Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Computer Engineering and
Networks Laboratory

Bachelor Gruppenarbeit

am Departement für Informationstechnologie

und Elektrotechnik

Mobile Application for On-Site Sensor Installation Support

Guido Hungerbühler

Oliver Knecht

Suhel Sheikh

Betreuer: Matthias Keller und Dr. Jan Beutel

Professor: Prof. Dr. Lothar Thiele

Frühlingssemester 2009

Abstract

In the context of geological studies at the University of Zurich, the variability and the influence of topography on soil temperatures should be explored in the mountains. For this purpose, geologists will distribute hundreds of temperature sensors in the Swiss Alps and collect them after a certain time to analyze the data.

This group project describes the development of a portable system to assist the geologists in collecting measurements with the aid of iButton sensors. With this customized solution it is possible to save a lot of time due to automated processes for distributing, collecting and reading the temperature sensors. This solution consists of the developed software, as well as three major hardware components. These are a reader for the sensors, a GPS and a digital camera. All these components are connected to a net-book and form an ideal tool adapted to the particular needs of the geologists.

In this work, we will describe the components of the system, the functionality and the design of the software in more detail.

Kurzbeschreibung

Im Rahmen von geologischen Untersuchungen der Universität Zürich soll mit umfangreichen Temperaturmessungen die Variabilität von Bodentemperaturen und der Einfluss der topografischen Gegebenheiten im Gebirge erforscht werden. Dazu werden hunderte Temperatursensoren in den Schweizer Alpen verteilt und nach einer gewissen Zeit wieder eingesammelt und ausgewertet.

Diese Gruppenarbeit beschreibt die Entwicklung eines portablen Systems zur Unterstützung der Geologen bei der Erhebung von Messdaten mit Hilfe von iButton Sensoren. Durch eine angepasste Lösung soll, durch automatisierte Abläufe für das Verteilen und Auslesen der Temperatursensoren, viel Zeit eingespart werden. Das System besteht aus der entwickelten Software sowie drei wesentlichen Hardwarekomponenten. Dies sind ein Lesegerät für die Sensoren, ein GPS Modul für die Standortbestimmung und Navigation im Gelände und eine Digitalkamera. Alle diese Komponenten werden an einem Netbook angeschlossen und bilden ein auf die Aufgabe der Geologen angepasstes Werkzeug.

In dieser Arbeit werden die Komponenten des Systems näher vorgestellt und die Funktionalität sowie der Aufbau der Software im Detail beschrieben.

Inhaltsverzeichnis

1 Einleitung 1
1.1 Aufgabenstellung 2

2 Vorstellung der Hardware 3
2.1 iButton Temperatursensoren 3
2.2 Serielles Lesegerät 4
2.3 Openmoko Neo FreeRunner 6
2.3.1 Probleme 7
2.4 Asus EeePC 10
2.5 USB Lesegerät 11
2.6 Digitalkamera 11

3 Verwendete Softwarekomponenten 13
3.1 SQLite 13
3.2 QT 14
3.3 libgphoto2 14
3.4 GPS Daemon 14
3.5 Maxim Software Ressourcen 15

4 Beschreibung der Klassen 17
4.1 Programmstart und Hauptfenster 17
4.2 Programmieren der iButtons vor Ort 17
4.3 Programmieren der iButtons 18
4.4 Einsammeln der iButtons 19
4.5 Importieren von Daten 20
4.6 Löschen von Datensätzen 22
4.7 Bearbeiten des Mission Parameter File 22
4.8 Generelle Klassen 23
4.8.1 Datenbank 23
4.8.2 Grafische Widgets 24
4.8.3 Kommunikation mit iButton Sensoren 25
4.8.4 Kommunikation mit dem GPS Empfänger 26
4.8.5 Kommunikation mit der Kamera 26
4.8.6 Sonstige Klassen 27

5 Test 29
5.1 Funktionalität der Software 29
5.2 Sensoren stationär programmieren 29
5.2.1 Programmieren der iButton Sensoren 29
5.2.2 Importieren externer Daten 30
5.2.3 Einsammeln der Sensoren 30
5.2.4 Daten entfernen 30
5.3 iButton vor Ort programmieren 31
5.3.1 Programmieren der iButton Sensoren 31
5.3.2 Import von zusätzlichen Daten 32

6 Bedienungsanleitung 33
6.1 Installation 33
6.2 Das Hauptfenster 34
6.3 Programmieren vor Ort 35
6.4 Programmieren im Büro 37
6.5 Einsammeln der iButtons 38
6.6 Hinzufügen externer Daten 40
6.6.1 Format der .csv Dateien: 41
6.7 Bearbeiten der Datenbank 43
6.8 Bearbeiten des Parameter Files 45
6.9 Die Datenbank 47
6.9.1 iButtons 47
6.9.2 Footprints 48
6.9.3 Photos 48
6.9.4 MeasurementProfile 49
6.9.5 Measurement 49
6.10 Fehlerbehandlung 50
6.10.1 Programmieren vor Ort 50
6.10.2 Programmieren im Büro 51
6.10.3 Einsammeln der iButtons 51
6.10.4 Hinzufügen externer Daten 52
6.10.5 Bearbeiten der Datenbank 53
6.10.6 Bearbeiten des Parameterfiles 53

7 Schlusswort 55

I. Abbildungsverzeichnis 57
II. Abkürzungsverzeichnis 59
III. Literaturverzeichnis 61

1 Einleitung

Im Rahmen von geologischen Untersuchungen der Universität Zürich, soll eine grosse Anzahl von Temperatursensoren in den Schweizer Alpen verteilt werden. Bei diesen Untersuchungen geht es im Wesentlichen darum, die Variabilität von Bodentemperaturen im Gebirge zu erforschen.

Dazu werden an ungefähr 40 verschiedenen Standorten jeweils etwa 10 Temperatursensoren ausgelegt, welche über einen Zeitraum von einem Jahr die Bodentemperatur aufzeichnen sollen. Die Standorte der Sensoren unterscheiden sich in Neigung, Exposition, Höhe über Meer und Bodenbeschaffenheit.

Schlussendlich möchte man durch diese umfangreichen Messungen Erkenntnisse darüber gewinnen, welchen Einfluss diese topografischen Faktoren auf die Temperaturen in geringer Bodentiefe haben.

Auf folgender Abbildung sieht man die Standorte nachfolgend auch Footprints genannt, an welchen im Sommer 2009 die Temperatursensoren auf dem Piz Corvatsch verteilt wurden.

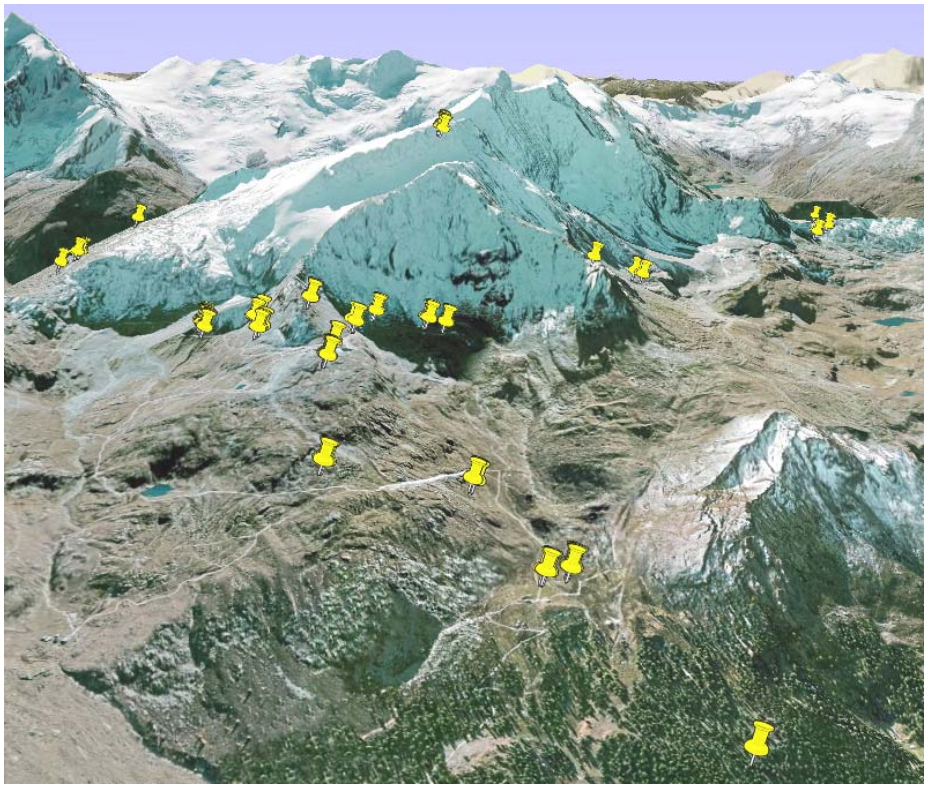

Abb. 1: Footprints auf dem Piz Corvatsch

1.1 Aufgabenstellung

Um die Daten zu gewinnen, müssen alle Sensoren programmiert und verteilt werden. Nach vollendeter Messung müssen dieselben Sensoren wieder gefunden, eingesammelt und ausgelesen werden. Das Programmieren und Auslesen der Sensoren, verbunden mit der Aufzeichnung aller relevanten Daten, ist sehr zeitaufwändig.

Unsere Aufgabe war es nun, eine mobile und robuste Lösung zu entwickeln, welche die Geologen bei ihrer Forschung bestmöglich unterstützt. Das heisst, es sollte eine Software geschrieben werden, welche die Abläufe vom Programmieren, Auslesen, Austeilen und Einsammeln der Sensoren automatisiert und somit die Arbeit im Feld erleichtert und zeitsparender gestaltet. Die fertige Software soll dann auf ein mobiles Endgerät portiert werden.

Es ist unbedingt notwendig die genaue Position der Sensoren zu kennen, um diese nach erfolgter Messung wieder zu lokalisieren und einen möglichst aussagekräftigen Datensatz zu erhalten. Deshalb sollen beim Austeilen der Sensoren die GPS-Koordinaten des Standorts bestimmt und automatisch zum jeweiligen Temperatursensor gespeichert werden. Bei früheren Messungen hat sich gezeigt, dass es ebenfalls sinnvoll ist als zusätzliche Hilfe vom genauen Standort und der Umgebung des Sensors einige Fotos zu machen. Deshalb soll die Software ebenfalls die Benutzung einer Digitalkamera unterstützen.

Alle gesammelten Daten sollen dann in einer Datenbank abgelegt werden, um die Konsistenz und eine einfache Auswertung bzw. Weiterverarbeitung der Daten zu gewährleisten.

Zusammenfassend können folgende Funktionalitäten der geplanten Software aufgelistet werden:

- Programmierung, Auslesen der Temperatursensoren.
- Verarbeitung von GPS Daten, GPS Navigation.
- Anbindung einer Digitalkamera.
- Speicherung der Daten in einer Datenbank.

2 Vorstellung der Hardware

2.1 iButton Temperatursensoren

Ein iButton besteht im Wesentlichen aus einem Mikroprozessor und einer Batterie, eingekapselt in einer wasserdichten, kompakten Edelstahlhülle (1). Ein solcher Sensor hat etwa den Durchmesser eines Knopfes. Die iButtons gibt es in verschiedenen Ausführungen als EEPROM, RTC, RAM, ID, und Datenlogger (2).

Der in diesem Projekt verwendete iButton DS1922L ist ein programmierbarer Datenlogger mit integriertem digitalem Thermometer. Der im iButton eingebettete Mikroprozessor besitzt einen integrierten 1-Wire Transmitter und Receiver zum einfachen Datenaustausch. Ausserdem enthält er eine einzigartige zehnstellige Identifikationsnummer und eine Echtzeituhr. Der integrierte Sensor misst die Temperatur und speichert den Wert in einem separaten, geschützten Speicherbereich. Dabei kann das Messintervall von 1 Sekunde bis zu 273 Stunden eingestellt werden. Je nach gewählter Einstellung kann der Start der Messreihe von 0 Sekunden bis zu theoretischen 31 Jahren verzögert werden.

Technische Daten (3):

- 8 kB Daten Speicher im 8-Bit (8192 Messungen) oder 16-Bit Format (4096 Messungen).
- Digitales Thermometer mit 8-Bit (0.5 °C) oder 11-Bit (0.0625 °C) Auflösung.
- Sensorgenauigkeit von ±0.5 °C von -10 bis +65 °C mit Softwarekorrektur.
- Kommunikation über 1-Wire Protokoll mit 15.4 kbps im Standardmodus und bis zu 125 kbps im Overdrive-Modus.
- Kurzschlussfester Kontakt über das Gehäuse.
- Enthält eine einzigartige 64-Bit ID um eine eindeutige Identifikation zu ermöglichen.
- Interne 3V Lithium Batterie.

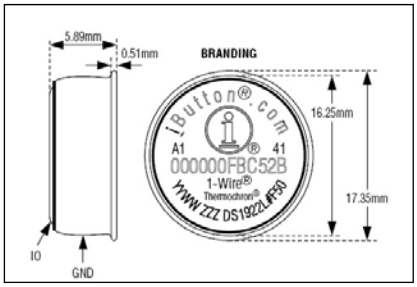

Abb. 2: Pin Belegung und Abmessungen

Das Blockdiagramm zeigt die Beziehungen der einzelnen internen Komponenten des iButtons.

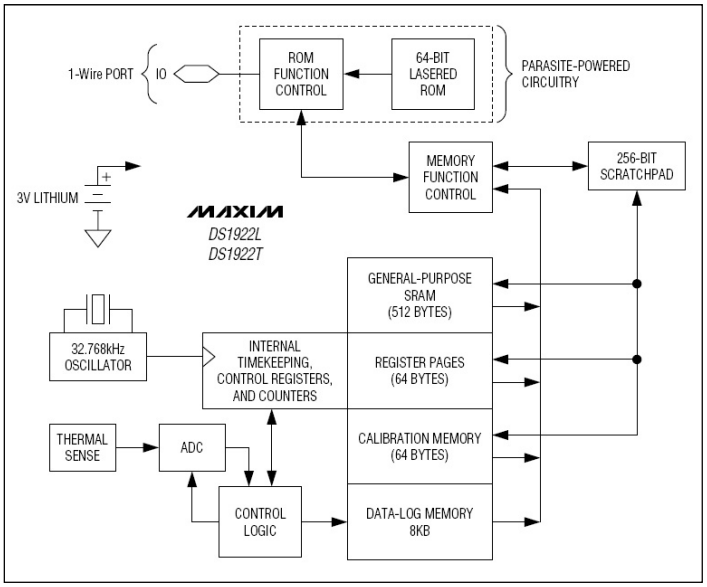

Abb. 3: DS1922L Block Diagram

2.2 Serielles Lesegerät

Um die Kommunikation mit den iButtons zu testen wurde zuerst ein serielles Lesegerät gebaut. Das Herz des Lesegerätes ist das IC DS2480B (4) von Maxim. Es handelt ich dabei um ein 1-Wire zu UART Schnittstellenwandler, mit dem das UART Protokoll vom Computer in das 1-Wire Protokoll umgewandelt wird. Das Lesegerät wurde nach dem Vorbild des seriellen Lesegerätes DS9097U von Maxim aufgebaut (5). Folgendes Blockdiagramm zeigt die vereinfachte Funktion des Gerätes.

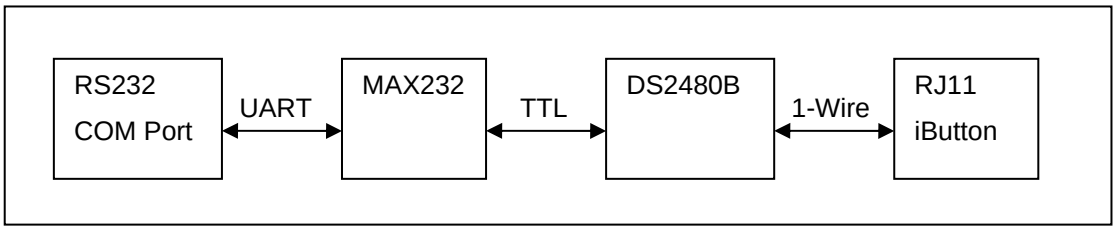

Abb. 4: Blockdiagramm, Serielles Lesegerät

Der Baustein MAX232 ist ein Pegelwandler und wandelt den 5 Volt TTL Pegel vom DS2480B in den ±12 Volt RS232 Pegel um, sodass man das Lesegerät an die serielle Schnittstelle des Computers anschliessen kann.

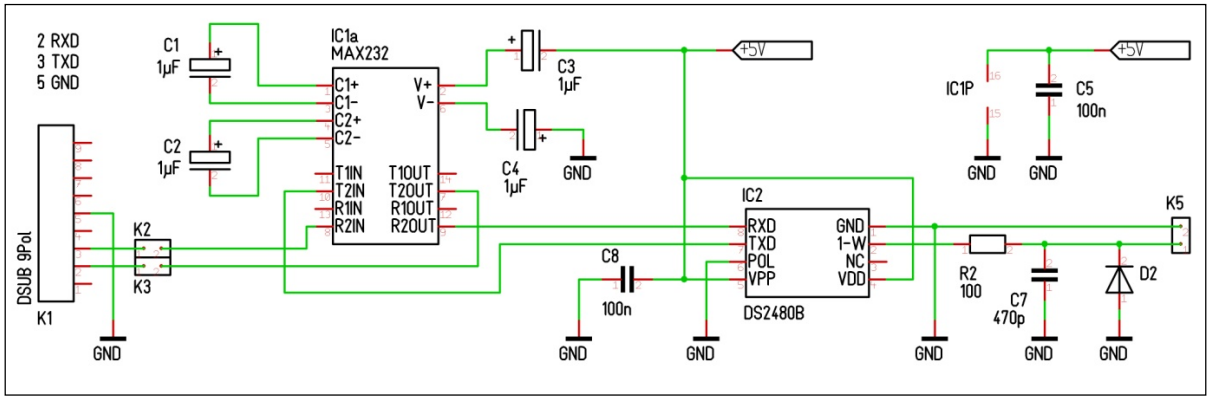

Abb. 5: Schaltplan, Serielles Lesegerät

2.3 Openmoko Neo FreeRunner

Das Openmoko Neo FreeRunner ist ein Mobiltelefon mit einem Open Source Softwarestack und erlaubt somit dem Benutzer die mobile Plattform frei zu verändern und zusätzliche Funktionalitäten hinzuzufügen. Neben der Software-Struktur ist auch die komplette Hardware inklusive Schaltplänen und CAD Dateien frei erhältlich (6).

Abb. 6: Openmoko Neo Freerunner

Um die Forscher bei der Installation der iButton Sensoren, sowie bei der Datenübertragung zu unterstützen soll ein Gerät gefunden werden, welches einfach zu bedienen und portabel ist. Mit den folgenden Eigenschaften erfüllt das Openmoko alle gesuchten Voraussetzungen und wurde deshalb auch am Anfang der Arbeit als Hardwarebasis für dieses Projekt gewählt (7).

- Der Openmoko Neo Software Stack enthält ein vollständiges Linux mit X-Server.
- Grosses VGA, 640x480 Pixel Touchscreen Display.
- Integrierter GPS-Empfänger mit der Möglichkeit eine externe Antenne anzuschliessen.
- USB Host Funktion mit 500mA Energieversorgung.
- Debug-Schnittstelle mit Zugang zu diversen Kommunikationsbussen wie SPI, I2C und UART.
- Mobiler und stationärer Internetzugang über Ethernet, Wireless LAN und GPRS.

Der Neo Software Stack bietet eine Vielzahl von Bibliotheken an, welche die Implementierung der Software enorm erleichtert. Ausserdem kann der Software Stack mit einem Packetmanager um viele weitere Bibliotheken und Pakete erweitert werden (8).

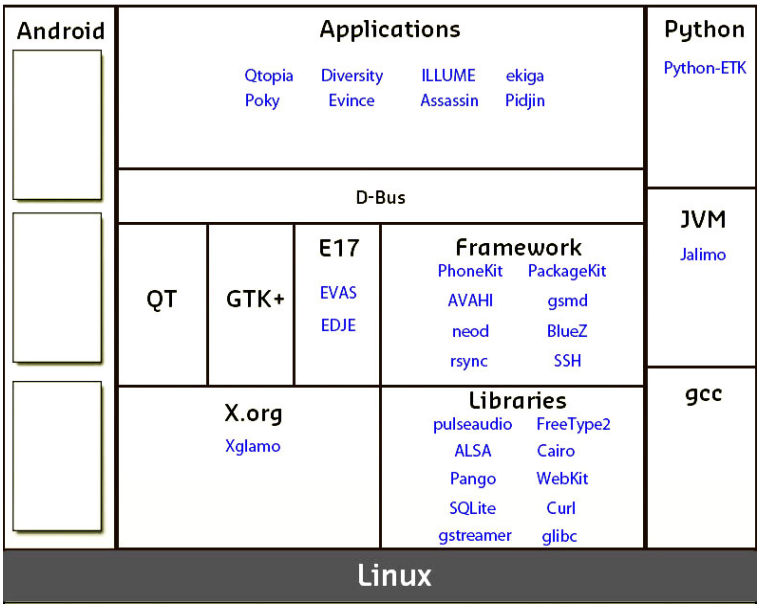

Abb. 7: Neo Software Stack

Da das Gerät über keine eigene Kamera verfügt, ist es notwendig eine externe Digitalkamera anzuschliessen. Ein Problem war, dass nur eine begrenzte Anzahl Anschlüsse vorhanden sind. Um auf einen zusätzlichen USB-Hub verzichten zu können sollte die Kommunikation mit den iButtons mit einem seriellen Lesegerät über die UART Schnittstelle und die Kommunikation mit der Kamera über die USB Schnittstelle des Openmokos erfolgen.

2.3.1 Probleme

Bei ersten Tests stellten wir fest, dass die Akkulaufzeit des Neo FreeRunners sehr begrenzt ist und es somit kaum sinnvoll wäre mit diesem Gerät im Feld die iButton Sensoren zu verteilen. Beim Openmoko ist vorgesehen, das Gerät während dem Betrieb im „Device-Mode“ über die USB Schnittstelle zu laden. Um die Digitalkamera trotzdem am Openmoko betreiben zu können muss sich das Mobiltelefon jedoch im „Host-Modus“ befinden. Dabei versorgt das Openmoko externe Geräte mit Strom.
Die Richtung des Leistungsflusses im USB Port ist jedoch unabhängig von dessen Modus. D.h. es ist möglich den FreeRunner zu laden und den USB Port gleichzeitig im „Host Modus“ zu betreiben (9).

Da das Openmoko aber nur über einen USB Anschluss verfügt, aber einer zum Laden und einer für die Digitalkamera benötigt wird, wurde von uns eine kleine Platine entwickelt, auf welcher die USB-Anschlüsse sowie die Schaltung des seriellen Lesegeräts und ein Anschluss für den iButton-Adapter aufgebaut sind.
Ein DC-DC-Konverter sorgt dafür, die 12V einer externen Batterie in die Ladespannung von 5V umzuwandeln.

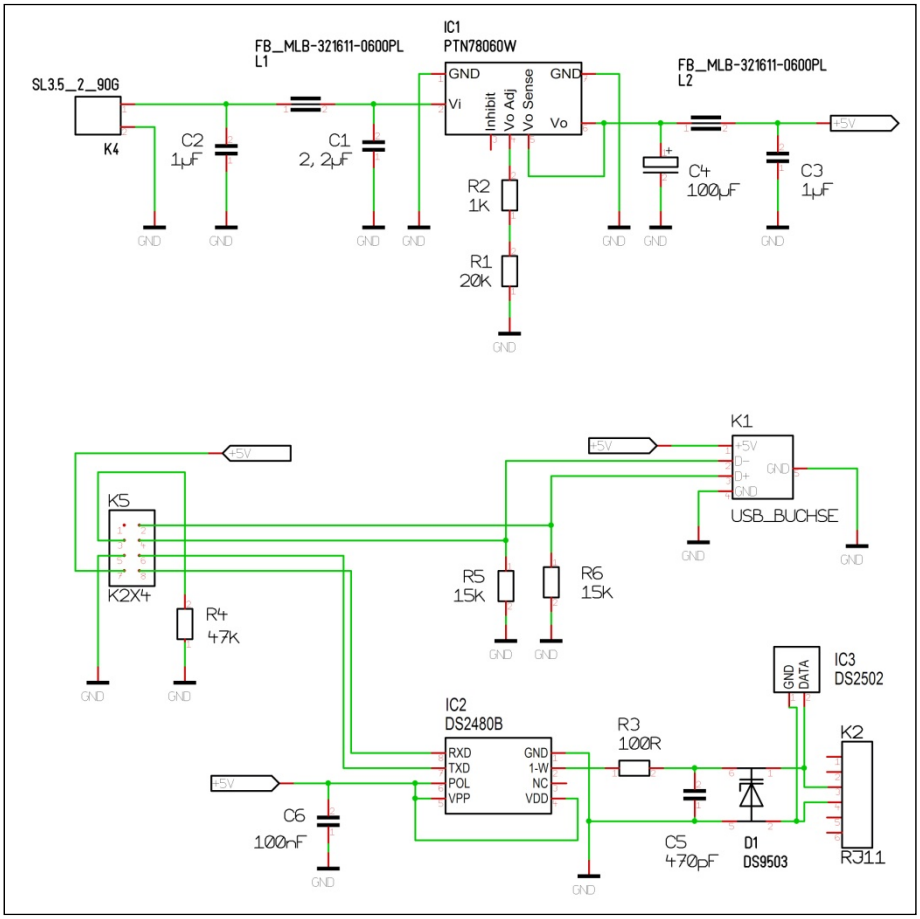

Abb. 8: Schaltplan, Openmoko Zusatz PCB

Das PCB wird auf der Rückseite des Openmokos angebracht und durch eine kleine Edelstahlhülle geschützt.

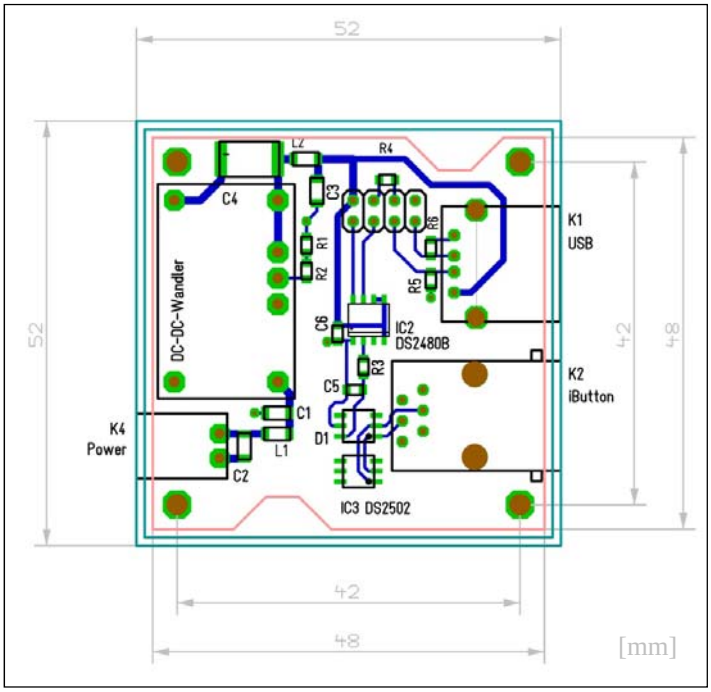

Abb. 9: CAD Layout der Platine.

Wie beim Test-Lesegerät erfolgt die Kommunikation mit den iButtons über den Seriell-zu-1-Wire Konverter DS2480B. Die Debug-Schnittstelle im inneren des Openmoko verfügt über einen UART Datenbus, über welchen die Kommunikation mit den iButton Sensoren stattfindet. Dazu soll der UART Bus mit zwei Datenleitungen auf das PCB geführt und mit dem DS2480 verbunden werden.
Beim Testen der Kommunikationsverbindung traten jedoch erhebliche Probleme auf. Wir bemerkten, dass der Kernel des Betriebssystems auf dem Openmoko diese Schnittstelle ebenfalls benutzt. D.h. es kam zu einer Kollision von Daten auf den Leitungen, welche zu einem Absturz des Betriebssystems führten.
Dieses Problem kann nicht behoben werden und deshalb war es notwendig einen neuen Ansatz für die Kommunikation mit den iButtons zu suchen.

Diese und weitere kleinere Probleme brachten uns zum Schluss, dass das Openmoko FreeRunner noch nicht die Marktreife erreicht hatte, welche für dieses Projekt notwendig gewesen wäre. Deshalb suchten wir nach einer alternativen Hardware-Plattform und fanden den Asus EeePC.

2.4 Asus EeePC

Der EeePC von Asus gehört zu einer Reihe von handlichen, kostengünstigen und verhältnismässig leistungsfähigen Netbooks. Das von uns gewählte Modell hat folgende wichtige technische Daten (10):

- Grosses 10,2-Zoll-Display
- 160GB 2,5-Zoll Harddisk
- 1,66 GHz Intel Atom Prozessor
- Akkulaufzeit von bis zu 9 Stunden
- Linux Betriebssystem

Abb. 10: Asus EeePC

Der Hauptgrund für die Wahl dieses Netbooks war die hohe Akkulaufzeit, was für die Arbeit im Feld enorm wichtig ist. Der EeePC verfügt über kein integriertes GPS-Modul, hat aber genügend USB Schnittstellen um zusätzlich zu einem iButton Lesegerät eine Digitalkamera und ein externes GPS-Gerät zu betreiben, ohne einen USB Hub zu benötigen.
Es ist ebenfalls vorteilhaft, dass wir auf dem EeePC die gleiche Linux Version installieren können, die wir auch auf unseren Entwicklungscomputern haben. Dadurch fällt das Portieren der Software mit einem Cross-Compiler auf ein anderes System weg.

2.5 USB Lesegerät

Mit dem EeePC wurde das serielle Lesegerät aufgrund der fehlenden seriellen Schnittstelle am Computer überflüssig und wir entschieden uns für das USB Lesegerät DS9490 des Herstellers Maxim (11).

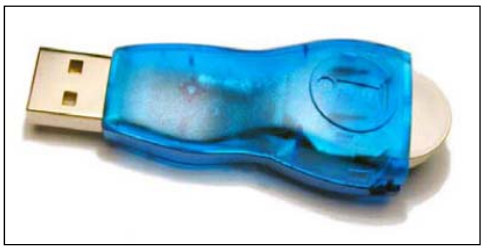

Abb. 11: DS2490 USB Lesegerät

2.6 Digitalkamera

Um beim Austeilen der iButtons Fotos von deren Standort zu machen, suchten wir eine Digitalkamera, welche man vom Computer mit gPhoto fernsteuern konnte. Damit sollte es möglich sein, ein Foto direkt vom Computer auszulösen, das gemachte Bild auf die Harddisk herunter zu laden und automatisch in der Datenbank abzulegen.

Von einer Liste mit unterstützten Kameras wurde von den Geologen die Nikon Coolpix L1 ausgewählt und uns zur Verfügung gestellt.

Abb. 12: Nikon Coolpix L1

3 Verwendete Softwarekomponenten

3.1 SQLite

Für die Speicherung der Datensätze haben wir uns für SQLite entschieden. Der Hauptgrund lag vor allem darin, dass die Applikation zuerst für das Openmoko FreeRunner gedacht war und SQLite bereits bei Betriebssystemen wie Symbian OS, Android und beim iPhone eingesetzt wird. Es existieren auch SQLite Bibliotheken für das Openmoko.
SQLite zeichnet sich dadurch aus, dass es sich um eine textbasierte Datenbank handelt, die in einer einzigen Datei gespeichert werden kann. Um die Datenbank zu verwenden muss auf dem betreffenden System einzig SQLite installiert sein und die gewünschte Datenbankdatei kann geöffnet werden. Dadurch ist SQLite sehr portabel und einfach zu verwenden (12).

SQLite eignet sich besonders für:

- Eingebettete Systeme und Anwendungen. Eine SQLite Datenbank benötigt beinahe keine Administration.
- Als Ersatz für das Speichern der Daten in einer Datei (bietet nur eine relationale Datenbank)

SQLite eignet sich nicht für:

- Client/Server Anwendungen
- Grosse Datensätze: grösser 2 tebibytes
- Gleichzeitiger Zugriff

So stellt SQLite für unsere Applikation eine gute Funktionalität zur Verfügung, da wir einzig eine kleine Menge an Daten verwalten müssen und es kein Problem darstellt gleichzeitige Zugriffe zu vermeiden.

3.2 QT

Das Openmoko Mobiltelefon stellt QT Embedded und Gtk+ als Werkzeuge zur Erstellung grafischer Oberflächen zur Verfügung. Beide Frameworks bieten eine plattformübergreifende Einsatzmöglichkeit. Die Wahl fiel hier hauptsächlich wegen der übersichtlichen API zugunsten von QT aus (13).

Ein weiterer Vorteil von QT ist die grosse Palette an Klassen und Funktionen. QT bietet sehr viele Funktionen und Klassen zur Verwaltung von Datenbanken, Dateien, Zeiten usw. Zudem sind praktisch zu allen benötigten Funktionalitäten unseres Programms Beispiele vorhanden, die nach unseren Bedürfnissen adaptiert werden können. Ein Beispiel dafür ist die Richtungsanzeige.

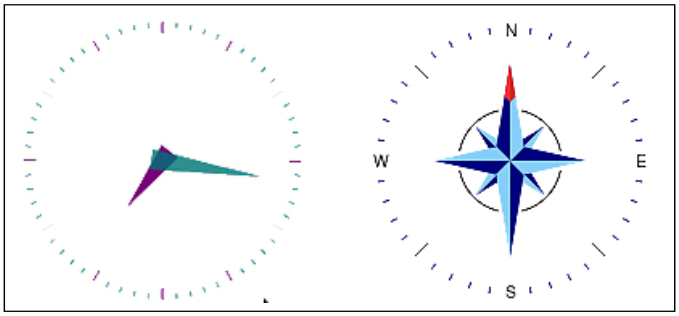

Abb. 13: AnalogClock Beispiel aus der QT API und Richtungsanzeige von iAssist

Bei iAssist werden wann immer möglich Funktionen und Klassen aus der QT-API verwendet.

3.3 libgphoto2

Libgphoto2 ist eine Bibliothek, welche den Zugriff auf über 1100 Digitalkameras ermöglicht (14). Dabei hängt es vom Kameramodell ab, welche Funktionen über libgphoto2 verfügbar sind. Viele der 1100 Kameras werden zwar unterstützt, können aber beispielsweise nicht vom PC aus ausgelöst werden, was sie für unseren Zweck unbrauchbar macht.

Die Nikon Coolpix L1 unterstützt im PTP-Modus (Picture Transfer Protocol) das ferngesteuerte Auslösen eines Bildes, sowie die Übertragung auf den PC und das Löschen des Bildes auf der Kamera.

3.4 GPS Daemon

GPSD ist ein Daemon für Linux, welcher die Verbindung zum GPS Gerät regelt und alle Informationen über den TCP Port 2947 zur Verfügung stellt. Die Informationen können dort entweder direkt mit Telnet oder über die libgps Bibliothek ausgelesen werden. Es werden verschiedene Anfragen unterstützt, welche alle durch einzelne Buchstaben repräsentiert werden. In unserem Programm verwenden wir lediglich den Query-String „o“, welcher alle verfügbaren Daten vom GPS Gerät liest (15).

3.5 Maxim Software Ressourcen

Für die Kommunikation mit den iButton Sensoren wurde von uns der Source Code des 1-Wire Public Domain Kits von Maxim verwendet. Dabei handelt es sich um einen Software Development Kit für die 1-Wire Produkte. Die von uns verwendete SDK der Version 3.10 für Linux enthält eine komplette 1-Wire API geschrieben in C.
Für den iButton DS1922L gibt es dazu das Applikationsbeispiel „humalog“, mit welchem wir die ersten Tests mit den Sensoren durchführten und uns auch beim Schreiben der ButtonIO Klasse als Referenz diente. Die ButtonIO Klasse benötigt folgende Dateien des Software Development Kits:

crcutil.c	libusbds2490.h	mbee77.c	mbnvcrc.h	mbscrex.c	ownet.h
findtype.c	libusblnk.c	mbee77.h	mbscr.c	mbscrex.h	pw77.c
findtype.h	libusbnet.c	mbeprom.c	mbscr.h	mbscrx77.c	pw77.h
humutil.c	libusbses.c	mbeprom.h	mbscrcrc.c	mbscrx77.h	
humutil.h	libusbtran.c	mbnv.c	mbscrcrc.h	mbsha.c	
ioutil.c	mbappreg.c	mbnv.h	mbscree.c	mbsha.h	
libusbds2490.c	mbappreg.h	mbnvcrc.c	mbscree.h	owerr.c	

Um die ButtonIO Klasse zu schreiben, wurden von uns die Dateien so angepasst, dass sie von einem C++ Compiler verarbeitet werden können, und stimmen demnach nicht mehr mit den Dateien von Maxim überein. Funktionelle Veränderungen wurden in den Dateien „humutil.c“, „humutil.h“ und „libusbses.c“ vorgenommen. So wurden die Registerdefinitionen des iButtons von der Datei „humutil.c“ in die Header-Datei „humutil.h“ kopiert, damit diese Definitionen auch in der ButtonIO Klasse verwendet werden konnten. Ausserdem wurde die Funktion zum Auslesen der Daten aus dem iButton in die ButtonIO Klasse ausgelagert um diese nach unseren Wünschen anzupassen. Um herauszufinden, ob der USB Port geöffnet ist, wurde in der Datei „libusbses.c“ eine entsprechende Funktion hinzugefügt. Bei Tests mit der Auslesefunktion ist uns einen Fehler in der Maxim-Software aufgefallen. Machte der iButton Aufzeichnungen mit 16-Bit Auflösung, so konnten immer nur 2048 Messwerte ausgelesen werden. Die restlichen Daten wurden mit den bereits gelesenen Daten ersetzt. Dieser Fehler konnte von uns behoben werden.

4 Beschreibung der Klassen

4.1 Programmstart und Hauptfenster

Main

Main.cpp

In der *main()* Funktion wird überprüft, ob die Ordnerstruktur des Programms korrekt ist und die nötigen Komponenten (gpsd, SQLite Treiber) auf dem Computer installiert sind. Danach wird das Hauptfenster aufgerufen.

iAssist - Das Hauptfenster

IAssist.cpp, IAssist.h, IAssist.ui

Das Hauptfenster wird durch die User Interface Klasse IAssist gebildet. Das Hauptfenster wird direkt von *main()* aufgerufen und zeigt dem Benutzer eine Auswahlmöglichkeit der verschiedenen Funktionalitäten von iAssist.

4.2 Programmieren der iButtons vor Ort

AutoDistributeUI

AutoDistributeUI.cpp, AutoDistributeUI.h, AutoDistributeUI.ui

Die AutoDistributeUI Klasse ist die Schnittstelle zwischen dem Benutzer und dem AutoDistributeThread.

Über diese Klasse wird dem AutoDistributeThread mitgeteilt, falls der Benutzer auf dem GUI den aktuellen Footprint ändert. AutoDistributeUI bietet ebenfalls „Slots“ um Rückmeldungen aus dem AutoDistributeThread auf dem GUI anzuzeigen.

Neben den Möglichkeiten das Austeilen vor Ort zu starten/stoppen und den aktuellen Footprint zu ändern, kann der Benutzer über das GUI auch ein Foto erstellen und dem aktuellen iButton oder dem aktuellen Footprint zuweisen. Will der Benutzer ein Foto löschen, kann er das aktuell angezeigte Foto mit der dafür vorgesehenen Taste auf dem GUI löschen.

AutoDistributeThread

AutoDistributeThread.cpp, AutoDistributeThread.h

Die AutoDistributeThread Klasse koordiniert alle Aufgaben, welche beim Austeilen der iButtons vor Ort anfallen.

Gestartet wird die Klasse von der AutoDistributeUI Klasse als Thread. Der Thread überprüft kontinuierlich, ob ein iButton im Lesegerät ist. Wird ein iButton gefunden und entspricht dieser nicht dem zuletzt gefundenen, so wird er automatisch programmiert. Die Parameter werden aus der „mission_start_parameter.txt“ Datei und vom aufrufenden AutoDistributeUI GUI übernommen.

Neben den Daten, welche direkt beim Programmieren zum jeweiligen iButton aus der „mission_start_parameter.txt“ Datei übernommen werden (z.B. die Startzeit für die Messung) und den Daten, welche vom Benutzer im GUI ausgewählt werden (Footprint zu dem der iButton gehört), können nach dem Programmieren zum jeweiligen iButton und dessen Footprint noch Fotos angefügt werden.

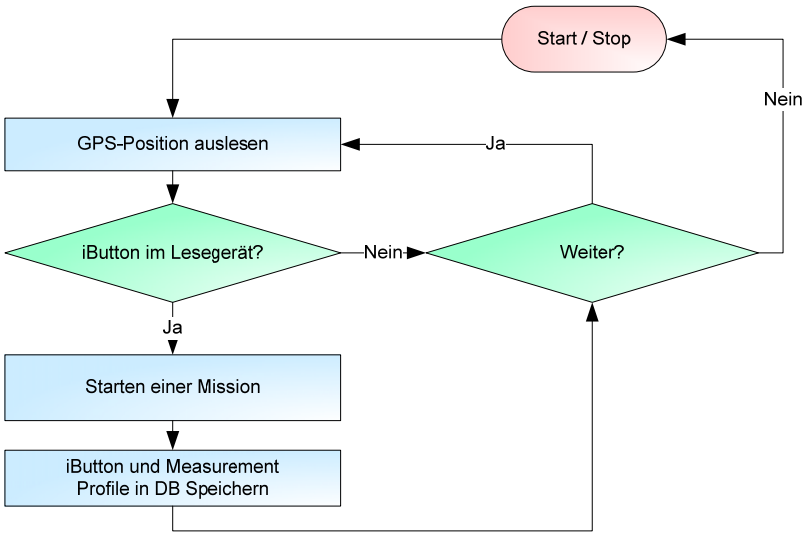

Abb. 14: AutoDistribute Thread

4.3 Programmieren der iButtons

AutoProgramUI

AutoProgramUI.cpp, AutoProgramUI.h, AutoProgramUI.ui

Die AutoProgramUI Klasse ist die Schnittstelle zwischen AutoProgramThread und Benutzer.

Sie informiert sowohl den Benutzer über die Rückmeldungen des AutoProgramThreads, als auch den Thread selbst über die vom Benutzer gewählten Parameter (FootprintID).

AutoProgramThread

AutoProgramThread.cpp, AutoProgramThread.h

Die AutoProgramThread Klasse koordiniert alle Aufgaben, welche beim Programmieren der iButtons anfallen.

Die AutoProgramUI Klasse ruft diese Klasse als Thread auf. Der Thread überprüft in seinem Ablauf kontinuierlich, ob ein iButton im Lesegerät vorhanden ist. Wenn ein iButton erkannt wird und seine ID nicht der zuletzt erkannten ID entspricht, wird dieser automatisch mit den Daten aus der „mission_start_parameter.txt“ Datei programmiert.

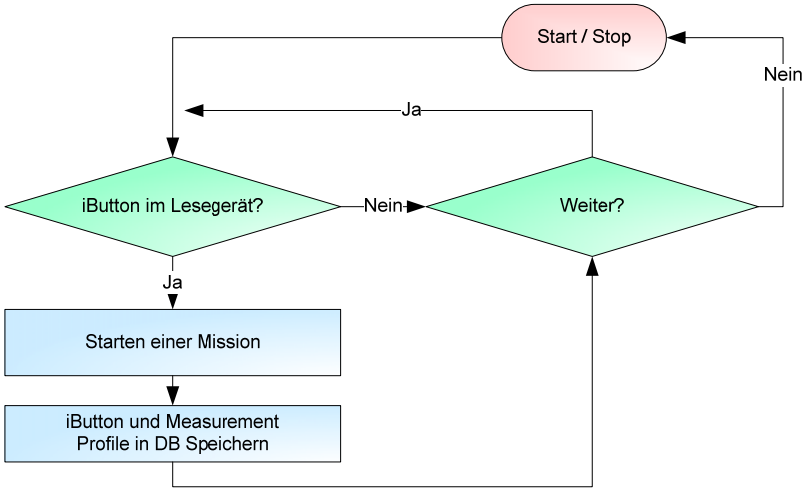

Abb. 15: AutoProgram Thread

4.4 Einsammeln der iButtons

CollectUI

CollectUI.cpp, CollectUI.h, CollectUI.ui

Die Klasse zeigt dem Benutzer ein Fenster in welchem er das Einsammeln der iButtons starten kann. Es beinhaltet die drei Widgets zur Richtungsangabe (DirectionWidget), zur Distanzangabe (DistanceWidget) und für die Anzeige von Bildern (PhotoDisplayWidget). CollectUI wird verwendet, um dem Benutzer alle nötigen Informationen zu seiner jeweiligen Auswahl anzuzeigen und das Starten und Stoppen des Threads (CollectThread), welcher zum Einsammeln der iButtons aufgerufen wird, zu steuern.

CollectThread

CollectThread.cpp, CollectThread.h

Diese Klasse ist als Thread implementiert, der das Einsammeln der iButtons steuert. Aufgerufen durch Start im UI aktualisiert er dauernd die GPS Daten und wartet bis ein iButton in das Lesegerät gehalten wird. Er liest diesen iButton, aktualisiert die Datenbank und programmiert diesen iButton, sofern vom Benutzer gewünscht, erneut.

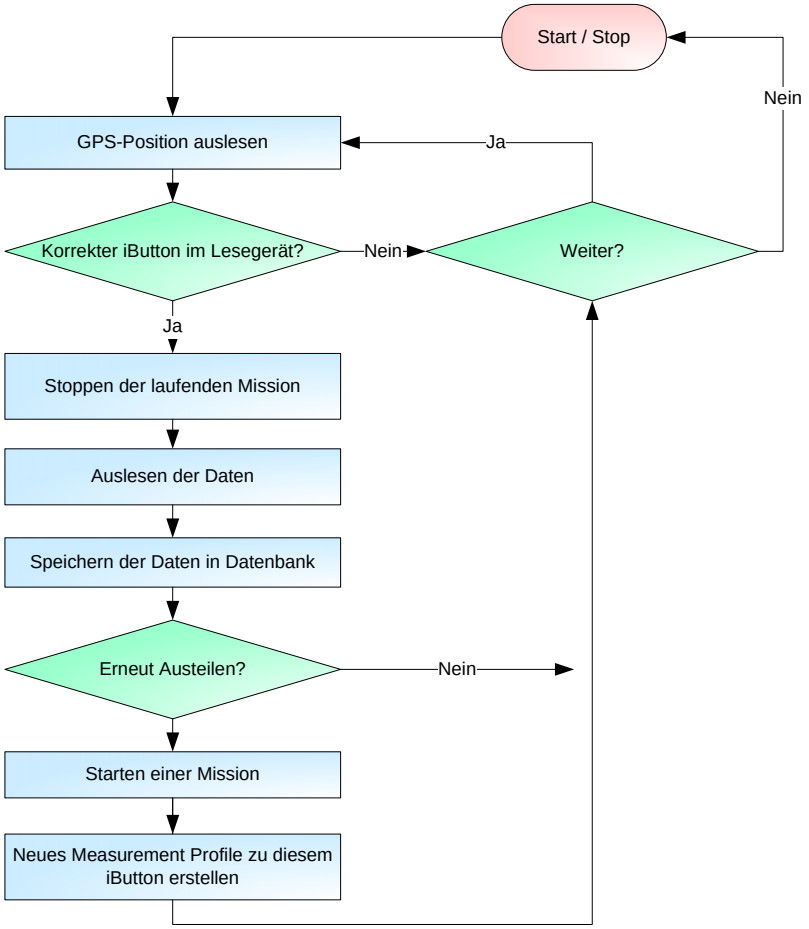

Abb. 16: CollectThread

4.5 Importieren von Daten

CSVImportUI

CSVImportUI.cpp, CSVImportUI.h, CSVImportUI.ui

Diese Klasse stellt die Oberfläche zur Verfügung, von welcher aus verschiedene Datensätze importiert werden können. Es wird zwischen Importieren von iButton-, Footprint- und Fotodatensätzen unterschieden. Sofern der Benutzer auf Importieren klickt, wird der entsprechende Thread gestartet.

CSVImportPhotoThread

CSVImportPhotoThread.cpp, CSVImportPhotoThread.h

Da das Importieren der Fotos einige Zeit benötigt, wird dieser Task von der Benutzeroberfläche als Thread gestartet. CSVImportPhotoThread startet das Importieren der Fotos und gibt Statusmeldungen an die Benutzeroberfläche zurück.

CSVImportButtonThread

CSVImportButtonThread.cpp, CSVImportButtonThread.h

Da das Importieren der iButtons einige Zeit benötigt, wird dieser Task von der Benutzeroberfläche als Thread gestartet. CSVImportButtonThread startet das Importieren der iButton-Daten und gibt Statusmeldungen an die Benutzeroberfläche zurück.

CSVImportAreaThread

CSVImportAreaThread.cpp, CSVImportAreaThread.h

Da das Importieren der Footprints einige Zeit benötigt, wird dieser Task von der Benutzeroberfläche als Thread gestartet. CSVImportAreaThread startet das Importieren der Footprint-Daten und gibt Statusmeldungen an die Benutzeroberfläche zurück.

CSVFileReader

CSVFileReader.cpp, CSVFileReader.h

CSVFileReader bietet im Wesentlichen drei Funktionalitäten:

- Importieren eines iButton Datensatzes:
  Sämtliche Daten werden aus der Datei gelesen und falls noch nicht in der Datenbank vorhanden als neuer iButton hinzugefügt. Falls ein iButton bereits vorhanden ist, werden die Werte in der Datenbank durch die Werte in der CSV-Datei ersetzt.
- Importieren eines Footprint Datensatzes:
  Sämtliche Daten werden aus der Datei gelesen und falls noch nicht in der Datenbanktabelle ‚Footprints' vorhanden als neuen Footprint hinzugefügt. Falls ein bestimmter Footprint bereits in der Datenbank existiert, werden die Werte von der CSV-Datei übernommen.
- Importieren der Fotos:
  In einer CSV-Datei ist jedem Dateiname eines Fotos ein iButton oder ein Footprint zugeordnet. Anhand dieser Zuordnung, werden die Fotos in das iAssist Dateisystem eingefügt und der dazugehörige Dateiname in der Datenbank gespeichert.

4.6 Löschen von Datensätzen

DBManagementUI

DBManagementUI.cpp, DBManagementUI.h

Die DBManagementUI Klasse ermöglicht das Löschen von Daten aus der Datenbank. Es werden dabei drei verschiedene Funktionen angeboten:

- Löschen eines Footprints. Dabei werden alle iButtons zu dem gewählten Footprint, mit sämtlichen zugehörigen Daten, wie Fotos und Messungen gelöscht.
- Löschen eines iButtons. Dabei wird der ausgewählte iButton mit sämtlichen zugehörigen Daten und Fotos aus der Datenbank gelöscht.
- Löschen eines Fotos. Es wird das ausgewählte Foto aus der Datenbank gelöscht.

Um zu vermeiden, dass der Benutzer versehentlich Daten aus der Datenbank entfernt, muss er noch einmal bestätigen, dass er die ausgewählten Daten wirklich löschen will.

4.7 Bearbeiten des Mission Parameter File

MissionParameterUI

MissionParameterUI.cpp, MissionParameterUI.h

Die MissionParameterUI Klasse ermöglicht es die „mission_start_parameter.txt“ Datei über eine Benutzeroberfläche zu bearbeiten.

Die Werte werden beim Öffnen des Dialogs aus der Datei gelesen und über die Taste „Save“ wieder in diese gespeichert. Am GUI können die Werte bequem über Kontrollkästchen und Zahlenfelder bearbeitet werden, wodurch ungültige Eingaben automatisch vermieden werden.

4.8 Generelle Klassen

4.8.1 Datenbank

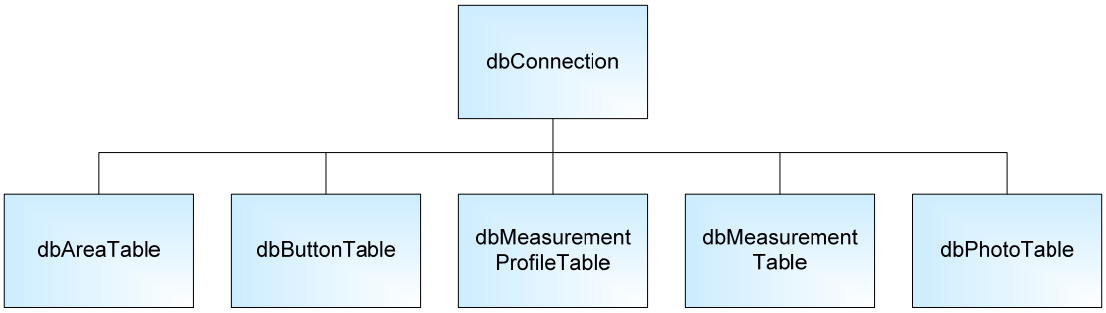

Abb. 17: Vererbung der Datenbankklassen

DBConnection

DBConnection.cpp, DBConnection.h

DBConnection stellt die grundlegende Datenbank Funktionalität zur Verfügung. Funktionen wie Erstellen, Öffnen und Schliessen sind darin implementiert. Zudem kann man Backups der iAssist Datenbank erstellen und notfalls ein solches Backup wieder herstellen.

Erbende Klassen

DBAreaTable.cpp, DBAreaTable.h, DBButtonTable.cpp, DBButtonTable.h, DBMeasurementProfileTable.cpp, DBMeasurementProfileTable.h, DBMeasurementTable.cpp, DBMeasurementTable.h, DBPhotoTable.cpp, DBPhotoTable.h

Die von DBConnection erbenden Klassen verwalten jeweils eine Tabelle in der Datenbank. Sie stellen jeweils die benötigten Funktionen zum Auslesen oder Bearbeiten der Tabellen zur Verfügung.

4.8.2 Grafische Widgets

Richtungsangabe

DirectionWidget.cpp, DirectionWidget.h, DirectionWidget.ui

Abb. 18: Widget zur Richtungsangabe

Das Widget gibt einen Winkel grafisch aus. Es wird benötigt um dem Benutzer die Richtung zu einem iButton zu weisen. Da der verwendete GPS-Empfänger über keinen digitalen Kompass verfügt, wird die Richtung nur relativ zu Nord angegeben und der Benutzer kann die einzuschlagende Richtung ablesen.

Distanzangabe

DistanceWidget.cpp, DistanceWidget.h

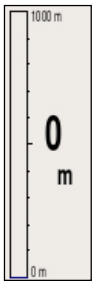

Abb. 19: Widget zur Distanzangabe

Dieses Widget stellt eine Distanz grafisch dar. Es enthält einen Distanzbalken, der abnimmt sofern man sich dem Ziel nähert.

Fotoanzeige

PhotoDisplayWidget.cpp, PhotoDisplayWidget.h

Die Fotoanzeige ist ein eigenständiges Widget. Es kann einzelne Fotos anzeigen oder auch eine Liste mit Fotos. Dabei ist es möglich die Fotos zu vergrössern oder zu verkleinern. Sofern eine Liste von Fotos angezeigt wird, kann zusätzlich durch die Liste navigiert werden.

4.8.3 Kommunikation mit iButton Sensoren

ButtonIO

ButtonIO.h, ButtonIO.cpp

Die Klasse ButtonIO bietet alle Funktionalitäten zur Kommunikation mit den iButton Sensoren. Die Grundlage dieser Klasse bilden die Bibliotheken des 1-Wire Public Domain Kit Version 3.10 von Maxim. Die Klasse kann in fünf verschiedene Bereiche eingeteilt werden:

- Messreihe starten und stoppen. Mit diesen Funktionen ist es möglich einen iButton gemäss den gewählten Einstellungen zu programmieren und eine Messreihe zu starten und auch wieder zu beenden. Folgende Einstellungen sind möglich:
  - Grösse des Messintervalls.
  - Auflösung der Temperaturmessung.
  - Startverzögerung in Minuten.
  - Startverzögerung in Tagen mit Start der Messreihe um Mitternacht.
  - Überschreiben der Messungen, bei vollem Speicher.
- iButton erkennen und USB Port öffnen oder schliessen. Diese Funktionen ermöglichen den USB-Port zu öffnen, das USB Lesegerät zu erkennen um eine Kommunikation mit dem iButton aufzubauen oder diese wieder ordnungsgemäss zu beenden. Weiter werden Funktionen zur Verfügung gestellt um einen iButton im Lesegerät zu erkennen und dessen einzigartige ID auszulesen.
- Zeitstempel Funktionen. Diese Funktionen ermöglichen das Auslesen der RTC des iButtons und des Zeitstempels des Betriebssystems. Dadurch wird die Korrektur der Zeitverschiebung zwischen iButton und Computer ermöglicht. Ausserdem kann auch der Startpunkt der Messreihe ermittelt werden.
- Auslesen der Daten. Diese Funktion ermöglicht das Herunterladen der Messdaten vom iButton. Dabei kann zusätzlich eine automatische Softwarekorrektur gewählt werden.

4.8.4 Kommunikation mit dem GPS Empfänger

IGPS

IGPS.cpp, IGPS,h

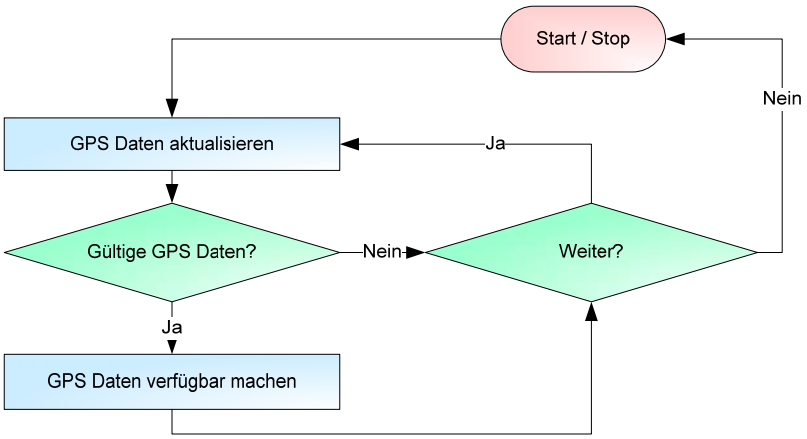

Abb. 20: IGPS Thread

Die IGPS Klasse stellt die Verbindung zum GPS Daemon (GPSD) her und dient als Vermittler zwischen dem GPS Gerät und dem iAssist Programm.
Die Idee hinter dieser Klasse ist, GPS Daten vom GPSD kontinuierlich im Hintergrund auszulesen. Sie stellt eine Schnittstelle zur Verfügung, welche es dem iAssist Programm ermöglicht die aktuellen GPS Daten in einer einfachen Weise zu erhalten. Die Daten werden als GPSData Objekt übergeben.

4.8.5 Kommunikation mit der Kamera

ICam

ICam.cpp, ICam.h

Die ICam Klasse stellt eine Verbindung zur Digitalkamera her. Sie kann ein Foto auslösen und dieses an den Rechner übertragen. Um die Verbindung zur Kamera herzustellen nutzt sie die gPhoto2 Bibliothek, welche für diese Aufgaben bereits Funktionen zu Verfügung stellt und in ICam für unsere Zwecke angepasst wird.

4.8.6 Sonstige Klassen

Log

Log.cpp, Log.h

Die Log Klasse verfügt über drei statische Befehle. Ein Logfile kann neu erzeugt werden, was bei iAssist zum Ausführungszeitpunkt geschieht, um nicht beliebig lange Logs zu generieren. Es wird unterschieden zwischen allgemeinen Einträgen und Fehlern. Die Einträge informieren über den Ablauf des Programms und geben detailliertere Beschreibungen zu Fehlern aus.

SystemCommand

SystemCommand.cpp, SystemCommand.h

Diese Klasse bietet eine statische Funktion, die ein Systembefehl ausführt und danach die Rückgabe des Systems zur Verfügung stellt.

UserDialog

UserDialog.cpp, UserDialog.h

Diese Klasse bietet statische Funktionen, die es ermöglichen dem Benutzer ein Fenster anzuzeigen, um ihn über ein Ereignis zu informieren. Ebenso ist es möglich dem Benutzer eine Frage zu stellen. Diese Funktion kann nur von einem GUI-Thread aufgerufen werden, da QT es nicht zulässt, dass andere Threads als der GUI-Thread grafische Elemente verwaltet.

GPSMath

GPSMath.cpp, GPSMath.h

Die GPSMath Klasse stellt statische Funktionen zum Rechnen mit GPS Daten zu Verfügung.
Es werden vier Funktionen angeboten. Es kann aus zwei GPSData Objekten die Entfernung oder die Richtung berechnet werden. Zudem kann aus den Double-Werten für Längen-, Breitengrad und Höhe eine formatierte Ausgabe generiert werden.

Area

Area.cpp, Area.h

Die Area Klasse enthält alle Daten, die über einen Footprint gespeichert werden.

ButtonData

ButtonData.cpp, ButtonData.h

Die ButtonData Klasse enthält alle Daten, die über einen iButton gespeichert werden.

GPSData

GPSData.cpp, GPSData.h

Die GPSData Klasse enthält alle Daten, die über eine GPS Koordinate gespeichert werden.

Photo

Photo.cpp, Photo.h

Die Foto Klasse enthält alle Daten, die über ein Foto gespeichert werden.

5 Test

5.1 Funktionalität der Software

1	Nur die ausführbare iAssist Datei und den Ordner ‚ico' in einen leeren Ordner verschieben. iAssist lässt sich starten und erstellt die fehlenden Ordner ‚img' und ‚log'.	✓
2	Überprüfen ob Änderungen die in „Edit Mission Parameter" gemacht werden, richtig auf die Konfiguration übertragen werden.	✓

5.2 Sensoren stationär programmieren

5.2.1 Programmieren der iButton Sensoren

1	2 Sensoren mit Footprint „AA" und Messparameter 1 programmieren. *Messparameter 1:* automatic temperature calibration: 0 sample rate: 7 start delay: 3 enable rollover: 1 synchronize time with computer: 1 high temperature resolution: 1 start at 00:00: 0 day delay: 0	✓
2	2 Sensoren mit Footprint „ZC" und Messparameter 2 programmieren. *Messparameter 2:* automatic temperature calibration: 1 sample rate: 30 start delay: 0 enable rollover: 0 synchronize time with computer: 0 high temperature resolution: 0 start at 00:00: 1 day delay: 6	✓
3	Daten werden richtig in Datenbank eingetragen. iButton: Button Nr., Button ID, Rest leer Footprint: Footprint ID, Rest leer MeasurementProfile: MeasProfileID, ButtonNr., SessionNr, SamplingRate, SamplingStartTimeStamp, Rest leer	✓
4	Register werden im iButton richtig gesetzt. Überprüfen mit 'OneWireViewer' Software von Maxim.	✓

5.2.2 Importieren externer Daten

5	iButton csv Datei mit iButton Daten ergänzen und importieren. MeasurementProfile: Werte korrekt iButton: Werte korrekt.	✓
6	Footprint csv Datei mit Footprint Daten ergänzen und importieren. Footprint: Werte korrekt.	✓
7	Foto csv Datei erstellen und Fotos importieren (2 Fotos pro iButton und Footprint). Tabelle Photo: Werte korrekt Dateisystem: Alle Dateien wurden korrekt umbenannt und sind vorhanden	✓

5.2.3 Einsammeln der Sensoren

8	Redistribute aktivieren und iButton des Footprint „AA“ einsammeln. Datenbank: Die zugehörigen Measurement Profile wurden korrekt ergänzt. Die Messdaten wurden in die Datenbank eingetragen. Es wurde ein neues Measurement Profile erstellt mit dem Messprofil 2.	✓
9	Redistribute deaktivieren und iButton des Footprint „ZC“ einsammeln. Datenbank: Die zugehörigen Measurement Profile wurden korrekt ergänzt. Die Messdaten wurden in die Datenbank eingetragen. Es wurde kein neues Measurement Profile erstellt.	✓
10	Register der iButton von „AA“ überprüfen. Die iButton von „AA“ wurden mit den gewählten Parameter programmiert.	✓
11	Register der iButton von „ZC“ überprüfen. Die iButton von „ZC“ wurden nicht mehr programmiert.	✓

5.2.4 Daten entfernen

12	Einzelnes Foto zu iButton „AA001“ löschen. Foto wurde in Datenbank entfernt. Foto wurde von Dateisystem entfernt.	✓
13	iButton „AA001“ löschen. Alle Measurement Profile, Measurements, Fotos und iButton Daten wurden aus Datenbank entfernt. Fotos wurden von Dateisystem entfernt.	✓
14	Footprint „ZC“ löschen. Alle Measurement Profile, Measurements, Fotos, iButton und Footprint Daten wurden aus der Datenbank entfernt. Fotos wurden von Dateisystem entfernt.	✓

5.3 iButton vor Ort programmieren

5.3.1 Programmieren der iButton Sensoren

1	Footprint „QV“ auswählen und Footprint Foto machen. Footprint und Foto sind in Datenbank eingetragen.	✓
2	iButton Foto machen, darf noch nicht funktionieren.	✓
3	iButton Programmieren mit Messparameter 1 *Messparameter 1:* automatic temperature calibration: 0 sample rate: 7 start delay: 3 enable rollover: 1 synchronize time with computer: 1 high temperature resolution: 1 start at 00:00: 0 day delay: 0	✓
4	Daten werden richtig in Datenbank eingetragen: iButton: Button Nr., Button ID, GPS Position, Rest leer Footprint: Footprint ID, Rest leer MeasurementProfile: MeasProfileID, ButtonNr, SessionNr, SamplingRate, SamplingStartTimeStamp, DistributionTimeStamp, Rest leer	✓
5	Foto zu iButton machen. Foto wird in Datenbank eingetragen und ist im Dateisystem vorhanden.	✓
6	iButton Foto wieder löschen. Foto wurde aus Datenbank entfernt und ist vom Dateisystem gelöscht.	✓
7	Footprint Foto wieder löschen. Foto wurde aus Datenbank entfernt und ist vom Dateisystem gelöscht.	✓
8	2 Footprint Fotos und 2 iButton Fotos erstellen. Fotos wurden in Datenbank korrekt eingetragen und sind im Dateisystem vorhanden.	✓
9	3 weiter iButton programmieren. 3 verschiedene Footprints wählen und je 2 Footprint Fotos und 2 iButton Fotos machen. Fotos sind im Dateisystem vorhanden. Fotos sind in der Datenbank korrekt eingetragen. Footprints sind korrekt in Datenbank. iButton sind korrekt in Datenbank. Measurement Profile sind korrekt in Datenbank.	✓

5.3.2 Import von zusätzlichen Daten

10	Import von Foto .csv Datei. Die Daten werden in die Datenbank eingetragen und die Dateien korrekt im Dateisystem gespeichert (kein Überschreiben alter Daten).	✓
11	Import von Footprint .csv Datei (teilweise ausgefüllt). Eingetragene Werte werden in die Datenbank eingetragen. Sofern keine Werte eingetragen wurden, werden alte Daten in Datenbank belassen.	✓
12	Import von iButton .csv Datei (teilweise ausgefüllt). Eingetragene Werte werden in die Datenbank eingetragen. Sofern in der .csv Datei keine Daten eingetragen wurden, werden die alten Daten in der Datenbank belassen.	✓
13	Import fehlerhafter .csv Dateien wird erkannt.	✓

6 Bedienungsanleitung

6.1 Installation

Um iAssist auszuführen, müssen folgende Pakete auf dem Computer installiert sein:

- gpsd
- gphoto2
- libqt4-sql-sqlite

Das iButton Lesegerät wird von Linux möglicherweise mit dem falschen Treiber geladen und muss deshalb auf die „Blacklist“ gesetzt werden. Dies kann folgendermassen erreicht werden (unter Ubuntu):

Einmaliges entfernen:

1. Konsole öffnen.
2. Eingeben: *sudo rmmod ds2490*

Dauerhaft auf die Blacklist setzen:

1. Konsole öffnen.
2. Eingeben: *sudo gedit /etc/modprobe.d/blacklist*
3. Folgende Zeile hinzufügen: *blacklist ds2490*
4. Speichern.

Da iAssist libusb verwendet, muss iAssist als „root“ gestartet werden. Alternativ kann man die Zugriffsrechte für die USB-Schnittstelle folgendermassen anpassen (unter Ubuntu):

1. Konsole öffnen.
2. Eingeben: *sudo gedit /etc/udev/rules.d/40-basic-permissions.rules*
3. Das Dokument folgendermassen verändern:
   #USB devices (usbfs replacement)
   SUBSYSTEM==”usb”, ENV{DEVTYPE}==”usb_device”, MODE=”0666”
   SUBSYSTEM==”usb_device”, MODE=”0666”
4. Speichern und schliessen.
5. Eingeben: *sudo /etc/init.d/udev reload*
6. Gegebenenfalls Computer neu starten.

Mit dieser Änderung kann iAssist ohne root Rechte ausgeführt werden. Es empfiehlt sich die ausführbare Datei in einen leeren Ordner zu kopieren. Alle iAssist Daten werden dann in diesem Ordner erstellt.

6.2 Das Hauptfenster

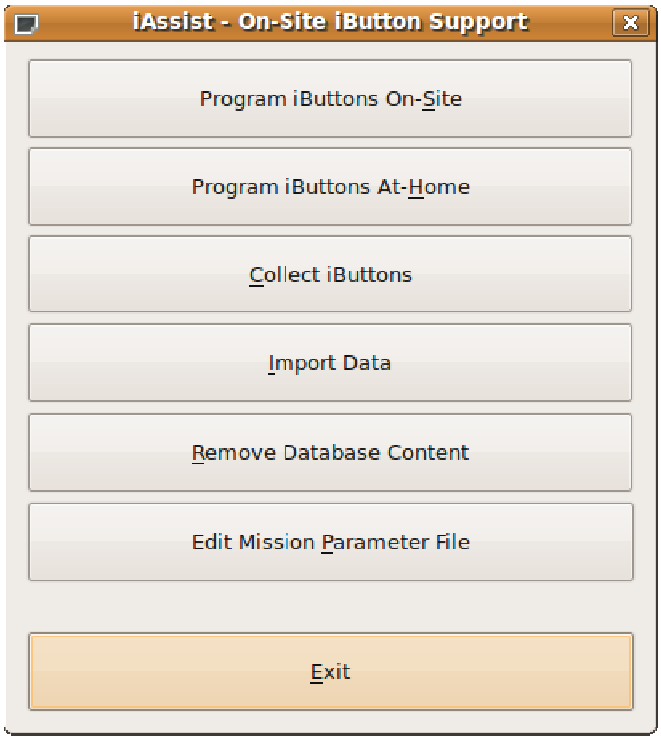

Abb. 21: Hauptfenster

Das Hauptfenster bietet einen Überblick über alle Funktionen des Programms. Eine Funktion kann durch Drücken der entsprechenden Schaltfläche gestartet werden. Es kann nur eine Funktion gleichzeitig geöffnet sein.

6.3 Programmieren vor Ort

Mit folgender Benutzeroberfläche können die iButtons vor Ort programmiert werden.

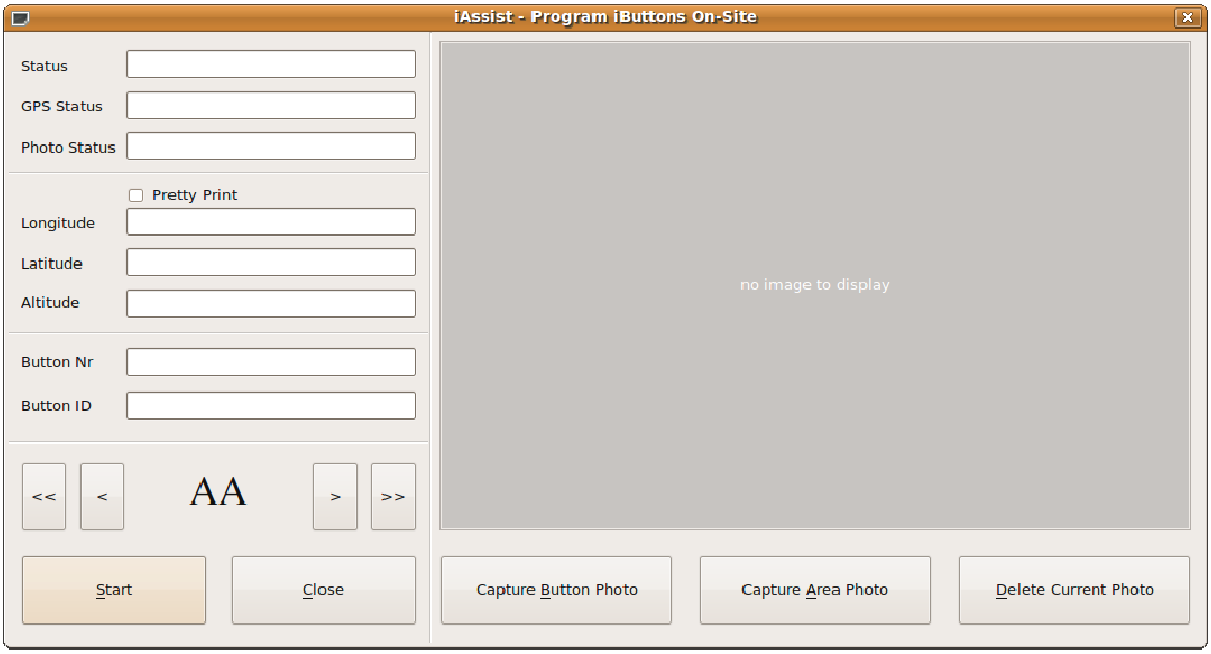

Abb. 22: Programmierung vor Ort

Das Programm unterstützt Sie bei folgenden Aufgaben:

- Programmieren der iButtons
- Speichern der iButton ID
- Speichern der iButton Konfiguration
- Speichern der aktuellen GPS Position
- Speichern der aktuellen Zeit
- (Optional) Erstellen und Speichern von Fotos zu den iButtons und Footprints

Vorgehensweise:

1. Drücken Sie die Schaltfläche “Start” um dem Programm mitzuteilen, dass nun iButtons programmiert werden sollen.
2. Das Programm zeigt nun im Statusfeld die Mitteilung Please insert NEW iButton und im GPS Statusfeld die GPS Signalqualität an. Sollte im Statusfeld die Mitteilung ERROR: No reader connected! oder im GPS Statusfeld die Mitteilung No Signal erscheinen, obwohl das Lesegerät und der GPS Empfänger angeschlossen sind, lesen Sie bitte im Kapitel Fehlerbehebung weiter.
3. Wählen Sie anhand der Pfeiltasten die gewünschte Footprint ID aus. Durch betätigen der Schaltfläche “<” bzw. “>” schalten Sie um eine ID zurück oder vor. Durch betätigen der Schaltfläche “<<” bzw. “>>” schalten Sie 26 IDs zurück oder vor.

4. Legen Sie einen iButton in das Lesegerät. Entfernen Sie den iButton erst, wenn Ihnen das Programm mitteilt, dass der iButton programmiert wurde.
5. Wenn der iButton erfolgreich programmiert wurde, wird die Mitteilung Programming iButton DONE im Statusfeld angezeigt. Das korrekte Programmieren des iButtons wird Ihnen zusätzlich akustisch durch einen einzelnen Piepton bestätigt. Sie können den iButton nun entfernen. Er wurde mit den im Dialog “Edit Mission Parameter File” gewählten Einstellungen programmiert. Sollte eine Fehlermeldung angezeigt werden, lesen Sie bitte im Kapitel Fehlerbehebung weiter. Falls die Mitteilung Check Mission Paremter file erscheint, so ist entweder keine Datei mit den Missionsparametern vorhanden oder der Inhalt ist korrupt. Stellen Sie im Dialog “Edit Mission Parameter File” die Parameter erneut ein und drücken Sie auf “Save”. Beginnen Sie anschliessend wieder bei Punkt 1.
6. (Optional) Sie können nun zum soeben programmierten iButton ein Foto speichern. Schliessen Sie hierfür die Kamera an und richten Sie diese auf den iButton, den Sie fotografieren wollen. Betätigen Sie anschliessend die Schaltfläche “Capture iButton Photo”. Warten Sie bis im Photo Status Feld die Meldung Picture taken angezeigt wird. Sollte eine Fehlermeldung anzeiget werden, lesen Sie bitte im Kapitel Fehlerbehebung weiter.
7. (Optional) Sie können nun zum entsprechenden Footprint ein Foto Speichern. Schliessen Sie hierfür die Kamera an und richten Sie diese auf das Gebiet, das sie fotografieren wollen. Betätigen Sie anschliessend die Schaltfläche “Capture Footprint Photo”. Warten Sie bis im Photo Status Feld die Meldung Picture taken angezeigt wird. Sollte eine Fehlermeldung anzeiget werden, lesen Sie bitte im Kapitel Fehlerbehebung weiter.
8. Wiederholen Sie den Vorgang bei Punkt 2. oder drücken Sie die Schaltfläche “Stop” und anschliessend “Close”, um das Programmieren der iButtons zu beenden.

6.4 Programmieren im Büro

Mit folgender Benutzeroberfläche können Sie die iButtons im Büro programmieren.

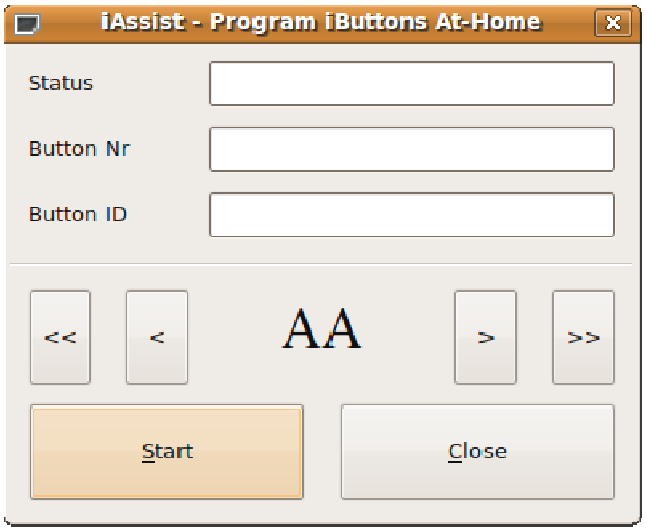

Abb. 23: Programmieren im Büro

Das Programm unterstützt Sie bei folgenden Aufgaben:

- Programmieren der iButtons
- Speichern der iButton ID
- Speichern der iButton Konfiguration

Vorgehensweise:

1. Drücken Sie die Schaltfläche “Start” um dem Programm mitzuteilen, dass nun iButtons programmiert werden sollen.
2. Das Programm zeigt nun im Statusfeld die Mitteilung Please insert NEW iButton an. Sollte im Statusfeld die Mitteilung ERROR: No reader connected! erscheinen, obwohl das Lesegerät angeschlossen ist, lesen Sie bitte im Kapitel Fehlerbehebung weiter.
3. Wählen Sie anhand der Pfeiltasten die gewünschte Footprint ID aus. Durch betätigen der Schaltfläche “<” bzw. “>” schalten Sie um eine ID zurück oder vor. Durch betätigen der Schaltfläche “<<” bzw. “>>” schalten Sie 26 IDs zurück oder vor.
4. Legen Sie einen iButton in das Lesegerät. Entfernen Sie den iButton nicht bevor Ihnen das Programm mitteilt, dass der iButton erfolgreich programmiert wurde.
5. Wenn der iButton korrekt programmiert wurde, wird die Mitteilung Programming iButton DONE im Statusfeld angezeigt. Das erfolgreiche Programmieren des iButtons wird Ihnen zusätzlich akustisch durch einen einzelnen Piepton bestätigt. Sie können den iButton nun entfernen, er wurde mit den im Dialog “Edit Mission Parameter File” gewählten Einstellungen programmiert. Sollte eine Fehlermeldung angezeigt werden, lesen Sie bitte im Kapitel Fehlerbehebung weiter. Falls die Mitteilung Check Mission Paremter file erscheint, so ist entweder

keine Datei mit den Missionsparemetern vorhanden oder ihr Inhalt ist korrupt. Stellen Sie im Dialog “Edit Mission Parameter File” die Parameter erneut ein und drücken Sie auf “Save”. Beginnen Sie anschliessend wieder bei Punkt 1.

6. Wiederholen Sie den Vorgang bei Puntk 2. oder drücken Sie die Schaltfläche “Stop” und anschliessend “Close”, um das Programmieren der iButtons zu beenden.

6.5 Einsammeln der iButtons

Diese Funktion dient zum Einsammeln der iButtons.

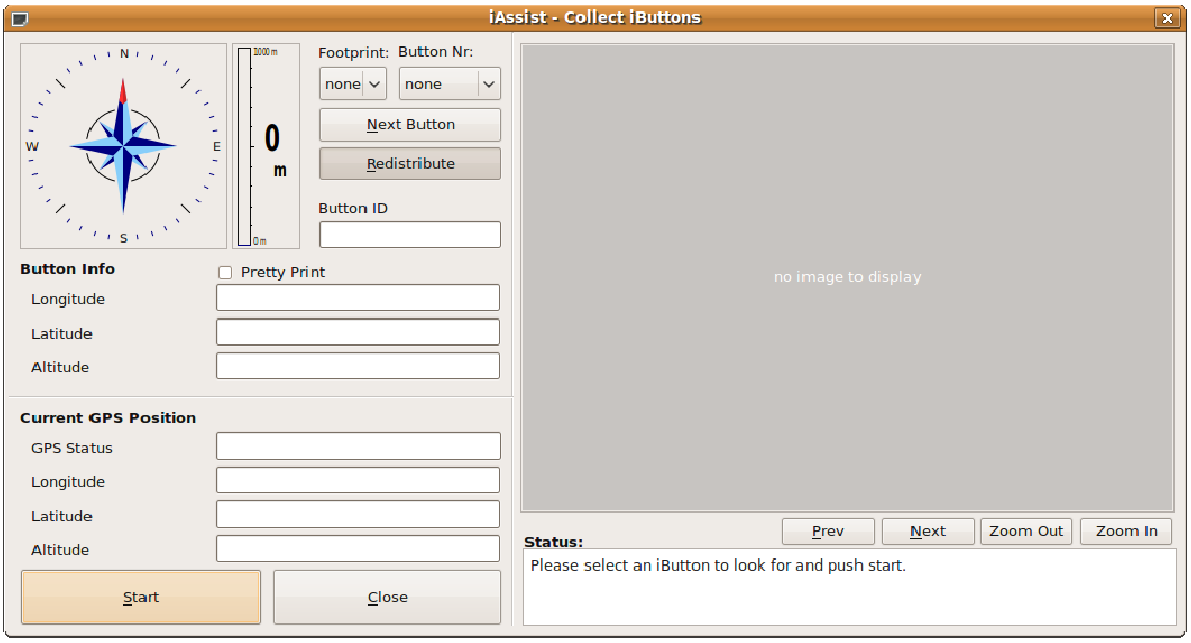

Abb. 24: Einsammeln der iButtons

Das Programm unterstützt Sie bei folgenden Aufgaben:

- Navigation zum ausgewählten iButton im Feld mit Angabe von Distanz und Richtung.
- Anzeige der gespeicherten Fotos zum ausgewählten iButton und Footprint.
- Auslesen der Messdaten aus den iButtons.
- (Optional) Erneutes Programmieren der iButtons.

Vorgehensweise:

Drücken Sie die Schaltfläche “Start” um dem Programm mitzuteilen, dass nun iButtons eingesammelt werden sollen. Wählen Sie gemäss der Statusmeldung Please select an iButton to look for and press start und drücken Sie „Start“. Sollte die Mitteilung ERROR: Cannot connect to reader. erscheinen, obwohl das Lesegerät angeschlossen ist, lesen Sie bitte im Kapitel Fehlerbehebung weiter.

1. Wählen Sie in den entsprechenden Dropdown Menüs einen Footprint und einen iButton aus. Es erscheint nun die Meldung Looking for iButton <iButton Nr> im Statusfeld, wobei <iButton Nr> dem von Ihnen ausgewählten iButton entspricht.
2. Es werden, falls vorhanden, die zugeordneten GPS Daten und Fotos geladen und angezeigt. Falls zum ausgewählten iButton GPS Daten vorhanden sind und Sie momentan über ein GPS Signal verfügen, werden Richtung und Distanz zum iButton angezeigt.
3. Überprüfen Sie die Schaltfläche “Redistribute”. Aktivieren Sie diese, falls Sie die iButtons nach dem Auslesen erneut, mit den im Dialog “Edit Mission Parameter File” eingestellten Einstellungen, programmieren wollen.
4. Legen Sie den ausgewählten iButton in das Lesegerät. Entfernen Sie den iButton nicht, bis das Programm Ihnen mitteilt, dass der iButton ausgelesen wurde. Während dem Auslesen erscheint im Statusfeld die Mittelung Reading mission data.... Ist der Vorgang beendet, steht im Statusfeld die Mitteilung Successfully read iButton data bzw., falls die Funktion “Redistribute” aktiviert ist, Successfully read iButton data and repogrammed it. Sollte die Meldung This iButton does not match with the selected iButton: <Button Nr> angezeigt werden, entspricht der von Ihnen eingelegte iButton nicht dem ausgewählten iButton. Wählen Sie entweder den eingelegten iButton im Dropdownmenü aus oder legen Sie den im Dropdownmenü ausgewählten iButton in den Reader ein.
5. Betätigen Sie die Schaltfläche “Next iButton”, um nach dem nächsten iButton zu suchen, wählen Sie einen anderen iButton aus dem Dropdownmenü oder wählen sie “Stop” um das Einsammeln zu beenden.

6.6 Hinzufügen externer Daten

Diese Funktion dient dem Einfügen von externen Daten in die Datenbank.

Abb. 25: Importieren externer Daten

Das Programm unterstützt Sie bei folgenden Aufgaben:

- Importieren von Daten zu einem iButton.
- Importieren von Daten zu einem Footprint.
- Importieren von Fotos zu iButtons.
- Importieren von Fotos zu Footprints.

Vorgehensweise:

1. Wählen Sie die Schaltfläche “Browse” und geben Sie den Pfad zur .csv Datei an, welche die Informationen zu den iButtons, Footprints oder Fotos enthält.
2. Betätigen Sie die Schaltfläche “Import iButton Info”, “Import Footprint Info” oder “Import Photos”. Das Programm beginnt mit dem Import der externen Daten. Sie können den aktuellen Status im Statusfeld mit verfolgen.
3. Wiederholen Sie den Vorgang falls gewünscht für andere Daten oder beenden Sie den Dialog, indem Sie auf “Close” drücken.

6.6.1 Format der .csv Dateien:

Dieses Format muss exakt eingehalten werden! Dateien mit einem falschen Format können nicht oder nur fehlerhaft eingelesen werden.

Footprint .csv Datei:

```
FootprintID;HAE_GPS;Long_GPS;Lat_GPS;HAE_DEM;East_DEM;North_DEM;Slope_DEM;
Aspect_DEM;SolDy_DEM;SolYr_DEM;Start_Longitude;End_Longitude1;
End_Longitude2;Start_Latitude;End_Latitude1;End_Latitude2
```

Beispiel:

```
AA;2456;9.12;8.12;2697;783288.04;144765.13;35.81;12.57;18.6;1287088.13;
9.82;9.82;9.82;46.43;46.43;46.43
```

Foto .csv Datei:

```
ButtonNr;PhotoName;TimeStamp
```

Beispiel:

```
AA0001;DSCNTest.jpg;12.12.09 15:00
```

iButtons .csv Datei:

```
FootprinttID;ButtonNr;ButtonID;Start_time;Time_when_programmed;Time_GPS;
HAE_GPS;Long_GPS;Lat_GPS;PDOP;HDOP;VerticalError;HorizontalError;
StandardDeviation;HAE_DEM;East_DEM;North_DEM;Slope_DEM;SolDy_DEM;SolYr_DEM
```

Beispiel:

```
AA;1;4F0000000B367841;14.07.09 00:00;13.07.09 08:14;14.07.09 15:26;2693;
9.82;46.43;4.5;2;1.3;0.6;2.44;2697;783288.04;144765.13;35.81;18.6;
1287088.13
```

Falls Sie nur einige ausgewählte Daten in die Datenbank eintragen wollen, so können Sie diese in die entsprechende Spalte in der .csv Datei eintragen und die anderen Spalten leer lassen. Zum Beispiel für den Import einer iButton .csv Datei.

Beispiel:

```
AA;1;;;;;;;;;;;;;;;;;
AA;2;;;;;;;;3.75;;;;;;;;;;
AA;3;;;;;;;;;;;;;;;;;
…
```

Somit wird für den iButton AA002 der zehnte Eintrag, welcher ‚PDOP' entspricht, in der Datenbank mit dem Wert 3.75 überschrieben. Die Einträge „FootprintID", „ButtonNr" und „ButtonID" können nicht überschrieben werden, denn damit wird die Konsistenz der Datenbank sichergestellt.

Fügen sie in keine Spalte einen Leerschlag ein, falls sie dies nicht ausdrücklich wollen, denn sonst wird dieser Wert aus der Datenbank gelöscht!

Beispiel: `AA;1;;;;;;;; ;;;;;;;;;;`

So wird der zehnte Eintrag zum iButton „AA001" in der Datenbank gelöscht!

6.7 Bearbeiten der Datenbank

Diese Funktion dient dazu, unerwünschte Daten aus der Datenbank zu entfernen.

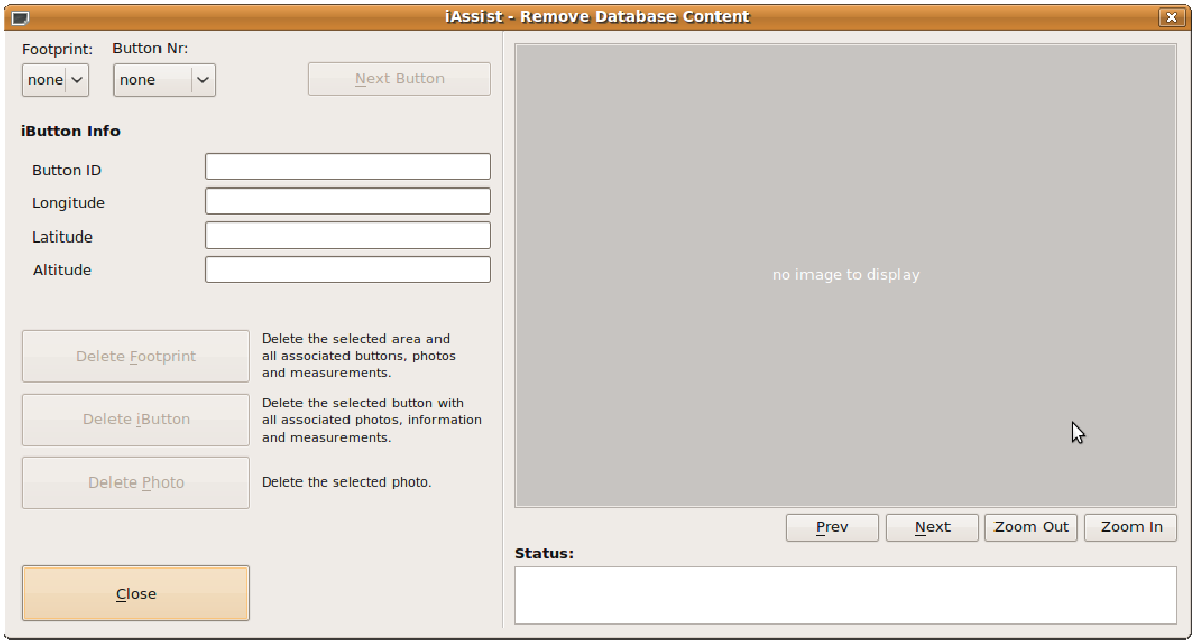

Abb. 26: Bearbeiten der Datenbank

Das Programm unterstützt Sie bei folgenden Aufgaben:

- Löschen von Footprints
- Löschen von iButtons
- Löschen von Fotos

Vorgehensweise um einen Footprint zu löschen:

1. Wählen Sie einen Footprint aus dem Dropdownmenü aus.
2. Betätigen Sie die Schaltfläche “Delete Footprint”.
3. Das Programm fragt Sie, ob Sie sich sicher sind, dass Sie diesen Footprint löschen wollen. Bestätigen Sie mit “Yes”.
4. Das Programm bestätigt Ihnen den Vorgang mit Footprint <Footprint> successfully deleted.

Mit dem Löschen eines Footprints werden alle zugehörigen iButtons, Messdaten, Informationen und Fotos gelöscht!

Vorgehensweise um einen iButton zu löschen:

1. Wählen Sie einen iButton aus dem Dropdownmenü aus.
2. Betätigen Sie die Schaltfläche “Delete iButton”.
3. Das Programm fragt Sie, ob Sie sich sicher sind, dass Sie diesen iButton löschen wollen. Bestätigen Sie mit “Yes”.
4. Das Programm bestätigt Ihnen den Vorgang mit iButton <iButton> successfully deleted.

Mit dem Löschen eines iButtons werden alle zugehörigen Informationen, Messdaten und Fotos gelöscht!

Vorgehensweise um ein Foto zu löschen:

1. Wählen Sie ein Foto aus dem Dropdownmenü aus.
2. Wählen Sie mit den Schaltflächen “Prev” und “Next” ein Foto aus.
3. Betätigen Sie die Schaltfläche “Delete Photo”.
4. Das Programm fragt Sie, ob Sie sich sicher sind, dass Sie dieses Foto löschen wollen. Bestätigen Sie mit “Yes”.
5. Das Programm bestätigt Ihnen den Vorgang mit Photo <Photo> successfully deleted.

6.8 Bearbeiten des Parameter Files

Mit dieser Funktion ist es möglich die Parameter zu wählen, mit denen Sie die iButton Sensoren programmieren wollen.

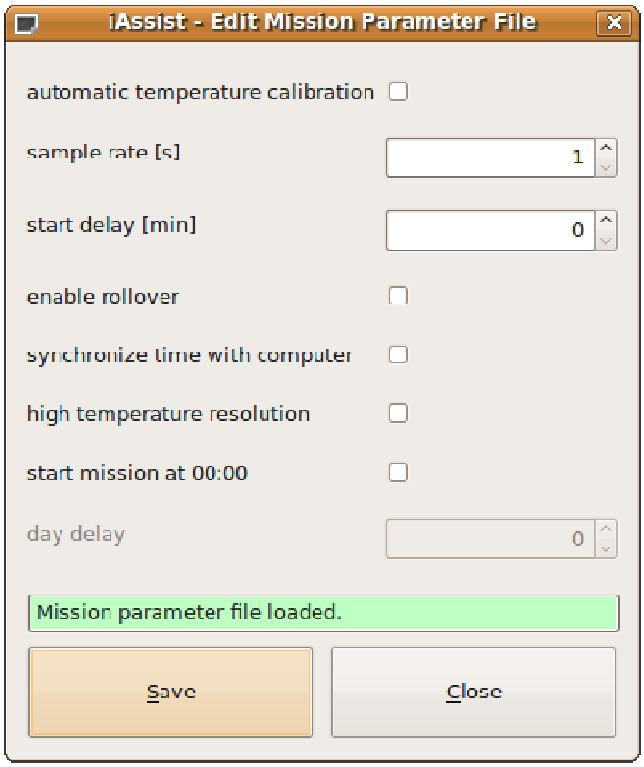

Abb. 27: Bearbeiten des Parameter Files

Das Programm unterstützt Sie bei folgenden Aufgaben:

- Bearbeiten der Einstellungen mit denen die iButtons programmiert werden sollen.

Vorgehensweise:

1. Wählen Sie die Einstellungen entsprechend Ihren Wünschen aus.
2. Damit die gewählten Einstellungen übernommen werden, betätigen Sie die Schaltfläche “Save”. Das Programm bestätigt Ihnen mit der Meldung Mission file saved., dass die Einstellungen erfolgreich übernommen wurden.
3. Beenden Sie den Dialog mit “Close”.

Mögliche Einstellungen:

- **automatic temperature calibration:** Die Genauigkeit der Temperaturmessungen kann durch diesen Korrekturalgorithmus verbessert werden. Dieser Algorithmus wird erst beim Auslesen der Daten angewendet. Das heisst, falls Sie diese Einstellung nutzen wollen, müssen Sie diese Option im Parameter File vor dem Auslesen des iButtons auswählen und das Parameter File speichern.
- **sample rate:** Zeitabstand zwischen den Messungen in Sekunden.
- **start delay:** Verzögerung in Minuten, mit der die Messung auf dem iButton gestartet wird. Der Timer für die Verzögerung läuft ab dem Programmierzeitpunkt. Diese Option kann nur gewählt werden, wenn “start mission at 00:00” nicht aktiviert ist.
- **enable rollover:** Falls diese Option aktiviert ist, werden die Messdaten vom Anfang überschrieben, sobald der Speicher des iButtons voll ist. Falls diese Option deaktiviert ist, wird die Mission gestoppt, sobald der Speicher des iButtons voll ist.
- **synchronize time with computer:** Synchronisiert die Zeit des iButtons mit der Systemzeit des Computers.
- **high temperature resolution:** Falls diese Option aktiviert wird, werden die Messwerte mit 16-Bit Genauigkeit gespeichert, ansonsten mit 8-Bit.
- **start mission at 00:00:** Startet die Messung auf dem iButton um 00:00. Dies Option kann genutzt werden, falls mehrere iButtons synchron mit der Messung beginnen sollen.
- **day delay:** Diese Option kann nur angewählt werden, wenn die Mission um 00:00 gestartet wird. Mit dieser Option kann die Messung 0-365 Tage nach dem Programmieren um 00:00 gestartet werden.

6.9 Die Datenbank

Die Datenbank ist in 5 Tabellen gegliedert.

6.9.1 iButtons

Die Tabelle iButtons enthält sämtliche Informationen die direkt einem iButton zugeordnet werden können.

ButtonNr	Die im Programm verwendete Bezeichnung, zur Kennzeichnung jedes iButtons. Diese setzt sich aus FootprintID (z.B. AA) und einer dreistelligen Nummer zusammen.		AA001
ButtonID	Die ID welche auf jedem iButton aufgeprägt ist.		0B0000000B6A3F41
Longitude	Longitude der Position des iButtons	deg	9.8248
Latitude	Latitude der Position des iButtons	deg	46.43043819
Altitude	Höhe der Position des iButtons in Meter	m	2692.4
GPSTimeStamp	Zeit zu der die GPS Position bestimmt wurde. Diese erlaubt Rückschlüsse auf die Aktualität der Daten.		14.07.2009 15:43:02
HorError	Der horizontale Fehler der GPS Daten	m	0.6
VerError	Der vertikale Fehler der GPS Daten	m	1.2
Distributor	Zusätzliche Felder, die von den Geologen nachträglich genutzt werden, um Geländemodelldaten zu speichern.		
PDOP			
HDOP			
StandardDeviation			
HAE_DEM			
East_DEM			
North_DEM			
Slope_DEM			
SolDy_DEM			
SolYr_DEM			

6.9.2 Footprints

Die Tabelle Footprints enthält sämtliche Informationen die zu einem jeweiligen Footprint gespeichert werden.

FootprintID	Die im Programm verwendete Bezeichnung, zur Kennzeichnung jedes Footprints.		AA
Altitude	Höhe des Footprints	m	2694.6
Longitude	Longitude der Position des Footprints	deg	9.8248
HAE_DEM	Zusätzliche von den Geologen benötigte Zellen um Daten über die Position und das Gelände des Footprints zu speichern.		
East_DEM			
North_DEM			
Slope_DEM			
Aspect_DEM			
SolDy_DEM			
SolYr_DEM			
StartLongitude			
StartLatitude			
EndLongitude1			
EndLatitude1			
EndLongitude2			
EndLatitude2			

6.9.3 Photos

Die Tabelle Photos enthält die Informationen, welche Bilder zu welchem iButton oder Footprint gehören.

PhotoID	Primärer Schlüssel dieser Tabelle.		
ButtonNr	ButtonNr oder FootprintID zu der das zugehörige Foto gehört.		AA001 oder AA
PhotoName	Dateiname des Fotos. Unter diesem Namen ist es ebenfalls im Ordner „img“ finden.		iButton: AA001_01.JPG Footprint: AA_01.JPG
TimeStamp	Zeitpunkt zu dem das Foto aufgenommen wurde.		14.07.2009 15:43:02

6.9.4 MeasurementProfile

Die Tabelle MeasurementProfile enthält Informationen zu den jeweiligen Messungen. Sofern ein iButton neu programmiert wird, werden in dieser Tabelle die zugehörigen Werte gespeichert. Beim Einsammeln des iButtons wird die Tabelle noch mit Einsammelzeit und Zeitdrift des iButtons ergänzt.

MeasurementProfileID	Primärer Schlüssel dieser Tabelle.		
ButtonNr	ButtonNr zur Identifikation zu welchem iButton diese Messung gehört.		AA001
SessionNr	Entspricht der Anzahl Messungen dieses iButtons		
DistributingTime	Zeitpunkt zu dem der iButton ausgelegt wurde		14.07.2009 15:26:05
CollectingTime	Zeitpunkt zu dem der iButton wieder eingesammelt wurde bzw. die Messung ausgelesen wurde.		14.12.2009 15:29:09
TimeShif	Zeitdifferenz zwischen der Realtime Clock auf dem iButton und der Systemzeit auf dem Computer mit dem der iButton ausgelesen wird.	sec	
SamplingRate	Das Messintervall mit dem der iButton Messungen vornimmt.	sec	
SamplingStart TimeStamp	Zeitpunkt zu dem der iButton mit der ersten Messung beginnt.		15.07.2009 00:00:00
Resolution	Die Auflösung der Messungen. 8-Bit oder 16-Bit		16

6.9.5 Measurement

Die Tabelle Measurement enthält die eigentlichen Messdaten.

MeasurementID	Primärer Schlüssel dieser Tabelle.		
ButtonNr	ButtonNr zur Identifikation zu welchem iButton diese Messung gehört.		AA001
MeasurementProfileID	Entspricht dem Messprofil mit welchem diese Messungen gemacht wurden.		
MeasurementNr	Nummerierung der Messungen.		
Measurement	Der Messwert.	°C	27.0

6.10 Fehlerbehandlung

6.10.1 Programmieren vor Ort

ERROR: No reader connected!	Es ist kein iButton Lesegerät angeschlossen oder es kann keine Verbindung zum Lesegerät hergestellt werden. Dies tritt auf, wenn Sie einen Dialog beenden, ohne die Funktion erst zu stoppen. In diesem Fall kann das Problem durch Neustarten des Programms behoben werden. Falls das Neustarten des Programms nicht zum Erfolg führt, wurde das Gerät eventuell vom System bereits registriert und muss erst entfernt werden. Führen Sie dazu in einer Shell den Befehl " sudo rmmod ds2490" aus und geben Ihr Passwort ein.
Cannot stop running mission.	Die laufende Mission auf dem iButton konnte nicht beendet werden. Sie können versuchen den iButton erneut zu programmieren.
Check Mission Parameter file.	Die Datei, welche die Missionsparameter enthält, konnte nicht geöffnet werden oder enthält korrupte Daten. Eventuell wurde nie eine solche Datei angelegt. Öffnen Sie den Dialog "Edit Mission Parameter File" um eine solche Datei anzulegen.
Programming iButton FAILED.	Der iButton konnte nicht programmiert werden. Versuchen Sie den iButton erneut zu programmieren.
Please program an iButton first.	Sie haben versucht ein Foto aufzunehmen, bevor Sie einen iButton programmiert haben. Dies ist nicht möglich, da das Programm dann nicht weiss zu welchem iButton Sie das Foto zuordnen wollen. Programmieren Sie einen iButton und nehmen Sie das Foto anschliessend erneut auf.
No connection.	Es konnte keine Verbindung zum GPS Daemon hergestellt werden. Sie können den GPS Daemon in einer Shell starten, indem Sie den Befehl "sudo gpsd /dev/ttyACM0" ausführen und anschliessend Ihr Passwort eingeben.
Photo could not be deleted.	Das Foto konnte nicht gelöscht werden. Sie können versuchen das Foto erneut zu löschen.

6.10.2 Programmieren im Büro

Cannot stop running mission.	Die laufende Mission auf dem iButton konnte nicht beendet werden. Sie können versuchen den iButton erneut zu programmieren.
Check Mission Parameter file.	Die Datei, die die Missionsparameter enthält, konnte nicht geöffnet werden oder enthält korrupte Daten. Eventuell wurde nie eine solche Datei angelegt. Öffnen Sie den Dialog “Edit Mission Parameter File” um eine solche Datei anzulegen.
Programming iButton FAILED.	Der iButton konnte nicht programmiert werden. Sie können versuchen den iButton erneut zu programmieren.
ERROR: No reader connected!	Es ist kein iButton Lesegerät angeschlossen oder es kann keine Verbindung zum Lesegerät hergestellt werden. Dies tritt auf, wenn Sie einen Dialog beenden, ohne die Funktion zuerst zu stoppen. In diesem Fall kann das Problem durch Neustarten des Programms behoben werden. Falls das Neustarten des Programms nicht zum Erfolg führt, wurde das Gerät eventuell vom System bereits registriert und muss erst entfernt werden. Führen Sie dazu in einer Shell den Befehl “sudo rmmod ds2490”, aus und geben Sie Ihr Passwort ein.

6.10.3 Einsammeln der iButtons

ERROR: Cannot connect to Reader.	Es ist kein iButton Lesegerät angeschlossen oder es kann keine Verbindung zum Lesegerät hergestellt werden. Dies tritt auf, wenn Sie einen Dialog beenden, ohne die Funktion zuerst zu stoppen. In diesem Fall kann das Problem durch Neustarten des Programms behoben werden. Falls das Neustarten des Programms nicht zum Erfolg führt, wurde das Gerät eventuell vom System bereits registriert und muss erst entfernt werden. Führen Sie dazu in einer Shell den Befehl “sudo rmmod ds2490”, aus und geben Sie Ihr Passwort ein.

ERROR: Could not detect a running mission on iButton: <ButtonNr>	Auf dem eingelegten iButton konnte keine laufende Messung gefunden werden.
ERROR: Could not stop the mission on iButton: <ButtonNr>	Die laufende Messung auf dem iButton konnte nicht beendet werden.
ERROR: Could not get mission data of iButton: <ButtonNr>	Die Messdaten konnten nicht ausgelesen werden.
ERROR: Could not save new measurement data to database.	Die Messdaten konnten nicht in der Datenbank gespeichert werden.
ERROR: Could not save new measurement profile to database.	Die Informationen zu den Messungen konnten nicht in der Datenbank gespeichert werden.
ERROR: Could not update the measurement profile in the database.	Die neuen Informationen zu den Messungen konnten nicht zur Datenbank hinzugefügt werden.
This iButton does not match the selected iButton: <ButtonNr>	Sie haben den falschen iButton in das Lesegerät eingelegt. Bitte legen Sie den richtigen iButton in das Lesegerät oder wählen Sie den entsprechenden iButton in der Liste aus.
Stopped collecting iButtons.	Das Einsammeln der iButtons wurde nach einem Fehler beendet.

6.10.4 Hinzufügen externer Daten

Import failed.	Beim Import der Daten ist ein Fehler aufgetreten. Versuchen Sie den Vorganz erneut.

6.10.5 Bearbeiten der Datenbank

No footprint selected.	Sie müssen zuerst einen Footprint auswählen, um ihn zu löschen.
No iButton selected.	Sie müssen zuerst einen iButton auswählen, um ihn zu löschen.
No photo selected.	Sie müssen zuerst ein Foto auswählen, um es zu löschen.
Footprint <FootprintID> could not be deleted.	Der Footprint konnte nicht aus der Datenbank gelöscht werden.
iButton <iButtonNr> could not be deleted.	Der iButton konnte nicht aus der Datenbank gelöscht werden.
Photo <PhotoID> could not be deleted.	Das Foto konnte nicht vom Dateisystem gelöscht werden.
Could not remove <PhotoID> from database, but it was removed from the filesystem.	Das Foto konnte nicht aus der Datenbank gelöscht werden, wurde aber vom Dateisystem gelöscht.

6.10.6 Bearbeiten des Parameterfiles

Could not load mission parameter file.	Die Missionsparameter Datei konnte nicht geöffnet werden. Möglicherweise wurde noch nie eine erstellt. Drücken Sie im Dialog auf “Save”, um eine Datei zu erstellen.
Saving mission parameter file failed.	Das Speichern der Missionsparameter Datei ist fehlgeschlagen.

7 Schlusswort

Das entwickelte System enthält alle geforderten Funktionen für das Austeilen, Einsammeln und Auslesen der Temperatursensoren. Die einfache Bedienung der Software wird durch eine übersichtliche und intuitive grafische Benutzeroberfläche gewährleistet. Das System bietet eine maximale Unterstützung durch die Positionsbestimmung der Sensoren und Navigation mittels GPS sowie den automatisierten Abläufen für das Austeilen und Einsammeln der Sensoren. Die Datenbank bietet eine sichere Ablage für alle Informationen zu den Messungen und Messdaten zur späteren Weiterverarbeitung.

Diese Gruppenarbeit war ein sehr spannendes Projekt, bei welchem wir einen tiefen Einblick in verschiedene Bereiche erhielten. Spannend waren vor allem die Entwicklung einer kundenspezifischen Software sowie der Umgang mit den diversen Hardware Komponenten, wie den Sensoren oder dem GPS. In dieser Arbeit konnten wir unsere Fähigkeiten in der objektorientierten Programmierung in C++ ausbauen und erlernten die Programmierung von grafischen Benutzeroberflächen sowie den Umgang mit dem Linux Betriebssystem.

Die entwickelte Software ist auf jedem Linux Computer lauffähig und kann durch viele weitere Funktionen erweitert werden. So könnte man Funktionen zur Auswertung der Messdaten direkt in unsere Software integrieren oder zusätzlich die Lage der Sensoren auf einer Karte darstellen.
Durch diverse Probleme konnte das Openmoko Neo FreeRunner nicht als Hardware Plattform genutzt werden. Es ist jedoch möglich die Software auf ein anderes vergleichbares Gerät oder ein Nachfolgemodell zu portieren um die Mobilität des Systems weiter zu erhöhen.

I. Abbildungsverzeichnis

Abb. 1: Footprints auf dem Piz Corvatsch 1
Abb. 2: Pin Belegung und Abmessungen 3
Abb. 3: DS1922L Block Diagram 4
Abb. 4: Blockdiagramm, Serielles Lesegerät 4
Abb. 5: Schaltplan, Serielles Lesegerät 5
Abb. 6: Openmoko Neo Freerunner 6
Abb. 7: Neo Software Stack 7
Abb. 8: Schaltplan, Openmoko Zusatz PCB 8
Abb. 9: CAD Layout der Platine. 9
Abb. 10: Asus EeePC 10
Abb. 11: DS2490 USB Lesegerät 11
Abb. 12: Nikon Coolpix L1 11
Abb. 13: AnalogClock Beispiel aus der QT API und Richtungsanzeige von iAssist 14
Abb. 14: AutoDistribute Thread 18
Abb. 15: AutoProgram Thread 19
Abb. 16: CollectThread 20
Abb. 17: Vererbung der Datenbankklassen 23
Abb. 18: Widget zur Richtungsangabe 24
Abb. 19: Widget zur Distanzangabe 24
Abb. 20: IGPS Thread 26
Abb. 21: Hauptfenster 34
Abb. 22: Programmierung vor Ort 35
Abb. 23: Programmieren im Büro 37
Abb. 24: Einsammeln der iButtons 38
Abb. 25: Importieren externer Daten 40
Abb. 26: Bearbeiten der Datenbank 43
Abb. 27: Bearbeiten des Parameter Files 45

II. Abkürzungsverzeichnis

ADC	Analog/Digital Converter
API	Application Programming Interface
CAD	Computer Aided Design
EEPROM	Electrically Erasable Programmable Read Only Memory
GPRS	General Packet Radio Service
GUI	Graphical User Interface
I2C	Inter-Integrated Circuit Bus
IC	Integrated Circuit
LAN	Local Area Network
OS	Operating System
PCB	Printed Circuit Board
RAM	Random Access Memory
RTC	Real Time Clock
SPI	Serial Peripheral Interface
TCP	Transmission Control Protocol
TTL	Transistor-Transistor-Logik
UART	Universal Asynchronous Receiver and Transmitter
USB	Universal Serial Bus

III. Literaturverzeichnis

1. **Maxim Integrated Products.** iButton Beschreibung, 3. Mai 2009. [Online] http://www.maxim-ic.com/products/ibutton/ibuttons/.
2. —. Produktinformationen, 8. September 2009. [Online] http://www.maxim-ic.com/products/ibutton/ibuttons/thermochron.cfm.
3. —. Datenblatt iButton, 28. August 2009. [Online] http://datasheets.maxim-ic.com/en/ds/DS1922L-DS1922T.pdf.
4. —. Datenblatt DS2480B, 8. Mai 2009. [Online] http://datasheets.maxim-ic.com/en/ds/DS2480B.pdf.
5. —. Datenblatt DS9097U, 8. April 2009. [Online] http://datasheets.maxim-ic.com/en/ds/DS9097U-009-DS9097U-S09.pdf.
6. **Openmoko Wiki.** Neo FreeRunner Hardware, 10. Mai 2009. [Online] http://wiki.openmoko.org/wiki/Neo_FreeRunner_Hardware.
7. —. Neo FreeRunner, 10. Mai 2009. [Online] http://wiki.openmoko.org/wiki/Neo_FreeRunner/de.
8. —. Why Openmoko, 10 Mai 2009. [Online] http://wiki.openmoko.org/wiki/Why_Openmoko.
9. —. USB Host, 16. Mai 2009. [Online] http://wiki.openmoko.org/wiki/USB_host.
10. **Asus.** Produktebeschreibung EeePC, 8. September 2009. [Online] http://www.asus.ch/de/.
11. **Maxim Integrated Products.** Datenblatt Lesegerät DS9490, 29. September 2009. [Online] http://datasheets.maxim-ic.com/en/ds/DS9490-DS9490R.pdf.
12. **SQLite.** Dokumentation, 12. September 2009. [Online] http://www.sqlite.org/docs.html.
13. **Trolltech.** QT Reference, 12. September 2009. [Online] http://doc.trolltech.com/.
14. **gPhoto.** supported cameras, 8. September 2009. [Online] http://www.gphoto.org/proj/libgphoto2/support.php.
15. **Berlios.** gpsd Documentation, 4. September 2009. [Online] http://gpsd.berlios.de/#documentation.

Beiblatt zu an der ETH Zürich verfassten schriftlichen Arbeiten

Wir erklären mit unserer Unterschrift, das Merkblatt Plagiat zur Kenntnis genommen, die vorliegende Arbeit selbständig verfasst und die im betroffenen Fachgebiet üblichen Zitiervorschriften eingehalten zu haben.

Merkblatt Plagiat: http://www.ethz.ch/students/semester/plagiarism_s_de.pdf

Ort, Datum	Unterschrift
	Unterschrift
	Unterschrift

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institut für Technische Informatik (TIK)

Frühjahrsemester 2009

GRUPPENARBEIT

für
Guido Hungerbühler, Oliver Knecht und Suhel Sheikh

Betreuer: Matthias Keller
Stellvertreter: Jan Beutel

Ausgabe: xx. März 2009
Abgabe: 11. September 2009

Mobile Application for On-Site Sensor Installation Support

Motivation

Im Rahmen von geologischen Untersuchungen sollen im Sommer 2009 eine grosse Anzahl von iButton-Sensoren [3] für Feldversuche in den Schweizer Alpen installiert werden.

Jeder Sensor kann über eine eindeutige ID identifiziert werden. Eine möglichst genaue Kenntnis über die Position jedes Sensors ist dabei aus zwei Gründen von hohem Interesse. Zunächst trägt diese zusätzliche Information zu einem möglichst genauen und hochwertigen Datensatz bei, der für wettbewerbsfähige Forschung notwendig ist. Viel wichtiger ist allerdings der Aspekt, dass die installierten Sensoren nach mehreren Monaten zum Auslesen der Daten wieder lokalisiert werden müssen. Dies ist nur über eine Kenntnis der Position, bevorzugt anhand von GPS-Koordinaten, möglich.

Konzept

Zur Unterstützung der Forscher bei der Installation und bei der Datenübertragung am Ende des Versuches soll eine mobile, handliche und robuste Lösung entwickelt werden.

Diese muss in einer Basisvariante die folgenden Funktionen implementieren:

- Kommunikation mit iButton-Sensoren zur Programmierung und zum Auslesen des Speichers
- Anbindung an mindestens einen GPS-Empfänger zur Bestimmung der Position
- Speicherung der aufgenommenen Daten auf dem Endgerät, Export-Funktion zur Weiterverarbeitung (z.B. CSV-Datei)

Für die Implementierung wird ein mobiles Endgerät vom Typ Openmoko Neo Freerunner [5] verwendet. Die für das Vorhaben besonders relevanten Eigenschaften dieses Gerätes sind:

- Embedded Linux mit X-Server

- Touchscreen Display (VGA, 640x480)
- Integrierter GPS-Empfänger mit Möglichkeit zum Anschluss einer externen Antenne
- USB-Schnittstelle mit Host-Modus
- Zugang zu I2C, SPI und UART Kommunikationsbussen über Debug-Schnittstelle
- Mobiler und stationärer Internetzugang über Ethernet, Wireless LAN und GPRS

Für die Softwareentwicklung steht eine Vielzahl an Bibliotheken zur Verfügung (siehe Abb. 1). Mit Hilfe dieser Bibliotheken soll eine grafische Anwendung entwickelt werden, die vor allem auf die Benutzbarkeit bei Feldarbeiten optimiert ist. Vergleichbare Anforderungen (Bedienung mit dem Finger ohne Stylus, einfache Aktionen wie Drücken von Buttons) lassen sich unter Anderem in Navigationsgeräten finden (siehe Abb. 2).

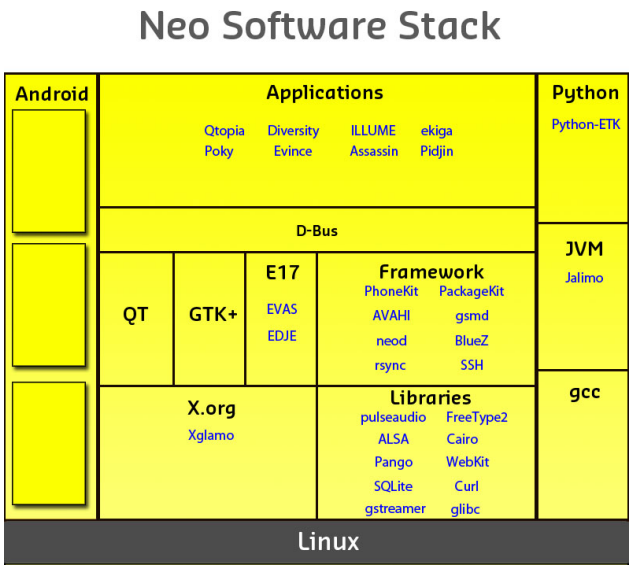

Abbildung 1: Openmoko Software Framework [6]

Abbildung 2: Beispiel für GUI-Applikation auf einem Eingebetteten System

Arbeitspakete

1. Erstellung einer grafischen Anwendung zur Anzeige von Rohdaten des GPS-Empfängers (NMEA-Protokoll). Es wird erwartet, dass diese Anwendung mit GTK+ oder QT4 implementiert wird und auf dem mobilen Endgerät lauffähig ist.

2. Implementierung eines Moduls zur Kommunikation mit iButton-Sensoren. Es ist hierbei erwünscht auf vorhandene Hardware- und Softwarelösungen zurückzugreifen. Beispiele hierfür sind handelsübliche USB-Adapter [4], 1-Wire-To-Serial-Konverter ICs [2] sowie Software-Referenzimplementierungen.
3. Durchführung einer Benutzerstudie zur Festlegung der Funktionen sowie vor allem der grafischen Oberfläche des Endprodukts. In Zusammenarbeit mit Geologen der Universität Zürich sollen hierbei Methoden wie zum Beispiel Paper Prototyping angewandt werden.
4. Implementierung eines lauffähigen Prototypen mit den folgenden Funktionen: Programmierung eines iButton Sensors (Start der Messungen), Auslesen der ID, gemeinsame Speicherung von ID und GPS-Koordinaten.
5. Erweiterung des Prototypen aus Arbeitspaket 4. Neue Funktionen: Auslesen von Daten aus iButton Sensoren, Export der gesammelten Daten in CSV-Datei.
6. Implementierung einer Lösung zur Unterstützung bei der Lokalisation von installierten Sensoren. Eine gute Lösung informiert den Benutzer über Winkel und Länge des Vektors zwischen seiner aktuellen Bewegungsrichtung sowie einem per ID spezifizierten Sensor. Eine Integration von Kartendiensten für eine visuelle Unterstützung ist ebenso denkbar.
7. Entwicklung eines Lesegerätes für iButton Sensoren, das sehr eng an das mobile Handgerät gekoppelt ist. Eine gute Lösung ist besonders robust im Bezug auf die Benutzung bei Feldarbeiten.
8. Fertigstellung des Prototypen. Es wird erwartet, dass das Endprodukt in einen Paketmanager integriert wird um auf weiteren Geräten installiert werden zu können.
9. Erstellung einer ausführlichen Dokumentation. Neben dem Abschlussbericht wird hierbei eine Kurzbedienungsanleitung ueber den Betrieb der Software erwartet.

Es wird empfohlen, dass mehrere Arbeitspakete parallel von unterschiedlichen Bearbeitern durchgeführt werden.

Durchführung der Gruppenarbeit

Allgemeines

- Der Verlauf des Projektes soll laufend anhand des Projektplanes und der Meilensteine evaluiert werden. Unvorhergesehene Probleme beim eingeschlagenen Lösungsweg können Änderungen am Projektplan erforderlich machen. Diese sollen dokumentiert werden.
- Sie verfügen über PC's mit Linux/Windows für Softwareentwicklung und Test. Für die Einhaltung der geltenden Sicherheitsrichtlinien der ETH Zürich sind Sie selbst verantwortlich. Falls damit Probleme auftauchen wenden Sie sich an Ihren Betreuer.
- Stellen Sie Ihr Projekt zu Beginn der Gruppenarbeit in einem Kurzvortrag vor und präsentieren Sie die erarbeiteten Resultate am Schluss im Rahmen des Institutskolloquiums Ende Semester.
- Besprechen Sie Ihr Vorgehen regelmässig mit Ihren Betreuern. Verfassen Sie dazu auch einen kurzen wöchentlichen Statusbericht (email).

Abgabe

- Geben Sie zwei unterschriebene Exemplare des Berichtes spätestens am 11. September 2009 dem betreuenden Assistenten oder seinen Stellvertreter ab. Diese Aufgabenstellung soll vorne im Bericht eingefügt werden.
- Räumen Sie Ihre Rechnerkonten soweit auf, dass nur noch die relevanten Source- und Objectfiles, Konfigurationsfiles, benötigten Directorystrukturen usw. bestehen bleiben. Der Programmcode sowie die Filestruktur soll ausreichen dokumentiert sein. Eine spätere Anschlussarbeit soll auf dem hinterlassenen Stand aufbauen können.

Wichtige Termine

- xx.04.2009, 9-10 Uhr: Einführungspräsentation im TEC Group Meeting
- bis Mitte Juni: Fertigstellung der Arbeitspakete 1, 2 und 4.
- Prüfungssession Sommer 2009: 03.08. bis 28.08.2009
- voraus. Anfang September: Abschlusspräsentation
- spätestens 11.09.2009: Abgabe des Abschlussberichtes

Literatur

[1] Matthias Keller. SADA Survival Guide. http://www.tik.ee.ethz.ch/~kellmatt/ThesisSurvivalGuide.

[2] Maxim IC. DS2480B – Serial to 1-Wire Line Driver. http://www.maxim-ic.com/quick_view2.cfm/qv_pk/2923.

[3] Maxim IC. iButton Landing Page. http://www.maxim-ic.com/products/ibutton/.

[4] Maxim IC. iButton Product: Adapters. http://www.maxim-ic.com/products/ibutton/products/adapters.cfm.

[5] Openmoko. Neo FreeRunner Hardware. http://wiki.openmoko.org/wiki/Neo_FreeRunner_Hardware.

[6] Openmoko. Why Openmoko/Software Architecture. http://wiki.openmoko.org/wiki/Why_Openmoko.

[7] E. Zitzler. Studien- und Diplomarbeiten, Merkblatt für Studenten und Betreuer, March 1998.

Date	Section	Changes
March 18, 2009		Initial version
March 24, 2009		Added work packages and due dates

Tabelle 1: Revision History