**1. Backup und Wiederherstellung des Betriebssystems**

Um die Implementationen und Konfigurationen des Pi's abzusichern, wird eine Backup Lösung eingesetzt. Die Sicherung relevanter Daten soll hierbei einem möglichen Defekt oder Inkompatibilität durch Updates o.ä. entgegenwirken.

'FauBackup' und 'gitbac' bieten hierbei eine Lösung mittels externer Programme an.

Debian bietet jedoch bereits standardmäßig einige Aufrufe welche zur Anlegung von Backups verwendet werden können. Diese wurden im Folgenden in Ubuntu anhand eines Bash Skripts zur Automatisierung getestet und implementiert.

**1.1 dd**

Als erste Ausführung wurde der Aufruf 'dd' verwendet. Hierbei wird der Inhalt der gesamten SD Karte als Image Datei abgespeichert

1.

Übersicht der vorhandenen Dateisysteme mittels des Terminalaufrufs 'df'

[BILD1 Vorhandene Dateisysteme]

2.

Terminalaufruf für den Backupprozess

sudo dd if=INPUTPARTITION of=OUTPUTFILE

sudo -> Backupprozess benötigt root-Rechte

dd -> bit-genaues Kopieren der Dateien

if=FILE -> Die Datei oder Partition welche integriert wird

of=FILE -> Die Output Datei welche angelegt wird

3.

Optionale Nutzung von Terminalaufruf 'pv' um den Fortschritt des Backup Prozesses zu sehen

Die mögliche Restzeit lässt sich nur durch das Hinterlegen der Größe der Partition anzeigen.

sudo dd if=INPUTPARTITION |pv| sudo dd of=OUTPUTFILE

[BILD2 Laufender Backupprozess mit Statusanzeige]

4.

Wiederherstellen eines hinterlegten Backups läuft ähnlich wie der ursprüngliche Prozess ab.

sudo dd if=OUTPUTFILE |pv| of=INPUTPARTITION

[BILD3 Laufender Wiederherstellungsprozess mit Statusanzeige]

Der Aufruf 'dd' führt zu vielen unnötigen Schreibprozessen was schädlich für die hinterlegte SSD ist. Daher muss die Blockgröße von standardmäßig 512 Bytes durch Nutzung des Parameters 'bs='auf ein vielfaches der SSD gesetzt werden.

**1.2 rsync**

Da der Vorgang mittels 'dd' als suboptimal angesehen wird, wurde alternativ der Aufruf 'rsync' verwendet. Dieser bietet die Möglichkeit eines inkrementellen Backups wodurch die Dauer des Prozesses erheblich reduziert werden kann. Hierbei werden die Größe und die Änderungszeit der Dateien in Quelle und Ziel miteinander verglichen. Eine Aktualisierung findet demnach nur statt, wenn Unterschiede vorzufinden sind.

rsync -aAXv --delete --exclude={"/dev/\*","/proc/\*","/sys/\*","/tmp/\*","/run/\*","/mnt/\*","/media/\*","/lost+found"} / /path/to/backup/folder

rsync -> Kopieren der Dateien

-aAX -> Übertragung im Archiv Modus wodurch alle symbolischen Verweise beibehalten werden

--delete -> Dateien die im Ursprungsverzeichnis nicht mehr existieren werden im Zielverzeichnis ebenfalls gelöscht

--exclude -> Dateien werden ausgelassen

Wiederherstellen des Rsync Backups durch folgenden Befehl:

rsync -aAXv /path/to/backup/location/\* /mount/point/of/new/install/ --exclude={/dev/\*,/proc/\*,/sys/\*,/tmp/\*,/run/\*,/mnt/\*,/media/\*,/lost+found,/home/\*}

**1.3 tar**

Eine weitere Anwendungsmöglichkeit bietet die 'tar' Archivierung. Vorteil dieses Aufrufs ist, dass durch Angabe von Parametern die Berechtigungen aller zu sichernden Daten ebenfalls beibehalten werden und die Archivierung Speicherplatz spart.

Wechsel in das Backupverzeichnis dann:

tar -cpzf Backup.tar ORDNER

tar -> Archivieren von Daten

-c -> Archiv wird erzeugt (create)

-p -> Berechtigungen beibehalten (privilige)

-z -> Zusätzliche Komprimierung mit gzip

-f -> Archiv in Datei schreiben (finish)

**1.4 Automatisierung mittels Bash-Skript**

Damit der Nutzer die Aufrufe nicht händisch zu bestimmten Zeiten ausführen muss, wurden zwei Bash-Skripte zur Automatisierung geschrieben. Es gibt ein monatliches Backup mittels (tar) und wöchentliche inkrementelle Backups (rsync) auf die zurückgegangen werden kann.

Um das Zeitintervall der Backups einzustellen wird der 'Cron' Dienst verwendet.

Hiermit können Skripte und Programme zu festgelegten Zeiten gestartet werden.

Wenn ein hinterlegter Job täglich zu einer bestimmten Uhrzeit ausgeführt wird muss allerdings auch der Rechner zu dem Zeitpunkt aktiv sein. Ist dies nicht der Fall, startet der Prozess nicht. Um dies zu umgehen wird 'Anacron' verwendet.

Durch ablegen des Skripts in eines der entsprechenden Verzeichnisse wird der Prozess entsprechend ausgeführt.

/etc/cron.hourly/ - Stündlich ausführen

/etc/cron.daily/ - Täglich ausführen

/etc/cron.weekly/ - Wöchentlich ausführen

/etc/cron.monthly/ -Monatlich ausführen

[BILD4 Ausführung wöchentliches Backup]

[BILD5 Ausführung monatliches Backup]

**2. Update des Banana Pi**

Ziel war es das Betriebssystem und alle Programme immer auf dem aktuellsten Stand zu halten. Hierbei kann es jedoch zu Inkompatibilität bestimmter Funktionen oder Konfigurationen kommen. Daher wurde die Umsetzung auf die relevantesten Updates (Sicherheitsupdates) reduziert. Im Normalfall können mittels des Konsolenaufrufs 'apt-get update' die Updateliste und mit 'apt-get upgrade' die Programm Pakete selbst aktualisiert werden. Durch Nutzung des Aufrufs 'unattended-upgrade' wird auf Sicherheitsupdates des Systems überprüft und diese anschließend installiert.

Durch 'unattended-upgrade –dry-run -d' wird auf Verfügbarkeit von Updates geprüft ohne anschließende Installation. Nach jedem Durchgang wird eine Logdatei in /var/log/unattended-upgrades/ angelegt welche genauere Informationen zu den aktualisierten Dateien liefert.

**3. Implementierung einer Displaystatusanzeige**

TODO

- Skripte in Anhang!

- Update des Bananian Systems überarbeiten

- Text zu 'tmux'

- Beispielbilder Ubuntu / Bananian