Projektaufgabe Embedded Systems

Emmanuel Panagou, 1283985

09.09.2024

Inhaltsverzeichnis

1 Unterschriebene Eigenständigkeitserklärung

Eidesstattliche Erklärung
zur Projektarbeit Eingebettete Systeme im SS2024

Name: Emmanuel Panagou
Matrikelnummer: 1283985

Ich versichere durch meine Unterschrift, dass die vorgelegte Arbeit ausschließlich von mir erstellt und verfasst wurde. Es wurden keine anderen als die von mir angegebenen Quellen und Hilfsmittel benutzt.

Viersen, 09.09.2024 _____
Ort, Datum Unterschrift

2 Angaben

2.1 Persönliche Angaben

Name: Emmanuel Panagou

Matrikelnummer: 1283985

Studiengang: Master Informatik

Datum: 9.9.2024

2.2 Systemangaben

2.2.1 Entwicklungsrechner

Host: Thinkpad E15 Gen 4
OS: Ubuntu 24.04.1 LTS

2.2.2 Smartphone

Host: Google Pixel 4
OS: Android 13
MQTT-App: MyMQTT & MQTT Dashboard Client

3 Einleitung

3.1 Motivation

Diese Projektarbeit befasst sich der praktischen Anwendung von diversen Embedded-Systems-Technologien, insbesondere der Verwendung von Buildroot und dem Netzwerk-Boot auf einem Raspberry Pi 4.

Ich habe ein persönliches Interesse an dieser Projektarbeit, da mich das Thema "Embedded Systems" bereits seit dem Bachelor fasziniert und deswegen auch das Wahlpflichtmodul "Microcontroller" absolviert habe. Mein Interesse stammt unter anderem daher, dass Embedded-Systeme in unserer heutigen Zeit immer mehr an Bedeutung gewinnen und in immer mehr Gebieten eingesetzt werden.

Da ich dieses Themengebiet auch für meine Masterarbeit und berufliche Laufbahn in Betracht ziehe, aber zuvor nur vereinzelt Kontakt mit hatte, dient mit sowohl diese Projektarbeit, als auch das gesamte Modul als tieferer Eindruck in dieses Thema.

3.2 Aufgabenstellung

Die Projektarbeit beinhaltet drei Kernaufgaben. Diese umfassen:

1. **Networkboot und Systemkonfiguration**
   Das eingebette System für den Raspberry Pi 4 soll per TFTP booten und diverse Funktionen bereitstellen.
   - Es sollen zwei Logins eingerichtet werden:
     - **Root-Login**
       - Username: `root`
       - Passwort: `root`
     - **User-Login**
       - Username: `Panagou`
       - Passwort: `Qbobhpv`
   - Der Pi soll den Hostnamen `lilibiti80` besitzen.
   - Es soll ein WLAN-Netzwerk zur Verfügung gestellt werden:
     - SSID. `VimIsTheBest`
     - Passwort: `ancprcuuaq`
     - Adressraum: `192.168.123.0/24`
   - Es soll möglich sein, sich per SSH auf dem Pi einzuloggen:
     - IP-Adresse von eth0: `192.168.42.69`

2. **Gerätetreiber**
   - Das System besitzt einen Gerätetreiber `signalan`
   - Der Gerätetreiber soll eine LED über die Gerätedatei `/dev/led_onoff_an` ansteuern können
   - Die LED ist an GPIO-Pin `23` angeschlossen

3. **MQTT**
   - Auf dem Pi soll ein MQTT-Broker beim Hochfahren gestartet werden
   - Der MQTT-Broker veröffentlicht diverse Topics
   - Auf einem Smartphone soll ein MQTT-Client die veröffentlichten Topics darstellen und die Blinkfrequenz der LED einstellen können

4 Installationsanleitung

Hier folgen alle Dinge, die man zum Nachbau wissen muss. Dabei ist nicht jeder einzelne Schritt gemeint, aber eine Beschreibung, wie man die Buildroot-Konfig auf Basis der `.config`*-Datei erstellt und wohin man welche Dateien der Lieferung einspielen muss, welche Konfigurationen anzupassen sind.*

Hier werden unter anderem folgende Dinge erwartet:

- *Aufsetzen der Entwicklungsumgebung/Testumgebung (beteiligte Komponenten, Hardware, Software, Host, Target, VirtualBox, Switch, Smartphone...)*
- *Hinweise zur Konfiguration*
- *Screenshot vom Hardwareaufbau*
- *Systemgenerierung*
- *Aufsetzen und Konfiguration Smartphone-App*
- *Screenshot von der Smartphone-App (Dashboard)*
- Networkadapter LAptop
- Pi konfiguration mit sd karte
-

5 Systemtest

5.1 Komponententest

Bevor das Gesamtsystem getestet wird, werden zunächst die einzelnen Komponenten größtenteils unabhängig voneinander auf Funktionalität und korrekte Konfiguration getestet.
Ziel ist es festzustellen, ob alle geforderten Aufgaben und die Individualisierung erfüllt wurden.

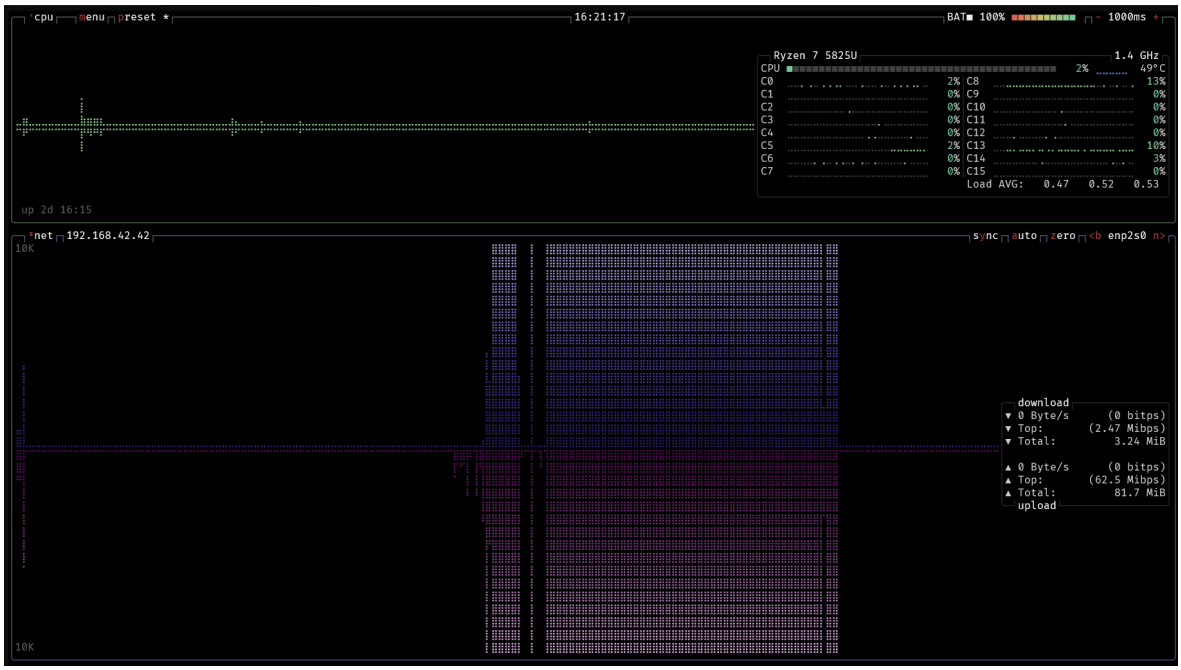
Für die folgenden Tests ist das System wie in `Kapitel 4: Installationsanleitung` beschrieben aufgebaut und konfiguriert.

5.1.1 Grundkonfiguration des Raspberry Pi 4

In diesem Abschnitt wird die Grundkonfiguration des Raspberry Pi 4 getestet. Diese umfasst die Korrekte Konfiguration:

- Des Networkboot per TFTP
- Des Hostnames
- Der User
- Des LAN-Port `eth0`
- Des SSH-Servers

Der Pi wird durch anschließen des Stromkabels gestartet. Auf dem Hostrechner kann durch die Netzwerkauslastung am LAN-Port beobachtet werden, ob der TFTP-Server die Daten für den Raspberry Pi zur Verfügung stellt. Dafür geeignet sind Programme wie `btop`:



Des Weiteren kann der Output des Befehls `tail -f /var/log/syslog | grep tftp` Schlüsse auf die korrekte Funktionsweise des TFTP-Servers geben.

Nachdem der Pi gestartet wurde kann versucht werden sich über SSH auf diesem einzuloggen. Dabei wird als User `Panagou` und als IP-Adresse `192.168.42.69` angegeben. Wird die Verbindung erfolgreich aufgebaut, kann sich mit dem Passwort `Qbobhpv` eingeloggt werden.

```
nemdos@ThinkpadUbuntu ~ $ ssh Panagou@192.168.42.69
Panagou@192.168.42.69`s password: Qbobhpv
```

Nach dem Einloggen kann durch den Befehl `hostname` die Konfiguration des Hostnames überprüft werden.

```
$ hostname
lilibiti80
```

5.1.2 Konfiguration des Access-Points über den Raspberry Pi 4

In diesem Abschnitt wird die Konfiguration des Raspberry Pi 4 als Access-Point getestet. Diese umfasst:

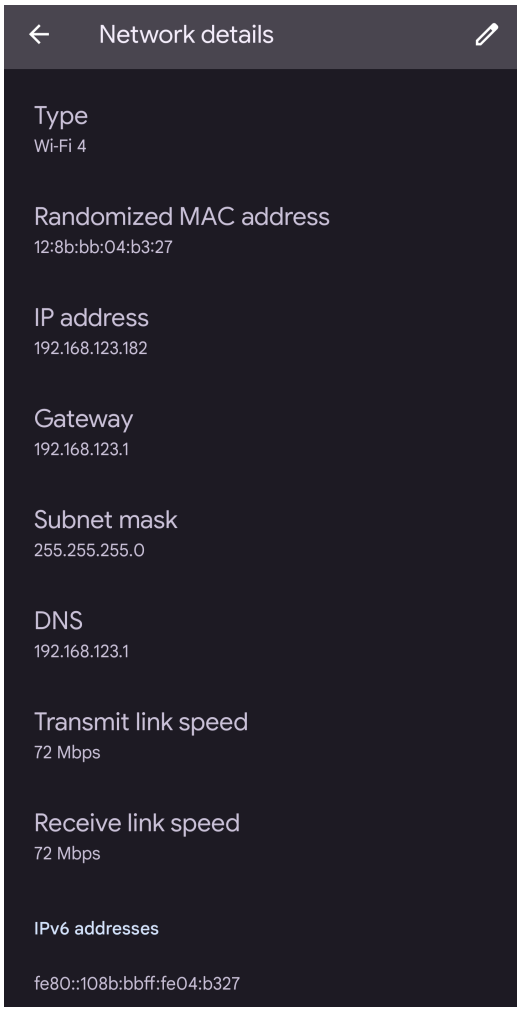
- Das erfolgreiche Aufspannen des WLAN-Netzes
- Den Namen und das Passwort des Netzes
- Der Adressraum des Netzes
- Die Konfiguration des `wlan0`-Interfaces

Durch aufrufen des Befehls `ifconfig wlan0` ist es möglich die Konfiguration des Interfaces einzusehen.

```
$ ifconfig wlan0
wlan0     Link encap:Ethernet  HWaddr DC:A6:32:9B:B1:30
          inet addr:192.168.123.1  Bcast:192.168.123.255  Mask:255.255.255.0
          inet6 addr: fe80::dea6:32ff:fe9b:b130/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:12 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:936 (936.0 B)
```

Ob das WLAN-Netz korrekt aufspannt ist, lässt allerdings direkt mit dem Smartphone testen. Dazu wird mit diesem versucht sich mit dem Netz zu verbinden.
Nach öffnen der WLAN-Einstellungen des Smartphone sollte das WLAN-Netz auswählbar und mittels des Passwortes `ancprcuuaq` nutzbar sein.

In der Detailansicht der WLAN-Verbindung kann nun überprüft werden, ob der der AP des Pi korrekt Adressen aus dem Netz `192.168.123.0/24` verteilt.



5.1.3 Konfiguration des Gerätetreibers und Ansteuerung der LED

In diesem Abschnitt wird die Funktionalität und Ansteuerung der LED über den Gerätetreiber `signalan` getestet. Dies beinhaltet:

- Der Gerätetreiber wird während des Hochfahrens des Systems geladen
- Die Gerätedatei wird automatisch angelegt
- Durch lesen des Gerätetreibers kann die aktuelle Blinkfrequenz der LED ausgelesen werden
- Durch schreiben des Gerätetreibers kann die aktuelle Blinkfrequenz der LED geändert werden

---

Ob der Gerätetreiber erfolgreich geladen wurde, lässt sich einfach mittels des Befehls `lsmod` ermitteln. War das Laden erfolgreich, so wird der Gerätetreiber in der Ausgabe aufgelistet.

```
$ lsmod
Module                  Size  Used by
signalan               12288  0
ipv6                  528384  32
```

Ähnliches Vorgehen kann das Anlegen der Gerätedatei überprüfen. Dazu wird der Befehl `ls /dev/ | grep led_onoff_an` verwendet. Dieser listet alle Gerätedateien des Systems auf und sucht darin nach der speziellen Gerätedatei. Falls die Gerätedatei nicht angelegt wurde, gibt dieser Aufruf nichts zurück.

```
$ ls /dev/ | grep led_onoff_an
led_onoff_an
```

Um das Lesen und Schreiben auf der Gerätedatei testen zu können ist es notwendig sich mit dem root-User einzuloggen.

```
$ su
Password: root
```

Nun kann die Frequenz der LED aus der Gerätedatei ausgelesen, anschließend geändert und abschließend erneut ausgelesen werden. Dabei sollte die LED an GPIO-Pin 23 des Raspberry Pi angeschlossen sein, um eine visuelle Bestätigung der Änderung der Blinkfrequenz bieten zu können.

```
$ cat /dev/led_onoff_an
1
$ echo -e -n "\x05" >/dev/led_onoff_an
$ cat /dev/led_onoff_an
5
```

5.1.4 Konfiguration Mosquitto und MQTT-App

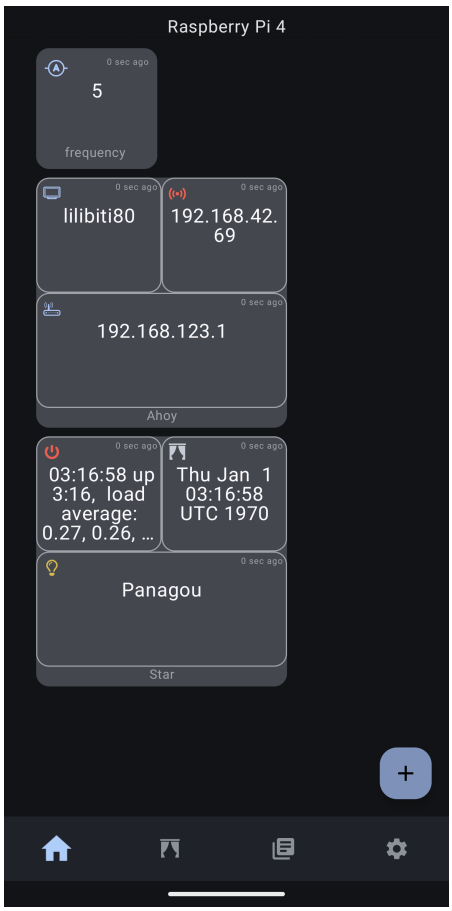
In diesem Abschnitt wird die korrekte Konfiguration von Mosquitto und der MQTT-App betrachtet. Dies schließt folgende Aspekte ein:

- Mosquitto wird beim Hochfahren des Systems automatisch gestartet
- Mosquitto veröffentlicht regelmäßig mehrere Topics, die verschiedene Systeminformationen beinhalten
- Die MQTT-App ist in der Lage die Frequenz der LED über das Topic `appl/frequency_set`

Wie bei den bereits vorangegangen Tests kann das Starten von `mosquitto` durch einen Befehl überprüft werden. `ps | grep mos` gibt eine Liste der momentan laufenden Programme zurück, welche durch den `grep`-Befehl auf die mit `mosq` im Namen gefiltert werden.

```
$ ps | grep mosq
  166 mosquitt /usr/sbin/mosquitto -c /etc/mosquitto/mosquitto.conf
  182 root     sh /etc/init.d/mosq_pub.sh
  185 root     mosquitto_sub -t appl/frequency_set
  ...
```

Um zu überprüfen, ob die Topics erfolgreich veröffentlicht werden, kann erneut das Smartphone verwendet werden. Dafür wird sichergestellt, dass dieses mit dem WLAN-Netzwerk des Raspberry Pi verbunden ist. Anschließend kann die MQTT-App geöffnet und die eingerichteten Topics beobachtet werden. Diese sollten sich regelmäßig aktualisieren. Des Weiteren kann hier das setzen einer neuen Blinkfrequenz über die App getestet werden. Wenn das entsprechende Topic eingerichtet ist, sollte ein Antippen genügen, um die LED steuern zu können.



Hierbei gibt es eine Auffälligkeit bei dem Topic `system/date`. Das angezeigte Datum und die Uhrzeit sind nicht korrekt. Dies lässt sich allerdings einfach dadurch erklären, dass der Raspberry Pi aktuell keine Verbindung zum Internet besitzt. Die Verbindung ist Notwendig, um das korrekte Datum und die korrekte Uhrzeit setzen zu können.

5.2 Test des Gesamtsystems

Nach den Tests der einzelnen Komponenten, folgt nun der Test des Gesamtsystems.
Ziel dieses Tests ist es, festzustellen, ob alle Komponenten korrekt miteinander interagieren.
Die Funktionalität der einzelnen Komponenten, wie dem Gerätetreiber und dem MQTT-Broker, müssen nicht nur in isolierten Tests, sondern auch im normalen Betrieb problemlos funktionieren.

Das System ist wie in `Kapitel 4: Installationsanleitung` beschrieben aufgebaut und konfiguriert.

- Die LED ist mit GPIO-Pin 23 verbunden
- Das aufgespannte WLAN-Netz trägt den Namen `VimIsTheBest` und vergibt Adressen aus dem Netz `192.168.123.0/24`
- Die MQTT-App ist zum Empfangen und Verschicken aller Topics korrekt konfiguriert

5.2.1 Testdurchführung

Der Raspberry Pi 4 wird durch anschließen des Stromkabels eingeschaltet.
Ob der Netzworkboot erfolgreich war, ist anhand der LED feststellbar. Fängt diese nach dem Hochfahren an auf der niedrigsten Geschwindigkeit zu blinken, war das Booten per TFTP erfolgreich.

Im nächsten Schritt wird die WLAN-Verbindung mit dem Smartphone hergestellt. Dazu wird das WLAN-Menü des Smartphones geöffnet und durch Antippen des Eintrags `VimIsTheBest` eine Verbindung hergestellt. Sobald die Verbindung aufgebaut wurde, kann die MQTT-App geöffnet werden.

Auf dem Dashboard sollten nun alle Topics einsehbar sein. Im Folgenden kann nun getestet werden, ob durch versenden des Topics `appl/frequency_set` die Blinkfrequenz der LED gesteuert werden kann.

6 Zusammenfassung

Eine Zusammenfassung wiederholt kurz* die Aufgabenstellung, danach folgen die eigentlichen Inhalte und schließlich wird das Ergebnis beschrieben.*

Meistens gibt es noch einen kurzen Ausblick, was noch zu tun ist beziehungsweise was noch getan werden kann.