Technische Informatik IV: Praktikum

Protokoll zu Aufgabe A14

Max Wisniewski, Alexander Steen

Vorbereitung

Zur Vorbereitung haben wir den Abschnitt SMTP POP3 des *WIP Development Guide* gelesen und die Datei *smtc.c* aus dem Rahmenwerk studiert. Um die GPRS-Funktionalitäten nutzen zu können, wurde in der *config.h* `INIT_GPRS` aktiviert und dann ein clean build des Projekts gemacht.

Aufgaben

1. Versenden der ersten fünf Telefonbucheinträge via E-Mail bei Tastendruck auf Taster 3

Dokumentation

wip_SMTPClientCreateOpts Diese Funktion erstellt ein neue Verbindung zu einem SMTP-Server. Die Syntax lautet:
`wip_SMTPClientCreateOpts (ascii *serverAddr, wip_eventHandler_f evHandler, void *userdata, ... );`
Dabei bedeuten die Parameter (in der Reihenfolge): Adresse (IP oder Hostname) des Mailservers; Eventhandler, der bei einem Event auf diesem Kanal aufgerufen werden soll; etwaige Userdata, die jedes Mal beim Aufruf des Handlers mitgegeben werden soll.
Anschließend kann man am Ende des Kommandos eine Reihe von Parameter-Konstanten der Form `WIP_COPT_X` angeben, wobei die Auflistung mit `WIP_COPT_END` enden muss. Üblich ist z.B. der Parameter WIP_COPT_SMTP_AUTH_TYPE, WIP_SMTP_AUTH_MIME64, der angibt, das die Verbindung durch ein Passwort gesichert ist. Die Funktion gibt einen Zeiger auf den erzeugten Kanal zurück.

wip_putFileOpts Diese Funktion versucht aus dem übergebenen Kanal eine Datei mit dem Namen file_name zu schreiben. Dabei wird bei einem Event der Handler evh aufgerufen. Die Syntax ist:
`wip_read ( wip_channel_t channel, ascii *file_name, wip_eventHandler_f evh, void *ctx, ...);`

Durchführung und Auswertung

SMTP Das Simple Mail Transfer Protocol ist ein Klartext-Protokoll zum Versenden von E-Mails und nutzt standardmäßig Port 25.

POP3 Das Post Office Protocol ist ein Protokol zum Abrufen von E-Mails vom Mailserver. Dabei werden die Nachrichten normalerweise vom Server heruntergeladen.

```
wip_channel_t sessionchannel; // Sitzung
u8 entriesRead = 0; // Gelesene Telefonbucheinträge
ascii mailbodybuffer[512]; // Puffer der Mail-Nachricht
ascii entries[5][64];  // Puffer für die Telefonbucheinträge
```

Wir speichern zunächst den Kanal zum SMTP-Server, da dieser später von einem der Callbacks genutzt werden wird. `entriesRead` zählt, wie viele Einträge bereits aus dem Telefonbuch ausgelesen wurden. Erreicht dieser Wert fünf, so haben wir alle nötigen Einträge und können die E-Mail schicken. `mailbodybuffer` und `entries` sind Zeichenpuffer für die zu schickenden Informationen.

```
void main_task(void) {
    // Keymeldungen aktivieren
    adl_atCmdCreate("AT+CMER=,1", FALSE, (adl_atRspHandler_t) NULL, NULL);
    // Keyhandler anlegen auf Key 3
    adl_atUnSoSubscribe("+CKEV: 3,1", keyhandler);

    sessionchannel = // Verbindung einrichten
        wip_SMTPClientCreateOpts("mail.o2online.de", // Host
            smtp_handler, // Handler für Vorbereitung
            NULL, // Kontext
            WIP_COPT_PEER_PORT, 25, // Port
            WIP_COPT_USER, "017665374344@o2online.de", // User
            WIP_COPT_PASSWORD, "fu1234", // Passwort
            WIP_COPT_SMTP_AUTH_TYPE, WIP_SMTP_AUTH_MIME64, WIP_COPT_END);
}
```

Wir aktivieren in der main-Funktion die Tastenmeldungen von Taste 3 und registrieren die Funktion `keyhandler` darauf. Außerdem öffnen wir einen Kanal zum SMTP-Server mit den gegebenen Parametern und dem Handler `smtp_handler`.

```
void smtp_handler(wip_event_t *event, void *context) {
    if (event->kind == WIP_CEV_ERROR) {
        // Verbindung zum SMTP-Server war nicht erfolgreich
        ERROR("Verbindung zum Mail-Server abgebroche, Neustart erforderlich\r\n");
        wip_close (sessionchannel);
        return;
    }
}
```

Dieser SMTP-Handler tut eigentlich nichts, da wir ja noch keine Mail verschicken wollen. Hier fangen wir nur Fehler (z.B. Verbindungsfehler durch falsche Passwörter o.ä.) ab. Ist dies der Fall geben wir eine entsprechende Fehlermeldung aus und schließen den Kanal.

```
bool keyhandler(adl_atUnsolicited_t *paras) {
    adl_atSendResponse(ADL_AT_RSP, "Taste 3 wurde gedrückt\r\n");
    // Feuere abfrage an Telefon ab
    if (entriesRead == 0)       {
        adl_atCmdCreate("AT+CPBR=1,5", FALSE, phonebook_handler, "+CPBR: ", NULL);
    }
    return FALSE;
}
```

Auch der `keyhandler` ist nicht weiter kompliziert: Wird die Taste gedrückt, so feuern wir eine Abfrage an das Telefonbuch ab, uns die ersten fünf Einträge zu liefern. Darauf registrieren wir den `phonebook_handler`, der die weitere Verarbeitung übernimmt. All dies wird nur ausgelöst, falls der `entriesRead`-Wert Null ist, da wir uns sonst noch mitten in einer Maildaten-Abrufprozedur befinden.

```
bool phonebook_handler(adl_atResponse_t *paras) {
    ascii nameBuffer[16];
    ascii telNrBuffer[16];
    // Hole Infos aus dem Telefonbucheintrag
    wm_strGetParameterString(telNrBuffer, paras->StrData, 2); // Nummer
    wm_strGetParameterString(nameBuffer, paras->StrData, 4);  // Name

    wm_sprintf(entries[entriesRead],
            "Name: %s, Nummer: %s \r\n", nameBuffer, telNrBuffer);
    ++entriesRead;
    if (entriesRead == 5) {
        // Alle Einträge gelesen, Mail schicken
        wip_putFileOpts(sessionchannel, // Session Channel
            NULL, // Filename
            smtp_send_handler, // Handler für Versand
            NULL, // Kontext
            WIP_COPT_SMTP_SENDERNAME, "HWP8", // Sendername HWP8
            WIP_COPT_SMTP_SENDER, "017665374344@o2online.de", // Sender eMailadresse
            WIP_COPT_SMTP_REC, "017665374344@o2online.de", // Empfänger eMailadresse
            WIP_COPT_SMTP_SUBJ, "E-Mail-Test von HWP8", // Betreff
            WIP_COPT_END);
        entriesRead = 0;
    }
    return FALSE;
}
```

Dieser Handler wird für jede der fünf Antworten der Telefonbuch-Abfrage angesprungen. Dabei lesen wir zunächst die Eintragsdaten für den Namen und die Nummer aus und speichern sie in lokale Puffer. Dann bauen wir uns einen kleinen String der Form "*Name: <Name>, Nummer: <Nummer>*" zusammen und speichern diesen in dem `entries`-Feld. Hierbei benutzen wir einfach die Anzahl der bereits gelesenen Datensätze als Index, den wir nach dem Schreiben inkrementieren. Hat der Zähler nun fünf erreicht, haben wir gerade den letzten Eintrag ausgelesen und veranlassen, dass eine Mail geschickt wird. Hier tragen wir den Absender, den Empfanger, den Betreff und den Callback-Handler `smtp_send_handler` an. Dieser baut für uns dann den Mailkörper zusammen.

```
void smtp_send_handler(wip_event_t *event, void *context)   {
    switch (event->kind) {
        case WIP_CEV_OPEN:
            // Wenn Kanal offen: Mail in den Kanal schreiben
            wm_sprintf(mailbodybuffer,
                "Hallo Welt!\r\n Die Einträge aus dem Telefonbuch sind:
                \r\n\r\n%s %s %s %s %s Tschüss!\r\n",
                entries[0], entries[1], entries[2], entries[3], entries[4]);
            adl_atSendResponse(ADL_AT_RSP, mailbodybuffer);
            wip_write(event->channel, mailbodybuffer, wm_strlen(mailbodybuffer));
            adl_atSendResponse(ADL_AT_RSP, "Mail gesendet!\r\n");
            break;
        default: /* Fehler oder fertig */
            wip_close (event->channel); // schließt die Übertragung
            break;
    }
}
```

Ist der Sendekanal offen, so können wir unsere Mail schreiben. Dazu bauen wir uns einen kleinen E-Mail-Text zusammen und schicken den Text via `wip_write` auf den Kanal.

Testlauf:

```
+GSM: Anmeldung im Netz abgeschlossen
+GPRS: Warte auf Verbindung
+GPRS: Verbindung wurde hergestellt
Taste 3 wurde gedrückt
Hallo Welt!
Die Einträge aus dem Telefonbuch sind:
Name: Test, Nummer: 200
Name: 129, Nummer: 017665316700
Name: 129, Nummer: 017665338235
Name: 129, Nummer: 017665348409
Name: 129, Nummer: 017683269836
Tschüss!
Mail gesendet!
```

Leider haben wir bei diesem Testlauf aus Versehen nicht den Namen, sondern den Nummertyp ausgelesen. Da es aber so lange dauerte, bis wir die GPRS-Verbindung hatten und das Programm an sich ja funktioniert, wollten wir keinen neuen Testlauf starten.

Wir haben lediglich in dem `phonebook_handler` den Index von `wm_strGetParameterString` für den Namen um Eins erhöhen müssen, damit der Name korrekt benutzt wird.

Hier haben wir auch einen kleinen Screenshot von der Mail im O2-Postfach: