YILDIZ TEKNİK ÜNİVERSİTESİ
ELEKTRİK-ELEKTRONİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BİLGİSAYAR PROJESİ

LİNUX ORTAMI İÇİN MOTOR KONTROL KARTI SÜRÜCÜSÜ TASARIMI

Proje Yöneticisi: Arş. Gör. Dr. Erkan USLU

Proje Grubu:
11011024 Oğuz KAHRAMAN
12011009 Hüseyin Can ERCAN

İstanbul, 2015

İÇİNDEKİLER

KISALTMA LİSTESİ

BBB Beagle Bone Black

IO Input/Output

ROS Robot Operating System

SBC Single-Board Computer

PWM Pulse-width modulation

ŞEKİL LİSTESİ

ÖNSÖZ

Başta okulumuzun değerli öğretim üyelerine ve proje yöneticimiz Dr. Erkan Uslu'ya teşekkür ederiz. Linux işletim sistemine katkı sağlamış her bireye ve topluluğa da teşekkürlerimizi sunarız.

ÖZET

Proje gömülü linux sistemlerinde donanımsal ara yüzün kernel space ve user space tarafında kullanılmasıyla alternatif donanımların kontrolüne örnek teşkil etmektedir. Arduino shell'i olarak tasarlanan Polulu motor kontrol kartı Beagle Bone Black kartı kullanılarak kontrol edilmiş ve bu kullanımı basitleştirip etkin kılacak yazılımsal altyapı hazırlanmıştır. Ayrıca proje kernel derlenmesi, kaynak allokasyonu konusunda da uygulamalar içermektedir.

Bu projeyle Polulu kontrol kartı Linux ortamında güvenli şekilde erişilebilir bir kaynak haline getirilmiştir. Bu amaca yönelik kontrol kütüphanesi hazırlanmış ve genel amaçlı io kontrolü sağlayan kernel mödülleri hazırlanarak kullanılmıştır.

Kısaca bu proje kernel ortamı, kernel derlenmesi, io kontrolü, pwm çıkışı alınması konularında bilgi içermektedir.

ABSTRACT

This project is an example of kernel space, user space hardware control software. In this project main hardwares are Beagle Bone Black and Polulu VNH5019 motor control board. Polulu board is an Arduino shield, with this software group user can control Polulu board with a proper Linux board. This project aims to make easier to use Polulu boards. Also documents contains information about kernel, kernel compilation, kernel space resource allocation.

With this project Polulu control board become a resource in Linux. We created a control library and created kernel space io control modules for Polulu control card.

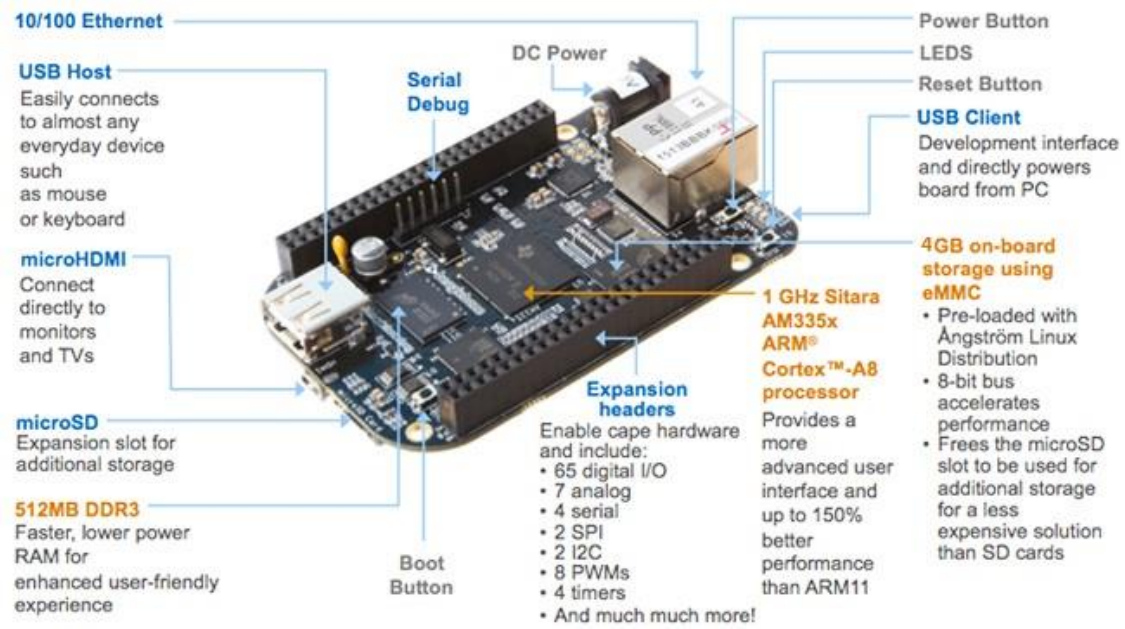
This project contains information about kernel space, kernel compilation, io control, pwm output.

1. GİRİŞ

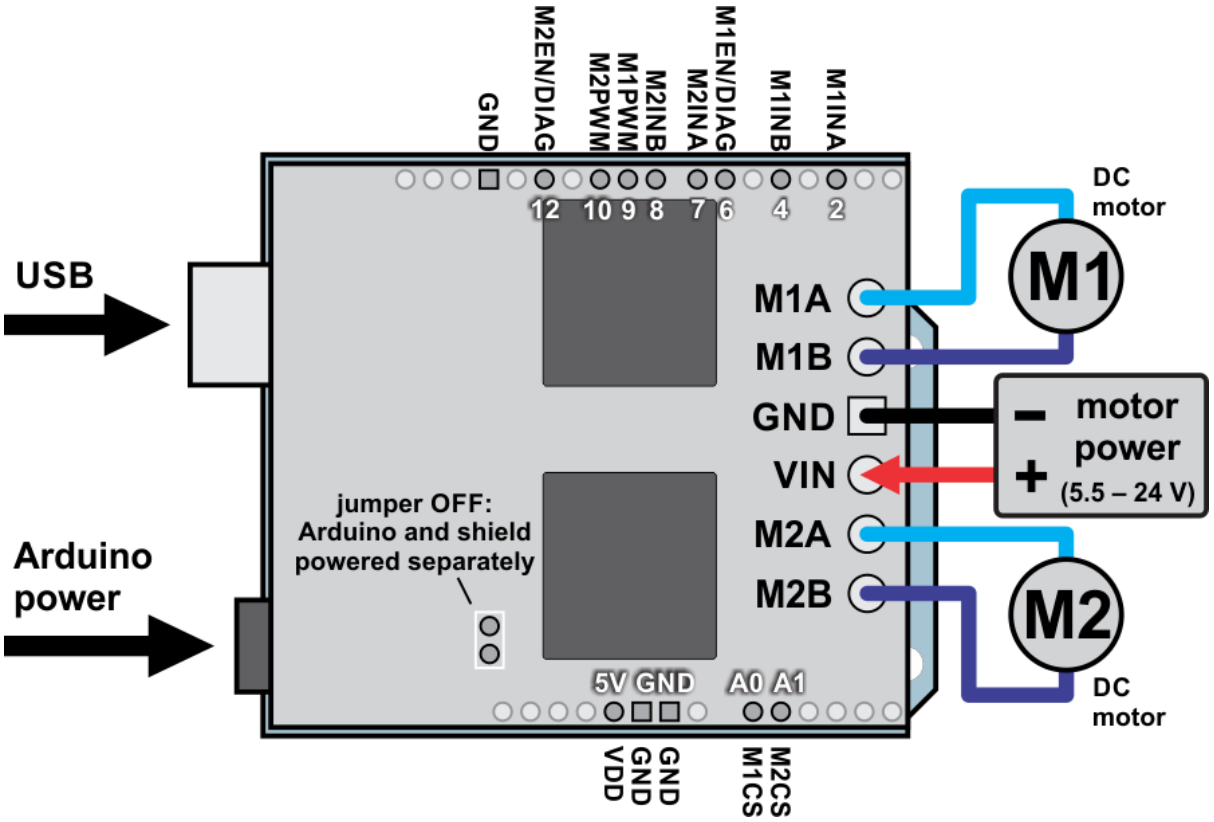
Pololu VNH5019 motor kontrol kartı için Robot Operating System (ROS) ortamında çalışacak kernel sürücüsünün gerçekleştirilmesi hedeflenmektedir. Oluşturulacak sürücünün, kernel space sürücülerinin user space sürücülerine karşı sağladığı avantajları kullanıcıya sunması planlanmaktadır. Ayrıca test imkânı sağlayıp, alternatifleri için örnek teşkil edebilecek ve genel kullanım için kullanılabilecek bir c kütüphanesinin oluşturulması da hedeflenmektedir.

Donanım ortamı olarak BeagleBone Black seçilmiştir. Kartta ethernet girişi ve 4 adet pwm çıkışı bulunması bunun sebebidir. Motor kontrol kartı ancak pwm çıkışı verebilen donanımlarla kullanılabilmektedir. Sürücü yazılımında, dış dünya interruptlarının tespiti için GPIO alt sisteminin, PWM sinyali üretilebilmesi için ise PWM ara yüzünün sağladığı imkânların kullanılması yapılan ön araştırmada planlanmıştır.

Seçilen kontrol kartının user space tarafında kullanımını kolaylaştıran ve daha hızlı cevap verebilen hale getiren sürücü yazılımı, test ve basit uygulamalarda kullanılabilecek cihaza yönelik c kütüphanesi hazırlanması amaçlanmıştır.



Şekil 1-1 Kullanmış olduğumuz BBB SBC siteminin genel özellikleri



Şekil 1-2 Kullanmış olduğumuz motor kontrol kartının genel io pinleri

2. ÖN İNCELEME

Gömülü Linux sistemlerinde geliştiriciler tarafından karşılaşılan en büyük sorun tasarlanan sistemin kullanacağı donanımın piyasada bulunmayışı veya mevcut donanım için sürücü yazılımının kullanılacak işletim sistemi için mevcut olmayışıdır.

Örneğin gömülü sistem kartlarında, normal sistemlerde yaygın olarak kullanılan usb kamera donanımlarının sistem kaynaklarına aşırı yüklenmesinden dolayı kullanılamaması bilinen bir sorundur. Bu aşamada geliştirici projesi için uygun olmayan bir kamera donanımı kullanmak durumunda kalabilir. Bu sorunun çözülebilmesi için geliştirici daha optimize çalışan, board üstündeki kamera ara yüzünü kullanan bir kamera donanımı kullanmalıdır. Fakat bu ara yüzü kullanabilecek kamera donanımları için sürücü bulması çalıştığı donanıma bağlı olarak oldukça güç ve pahalı olacaktır.

Bu şartlar altında yapılan yatırımı karşılayabilecek potansiyele sahip projeler için driver geliştirilmesi en verimli çözümü sağlayacaktır. Bu şekilde oluşturulan bir ürün rakiplerine göre çok daha yüksek verimde çalışacaktır.

Ayrıca geniş çaplı donanımların yanında geliştiricinin kendi tasarladığı daha basit donanımlar için de driver oluşturabilmesi tasarlanan donanımdan tam kapasitede verim alınabilmesi için gereklidir. Bu aşamada geliştirici user space veya kernel space sürücüsü kullanabilir. Fakat user space sürücüleri kernel space sürücülerine göre bazı yönlerden zayıftır. Interrupt servisinin tahsis edilmesi, responsive yapının sağlanması kernel space sürücüleri için çok daha kolaydır.

Tüm bunlar göz önüne alındığında belirlenen donanım için kernel sürücüsü geliştirilmesine karar verilmiştir. Bu projenin daha ileride karmaşık donanımlar için de temel teşkil etmesi planlanmıştır.

3. FİZİBİLİTE

3.1. Teknik Fizibilite

Tasarladığımız sistem, sadece kullandığımız araçlarla uyumludur. Bu sistem amacı önceden belirlenmiş ve belirlenmiş sistemler dışınca başka bir sistemle çalışma şansı yoktur. Fakat yazılmış kodlar, üstünde gerekli değiştirmeler yapılarak her türlü sisteme uygun hala getirilebilir.

Burada sistemin kendi içinde kapalı olmasının sebebi kullanılan kartların her birinin kendine özgü kod takımları ve sürücüleri olmasıdır. Bu sorunu aşmanın tek yolu ise bu sistemle aynı özelliklere sahip sistem kullanmaktır.

3.2. Zaman Fizibilitesi

Proje verilen 3 aylık süre içinde bitirilmesi için, zamanı mümkün olduğunca zamanın etkin kullanımı sağlayan gant çizelgesi çizilmiştir ve ekler kısmında EK-1 olarak belirtilmiştir.

Bu diyagramda işler efektif bir biçimde tanımlanmıştır. Bu kısımda en zor olacak kısım kernel derlenmesi kısmı olacağı için oraya verilen süre daha uzun tutulmuştur.

3.3. Yasal Fizibilite

Proje tamamen açık kaynak kodlu yazılımlar ve kullanımı serbest olan SBC(Single-Board Computer) kullanılarak geliştirileceği için yasal açıdan herhangi bir sorun içermemektedir.

3.4. Ekonomik Fizibilite

Sistem mali açıdan herhangi bir gelir beklentisi olmadan geliştirilmektedir. Bu nedenle sistemin ekonomik bir kaybı ya da kazancı beklenmemektedir.

4. SİSTEM ANALİZİ

Proje her açıdan değerlendirilip en uygun çözümler seçilmeye çalışılmıştır. Bunun için, sistemin gereksinimleri belirlenmiştir.

1- Sistem Ethernet ile kontrol edilecektir.

2- Sistemin PWM girişleri olmalıdır.

3- Sistem ROS ile uyumlu olmalıdır.

İlk olarak belirlenen gereksinimler bunlar olmuştur. Bu çerçeve de, sistem için en uygun SBC olarak BeagleBone Black seçilmiştir. Daha sonra ise bu sistem üzerine çalışacak işletim sistemi kararlaştırılmıştır. Burada ücretsiz ve açık kaynak kodlu olduğu için SBC üstüne Debian kurulmuştur. Geliştirme ortamı olarak ise Debian ile aynı çekirdek yapısını kullanan Ubuntu seçilmiştir.

Gerekli her türlü yazılım ise C ve C++ kullanılarak yapılmaktadır.

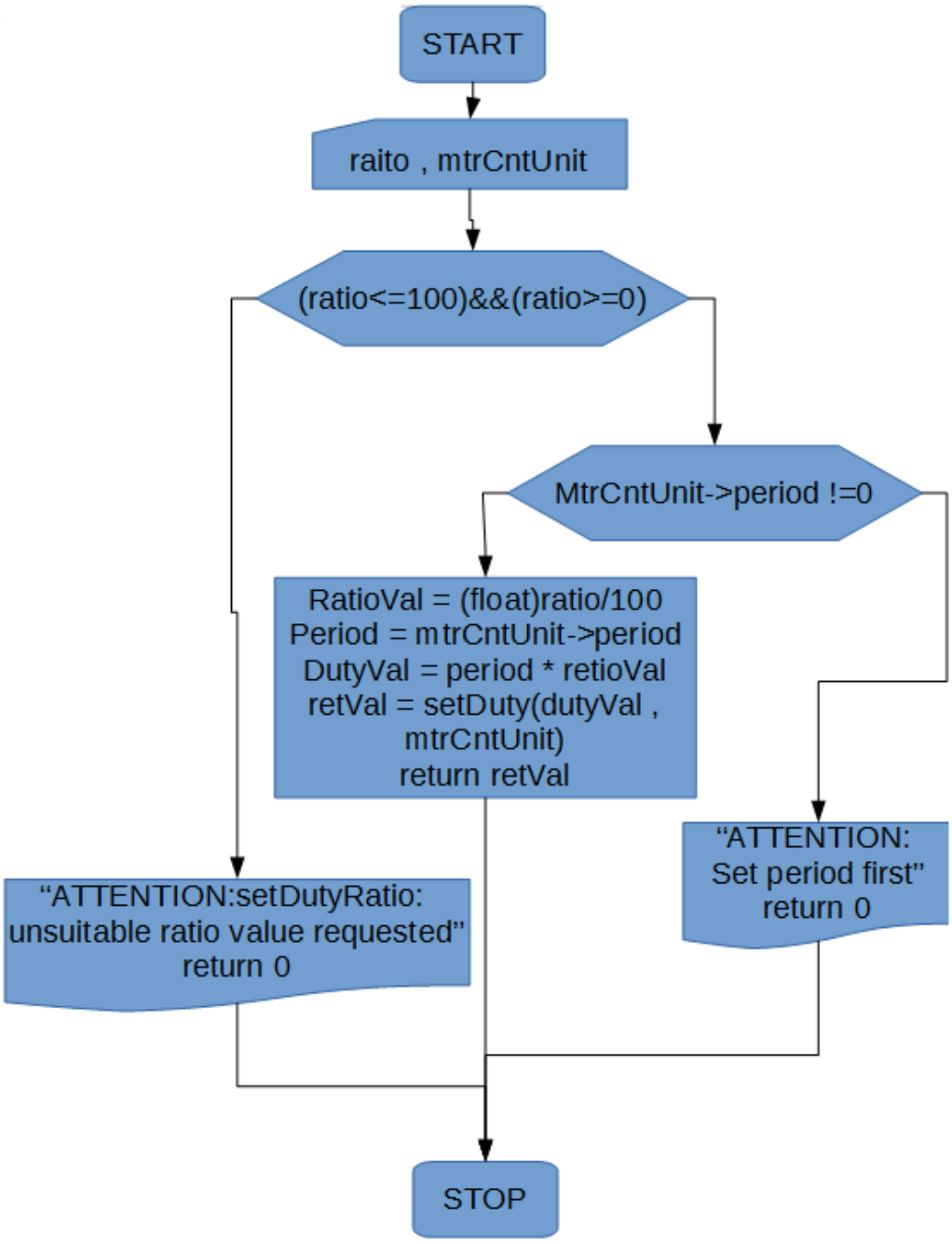
Bu gereksinimlerin belirlenmesinde yürütücü öğretim görevlisinin istekleri doğrultusunda karar verilmiş ve sonuca bağlanmıştır. Çünkü bu sistemin geliştirmesinin amacı, ortak kullanılacağı diğer sistemler ile tam uyumlu olarak çalışabilmesidir.

5. SİSTEM TASARIMI

5.1. Yazılım Tasarımı

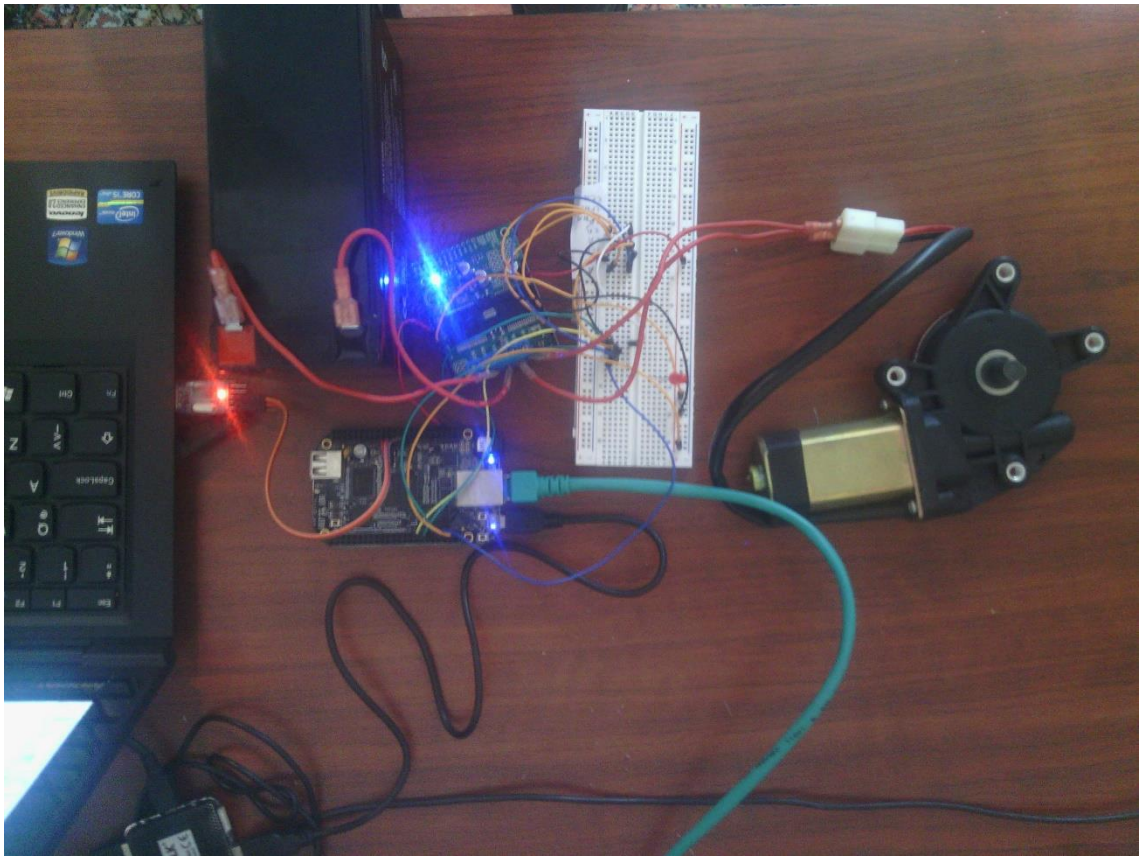
Kontrol sistemi kernel space ve user space taraflarında kullanılan yazılım parçalarından oluşmaktadır. Tasarım yolu izlenirken donanım kontrolü temel amaç olarak izlenmiş ve ağır algoritmik operasyonlardan ve io işlemlerinden olabildiğince uzak durulmuştur. Olabildiğince hızlı tepki verebilecek bir sistem oluşturulmaya çalışılmıştır, debug altyapısı kurularak runtime esnasında ve geliştirme esnasında yazılımın farklı çalışması sağlanarak yazılımın geliştirilmesi kolaylaştırılmıştır.

İleri sayfalarda bulunan algoritma kütüphanenin kullanımına örnek teşkil eden bir uygulamaya aittir. Operasyonlar temel adımlara bölünerek algoritması çizilmiştir. Kullanıcı adımları takip ederek sistemin nasıl çalıştığına dair fikir elde edebilir.



Şekil 5-1 SetDutyRatio fonksiyonunun algoritması

6. UYGULAMA



Şekil 6-1 Sistemin hazırlanmış ve gerekli bağlantılarının yapılmış şekli

6.1. Kernel Derlenmesi

İlk olarak kernelin üzerinde kullanılacağı donanımla uyumlu toolchain elde edilmelidir. Bu yazılım kerneli hedef board için derleyecek cross compiler görevini alacaktır. Söz konusu board Beaglebone Black için çok yaygın olarak kullanılan arm cross compileı kullanılacaktır.

apt-get install gcc-arm-linux-gnueabi komutuyla kullanıcı derleyiciyi elde edebilir.

lzop yazılımının elde edilmesi ileride bazı yayıncı scriptlerinin çalışması için gerekli olacak. apt-get install lzop komutuyla izop yazılımı yüklenir.

Bu aşamada boot loader yazılımı elde edilir. Bu yazılımın ile derlenmiş kernel imajları hakkında bilgi edinebileceğiz. mkimage komutuyla kullanıcı herhangi bir kernel imajının tarih, sürüm gibi bilgilerini görüntüleyebilir.

Wget ftp://ftp.denx.de/pub/u-boot/u-boot-latest.tar.bz2 komutuyla sıkıştırılmış dosya indirilir.

tar -xjf u-boot-latest.tar.bz2 indirilen dosya unzip edilir.

cd u-boot dosyaya dallanılır.

Versiyonu v1014.10 'dan büyük U-boot yazılımı için:

make sandbox_defconfig tools-only komutuyla yazılımın kullanıma uygun olarak derlenmesi sağlanır. Farklı kaynaklarda görüldüğü üzere kullanıcı bu aşamada boot loader yazılımıyla yeni bir sd kart imajı oluşturmayı amaçlıyorsa kendi kullandığı devre kartına yönelik olarak da yazılımı derleyebilmektedir. Bu doküman kapsamında bu gerçekleştirilmeyecektir.

openssl/evp.h ile ilgili bir problemle karşılaşılması durumunda sudo apt-get install libssl-dev komutu kullanılmalıdır.

sudo install tools/mkimage /usr/local/bin komutuyla derlenen yazılım yüklenerek kullanıma hazır hale getirilir.

Bu aşamada kernel indirilir ve boot edilebilir kernel imajları derlenir. Derlenecek kodun ve config dosyasının BBB resmi kaynakların temin edilmesi şiddetle tavsiye edilir.

Daha önce git sistemine kimlik tanıtımı yapmamış kullanıcıların git servisini kullanabilmek için aşağıdaki komutları terminallerinden girmeleri gerekmektedir. Aksi takdirde sistemden herhangi bir dosya indiremezler.

git config --global user.name My Name

git config --global user.email me@mydomain.net

git clone git://github.com/beagleboard/kernel.git

Komutlarıyla kullanıcı kernel sürümünü set edeceği ve indireceği script ve boarda yönelik config dosyasını elde etmiş olur.

cd kernel

git checkout 3.12

Komutlarıyla kullanıcı 3.12 ana dalı altında en son yayınlanmış alt sürümü indirmek için ayarlama yapar. İndirilebilir diğer sürümlere git://github.com/beagleboard/kernel.git sayfasından branch seçeneğiyle göz atabilir

./patch.sh

Komutuyla kullanıcı kernel kaynak kodunu indirmeye başlar. Bu adım bağlantı hızına göre vakit alıcıdır.

cp configs/beaglebone kernel/arch/arm/configs/beaglebone_defconfig

İndirme bittiğinde kullanıcı, BBB için oluşturulmuş config dosyasını derlemede kılavuz olarak kullanılacak şekilde ayarlar. Bu dosya derlenecek kernel için büyük öneme sahiptir. .config dosyaları derlenecek kerneli ciddi olarak biçimlendirirler.

wget http://arago-project.org/git/projects/?p=am33x-cm3.git\;a=blob_plain\;f=bin/am335x-pm-firmware.bin\;hb=HEAD -O kernel/firmware/am335x-pm-firmware.bin

Komutuyla önceden derlenmiş güç yönetimi yazılımı indirilir.

Dokümandaki adımlar takip edilirken kullanıcı satırların terminalde birden fazla satıra taşmadığından emin olmalıdır. Örneğin yukarıdaki gibi uzun bir komutun birden fazla satıra yayılmış sürümü dosyanın sağlıklı olarak indirilmesini engelleyecektir. Bu tip bir sorunun tespit edilmesi güçtür. Kullanıcı kaynak olarak seçtiği dokümanların veya internet sitelerinin uygun karakter formatında olduğundan ve komutlarının arasında sorunlu karakterler olmadığından emin olmalıdır. Bunun için basit bir metin editörünün kullanılarak çalışmayan komutların incelenmesi tavsiye edilir.

cd kernel

make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi- beaglebone_defconfig

Config dosyası set edilir.

make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi- uImage dtbs

uImage ve dtbs dosyaları derlenir. Bu işlem kullanılan donanım kapasitesine göre ciddi vakit almaktadır.

Bu aşamada elde edilen dosyalar Free-Electrons dokümanlarını takip edecek bir kullanıcı için yeterli olacaktır.

İlerideki adımlar geliştirmeye yönelik tam kapasiteli bir kernel elde edilmesini sağlar.

make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi- uImage-dtb.am335x-boneblack

Komutuyla kullanıcı uImage ve dtbs dosyalarının birleştirilmiş haliyle bir imaj elde eder.

make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi- modules

Komutuyla modüller derlenir. Bu aşamada derleme tamamlanmıştır.

mkimage -l uImage-dtb.am335x-boneblack

Komutuyla kullanıcı derlediği imaj hakkında bilgi elde edebilir. Bu süreç boyunca BBB için hazırlanmış derleme rehberi kullanılmıştır.

6.2. Debug Ortamının Hazırlanması

Bu adım boyunca Free Elctrons tarafından yayınlanmış Linux Kernel and Driver Development Training dokümanı kullanılarak önceki adımda derlenen kernel boot edilecek ve debug ortamı hazırlanacaktır.

İlk olarak Free Elctrons tarafından sağlanan dokümanlar ve dosyalar indirilmelidir. Bu dosyalar içerisinde ileride file system root olarak kullanılacak dosyalar da bulunmaktadır.

cd

wget http://free-electrons.com/doc/training/linux-kernel/linux-kernel-labs.tar.xz

sudo tar Jvxf linux-kernel-labs.tar.xz

sudo chown -R <user>.<user> linux-kernel-labs

Son satır dosya izinlerinin değiştirilmesi açısından önemlidir.

Bu adımda seri iletişim kurulabilir olduğundan emin olunan boot loader yazılımı board üstündeki MMC hafızaya aktarılacaktır. Bu işlem BBB üzerinde hazır olarak bulunan işletim sisteminin imajını, boot loader yazılımını ve kerneli silecektir bu nedenle

kullanıcı boardı sd ileriki süreçte MMC üzerinden boot edemeyeceğini bilerek işleme devam etmelidir. İhtiyaç halinde hazır bir sd kart imajının kullanılması tavsiye edilir.

İlk olarak bir sd kart dosyaların bulunduğu bilgisayara takılır. Kullanılan işletim sistemin virtual machine yardımıyla çalıştırılıyorsa sabit sd kart okuyuculara erişilemeyeceği dikkate alınmalıdır, bu tip ortamlarda usb sd kart okuyucu kullanılması tavsiye edilmektedir. Kart bilgisayara bağlandığında

Sabit okuyucularda '/dev/mmcblk0'

Usb okuyucularda '/dev/sdb' , '/dev/sda' benzeri isimli bir device görünür olacaktır. Bu hafıza kartlarına ait partitionlar sda1, sda2 gibi numaralandırılmış isimlerle gözükür. Kullanıcı sonraki adımda kartı formatlayacaktır. Karttaki verilerin kaybolacağı dikkate alınmalıdır.

sudo umount /dev/mmcblk0p1

sudo umount /dev/sda benzeri bir komutla kullanıcı formatlayacağı bölgenin sd kart olduğundan emin olmalıdır, farklı bir partition formatlandığından işletim sistemi geri dönülemez biçimde hasar görebilir. Bu komut çalıştırıldığında sd kart ulaşılamaz hale gelecektir, kullanıcı bu şekilde device ismi konusunda emin olabilir.

Uç bir ihtimal olarak sda1 partition'ı da umount edilebilmektedir, ama bu durumda kullanıcı file browser ile hala sd kartın erişilebilir olduğunu görecektir.

sudo sfdisk --in-order --Linux --unit M /dev/mmcblk0 << EOF

1,48,0xE,*

,,,-

EOF

sudo mkfs.vfat -F 16 /dev/mmcblk0p1 -n boot

Komutuyla formatlama ve partition oluşturma işlemleri gerçekleştirilir. Sd kart sökülüp takıldığında artık boot isimli bir partition file browserda görünür hale gelecektir.

cp am335x-boneblack.dtb MLO MBR u-boot.img MLO.final u boot.img.final uEnv.txt uImage /media/$USER/boot

Komutuyla sd kartı , BBB üzerinde boot edilebilir hale getirmiş oluruz. Bu dosyaların hepsi ilk aşamada indirilen Free Electrons dokümanları içerisinde bulunmaktadır.

/home/user/linux-kernel-labs/bootloader/beaglebone-black

BBB üzerinde boot edilebilir hale getirmiş oluruz. Bu dosyaların hepsi ilk

aşamada indirilen dokümanlar içerisinde yukarıdaki uzantı içerisindedir , daha ayrıntılı bilgi aynı uzantıdaki README.txt'de mevcuttur.

Sd kart boarda yerleştirilir, boot düğmesine basılır haldeyken boarda elektrik verilir. 20-30 saniye içerisinde MMC memorye flash işlemi tamamlanmalı ve kart kullanılan uImage ile başlamalıdır. Bellek işlemi bittiğinde 4 led de yanar.

Bu aşamada kernel dosyaları boarda aktarılarak, host makina üzerindeki bir file system ile boot edilecektir. Bu aşamada kullanıcı board üzerindeki debug ara yüzünü kullanabileceği bir kabloya ihtiyaç duymaktadır. Tavsiye edilen çeşitli kablolar vardır. Kabloların bulunamaması durumunda PL2303 entegresi bulunduran alternatif cihazlar da kullanılabilir. Arduino ve benzeri kartlarda da debug için kullanılan cihazlar istenilen iletişimi sağlayacaktır.

Usb cihaz host olarak kullanılacak makinaya takıldığında kablo usb bir cihaz olarak tanınmalıdır. Kullanıcı USB-TTL kablonun veya boardın TX , RX , GND uçlarını BBB üzerindeki debug konsoluna bağlar. Debug konsolunun TX ve RX uçları işe kablonun TX ve RX uçlarının ters bağlanmasına dikkat edilmelidir.

Debug uçlarının yakın olduğu pin girişi serisi yukarı tutularak bakıldığında, soldan 2. ve 3. uçlar TX-RX, 6. uç ise GND'nin bağlanacağı uçlardır.

Kullanıcı bundan sonra seri iletişim için kullanacağı programı kurar.

sudo apt-get install picocom

sudo adduser $USER dialout

Komutu ile kendini bu seri iletişim ara yüzünün kullanıcıları arasına ekler. Logout, login olduktan sonra picocom kullanıcının erişimine girecektir.

picocom -b 115200 /dev/ttyUSB0

Komutuyla artık board ile iletişim kurabiliriz. Fakat iletişim entegresinin birden çok usb cihaz ile kullanılmaması, usb hub aracılığıyla kullanılmaması gereklidir. Device isminin değişmesine sebep olacak uygulamalar komutun çalışmasını önler.

Artık kullanıcı boarda güç vererek boot işlemini gözlemleyebilir. Boot esnasında, boot işlemini durdurarak boot loarder yazılımına erişebilir.

Bu aşamada daha önce derlediğimiz kernel imajlarını boarda tftp server yardımıyla indirip, host makina üzerindeki bir file system ile başlatacağız.

sudo apt-get install tftpd-hpa

Komutuyla kullanacağı tftp yazılımını elde eder. Daha sonra ethernet bağlantısı

sağlanabilmesi için host olarak kullanılacak makinada bir bağlantı tanımlar. Static ip ayarlar , 192.168.0.1. Netmask ayarlar 255.2555.255.0.

U-boot'a eriştiği terminal yardımıyla;

setenv ipaddr 192.168.0.100

setenv serverip 192.168.0.1 komutlarını girerek boardın, dosyaların bulunduğu makinaya erişmesini sağlar.

Kullanıcı boarda indirmek istediği dosyaları /var/lib/tftpboot klasörüne kopyalar. İlk aşamada az sayıda karakterden oluşan bit txt dosyasının indirilerek test edilmesi tercih edilmelidir. Bu klasör tftp kurulumundan sonra yüksek izinle kurulmuş olabilir. Bu durum boot loarder terminalinde permission denied ile kendini gösterecektir. Kullanıcı chmod komutuyla dosya iznini değiştirirse engel kalkacaktır.

tftp 0x81000000 samplefile.txt

Komutuyla örnek dosya belirtilen adrese indirilir.

md 0x81000000

İşletim sistemi her başlatıldığında TFTP bağlantısı kurulmadan önce

sudo service tftpd-hpa restart komutuyla yeniden başlatılması gereklidir. Bu Ubuntu 12.04 ve sonrasında görülen bir sorundur.

Komuyutla gönderilen dosya board üzerindeki memoryde görüntülenebilir.

Şimdiki işlerler ile NFS server kurulumu yaparak file systemin çalışmasını sağlayacak ayarlamalar yapılacaktır.

sudo apt-get install nfs-kernel-server komutuyla NFS server kurulur. /etc/exports dosyasına root yetkisyle aşağıdaki satır eklenir.

/home/<user>/linux-kernel-labs/modules/nfsroot

192.168.0.100(rw,no_root_squash,no_subtree_check)

make LOADADDR=0x80008000 uImage

Kullanıcı daha önce derlediği imaj dosyalarını adrese yönelik set etmelidir.

Bundan sonraki aşama kernel dosyalarının kopyalanarak, boot edilmelerini içermektedir.

tftp 0x81000000 uImage

tftp 0x82000000 am335x-boneblack.dtb

bootm 0x81000000 – 0x82000000 komutlarıyla her şey yolunda giderse kullanıcı bir login ekranıyla karşılaşacaktır. Şifre 'root'.

6.3. Modül ve Örnek C Uygulaması

Bu adımda kullanıcı test amaçlı olarak bir io driverı ve bu driverı kullanan C kodu derleyecek ve çalıştıracaktır.

```
ifneq ($(KERNELRELEASE),)
obj-m := bbb-gpio-2.o
else
KDIR :=
/home/galadriel/Desktop/Carry_To_Thinkpad/Beagle_Board_Official/kernel/ker
nel
all:
$(MAKE) -C $(KDIR) M=$$PWD modules
endif
```

Modülü derleyebilmek için yukarıdaki Malefile'a benzer bir dosyaya ihtiyaç vardır. KDIR: derlemiş olduğumuz kerneli işaret ederobj-m: derlemek istediğimiz modülün ismi, kaynak kodun ismidir.

Kullanıcı örnek modül kodunun ve Makefile'ın bulunduğu uzantıda

make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi-

Komutunu çalıştırarak modülü derleyebilir. Çıktı olarak .ko uzantılı dosyayı insmod komutuyla kernele dahil edecektir. Dosyanın BBB üzerinde kullanılabilir olması için işaret edilen file system içerisinde olması gereklidir. Burada kurduğumuz debug imkanlarıyla derlediğimiz modül ve uygulamaları aynı makina üzerinden derleyebilir ve test edebiliriz.

insmod bbb-gpio-2.ko

Örnek olarak bu komuta benzeyecektir.

Son olarak modülü kullanan C kodu derlenir ve çalıştırılır. Kodun cross compiler ile derlenmesi yeterli olacaktır. Ardından file systeme atılarak

.\aaa.out benzeri bir komutla program BBB üzerinde çalıştırılabilir.

Kullanıcı rmmod komutuyla driver yazılımını kernelden runtime esnasında çıkarabilir. Bu imkân kolaylık sağlayarak kodun test edilme hızını arttıracaktır.

6.4. PWM Sinyali Çıkışı Alınması ve Test Amaçlı Yazılımın Gerçeklenmesi

6.4.1. Pwm Sinyali Çıkışı

EXPORT NUMBER	PIN NAME	PINS
0	EHRPWM0A	P9.22,P9.31
1	EHRPWM0B	P9.21,P9.29
2	ECAPPWM0	P9.42
3	EHRPWM1A	P9.14,P8.36
4	EHRPWM1B	P9.16,P8.34
5	EHRPWM2A	P8.19,P8.45
6	EHRPWM2B	P8.13,P8.46
7	ECAPPWM2	P9.28

Şekil 6-2 BBB PWM pin grupları

Tablolarda görülen pwm çıkışı pinlerinin kullanılabilmesi için ilk olarak device kerneline eklenmeleri gerekmektedir. Beagle Bone B. gibi kartlarda device treenin statik bir yapı olması kullanıcıya sorun çıkarmıştır. Kullanıcı, donanımında yaptığı bir değişiklik için veya karttaki pinlerin kullanımını değiştirmek istediği her seferde kerneli tekrar derlemek durumunda kalıyordu. Bunun önüne geçebilmek için capemanager geliştirilmiştir.

Generic deviceler haricindeki, SoC(system on chip) veya daha basit donanımlar varlıkları hakkında bağlandıkları donanımı bilgilendiremezler. Usb bir cihaz kendini tanıtabilir fakat basit bit sıcaklık ölçüm sensörü bu kapasiteye sahip değildir. Bu durum tam kapasiteli sistemler için bir sorun teşkil etmez. Fakat Beagle Bone geliştirme amaçlı üretilmiş ve kullanılan bir kart ve kullanıcının küçük çaplı donanımları kullanabilmesi kartı almasındaki yegâne sebeplerden biri. Bu sebeple kernel tarafında çalışacak capemanager yazılımı ortaya konarak, kullanıcının istediği donanımı runtime esnasında karta bağlayabilmesi sağlanmıştır.

Pwm çıkışı alınabilmesi için ilk olarak gerekli cihazın kernele dâhil edilmesi gereklidir. Capemanager ilk olarak dâhil edilen cihazın ismine yönelik bir yönlendirme olup olmadığını kontrol eder. Bulunursa dâhil edilen cihaz için özel hazırlanmış dtbo dosyası kullanılır. Bu dosya özelliklerinin belirlenmesi konusunda önemlidir. En önemlisi

de cihazın birlikte çalışacağı sürücü isimleri bu dosyada belirtilir.

6.4.2. Kernel Versiyon Düşürme

Bundan önceki çalışmalarda Beagle Bone B. üzerinde 3.12 versiyonu kullanılıyorken yapılan testlerde kernelin sorunlu olduğu, kernelin bir kısmının daha önceki versiyonlara ait olduğu anlaşılarak 3,8 versiyonlu kernel kullanılmaya başlanmıştır.

[116.588550] bone-capemgr bone_capemgr.4: part_number 'DM-GPIO-Test', version 'N/A'
[116.599728] bone-capemgr bone_capemgr.4: slot #7: generic override
[116.606427] bone-capemgr bone_capemgr.4: bone: Using override eeprom data at slot 7
[116.614577] bone-capemgr bone_capemgr.4: slot #7: 'Override Board Name,00A0,Override Manuf,DM-GPIO-Test'
[116.628648] bone-capemgr bone_capemgr.4: slot #7: Requesting part number/version based 'DM-GPIO-Test-00A0.dtbo
[116.639361] bone-capemgr bone_capemgr.4: slot #7: Requesting firmware 'DM-GPIO-Test-00A0.dtbo' for board-name 'Override Board Name', version '00A0'
[116.659966] bone-capemgr bone_capemgr.4: slot #7: dtbo 'DM-GPIO-Test-00A0.dtbo' loaded; converting to live tree
[116.673128] of_resolve: Could not find symbol 'ocp'
[116.678438] bone-capemgr bone_capemgr.4: slot #7: Failed to resolve tree

Bu operasyonda da görülebilen 'ocp' sembolünün bulunamaması bir sonraki başlıkta anlatılacak device tree işlemlerinin başarısızlıkla sonuçlanmasına neden olmuştur. Bu yüzden kernel sürüm düşürülmüştür.

6.4.3. Platform Device

İstenilen device eklendikten sonra kernel bu device için bir sürücü arayacaktır. Aşağıdaki dosya P8_13 için hazırlanmış pwm dosyasıdır.

```
        bone_pwm_P8_13-00A0.dts
/*
 * Copyright (C) 2013 CircuitCo
 * Copyright (C) 2013 Texas Instruments
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License version 2 as
 * published by the Free Software Foundation.
 */
/dts-v1/;
/plugin/;

/ {
      compatible = "ti,beaglebone", "ti,beaglebone-black";

      /* identification */
      part-number = "sc_pwm_P8_13";
      version = "00A0";

      /* state the resources this cape uses */
      exclusive-use =
            /* the pin header uses */
            "P8.13",              /* pwm: ehrpwm2B */
            /* the hardware IP uses */
            "ehrpwm2B";

      fragment@0 {
            target = <&am33xx_pinmux>;
            __overlay__ {
                  pwm_P8_13: pinmux_pwm_P8_13_pins {
                        pinctrl-single,pins = <0x024  0x4>; /* P8_13 (ZCZ ball T10) | MODE
4 */
```

```
            };
        };
    };

    fragment@1 {
        target = <&ocp>;
        __overlay__ {
            pwm_test_P8_13 {
                compatible      = "pwm_test";
                pwms            = <&ehrpwm2 1 0 1>;
                pwm-names       = "PWM_P8_13";

                pinctrl-names   = "default";
                pinctrl-0       = <&pwm_P8_13>;

                enabled         = <0>;
                duty            = <0>;
                status          = "okay";
            };
        };
    };
};
```

Fragment kısmı içerisinde görülebileceği gibi uyumlu olduğu platfrom driverın ismi daha önceden belirlenmiş durumdadır. Bu isim geliştirici tarafından değiştirilerek veya farklı bir isim eklenerek kernel tekrar derlenirse kullanıcı ismini kendisinin belirlediği bir sürücüyle cihazı kontrol edebilir. Veya “pwm_test” isimli sürücünün kodlarını değiştirerek de kendi driverını kullanabilir. Fakat iki yöntem de taşınabilir değildir. İki yöntem de sadece kullanıcının şahsi ortamında kabul görebilecek modifikasyonlardır. Ayrıca Texas Instruments tarafından sağlanan bu sürücü sadece test amaçlı bir yazılım olmasına karşın isminin dtbo dosyalarında yer alması durumu ilginç bir pozisyona getirmektedir.

Kullanıcının kernel dosyalarında yaptığı değişiklikler pyton kütüphaneleri gibi bazı 3. kullanıcıların artık çalışmama riskini ortaya çıkarmaktadır.

Bu aşamada kullanıcı tasarım konusunda bir seçim yapmalıdır. Eğer mevcut driver yazılımı yeterli değilse kernel tarafında bir değişiklik yapmak zorundadır. Bunun en güvenli yolu da kendine özel .dtbo dosyaları oluşturmasıdır.

Kullanıcı kullandığı kerneli derlemeden önce "pwm_test" sürücüsünü kernele dâhil edebilir veya runtime esnasında kernele dâhil edebilir. İki koşulda da sürücü yazılımı, device tree değiştirilmeden önce yapılmalıdır aksi takdirde pwm sinyalinin özelliklerini belirleyen attribute dosyaları oluşmayacaktır. Bu dosyalar sinyalin özelliklerini belirleyen parametreleri tutmaktadır.

6.4.4. Pwm_test

Kullanıcı derleyerek kullandığı kernel kaynak kodunun içerisinde kernel/drivers/pwm klasörü altında "pwm-test" ismiyle kullanılacak sürücünün kaynak kodlarına erişebilir. Bu kodlar kullanılan kernelde çalışacak biçimde bir önceki raporda anlatıldığı gibi derlenmeli ve insmod komutuyla kernele dâhil edilmelidir.

Modül yüklendikten sonra pwm pin kontrolü yapılır.

Driver yazılımının insmod komutuyla kernele eklenmesi son üç komutun çalışabilmesi için zorunludur.

```
echo am33xx_pwm >  /sys/devices/bone_capemgr.9/slots
echo bone_pwm_P8_13 > /sys/devices/bone_capemgr.9/slots
echo 1000000 > /sys/devices/ocp.3/pwm_test_P8_13.15/period
echo 500000 > /sys/devices/ocp.3/pwm_test_P8_13.15/duty
echo 0 > /sys/devices/ocp.3/pwm_test_P8_13.15/polarity
```

Capemanager sürümü ve ocp numarası kernelden kernele farklılık gösterebilir.

Bu komut serisiyle kullanıcı ilk testi gerçekleştirebilir. P8_13 pini üzerinde 1000000 nano saniye periyotlu yarı zamanda pozitif sintal elde edilecektir. Polaritenin 0 olarak set edilmemesi durumunda planlanan sinyalin tersi elde edilecektir. Kullanıcı kullandığı kernelin özelliklerine göre bazı pinlerden pwm sinyali çıkışı alamayabilir. Bazı pinler farklı kaynaklar için ayrılmış olabilir. Bu durumda kullanıcı farklı pinleri pwm çıkışı olarak kullanabilir. Dikkat edilmesi gereken bir etken de ilk

sayfada yer alan tabloda görüldüğü üzere aynı çıkışı veren pin uçlarının farklı frekanslarda set edilemeyeceğidir.

6.4.5. Yazılım İle Kontrol

Kullanıcı önceki parçada anlatıldığı gibi terminal aracılığıyla gerçekleştirdiği pin kontrolünü, driver tarafından oluşturulan dosyalara uygun bir programlama diliyle de ulaşabilir. Normal dosya operasyonlarına benzer şekilde okuma ve yazma yapabilir. Önemli olan driverın nasıl çalıştığının incelenmesidir. Driver set edilen parametreleri karakter dizileri olarak okumaktadır.

Ayrıca cihazım belirli frekans aralığında ve duty uzunluğunun sinyal periyodundan uzun olamayacağı da dikkate alınmalıdır.

Projenin ana amacı Polulu Dual VNH5019 motor kontrol kartı için sürücü yazılması olduğu için planlanan ilk test yazılımında pwm çıkışı alınmasının yanında motor kontrolünde önemi olan 3 pin için statik bir yapı gösteren gpio kontrolüne yönelik driver da gerçeklemiştir.

3 kontrol pini her motor için saat yönünde dönmeyi, saat yönünün tersinde dönmeyi ve motora güç verilmesini kontrol etmektedir. Bu koşullara yönelik olarak ilk örnek sürücüde bu üç pinin kontrolü için ioctl'e dayalı bir kontrol mekanizması gerçeklemiştir. Bu sürücü ile kullanıcı önceden belirlenmiş anlamlı mesajlar yardımıyla motora istediği sinyallerin önceden belirlenmiş pinler üzerinden aktarılmasını sağlayabilecektir. Tasarımda güvenlik ve donanımsal koşullar dikkate alınarak pinlerin ve sürücüye file sistemde erişilecek ismin önceden statik olarak belirlenmesine karar verilmiştir. Bu kısıtlayıcı bir seçim olmasına karşın güvenlidir.

Kullanılan kernel ayrıntılı biçimde incelenmediği sürece kullanıcı hangi pinin ne için kullanıldığını veya kullanılamayacağını tahmin edemez. Gpio olarak kullanılamayacak bir pin serisinin istenmesi, mevcut kullanımda olan bir kaynağın veya pinin istenilmesi kernel için normal şartlarda bir sorun oluşturmamasına karşın kullanıcı tarafında sorun ortaya çıkaracaktır. Bu nedenle kullanıcı tarafının biraz daha güvenli olabilmesi için özgürlükten feragat edilerek statik bir tasarım seçilmiştir.

Daha sonra gerekli yazılım kodları yazılmıştır. Bu kodların önemli kısımları ekler kısmında EK-2 olarak verilmiştir.

7. DENEYSEL SONUÇLAR

Proje geliştirme aşamasında gerçekleştirilen testlerde kullanıcının kendisi tarafından derlenmemiş kernel kullanması halinde mevcut device tree hakkında bilgisi olmaması halinde istenilen kaynakların allokasyonunda sorun yaşayabilmektedir. Kernel hali hazırda kullanımda bulunan kaynakları tahsis etmeyerek anlaşılamayan problemler yaşanmasına sebep olmaktadır. Bunu önleyebilmek için elimizden geldiğinde debug altyapısının detaylı hazırlanmasına özen gösterdik. Uç ihtimallerde gerçekleşecek problemler için mesajlar hazırladık. Yine de yazılımın taşınabilirliği kısıtlı sağlanabildi - , kullanıcının kendi hazırladığı ve hâkim olduğu kerneller üzerinde yazılım parçalarını çalıştırması daha sağlıklı çalışan bir sistem oluşturacaktır.

8. SONUÇ

Gerçekleştirilen çalışmalar sonunda projenin yazılım tasarımının nasıl yapıalcağı ve gerekli tasarımların yapılmasına karar verilmiştir ve bu çerçevede çalışmalara başlanılmıştır. Bu aşamalar esnasında kernelin temin edilerk derlenmesi, kernel space tarafında çalışacak modüllerin tasarlanması veya açık kaynaklı modül kaynaklarının temin edilmesi, bu modüllerin user space tarafında kontrolünü sağlayacak yazılım kütüphanelerinin tasarlanması adım adım takip edilen aşamalardır.

Bu konuda çalışmak isteyen kişiler öncelikle temel Linux özellikleri hakkında bilgiye sahip olmalıdır. Kernelin derlenmesi, kernel space modüllerinin derlenmesi, user space tarafında çalışacak aplikasyonların derlenmesi hemen hemen her adımda kullanıcının önüne çıkan engellerdir.

Ayrıca bu ve benzeri konular teknik riski yüksek olduğu için kullanıcı seçtiği donanımların ön incelemesini ayrıntılı yapmalıdır.

KAYNAKLAR

[1] Free Electron (2015). Linux kernel and driver development training. [Online]. Available: http://free-electrons.com/doc/training/linux-kernel/linux-kernel-labs.pdf

[2] Elinux (2015). Building BBB Kernel. [Online]
Available: http://elinux.org/Building_BBB_Kernel

[3] Ranjeet Mishra (2015). Implementing a Simple Char Device in Linux. [Online]. Available: http://linuxgazette.net/125/mishra.html.

[4] Elinux (2015). BeagleBoard:BeagleBone Black Accessories. [Online]
Available: http://elinux.org/Beagleboard:BeagleBone_Black_Accessories

[5] Xavier Calbet(2006). Writing Device Drivers in Linux:A Brief Tutorial. [Online]. Available: http://www.freesoftwaremagazine.com/articles/drivers_linux

[6] Jonathan Corbet, Alessandro Rubini, Greg Kroah-Hartman (2005). Linux Device Drivers (3rd Edition). [Online].
Available: http://www.oreilly.com/openbook/linuxdrive3/book/

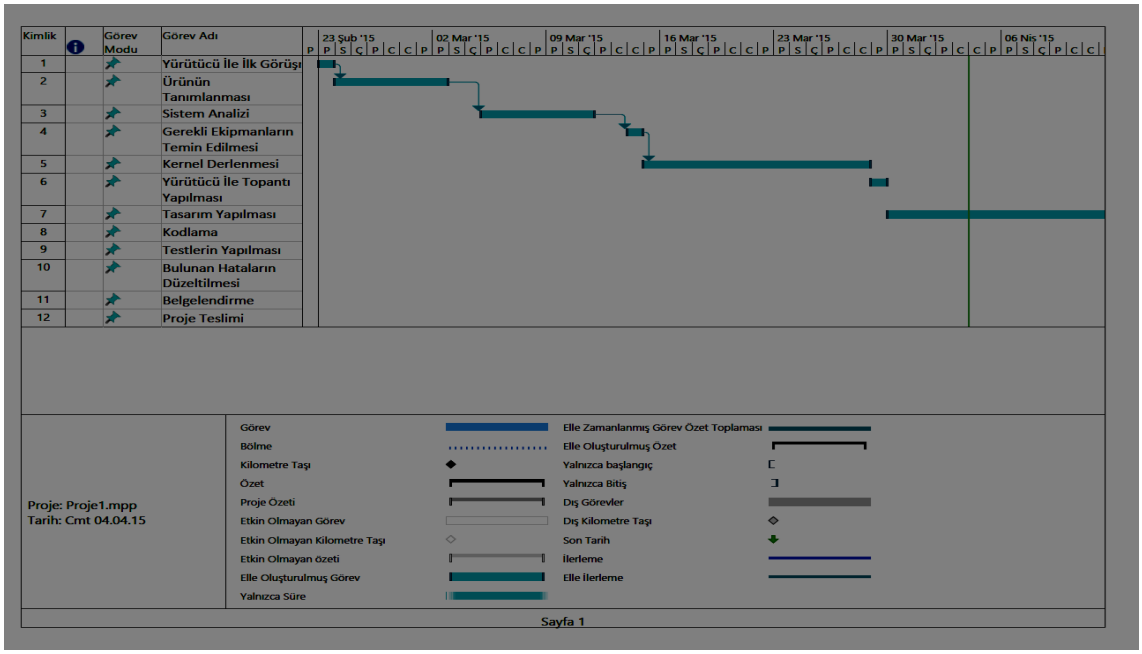
[7] Fedoraproject (2015). Building a custom kernel. [Online].
Available: https://fedoraproject.org/wiki/Building_a_custom_kernel

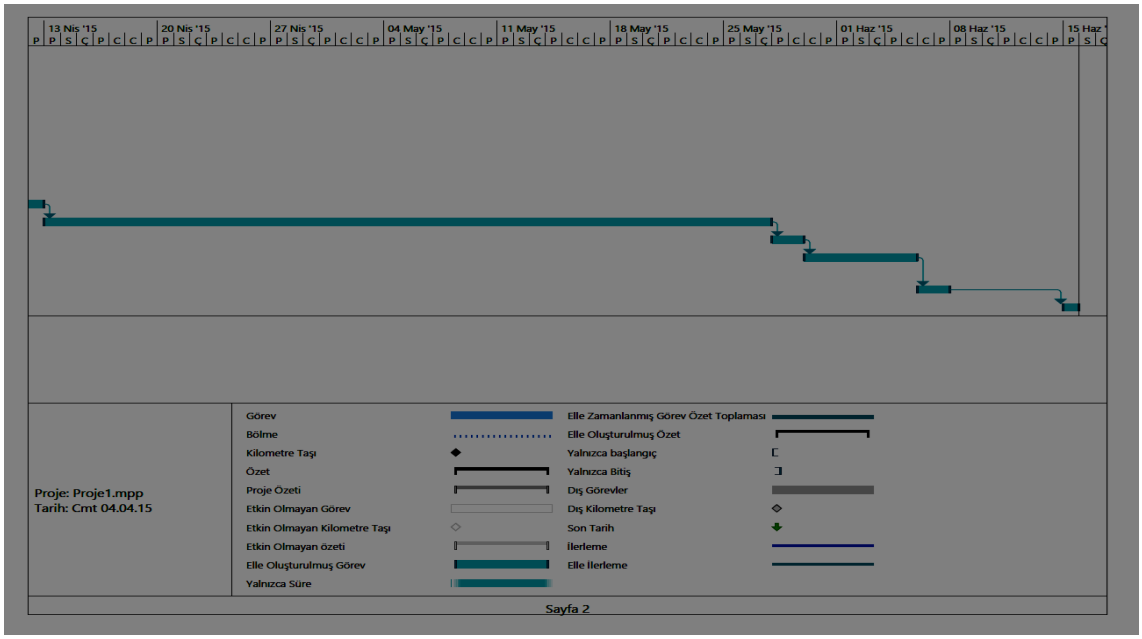
[8] Tobias Wolfgang Andreas Müller (2014). U-Boot on BeagleBone Black. [Online]. Available: http://www.twam.info/hardware/beaglebone-black/u-boot-on-beaglebone-black

[9] Sanchayan Maity (2014). How to Write a Platform Device/Driver PWM Driver for Tegra2. [Online]. Available: https://coherentmusings.wordpress.com/2014/01/19/how-to-write-a-platform-devicedriver-pwm-driver-for-tegra2/

EKLER

EK-1


Kimlik
Görev Modu
Görev Adı
23 Şub '15
02 Mar '15
09 Mar '15
16 Mar '15
23 Mar '15
30 Mar '15
06 Nis '15
1 Yürütücü İle İlk Görüşü
2 Ürünün Tanımlanması
3 Sistem Analizi
4 Gerekli Ekipmanların Temin Edilmesi
5 Kernel Derlenmesi
6 Yürütücü İle Topantı Yapılması
7 Tasarım Yapılması
8 Kodlama
9 Testlerin Yapılması
10 Bulunan Hataların Düzeltilmesi
11 Belgelendirme
12 Proje Teslimi
Proje: Proje1.mpp
Tarih: Cmt 04.04.15
Görev
Bölme
Kilometre Taşı
Özet
Proje Özeti
Etkin Olmayan Görev
Etkin Olmayan Kilometre Taşı
Etkin Olmayan özeti
Elle Oluşturulmuş Görev
Yalnızca Süre
Elle Zamanlanmış Görev Özet Toplaması
Elle Oluşturulmuş Özet
Yalnızca başlangıç
Yalnızca Bitiş
Dış Görevler
Dış Kilometre Taşı
Son Tarih
İlerleme
Elle İlerleme
Sayfa 1


13 Nis '15
20 Nis '15
27 Nis '15
04 May '15
11 May '15
18 May '15
25 May '15
01 Haz '15
08 Haz '15
Proje: Proje1.mpp
Tarih: Cmt 04.04.15
Görev
Bölme
Kilometre Taşı
Özet
Proje Özeti
Etkin Olmayan Görev
Etkin Olmayan Kilometre Taşı
Etkin Olmayan özeti
Elle Oluşturulmuş Görev
Yalnızca Süre
Elle Zamanlanmış Görev Özet Toplaması
Elle Oluşturulmuş Özet
Yalnızca başlangıç
Yalnızca Bitiş
Dış Görevler
Dış Kilometre Taşı
Son Tarih
İlerleme
Elle İlerleme
Sayfa 2

EK-2

Texas Instruments pwm kontrol sürücüsü:

İnceleme:

Probe fonksiyonu device tree'ye pwm pini eklendiği sırada devreye girer.

```
static struct of_device_id pwm_test_of_match[] = {
{ .compatible = "pwm_test" },
{ }
 };
```

Kod parçasında da görülebileceği gibi uyumlu device ismi önceden set edilmiş halde.

Attribute grubu ve attribute erişim fonksiyonları nesneye yönelik bir yaklaşım gösteriyor.

```
pwm_test->pwm = devm_pwm_get(&pdev->dev, NULL);
if (IS_ERR(pwm_test->pwm)) {
        dev_err(dev, "unable to request PWM\n");
        return PTR_ERR(pwm_test->pwm);
}
```

Kod parçasında kullanılan devm_pwm_get(&pdev->dev, NULL); fonksiyonu parametre olarak NULL kullanılırsa mevcut pwm çıkışı verebilen kaynaklara ait bir listeden seçim yaparak kaynak ayırımı gerçekleştirmektedir. Bu ve benzer özellikler yazılımın taşınabilirliğini oldukça arttırmaktadır. Capemanager olmadığı takdirde kullanıcının her board için ayrı bir driver yazması kaçınılmaz olacaktır.

Motor kontrol kartı içim gpio kontrolüne yönelik sürücü yazılımı:

Bu yazılımda önceki gpio sürücüsü yazılımından farklı olarak anlamlı measjların kernel tarafına iletilebilmesi için önceden belirlenmiş ioctl mesajları kullanılmıştır. Bu yöntem sürücünün sağlıksız davranmasını engelleyerek write ve read gibi fonksiyonların amacı dışında kullanılmasını engellemektedir.

ÖZGEÇMİŞ

Ad Soyad	: Oğuz Kahraman
Doğum Tarih	: 16/03/1994
Doğum Yeri	: Gürgentepe
Lise	: Ordu Anadolu Lisesi
Staj Yaptığı Yerler	: -
İş Tecrübesi	: -
Başarılar	: -
Üyelikler	: IEEE

Ad Soyad	: Hüseyin Can Ercan
Doğum Tarih	: 10/02/1994
Doğum Yeri	: Çanakkale
Lise	: Çanakkale Fen Lisesi
Staj Yaptığı Yerler	: -
İş Tecrübesi	: -
Başarılar	: -
Üyelikler	: -