

# Protokoll 3

## Praktikum Technische Informatik

Tas Soti, Richard Wilhelm, Naja v. Schmude

9. Juni 2008

### 1 Die akustische Raumüberwachung

#### 1.1 Vorbereitung

Da man die SMS- und Anrufsteuerungsfunktionalität benötigt, werden zuerst die entsprechenden Kapitel in der Dokumentation durchgearbeitet.

Zuerst beschäftigen wir uns mit dem Empfangen und lesen von SMS. Wie auch in den vorangehenden Aufgaben muss erstmal die SMS-Funktionalität registriert werden. Dies geschieht durch die Funktion `adl_smsSubscribe(adl_smsHdlr_f SmsHandler, adl_smsCtrlHdlr_f SmsCtrlHandler, u8 Mode)`. Die ersten beiden Parameter entsprechen wieder Handlerfunktionen, die bei dem Empfangen (erster Parameter) bzw. Versenden von SMS (zweiter Parameter) aufgerufen werden. Im dritten Parameter wird definiert, ob eine eingehende SMS im Text-Modus (`ADL_SMS_MODE_TEXT`) oder im PDU-Modus (`ADL_SMS_MODE_PDU`) behandelt werden soll.

Jetzt kommen wir zur Anrufssteuerung. Wir wollen einen Anruf absetzen, also benötigen wir die Funktion `adl_callSetup (ascii *PhoneNb, u8 Mode)`. Hier wird als erster Parameter die zu wählende Nummer erwartet und als zweites der Modus, d.h. ob es sich um einen Voice-Call (`ADL_CALL_MODE_VOICE`) oder um einen Data-Call (`ADL_CALL_MODE_DATA`) handeln soll.

#### 1.2 Aufgaben und Durchführung

- *Über eine SMS mit dem Inhalt "call", soll der Absender angerufen werden. Diese SMS soll nicht gespeichert werden.*

---

```
void a08_apply(void)
{
    adl_smsSubscribe(a08smsHandler, a08smsControlHandler,
        ADL_SMS_MODE_TEXT);
}
```

```
/* SMS Handler, der bei einer einkommenden SMS aufgerufen
wird.*/
bool a08smsHandler(ascii* smsTel, ascii* smsTimeLength,
    ascii* smsText) {
    if(wm_strcmp(smsText,"call") == 0) { // Der Text ist "
        call" -> Ruf also die Absendernummer an und verwirf
        die SMS
        adl_callSetup(smsTel,ADL_CALL_MODE_VOICE);
        return FALSE;
    }
    return TRUE;
}

/* SMS Handler, der eigentlich für ausgehende SMS ist,
wird hier aber nicht gebraucht. */
void a08smsControlHandler(u8 Event, u16 Nb) {
}
```

---

Als erstes wird also wie beschrieben die SMS-Funktionalität registriert, wobei der `a08smsHandler` die wirklich wichtige Funktionalität enthält, der `a08smsControlHandler` wird nicht benötigt und erscheint daher nur als Funktionsrumpf ohne Inhalt. Es wird sich auch im Text-Modus registriert, so dass die SMS dann einfach bearbeitet werden kann.

Der SMS-Handler erwartet drei Parameter, die Absendernummer, den Zeitstempel der SMS und natürlich die Nachricht selbst, alle Parameter werden automatisch bei eingehenden SMS übergeben. Zusätzlich hat die Funktion noch einen booleschen Rückgabewert, der `TRUE` ist, wenn die SMS gespeichert werden soll und andernfalls `FALSE`.

Wenn eine SMS ankommt, wird zuerst mit der Funktion `wm_strcmp()` verglichen, ob der Text der SMS auch "call" ist. Wenn ja, dann wird die in `smsTel` übergebende Nummer angerufen über `adl_callSetup()` und zwar wie beschrieben im Voice-Modus. Zusätzlich wird dann hier `FALSE` zurückgegeben, denn diese SMS soll ja nicht gespeichert werden.

Wenn der Inhalt der Nachricht jedoch ein anderer ist, so wird einfach nur `TRUE` zum speichern der SMS zurückgeliefert.

## 2 Eine einfache Alarmanlage

### 2.1 Vorbereitung

Als erstes muss sich natürlich damit befasst werden, wie man überhaupt einen Tastendruck registriert und ihn behandelt. Um überhaupt Tasten-Ereignisse angezeigt zu bekommen, gibt es den Befehl `AT+CMER`. Dieser Befehl sorgt dafür, dass bei jedem Tastendruck bzw. Loslassen einer Taste eine spezielle Nachricht geworfen wird, die auch auf der Konsole geschrieben wird. Durch diese Nachrichten kann dann die gedrückte Taste ermittelt werden und auch, ob

es sich um das Drücken oder das Loslassen handelt.

Es soll ja nun immer der aktuelle Zeitstempel ausgelesen werden, daher benötigen wir auch hierfür einen passenden Befehl; dies leistet das Kommando AT+CCLK. Durch AT+CCLK? wird uns stets der aktuelle Zeitstempel zurückgeliefert.

Um nun ein AT-Kommando im Quellcode abzusetzen, gibt es die Funktion `adl_atCmdCreate`. Sie erwartet natürlich das Kommando als Parameter und noch einige mehr. Z.B. kann man optional einen Handler einrichten, der auf bestimmte Antworten, die der Kommandoaufruf bewirkt, reagiert wird. Zusätzlich kann dann noch definiert werden, ob nicht registrierte Antworten an die Applikation weitergeleitet werden sollen oder nicht. Als letzten Parameter wird übrigens immer NULL erwartet.

## 2.2 Aufgaben und Durchführung

- Beim Drücken und Loslassen der Taste\_0 soll ein Alarmtext mit aktuellem Zeitstempel über die serielle Schnittstelle ausgegeben werden.

---

```
void a09_apply(void)
{
    adl_atCmdCreate("AT+CMER=,1", FALSE, (adl_atRspHandler_t)
        )NULL, NULL); // Aktivierung der Schalter
    adl_atUnSoSubscribe("+CKEV:0,1", (adl_atUnSoHandler_t)
        tasteGedrueckt_Handler); // Registrierung des
        Drückens von Taste 0
    adl_atUnSoSubscribe("+CKEV:0,0", (adl_atUnSoHandler_t)
        tasteLosgelassen_Handler); // Registrierung des
        Loslassens von Taste 0
}

/* Handler, der aufgerufen wird, wenn die Taste gedrückt
   wird. */
bool tasteGedrueckt_Handler(adl_atUnsolicited_t *paras)
{
    a09gedrueckt = TRUE;
    adl_atCmdCreate("AT+CCLK?", FALSE, (adl_atRspHandler_t)
        ) zeitAuslesen_Handler, "+CCLK" , NULL); //
        Auslösen des Zeitauslesens
    return FALSE;
}

/* Handler, der aufgerufen wird, wenn die Taste
   losgelassen wird. */
bool tasteLosgelassen_Handler(adl_atUnsolicited_t *paras)
{
    a09losgelassen= TRUE;
    adl_atCmdCreate("AT+CCLK?", FALSE, (adl_atRspHandler_t)
        ) zeitAuslesen_Handler, "+CCLK" , NULL); //
```

```

        Auslösen des Zeitauslesers
    return FALSE;
}

/* Handler, der aufgerufen wird, wenn das Datum ausgelesen
   werden soll. Da dies als letztes geschieht, passiert
   hier auch die ganze andere Arbeit. */
bool zeitAuslesen_Handler(adl_atUnsolicited_t *parameter)
{
    wm_strGetParameterString(a09datum, parameter->StrData,
        1); // Datum einlesen und speichern
    if(a09gedrueckt) { // Taste gedrückt
        wm_sprintf(a09message, "\r\nKontakt geschlossen um %s\r\n", a09datum);
        adl_atSendResponse(ADL_AT_RSP, a09message);
        a09gedrueckt = FALSE;
    }
    if(a09losgelassen) { // Taste losgelassen
        wm_sprintf(a09message, "\r\nKontakt geöffnet um %s\r\n", a09datum);
        adl_atSendResponse(ADL_AT_RSP, a09message);
        a09losgelassen = FALSE;
    }
    return FALSE;
}

```

---

Als erstes wird erstmal durch wie oben beschrieben durch `adl_atCmdCreate("AT+CMER=,1", FALSE, (adl_atRspHandler_t) NULL, NULL)`; die Registrierung der Tasten durchgeführt. Dabei wollen wir einfach nur den Befehl absetzen und auf keine Antworten, die durch den Befehl ausgelöst werden, reagieren, deshalb ist kein Handler an dritter Stelle definiert und auch die Variable zum Weiterleiten der unregistrierten Antworten auf FALSE gesetzt.

Wir bauen hier keinen Antwort-Handler ein, da wir bereits genau die Antworten kennen, auf die reagiert werden sollen, darum können wir auch direkt über die Funktion `adl_atUnSoSubscribe` auf Nachrichten reagieren. Diese Methode erwartet den Antwortstring, auf den reagiert werden soll, und einen Handler, der dann bei Auftreten der Antwort durchgeführt wird. Wir wollen natürlich auf das Drücken und Loslassen des Schalters Nummer 0 reagieren. Und zwar wird beim Drücken des Schalters +CKEV: 0,1 ausgegeben und beim Loslassen +CKEV: 0,0, was dann als Parameter übergeben wird. Das Drücken wird dann in der Funktion `tasteGedrueckt_Handler` aufgerufen, in der dann der Befehl `AT+CCLK?` zum Auslesen der Uhrzeit ausgelöst wird. Die `tasteLosgelassen_Handler` funktioniert analog.

In der Funktion zum Zeitauslesen wollen wir diesmal auf die produzierte Antwort (also die Uhrzeit) reagieren, und definieren einen `zeitAuslesen_Handler`, der bei Befehlen mit dem Substring `+CCLK` reagieren soll.

Mit der bereits bekannten Funktion `wm_strGetParameterString` wird dann der Parameter der Datumsrückgabe ausgelesen und gespeichert. Und je nach gesetzter booleschen Variable wird dann entweder Kontakt geschlossen oder Kontakt geöffnet ausgegeben.

Die Ausgabe sieht bis hierhin also wie folgt aus:

---

```
// Modul-Initialisierung: Gerät funktionsbereit!
```

```
Kontakt geschlossen um 08/05/26,17:15:30
```

```
Kontakt geöffnet um 08/05/26,17:16:42
```

```
Kontakt geschlossen um 08/05/26,17:16:50
```

```
Kontakt geöffnet um 08/05/26,17:16:51
```

---

- *Es soll zusätzlich eine SMS mit dem Alarmtext beim Schließen des Kontakts versendet werden. Frühestens nach 60 Sekunden soll erneut eine SMS verschickt werden.*

---

```
void a09_apply(void)
```

```
{
```

```
...
```

```
    a09smsValue = adl_smsSubscribe(a09smsHandler,
        a09smsControlHandler, ADL_SMS_MODE_TEXT); // SMS
        wird registriert
```

```
}
```

```
/* Handler, der aufgerufen wird, wenn die Taste gedrückt
wird. */
```

```
bool tasteGedrueckt_Handler(adl_atUnsolicited_t *paras)
```

```
{
```

```
    a09gedrueckt = TRUE;
```

```
    adl_atCmdCreate("AT+CCLK?", FALSE, (adl_atRspHandler_t)
        zeitAuslesen_Handler, "+CCLK", NULL); //
```

```
        Auslösen des Zeitauslesens
```

```
    a09timer = (adl_tmr_t *) adl_tmrSubscribe(TRUE, 600,
        ADL_TMR_TYPE_100MS, (adl_tmrHandler_t)
```

```
        a09Timer_Handler); // 60-Sekunden Timer für SMS
```

```
        Sperre wird gestartet
```

```
    return FALSE;
```

```
}
```

```
/* Handler, der aufgerufen wird, wenn das Datum ausgelesen
werden soll. Da dies als letztes geschieht, passiert
hier auch die ganze andere Arbeit. */
```

```
bool zeitAuslesen_Handler(adl_atUnsolicited_t *parameter)
```

```
{
```

```
    wm_strGetParameterString(a09datum, parameter->StrData,
        1); // Datum einlesen und speichern
```

```

        if(a09gedrueckt) { // Taste gedrückt
            wm_sprintf(a09message, "\r\nKontakt geschlossen um %s\r\n", a09datum);
            adl_atSendResponse(ADL_AT_RSP, a09message);
            if(a09smsErlaubt) { // SMS darf geschickt werden
                adl_atSendResponse(ADL_AT_RSP, "SMS soll verschickt
                    werden!");
                adl_smsSend(a09smsValue, "+4915159075909",
                    a09message, ADL_SMS_MODE_TEXT);
                a09smsErlaubt = FALSE;
            }
            else
                adl_atSendResponse(ADL_AT_RSP, "Timer für SMS noch
                    nicht abgelaufen");
            a09gedrueckt = FALSE;
        }
        ...
    }

    /* SMS Timer-Handler, meldet nach Ablauf der 60s den Timer
        ab und erlaubt das SMS schicken wieder */
    void a09Timer_Handler (u8 Id) {
        adl_tmrUnSubscribe(a09timer, a09Timer_Handler,
            ADL_TMR_TYPE_100MS);
        a09smsErlaubt = TRUE;
    }

    bool a09smsHandler(ascii* smsTel, ascii* smsTimeLength,
        ascii* smsText) {
        return TRUE;
    }

    void a09smsControlHandler(u8 Event, u16 Nb) {
        if(Event == ADL_SMS_EVENT_SENDING_OK)
            adl_atSendResponse(ADL_AT_RSP, "\r\nSMS erfolgreich
                verschickt.");
    }

```

---

Wir erweitern jetzt unsere a09\_apply-Methode um das Registrieren der SMS-Funktionalität. Diesmal bleibt der a09smsHandler, der beim Empfangen der SMS wichtig ist, leer, aber der a09smsControlHandler wird nur für Logging-Zwecke benutzt, um das erfolgreiche Verschicken angezeigt zu bekommen. In der tasteGedrueckt\_Handler-Funktion wird nun zusätzlich noch der Timer gestartet, der nach Ablauf der 60 Sekunden wieder das Abschicken der SMS erlaubt, in dem a09smsErlaubt wieder auf TRUE gesetzt wird (Initialwert ist auch TRUE). Als letztes muss in dem Handler der Zeitauslesefunktion noch beim Taste-Gedrückt-Zweig eine weitere Abfrage eingebaut

werden, ob SMS aktuell verschickt werden dürfen. Wenn ja, wird mit der altbekannten Methode die SMS dann verschickt, ansonsten einfach nur eine Meldung ausgegeben, dass es aktuell nicht erlaubt ist.

Ein Testlauf sieht dann wie folgt aus:

---

```
// Modul-Initialisierung: Gerät funktionsbereit!

Kontakt geschlossen um 08/05/26,17:22:21
SMS soll verschickt werden!SMS errorSMS erfolgreich
  verschickt.
Kontakt geöffnet um 08/05/26,17:22:24

Kontakt geschlossen um 08/05/26,17:22:54
Timer für SMS noch nicht abgelaufen
Kontakt geöffnet um 08/05/26,17:22:54

Kontakt geschlossen um 08/05/26,17:23:00
Timer für SMS noch nicht abgelaufen
Kontakt geöffnet um 08/05/26,17:23:00

Kontakt geschlossen um 08/05/26,17:23:01
Timer für SMS noch nicht abgelaufen
Kontakt geöffnet um 08/05/26,17:23:02

Kontakt geschlossen um 08/05/26,17:25:17
SMS soll verschickt werden!SMS errorSMS erfolgreich
  verschickt.
Kontakt geöffnet um 08/05/26,17:25:20

Kontakt geschlossen um 08/05/26,17:25:41
Timer für SMS noch nicht abgelaufen
Kontakt geöffnet um 08/05/26,17:25:41
```

---