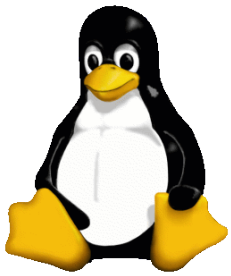
Gömülü Linux Sistemleri
Lab. Çalışması

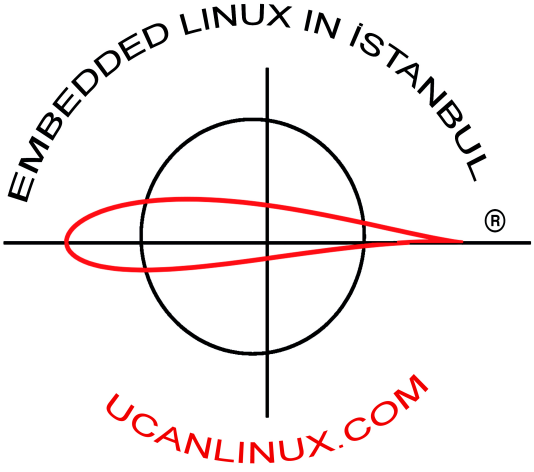
Nazım KOÇ

http://UCanLinux.com

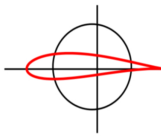
nazim@ucanlinux.com
nazim.koc@gmail.com

0.543.275 2907

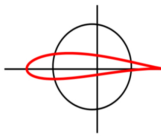




file: /nk/workspace/projects/linux.egitimi/bbb4/lab_calismasi/



Telif Hakları

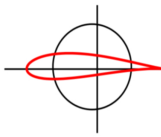


Amaçlar

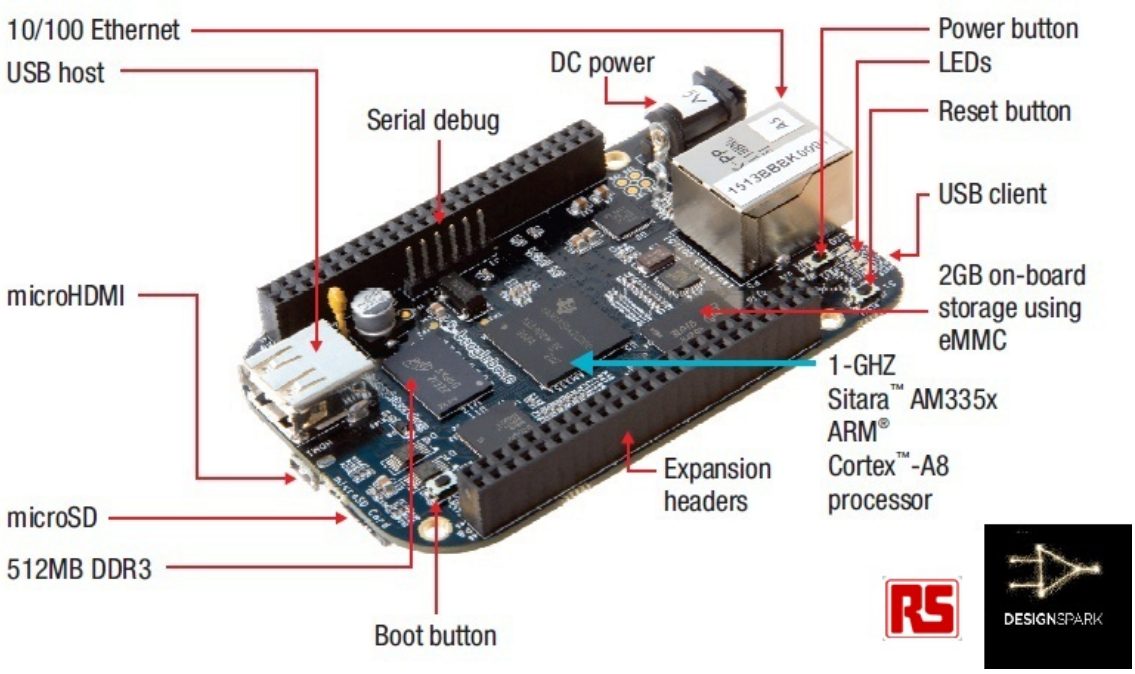
PowerOn ile Login arasındaki bütün adımları FARKINDA OLARAK yapmak.

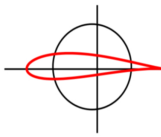
Projeye uygun gömülü sistemin seçilmesini sağlamak.

Sistematik olarak bir gömülü sistemin kurulmasını sağlamak.



Test Cihazı, BeagleBoneBlack, BBB





Test Cihazı

BBB sadece test amacı ile kullanılacaktır.

Eğitimin BBB ile doğrudan ilgisi yoktur. Anlatılacak konular BBB'den bağımsızdır.

Donanım ile ilgili konulardan bahsedilmeyecektir. Eğitim tamamen yazılım tabanlıdır.

Bahsedilecek bütün konular qemu emülatörü ile de test edilebilir.

Temeller

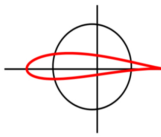
Linux sistemleri 4 temel üzerine kurulur.

Boot Loader:
   u-boot

Kernel:
   Linux

Root File System:
   Busybox, buildroot

Boot Scripts:
   handmade, buildroot



Yardımcılar

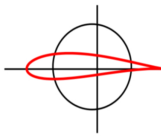
Hiç bir yardımcı betik, program veya tool kullanılmayacak,

Bütün kuruluşlar standard Linux komutları ile yapılacak,

Hiç bir Linux dağıtımına bağlı kalınmayacaktır.

Herhangi bir 64 Bit Ubuntu LTS sürümü tavsiye edilir.

Emülatör ortamları çoook sorunludur!

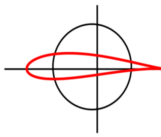


Gömülü Sistemin Karakteristiği

Çekirdeğin ve Kök Dosya Sisteminin oturduğu yer sistemin karakteristiğini verir.

Çekirdek:
- Network (uzakta)
- SD/MMC (BBB üzerinde, SD kart)
- eMMC (BBB üzerinde, cihaza gömülü)

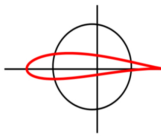
RootFS:
- RAM (çekirdeğe gömülü)
- SD/MMC
- eMMC
- NFS



Ön Hazırlık

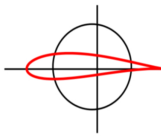
Terminalden girilen komutlar ve ekran görüntüleri otomatik olarak aşağıdaki gibi saklanabilir.

```
$ script ilk_gun

# Gün sonuna kadar çalışma yapılır.
# Sonra exit komutu ile script kapatılır.

$ exit

# Bütün ekran görüntüleri artık ilk_gun dosyası içindedir.
# Bu dosya metin tabanlıdır, cat/more/vi/less gibi komutlarla incelenebilir.
# Sıkıştırılıp saklanabilir.

$ more ilk_gun

# Ubuntu'da automounter mutlaka aşağıdaki gibi kapatılmalıdır.

$ dconf-editor
  Find ile automount özelliğini bul ve seçili kutuları kapat.
```



Dizin Yapısının Kurulması

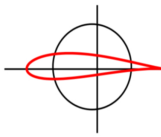
```
# usb tak, terminalden tar dosyasını aç.
#
# Altı çizili olanlar kullanıcıya veya dosya ismine bağımlıdır.


$ df
/dev/sdb1        7860164    239436   7620728   4% /media/nazim/EE6E-65E4

$ cd /opt

$ sudo tar xvf /media/nazim/EE6E-65E4/şirketAdı.tar

$ id

$ sudo chown -R can:can gomsis

$ ls -l gomsis
```



Dizin Yapısı

```
/opt/gomsis/
├── bin
│   └── mkimage
├── distro
│   ├── 1 -> tftp.kernel
│   ├── 2 -> tftp.tftp_cpio
│   ├── 3 -> tftp.nfs
│   ├── 4 -> mmc.mmc
│   ├── 5 -> mmc.mmc_cpio
│   ├── 6 -> emmc.emmc
│   ├── 7 -> tftp.nfs_br
│   ├── config
│   ├── emmc.emmc
│   ├── mmc.mmc
│   ├── mmc.mmc_cpio
│   ├── tftp.kernel
│   ├── tftp.nfs
│   ├── tftp.nfs.32
│   ├── tftp.nfs_br
│   └── tftp.tftp_cpio
```

```
/opt/gomsis/
├── ftp
│   ├── gcc-linaro-4.8-2015.06-x86_64_arm-linux-gnueabihf.tar.xz
│   ├── gcc-linaro-arm-linux-gnueabihf-4.8-2014.04_linux.tar.bz2
│   ├── linaro-prebuilt-sysroot-2013.07-2.tar.bz2
│   └── strace-4.10.tar.gz
├── out
│   ├── br
│   ├── busybox
│   ├── linux
│   └── u-boot
├── pdf
│   ├── bbb_kapak_ucanlinux.pdf
│   └── bbb_ucanlinux.pdf
├── src
│   ├── br
│   ├── busybox
│   ├── linux
│   ├── oku
│   └── u-boot
├── test
│   ├── hello.c
│   └── zebra.c
└── toolchain
    ├── arm -> gcc-linaro-arm-linux-gnueabihf-4.8-2014.04_linux
    ├── gcc-linaro-4.8-2015.06-x86_64_arm-linux-gnueabihf
    └── gcc-linaro-arm-linux-gnueabihf-4.8-2014.04_linux
```

Kaynak Kodları

U-Boot

```
$ git clone git://git.denx.de/u-boot.git
```

Kernel

```
$ git clone git://github.com/beagleboard/linux.git
$ git checkout 4.4
```

Busybox

```
$ git clone git://busybox.net/busybox.git
$ git checkout remotes/origin/1_NN_stable
```

BuildRoot

```
$ git clone git://git.buildroot.net/buildroot br
```

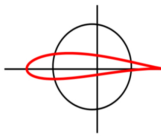
32Bit Toolchain (çok eski kaldı)

```
$ wget http://releases.linaro.org/14.04/components/toolchain/binaries/ \
            gcc-linaro-arm-linux-gnueabihf-4.8-2014.04_linux.tar.bz2
```

64-bit Toolchain

```
$ wget https://snapshots.linaro.org/openembedded/sources/ \
            gcc-linaro-4.8-2015.06-x86_64_arm-linux-gnueabihf.tar.xz
```

Bakınız: /opt/gomsis/src/oku



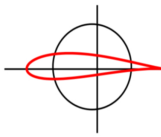
Günlük Çalışma Planı

1. gün:
   İlk sistem üzerinde çalışılacak.
   Sistematik kuruluş yapılacak.

2. gün:
   İlk sistem bitirilecek ve test edilecek.

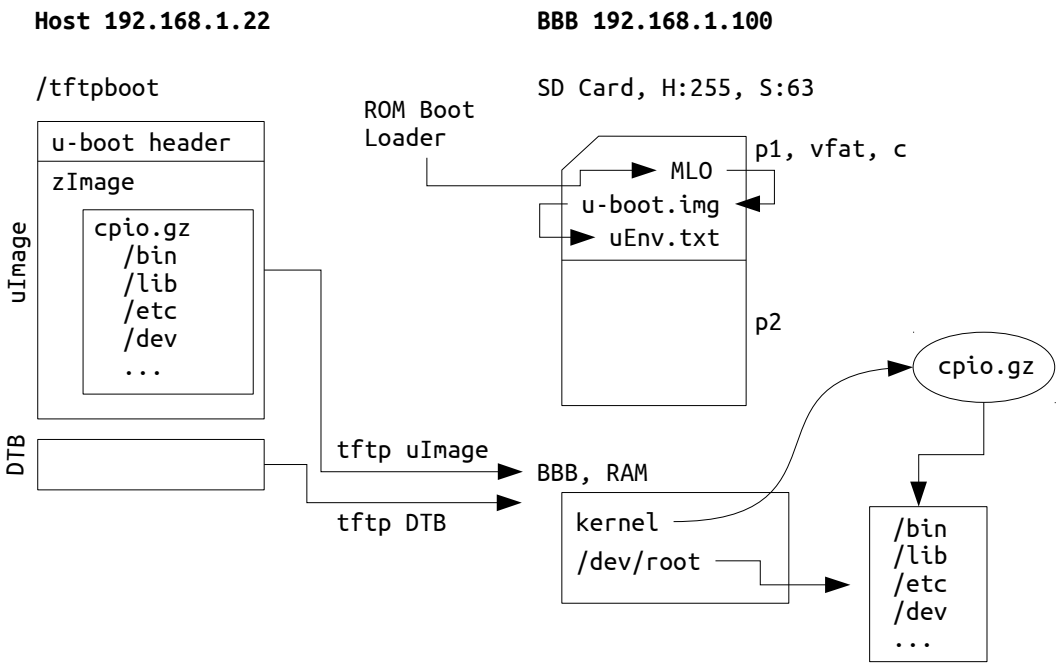
3. gün:
   2, 3, 4, 5 ve 6. sistemler kurulacak ve
   test edilecek. Birisi ödev olacak.

4. RootFS, buildroot ile kurulacak ve NFS
   yardımı ile test edilecek.

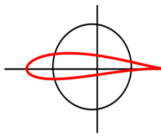


1. Sistem, initramfs destekli açılış

Kernel: uzakta, RootFS: kernel içinde



Bakınız: host:/opt/gomsis/distro/1 ve Belge:Bölüm 2

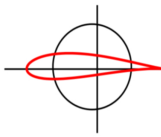


1.1 Çalışma ortamını kur

Tek bir dizin altında bütün projeyi kapsayan bir yapı kurulmalıdır.

```
$ tree -L 1 gomsis

gomsis
├── bin
├── distro
├── ftp
├── out
├── pdf
├── src
├── test
└── toolchain
```

Bakınız: host:/opt/gomsis/ ve Belge:Bölüm 2



1.2 Toolchain

/opt/gomsis/toolchain altında toolchain paketini aç.

```
# 64 bit host için gcc kurluşu.
# xz için J, bz2 için j kullan.

$ cd /opt/gomsis/toolchain

$ tar Jxvf /opt/gomsis/ftp/gcc-linaro-4.8-2015.06-x86_64_arm-linux-gnueabihf.tar.xz

$ ln -s gcc-linaro-4.8-2015.06-x86_64_arm-linux-gnueabihf arm
```

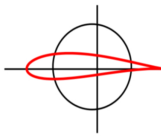
PATH'e bin/ ekle.

```
# Ön tarafa ekleme yapılmalıdır.

$ export PATH=/opt/gomsis/toolchain/arm/bin/:$PATH


# Kalıcı olması için ~/.bashrc'ye ekleme yapılabilir.
```

1.2 Toolchain

Test çalışması

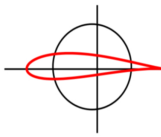
- ✔ /opt/gomsis/tets/dene.c programı üzerinde çalış.
- ✔ native derle.
- ✔ boy ve lib bağımlılıklarını incele.
- ✔ ufalt.
- ✔ statik derle.
- ✔ boy ve lib bağımlılıklarına bak.
- ✔ arm- <tab> <tab> ile kuruluşu incele.
- ✔ which ile tam yerini bul.
- ✔ yukarıdaki bütün adımları cross derleme ile yap.
- ✔ CROSS_COMPILE ile derle.



1.2 Toolchain

Çapraz derleme tekniği

```
$ make ARCH=arm
       CROSS_COMPILE=arm-linux-gnueabihf-
       -C soruce_dir
        O=output_dir
       -j cpu_number
       ...


# cpu adedi, /proc/cpuinfo'dan elde edilebilir.

# ARCH ve CROSS_COMPILE değişkenleri export tablosuna atılabilir.
# Bu durumda bu iki değişkeni yazmaya gerek kalmaz.

# O= ile çıkışlar kaynak dizin içinde oluşmaz.
# Fakat kernel derlemede sorunludur.

# -C  change directory
#  O  output
# -j  job number
```

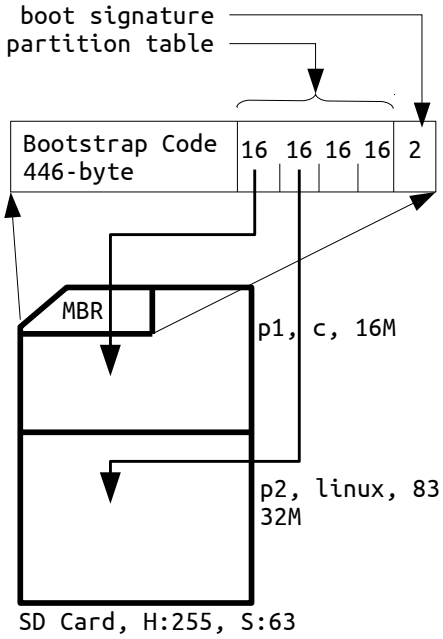


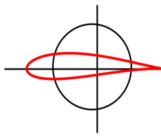
1.3 Das U-Boot, the Universal Boot Loader

```
$ cd /opt/gomsis/src/u-boot

$ make help

$ make menuconfig

$ ls configs

$ more configs/am335x_evm_defconfig


$ make distclean

$ make am335x_evm_deconfig

$ make menuconfig   # PROMPT değiş.
# vi ile de .config'deki prompt değiştirilebilir.

$ make CROSS_COMPILE=arm-linux-gnueabihf- -j2


$ sudo cp tools/mkimage /usr/local/bin

$ ls -l MLO
$ ls -l u-boot.img
$ ls -l u-boot.bin
$ file u-boot.img    # Bakınız ekler:u-boot programları
```

1.4 Disk Bölümlendirme, MS-DOS Partition

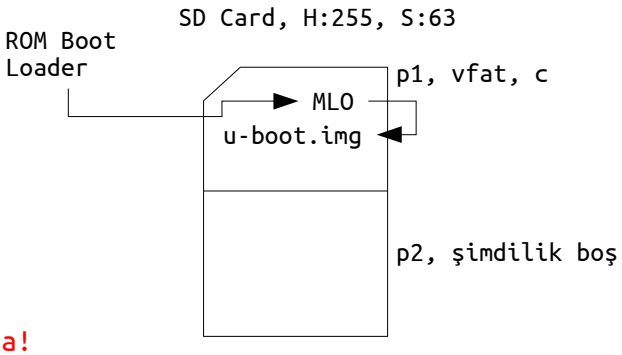
```
# SD kartı tak.

$ dmesg|tail

# Otomatik bağlanmış ise umount yap.
# ROM boot loader'lar disk geometrisine
# aşırı duyarlıdır.

$ fdisk -H 255 -S 63 /dev/mmcblk0   # veya
$ fdisk -H 255 -S 63 /dev/sdb

# 16MB vfat + 32MB Linux bölümü yap.

o

n p 1 +16M
a 1
t c

n p 2 +32M

p
w

$ sync   # veya
$ sudo sync

# sync demeden ASLA SD'yi çekme.
# Çıkar, tak. Böylece yeni bölüm tablosu kernel tarafında kullanılabilir.
```





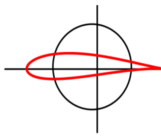
1.5 İlk bölümün kurulması

```
# SD kartı tak.
# mount edilmişse, unmount yap.

$ sudo -s

$ mkfs.vfat /dev/mmcblk0p1

$ mkdir /mnt/boot

$ mount /dev/mmcblk0p1 /mnt/boot

$ cd /opt/gomsis/src/u-boot/

$ cp MLO /mnt/boot   # en önce bunu kopyala!

$ cp u-boot.img /mnt/boot

$ sync

$ umount /mnt/boot
```



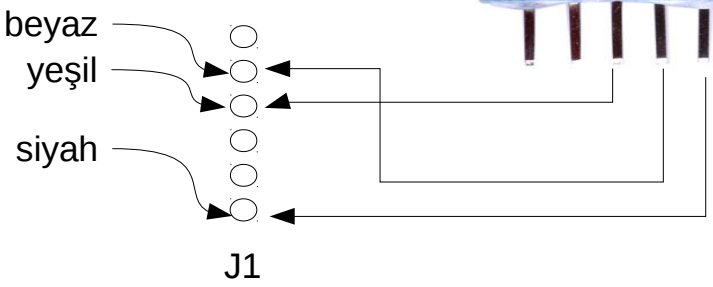
1.6 Açılış testi

```
# Aşağıdaki sırayı takip et, cihazları yakma!!!

# Seri kabloyu tak, ASLA seri power ucunu bağlama.

$ dmesg

$ minicom -c on
          -w
          -D /dev/ttyUSB0
          -b 115200

# SD'yi tak.
# BBB'yi USB3 ile besle, USB2 ASLA kullanma.
# USB3 yoksa, harici kaynak ile besle.

# S2 butonuna basılı iken power ver.
# prompt gelirse proje bitmiştir :)

=> help              # Bakınız ekler:u-boot örnekleri, u-boot programları
=> print ipaddr
=> mmc part
=> mmc info
=> mmc list
=> ls mmc 0
=> boot

# Tıkanır, kernel yok!
```

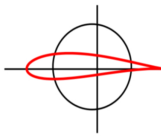


1.6 Açılış testi

Seri Bağlantı

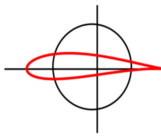


Uyarı Kırmızı ucu (power) asla bağlama, BBB'yi USB3 ile besle!!!



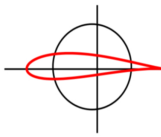
1.7 Çekirdek

```
# Ubuntu için aşağıdaki paketler yüklü olmalıdır.
# apt-get install libncurses5-dev lzop xinetd tftpd

$ cd /opt/gomsis/src/linux
$ make ARCH=arm help

$ ls -l arch/arm/configs
$ make ARCH=arm bb.org_defconfig


$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- menuconfig


# Örnek config dosyasını kopyala.
$ cp ../../distro/tftp.kernel/kernel.config  .config


$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- LOADADDR=0x80008000 -j2 uImage

$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- -j2 modules

$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- -j2 dtbs
```



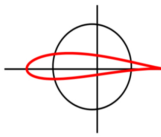
1.7 Çekirdek

```
$ modinfo drivers/i2c/busses/i2c-simtec.ko

$ cd arch/arm/boot

$ file uImage

$ ls -l uImage

$ mkimage -l uImage


# Üretilen Dosyalar

uImage   u-boot header + zImage
dtb      device tree blob, hardware description file
*.ko     kernel objects, modules
*.fw     firmware codes


Altın Kurallar:

uImage ve dtb dosyaları beraber bulunurlar ve u-boot tarafından yüklenirler.

*.ko dosyaları kök içinde ve /lib/modules/`uname -r`/ içinde otururlar.

*.fw dosyaları kök içinde ve /lib/firwmare/ içinde otururlar.
```



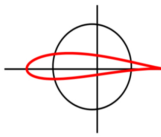
1.7 Çekirdek, test

```
$ cd /opt/gomsis/src/linux
$ cp arch/arm/boot/uImage   /tftpboot
$ cp arch/arm/boot/dts/am335x-boneblack.dtb   /tftpboot

# Açılışta ENTER'a bas u-boot promptuna gel.
=> setenv ipaddr 192.168.1.100
=> setenv serverip 192.168.1.22

=> tftp 80007FC0 uImage
=> tftp 80F80000 am335x-boneblack.dtb

=> setenv bootargs console=ttyS0,115200n8
=> bootm 80007FC0  -  80F80000

# RootFS yok, sistem panikler
```

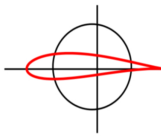
1.7 Çekirdek, test

```
# autoboot
# p1:/uEnv.txt

ipaddr=192.168.1.100

serverip=192.168.1.22

bootargs=console=ttyS0,115200n8

tftp_test=tftpboot 0x80F80000 am335x-boneblack.dtb ; \
          tftpboot 0x80007FC0 uImage ; \
          bootm 0x80007FC0 - 0x80F80000

uenvcmd=run tftp_test



# Not: LiveCD kullananlar için:
# squashfs için NFS desteği yoktur.
# sda'da dizin yarat ve sembolik link ile sorunu çöz.
```

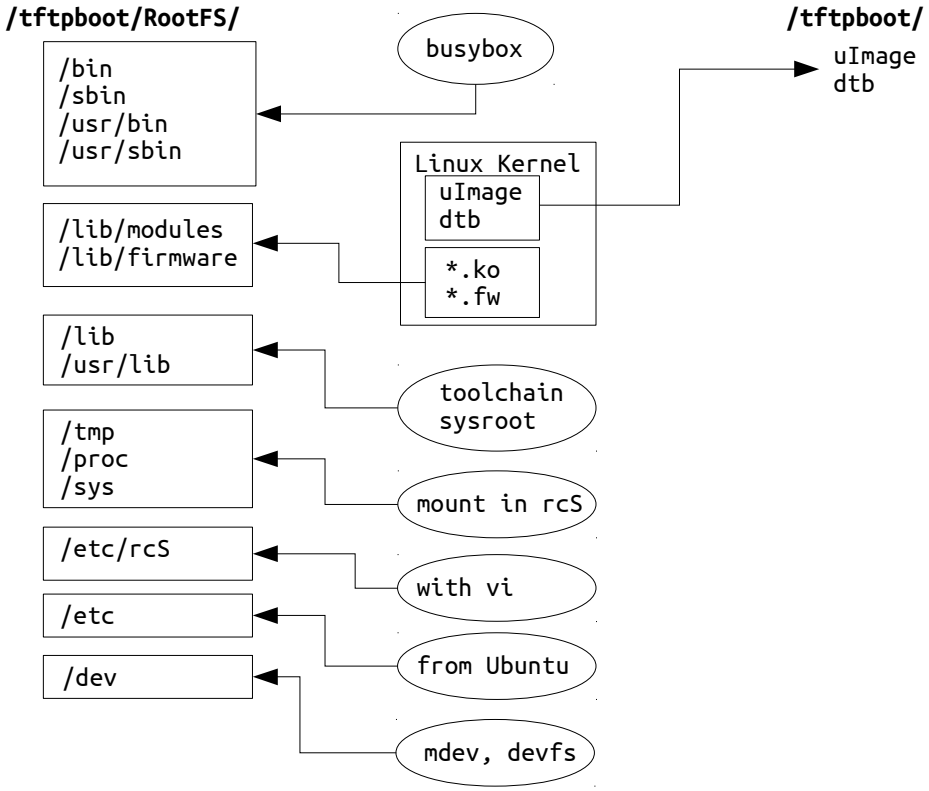


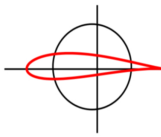
1.8 Busybox

```
$ cd /opt/gomsis/src/busybox
$ make CROSS_COMPILE=arm-linux-gnueabihf- clean
$ make CROSS_COMPILE=arm-linux-gnueabihf- menuconfig
$ make CROSS_COMPILE=arm-linux-gnueabihf- -j2

$ ls -l busybox
$ file busybox
```



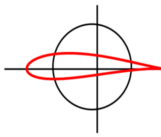
1.9 RootFS'in İnşası





1.9 RootFS'in İnşası

```
$ cd /tftpboot
$ mkdir RootFS
$ cd RootFS


# skeleton

$ cp -a /opt/gomsis/RootFS.skel/* .        # veya

$ cp -a /opt/gomsis/src/br/system/skeleton/* .


# kernel

$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf-  \
       INSTALL_MOD_PATH=/tftpboot/RootFS            \
       modules_install

$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf-  \
       INSTALL_MOD_PATH=/tftpboot/RootFS            \
       firmware_install



# firmware_install'da INSTALL_MOD_PATH vermilmezse, modules_install'da
# verilen değer kabul edilir.
```

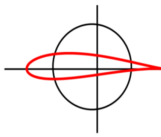



1.9 RootFS'in İnşası

```
# busybox

$ cd /opt/gomsis/src/busybox

# 1. yöntem
$ rm -fr _install
$ make CROSS_COMPILE=arm-linux-gnueabihf- install
$ cp -a  _install/*   /tftpboot/RootFS/

# 2. yöntem
# Busybox Settings --->
    Installation Options -->
        (/tftpboot/RootFS/) BusyBox installation prefix
$ make CROSS_COMPILE=arm-linux-gnueabihf- install

# minimum libs= { interpreter, libc, libm }

$ SYSROOT=`arm-linux-gnueabihf-gcc -print-sysroot`

$ echo $SYSROOT

$ ls -l $SYSROOT

# 64 bit
$ cp -a $SYSROOT/lib/ld-linux-armhf.so.3  /tftpboot/RootFS/lib/
$ cp -a $SYSROOT/usr/lib/libc.so.6        /tftpboot/RootFS/usr/lib/
$ cp -a $SYSROOT/usr/lib/libm.so.6        /tftpboot/RootFS/usr/lib/
```

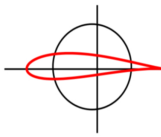
1.10 RootFS'in Çekirdeğe Gömülmesi

```
# Kozmetik çalışması.

$ figlet Foo > /tftpboot/RootFS/etc/issue
$ vi /tftpboot/RootFS/etc/issue
  Welcome to YourName


$ cd /opt/gomsis/src/linux

$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- menuconfig

# /tftpboot/RootFS'i çekirdeğe ekle.
# UID/GID girişini unutma.

$ rm usr/initramfs_data.cpio.gz

$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- LOADADDR=0x80008000 -j2 uImage


$ cd arch/arm/boot

$ ls -l uImage   # boy büyür.
```



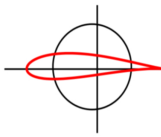
1.11 /tftpboot'a Kernel/DTB'nin atılması

```
$ cd /opt/gomsis/src/linux
$ cp arch/arm/boot/uImage /tftpboot
$ cp arch/arm/boot/dts/am335x-boneblack.dtb /tftpboot
$ ls -l /tftpboot
```



1.12 /etc/rcS

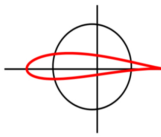
```
#!/bin/sh +x
#
export PATH=/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/bin

mount -t proc  proc  /proc
mount -t sysfs sysfs /sys

# mount ile de yapılabilir!
echo /sbin/mdev >/proc/sys/kernel/hotplug
mdev -s

mkdir /dev/pts
mount -t devpts devpts /dev/pts

mkdir /dev/shm
mount -t tmpfs  tmpfs  /dev/shm

hostname UCanLinux

syslogd
klogd

ifconfig lo 127.0.0.1 up
route add -net 127.0.0.0 netmask 255.0.0.0 gw 127.0.0.1 lo

ifconfig eth0 192.168.1.100 up
route add default gw 192.168.1.1

telnetd
```

1.13 Test 1, Hazırlık

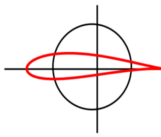
```
# /etc/xinetd.d/tftp

service tftp
{
     protocol        = udp
     port            = 69
     socket_type     = dgram
     wait            = yes
     user            = nobody
     server          = /usr/sbin/in.tftpd
     server_args     = /tftpboot
     disable         = no
}


$ service xinetd restart
```

Bak:/opt/gomsis/1/tftp.config



1.13 Test 1, Açılış

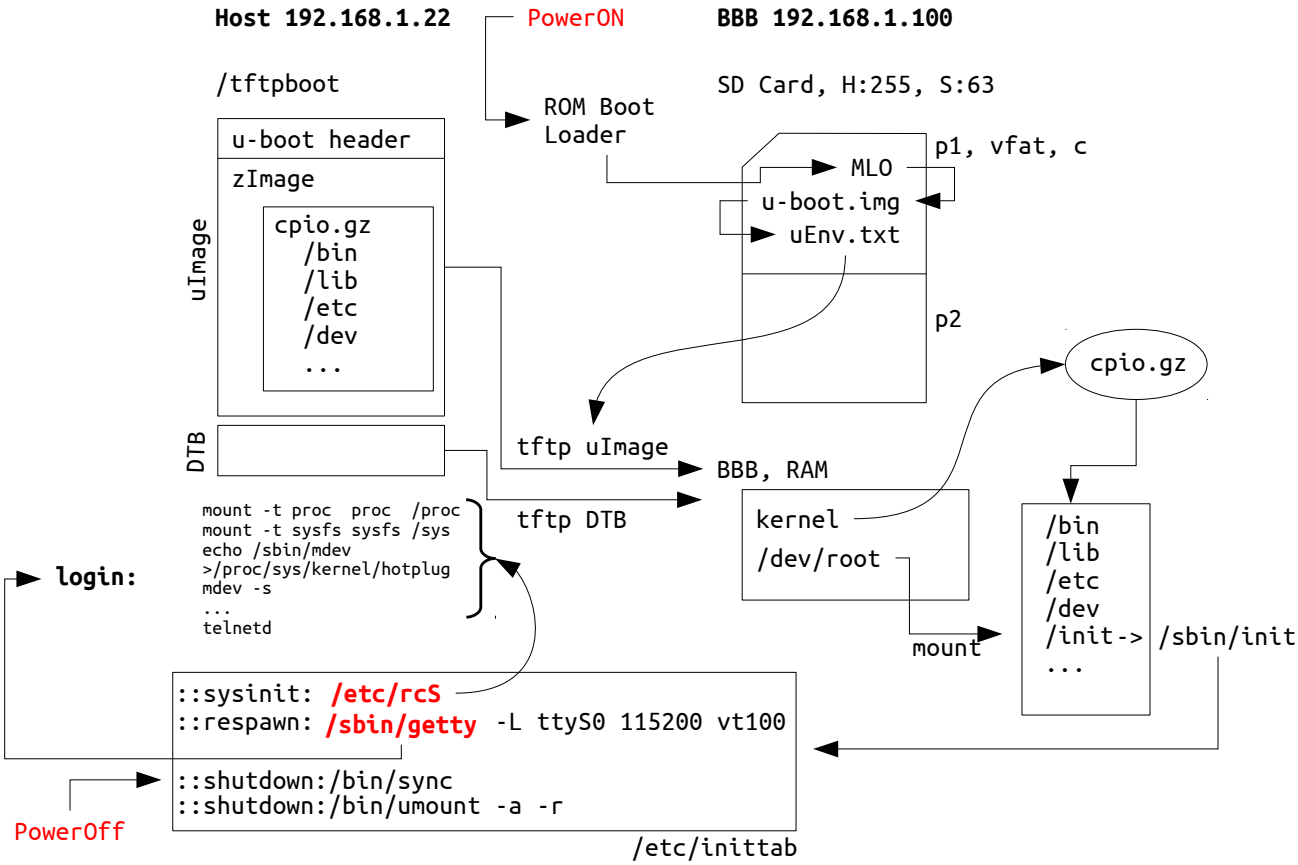
```
# S2'ye basarak SD'den açılışa zorla.
# Açılışta ENTER'a bas, u-boot promptuna gel.

=> setenv ipaddr 192.168.1.100

=> setenv serverip 192.168.1.22

=> tftp 80007FC0 uImage

=> tftp 80F80000 am335x-boneblack.dtb

=> setenv bootargs console=ttyS0,115200n8

=> bootm 80007FC0  -  80F80000


# login gelir.
# telnet 192.168.1.100 root/root ile host'tan giriş yap.

$ pstree -p
$ busybox


Altın Kural: initramfs destekli sistemlerde çekirdek /init'i çalıştırır.

Niye 80007FC0? Bak s33, s200, s202
```



1.13 Test 2, vfat'in İncelenmesi

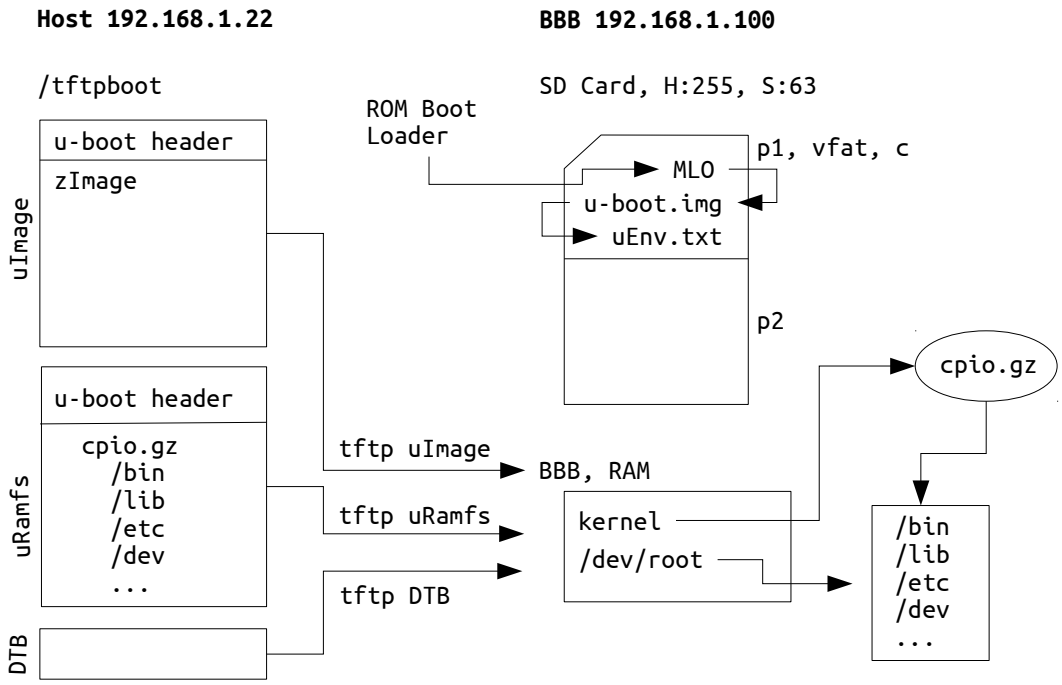
```
# p1'i bağla ve içini gör.


$ mount /dev/mmcblk0p1 /mnt/boot

$ ls -l


# Altın Kural:
#   MLO hariç bütün dosyalar mount edildikten sonra güncellenebilir.
#   MLO'ya dokunma, yerini değiştirme!

# Autoboot için,

$ cat /mnt/boot/uEnv.txt


ipaddr=192.168.1.100
serverip=192.168.1.22

bootargs=console=ttyS0,115200n8

uenvcmd=tftpboot 0x80F80000 am335x-boneblack.dtb ; \
        tftpboot 0x80007FC0 uImage ;               \
        bootm 0x80007FC0 - 0x80F80000
```

1.13 Test 3, Initramfs Açılış Sırası



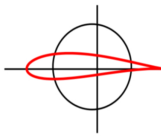
2. Sistem, ayrık initramfs destekli açılış

Kernel: uzakta, RootFS: uzakta



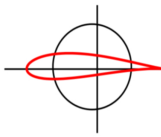
DTS:device tree source, DTB:device tree blob(kernel readable), dtc(DTS)-> DTB

Bakınız: host:/opt/gomsis/distro/2 ve Belge:Bölüm 3



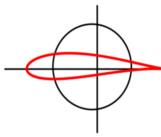
2. Sistem, ayrık initramfs destekli açılış

```
# Çekirdeği tekrar derle, içindeki RootFS'i iptal et.
$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- menuconfig
$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- -j2 LOADADDR=0x80008000 \
      uImage
$ cp uImage /tftpboot

# Dışarıda bir yere RootFS'in yedeğini al.
$ cd
$ cp -a /tftpboot/RootFS .

# u-boot ve çekirdeğin anlayacağı bir imaj haline getir.
$ sudo -s
$ cd /tftpboot
$ chown -R root:root RootFS
$ cd RootFS
$ find . | cpio -o -H newc | gzip > ../rootfs.cpio.gz
$ cd ..
# Aşağıdaki komut tek satırda yazılmalıdır.
$ mkimage -A arm
          -T ramdisk
          -C none
          -n ”test2”
          -d rootfs.cpio.gz
             /tftpboot/uramfs.img

$ ls -l      rootfs.cpio.gz
$ ls -l      uramfs.img  # cpio.gz ile olan boy farkına bak. 64 bayt’ı gör.
$ file       uramfs.img
$ mkimage -l uramfs.img
```



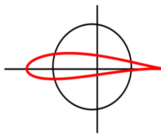
2. Sistem, /etc/rcS

```
#!/bin/sh +x
#
export PATH=/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/bin

mount -t proc  proc  /proc
mount -t sysfs sysfs /sys

# mount ile de yapılabilir!
echo /sbin/mdev >/proc/sys/kernel/hotplug
mdev -s

mkdir /dev/pts
mount -t devpts devpts /dev/pts

mkdir /dev/shm
mount -t tmpfs  tmpfs  /dev/shm

hostname UcanLinux

syslogd
klogd

ifconfig lo 127.0.0.1 up
route add -net 127.0.0.0 netmask 255.0.0.0 gw 127.0.0.1 lo

ifconfig eth0 192.168.1.100 up
route add default gw 192.168.1.1

telnetd
```



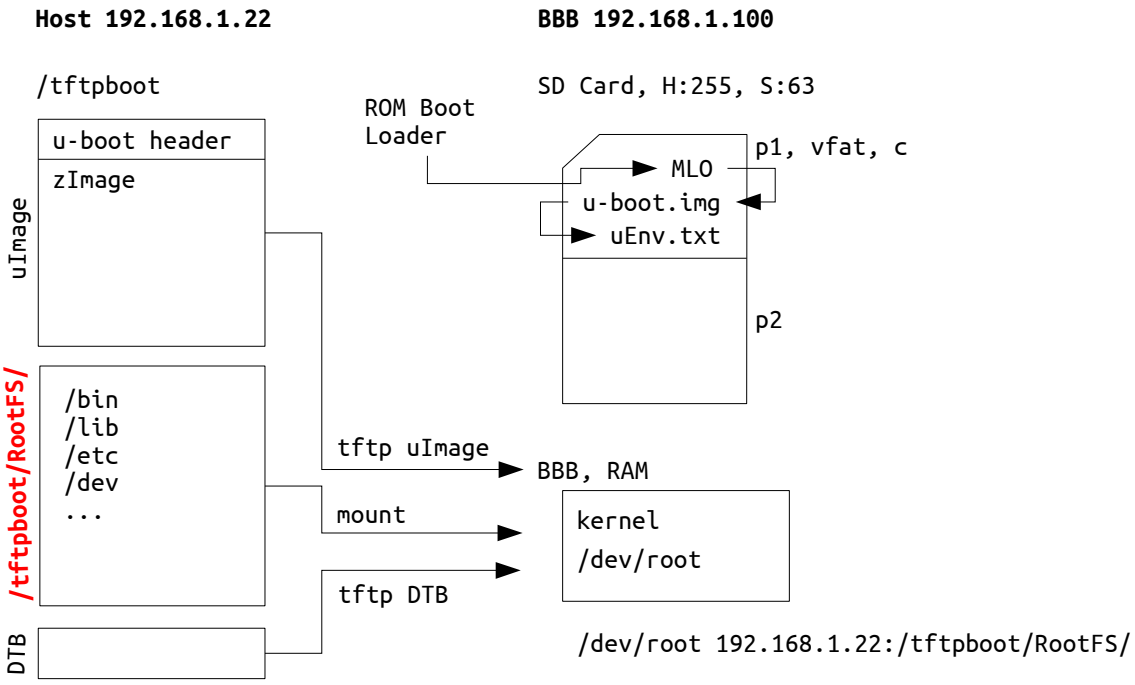
2. Sistem, uEnv.txt

```
# p1:/uEnv.txt

ipaddr=192.168.1.100
serverip=192.168.1.21

bootargs=console=ttyS0,115200n8

uenvcmd=tftpboot 0x80007FC0 uImage; \
        tftpboot 0x88080000 uramfs.img; \
        tftpboot 0x80F80000 am335x-boneblack.dtb; \
        bootm 0x80007FC0 88080000 0x80F80000



# Bu sistem açıldıktan sonra SD/MMC kart yerinden çıkarılabilir.
# Çünkü RootFS tamamen RAM içindedir.
# Artık SD/MMC'deki bilgilere gerek duyulmaz.
```

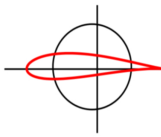



3. Sistem, ideal test sistemi

Kernel: uzakta, RootFS: uzaktan mount edilir, yüklenmez.

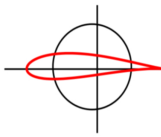


Bakınız: host:/opt/gomsis/distro/3 ve Belge:Bölüm 4



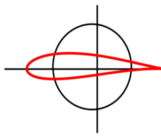
3. Sistem, ideal test sistemi

```
# Çekirdeğe RootFS'i uzaktan bağlayabilme özelliğini ekle.

File Systems -->
   Network File Systems
   Root File System on NFS


# /tftpboot/RootFS'i olduğu gibi kullan.
# Sadece /etc/rcS değiş.

# Ayrıca uEnv.txt'yi yeniden yaz.


# Host tarafında NFS server için hazırlık.
# Sadece Ubuntu içindir.

$ apt-get install nfs-kernel-server
$ service nfs-kernel-server restart

$ vi /etc/exports
/tftpboot  192.168.1.100(rw,insecure,no_subtree_check,no_root_squash)

$ exportfs -avr
```



3. Sistem, /etc/rcS

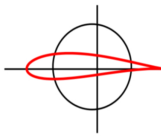
```
#!/bin/sh +x
#
export PATH=/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/bin

mount -t tmpfs  tmpfs  /tmp
mount -t sysfs  sysfs  /sys
mount -t proc   proc   /proc

mkdir /dev/pts
mount -t devpts devpts /dev/pts

mkdir /dev/shm
mount -t tmpfs  tmpfs  /dev/shm

mount -t tmpfs tmpfs /var
mkdir /var/log
mkdir /var/run
mkdir /var/tmp
mkdir /var/cache

hostname UcanLinux

syslogd
klogd

ifconfig lo 127.0.0.1 up
route add -net 127.0.0.0 netmask 255.0.0.0 gw 127.0.0.1 lo

telnetd
```



3. Sistem, p1:/uEnv.txt

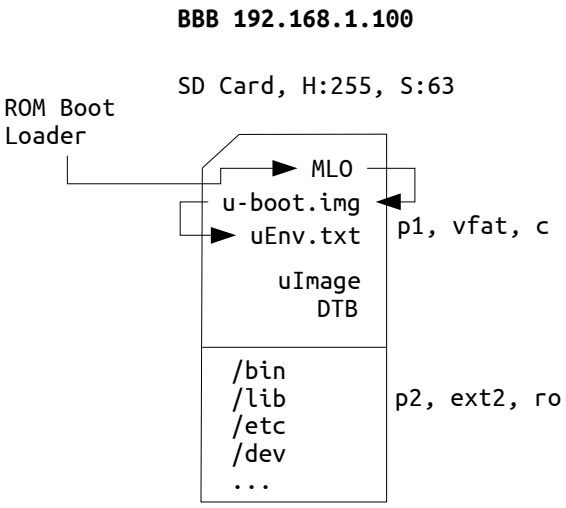
```
# p1:/uEnv.txt

ipaddr=192.168.1.100
serverip=192.168.1.22

bootargs=console=ttyS0,115200n8 \
ip=192.168.1.100:192.168.1.22:192.168.1.22:255.255.255.0:ideal_test:eth0:off \
         root=/dev/nfs \
         rw \
         nfsroot=192.168.1.22:/tftpboot/RootFS

tftp_test=tftpboot 0x80F80000 am335x-boneblack.dtb ; \
          tftpboot 0x80007FC0 uImage ; \
          bootm    0x80007FC0 - 0x80F80000

uenvcmd=run tftp_test

# ip=client_ip:server_ip:gw:mask:hostname:device:autoconf:dns1:dns2


# “merhaba dünya” kodunu cross derle ve NFS üzerinden çalıştır.
# NFS’in güzelliğini gör.
# Bir önceki sistemdeki rcS ile olan farkını analiz et.
```

Bakınız Belge: Bölüm 4, İdeal Test Sistemi

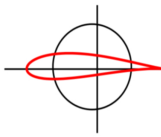


4. Sistem, tamamı SD kartta olan sistem

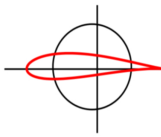
Kernel: SD/vfat, RootFS: SD/ext2



Bakınız: host:/opt/gomsis/distro/4 ve Belge:Bölüm 5

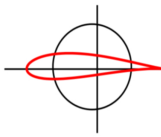


```
# Önce SD kartı bölümlendir.
# dmesg | tail ile SD kartın cihaz ismini tespit et.
# Bölümler bağlı ise, unmount etmeyi unutuma!!!

$ fdisk -H255 -S63 /dev/mmcblk0  veya
$ fdisk -H255 -S63 /dev/sdb

o n
p 1 ENTER +16M
a 1
t c
n p 2 ENTER +16M
p
w

$ sync

# SD kartı çıkart ve tekrar tak ki kernel yeni bölümlendirme
# tablosundan haberdar olsun.


# Dosya sistemi kur.

$ mkfs.vfat -c -n BOOT4 /dev/mmcblk0p1
$ mkfs.ext2 -c -L ROOT4 /dev/mmcblk0p2
```



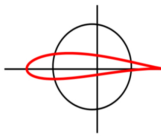
4. Sistem, Vfat ve Ext2'nin İnşası

```
$ mount /dev/mmcblk0p1 /mnt/boot

$ cp /tftpboot/MLO                      /mnt/boot   # İlk kopyalanır!!!
$ cp /tftpboot/u-boot.img               /mnt/boot

$ cp /tftpboot/uImage                   /mnt/boot
$ cp /tftpboot/am335x-boneblack.dtb     /mnt/boot

$ cp /opt/gomsis/distro/4/p1/uEnv.txt   /mnt/boot

$ ls -l

total 5196
-rw-r--r-- 1 nazim nazim   60627 Nis 23 08:56 bbb.dtb
-rw-r--r-- 1 nazim nazim   66268 Nis 23 08:55 MLO
-rw-r--r-- 1 nazim nazim  318824 Nis 23 08:55 u-boot.img
-rw-r--r-- 1 nazim nazim     212 Nis 23 08:55 uEnv.txt
-rw-r--r-- 1 nazim nazim 4863480 Nis 23 08:56 uImage

$ sync
$ umount /mnt/boot

$ mount /dev/mmcblk0p2     /mnt/root
$ cp -a /tftpboot/RootFS/* /mnt/root
$ chown -R root:root       /mnt/root
$ sync
$ unmount /mnt/root
```



4. Sistem, /etc/rcS

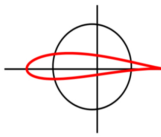
```
#!/bin/sh +x
export PATH=/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/bin

mount -t tmpfs  tmpfs  /tmp
mount -t sysfs  sysfs  /sys
mount -t proc   proc   /proc

mkdir /dev/pts
mount -t devpts devpts /dev/pts

mkdir /dev/shm
mount -t tmpfs  tmpfs  /dev/shm

mount -t tmpfs tmpfs /var
mkdir /var/log  /var/run  /var/tmp  /var/cache

hostname UcanLinux
syslogd
klogd

ifconfig lo 127.0.0.1 up
route add -net 127.0.0.0 netmask 255.0.0.0 gw 127.0.0.1 lo

ifconfig eth0 192.168.1.100 up
route add default gw 192.168.1.1

telnetd

ifconfig eth0
route -n
df -a
```

4. Sistem, p1:/uEnv.txt

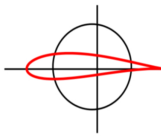
```
bootargs=console=ttyS0,115200n8 \
         root=/dev/mmcblk0p2    \
         ro                     \
         rootfstype=ext2        \
         rootwait               \
         fixrtc

mmc_test=load mmc 0:1 0x80007FC0 uImage  ; \
         load mmc 0:1 0x80F80000 bbb.dtb ; \
         bootm 0x80007FC0 - 0x80F80000

uenvcmd=run mmc_test
```

```
# Altın Kural: Kök dosya sistemi her zaman Read/Only bağlanmalıdır.
# SD/MMC çıkartılırsa sistem çöker.
# Çünkü kök dosya sistemi tamamen SD/MMC üzerinde oturmaktadır.
# Kök dosya sistemi, güncellemeler için aşağıdaki gibi rw bağlanabilir.

$ mount -o remount -o rw /

# Sistemin SD/MMC'den açıldığı nasıl anlaşılır?

$ stat /    # b302h gör.
$ printf "%d\n", 0xb3
$ ls -l /dev/mmcblk0p2   # majör ve minör 0xb3 0x02'dir.
```



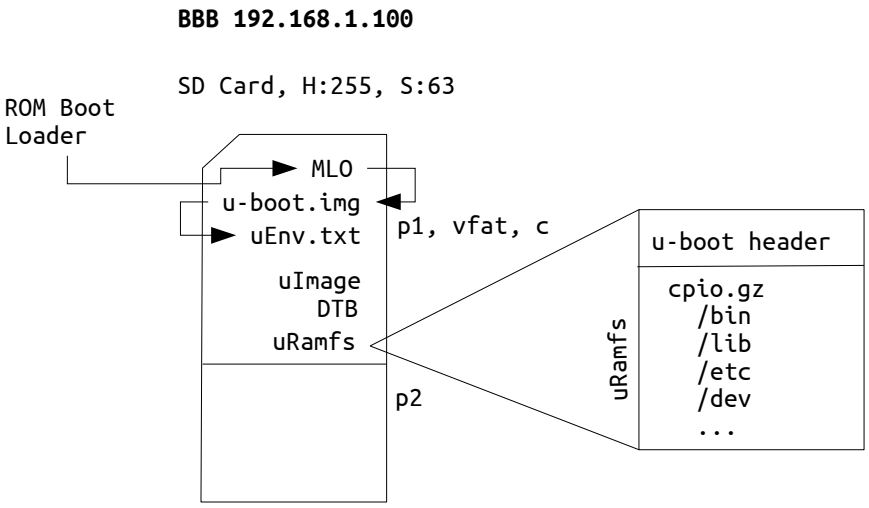
4. Sistem, ram disk kullanımı

```
# Kök dosya sistemi her zaman ro bağlıdır.
# rw ihtiyaçları için ram disk (ram file system) kurulabilir.

$ mkdir /tmp/disk
$ mount -t tmpfs none /tmp/disk
$ mount
$ cd /tmp/disk
$ touch foo
$ ls -l
$ df .
$ umount /tmp/disk


# Güç kesilince ram diskler yok olur!
```

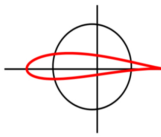


5. Sistem, SD'de cpio imajı

Kernel: SD/vfat, RootFS: SD/vfat/cpio

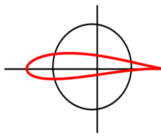


Bakınız: host:/opt/gomsis/distro/5 ve Belge:Bölüm 6



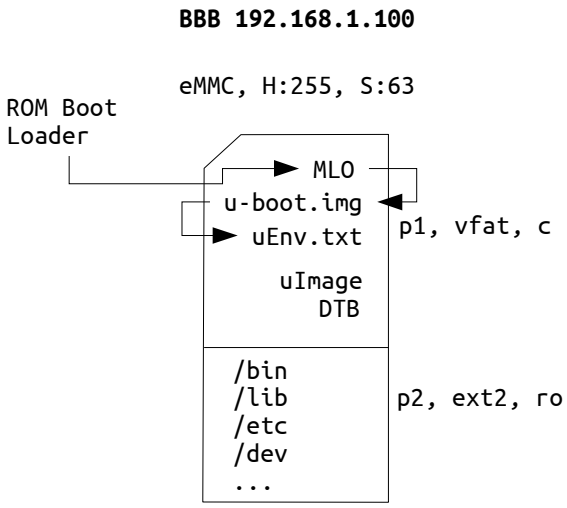
5. Sistem, p1:/uEnv.txt

```
bootargs=console=ttyS0,115200n8

uenvcmd=load mmc 0:1 0x80007fc0 uImage ;                \
        load mmc 0:1 0x80F80000 am335x-boneblack.dtb ; \
        load mmc 0:1 0x88080000 uramfs.img ;            \
        bootm 0x80007FC0 88080000 0x80F80000



# Bu sistem 2. sistem ile tamamen aynıdır.
# Sadece ramfs imajı farklı yerde oturur.


# Katılımcı için ödevdir.
```



6. Sistem, tamamı eMMC'de olan sistem

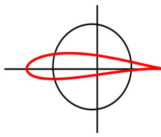
Kernel: eMMC/vfat, RootFS: eMMC/ext2



Tamamı SD'de olan 4. sistemle tamamen aynıdır.

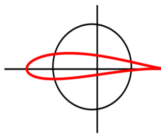
BBB, mevcut sistemlerin biriye SD'den açılır ve NFS yardımı ile p1 ve p2 bölümleri /tftpboot üzerinden kopyalanır.

Bakınız: host:/opt/gomsis/distro/6 ve Belge:Bölüm 7



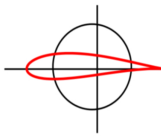
6. Sistem, Bölümlendirme

```
# Önceki sistemlerin biri ile BBB'yi aç.

$ cat /proc/partitions

$ fdisk -H255 -S63 /dev/mmcblk1

o n
p 1 ENTER +64M      # 16M yapma, açılmıyor!
a 1
t c
n p 2 ENTER +64M
p
# start LBA  End LBA  Sectors Type
#      63     144584   144522   c
#  144585     289169   144585  83
#
# 477 C, 255 H, 63 sector/track
# 1 C= 255*63*512= 8,225,280 bayt
w
$ sync

# Dosya sistemi kur.

$ mkfs.vfat -n BOOT6 /dev/mmcblk1p1
$ mkfs.ext2 -L ROOT6 /dev/mmcblk1p2
```

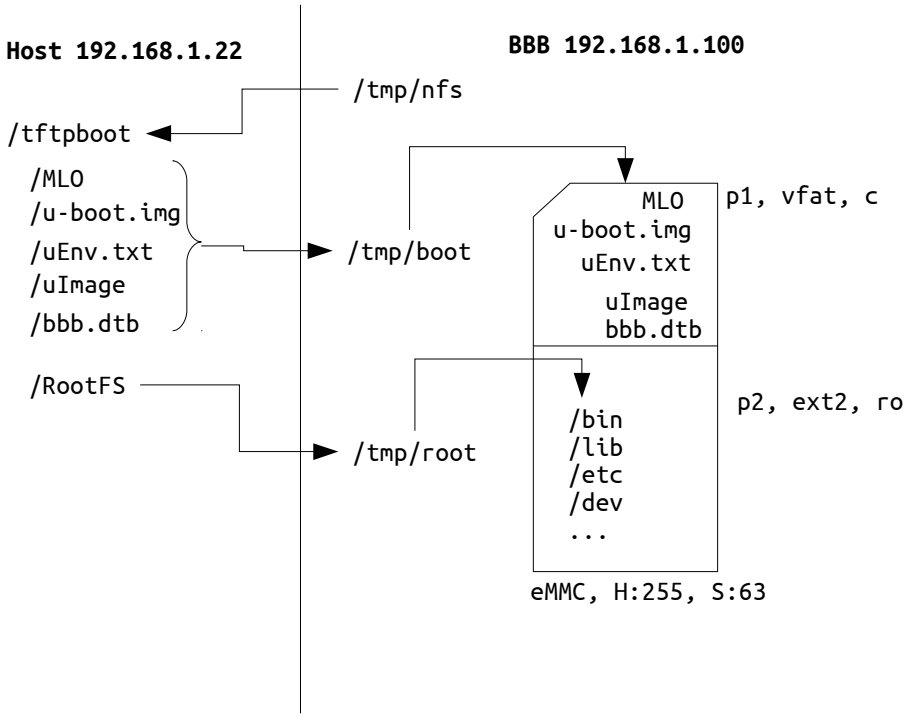


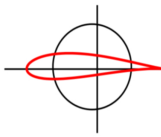
6. Sistem, NFS Yapılandırması

```
# Ubuntu için, nfs-kernel-server yüklü olmalıdır.

$ cat /etc/exports

/tftpboot  192.168.1.100(rw,insecure,no_subtree_check,no_root_squash,async)


$ service nfs-kernel-server restart

$ exportfs -avr

# BBB tarafında NFS ile yükleme.

$ mkdir /tmp/nfs
$ ifconfig eth0 192.168.1.100 up   # /etc/rcS'de yoksa.
$ mount -t nfs -o nolock 192.168.1.22:/tftpboot /tmp/nfs

$ mkdir /tmp/boot
$ mount /dev/mmcblk1p1 /tmp/boot

$ mkdir /tmp/root
$ mount -t ext2 /dev/mmcblk1p2 /tmp/root

# Host tarafında MLO, u-boot.img ve uEnv.txt'yi /tftpboot altına kopyala.
```



6. Sistem, Genel Görünüm



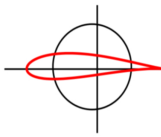


6. Sistem, Vfat ve Ext2'nin İnşası

```
$ df
$ cd /tmp/nfs

$ cp MLO                   /tmp/boot   # İlk kopyalanmalıdır!
$ cp u-boot.img            /tmp/boot
$ cp uImage                /tmp/boot
$ cp am335x-boneblack.dtb /tmp/boot/bbb.dtb
$ cp uEnv.txt              /tmp/boot


$ cp -a RootFS/* /tmp/root

$ sync

$ df

$ umount /tmp/nfs
$ umount /tmp/root
$ umount /tmp/boot

$ sync
$ reboot
```

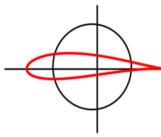
6. Sistem, /etc/rcS

```
#!/bin/sh +x
#
export PATH=/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/bin

mount -t tmpfs  tmpfs  /tmp
mount -t sysfs  sysfs  /sys
mount -t proc   proc   /proc

mkdir /dev/pts
mount -t devpts devpts /dev/pts

mkdir /dev/shm
mount -t tmpfs  tmpfs  /dev/shm

mount -t tmpfs tmpfs /var
mkdir /var/log /var/run /var/tmp /var/cache

hostname UcanLinux
syslogd
klogd

ifconfig lo 127.0.0.1 up
route add -net 127.0.0.0 netmask 255.0.0.0 gw 127.0.0.1 lo

ifconfig eth0 192.168.1.100 up
route add default gw 192.168.1.1

telnetd
```



6. Sistem, p1:/uEnv.txt

```
bootargs=console=ttyS0,115200n8 \
         root=/dev/mmcblk1p2    \
         ro                     \
         rootfstype=ext2        \
         rootwait               \
         fixrtc

mmc_test=load mmc ${mmcdev}:1 80007fc0 uImage   ; \
         load mmc ${mmcdev}:1 0x80F80000 bbb.dtb; \
         bootm 0x80007FC0 - 0x80F80000

uenvcmd=run mmc_test


# mmcdev bir u-boot çevre değişkenidir.
# Boot edilen cihazın numarasını saklar.
# 0 veya 1 olabilir.
```



7. Sistem, BR destekli RootFS

En genel sistemdir.

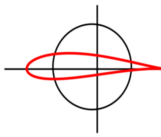
Kök dosya sistemi RootFS ile kurulur.
Açılış betikleri RootFS ile kurulur.

U-boot, kernel ve toolchain desteği olmasına rağmen, bu destekler çok da kullanışlı değildir.

Nihayetinde rootfs.tar elde edilir.

Geçmiş 6 sistemin herhangi birinde buradaki rootfs.tar kullanılır.

Bakınız: host:/opt/gomsis/distro/7 ve Belge:Bölüm 8



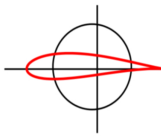
7. Sistem, BR destekli RootFS

Genel mantık silsilesi, örnek:

[*] php seçilirse...

git/wget/rsync/... ile paket indirilir.
dl/ altında depolanır.
configure ile Makefile kurulur.
make ile derleme yapılır.
make install ile iskelet'e kuruluş yapılır.
...
iskelet, rootfs.tar haline getirilir.
br/output/images/ altına atılır.

Bakınız: host:/opt/gomsis/distro/7 ve Belge:Bölüm 8

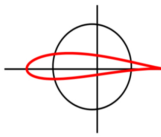


7. Sistem, BR destekli RootFS

rootfs.tar'ı kurmak için..

```
$ cd /opt/gomsis/src/br
$ make clean    # !!!
$ make list-defconfigs  # Bordun burada gözükmemesi hiç önemli değildir :)
$ make beaglebone_defconfig
$ make menuconfig
# toolchain'in external seçilmesi tavsiye edilir.
$ make -j2
# Nihayetinde rootfs.tar elde edilir.
```

Bakınız: host:/opt/gomsis/distro/7 ve Belge:Bölüm 8



7. Sistem, BR destekli RootFS

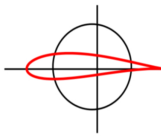
rootfs.tar arşivi, /tftpboot/RootFS/ altına açılır.

Kernel tarafında elde edilen firmware ve modules ayrıca install edilmelidir.

```
$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf-  \
       INSTALL_MOD_PATH=/tftpboot/RootFS            \
       modules_install

$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf-  \
       INSTALL_MOD_PATH=/tftpboot/RootFS            \
       firmware_install
```

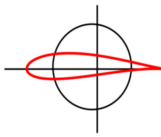
/tftpboot/RootFS artık bütün örnek sistemlerde test edilebilir.

Bakınız: host:/opt/gomsis/distro/7 ve Belge:Bölüm 8



7. BR'deki gcc'nin Kernel Sürümü

```
# toolchain'de libc/usr/linux/version.h içine bak.

$ cd /opt/gomsis/toolchain/arm/arm-linux-gnueabihf/libc/usr/include/linux

$ more version.h

#define LINUX_VERSION_CODE 196865
#define KERNEL_VERSION(a,b,c) (((a) << 16) + ((b) << 8) + (c))



# Örnek, 3.1.1'in elde edilmesi.

$ printf "%X\n" 196865
30101



# Örnek, 4.0.0'ın elde edilmesi.

$ printf "%X\n" 262144
40000
```



7. Dinamik Kütüphaneler Hakkında

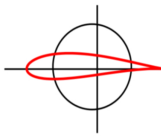
Altın kural:

Toolchain'deki kütüphaneler nerede oturuyorsa, RootFS'de de aynı yerde oturmalıdır.

-print-sysroot kullan.

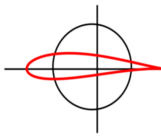
Buradaki dizin yerleşimi ile RootFS'deki Dizin yerleşimi birebir aynı olmalıdır.

Bakınız: host:/opt/gomsis/distro/6 ve Belge:Bölüm 7



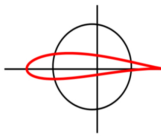
7. Temel Kütüphanelerin Yerleri, 32bit

```
/lib                        # 32 bit toolcahin, artık kullanmıyoruz.
|
|-- arm-linux-gnueabihf
|   |-- ld-2.19-2014.04.so
|   |
|   |-- libc-2.19-2014.04.so
|   |-- libc.so.6 -> libc-2.19-2014.04.so
|   |
|   |-- libdl-2.19-2014.04.so
|   |-- libdl.so.2 -> libdl-2.19-2014.04.so
|   |
|   |-- libm-2.19-2014.04.so
|   `-- libm.so.6 -> libm-2.19-2014.04.so
|
`-- ld-linux-armhf.so.3 -> arm-linux-gnueabihf/ld-2.19-2014.04.so



/lib                                    # 64 bit toolchain
|
`-- ld-linux-armhf.so.3

/usr/lib
|
|-- libc.so.6
`-- libm.so.6
```



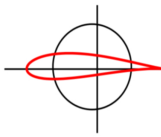
7. RootFS'in İşlenmesi, Örnek

```
#!/bin/sh +x

TFTP_ROOTFS="/tftpboot/RootFS"

root_user(){

cd $TFTP_ROOTFS/root || exit 1
rm -f .bash* .ash*
ln -f -s /tmp/ash_history .ash_history

# profile güncelle.
#
cat <<EOF >> $TFTP_ROOTFS/etc/profile

PS1="\u@\h:\w \$ "
export PS1

alias ls="ls --color"

echo
df
echo

ifconfig eth0 | head -n2
echo

uname -a
echo

EOF
```



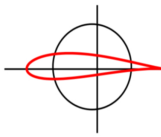
7. RootFS'in İşlenmesi, Örnek

```
# ssh'da root girişine izine ver.
#
printf "\nPermitRootLogin yes\n" >> $TFTP_ROOTFS/etc/ssh/sshd_config

# /home yarat.
#
mkdir $TFTP_ROOTFS/home

ok
}

issue(){
figlet -C utf8 UCanLinux > $TFTP_ROOTFS/etc/issue || exit 1
sed -i 's/\\/\\\\/g' $TFTP_ROOTFS/etc/issue || exit 1
echo "\nWelcome to UcanLinux\nhttp://ucanlinux.com\n" >> $TFTP_ROOTFS/etc/issue
}

root_user
issue
```



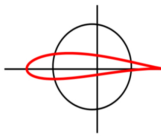
7. /etc/inittab

```
# /etc/inittab
#
# This inittab is a basic inittab sample for sysvinit, which mimics
# Buildroot's default inittab for BusyBox.
id:3:initdefault:

si0::sysinit:/bin/mount -t proc proc /proc
#si1::sysinit:/bin/mount -o remount,rw /
si2::sysinit:/bin/mkdir -p /dev/pts
si3::sysinit:/bin/mkdir -p /dev/shm
si4::sysinit:/bin/mount -a
si5::sysinit:/bin/hostname -F /etc/hostname
si6::sysinit:/etc/init.d/rcS

sole::respawn:/sbin/getty -L  console ttyS0 vt100 # GENERIC_SERIAL

# Stuff to do for the 3-finger salute
ca::ctrlaltdel:/sbin/reboot

# Stuff to do before rebooting
shd0:06:wait:/etc/init.d/rcK
shd1:06:wait:/sbin/swapoff -a
shd2:06:wait:/bin/umount -a -r

# The usual halt or reboot actions
hlt0:0:wait:/sbin/halt -dhp
reb0:6:wait:/sbin/reboot
```

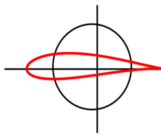
7. /etc/init.d

```
$ ls -l  /etc/init.d

total 52
-rwxr-xr-x 1 root root  423 Nis 28 23:58 rcK
-rwxr-xr-x 1 root root  408 Nis 28 23:58 rcS

-rwxr-xr-x 1 root root  404 Nis 29 00:32 S01logging
-rwxr-xr-x 1 root root 1321 Nis 28 23:58 S20urandom
-rwxr-xr-x 1 root root 1767 Nis 28 23:43 S30dbus
-rwxr-xr-x 1 root root  359 Nis 28 23:58 S40network
-rwxr-xr-x 1 root root  675 Nis 29 00:09 S49ntp
-rwxr-xr-x 1 root root  476 Nis 29 00:05 S50lighttpd
-rwxr-xr-x 1 root root  530 Nis 29 00:10 S50sshd
-rwxr-xr-x 1 root root 1088 Nis 28 23:43 S80dhcp-relay
-rwxr-xr-x 1 root root 1090 Nis 28 23:43 S80dhcp-server
-rwxr-xr-x 1 root root 1421 Nis 29 00:33 S80tftpd-hpa
-rwxr-xr-x 1 root root  494 Nis 28 23:12 S99at
```

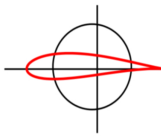
7. /etc/init.d/rc.[SK]

```
# /etc/init.d/rcS özeti

#!/bin/sh

for i in /etc/init.d/S??*
    do
    $i start
    done



# /etc/init.d/rcK özeti.

#!/bin/sh

for i in $(ls -r /etc/init.d/S??*)
    do
    $i stop
    done
```



7. /etc/init.d/S??* Özeti

```
#!/bin/sh
#

case "$1" in

  start)
     echo "Starting command..."
     command_start
     ;;

  stop)
     printf "Stopping command..."
     command_stop
     ;;

  restart|reload)
     "$0" stop
     "$0" start
     ;;

  *)
     echo "Usage: $0 {start|stop|restart}"
     exit 1
esac

exit $?
```

7. Sistem, Test

```
# rootfs.tar dosyası /tftpboot altına açılır.
# Eski RootFS'in bozulmaması için adı değiştirilir.

$ cd /tftpboot
$ mv RootFS RootFS.ydk

$ mkdir RootFS
$ cd RootFS
$ tar xvf /opt/gomsis/src/br/output/images/rootfs.tar

# Burada gerekirse update.rootfs çalıştır ve güncellemeler
# otomatik olarak yapılsın.
# Güncellemeleri el ile yapma, betik ile yap.


# 3. sistem ile NFS üzerinden açılış yapılır.

# rootfs.tar daha önceden kurulan herhangi bir sistem ile denenebilir.
```

Bu son sayfadır.