1 Netzwerkboot

Um das Projekt zu realisieren wurde das Team in Mehrere Gruppen aufgeteilt. Jonas und Alexander versuchten den Netzwerkboot umzusetzen.

Beim Netzwerkboot war das Ziel, dass der Pi 3 der im Zug verbaut ist, beim Start nicht ber die SD Karte oder ein anderes angeschlossenes Speichergert Startet, sondern ber ein Betriebssystem Image das auf einem DHCP/TFTP Server liegt. Der groe Vorteile hierbei ist, dass es knftig keine Probleme mehr mit kaputten SD Karten gibt.

In ersten Tests richteten wir eine Pi 3 VM auf dem PC ein. Diese diente als Server einen Pi 3 den wir zur Verfgung hatten richteten wir als Client ein.

1.1 Client Konfiguration

Der Client muss zuerst mithilfe einer SD Card gebootet werden. Mit dem Befehl

echo program_usb_boot_mode=1 — sudo tee -a /boot/config.txt

schalten wir den USB boot mode ein. Um zu berprfen ob der OTP Speicher im Pi nun richtig konfiguriert ist, starten wir den Pi neu und fhren den Befehl

vcgencmd otp_dump — grep 17:

aus. Die Ausgabe muss 0x3020000a sein. Falls die Ausgabe Korrekt ist, entfernen wir nun wieder den Eintrag in der config.txt. Der Client ist nun Konfiguriert.

1.2 Server Konfiguration

Bei der Server Konfiguration erweitern wir das Filesystem auf die komplette SD Karte mit dem Konfigurationstool

sudo raspi-config

Da der Client ein root filesystem bentigt zum booten, Erstellen wir ein Abbild des aktuellen Filesystem und hinterlegen es im Verzeichniss /nfs/client1.

sudo mkdir -p /nfs/client1 sudo rsync -xa –progress –exclude /nfs / /nfs/client1

Nun mssen die SSH Schlssel erneuert werden

cd /nfs/client1 sudo mount –bind /dev dev sudo mount –bind /sys sys sudo mount –bind /proc proc sudo chroot . rm /etc/ssh/ssh_host_* dpkg-reconfigure openssh-server exit sudo umount dev sudo umount sys sudo umount proc

wir bentigen nun noch die Netzwerkinformationen. Hierzu muss der Pi mit dem Netzwerk verbunden sein.

ip route — grep default — awk 'print $3' ip -4 addr show dev eth0 — grep inet

Die Ausgabe sollte so aussehen

inet 192.168.1.101/24 brd 192.168.1.255 scope global eth0

Server IP Adresse: 192.168.1.101 Brodcast Adresse: 192.168.1.255 eth0: Verbindung ber LAN Kabel

die IP Adresse vom DNS Server bekommen wir mit

cat /etc/resolv.conf

der Pi selber bentigt noch eine Statische IP Adresse

sudo nano /etc/network/Interfaces

Hier die Zeile

iface eth0 inet manual

ersetzen mit

auto eth0 iface eth0 inet static address 192.168.1.2 netmask 255.255.255.0 gateway 192.168.1.1

Die gateway Adresse ist die Adresse die wir aus resolv.conf ausgelesen haben

den DHCP client Daemon mssen wir auch noch ausschalten

sudo systemctl disable dhcpcd sudo systemctl enable Networking

sudo reboot

Mit dem Neustart wurden unsere nderungen bernommen. Wir haben nun kein funktionierendes DNS mehr. Wir fgen nun unsere Server Konfiguration ein. nameserver entspricht der gateway Adresse.

echo "nameserver 192.168.1.1" — sudo tee /etc/resolv.conf

die Datei darf durch dnsmasq nicht mehr verndert werden weshalb wir sie mit dem Befehl

sudo chattr +i /etc/resolv.conf

unvernderbar machen. Es ist nun zwingend erforderlich zustzliche Software zu installieren

sudo apt-get update sudo apt-get install dnsmasq tcpdump

Mit dem nchsten Befehl stoppen wir die DNS Namensauflsung mit dnsmasq

sudo rm /etc/resolvconf/update.d/dnsmasq sudo reboot

Wir starten nun tcpdump um nach DHCP Paketen vom Client Raspberry Pi zu suchen

sudo tcpdump -i eth0 port bootpc

1.3 Inbetriebnahme

Nachdem nun auch die Netzwerk config abgeschlossen ist schlieen wir den Client Raspberry ans Netzwerk an und starten ihn. Nach 10 Sekunden sollten wir auf dem Client beobachten knnen, dass die LED anfngt zu leuchten. Auf dem Bildschirm sollten wir nun Pakete mit der Bezeichnung BOOTP/DHCP, Request bekommen.

IP 0.0.0.0.bootpc ¿ 255.255.255.255.bootps: BOOTP/DHCP, Request from b8:27:eb...

Der Server empfngt nun Anfragen vom Client kann aber noch nicht Antworten. Hierfr mssen wir erst noch dnsmasq konfigurieren

echo — sudo tee /etc/dnsmasq.conf sudo nano /etc/dnsmasq.conf

Den gesamten Inhalt der Datei ersetzen durch

port=0 dhcp-range=192.168.1.255,proxy log-dhcp enable-tftp tftp-root=/tftpboot pxe-service=0,“Raspberry Pi Boot”

dhcp-range = Brodcast Adresse

Das Verzeichniss /tftboot mssen wir noch erstellen

sudo mkdir /tftpboot sudo chmod 777 /tftpboot sudo systemctl enable dnsmasq.service sudo systemctl restart dnsmasq.service

Wir berwachen jetzt das dnsmasq Protokoll mit

tail -F /var/log/daemon.log

und suchen dort einen Eintrgen die ungefhr so aussehen

raspberrypi dnsmasq-tftp[1903]: file /tftpboot/bootcode.bin not found

wir holen uns nun die Datei bootcode.bin und start.elf in unser /tftboot Verzeichniss indem wir das komplette boot Verzeichnis kopieren

cp -r /boot/* /tftpboot sudo systemctl restart dnsmasq

Fr einen ersten Test verwendeten wir nun das am Anfang kopierte root filesystem in /nfs/client1 dieses sollte spter durch unser in Yocto erstelltes root filesystem ersetzt werden

sudo apt-get install nfs-kernel-server echo “/nfs/client1 *(rw,sync,no_subtree_check,no_root_squash)” — sudo tee -a /etc/exports sudo systemctl enable rpcbind sudo systemctl restart rpcbind sudo systemctl enable nfs-kernel-server sudo systemctl restart nfs-kernel-server

/tftpboot/cmdline.txt muss nun noch angepasst werden

root=/dev/nfs nfsroot=192.168.1.2:/nfs/client1 rw ip=dhcp rootwait elevator=deadline

Die IP Adresse entspricht der vom Client. Abschlieend mssen noch die SD Karten Eintrge in fstab gelscht werden

sudo nano /nfs/client1/etc/fstab

Hier die Eintrge

/dev/mmcblkp1 /dev/mmcblkp2

lschen. Nun sollte der Netzwerkboot richtig konfiguriert sein.

1.4 rsync

Leider funktionierte der Netzwerkboot nicht. Wir haben im Internet noch diverse andere Anleitungen ausprobiert aber alle ohne Erfolg. In einem Versuch hatten wir das System soweit, dass sich der Client ein Paket vom Server holte, danach aber aufhrte. Als Alternative zum Netzwerkboot hinterlegten wir nun ein root filesystem auf dem Server im Labor im Verzeichnis /projects/ss18_fhftrain/rfs. Dieses ist erreichbar ber

Server:141.28.57.62 Port: 12345

Das root filesystem ist mithilfe von Yocto erstellt worden und luft auch auf dem Pi. Mithilfe von rsync bernimmt der Pi beim Einschalten alle nderungen die im Verzeichnis /projects/ss18_fhftrain/rfs gemacht wurden.

Als erstes muss rsync installiert werden

sudo apt-get install rsync

Die Syntax des rsync Befehls sieht folgendermaen aus

rsync [OPTIONEN] QUELLE(N) ZIEL

Quelle: /projects/ss18_fhftrain/rfs/ Ziel: /

Optionen: Es ist wichtig einige Pfade und Ordner zu excludieren. Diese enthalten unter anderem Hardware spezifische Daten die nicht berschrieben werden drfen. Das Verzeichnis mit dem rsync Befehl mssen wir auch excludieren.

–exclude=/mnt –exclude=/media –exclude=/sys –exclude=/tmp –exclude=/var/log –exclude=/etc –exclude=/dev –exclude=/proc –exclude=/root –exclude=/boot –exclude=/lib/modules –exclude="lost+found"

–exclude=/home/pi/Programm –exclude=/home/pi/.ssh

Andere Optionen:

-vvv gibt uns whrend des Synchronisierens Debuginfos aller ausgefhrter Schritte aus

-av

–delete vergleicht Quellverzeichnisse und Zielverzeichnisse und sorgt dafr, dass Dateien, die im Quellverzeichnis nicht (mehr) vorhanden sind, im Zielverzeichnis gelscht werden