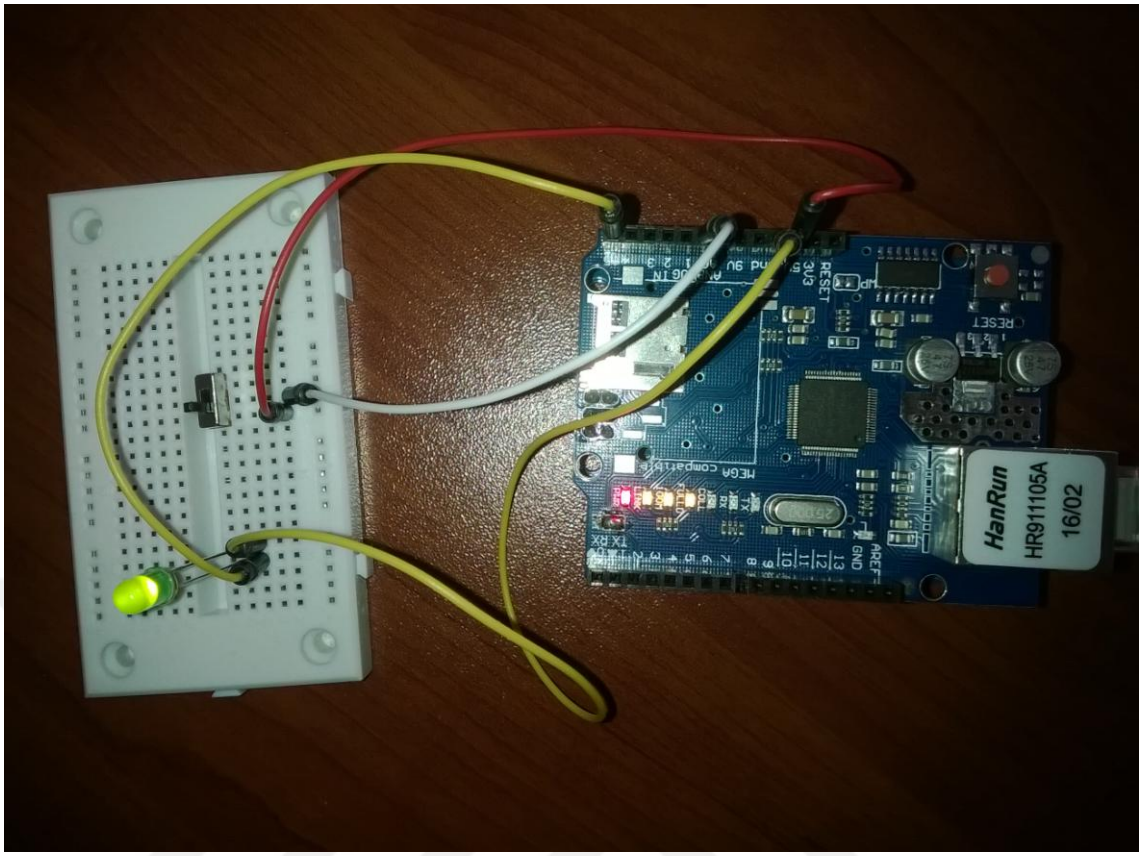
time_total	4,417

4.7. Geliştirilen Yazılımın Düzgün Çalışıp Çalışmadığının Sınanması

Bu çalışmada, Micro Boot önyükleyicisinin örnek bir uygulama programıyla sınanmıştır. Bu örnek uygulama programı, Micro Boot önyükleyicisi için kullanılan aynı araç zinciri kullanılmıştır. Bu örnek uygulama programının kaynak kodu ekte verilmiştir. Önyükleyicinin düzgün çalışıp çalışmadığını anlamak için geliştirme ortamına bir yeşil LED bağlanmıştır (Şekil 4.4). Yeşil renkli LED'in kullanılma nedeni, 5V gerilimde düzgün çalışmasıdır. Örnek kod yüklendikten sonra yeşil LED'in yandığı görülmüştür (Şekil 4.5).



Şekil 4.4. Geliştirme Ortamına Bir Yeşil LED Bağlanması



Şekil 4.5. Uygulama Programının Çalışması

BÖLÜM 5

SONUÇLAR VE TARTIŞMA

Tez çalışması kapsamında geliştirilen Micro Boot önyükleyicisi, mevcut önyükleyiciler ile bazı ölçütler göz önünde bulundurularak kıyaslanmıştır. Kıyaslamalarda kullanılan ölçütler şunlardır:

- Program Yükleme Arabirimi
- Uygulama Programı İçin Ethetnet Arabirimi Gerekmeyen Sistemlerde Olabilecek Ek Maliyet
- Uygulama Programı İçin Ethernet Arabirimi Gereken Sistemlerde Olabilecek Ek Maliyet
- Özel Program Gereksinimi
- Kullanım Kolaylığı
- Açılışta Bekleme Süresi

Diğer önyükleyicilerden farklı olarak Micro Boot ve AN_42788 önyükleyicilerinin özel program gereksinimi yoktur.

Diğer önyükleyicilerle kıyaslandığında, Micro Boot ve AN_42788 önyükleyicileri kullanım kolaylığına sahiptir.

Diğer önyükleyicilerden farklı olarak, Micro Boot önyükleyicisinin bekleme süresi yoktur. Bunun nedeni ek bir anahtar kullanmasıdır. Bu anahtar hızlı açılması istenen sistemlerde kullanılabilir. Diğer önyükleyiciler de bu yöntemi izleyecek şekilde

geliştirilebilirler. Micro Boot önyükleyicisinin anahtar kullanılmadığında bekleme süresi gerekir ve HTTP isteği gönderme komutu manuel verilmelidir.

Micro Boot önyükleyicisi yalnızca Ethernet arabirimi olan sistemlerde son kullanıcı için iyi bir seçenektir. TFTP Bootloader önyükleyicisi de benzer sistemlere bir önyükleyici alternatifi olabilir.

Uygulama programı yükleme arabirimi olarak hem Ethernet hem de UART mevcutsa, Micro Boot önyükleyicisi iyi bir alternatif olabilir. Ancak, Arduino IDE gibi geliştirme ortamları kullanılıyorsa Arduino Bootloader ya da Optiboot kullanılması daha uygun olabilir.

Uygulama programı yükleme arabirimi olarak hem Ethernet hem de SD kart mevcutsa, Micro Boot ya da AN_42788 önyükleyicileri iyi alternatifler olabilir. Bu tarz sistemlerde SD kartla programlama yükleme dışında Ethernet arabirimi üzerinden HTTP iletişim kuralıyla uygulama programı yükleme işlemi yapılabilir. Ethernet üzerinden uygulama programı yükleme işlemi İnternet üzerinden yapılıyorsa ağ geçidinin yapılandırılması gerekir.

Uygun donanıma sahip olan bazı sistemlerde UART ve USB arabirimleri üzerinden uygulama programı yükleme işlemi gerçekleştirebilir. Micro Boot ve TFTP Bootloader önyükleyicileri bu tarz sistemler için uygun değildir. Gerçekleştirilen kıyaslamaların sonuçları Tablo 5.1'de sunulmaktadır.

Sonuç olarak, Micro Boot önyükleyicisi, Ethernet arabirimine sahip, güvenliğin kritik düzeyde öneme sahip olmadığı sistemlerde özellikle son kullanıcılar için iyi bir önyükleyici alternatifi olabilir. Ayrıca, güvenliğin kritik düzeyde öneme sahip olduğu sistemlerde güvenlik duvarı kullanımıyla gerekli bilgi güvenliği tedbirleri alınabilir.

Tablo 5.1. Önyükleyicilerin Karşılaştırma Tablosu

	Arduino Bootloader	Optiboot	AN_8429	AN_42788	TFTP Bootloader	Micro Boot
Program Yükleme Arabirimi	UART	UART	USB	SD Kart	Ethernet ve TFTP İletişim Kuralı	Ethernet ve HTTP İletişim Kuralı
Uygulama Programı İçin Ethetnet Arabirimi Gerekmeyen Sistemlerde Olabilecek Ek Maliyet	Az	Az	Orta	Orta	Çok Fazla	Çok
Uygulama Programı İçin Ethernet Arabirimi Gereken Sistemlerde Olabilecek Ek Maliyet	Orta	Orta	Çok	Çok	Az	Az
Özel Program Gereksinimi	Var	Var	Var	Yok	Var	Yok
Kullanım Kolaylığı	Orta	Orta	Orta	Kolay	Orta	Kolay
Açılışta Bekleme Süresi	Var	Var	Var	Var	Var	Yok

KAYNAKLAR

Calcutt, D., Cowan, F., Parchizadeh, H. (2004). *8051 Microcontrollers*. Oxford: Newnes.

Fan, X. (2015). *Real-Time Embedded Systems*. Oxford: Newnes.

Goralski, W., (2017). The Illustrated Network. (2. Baskı) Cambridge: Morgan Kaufmann

Heath, S. (2002). *Embedded Systems Design*. (2. Baskı). Oxford: Newnes.

Kraeling, M., McKay, A. (2013). *In Software Engineering for Embedded Systems*. Oxford: Newnes.

Jiménez, M., Palomera, R., Couvertier, I. (2014) *Introduction to Embedded Systems*. doi: 10.1007/978-1-4614-3143-5

Lipovski, G.J., (2004). *Introduction to Microcontrollers*. (2. Baskı). Burlington: Academic Press.

Marcin, L., Tomasz, O., Piotr, P. (2017). *Dedicated AVR bootloader for performance improvement of prototyping process*. Sosnowiec: MIXDES

Morton, J., (2002). *AVR*. Oxford: Newnes.

Pan, T., Zhu, Y. (2018). *Designing Embedded Systems with Arduino*. Singapore: Springer.

Parab, J., Shinde, S.A., Shelake, V.G., Kamat, R.K., Naik, G.M. (2008). *Practical Aspects of Embedded System Design using Microcontrollers*. doi: 10.1007/978-1-4020-8393-8

Park, J. (2003). *Practical Embedded Controllers*. Oxford: Newnes.

Robertazzi T. G., (2017). *Introduction to Computer Networking*. doi: 10.1007/978-3-319-53103-8

Sickle, T. V., (2004). *Programming Microcontrollers in C*. (2. Baskı). Burlington: Newnes.

Siegesmund, M. (2015). *Embedded C Programming*. Oxford, Waltham: Newnes.

Sloss, A. N., Symes, D., Wright, C., Rayfield, J. (2004). *ARM System Developer's Guide*. Burlington: Elsevier.

Toulson, R., Wilmshurst, T. (2017). *Fast and Effective Embedded Systems Design*. (2. Baskı). Oxford, Cambridge: Newnes.

Trevennor, A. (2012). *Practical AVR Microcontrollers*. Berkeley, CA: Apress.

Wheat, D. (2011). *Arduino Internals*. Apress.

https://git.pengutronix.de/cgit/tools/microcom

http://playground.arduino.cc/Code/TFTPBootloader1

http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf

http://ww1.microchip.com/downloads/en/AppNotes/doc8457.pdf

http://ww1.microchip.com/downloads/en/AppNotes/doc8429.pdf

http://www.nongnu.org/avrdude/

http://www.nongnu.org/avr-libc/

http://www.skbuff.net/iputils/

http://www.wiznet.io/wp-content/uploads/wiznethome/Chip/W5100/Document/W5100_Datasheet_v1.2.7.pdf

https://curl.haxx.se/

https://gcc.gnu.org/

https://github.com/Optiboot/optiboot

https://store.arduino.cc/usa/arduino-uno-rev3

https://www.arduino.cc/

https://www.arduino.cc/en/Hacking/Bootloader

https://www.arduino.cc/en/Main/Products

https://www.arduino.cc/en/Main/Software

https://www.gnu.org/software/binutils/

https://www.gnu.org/software/make/

https://www.microchip.com/design-centers/8-bit/microchip-avr-mcus

EK 1

GELİŞTİRME ORTAMINDA AYARLANAN FUSE BİTLERİ

Ek 1.1. Genişletilmiş Fuse(Extended Fuse) Bitleri

Fuse Biti	Değer
-	1
-	1
-	1
-	1
-	1
BODLEVEL2	1
BODLEVEL1	1
BODLEVEL0	1

Ek 1.2 – Yüksek Fuse (High Fuse) Bitleri

Fuse Biti	Değer
RSTDISBL	1
DWEN	1
SPIEN	0
WDTON	1
EESAVE	1
BOOTSZ1	0
BOOTSZ0	0
BOOTRST	0

Ek 1.3. Düşük Fuse (Low Fuse) Bitleri

Fuse Biti	Değer
CKDIV8	1
CKOUT	1
SUT1	1
SUT0	1
CKSEL3	1
CKSEL2	1
CKSEL1	1
CKSEL0	1

EK 2

GELİŞTİRİLEN YAZILIMIN KAYNAK KODLARI

README Dosyası:

```
Micro Boot 0.1

Engilish (US): English documentation of this software is in
"documentation/en-US" directory.

Türkçe (TR): Bu yazılımın Türkçe belgelendirmesi,
"documentation/tr-TR"

dizininin içindedir.
```

configurations.h Dosyası:

```
/*****************************************
 * Micro Boot Configuration Header File *
 *****************************************/

// Main Configurations
```

```
#define STRINGS_LANGUAGE TR
// Strings' language (EN or TR)


// Driver Configurations

#define UART_ENABLE 0
// Enable UART interface
#define UART_BAUD_RATE 19200
// UART's baud rate

#define ETHERNET_ADAPTER_ENABLE 1
// Enable Ethernet interface
#define ETHERNET_ADAPTER DEVICE_WIZNET_5100
// Type of Ethernet
#define ETHERNET_ADAPTER_PORT_MODE_ADDRESS DDRB
// Ethernet adapter port mode address
#define ETHERNET_ADAPTER_PORT_ADDRESS PORTB
// Ethernet adapter port address
#define ETHERNET_ADAPTER_CHIP_SELECTION PORTB2
// Ethernet adapter chip selection


// Switch Configurations

#define SWITCH_PORT PINC
// Switch port
#define SWTICH_PIN PC0
// Switch pin

// User Interface Configurations
```

```
#define USER_INTERFACE_TYPE HTTP
// User interface type
#define USER_INTERFACE_PORT 8000
// User interface port

// Network Configurations

#define HARDWARE_MAC_ADDRESS_1 0x00
#define HARDWARE_MAC_ADDRESS_2 0x16
#define HARDWARE_MAC_ADDRESS_3 0x36
#define HARDWARE_MAC_ADDRESS_4 0xDE
#define HARDWARE_MAC_ADDRESS_5 0x58
#define HARDWARE_MAC_ADDRESS_6 0xF6

#define SOURCE_IP_ADDRESS_1 192
#define SOURCE_IP_ADDRESS_2 168
#define SOURCE_IP_ADDRESS_3 0
#define SOURCE_IP_ADDRESS_4 5

#define SUBNET_MASK_1 255
#define SUBNET_MASK_2 255
#define SUBNET_MASK_3 255
#define SUBNET_MASK_4 0

#define GATEWAY_IP_ADDRESS_1 192
#define GATEWAY_IP_ADDRESS_2 168
#define GATEWAY_IP_ADDRESS_3 0
#define GATEWAY_IP_ADDRESS_4 1
```

strings.h Dosyası:

```
/**********************************
```

```
 * Micro Boot Strings Header File *
 ********************************/

#include "configurations.h"

#define EN 1
#define TR 2

#if STRINGS_LANGUAGE == 1  // English
   #define STRING_FIRMWARE_TO_UPLOAD "Firmware to upload:"
#elif STRINGS_LANGUAGE == 2  // Turkish
   #define STRING_FIRMWARE_TO_UPLOAD "Yüklenecek bellenim:"
#endif
```

documentation/en-US/README Dosyası:

```
Micro Boot 0.1

All rights belong to Ercan Ersoy. This program is licensed
under the
GNU LGPL v3.

Micro Boot is being developed for microcontrollers. Program
uploading is
perfomed over HTTP. Currently only this bootloader is being
developed
```

for ATmega328 and ATmega328p from microcontrollers in AVR architecture.

Micro Boot is configured by configurations.h file.

Compile options, liking options and writting program memory options
by Makefile file.

For view HTTP user interface on browser, required supply logic one
determinated a pin and after, send request on
configuration.

Micro Boot is tested on Mozilla Firefox.

documentation/tr-TR/README Dosyası:

Micro Boot 0.1

Bütün hakları Ercan Ersoy'a aittir. Bu program GNU LGPL 3. Sürüm ile
lisanslanmıştır.

Micro Boot, mikrodenetleyiciler için geliştirilen bir önyükleyicidir.
Program yükleme işlemi HTTP üzerinden gerçekleştirilir. Şu anda
yalnızca AVR mîmarisinde olan mikrodenetleyicilerden ATmega328 ve
ATmega328p için geliştirilmektedir.

Micro Boot configurations.h dosyasıyla yapılandırılır.

Derleme seçenekleri, bağlama seçenekleri ve program belleğine yazma
seçenekleri Makefile dosyasıyla yapılır.

HTTP kullanıcı arayüzünü tarayıcıda görüntülemek için yapılandırmada
belirlenen bir bacağa mantıksal bir verilmesi ve ondan sonra istek
gönderilmesi gerekir.

Micro Boot, Mozilla Firefox üzerinde denenmiştir.

Makefile Dosyası:

```
#############
# Variables #
#############

# Device type (GCC)
DEVICE = atmega328p

# Clock frequency in Hz
F_CPU = 16000000

# Programmer software
PROGRAMMER_SOFTWARE = avrdude

# Programmer software parameters
PROGRAMMER_SOFTWARE_PARAMETERS = -c usbtiny -p m328p -F

# Extended fuse byte of microcontroller
EXTENDED_FUSE = efuse:w:0xff:m
```

```
# High fuse byte of microcontroller
HIGH_FUSE = hfuse:w:0xd8:m

# Low fuse byte of microcontroller
LOW_FUSE = lfuse:w:0xff:m

# Start bootloader section
START_BOOTLOADER_SECTION = 0x7000

# Linker
LD = avr-gcc

# Compiler
CC = avr-gcc

# Objcopy
OBJCOPY = avr-objcopy

# Linker flags
LDFLAGS = -O1 -DF_CPU=$(F_CPU) -mmcu=$(DEVICE) -
Ttext=$(START_BOOTLOADER_SECTION)

# Compiler flags
CFLAGS = -O1 -c -DF_CPU=$(F_CPU) -mmcu=$(DEVICE) -
Ttext=$(START_BOOTLOADER_SECTION)

# Object Files
OBJECTS = main.o standard_library/standard_library.o
platforms/megaavr/http_user_interface.o
platforms/megaavr/uart.o platforms/megaavr/wiznet_5100.o

####################
```

```
# Building Events #
###################

# Last event
all: binaries/bootloader.hex binaries/bootloader.bin

# Create bootloader.bin
binaries/bootloader.bin: binaries/bootloader.elf
     $(OBJCOPY) -j .text -j .data -O binary $? $@

# Create bootloader.hex
binaries/bootloader.hex: binaries/bootloader.elf
     $(OBJCOPY) -j .text -j .data -O ihex $? $@

# Create bootloader.elf
binaries/bootloader.elf: $(OBJECTS)
     $(LD) $(LDFLAGS) $? -o $@

# Create main object
main.o: main.c
     $(CC) $(CFLAGS) $? -o $@

# Create standard library object
standard_library/standard_library.o:
standard_library/standard_library.c
     $(CC) $(CFLAGS) $? -o $@

# Create http object
platforms/megaavr/http_user_interface.o:
platforms/megaavr/http_user_interface.c
     $(CC) $(CFLAGS) $? -o $@

# Create uart object
```

```
platforms/megaavr/uart.o: platforms/megaavr/uart.c
	$(CC) $(CFLAGS) $? -o $@

# Create wiznet_5100 object
platforms/megaavr/wiznet_5100.o:
platforms/megaavr/wiznet_5100.c
	$(CC) $(CFLAGS) $? -o $@

####################
# Installing Event #
####################

install: binaries/bootloader.hex
	$(PROGRAMMER_SOFTWARE)
$(PROGRAMMER_SOFTWARE_PARAMETERS) -U
flash:w:binaries/bootloader.hex:i

################
# Setting Fuse #
################

set-fuses:
	$(PROGRAMMER_SOFTWARE)
$(PROGRAMMER_SOFTWARE_PARAMETERS) -U $(EXTENDED_FUSE)
	$(PROGRAMMER_SOFTWARE)
$(PROGRAMMER_SOFTWARE_PARAMETERS) -U $(HIGH_FUSE)
	$(PROGRAMMER_SOFTWARE)
$(PROGRAMMER_SOFTWARE_PARAMETERS) -U $(LOW_FUSE)

####################
# Cleaning Events #
####################
```

```
# Clean all compiled and linked files
clean:
     rm -f binaries/*.* $(OBJECTS)

# Clean all compiled files
clean-objects:
     rm -f $(OBJECTS)
```

main.h Dosyası:

```
/*******************************
 * Micro Boot Main Header File *
 *******************************/

#ifdef __AVR__
   #ifdef __AVR_MEGA__
      #include <avr/boot.h>
      #include <avr/io.h>
      #include <avr/interrupt.h>
      #include <avr/pgmspace.h>
      #include <stdint.h>
      #include <stdlib.h>
      #include <string.h>
      #include <util/delay.h>
   #endif
#endif

#include "configurations.h"
#include "strings.h"

#define VERSION "0.1"  // Version information
```

```
void starting_message(void);  // Starting message
prortotype

// Standart library functions prototypes
char* integer_to_string(uint16_t, char*);
uint16_t string_to_integer(char *);
int16_t substring_index(char *, char *);

// UART functions prototpye
void uart_initialization(void);
void uart_write_character(char);
void uart_write_character_array(char *);
void uart_close(void);

// Network functions prototypes
uint8_t network_device_register_read_byte(uint16_t);
void network_device_register_write_byte(uint16_t, uint8_t);
uint8_t network_listen(void);
void network_disconnect(void);
void network_close(void);
void network_initialization(void);
void network_transmit(char *, uint16_t);
void network_receive(char *, uint16_t);
uint16_t network_received_size(void);

// Interrupt functions prototypes
void interrupt_enable(void);
void interrupt_disable(void);

// User interface prototypes
void user_interface_main(void);

#ifdef __AVR__
```

```
   #ifdef __AVR_MEGA__
      #define APPLICATION_START 0
   #endif
#endif
```

main.c Dosyası:

```
/*****************************
 * Micro Boot Main Source File *
 *****************************/

#ifndef __GNUC__
   #error The software must be compiled by GCC.
#endif

#include "main.h"

void main(void)
{
#ifdef ETHERNET_ADAPTER_ENABLE
   network_initialization();  // Ethernet adapter
inintialization
#endif

   while(SWITCH_PORT & (1 << SWTICH_PIN) == 1)
   {
      if(network_device_register_read_byte(0x0403) == 0x17)
// Wait for connection established
      {
         user_interface_main();  // User interface
      }
   }
```

```
   goto *APPLICATION_START;  // Start application
   return;
}
```

standard_library/standard_library.c Dosyası:

```
/*****************************************
 * Micro Boot Standard Library Source File *
 *****************************************/

#include "../main.h"

char* integer_to_string(uint16_t integer, char buffer[]){
    char const digit[] = "0123456789";
    char *temporary = buffer;
    int shifter = integer;

   do
   {
      temporary++;
      shifter = shifter / 10;
   }while(shifter);

    *temporary = '\0';

   do
   {
      temporary--;
      *temporary = digit[integer % 10];
      integer = integer / 10;
   }
   while(integer);
```

```
   return buffer;
}

uint16_t string_to_integer(char buffer[])
{
    uint16_t multiple = 1;
    uint16_t result = 0;
    int string_length = strlen(buffer);

    for(int i = string_length -1 ; i >= 0 ; i--)
    {
        result = result + ((int)buffer[i] - 48) * multiple;
        multiple = multiple * 10;
    }
    return result;
}

int16_t substring_index(char *string, char *substring)
{
   uint8_t i = 0;
   uint8_t string_length = 0;

   string_length = strlen(substring);

   for(i = 0; *(string + i); i++)
   {
      if(!strncmp(string + i, substring, string_length))
      {
         return i;
      }
   }

   return -1;
```

```
}
```

platforms/megaavr/uart.c Dosyası:

```
/*************************************
 * Micro Boot ATmega UART Source File *
 *************************************/

#ifdef __AVR__
   #ifdef __AVR_MEGA__

      #include "../../main.h"

      #if UART_ENABLE == 1

         #define BAUDRATE (F_CPU / UART_BAUD_RATE / 16)

void uart_initialization(void)  // UART initialization
function
{
   UBRR0H = (BAUDRATE >> 8);
   UBRR0L = BAUDRATE;
   UCSR0B |= (1 << TXEN0);
}

void uart_write_character(char character)  // UART write
character function
{
   while (!( UCSR0A & (1 << UDRE0)));
   UDR0 = character;
}
```

```
void uart_write_character_array(char *character_array)  //
UART write character array function
{
   while(*character_array != '\0')
   {
      uart_write_character(*character_array);
      character_array++;
   }
}

void uart_close(void)  // UART ending function
{
   UCSR0B |= (0 << TXEN0) | (0 << RXEN0);
}
      #endif
   #endif
#endif
```

platforms/megaavr/wiznet_5100.c Dosyası:

```
/*****************************************
 * Micro Boot ATmega Wiznet 5100 Source File *
 *****************************************/

#ifdef __AVR__
   #ifdef __AVR_MEGA__

      #include "../../main.h"

      #if ETHERNET_ADAPTER_ENABLE == 1
         #if ETHERNET_ADAPTER == DEVICE_WIZNET_5100
```

```
uint8_t network_device_register(uint16_t address, uint8_t
byte, uint8_t opcode)
{
   ETHERNET_ADAPTER_PORT_ADDRESS &= ~(1 <<
ETHERNET_ADAPTER_CHIP_SELECTION);  // CS pin enable

   // Write write command
   SPDR = opcode;
   while(!(SPSR & (1 << SPIF)));

   // Write higher bytes of address
   SPDR = (address & 0xFF00) >> 8;
   while(!(SPSR & (1 << SPIF)));

   // Write lower bytes of address
   SPDR = address & 0x00FF;
   while(!(SPSR & (1 << SPIF)));

   // Write byte
   SPDR = byte;
   while(!(SPSR & (1 << SPIF)));

   ETHERNET_ADAPTER_PORT_ADDRESS |= (1 <<
ETHERNET_ADAPTER_CHIP_SELECTION);  // CS pin disable

   return SPDR;
}

void network_device_register_write_byte(uint16_t address,
uint8_t byte)
{
   network_device_register(address, byte, 0xF0);
}
```

```
uint8_t network_device_register_read_byte(uint16_t address)
{
   return network_device_register(address, 0, 0x0F);
}

uint8_t network_listen(void)
{
   if(network_device_register_read_byte(0x403) == 0x13)
   {
      network_device_register_write_byte(0x401, 0x02);  //
Network socket listen

      while(network_device_register_read_byte(0x401));  //
Wait listening

      if(network_device_register_read_byte(0x403) == 0x14)
// If listening
      {
         return 1;
      }
      else
      {
         network_close();  // Network socket close
      }
   }

   return 0;
}

void network_disconnect(void)
{
```

```
   network_device_register_write_byte(0x0401, 0x08);  //
Write disconnect command

   while(network_device_register_read_byte(0x0401));
}

void network_close(void)
{
   network_device_register_write_byte(0x0401, 0x10);  //
Write disconnect command

   while(network_device_register_read_byte(0x0401));
}

void network_initialization(void)  // Network
initialization function
{
   uint8_t hardware_mac_address[] = {
HARDWARE_MAC_ADDRESS_1, HARDWARE_MAC_ADDRESS_2,
HARDWARE_MAC_ADDRESS_3, HARDWARE_MAC_ADDRESS_4,
HARDWARE_MAC_ADDRESS_5, HARDWARE_MAC_ADDRESS_6 };
   uint8_t source_ip_address[] = { SOURCE_IP_ADDRESS_1,
SOURCE_IP_ADDRESS_2, SOURCE_IP_ADDRESS_3,
SOURCE_IP_ADDRESS_4 };
   uint8_t subnet_mask[] = { SUBNET_MASK_1, SUBNET_MASK_2,
SUBNET_MASK_3, SUBNET_MASK_4 };
   uint8_t gateway_ip_address[] = { GATEWAY_IP_ADDRESS_1,
GATEWAY_IP_ADDRESS_2, GATEWAY_IP_ADDRESS_3,
GATEWAY_IP_ADDRESS_4 };
   uint16_t network_socket_port = USER_INTERFACE_PORT;

   // Initialization pins and SPI
   DDRB = (1 << PORTB3) | (1 << PORTB5);
```

```
   ETHERNET_ADAPTER_PORT_MODE_ADDRESS |= (1 <<
ETHERNET_ADAPTER_CHIP_SELECTION);
   ETHERNET_ADAPTER_PORT_ADDRESS |= (1 <<
ETHERNET_ADAPTER_CHIP_SELECTION);
   SPCR = (1 << SPE) | (1 << MSTR);
   SPSR |= (1 << SPI2X);

   // Master mode
   network_device_register_write_byte(0x00, 0x80);
   _delay_ms(1);

   // Write gateway IP address
   network_device_register_write_byte(0x01,
gateway_ip_address[0]);
   network_device_register_write_byte(0x02,
gateway_ip_address[1]);
   network_device_register_write_byte(0x03,
gateway_ip_address[2]);
   network_device_register_write_byte(0x04,
gateway_ip_address[3]);
   _delay_ms(1);

   // Write subnet mask
   network_device_register_write_byte(0x05,
subnet_mask[0]);
   network_device_register_write_byte(0x06,
subnet_mask[1]);
   network_device_register_write_byte(0x07,
subnet_mask[2]);
   network_device_register_write_byte(0x08,
subnet_mask[3]);
   _delay_ms(1);
```

```
   // Write Hardware MAC address
   network_device_register_write_byte(0x09,
hardware_mac_address[0]);
   network_device_register_write_byte(0x0A,
hardware_mac_address[1]);
   network_device_register_write_byte(0x0B,
hardware_mac_address[2]);
   network_device_register_write_byte(0x0C,
hardware_mac_address[3]);
   network_device_register_write_byte(0x0D,
hardware_mac_address[4]);
   network_device_register_write_byte(0x0E,
hardware_mac_address[5]);
   _delay_ms(1);

   // Write soucre IP address
   network_device_register_write_byte(0x0F,
source_ip_address[0]);
   network_device_register_write_byte(0x10,
source_ip_address[1]);
   network_device_register_write_byte(0x11,
source_ip_address[2]);
   network_device_register_write_byte(0x12,
source_ip_address[3]);
   _delay_ms(1);

   // Write buffer size settings
   network_device_register_write_byte(0x1A, 0x55);
   network_device_register_write_byte(0x1B, 0x55);
   _delay_ms(1);

   socket_start:
```

```
   network_device_register_write_byte(0x400, 0x01);  // Set
TCP mode

   network_device_register_write_byte(0x404,
(network_socket_port & 0xFF00) >> 8);  // Setting socket
port
   network_device_register_write_byte(0x405,
network_socket_port & 0x00FF);  // Setting socket port

   network_device_register_write_byte(0x401, 0x1);  //
Network socket open

   if(network_device_register_read_byte(0x403) != 0x13)  //
If not TCP mode
   {
      network_device_register_write_byte(0x401, 0x10);

      goto socket_start;
   }

   if(network_listen() == 0)
   {
      goto socket_start;
   }
}

void network_transmit(char *buffer, uint16_t buffer_size)
{
   uint16_t tx_size = 0;  // Write bytes size
   uint16_t offset_address = 0;  // Offset address of write
buffer
   uint16_t real_address = 0;  // Real address of write
buffer
```

```
   tx_size = ((network_device_register_read_byte(0x0420) &
0x00FF) << 8) + network_device_register_read_byte(0x0421);
// Read TX size

   while(tx_size < buffer_size)
   {
      _delay_ms(1);

      tx_size = ((network_device_register_read_byte(0x0420)
& 0x00FF) << 8) +
network_device_register_read_byte(0x0421);  // Read TX size
   }

   offset_address = ((
network_device_register_read_byte(0x0424) & 0x00FF) << 8) +
network_device_register_read_byte(0x0425);  // Read offset
address

   while(buffer_size)
   {
      real_address = 0x4000 + (offset_address & 0x7FF);  //
Calculate real address
      network_device_register_write_byte(real_address,
*buffer);  // Write buffer

      buffer_size--;

      offset_address++;
      buffer++;
   }
```

```
   network_device_register_write_byte(0x0424,
(offset_address & 0xFF00) >> 8);  // Write offset address
   network_device_register_write_byte(0x0425,
(offset_address & 0x00FF));  // Write offset address

   network_device_register_write_byte(0x0401, 0x20);  //
Write send command

   while(network_device_register_read_byte(0x0401));
}

void network_receive(char *buffer, uint16_t buffer_size)
{
   uint16_t offset_address = 0;  // Offset address of read
buffer
   uint16_t real_address = 0;  // Real address of read
buffer

   offset_address = ((
network_device_register_read_byte(0x0428) & 0x00FF) << 8) +
network_device_register_read_byte(0x0429);  // Read offset
address

   while(buffer_size)
   {
      real_address = 0x6000 + (offset_address & 0x7FF);  //
Calculate real address
      *buffer =
(char)network_device_register_read_byte(real_address);  //
Write buffer

      buffer_size--;
```

```
      offset_address++;
      buffer++;
   }
   *buffer = '\0';

   network_device_register_write_byte(0x0428,
(offset_address & 0xFF00) >> 8);  // Write offset address
   network_device_register_write_byte(0x0429,
(offset_address & 0x00FF));  // Write offset address

   network_device_register_write_byte(0x0401, 0x40);
   _delay_us(5);
}

uint16_t network_received_size(void)  // Calculate received
size function
{
   uint16_t value_1 = 0;
   uint16_t value_2 = 0;

   do
   {
      value_1 = (network_device_register_read_byte(0x426)
<< 8) + network_device_register_read_byte(0x0427);
      if (value_1 != 0)
      {
        value_2 = (network_device_register_read_byte(0x426)
<< 8) + network_device_register_read_byte(0x0427);
      }
   }while (value_1 != value_2);

   return value_2;
}
```

```
         #endif
      #endif
   #endif
#endif
```

platforms/megaavr/http_user_interface.c Dosyası:

```
/*****************************************************
 * Micro Boot ATmega HTTP User Interface Source File *
 *****************************************************/

#ifdef __AVR__
   #ifdef __AVR_MEGA__
      #if USER_INTERFACE_TYPE == HTTP
         #include "../../main.h"

const char strings_1[] PROGMEM = "HTTP/1.0 200
OK\r\nContent-Type: text/html\r\nContent-Length: ";
const char strings_2[] PROGMEM = "\r\n<html><head><meta
charset=\"utf-8\" /> <title>Micro Boot ";
const char strings_3[] PROGMEM = VERSION;
const char strings_4[] PROGMEM = "</title><body><h1>Micro
Boot ";
const char strings_5[] PROGMEM = "</h1><p>";
const char strings_6[] PROGMEM = STRING_FIRMWARE_TO_UPLOAD;
const char strings_7[] PROGMEM = "</p><form
enctype=\"multipart/form-data\" method=\"POST\">";
const char strings_8[] PROGMEM = "<input type=\"file\"
name=\"firmware\" /><input type=\"submit\"
/></form></html>\r\n";
```

```
const char* const strings[] PROGMEM = {strings_1,
strings_2, strings_3, strings_4, strings_3, strings_5,
strings_6, strings_7, strings_8};

int i = 0;  // Counter
uint16_t content_length = 0;  // Page content length
char buffer_1[SPM_PAGESIZE + 1] = "\0";  // First buffer
char content_length_string[6] = {'\0', '\0', '\0', '\0',
'\0', '\0'};  // Content length string
char *buffer_1_pointer;  // First buffer's pointer
char buffer_2[2] = "\0";  // Second buffer
uint16_t received_size = 0;  // Receive
uint8_t network_status = 0;  // Network status
int16_t substring = 0;  // Index of substring
char boundary[64];  // Boundary buffer
uint8_t boundary_size = 0;  //  Boundary size
uint8_t user_interface_status = 0;  // User interface
status (0 = read boundary, 1 = read content-length, 2 =
read boundary, 3 - 6 = read new line, 7 = calculate read
size, 8 = read_program)
int j = 0;  // Counter
uint8_t page = 0;  // Page counter

int8_t http_getting_value(char* method, char buffer[], char
value[], uint8_t value_length)
{
   char *pointer;  // Buffer's pointer
   char read_buffer[2];  // Read buffer
   int16_t buffer_substring;  // Buffer's substring index

   buffer_substring = substring_index(buffer, method);
   if(buffer_substring >= 0)
   {
```

```
      pointer = buffer;
      network_receive(read_buffer, 1);
      while(*read_buffer != '\r')
      {
         *pointer = *read_buffer;
         pointer++;
         *pointer= '\0';

         network_receive(read_buffer, 1);  // Read
character
      }

      strncpy(value, buffer, value_length);  // Copy value
      pointer = buffer;
      *buffer = '\0';

      received_size = network_received_size();  // Get
received size

      return buffer_substring;
   }

   return -1;
}

void clear_buffer(void)
{
   int k;  // Counter

   for(k = 0; k < SPM_PAGESIZE; k++)
   {
      buffer_1[k] = 0xFF;  // Clear buffer_1
   }
```

```
}

void boot_write_page(uint32_t page, uint8_t *buffer)
{
    int k;  // Counter
    uint16_t word;  // Word

    eeprom_busy_wait ();
    boot_page_erase (page);
    boot_spm_busy_wait ();

    for (k = 0; k < SPM_PAGESIZE; k += 2)
    {
        word = buffer[k];
        word += buffer[k + 1] << 8;

        boot_page_fill (page + k, word);
    }

    boot_page_write (page);
    boot_spm_busy_wait();
    boot_rww_enable ();
}

void user_interface_main(void)
{
   for(i = 0; i < 9; i++)
   {
      content_length += strlen_P((char
*)pgm_read_word(&(strings[i])));  // Calculate content
length
   }
```

```
   for(i = 0; i < 9; i++)
   {
      strcpy_P(buffer_1, (char
*)pgm_read_word(&(strings[i])));  // Read from program
memory
      network_transmit(buffer_1, strlen_P((char
*)pgm_read_word(&(strings[i]))));  // Transmit data

      if(i == 0)
      {
         network_transmit(integer_to_string(content_length,
content_length_string), 3);  // Transmit content length
         network_transmit("\r\n", 2);  // Transmit end line
      }
   }

   network_disconnect();

   while(1)
   {
      network_status =
network_device_register_read_byte(0x0403);  // Reading
status register

      if(network_status == 0x00)  // If socket closed
      {
         if(network_device_register_read_byte(0x0403) ==
0x00)
         {
            network_close();  // Network socket close
         }
```

```
         network_device_register_write_byte(0x0400, 0x01);
// Network open

         network_device_register_write_byte(0x0404,
((USER_INTERFACE_PORT & 0xFF00) >> 8));  // Writting TCP
port
         network_device_register_write_byte(0x0405,
USER_INTERFACE_PORT & 0x00FF);  // Writting TCP port
         network_device_register_write_byte(0x0401, 0x01);
// Write open command to command register

         while(network_device_register_read_byte(0x0401));
// Wait writting command

         if(network_device_register_read_byte(0x0403) ==
0x13)  // If socket initialzition
         {
            if(network_listen() <= 0)
            {
               _delay_ms(1);
            }
         }
      }
      else if(network_status == 0x17)  // If socket
established
      {
         received_size = network_received_size();  // Get
received size

         if(received_size > 0)
         {
            network_receive(buffer_1, 32);  // Reading
first 32 bytes of HTTP request
```

```
            substring = substring_index(buffer_1, "POST
/");
            if(substring >= 0)
            {
               *buffer_1 == '\0';
               buffer_1_pointer = buffer_1;

               while(received_size > 0)
               {
                  substring = -1;

                  network_receive(buffer_2, 1);  // Reading
character

                  if(*buffer_2 != '\r' && *buffer_2 !=
'\n')  // If not netwline
                  {
                     *buffer_1_pointer = *buffer_2;
                     buffer_1_pointer++;
                     *buffer_1_pointer = '\0';

                     if(user_interface_status == 3)
                     {
                        content_length--;  // Calculate
content_length
                     }
                     else if(user_interface_status == 4)
                     {
                        content_length--;  // Calculate
content_length
                     }
                     else if(user_interface_status == 5)
```

```
                    {
                        content_length--;  // Calculate
content_length
                    }
                    else if(user_interface_status == 6)
                    {
                        content_length--;  // Calculate
content_length
                    }
                }
                else
                {
                    if(*buffer_2 == '\r' &&
user_interface_status >= 3 && user_interface_status <= 6)
                    {
                        content_length--;  // Calculate
content_length
                    }
                    else if(*buffer_2 == '\n' &&
user_interface_status >= 3 && user_interface_status <= 6)
                    {
                        content_length--;  // Calculate
content_length
                        user_interface_status++;
                    }

                    *buffer_1 = '\0';
                    buffer_1_pointer = buffer_1;
                }

                if(user_interface_status == 0 &&
http_getting_value("Content-Type: multipart/form-data;
boundary=", buffer_1, boundary, 61) >= 0)
```

```
                {
                    user_interface_status++;  // Readed
Content-Type

                    continue;
                }

                if(user_interface_status == 1 &&
http_getting_value("Content-Length: ", buffer_1,
content_length_string, 5) >= 0)
                {
                    content_length =
string_to_integer(content_length_string);
                    if(content_length >= 0)
                    {
                        for(i = 63; i > 1; i++)
                        {
                            boundary[i] = boundary[i - 2];
// Shift boundary buffer
                        }

                        boundary[0] = '-';
                        boundary[1] = '-';

                        user_interface_status++;  // Readed
Content-Length
                    }
                    else
                    {
                        break;
                    }
                }
```

```
                substring = substring_index(buffer_1,
boundary);
                if(user_interface_status == 2 &&
substring >= 0)
                {
                   substring = -1;

                   boundary_size = strlen(boundary);
// Calculate boundary size

                   content_length -= boundary_size;  //
Calculate content_length

                   buffer_1[0] = '\0';
                   buffer_1_pointer = buffer_1;

                   user_interface_status++;

                   goto end_loop;
                }

                if(user_interface_status == 7)
                {
                   content_length -= (boundary_size +
10);  // Calculate content_length

                   user_interface_status++;

                   goto end_loop;
                }

                if(user_interface_status == 8 &&
content_length != 0)
```

```
                {
                    clear_buffer();

                    buffer_1[0] = buffer_2[j];  // Copy
character into main buffer

                    j = 1;
                    page = 0;

                    for(i = 0; i < content_length; i++)
                    {
                        network_receive(buffer_2, 1);  //
Read character
                        buffer_1[j] = buffer_2[0];  // Copy
character into main buffer

                        j++;

                        if(j == SPM_PAGESIZE || i + 1 ==
content_length)
                        {
                            boot_write_page(page, buffer_1);
// Write porgram memory
                            page += SPM_PAGESIZE;

                            clear_buffer();
                            j = 0;
                        }
                    }

                    break;
                }
```

```
                    end_loop:

                    received_size = network_received_size();
// Get received size
                }
            }

            network_disconnect();  // Network socket
disconnect
            network_close();  // Network socket close

            goto *APPLICATION_START;

        }
        else
        {
            _delay_us(10);
        }
      }
      else if(network_status == 0x18 || network_status ==
0x1A || network_status == 0x1B || network_status == 0x1C ||
network_status == 0x1D)  // If connection termination
      {
         network_close();  // Network socket close
      }
   }
}
      #endif
   #endif
#endif
```

EK 3

SINAMA AMAÇLI KULLANILAN UYGULAMA PROGRAMININ KAYNAK KODLARI

```
#include <avr/io.h>
#include <util/delay.h>

void main(void)
{
   DDRC = 0xFF;

   loop:
      PORTC = 0xFF;
      _delay_ms(5000);
      PORTC = 0x00;
      _delay_ms(1000);
   goto loop;
}
```

ÖZGEÇMİŞ

15 Aralık 1993 yılında Edirne'de doğdum. İlköğretimi ve liseyi Edirne'de tamamladım. 2012 yılında Trakya Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği bölümgünde lisans eğitimine başladım. 2014 yılında TÜBİTAK Alternatif Enerjili Araçlar Yarışması'nda Formula G kategorisinde bir takımla Trakya Üniversitesi adına katıldım. 2015 yılında Önyükleyiciler adlı bir teknik kitap yazdım. 2016 yılında bu bölümden mezun oldum. Aynı yıl Trakya Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalında yüksek lisans eğitimine başladım. Uzmanlık alanım gömülü sistemlerdir. Aynı zamanda fen alanlarına ve işletme alanlarına merkalıyım. B sınıfı sürücü belgesine sahibim.