Von der Idee bis zum Info-Display

Schritt 1: Einrichten des Raspberry Pi's und Testen des Displays

Schritt 2: Verbinden des GSM-Moduls mit dem Raspberry Pi

Schritt 3: Abrufen von SMS

Schritt 4: Automatisches Anzeigen der Nachrichten auf dem Display

Schritt 5: Anzeigen der Nachrichten im Browserfenster

Schritt 6: Weitere Überlegungen und Versuche

Begriffe:

GSM-Modul(Global System for Mobile Communications): Der GSM Standard ist der weltweit am meisten verbreitete Mobilfunk-Standard. Das Modul macht es möglich, mit Hilfe einer SIM Karte am PC Nachrichten zu empfangen oder Anrufe entgegen zu nehmen.

Raspberry Pi: Ist ein Einplatinencomputer der ursprünglich eine günstige Möglichkeit zum Programmieren und Experimentieren für Schüler zu Verfügung stellen sollte.

Serielle Schnittstelle/RS232: Ist ein Standard für eine bei älteren Computern häufig vorhandene serielle Schnittstelle und eignet sich gut zum Programmieren von Mikrocontrollern oder zum Einrichten einer Telefonanlage.

Carriage Return(Wagenrücklauf): Ist auf Ausgabegeräten für Text die Anweisung, den Anfang der Zeile anzusteuern.

Baudrate: 1 Baud ist die Geschwindigkeit, wenn 1 Symbol pro Sekunde übertragen wird. Die Baudrate einer Datenübertragung muss auf Sende- und Empfangsseite gleich sein.

Schritt 1: Einrichten des Raspberry Pi's

- Betriebssystem installieren

Zum Starten des Raspberry Pi's ist ein Betriebssystem erforderlich. Man schreibt es auf eine SD-Karte , um später davon booten zu können. Das kann man unter Windows mit dem Programm WIN32DiskImager realisieren. Dafür lädt man sich einfach von der Website http://www.raspberrypi.org/downloads/ das gewünschte Image des Betriebssystems runter und öffnet dieses anschließend mit dem WIN32DiskImager. Ich entschied mich für das Betriebssystem „Raspbian“, welches eine leicht veränderte Version der PC Distribution „Debian“ ist. Dann schreibt man dieses Image mit Hilfe des oben genannten Programms auf die SD-Karte(Achtung! SD-Karte wird formatiert; deshalb vorher Daten sichern!).

Anschließend kann man die SD-Karte in sein Raspberry Pi stecken, Maus, Tastatur und Display anschließen und dann mit Strom versorgen. Wenn alles geklappt hat, sieht man auf dem Display den Startvorgang des gewählten Betriebssystems und die Arbeit mit dem Raspberry Pi kann beginnen.

Schritt 2: Verbinden des GSM-Moduls mit dem Raspberry Pi

Es ging weiter mit dem Anschließen des GSM-Moduls an das Raspberry Pi.

Das GSM-Modul kann nur über einen RS232-Anschluss mit einer Steuereinheit verbunden werden. Dies stellte sich zunächst als Problem dar, denn das Raspberry Pi besitzt zwar eine Serielle Schnittstelle, jedoch keinen RS232-Anschluss. Ich musste also eine Möglichkeit finden, das RS232-Kabel trotzdem an das Raspberry Pi anzuschließen.

Es gibt einen Adapter von USB zu Serielle Schnittstelle, woran man das RS232-Kabel anschließen könnte. Das Raspberry Pi hat jedoch die Besonderheit, dass es über so genannte GPIO-Pins (General Purpose Input/Output = Allzweck Ein- und Ausgänge) verfügt. Nach der Installation eines dafür vorgesehenen Programmes lassen sich diese GPIO-Pins einzeln ansprechen und auch auslesen.

Ich entschied mich für die GPIO-Variante der Kommunikation zwischen Raspberry Pi und GSM-Modul, weil ich die Vorzüge des Raspberry Pi`s auch nutzen wollte. Jedoch kann man auch an diese Pins nicht einfach die Serielle Schnittstelle des GSM-Moduls anschließen, sondern braucht dafür einen Adapter von RS232(Serielle Schnittstelle) auf TTL (Transistor Transistor Logic). Die TTL-Schnittstelle besteht aus nur 4 Anschlüssen: 5V, Masse, TXD und RXD, wobei die Anschlüsse TXD und RXD für das Senden und Empfangen der Daten zuständig sind. Solch einen Adapter stellte mir mein Betreuer Herr Bui freundlicherweise zu Verfügung, sodass ich nur noch recherchieren musste, wo ich welches Kabel anschließen muss. Mit Hilfe der Pin Belegung des Raspberry Pi's und des Schaltplans des RS232 zu TTL Adapters war dies kein Problem mehr.

Die Verbindung zwischen Raspberry Pi und GSM-Modul stand nun und es ging weiter mit einem ersten Funktionstest. Nur wie und mit welchem Programm? Ich habe vorher noch nie mit einem GSM-Modul gearbeitet, weder an einem Windows PC noch an einem Raspberry Pi. Ein Student gab mir den Tipp mal nach „AT-Befehle“ zu googeln. Das sind Befehle, die ursprünglich zum Konfigurieren von Modems genutzt worden sind. Mit ein paar solcher Befehle bewaffnet begab ich mich auf die Suche nach einem geeigneten Programm für Linux und speziell für das Raspberry Pi, auf dem eine abgeänderte Linux Version läuft. Dabei stieß ich auf das Programm „minicom“, welches ein geeignetes Terminal für AT-Befehle sein sollte. Nach der Installation über die Linux-Kommandozeile mit dem Befehl *sudo apt-get install minicom*, startete ich Minicom und verschaffte mir einen ersten Eindruck über das Programm. Auf den ersten Blick war Minicom sehr unübersichtlich und die Bedienung war auch nicht eindeutig, deshalb gelang es mir auch nicht auf Anhieb, das verbundene GSM-Modul anzusprechen.

Deshalb war die nächste Überlegung, das GSM-Modul an einen PC mit RS232 Schnittstelle anzuschließen um zu sehen, ob und wie das GSM-Modul funktioniert.

Mit dem „hyper Terminal“ unter Windows gelang es mir schnell, einen AT-Befehl wie „AT+CSQ“ zu senden. Wie gewollt, bekam ich dann eine Antwort über die aktuelle Signalstärke des GSM-Moduls. Daran konnte ich sehen, dass das GSM-Modul funktionstüchtig war.

Mit diesem Teilerfolg ging es dann wieder an das Raspberry Pi und an das Konfigurieren von Minicom. Nach längerem Ausprobieren mit Carriage Return(CR) und Baudrate gelang es mir dann mit dem GSM-Modul zu kommunizieren, die Signalstärke wurde mir endlich angezeigt!

Schritt 3: Abrufen von SMS

Weiter ging es jetzt mit dem Abrufen von SMS-Nachrichten vom GSM-Modul.

Wie schon beim Abfragen der Signalstärke fängt auch dieser Befehl mit „AT+“ an, gefolgt von „CMGR=X“ lässt sich eine SMS aus Speicherplatz X auslesen.

So weit die Theorie, bei Eingabe von „AT+CMGR=1“ bekam ich eine ERROR Meldung zurück, anstatt den Nachrichtentext von SMS 1. Nach wiederholtem Versuch mit erneuter Eingabe des Befehls, kam mir die Idee, dass diese ERROR Meldung etwas mit der noch nicht eingelegten SIM Karte zu tun haben könnte. Also habe ich eine SIM-Karte in das GSM-Modul eingelegt und den Befehl erneut gesendet. Als Antwort bekam ich wieder „ERROR“.

Nach erneuter Recherche im Internet kam ich auf die Idee, dass man zuerst die SIM-Karte entsperren muss, soweit ein Schutz eingerichtet ist (wie man das auch vom Handy gewohnt ist). Ob diese SIM-Karte geschützt ist, kann man mit dem Befehl „AT+CPIN?“ herausfinden. Erscheint daraufhin die Ausgabe „SIM PIN“ muss man mit dem Befehl „AT+CPIN=PIN“ die Karte entsperren.

Nach erfolgreichem Einrichten der SIM-Karte kann man nun die Nachrichten abfragen, sofern die Karte nicht wegen zu langer Inaktivität vom Betreiber gesperrt wurde. Dann muss eine andere Karte her oder das Guthaben wieder aufgeladen werden.

Man kann die Nachricht aus dem Speicherplatz 2 auslesen, indem man die Zahl am Ende, also auf „AT+CMGR=2“, ändert.

Das Löschen einer SMS erfolgt mit dem Befehl „AT+CMGD=X“(X steht wieder für den zu löschenden Speicherplatz).

Nun könnte man, mit den richtigen Befehlen, das Raspberry Pi in Verbindung mit dem GSM-Modul zum Senden und Empfangen von SMS oder auch als Telefon nutzen, sofern ein Mikrophon angeschlossen ist.

Schritt 4: Automatisches Anzeigen der Nachrichten auf dem Display

Im Moment lässt sich unser Raspberry Pi nur manuell und durch die Eingabe der einzelnen Befehle als Anzeige für SMS nutzen

Beim vorherigen Konfigurieren von Minicom ist mir aufgefallen,

- dass es die Möglichkeit gibt eine Capture Datei von Minicom erstellen zu lassen. Das bedeutet, dass alles was man in Minicom eintippt oder als Antwort bekommt, in einer Datei gespeichert wird und somit auch nach dem Schließen von Minicom noch zu Verfügung steht.
- dass man durch den Zusatz „-S“ beim Aufrufen von Minicom, eine vorher erstellte Script Datei öffnen kann, mit der Minicom dann arbeitet. Das löste das Problem, dass man sonst manuell jeden Befehl in Minicom eintippen musste um das GSM-Modul anzusprechen.

Ich war also auf dem Weg mir einen Befehl zu basteln, in dem alles, was mit Minicom zu tun hat, automatisch aufgerufen wird. Die AT-Befehle speicherte ich in einer externen Datei ab, sodass Minicom bei jedem Aufruf die gleichen Befehle ausführt. Der fertige Befehl für die Linux Kommandozeile, ergänzt durch die Baudrate, sah dann so aus:

minicom -S /home/pi/Anzeige/BefehlSMS1 -b 9600 -o -D /dev/ttyAMA0 -C /home/pi/Anzeige/minicom1.cap

der Zusatz -C steht für das Anlegen einer Capture Datei ,

der Zusatz -D legt den Ausgang fest, welchen Minicom ansprechen soll. In diesem Fall ist das der Pfad zu der TTL-UART-Schnittstelle (/dev/ttyAMA0).

-o steht für das Nichtinitialisieren des Modems beim Start von Minicom, wie man auch dem help-Befehl *minicom –help* von Minicom entnehmen kann:

```
pi@raspberrypi ~ $ minicom --help
Usage: minicom [OPTION]... [configuration]
A terminal program for Linux and other unix-like systems.

  -b, --baudrate         : set baudrate (ignore the value from config)
  -D, --device           : set device name (ignore the value from config)
  -s, --setup            : enter setup mode
  -o, --noinit           : do not initialize modem & lockfiles at startup
```

Das nächste Ziel war jetzt, diesen Befehl in einem bestimmten Intervall zu wiederholen, um kontinuierlich neue SMS abzurufen.

Dazu habe ich eine Datei *start.sh* erstellt, also ein Linux Shell Script in dem der Befehl für Minicom in einer Endlosschleife steht. Anschließend wartet das Script 20 Sek., bevor die Schleife wieder von vorne beginnt.

Jetzt mussten noch die Speicherplätze zugeordnet werden, damit immer die aktuelle SMS angezeigt wird.

Ich erstellte zunächst eine neue Variable in der *start.sh* Datei, welche in Zukunft nur noch die Werte 1 oder 2 annehmen sollte. Meiner Erfahrung nach werden neue SMS immer auf dem kleinsten freien Speicherplatz gespeichert. Das bedeutete, dass ich mindestens die zwei untersten Speicherplätze benutzen musste, um dort abwechselnd Nachrichten speichern oder löschen zu können. Dabei half mir eine selbstgeschriebene Funktion, die beide Speicherplätze ausliest und entscheidet, ob ein oder zwei Speicher belegt sind. Wenn beide Speicherplätze belegt sind, wird automatisch die zuvor angezeigte Nachricht gelöscht. Dann wird die neue Nachricht angezeigt und der andere Speicherplatz bleibt so lange unbesetzt, bis wieder eine neue Nachricht empfangen wird. So wird immer die aktuelle Nachricht angezeigt.

Der automatisierte Ablauf funktionierte jedoch nur, wenn nach dem Starten des Raspberry Pi's das Linux Script *start.sh* ausgeführt und dann der Browser manuell geöffnet wurde.

Da das Display aber vollautomatisch funktionieren sollte, musste ich die beiden Dateien/Programme dazu bekommen, mit dem Raspberry Pi zusammen zu starten. Dies gelang mir letztendlich über das Bearbeiten der Autostartdatei von der LXDE Oberfläche des Raspbian Betriebssystems.

Nun war das Raspberry Pi für den ersten vollautomatischen Textempfang bereit.

Bis auf kleine Schönheitsfehler bezüglich des *Start.sh* Linux Scripts war die grundsätzliche Funktionalität des Info-Displays und der Grundgedanke des Projektes erreicht.

So viel zum automatischen Auslesen der SMS Nachrichten.

Schritt 5: Anzeigen der Nachrichten im Browserfenster

Die Ausgabe des Terminals ist in sehr kleiner Schrift geschrieben. Da das Display später neben das Namensschild an der Bürotür des Professors installiert werden sollte, wäre die Schrift zu klein, um die Nachricht beim Vorbeigehen lesen zu können.

Die erste Frage, wie ich die SMS Anzeigen lassen will, war für mich sehr schnell geklärt. Da ich mich gut mit dem Erstellen von HTML-Dateien auskenne, wollte ich die spätere Nachricht per Browserfenster darstellen lassen und dann über das Zusammenspiel mit CSS die Schriftgröße und Farbe einstellen.

Dazu habe ich zuerst eine HTML-Datei *Index.html* erstellt, die in einem *Iframe* die Capture Datei von Minicom anzeigt. Das funktionierte auch sofort sehr gut, nur die Schriftgröße ließ sich nicht einstellen.

Nach längerem hin und her Überlegen entschied ich mich für den Linux Befehl *sed*. Mit dem kann man in Textdateien bestimmte Strings suchen und durch andere ersetzen, was insofern sinnvoll war, weil ich Angaben wie die Handynummer, die auch mit in der Capture Datei stand, durch HTML Tags ersetzen konnte.

Damit schlug ich zwei Fliegen auf einen Streich.

- Erstens habe ich die unnötigen Informationen entfernt und durch nicht sichtbare HTML Tags ersetzt.

- Zweitens legten diese die Schriftgröße sowie die Hintergrundfarbe fest.

Damit die Seite immer auf dem aktuellen Stand ist, habe ich eine Javascript Funktion in die *Index.html* Datei ergänzt, welche die komplette Seite alle 20 Sekunden neu lädt.

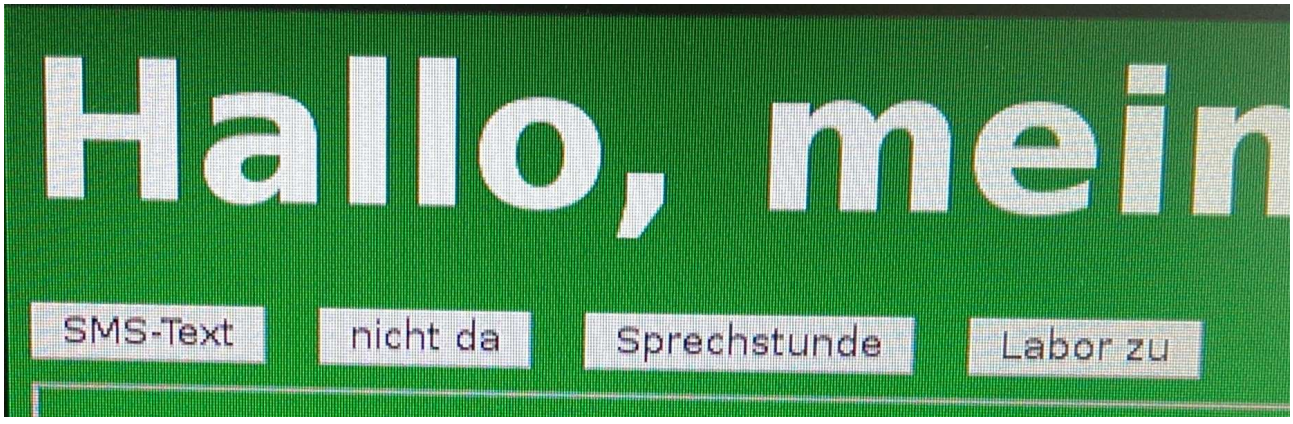
Zum Schluss habe ich die Startseite angepasst, damit bei jedem Browserstart die entsprechende *Index.html* Seite angezeigt wird.

Dann fiel mir auf, dass nicht nur der letzte, sondern auch alle vorherigen Befehle von Minicom auf der Anzeige erschienen. Das führte dazu, dass der aktuelle Befehl auf dem Display nicht mehr ohne scrollen sichtbar war. Schuld daran war das Programm Minicom, das einen neuen Befehl und die Ausgabe des Befehls unter die schon vorhandenen Befehle in die Capture Datei geschrieben hat anstatt den alten Befehl durch den neuen zu ersetzen. Dadurch häuften sich die Befehle auf dem Display.

Ich löste das Problem, indem ich eine Funktion in die Datei *start.sh* schrieb, die den Inhalt der Capture Datei nach jedem Durchgang löscht.

Schritt 6: Weitere Überlegungen und Versuche

Als weiteres Extra habe ich Textbausteine zur Auswahl einiger Standardnachrichten hinzugefügt, die man per Mausklick auf die entsprechende Schaltfläche auswählen kann. Dann wird diese auf dem Display angezeigt,ohne dass man dafür extra eine SMS versenden muss.



In anschließenden Tests des SMS Empfangs habe ich Versuche zur Darstellung von Umlauten gestartet, denn Buchstaben wie ö/ä/ü wurden als unlesbare Zeichen wie zum Beispiel „~ “

angezeigt. Dazu verwendete ich den Befehl sed, welcher in der Lage ist Textabschnitte zu suchen und dann durch anderen Text zu ersetzen. Beispielsweise sah ein Befehl zum Anzeigen des Buchstaben “ö“ dann so aus:

sed -e ' s/\/oe/g' /home/pi/Anzeige/minicom$speicher.cap > /home/pi/Anzeige/minicom.html

Diese Befehle habe ich zum Funktionsumfang der Anzeige hinzugefügt. Nur das Anzeigen des „Ü“ konnte ich mit diesen Befehlen leider noch nicht lösen.

Des weiteren habe ich mir Gedanken über eine Filterung der Telefonnummer des Absenders gemacht, sodass nur ausgewählte Personen eine Nachricht an das Info-Display schicken können.

Eine Erweiterung durch einen Bewegungsmelder war ebenfalls eine Überlegung. Dabei könnte man den Bewegungsmelder ein Relais steuern lassen, welches das Display anschaltet, sobald jemand davor steht.