Application Note

PiXtend per SMS steuern und auslesen

APP-PX-210

Stand 02.06.2016, V1.03

Qube Solutions UG (haftungsbeschränkt)
Arbachtalstr. 6, 72800 Eningen, Germany
http://www.qube-solutions.de/
http://www.pixtend.de

Versionshistorie

Version	Beschreibung	Bearbeiter
1.00	Dokument erstellt	PT
1.01	Dokument überarbeitet Titelbild eingefügt	TG
1.02	Dokument überarbeitet	PT
1.01	Befehl in Kapitel 3 korrigiert	TG

Inhaltsverzeichnis

1. Einleitung

In dieser Application Note beschäftigen wir uns mit der Möglichkeit eine **Kommunikation mit Raspberry Pi & PiXtend über das Mobilfunknetz (SMS)** herzustellen. So können Sie Ihr PiXtend-System auch in Gegenden erreichen, in denen kein ausreichend schnelles Mobiles Internet bzw. W-LAN zur Verfügung steht.

Das Mobilfunknetz in Deutschland hat eine nahezu lückenlose Abdeckung und selbst bei schlechtem Empfang kann doch meist eine SMS empfangen oder gesendet werden. Bei der heutigen Flut an E-Mails und Nachrichten sticht die gute alte SMS noch immer heraus, wenn es um kurze Texte geht.

Typische Anwendungsbeispiele:

- Messstationen (z.B. im Gebirge oder im ländlichen Gebiet)
- Statusmeldung/Warnung eines Sicherheitssystems, Alarmanlagen (stiller Alarm)
- Überwachung von Bienenstöcken (für Imker) oder Gewächshäusern
- Steuerung und Überwachung an abgelegenen Orten (Schrebergarten, Solaranlage...)

Bei den genannten Anwendungen ist es in der Regel ausreichend einzelne Aktion auszuführen oder Messergebnisse abzufragen. Eine Web-Visualisierung, wie wir sie oft mit CODESYS realisieren, ist nicht immer notwendig und ohne eine flotte Internet-Anbindung keine Freude.

Im Laufe dieser App-Note zeigen wir Ihnen alle nötigen Schritte der Installation und Konfiguration bis hin zum Schalten eines Relais und auslesen von Sensoren an PiXtend.

Alles bequem von Ihrem Handy aus und über SMS-Nachrichten („Short Message Service“) versendet!

Viele weitere Informationen, Tipps und Tricks zu PiXtend finden Sie auch in unserem Support-Forum unter: http://www.pixtend.de/forum/

Sollten trotzdem Fragen offenbleiben, so bitten wir Sie zuerst in den FAQ's im Anhang nachzuschauen. Wenn die Frage dort nicht geklärt wird können Sie uns gerne per E-Mail (support@pixtend.de) in Kenntnis setzen. Sie erhalten schnellst möglich eine Antwort und weitere Informationen.

1.1 Voraussetzungen

Diese Anleitung setzt voraus, dass Sie die Linux Tools ***pxdev*** (PiXtend Linux Library) bereits installiert haben. Falls Sie dies noch nicht getan haben, schauen Sie auf der Homepage im Download-Bereich nach der App-Note ***pxdev***.

Wir empfehlen Ihnen auf unserem vorbereiteten „PiXtend Image Linux Tools“ aufzusetzen, welches Sie entweder kostenlos auf unserer Homepage herunterladen oder als vorinstallierte SD-Karte im Qube Solutions Online-Shop bestellen können. Hier sind pxdev und der gcc-Compiler bereits installiert.

Wir verwenden in dieser Anleitung den **UMTS-USB-Stick *E3131* von der Firma Huawei**. Diesen gibt es für rund 30 € bei unterschiedlichen Händlern. Wir haben den Stick direkt an einen der vier USB-Ports des Raspberry Pi Modell 2 B angeschlossen und ein PiXtend V1.3 Board verwendet. Sollten Sie ein PiXtend-System der Version V1.2 einsetzen, so empfehlen wir den Anschluss des Sticks über einen externen USB-Hub mit eigenem Netzteil (powered hub) – die Sticks benötigen beim Senden relativ hohe Ströme!

Die Konfiguration eines solchen UMTS-Sticks läuft immer ähnlich ab. Wir beschreiben hier die Schritte für den E3131. Sollten Sie einen anderen Stick verwenden, so finden Sie hierzu eine Vielzahl an Anleitungen und Tipps in Raspberry Pi Foren.
Allgemein gilt jedoch: Besser ein älteres Modell eines UMTS-Sticks verwenden, da sonst ggf. noch keine Linux-Treiber und Infos vorhanden sind.

Noch einen Tipp, bevor wir beginnen:

Für die Schritte in dieser App-Note müssen Sie viele Befehle eingeben und Quelltexte kopieren. Wenn Sie von Ihrem Desktop-Rechner bzw. Notebook per SSH auf den Raspberry Pi zugreifen, so können Sie die Befehle und Texte einfach aus diesem Dokument kopieren. Das spart Zeit und erspart Ihnen die Suche nach Tippfehlern.

1.2 Haftungsausschluss

Die Qube Solutions UG kann nicht für etwaige Schäden verantwortlich gemacht werden die unter Umständen durch die Verwendung der zur Verfügung gestellten Software, Hardware, oder der hier beschriebenen Schritte entstehen können.

1.3 Warnhinweis

PiXtend und das Raspberry Pi sind nicht für den Einsatz im rauen industriellen Umfeld konzipiert! PiXtend darf nicht in sicherheitskritschen Systemen eingesetzt werden.

2. Einrichtung des E3131 UMTS Sticks

Zuerst sollten die Packages für den Raspberry Pi aktualisiert werden:

```
pi@raspberrypi ~ $ sudo apt-get update
```

Danach wird das Modeswitch-Package installiert:

```
pi@raspberrypi ~ $ sudo apt-get install usb-modeswitch
```

Dieses ermöglicht uns später den UMTS-Stick in den richtigen Modus zu versetzen. Denn unser E3131-Stick kann beispielsweise auch als Massenspeicher (microSD-Karte) eingesetzt werden – wir wollen Ihn aber als UMTS-Modem verwenden.

Modeswitch so einstellen, dass keine automatische Mode-Umschaltung ausgeführt wird:

```
pi@raspberrypi ~ $ cd /etc
pi@raspberrypi ~ $ sudo nano usb_modeswitch.conf
```

Disable Switching von 0 auf 1 setzen.

Das automatische Umschalten hat für den E3131-Stick leider nicht funktioniert. Wir setzen die Einstellung im Folgenden manuell. Wenn Sie einen anderen Stick verwenden, ist es sinnvoll einmal zu beobachten, was die automatische Mode-Umschaltung bewirkt.

Konfiguration abspeichern und den Raspberry Pi neu starten:

```
pi@raspberrypi ~ $ sudo reboot
```

Überprüfen ob der UMTS-Stick erkannt wurde:

```
pi@raspberrypi ~ $ lsusb
```

Auf der Konsole sollte nun folgende Zeile zu sehen sein.

```
Bus 001 Device 006: ID 12d1:1f01 Huawei Technologies Co., Ltd.
```

Der USB-Stick ist noch im falschen Modus.

Falls der Stick nicht erkannt wurde, USB erneut triggern und anschließend erneut auslesen:

```
pi@raspberrypi ~ $ sudo udevadm trigger
pi@raspberrypi ~ $ lsusb
```

Nun setzen wir den Huawei E3131 Stick in den richtigen Modus:

```
pi@raspberrypi ~ $ sudo usb_modeswitch -v 12d1 -p 1f01 -M
'5553424312345678000000000000001106200000100000000000000000000'
```

Die Zahlenkombination haben wir für unseren Stick im Internet gefunden. Diese müsste bei einem anderen USB-Stick angepasst werden.

Stick als Linux-Gerät verfügbar machen:

```
pi@raspberrypi ~ $ sudo modprobe option
pi@raspberrypi ~ $ echo "12d1 1001" | sudo tee /sys/bus/usb-
serial/drivers/option1/new_id
pi@raspberrypi ~ $ lsusb
```

In der Ausgabe sollte der Huawei Stick wie folgt erscheinen

```
Bus 001 Device 010: ID 12d1:1001 Huawei Technologies Co., Ltd. E169/E620/E800
HSDPA Modem
```

Die Bus/Device Nummer kann abweichen, die ID (12d1:**1001**) muss stimmen. Die *1001* ist beim Huawei E3131 Stick die ID für das UMTS-Modem. Die *12d1* steht für den Hersteller des USB-Sticks.

Die LED am Stick wechselt die Farbe (blinked) ***von grün auf blau*** (UMTS-Modus) gewechselt haben.

Damit der Stick beim Einstecken und nach dem Hochfahren richtig konfiguriert ist, legt man am Besten ein Bash-Script an, welches den UMTS Stick in den richtigen Modus versetzt. Dieses wird dann mit Hilfe eines Cronjobs z.B. einmal pro Minute aufgerufen werden. Genau das tun wir mit den folgenden Schritten:

Neues Verzeichnis erstellen:

```
pi@raspberrypi ~ $ sudo mkdir /home/pi/gammu
```

Danach legen wir in diesem Verzeichnis ein neues Bash-Script an:

```
pi@raspberrypi ~ $ cd /home/pi/gammu
pi@raspberrypi ~ $ sudo nano modeswitch.sh
```

Dem gerade angelegten Script geben wir folgenden Inhalt:

```
sudo usb_modeswitch -v 12d1 -p 1f01 -M
'5553424312345678000000000000001106200000100000000000000000000'
```

Danach speichern wir das Script ab und machen es ausführbar:

```
pi@raspberrypi ~ $ sudo chmod +x /home/pi/gammu/modeswitch.sh
```

Für dieses Script legen wir nun einen neuen Cronjob an

```
pi@raspberrypi ~ $ sudo nano /etc/crontab
```

Hier legen wir folgende Zeile ab:

```
*/1 * * * * root cd /home/pi/gammu && ./modeswitch.sh > /dev/null
```

Wir speichern die Datei und der Cronjob wird installiert. Das Script wird nun jede Minute aufgerufen. Wenn Sie den UMTS-Stick nun abziehen und wieder einstecken bzw. einen Neustart des Linux-Systems machen, so wird immer spätestens nach einer Minute die korrekte Konfiguration geladen. Sie können die Zeit im Cronjob natürlich nach Ihren Bedürfnissen anpassen.

Jetzt müssen wir uns um die Konfiguration des UMTS-Sticks keine Gedanken mehr machen.

Glückwunsch, der UMTS Stick ist eingerichtet.

Weiter geht es mit der Konfiguration des SMS-Tools ***gammu***.

3. Das SMS-Tool *gammu* installieren und einrichten

Im Folgenden werden die Schritte erklärt, um das SMS-Tool g*ammu* zu verwenden. Dieses ist später für das Senden und Empfangen der Nachrichten verantwortlich.

Das Gammu Package installieren und konfigurieren (SMS Tools):

```
pi@raspberrypi ~ $ sudo apt-get install gammu
pi@raspberrypi ~ $ sudo apt-get install python-gammu
pi@raspberrypi ~ $ sudo adduser pi dialout
```

Die *gammu* Konfigurationsdatei wird durch folgenden Befehl ausgegeben:

```
pi@raspberrypi ~ $ gammu-detect
```

Es werden nun Zeilen ausgegeben, die eine Standard-Konfigurationsdatei darstellen.

Die ausgegebenen Zeilen kopieren wir und wechseln wieder ins Home-Verzeichnis:

```
pi@raspberrypi ~ $ cd /home/pi
pi@raspberrypi ~ $ sudo nano .gammurc
```

Nun die kopierten Zeilen einfügen und abspeichern.

Einzelne gammu-Befehle können jetzt erteilt und Infos abgerufen werden:

```
pi@raspberrypi ~ $ gammu identify
```

liefert Informationen zum UMTS-Stick.

```
pi@raspberrypi ~ $ gammu getsecuritystatus
```

liefert den aktuellen Status - ist die Ausgabe *Nothing to enter*, braucht man keine PIN mehr eingeben.

Ansonsten sollte folgender Befehl und die PIN eingegeben werden, um die SIM-Karte freizuschalten:

```
pi@raspberrypi ~ $ gammu entersecuritystatus PIN <Hier die PIN eingeben>
```

Nun kann die erste Test-SMS versendet werden:

```
pi@raspberrypi ~ $ gammu sendsms TEXT <Rufnummer> -text Hallo Welt, Mit freundlichen Grüßen, dein PiXtend
```

Die Rufnummer kann mit der Vorwahl für das jeweilige Land (Deutschland +49...) angegeben werden. Die Zeichen < und > sind zu entfernen. Außerdem ist darauf zu achten, dass der SMS Text in Hochkomma zu schreiben ist.

Empfangene SMS können mit folgendem Befehl ausgelesen werden:

```
pi@raspberrypi ~ $ gammu getallsms
```

Der Live Monitor zeigt aktuelle Aktivitäten auf dem Stick an:

```
pi@raspberrypi ~ $ gammu monitor
```

Mit STRG → C wird der Monitor wieder beendet.

Wir wollen ja aber nicht jeden Befehl „von Hand“ aufrufen, weswegen wir im nächsten Kapitel den *gammu-smsd* Dienst einrichten.

4. Den SMS-Daemon „gammu-smsd“ installieren & konfigurieren

Um mit den empfangenen Nachrichten unser PiXtend steuern zu können, muss zunächst einmal der Text der Nachricht ermittelt werden. Der SMS-Daemon „gammu-smsd“ ist für die Verwaltung der Textnachrichten gut geeignet und bringt einige Features mit, die wir später in unserem C-Programm (mit *pxdev*) benötigen.

Zunächst installieren wird den SMS-Daemon:

```
pi@raspberrypi ~ $ sudo apt-get install gammu-smsd
```

Nun wechseln wir in die „gammu-smsd“ Konfigurationsdatei:

```
pi@raspberrypi ~ $ sudo nano /etc/gammu-smsdrc
```

Danach sollte die Konfigurationsdatei des SMS-Daemons, wie im folgenden Angegeben, abgeändert werden.

Sie können die Zeilen aus diesem Dokument kopieren. Der *Port* und die *PIN-Nummer* müssen noch angepasst werden.

```
#gammu-smsd Konfigurationsdatei
[gammu]
port = /dev/ttyUSB0
name = HUAWEI UMTS USB-Stick
connection = at
logformat = textalldate
use_locking = yes

# SMSD configuration, see gammu-smsdrc(5)
[smsd]
pin = XXXX
service = files
logfile = /var/log/gammu-smsd
debuglevel = 0

# Paths where messages are stored
inboxpath = /var/spool/gammu/inbox/

outboxpath = /var/spool/gammu/outbox/
sentsmspath = /var/spool/gammu/sent/
errorsmspath = /var/spool/gammu/error/

#process incoming messages
RunOnReceive = /home/pi/smsin.sh
```

Es gibt je einen Ordner für den Posteingang (*inboxpath*), Postausgang (*outboxpath*), Gesendet (*sentsmspath*) und für SMS, die nicht gesendet werden konnten (*errorsmspath*).

Der Befehl *RunOnReceive* dient dazu, um bei eingehenden Nachrichten ein Bash-Script auszuführen. Neben diesem Befehl gibt es auch noch weitere Befehle wie *RunOnSent* und *RunOnError*. Im Folgenden arbeiten wir aber ausschließlich mit der Direktive *RunOnReceive*.

Nun geben wir den benötigten Ordnern alle Rechte, damit es zu keinen Zugriffsproblemen von *gammu* kommt:

```
pi@raspberrypi ~ $ sudo chmod -R 0777 /var/spool/gammu/
```

Anschließend starten wir den SMS-Daemon neu:

```
pi@raspberrypi ~ $ sudo /etc/init.d/gammu-smsd restart
```

Mit dem Daemon-Tool können wir nun wieder eine Test-SMS versenden:

```
pi@raspberrypi ~ $ echo <Text> | sudo gammu-smsd-inject TEXT <Gewünschte Rufnummer>
```

Der Daemon besitzt ebenfalls einen eigenen SMS-Monitor:

```
pi@raspberrypi ~ $ sudo gammu-smsd-monitor
```

Dieser zeigt gesendete, empfangene und fehlgeschlagene SMS an. Daneben gibt er auch Auskunft über das aktuelle Signal.

Der Gammu SMS-Daemon ist nun eingerichtet.

Weiter geht es im nächsten Kapitel mit dem Anlegen des Bash-Scripts.

5. Bash-File für die *RunOnReceive* Direktive anlegen

Damit wir später bei einem SMS-Empfang auch etwas steuern können, müssen wir erst einmal wissen, was für ein Befehl in unserer SMS steht. Das C-Programm, dass nachher die I/Os auf PiXtend steuert, soll ebenfalls bei jeder empfangenen SMS aufgerufen werden.

Dazu legen wir nun ein Bash-Script an, dass bei jeder empfangenen SMS ausgeführt wird.

Am Besten erstellen wir das Script im Home-Verzeichnis:

```
pi@raspberrypi ~ $ sudo nano /home/pi/smsin.sh
```

Der Name der Bash-Datei muss dem eingegeben Namen in der Gammu-smsd-Konfigurationsdatei entsprechen, welches bei *RunOnReceive* angegeben wurde, ansonsten wird das Script nicht ausgeführt.

Diese Zeilen fügen wir in das Bash-Script *smsin.sh* ein:

```
#!/bin/sh

#Absender überprüfen
#if [ "$SMS_1_NUMBER" != "<IHRE-HANDYNUMMER>" ] ; then
#       exit
#fi

#Handler ausführen, wird im Syslog angelegt
for i in $seq $SMS_MESSAGES ;
do
        logger -t smsd "SMS Teil: $i"
        eval logger -t smsd \"Absender:\${SMS_${i}_NUMBER}\"
        eval logger -t smsd \"Text:\${SMS_${i}_TEXT}\"
done

#SMS Text in .txt Datei schreiben
sudo echo "$SMS_1_TEXT" > /home/pi/gammu/sms.txt

#C-Programm ausführen
cd /home/pi && ./smsin
```

Erklärung des Scripts

```
if [ "$SMS_1_NUMBER" != "<IHRE-HANDYNUMMER>" ] ; then
        exit
fi
```

Überprüft ob die aktuelle Nummer der empfangenen SMS, der gewünschten Nummer entspricht. Diese Abfrage kann natürlich auch entfallen, dann können aber von jeder Nummer aus die Aktionen auf Ihrem PiXtend-System ausgeführt werden. Das ist in der Regel nicht gewollt und die Abfrage macht Sinn!

```
for i in $seq $SMS_MESSAGES ;
do
        logger -t smsd "SMS Teil: $i"
        eval logger -t smsd \"Absender:\${SMS_${i}_NUMBER}\"
        eval logger -t smsd \"Text:\${SMS_${i}_TEXT}\"
done
```

Geht alle gespeicherten, empfangenen SMS durch und schreibt die SMS-Nummer, die Nummer des Senders und den Text in die System Logdatei.

```
sudo echo "$SMS_1_TEXT" > /home/pi/gammu/sms.txt
```

Schreibt den Text der aktuellsten empfangenen SMS in eine angelegte *.txt* Datei. Die Datei legen wir später noch an.

```
cd /home/pi && ./smsin
```

Springt erst in das Verzeichnis in dem das kompilierte Programm liegt und führt dieses aus. Dieses Programm erstellen wir im weiteren Verlauf dieser Anleitung.

Nun das Script ausführbar machen und die benötigten Benutzerrechte vergeben

```
pi@raspberrypi ~ $ sudo chmod +x /home/pi/smsin.sh
pi@raspberrypi ~ $ sudo chmod 0777 -R /home/pi/smsin.sh
```

Die Konfiguartion des Script ist damit abgeschlossen.

6. Text Datei für SMS-Nachrichten

Damit wir später unsere Befehle auslesen können, schreiben wir diese in eine *.txt* Datei. Diese Datei wird später von unserem C-Programm aufgerufen.

Wir wechseln in unser Gammu-Verzeichnis und legen eine neue *.txt* Datei an

```
pi@raspberrypi ~ $ cd /home/pi/gammu
pi@raspberrypi ~ $ sudo touch sms.txt
```

Überprüfen, ob die sms.txt Datei angelegt wurde:

```
pi@raspberrypi ~ $ cd /home/pi/gammu
pi@raspberrypi ~ $ ls
```

Dem Verzeichnis geben wir noch alle Benutzerrechte

```
pi@raspberrypi ~ $ sudo chmod 0777 -R /home/pi/gammu
```

Hier sind wir nun fertig, weiter geht es mit dem Erstellen des C-Programms, basierend auf der PiXtend Linux Library *pxdev*.

7. Das C-File anlegen, programmieren und kompilieren

Hier wird erklärt, wie Sie das C-File anlegen, programmieren und kompilieren, damit das PiXtend-System via SMS gesteuert werden kann.

Zuerst legen wir ein neues C-File (für unseren Quelltext) an:

```
pi@raspberrypi ~ $ sudo nano smsin.c
```

Unter Kapitel 8 finden Sie ein Beispiel-Programm (Quelltext). Im ersten Schritt am Besten den gesamten Quelltext kopieren und in die C-Datei einfügen. Natürlich können Sie den Quelltext später für Ihre Bedürfnisse anpassen.

Nachdem wir das C-File mit dem Quelltext aus Kapitel 8 befüllt haben, können wir dieses abspeichern und kompilieren.

Da auch die verwendeten Bibliotheken **wiringPi** und **pixtend** kompiliert werden müssen, verwenden wir dafür folgenden Befehl.

```
pi@raspberrypi ~ $ sudo gcc -Wall -o smsin smsin.c -lpixtend -lwiringPi
```

Das kompilierte Programm *smsin* sollte nun im Home-Verzeichnis liegen, was wir kurz überprüfen:

```
pi@raspberrypi ~ $ cd /home/pi
pi@raspberrypi ~ $ ls
```

Nun geben wir dem Programm alle benötigten Benutzerrechte:

```
pi@raspberrypi ~ $ sudo chmod -R 0777 /home/pi/smsin
```

Herzlichen Glückwunsch, nun können Sie PiXtend mithilfe einer SMS steuern bzw. Werte auslesen.

Versenden Sie als Test eine SMS (von Ihrem Handy aus) mit einem der folgenden Befehle an Ihr PiXtend-System:

Befehl.Temperatur

Schließen Sie einen DHT11-Temperatursensor an GPIO0 an. Nach Empfang der SMS wird die Temperatur des Sensors ausgelesen und an die Rufnummer, welche Sie im C-File angegeben haben, versendet.

Befehl.Relais

Nach Empfang der SMS wird das Relais 0 auf PiXtend, eine Sekunde lang geschaltet. Dies kann am „klacken“ bzw. der zugehörigen LED erkannt werden.

Probieren Sie etwas herum und machen Sie sich mit dem C-Programm bzw. dessen Quellcode vertraut. Sie können selbstverständlich auch den Code verändern und neue Befehle hinzufügen.

Eine Idee wäre z.B. auch das automatische Herunterfahren des Raspberry Pi nach Empfang einer SMS.

Das Qube Solutions Team wünscht Ihnen viel Spass beim ausprobieren und programmieren!

8. Anhang: Das C-Programm (Quelltext)

```
#include <pixtend.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>

FILE *fp;

int ID_DHT = 0;
int RELAIS_NUMBER = 1;
char sTelephoneNumber[100] = "XXXXXXXXXX"; //Ihre Nummer hier eintragen
char sCommandTemperature[100] = "Befehl.Temperatur\n";
char sCommandRelais[100] = "Befehl.Relais\n";

float fTemperature = 0;
char sSmsText[100] = "";
char stemp1[100] = "echo Temperatur: ";
char stemp2[100] = " Celsius | sudo gammu-smsd-inject TEXT ";
char stemp3[100] = "";

int main(void){
int a,temp=0;
Spi_Setup(0);

//Text File öffnen und auslesen
fp = fopen("/home/pi/gammu/sms.txt", "r");

if(fp == NULL){
        printf("Datei konnte nicht geoeffnet werden.\n");
}
else {
//komplette Datei zeichenweise ausgeben
        while((temp = fgetc(fp))!=EOF) {
                sSmsText[a] = (char)temp;
                a++;
        }
        fclose(fp);
}
```

```
//Befehl Temperatur auslesen
if(strcmp(sSmsText,sCommandTemperature)==0){

        //Bit 4 in GPIO_Cntrl setzen -> DHT0
        Spi_Set_GpioControl(16);

        //Auslesen der Temperatur des DHT0
        temp = Spi_Get_Temp(ID_DHT);
        //Temperatur in Wert umrechnen
        fTemperature = (float)(temp>>8)*1.0;

        //Temperatur in String umwandeln und Strings zusammenfuegen
        snprintf(stemp3,7,"%.2f",fTemperature);
        strcat(stemp1,stemp3);
        strcat(stemp1,stemp2);
        strcat(stemp1,sTelephoneNumber);

        //SMS mit Temperaturwert versenden
        system(stemp1);

        //In .txt Datei schreiben, dass Temperatur ausgelesen wurde
        system("sudo echo Temperatur wurde ausgelesen > /home/pi/gammu/sms.txt");
}

//Befehl Relais schalten
else if(strcmp(sSmsText,sCommandRelais) == 0){

        //Relais eine Sekunde setzen, danach wieder ausschalten
        Spi_Set_Relays(RELAIS_NUMBER);
        delay(1000);
        Spi_Set_Relays(0);

        //In .txt Datei schreiben, dass Relais geschalten wurde.
        system("sudo echo Relais wurde geschalten > /home/pi/gammu/sms.txt");
}

return 0;
}
```

9. Anhang: FAQ – Häufig gestellte Fragen & Fehlerbehandlung

Ich kann keine Textnachrichten empfangen oder senden

Das SMS Tool *gammu* hat sich nicht immer als hundert Prozent zuverlässig erwiesen. Sollten Sie keine SMS empfangen, schauen Sie zunächst in den gammu-smsd Monitor. Dieser zeigt Ihnen gesendete und empfangene SMS an. Sollten Sie den Monitor nicht erreichen, schauen Sie noch einmal ob der UMTS Stick richtig konfiguriert wurde.

```
pi@raspberrypi ~ $ sudo gammu-smsd-monitor
```

Sollten Sie in der *gammu-smsd* Konfigurationsdatei ein Logfile angegeben haben, können Sie hier ebenfalls nach möglichen Ursachen sehen:

```
pi@raspberrypi ~ $ less /var/log/gammu-smsd
```

Mit *Umschalt* → *G* springen Sie zu den aktuellen Einträgen des Logfiles. Mit *Umschalt* → *Q* beenden Sie den Logger.

Die RunOnReceive Direktive führt mein Script nicht aus

Es hat sich herausgestellt, dass der Gammu SMS-Daemon manchmal Probleme mit Benutzerrechten hat. Überprüfen Sie ob das Bash-Script die Text-Datei und das C-File, sämtliche Berechtigungen haben. Ansonsten vergeben Sie die Rechte.

```
pi@raspberrypi ~ $ sudo chmod -R 0777 /PFAD/DATEINAMEN
```

Sollte dies immer noch nicht helfen, gehen Sie wie folgt vor

Gammu Benutzerrechte geben

```
pi@raspberrypi ~ $ sudo visudo
```

Den folgenden Befehl eingeben

```
gammu ALL=(ALL) NOPASSWD: ALL
```

Die Gammu Datei öffnen

```
pi@raspberrypi ~ $ sudo nano /etc/init.d/gammu-smsd
```

Den folgenden Befehl verwenden

```
USER=gammu to USER=root
```

speichern und den SMS Deamon neu starten

```
pi@raspberrypi ~ $ sudo service gammu-smsd restart
```

Darauf achten, dass beim *RunOnReceive* Script alle Programmaufrufe und Befehle mit sudo erfolgen.

Weitere Informationen erhalten Sie unter folgenden Links

Anleitung zum Modeswitch Package

http://www.draisberghof.de/usb_modeswitch/

Huawei Sticks mit Modeswitch unter Linux

https://wiki.ubuntuusers.de/USB_ModeSwitch/#Huawei-Sticks

Gammu SMS TOOL

https://wiki.ubuntuusers.de/Gammu/#Konfiguration

RunOnReceive Directive

https://wammu.eu/docs/manual/smsd/run.html

SMSD Configuration File

https://wammu.eu/docs/manual/smsd/config.html