Eingebettete System - Projektaufgabe SS2024

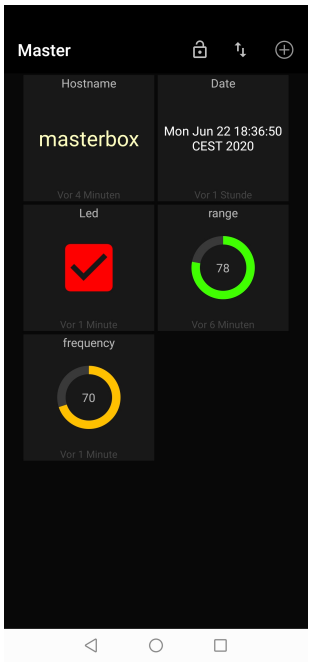
Prof. Dr.-Ing. Jürgen Quade, Hochschule Niederrhein

5.07.2024

Allgemeines

Die Prüfung im SS 2024 findet in Form einer benoteten, selbstständigen Projektarbeit statt. Im Rahmen dieser Projektarbeit erstellen Sie ein Smarthome-Gateway, über das mit Hilfe des Kommunikationsprotokolls MQTT technische Prozesse überwacht und gesteuert werden können. In der vorliegenden Aufgabe steuern Sie per Smartphone damit eine LED an und visualisieren betriebssystemrelevante Informationen wie beispielsweise *Name des Gateways*, *Laufzeit* oder *IP-Adresse*. Informationen zu MQTT finden Sie weiter unten. Beachten Sie bitte, dass die Aufgabenstellung in einigen Details **individualisiert** ist.

Zur Projektaufgabe gehört neben der Erstellung des Gateways eine Dokumentation im Markdown-Format.



Voraussetzung

Basis für die Aufgabe ist das während des Semesters entwickelte Buildroot-System, das folgende Komponenten enthält:

- SSH-Server (`dropbear`)
- WLAN-AP (`wpa_supplicant`, inklusive `dnsmasq`)
- Integrierter Treiber zur Ansteuerung einer LED

Formale Randbedingungen

- Ausgabe der Aufgabenstellung: 15.7.2024
- Letzer Abgabetermin: 9.9.2024 - 23.59 Uhr
- Arbeitsumfang: ca. 40h
- Moodle-Exam-Raum zur Abgabe der Dokumentation: https://moodle-exam.hsnr.de/course/view.php?id=4298
- Sciebo-Ordner zur Abgabe der TAR-Datei: https://hs-niederrhein.sciebo.de/s/OqBrUffr1kFAEPC

Vorlage für die Eigenständigkeitserklärung

```
Eidesstattliche Erklärung
zur Projektarbeit Eingebettete Systeme im SS2024


Name:
Matrikelnummer:

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.

---------------             ------------------
Ort, Datum                  Unterschrift
```

Abgabe

Einzureichen sind die folgenden Dateien:

- PDF der Dokumentation
- Tar-Datei, die die zum Nachbau notwendigen Dateien enthält

Generieren der Tar-Datei

Zur Generierung des *Tape Archives* (TAR) legen Sie am besten unterhalb von `~/embedded/` einen Ordner `export` und hier weitere Unterordner an:

```
mkdir ~/embedded/export
cd ~/embedded export
mkdir documentation/ images/ configs/ driver/ scripts/ target/
```

Kopieren Sie danach die benötigten Dateien in die zugehörigen Unterverzeichnisse. Danach erzeugen Sie die Tar-Datei.

```
cd ~/embedded/
tar cvfz 2024-09-08-esy-<IHR NAME>.tgz export/
```

Das TAR **muss** nebst anderen die nachfolgenden Dateien zwingend enthalten:

- Image des selbstgenerierten Userlands “rootfs.cpio”
- Image des selbstgenerierten Kernels “kernel7l.img”
- Buildroot-Konfigurationsdatei “.config” (bitte “`config.buildroot`” nennen)
- Kernel-Konfigurationsdatei “.config” (bitte “`config.linux`” nennen)
- Quellcode des Linux-Treibers `signalan.c`
- `post-build.sh`
- `post-image.sh`
- Entwicklungs-Dokumentation (mit unterschriebener, eingescannter Eigenständigkeitserklärung und Smartphone-Screenshot) sowohl als PDF als auch als Markdown-Datei
- Inhalte der Entwicklungs-Ordner `target/` und `scripts/` (beziehungsweise der Verzeichnisse, in denen Sie die Daten für das Target und die Hilfsskripte für die Generierung abgelegt haben)

Nachfolgend ein beispielhafter Inhalt:

```
tar tvf ~/embedded/2024-09-08-esy-quade.tgz
drwxrwxr-x quade/quade       0 2024-07-05 13:34 export/
drwxrwxr-x quade/quade       0 2024-07-05 13:33 export/images/
-rw-r--r-- quade/quade 18547200 2024-07-05 13:32 export/images/rootfs.cpio
-rw-rw-r-- quade/quade  7713536 2024-07-05 13:32 export/images/kernel7l.img
drwxrwxr-x quade/quade       0 2024-07-05 13:34 export/configs/
-rw-r--r-- quade/quade   107586 2024-07-05 13:33 export/configs/config.buildroot
-rw-rw-r-- quade/quade   210659 2024-07-05 13:33 export/configs/config.linux
drwxrwxr-x quade/quade       0 2024-07-05 13:36 export/driver/
drwxrwxr-x quade/quade       0 2024-07-05 13:38 export/driver/2024-ledone/
-rw-r--r-- quade/quade     758 2024-07-05 13:36 export/driver/20240607-ledone/blink.c
-rw-rw-r-- quade/quade     398 2024-07-05 13:36 export/driver/20240607-ledone/Makefile.applicatio
```

```
-rw-r--r-- quade/quade      290 2024-07-05 13:36 export/driver/20240607-ledone/Makefile
-rw-rw-r-- quade/quade      431 2024-07-05 13:36 export/driver/20240607-ledone/export.sh
-rw-r--r-- quade/quade     3728 2024-07-05 13:36 export/driver/20240607-ledone/led_one.c
drwxrwxr-x quade/quade        0 2024-07-05 13:35 export/scripts/
-rwxr-xr-x quade/quade     1810 2024-07-05 13:35 export/scripts/post-build.sh
-rwxr-xr-x quade/quade      287 2024-07-05 13:35 export/scripts/post-image.sh
drwxrwxr-x quade/quade        0 2024-07-05 13:34 export/documentation/
-rw-r--r-- quade/quade      483 2024-07-05 13:34 export/documentation/Makefile
-rw-rw-r-- quade/quade   209899 2024-07-05 13:34 export/documentation/2024-07-03-projekt-esy.pdf
-rw-r--r-- quade/quade     4404 2024-07-05 13:34 export/documentation/2024-07-03-projekt-esy.md
drwxrwxr-x quade/quade        0 2024-07-05 13:35 export/target/
-rw-r--r-- quade/quade       55 2024-07-05 13:35 export/target/index.html
-rw-rw-r-- quade/quade       67 2024-07-05 13:35 export/target/mosquitto.conf
-rw-rw-r-- quade/quade      110 2024-07-05 13:35 export/target/interfaces
-rw-r--r-- quade/quade      132 2024-07-05 13:35 export/target/ps.cgi
-rw-r--r-- quade/quade      154 2024-07-05 13:35 export/target/wpa_supplicant.conf
-rw-rw-r-- quade/quade      146 2024-07-05 13:35 export/target/S49mqttpub
-rw-rw-r-- quade/quade      229 2024-07-05 13:35 export/target/S99firewall
-rw-rw-r-- quade/quade      330 2024-07-05 13:35 export/target/dnsmasq.conf
-rw-r--r-- quade/quade    12530 2024-07-05 13:35 export/target/lighttpd.conf
-rw-r--r-- quade/quade     3982 2024-07-05 13:35 export/target/modules.conf
-rw-rw-r-- quade/quade       67 2024-07-05 13:35 export/target/S51httpd
-rw------- quade/quade     3148 2024-07-05 13:35 export/target/lighttpd.pem
-rw-rw-r-- quade/quade      335 2024-07-05 13:35 export/target/mosquitto-enc.conf
-rwxr-xr-x quade/quade      473 2024-07-05 13:35 export/target/mqtt.sh
-rw-rw-r-- quade/quade       47 2024-07-05 13:35 export/target/sysctl.conf
-rw-r--r-- quade/quade       27 2024-07-05 13:35 export/target/httpd.conf
-rw-rw-r-- quade/quade      107 2024-07-05 13:35 export/target/S61wpa
```

Einreichungsort und -frist

- Die Dokumentation **muss** über den Moodle-Exam-Lernraum abgegeben werden: https://moodle-exam.hsnr.de/course/view.php?id=4298
- Die TAR-Datei legen Sie auf dem als File-Drop konfigurierten Sciebo-Ordner https://hs-niederrhein.sciebo.de/s/OqBrUffr1kFAEPC ab
  ab. Das Passwort dazu lautet `20ftz24#51`.
- Bennen Sie die TAR-Datei nach folgendem Schema: `\<datum\>-esy-\<IHR NAME\>.tgz` (Beispiel: `2024-07-05-esy-quade.tgz`).
- Sie können auch mehrere Versionen ablegen; zur Benotung wird die jeweils letzte Version herangezogen.
- Der Zugriff auf den Sciebo-Ordner ist befristet bis zum 9.9.2024 - 23.59 Uhr möglich.

Bewertung

Die Arbeit gilt als **nicht bestanden**, wenn einer der nachfolgenden Punkte zutrifft:

- Die Einreichung ist **unvollständig**, es fehlen Teile.
- Die Dateien sind nicht **fristgerecht** eingereicht worden.
- Die Eigenständigkeitserklärung liegt nicht vor.
- Die Eigenständigkeitserklärung ist nicht **unterschrieben**.
- Die Projektarbeit ist mit **weniger als 50 Punkten** bewertet worden.
- Das Projekt weist nicht die geforderte **Individualisierung** auf.

Bewertungskriterien

- Dokumentation: 25 Punkte
- System/Implementierung: 75 Punkte

Notenskala:

```
00-49 = 5,0
50-54 = 4,0
55-59 = 3,7 | 60-64 = 3,3 | 65-69 = 3,0
70-74 = 2,7 | 75-79 = 2,3 | 80-84 = 2,0
85-90 = 1,7 | 90-94 = 1,3 | 95-100 = 1,0
```

Insbesondere werden für die folgende Funktionalitäten/Arbeiten Punkte verteilt:

- Booten
- Ansteuerung GPIO
- Einloggen unter spezifischem Namen
- Passwort ist gesetzt
- Vorhandensein des Gerätetreibers unter dem spezifischen Namen
- Gerätedatei mit spezifischem Namen
- Realisierung des Kernelthreads
- Globale Variable mit spezifischem Namen
- Schutz des kritischen Abschnitts
- Automatisches Anlegen der Gerätedatei
- Einstellung der Blinkfrequenz
- WLAN unter dem individualisierten Namen wird aufgespannt
- Einbuchen per Passwort möglich
- Netzadresse 192.168.123.0/24
- Hostname des Systems korrekt gesetzt
- Mosquitto is up and running
- Skript, um Topics zu setzen
- Topcis werden gesetzt
- Individualisiertes Topic wird gesetzt
- Skript zur Ansteuerung der LED
- Ansteuerung der LED ist per mqtt möglich
- Ansteuerung der LED per Smartphone ist möglich
- Packaging

Aufgabenstellung

Das von Ihnen zu entwickelnde, eingebettete System soll die folgenden Anforderungen erfüllen:

- Das zu entwickelnde System bootet per tftp von einem tftp-Server.
- Das System ist auf einem Raspberry Pi 4 lauffähig.
- Auf dem System gibt es neben dem Root-Login einen User-Login mit dem Namen `Panagou` und dem Passwort `Qbobhpv`.
- Über das System kann eine LED, die an GPIO-23 angeschlossen wird, angesteuert werden.
- Zur Ansteuerung der LED enthält das System einen Gerätetreiber. Der Gerätetreiber trägt den Namen `signalan.c`.
- Der Gerätetreiber wird über die Gerätedatei `/dev/led_onoff_an` angesprochen.
- Gerätedateien werden vom System automatisch angelegt.
- Der Gerätetreiber realisiert ein Blinklicht, dessen Frequenz durch Schreiben auf die Gerätedatei `/dev/led_onoff_an` an- und eingestellt werden kann:
  - Das Blinken wird über einen Kernel-Thread realisiert, der sich über `wait_event_interruptible_timeout()` schlafen legt.
  - Die Frequenz ist im Treiber in der globalen Variablen `freqPanagou` abgelegt.
  - Der Zugriff auf die Variable muss effizient geschützt sein.
- Das System spannt ein WLAN mit dem Namen `VimIsTheBest` auf.
- Das Passwort zum WLAN lautet `ancprcuuaq`.
- Die IP-Adressen, die per WLAN verteilt werden, stammen aus dem Netz **192.168.123.0/24**.
- Auf dem System läuft ein SSH-Server, über den man sich remote einloggen kann.
- Das System hat den Hostnamen `lilibiti80`.
- Auf dem System läuft der MQTT-Broker `mosquitto`, der beim Hochfahren automatisch gestartet wird.
- Per Skript auf dem Raspberry Pi sollen folgende Topics per Kommando `mosquitto_pub` regelmäßig gepublished werden:
  - `system/hostname`
  - `system/ipaddress/eth0`
  - `system/ipaddress/wlan0`
  - `system/uptime`
  - `system/date`
  - `system/Panagou`
  - `appl/frequency`
- Über das Topic `system/Panagou` wird Ihr Name publiziert.
- Die Ansteuerung der LED erfolgt über das Topic `appl/frequency_set`

  Hinweise, wie Sie mit Hilfe des Kommandos `mosquitto_sub` und einem Shell-Skript damit die LED ansteuern, finden Sie unten.
- Auf einem Smartphone soll ein MQTT-Client (MQTT-Dashboard) installiert und konfiguriert werden, so dass über das Smartphone die publizierten Topics visualisiert und das Blinklicht in seiner Frequenz eingestellt werden kann.
- Zum System gibt es eine Entwicklungsdokumentation, die die nachfolgenden Kriterien erfüllt.

Dokumentation

Die Dokumentation soll den Leser in die Lage versetzen, mit Hilfe der von Ihnen eingereichten Dateien das System nachzubauen. Er muss dazu beispielsweise nicht die genauen Konfigurationspunkte der Buildroot-Konfig kennen, wohl aber mitgeteilt bekommen, wie er die `config.buildroot` einspielt.

Die Dokumentation ist im **Markdown-Format** zu erstellen und sowohl im PDF-Format als auch als Quellcode (Dateierweiterung `.md`) einzureichen. Im **Moodle-Lernraum** finden Sie dazu ein **Template**, das Hinweise zum Umgang mit Markdown als auch Hinweise zu den Inhalten der Dokumentation gibt. Die Dokumentation wird grob 20 Textseiten umfassen.

Wissenswertes

MQTT

Message Queuing Telemetry Transport (MQTT) ist ein offenes Netzwerkprotokoll für Machine-to-Machine-Kommunikation (M2M), das die Übertragung von Telemetriedaten in Form von Nachrichten zwischen Geräten ermöglicht, trotz hoher Verzögerungen oder beschränkter Netzwerke.[1] Entsprechende Geräte reichen von Sensoren und Aktoren, Mobiltelefonen, Eingebetteten Systemen in Fahrzeugen oder Laptops bis zu voll entwickelten Rechnern.

Interessant ist, dass ein MQTT-Server („Broker“) die gesamte Datenlage seiner Kommunikationspartner hält, und so als Zustands-Datenbank benutzt werden kann. So ist es möglich, kleine unperformante MQTT-Geräte mit einem MQTT-Broker zu verbinden, wobei die Geräte Daten einsammeln und/oder Befehle entgegennehmen, während ein komplexes Lagebild nur auf dem MQTT-Broker entsteht und hier oder durch einen leistungsfähigen Kommunikationspartner ausgewertet werden kann. Stelleingriffe können so von einer oder mehreren leistungsfähigen Instanzen an den MQTT-Broker übermittelt und auf die einzelnen Geräte verbreitet werden. Dadurch eignet sich MQTT sehr gut für Automatisierungslösungen und findet im Bereich IoT durch die einfache Verwendung große Verbreitung.

[Quelle: https://de.wikipedia.org/wiki/MQTT]

MQTT ist ein Protokoll zwischen Server und Clients. Der Server ist der MQTT-Broker, der die Verteilstation von Nachrichten nach dem Publisher-Subscriber-Prinzip darstellt. Publisher versenden ihre Daten unter einem Topic an den Broker. Subscriber (Clients) abonnieren beim Broker Topics. Sobald für ein Topic eine neue Nachricht ankommt, verteilt der Broker diese an die Clients, die sich auf die Nachricht abonniert haben.

Die Topics sind ähnlich den Dateipfaden hierarchisch organisiert, vom Systemarchitekten aber frei wählbar.

Beispiel für Topics:

`system/core/ipadress` `system/core/hostname` `motorcontrol/motorleft/rpm`

Weitere Infos finden Sie zum Beispiel unter

[https://www.embedded-software-engineering.de/was-ist-mqtt-a-725485/]

MQTT in der Praxis

Mit `mosquitto` steht sowohl die Implementierung eines Brokers als auch von Clients zur Verfügung. `mosquitto` ist bereits in Buildroot integriert und muss nur ausgewählt werden.

Mit den Programmen `mosquitto_pub` und `mosquitto_sub` stehen Publisher- und Subscriber-Programme zur Verfügung.

Für Tablets und Smartphone gibt es diverse MQTT-Dashboards, z.B. *MQTT Dash* für Android.

Um per Smartphone auf den Broker zugreifen zu können, muss sich das Smartphone im gleichen Netz befinden. Buchen Sie also Ihr Smartphone ins WLAN des Raspberry Pi ein!

Bei den Apps ist als erstes der Broker zu kontaktieren.

Ansteuerung der LED über MQTT

Um kein eigenes C-Programm schreiben zu müssen, wird im Rahmen eines Rapid-Prototyping die LED per Shell-Skript `sw_led_an.sh` angesteuert. Das Shell-Skript liest von `STDIN` Nachrichten und schreibt diese in die Gerätedatei `/dev/led_onoff_an`.

mosquitto_sub -t led -h 192.168.123.1 | ./sw_led_an.sh

`sw_led_an.sh`:

```
#!/bin/sh

while true
do
    echo "waiting for input ..."
    read
    echo "switching led to ${REPLY}"
```

```
    echo -e -n "\x${REPLY}" >/dev/led_onoff_an
done
```

User-Verwaltung

Buildroot legt automatisiert User an, wenn eine Datei mit den benötigten Daten vorliegt und in der Buildroot-Konfiguration verlinkt ist. Die Datei `users` hat den folgenden Aufbau:

username	uid	group	gid	password	home	shell	groups	comment
quade	-1	quade	-1	=blink182	/home/quade	/bin/sh	adm	Dozent

Das Gleichheitszeichen vor dem Passwort sorgt dafür, dass Buildroot das Passwort als Klartext-Passwort interpretiert und den Hashwert bildet.

Die Datei `users`, die einen Eintrag für den User *foo* enthält, sieht damit folgendermaßen aus:

```
foo -1 bar -1 !=blabla /home/foo /bin/sh alpha,bravo Foo user
```

Weitere Informationsquellen

https://buildroot.org/downloads/manual/manual.html