Carl von Ossietzky
Universität Oldenburg

Bachelorstudiengang Wirtschaftsinformatik

BACHELORARBEIT

Entwicklung einer automatischen Wasserqualitätsprüfstation

vorgelegt von:
Eike Bockhorst

Betreuender Gutachter:
Prof. Dr.-Ing. Oliver Theel

Zweiter Gutachter:
Dipl.-Inform. Eike Möhlmann

Oldenburg, 11.12.2013

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Problemstellung

Aufgrund menschlichen Einflusses, wie zum Beispiel Düngung oder Ähnlichem, ist es immer wieder nötig, in Teichen, Seen oder Flüssen die Wasserqualität zu kontrollieren. Dies dient in erster Linie dem Schutz des Grundwassers. Weiterhin sollen dadurch auch Flora und Fauna in ihrem Lebensraum geschützt werden.
Gemessen wird dabei unter anderem der pH-Wert, um Übersäuerung oder Versalzung zu entdecken. Weitere relevante Werte sind die spezifische Leitfähigkeit und die Temperatur des Wassers. Allerdings gibt es keine einfachen, mobilen, kompakten und kostengünstigen Messgeräte, mit denen diese Werte über einen festen und auch längeren Zeitraum gemessen und aufgezeichnet werden können. Somit müssen die Messungen zeitaufwändig und in regelmäßigen Abständen „von Hand“ für jedes einzelne zu prüfende Gewässer vorgenommen werden.

2 Zielsetzung

Ziel dieser Arbeit ist es, eine tragbare, wiederverwendbare, kompakte und kostengünstige Messtation zu entwerfen, die die relevanten, oben beschriebenen, Messungen automatisch durchführt und diese aufzeichnet.
Diese Messstation soll an einem Gewässer aufgestellt werden können und dann die nötigen Werte (z.B. pH-Wert, Temperatur, spezifische Leitfähigkeit) messen. Diese Werte sollen intern an einen Computer weitergeleitet werden. Dieser soll die Werte mit einem Zeitstempel und optional mit einer per GPS ermittelten Position versehen und sie anschließend in einer integrierten Datenbank ablegen. Optional könnte man die Werte auch über ein GSM-Modul oder ein WLAN-Modul versenden. So könnten die Nutzer der Station immer einen aktuellen Überblick über die gemessenen Werte des zu untersuchenden Gewässers erhalten. Diese Messstation soll über eine Batterie mit Energie versorgt werden, um über einen längeren Zeitraum (etwa eine Woche) ohne Unterbrechungen Messwerte aufzuzeichnen.

3 Anwendungsszenario und Anforderungskatalog

In diesem Abschnitt soll beschrieben werden, wie die Station arbeiten soll und wie sie von den einzelnen Nutzern (den Informatikern, die die Station konfigurieren, den Biologen, die die Station in Betrieb nehmen und den Informatikern und Biologen, die hinterher die Daten auswerten) benutzt werden soll.
In der Station soll zuerst an einem Rechner eingestellt werden, welche Werte, von wann bis wann, in welchen Abständen gemessen werden sollen. Die dafür nötigen Parameter können von Biologen an die Informatiker weiter gegeben werden, damit diese die Konfiguration vornehmen. Danach soll die Station zu einem Gewässer gebracht und dort aufgebaut werden. Die Sensoren sollen dabei im Wasser plaziert werden, der Rest der Station soll am Ufer stehen. Die Station muss wasserdicht sein, um äußeren Umwelteinflüssen wie Regen oder Taubildung zu widerstehen. Die Station soll dann eingeschaltet werden und unter den voreingestellten Parametern die Messungen aufnehmen. Um demjenigen, der die Station aufstellt zu signalisieren, dass die Messungen wie konfiguriert aufgenommen wurden, sollte die Station eine Rückmeldung, z.B. in Form einer blinkenden LED, geben. Die Werte werden in einer Datenbank auf einem USB-Stick abgespeichert. In die Datenbank eingetragen werden Werte wie die GPS-ermittelte Position, die Uhrzeit und dazugehörige Messwerte (Temperatur/pH-Wert). Um die Station vor Diebstahl zu schützen, muss sie tarnbar sein, d.h. ihren geometrischen Ausmaßen müssen entsprechende Grenzen gesetzt werden. Die Messungen sollen so lange ausgeführt und die Werte in einer Datenbank abgespeichert werden, bis entweder der Akku leer ist, die Station abgeschaltet und abgeholt wird, oder die Zeitpläne der Messungen abgearbeitet sind. Danach kann die Datenbank ausgelesen werden.

Daraus ergeben sich für die Station einige Anforderungen. Anforderungen die sich durch die Auswahl an Geräten ergeben haben, werden ebenfalls aufgelistet.

1 funktionale Anforderungen

- die Messtation soll mobil und tarnbar sein, daher muss
  - die Messtation möglichst klein sein
  - die Messtation tragbar sein
- die Station soll wassserdicht sein
- die Parameter für die Messungen (Zeitpunkte, Zeitabstände, Messwerteauswahl) sollen einstellbar sein
- nach dem Aufstellen und Einschalten der Station sollen die Messungen direkt nach Zeitplan beginnen, auch soll es eine Stoppfunktion geben
- die Messwerte sollen persistent, z.B. in einer Datenbank abgelegt werden

- die Messwerte sollen auf einem USB-Stick abgelegt werden, der ausgewechselt werden kann
- der Prototyp soll zumindest pH-Wert, Temperatur und die spezifische Leitfähigkeit messen können
- bei Einschalten oder Abschalten der Station soll es eine Rückmeldung (z.B. über eine LED) an den Nutzer geben

2 technische Anforderungen

- die Station soll auf Hardware- und Softwareebene bezüglich weiterer Sensoren und deren Messwerte leicht erweiterbar sein
- der Computer der Station soll (nach Aufgabenstellung) ein Raspberry Pi sein
- das Betriebsystem der Station muss (durch den Einsatz des Raspberry Pi) Linux sein

3 Benutzungsschnittstellen

- der Raspberry Pi soll per Anschluss an Bildschirm, sowie Maus und Tastatur konfigurierbar sein
- eine Rückmeldemöglichkeit (z.B. eine LED) soll beim Ein/Auschalten gegeben sein
- die Station soll über einen Knopf o.Ä. einschaltbar sein

4 Qualitätsanforderungen

- die Station soll die Werte zuverlässig und mit möglichst geringen Abweichungen und mit möglichst geringen Messfehlern messen
- die Station soll mindestens eine Woche lang ohne Unterbrechungen (wie z.B. Akkutausch) Werte aufzeichen können

4 Architektur

Aus Anwendungsszenario und Anforderungskatalog ergeben sich für die Station eine mögliche Architektur. Im folgenden soll diese Architektur, unterteilt in Hardware und Software, dargestellt werden.

4.1 Hardware

Der Aufbau besteht auf Hardwareebene aus drei Komponenten.
Die erste Komponente ist für die Messung der Werte zuständig. Sie besteht ihrerseits aus mehreren Modulen. Ein Modul stellt einen Hardwaresensor dar, welcher mindestens einen, möglicherweise aber auch mehrere Werte misst. Die Module sollen ihre Werte an festen Zeitpunkten messen und diese dann an die zweite Komponente weitergeben. Es soll

auch möglich sein, die Messkomponente um weitere Module für weiterer Werte zu erweitern.

Die zweite Komponente ist für die Verarbeitung der Werte zuständing. Das erste und zentrale Modul dieser Komponente ist der Computer. Er steuert die Messkomponente, so dass die Werte nach einem fest konfigurierbaren Ablauf gemessen werden können. Die gemessenen Werte soll der Computer, mit einem Zeitstempel versehen, in einer Datenbank ablegen, welche sich auf dem zweiten Modul, dem Speichermodul (z.B einem USB-Stick) befinden. Zwei optionale Module könnten ein GPS-Modul und ein GSM- oder WLAN-Modul sein. Mit dem GPS-Modul könnte noch eine Position für die Werte ermittelt und ebenfalls abgespeichert werden, mit dem GSM- oder WLAN-Modull könnten die Werte auch direkt an einen festen Empfänger gesendet werden, um immer einen aktuellen Überblick über die Werte zu haben. Ein weiteres Modul der Verarbeitungskomponente ist eine Uhr. Der nach der Aufgabenstellung zu nutzende Raspbery Pi hat selbst keine integrierte Uhr, welche die Zeit immer aktuell misst bzw. zurückgibt.

Die dritte Komponente ist für die Energieversorgung zuständig. Im Optimalfall besteht sie nur aus zwei Modulen: einer Batterie und einem Spannungswandler, da die Station möglichst klein gehalten werden soll.
Je nach Standort, an dem die Messtation aufgestellt wird, könnte allerdings auch eine feste Energiequelle (z.B. über ein Kabel) angeschlossen werden, die die Station mit Energie versorgt.

Dieser schematische Aufbau wird in folgernder Grafik noch einmal verdeutlicht:

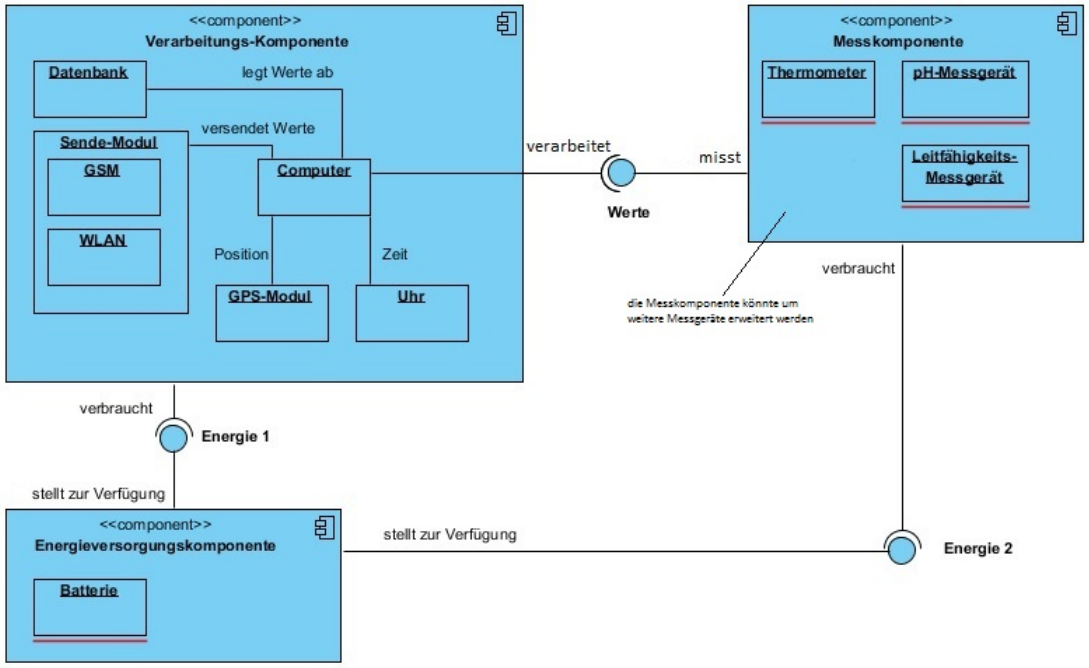

Abbildung 1: *schematischer Aufbau der Hardware*

4.2 Software

Die Software befindet sich hauptsächlich auf dem Raspberry Pi. Auch sie besteht aus mehreren Komponenten.

Die erste Software-Komponente ist das Betriebssystem. Da der Computer, der für diese Arbeit verwendet werden soll, ein Raspberry Pi ist, wird das Betriebssystem Linux-basiert sein (z.B. Raspbian o.Ä.). Das Betriebsystem soll in dieser Messtation typische Aufgaben übernehmen. Dazu gehören unter anderem die Geräteverwaltung, um die angeschlossenen Messgeräte des Messmoduls zu überwachen, oder die Prozessverwaltung, unter der das Programm zur Steuerung der Messkomponente oder das Datenbankprogramm laufen.

Die zweite Software-Komponente ist das Programm, welches die Zeitpläne der Messungen aus der Datenbank liest, die Messungen gemäß dieser Zeitpläne durch die Messgeräte vornehmen lässt und die Messergebnisse schließlich in die entsprechenden Datenbanktabellen einträgt. Der Zugriff auf die Messgeräte erfolgt über die entsprechende Geräteschnittstelle. Im Zuge der Erstellung dieser Arbeit, soll dieses Programm selbst geschrieben werden.

Die dritte Software-Komponente ist das Datenbank-Management-System (kurz DBMS), welches eine entsprechende SQL-Datenbank hält. In diesem sollen die Zeitpläne für die Messungen eingetragen werden, welche später durch das Messprogramm genutzt werden. Des Weiteren sollen in der Datenbank auch die Messwerte mit Zeitstempel abgelegt wer-

den.

Auch dieser Aufbau soll anhand folgender Grafik verdeutlicht werden.

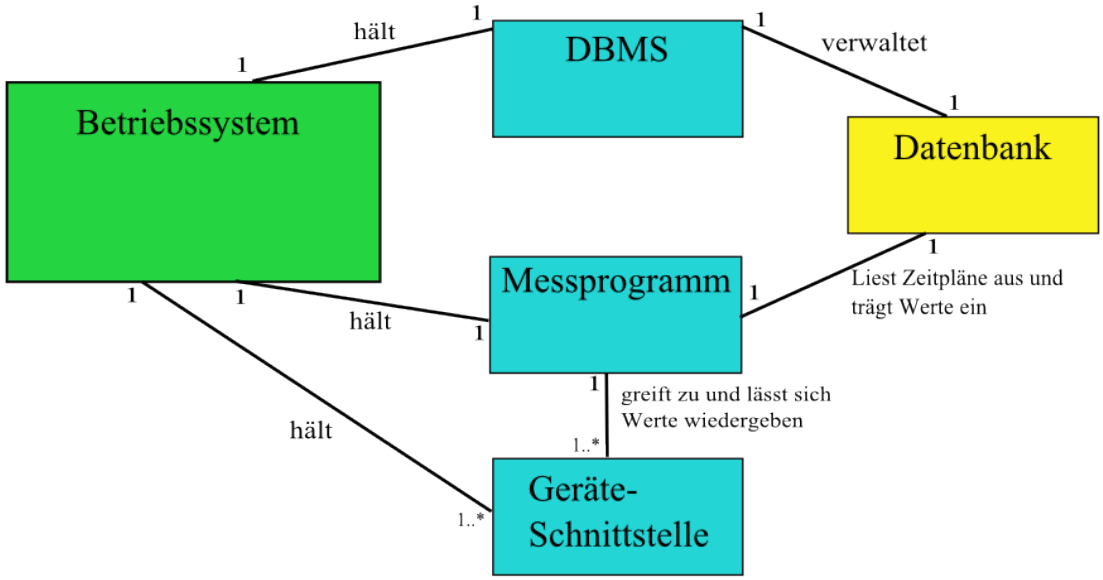

Abbildung 2: *schematischer Aufbau der Software*

5 Implementierung

Im Folgenden wird dargestellt, wie und welche Teile der geplanten Architektur umgesetzt wurden.

5.1 Hardware

Im Folgenden wird die Hardware in die drei bekannten Komponenten unterteilt.

5.1.1 Messkomponente

Für diese Komponente wurde eine Platine ausgewählt, welche einen großen Teil der benötigten Funktionalitäten zur Verfügung stellt. Es handelt sich hierbei um den USBDrDAQ-Data-Logger, eine kleine Platine der Firma Pico Technologies. Die große Besonderheit und auch ein großer Vorteil dieser Platine ist es, dass sie kompatibel mit dem Raspberry Pi und dessen Linux-Betriebssystem Debian 6 “Squeeze“ ist.
Weiterhin sind für diese Platine bereits eine große Anzahl an Sensoren verfügbar. Viele davon stammen ebenfalls aus dem Hause Pico Technologies, es ist jedoch auch möglich, externe Sensoren anzuschließen. Daher wurden auch zwei Sensoren von Pico Technologies ausgewählt, welche die nötigen Messungen vornehmen sollen. Es handelt sich hierbei um einen Sensor für die Temperatur und eine pH-Elektrode. Letztere ist ebenfalls für die Messungen von Spannungswerten einsetzbar, welche auf die spezifische Leitfähigkeit schließen lassen.
Die einzelnen Geräte im Detail:

Der USBDrDAQ-Data-Logger Der USBDrDAQ-Data-Logger ist eine kleine Platine, welche für die Datenaufzeichnung, als Oszilloskop oder als Signalgenerator eingesetzt werden kann.

Abbildung 3: *USBDrDAQ-Data-Logger* [Bild1]

Die Platine hat eine Abmessung von 77 x 70 x 23 Millimeter und ein Gewicht von 60 Gramm [1]. Daher ist sie, in Bezug auf die geringe Größe, welche die Station haben soll, ideal für den Einsatz in diesem Projekt.
Auf folgender Grafik wird schematisch dargestellt, welche Sensoren und Anschlüsse auf der Platine verbaut sind.

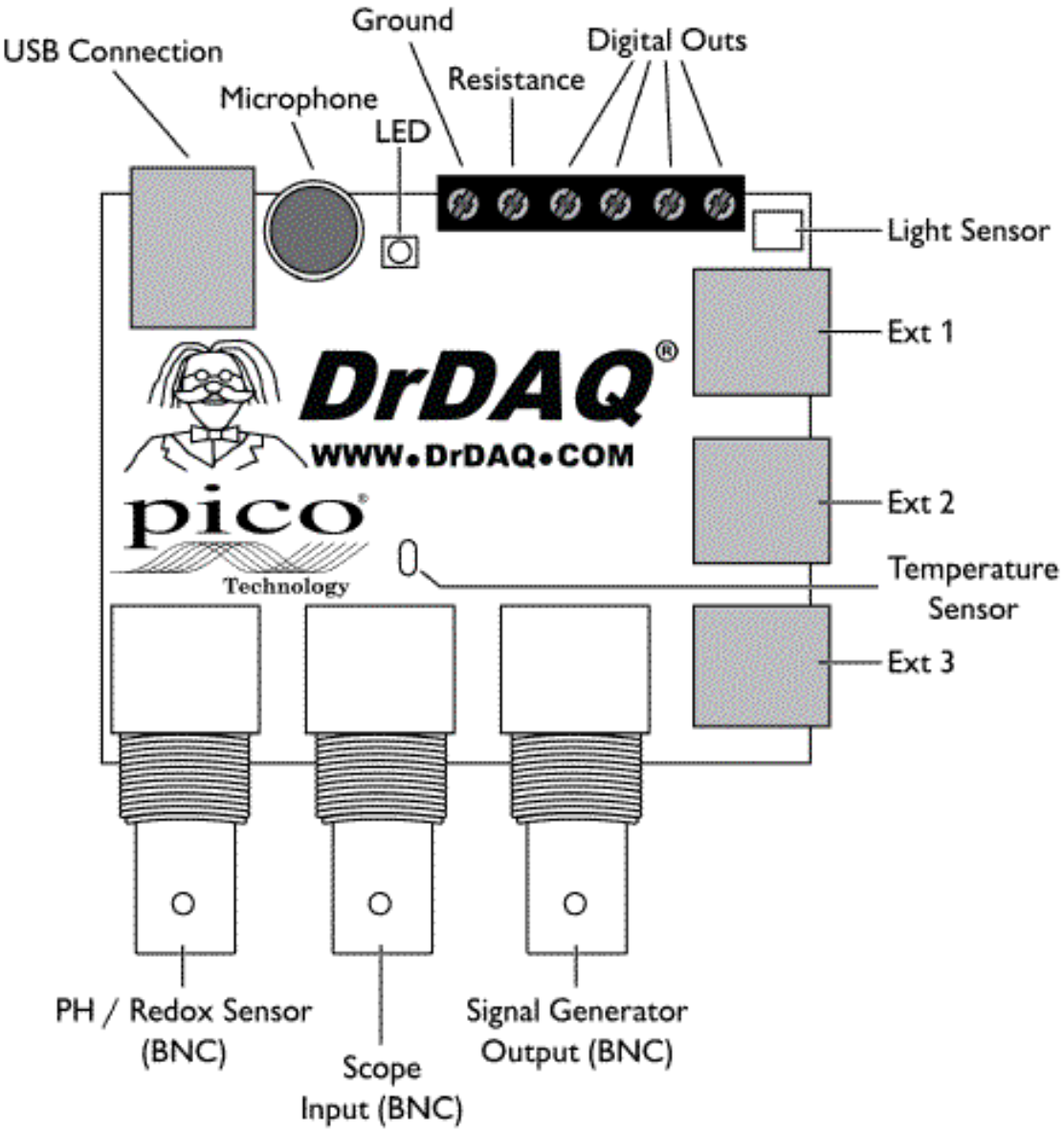

Abbildung 4: *schematische Darstellung der Messplatine* [Bild2]

Die Platine ist mit folgenden Anschlüssen und Sensoren ausgestattet [1]:

- ein BNC-Anschluss für einen pH- oder Redox-Sensor
- ein BNC-Anschluss für ein Oszilloskop
- ein BNC-Anschluss für einen Signalgeneratur
- drei externe Anschlüsse (Ext 1-3) für den Anschluss externer Sensoren
- ein USB-Anschluss für dem Anschluss an einen PC via USB-Kabel
- ein auf der Platine verbauter Temperatursensor

- ein Licht-Sensor
- ein Mikrofon, einsetzbar als Schall-Sensor
- eine LED, welche über die Software konfiguriert und eingestellt werden kann
- 6 Schraubanschlüsse, 4 benutzbar als digitale Ein- und Ausgänge, einer benutzbar für die Erdung und einer für die Messung von Widerständen

Für dieses Projekt sind vor allem der Anschluss einer pH-Elektrode und der Anschluss eines externen Temperatursensors von Relevanz. Beide wurden ebenfalls von Pico Technologies hergestellt und in einem Set mit der Platine (USB DrDAQ pH Measuring Kit) [3][4] verkauft. Anbei ist ebenfalls ein kostenloses Software Development Kit (SDK) [5]. In diesem sind Beispielprogramme in C, C++, Microsoft Excel und National Instruments LabVIEW enthalten. Außerdem ist die hauseigene Software PicoScope und PicoLog enthalten, welche unter Windows XP, Vista und 7 läuft.

Der Temperatursensor Der DrDAQ Temperatursensor ist für Flächen, Luft und Wasser geeignet. Er wird über ein 200 cm langes Kabel an einem der Anschlüsse für externe Sensoren an die Platine angeschlossen. Er kann Temperaturen zwischen $-10\,°C$ und $105\,°C$ messen, wobei er eine Auflösung von $0,1°$ und eine Genauigkeit von $0,3°$ hat [2].

Abbildung 5: *DrDAQ Temperatursensor* [Bild3]

Die pH-Elektrode Der DrDAQ pH-Sensor ist eine 12 x 120 Millimeter große pH-Elektrode. Sie deckt das komplette pH-Spektrum von 0 bis 14 ab, die Arbeitstemperatur liegt zwischen 0 °C und 60 °C und die Genauigkeit liegt bei 0,02 pH. Geliefert wird die Elektrode mit einem kleinen Fläschchen mit einer Aufbewahrungslösung, in welche die Elektrode getaucht werden muss, um außerhalb eines Einsatzes vor Austrocknung geschützt zu sein [2]. In diese Arbeit soll die pH-Elektrode für das Messen des pH-Wertes und das Messen der Spannung, welche auf pH-Wert und spezifische Leitfähigkeit schließen lässt, verwendet werden.

Abbildung 6: *DrDAQ pH-Elektrode* [Bild4]

5.1.2 Verarbeitungskomponente

Die Verarbeitungskomponente besteht aus mehreren Modulen. Der Hauptteil ist der, durch die Aufgabenstellung vorgegebene, Raspberry Pi. Auf diesem laufen das Betriebssystem, das Messprogramm sowie das Datenbank-Management-System (DBMS). Weitere Module sind der WLAN-Empfänger und der GPS-Empfänger.
Nachfolgend die einzelnen Geräte im Detail.

Der RaspberryPi In diesem Projekt kommt ein Rasperry Pi Modell A zum Einsatz.

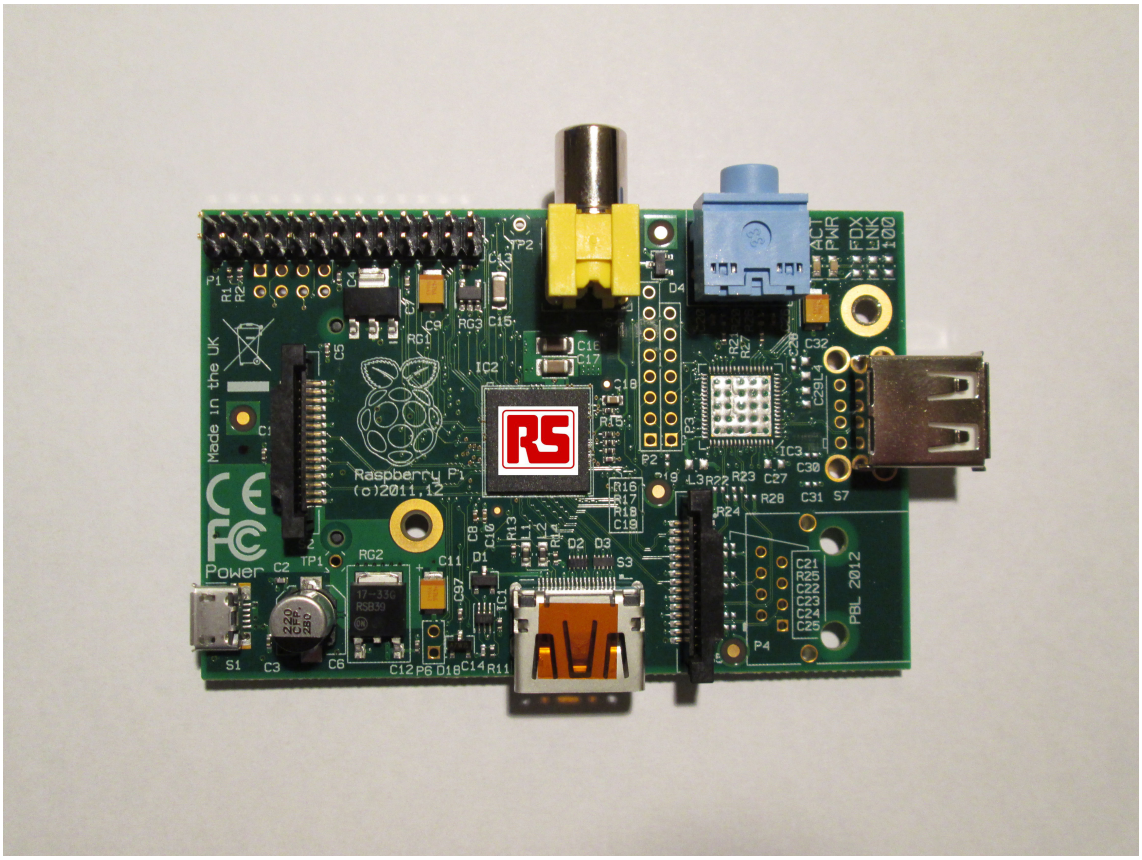

Abbildung 7: *Raspberry Pi Modell A* [Bild5]

Dieses Modell ist ein kleiner Ein-Platinen-Computer, welcher mit allen nötigen Anschlüssen ausgestattet ist. Er hat Abmessungen von 85,60 x 53,98 x 17 Millimetern (Keditkartengröße). Ausgestattet ist er mit einem 700 MHz ARM1176JZF-S Prozessor, einem Broadcom VideoCore IV Grafikprozessor und 256 Megabyte Arbeitsspeicher. Es wird mit einer Spannung von 5 Volt bei 500 Milliampere betrieben, was einer Leistungsaufnahme von 2,5 Watt entspricht[8].

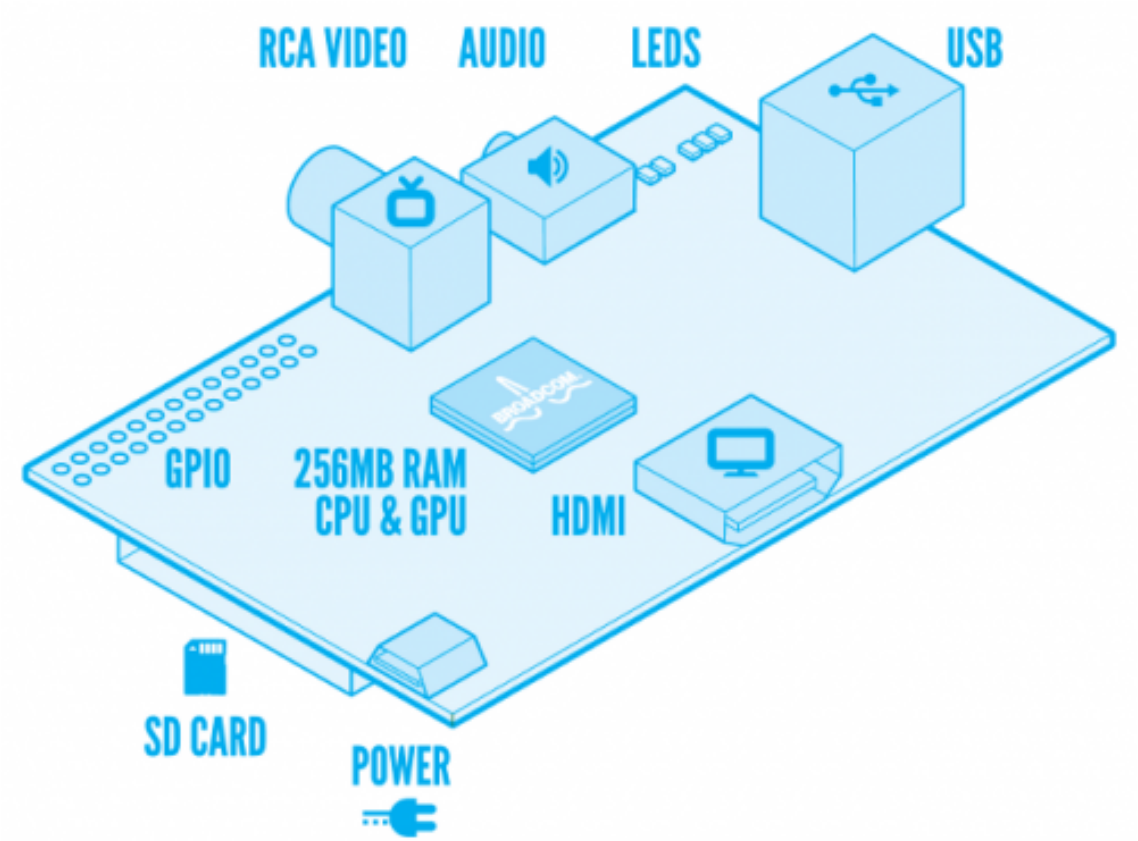

Abbildung 8: *schematische Darstellung des Raspberry Pi Modell A* [Bild6]

Der Raspberry Pi ist mit folgenden Anschlüssen ausgestattet[8]:

- ein Micro USB-Anschluss für die Stromversorgung
- ein SD-Kartenslot, für die SD-Karte mit Betriebssystem
- 10 GPIO-Pins für die freie Konfiguration
- ein HDMI-Anschluss
- ein Standard USB-Anschluss, welcher auch für die Stromversorgung genutzt werden kann.
- ein Audio-Ausgang
- ein Video-Ausgang

Der GPS-Empfänger Als GPS-Empfänger fungiert eine Holux GR-213U GPS-Maus der Firma Holux Technology, Inc. . Sie empfängt über in Reichweite befindliche GPS-Satelliten die aktuelle Position, wobei die Position je nachdem, wie viele Satelliten die Daten senden, genauer wird. Angeschlossen wird die Maus per USB.

Abbildung 9: *Holux GR-213U GPS-Maus* [Bild7]

Das Gerät meldet sich über den USB-Anschluss als serielles Gerät am Rechner an. Entsprechend können die Daten auch aus dem Gerät ausgelesen werden. Es ist etwa 84 Gramm schwer und hat eine Abmessung von 42 x 17,8 Millimeter[6].

Der WLAN-Empfänger Für eine effiziente Einrichtung und Installation des Systems ist auch ein WLAN-Empfänger nötig. Entscheident war hier, einen Empfänger mit einem Chipsatz zu finden, für den unter Debian 6 Treiber existieren. Der hier verwendete WLAN-Adapter ist ein D-Link DWL-G122 G WLAN-Stick. Dieser WLAN-Adapter wird über USB angeschlossen und unterstützt Verbindungen mit 54 Mbit pro Sekunde[7].

Abbildung 10: *D-Link DWL-G122 G WLAN-Adapter* [Bild8]

5.1.3 Energieversorgungskomponente

Aus Zeitgründen konnte dieser Teil der Station nicht ausgearbeitet werden. Es wird aber im Abschnitt *Test* eine Analyse zum Thema Energieverbrauch und möglicher Energiequellen geben.

5.1.4 Eingabegeräte/Verschiedenes

Um den Raspberry Pi sowie die Geräte ausreichend mit Strom zu versorgen und zu konfigurieren, waren weitere Geräte nötig. Zum einen ist ein aktiver USB-Hub mit vier USB-Ports nötig. An diesem werden die einzelnen USB-Geräte angeschlossen und auch der RaspberryPi mit Strom versorgt. Zum anderen waren noch eine USB-Maus von Logitech und eine Standard-USB-Tastatur von Cherry nötig. Auch ist eine 8 GB SD-Speicherkarte, auf der das Betriebssystem installiert wird, vonnöten.

5.1.5 Zusammenbau der Hardware

Da jedes Teil der Hardware über einen USB-Anchschluss verfügte, gestaltete sich der Zusammenbau sehr simpel. Der Raspberry Pi wird an den USB-Hub angeschlossen, danach können die anderen Geräte an den Hub angeschlossen werden. Mit einem HDMI-Kabel ließ sich der Raspberry Pi an einen Bildschirm anschließen. Wird nun der USB-Hub an einer normalen Steckdose angeschlossen, werden alle Geräte mit Energie versorgt und der Raspberry Pi startet und fährt das Betriebssystem hoch.

5.2 Software

Im Folgenden wird zuerst einmal erläutert, wie das Betriebssystem eingerichtet wurde. Danach wird erläutert, wie Treiber und Geräte auf Softwareebene konfiguriert wurden. Danach wird die Konfiguration des Datenbank-Management-Systems erklärt und zu guter Letzt wird das Messprogramm dargestellt, welches für dieses Projekt selbst erstellt wurde.

5.2.1 Das Betriebssystem

Als Betriebssystem wurde eine eigens für den Raspberry Pi herausgegebene Distribution von Debian 6 “Squeeze“ gewählt. Hauptgrund hierfür ist, dass es die einzige Debian Distribution ist, für die Treiber für die USB DrDAQ Platine existieren. Auch ist Debian 6 eine Linux-Distribution, die als komplettes Betriebssystem fungiert, mit welchem man auch die Messsoftware entwickeln konnte. Dies liegt unter anderem daran, dass mit dem gcc-Compiler und dem Aptitude Package-Manager alle nötigen Pakete zur Entwicklung des Messprogrammes vorhanden sind oder, wenn nötig, nachinstalliert werden können.

Das Betriebssystem wird mit einem auf der Homepage der Raspberry Pi Foundation empfohlenen Programm namens Win32DiskImager[14][13] auf die SD-Karte aufgespielt. Dieses Programm ist allerdings für Windows-Betriebssysteme konzipiert, unter Linux gibt es auch eigene Möglichkeiten das Betriebssystem auf die SD-Karte zu kopieren. Nach der Installation kann die SD-Karte in den SD-Kartenslot des Raspberry Pi gesteckt werden und durch den Anschluss des Stroms wird der Raspberry Pi eingeschaltet. Mit der Eingabe von "pi" als username und "raspberry" als password kann man sich im System anmelden. Danach sind allerdings weitere Einstellungen vonnöten um den Raspberry Pi und das Betriebssystem verwenden zu können. Entsprechende Anleitungen sind im Internet zu finden und wurden hier auch als Quellen angegeben.

1 Umstellung des Tastatur-Layouts[9]

- *sudo dpkg-reconfigure keyboard-configuration* in die Befehlszeile eingeben und mit RETURN bestätigen
- "Generic 105-key (Intl) PC" für eine normale deutsche Tastatur auswählen und mit RETURN bestätigen
- "Other" auswählen und mit RETURN bestätigen
- "Germany" auswählen und mit RETURN bestätigen
- wieder "Germany" (ohne weitere Optionen) auswählen und mit RETURN bestätigen
- wieder "Germany" auswählen und mit RETURN bestätigen
- die nächsten beiden Fenster können mit RETURN bestätigt werden
- zuletzt wird gefragt, ob "Control-Alt-Backspace" den X Server beenden soll; da dies nützlich sein kann, auch hier mit RETURN bestätigen

2 Einstellen der Zeitzone[9]

- *sudo dpkg-reconfigure tzdata* in die Befehlszeile eingeben und mit RETURN bestätigen
- "Europe" mit RETURN bestätigen
- zuletzt die Zeitzone von "London" auf "Berlin" umstellen

3 Installation des WLAN-Adapters

- als Nächstes sollten die Treiber für den WLAN-Adapter installiert und konfiguriert werden, dies wird unter *Geräteeinrichtung und Treiber* erklärt

4 Systemupdate[9]

- *sudo apt-get update* in die Befehlszeile eingeben, mit RETURN bestätigen; das System aktualisiert nun die Pakete
- *sudo apt-get upgrade* in die Befehlszeile eingeben, mit RETURN [Y] bestätigen

5 Erweiterung des Dateisystems[10]

- *sudo fdisk -uc /dev/mmcblk0* in die Befehlszeile eingeben und mit RETURN bestätigen.
- "p" eingeben um sich die Partitionstabelle anzeigen zu lassen
- mit dem Befehl "d" und dann der Eingabe von "3" wird die dritte Partition gelöscht
- mit dem Befehl "d" und dann der Eingabe von "2" wird die zweite Partition gelöscht
- mit dem Befehl "n", dann der Eingabe von "p" und zuletzt der Eingabe von "2" wird eine neue primäre Partition erstellt
- für diese muss der Startsektor angegeben werden, der in der Partitionstabelle bei "Start" von */dev/mmcblk0p2* steht, NICHT der Default-Wert
- für den letzten Sektor kann der angegebene Standard genutzt werden
- mit "w" wird dann die Partitionstabelle neu geschrieben
- danach fdisk verlassen und mit *sudo reboot* das System neu starten
- nach dem Neustart wird mit dem Befehl *sudo resize2fs /dev/mmcblk0p2* wird das Dateisystem an die neue Größe angepasst
- mit dem Befehl *df -h /* kann man nun überprüfen, ob jetzt die gesamte SD-Karte für das Dateisystem genutzt wird

5.2.2 Geräteeinrichtung und Treiber

In diesem Abschnitt wird dargestellt, wie die Geräte und deren Treiber installiert und konfiguriert werden.

1 Installation der WLAN-Treiber[11]

- mit Hilfe des Befehls *lsusb* kann nachgeschaut werden, welche USB-Geräte angeschlossen sind; hierbei wird beim WLAN-Adapter auch der Geräte-Hersteller angezeigt
- im Falle des in dieser Arbeit genutzten Sticks handelt es sich um ein Gerät der Firma Ralink
- mit der Suche nach "Ralink" auf der Webseite *packages.debian.org* kann das passende Paket[12] für Debian 6 "Squeeze" mit Hilfe eines anderen Rechners gefunden und heruntergeladen werden
- nach dem Herunterladen des Paketes, kann es mit Hilfe eines USB-Sticks auf die SD-Karte des Raspberry Pi kopiert werden
- danach wird das Paket mit dem Befehl
  sudo dpkg -i firmware-ralink_0.28+squeeze1_all.deb im Verzeichnis des Paketes installiert

2 Kofiguration des WLAN[11]

- mit dem Befehl *sudo nano /etc/wpa.conf* wird die Datei geöffnet, in der die WLAN-Einstellungen vorgenommen werden
- am Ende dieser Datei, müssen Folgendes eingetragen werden:

  network={
  ssid= „YOUR-SSID“
  proto=RSN
  key_mgmt=WPA-PSK
  pairwise=CCMP TKIP
  group=CCMP TKIP
  psk= „WPA-PASSWORD“
  }

- für YOUR-SSID und WPA-PASSWORD müssen selbstverständlich die passende WLAN-ID und Passwort eingegeben werden, danach werden die Änderungen mit STRG-O gespeichert und der Editor mit STRG-X beendet
- mit dem Befehl *sudo nano /etc/network/interfaces* wird nun eine weitere Konfigurationsdatei geöffnet
- an das Ende dieser Datei werden Folgende Zeilen geschrieben:

  auto wlan0
  iface wlan0 inet dhcp
  wpa-conf /etc/wpa.conf
  }

- auch diese Änderungen mit STRG-O speichern und den Editor mit STRG-X schließen
- danach muss der Raspberry Pi mit dem Befehl *sudo reboot* neu gestartet werden; danach sollte er sich sofort im WLAN anmelden

3 Installation der USBDrDAQ-Messplatine

- unter der Adresse *http://www.picotech.com/support/topic10959.html*[15] die neuesten Treiber herunterladen
- das Archiv in einem Verzeichnis entpacken
- mit den Befehlen
  sudo dpkg -i libusbpico_1.0-1_armel.deb und
  sudo dpkg -i USBDrDAQlinux1_0.9-1_armel.deb
  werden die Treiber für die Platine installiert
- mit dem Befehl *sudo groupadd pico* wird eine neue Nutzergruppe namens “pico“ erstellt

- mit dem Befehl *sudo usermod -a -G pico pi* wird der Nutzer “pi“ der neuen Nutzergruppe hinzugefügt
- mit dem Befehl *sudo echo ’+ATTRS{idVendor}==“0ce9“ , MODE= “664“ ,GROUP=“pico“ ’ >/etc/udev/rules.d/95-pico.rules* wird eine neue UDEV-Rule für die Nutzung der Messplatine hinzugefügt
- mit dem Befehl *./usbtest* kann das Testprogramm gestartet werden, in dem geprüft wird, ob das Gerät korrekt angeschlossen, installiert und die Nutzergruppen passend gesetzt sind

4 Installation des GPS-Empfängers

- da dieses Gerät sich als serielles Gerät an der USB-Schnittstelle anmeldet, muss es nicht weiter konfiguriert werden

5.2.3 Das Datenbank-Management-System

Da das Datenbank-System eine relationale Datenbank enthalten sollte, welche später auch für Bearbeitung und Auswertung zur Verfügung stehen sollte, musste ein SQL-konformes System ausgewählt werden. Anfangs war es eine Überlegung MySQL zu verwenden. Allerdings stellt dieses einen zu großen und zu komlexen Deamon dar, welcher im Hintergrund läuft und viele Ressourchen benötigt. Für diese Arbeit wurde daher SQLite ausgewählt. Die Besonderheit dieses Systems besteht darin, dass eine Datenbank immer durch eine Datei repräsentiert wird, in welcher die einzelnen Tabellen der Datenbank abgespeichert sind. Somit könnten die Datenbankdateien nach einem Durchlauf des Messprogrammes auch auf einen anderen Rechner kopiert und dort ausgewertet werden.

Unter Debian-Linux gibt es für dieses System auch eigene, Versionen die sich wie folgt installieren lassen:

- mit dem Befehl *sudo aptitude* wird der Paketmanager des Systems gestartet
- es wird nach dem Paket “sqlite3“[16] gesucht und per Druck auf + hinzugefügt
- dann wird nach dem Paket “sqlitebrowser“[17] gesucht und ebenfalls per + hinzugefügt; dieses Paket beinhaltet eine grafische Oberfläche, was Erstellung und Bearbeitung von Datenbanken vereinfacht
- danach wird nach dem Paket “libsqlite3-dev“[16] gesucht und dieses dann auch mit + hinzugefügt; dieses Paket beinhaltet die Schnittstellen für die Programmierung
- durch das Drücken von g und Bestätigung werden die ausgewählten Pakete installiert

Der nächste Schritt war die Entwicklung einer Struktur für die Datenbank. Hier sollte es für jedes Messgerät eine Tabelle geben, in welche die Werte zusammen mit einem Zeitstempel eingetragen werden. Auch sollte es eine Möglichkeit geben, Einsätze und entsprechende Einsatz-IDs einzutragen, Zeitpläne (Schedules) einzutragen und diese Schedules einem Einsatz und einem Messgerät zuzuordnen.
Anhand des folgenden EER-Diagramms, welches mit Xerdi[18] erstellt wurde, soll dies noch einmal verdeutlicht werden:

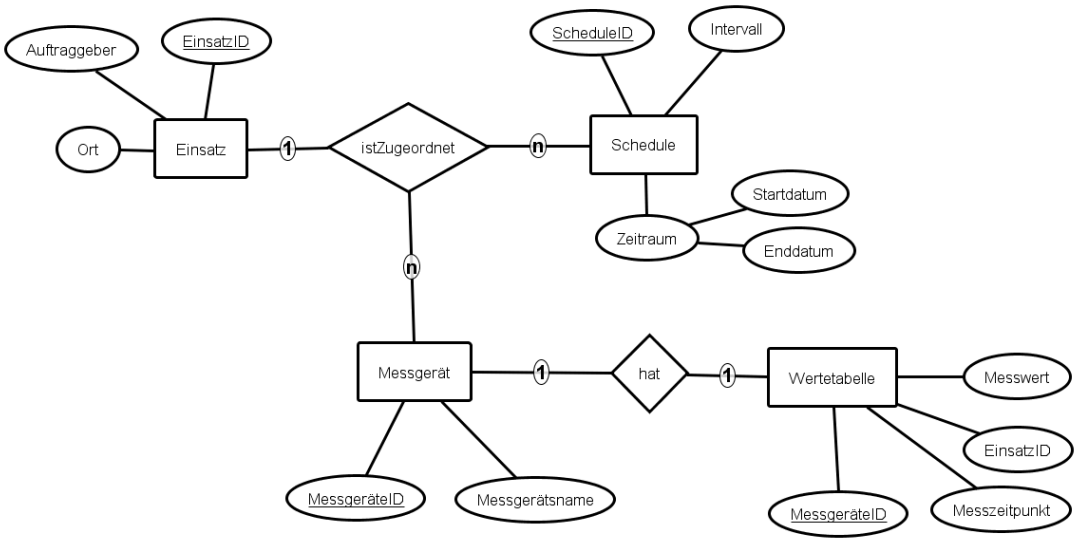

Abbildung 11: *EER-Datenbankmodell*

Mit Hilfe dieses Datenbanksystems wurde dann die Datei “wasserstation.db“ erstellt. Anhand des Datenbankmodells wurden dann mit dem SQLite-Browser die folgenden Tabellen erstellt:

Einsaetze:

EID	Ort	Auftraggeber

Hier werden die Einsätze mitsamt Ort und Auftraggeber abgespeichert. EID hat dabei ein INT-Format und Ort und Auftraggeber ein VARCHAR(20)-Format.

Schedules:

SID	von	bis	Intervall

Hier werden die Schedules abgespeichert. EID und Intervall haben ein INT-Format, wobei Intervall in Minuten angegeben wird. Von und bis haben beide das DATETIME-Format.

Messgeraete:

MgID	MgName

Hier sind die Messgeräte angegeben. EID ist im INT-Format und MgName im VARCHAR(20)-Format[21].

SIDtoEIDtoMgID:

EID	MgID	SID

Hier erfolgt über die IDs die Zuordnung von Einsätzen zu Schedules zu Messgeräten. Alle IDs sind im INT-Format.

SpannungsSensor:

EID	MessZP	SWert

TempSensor:

EID	MessZP	TempWert

pHSensor:

EID	MessZP	pHWert

GPSSensor:

EID	MessZP	Laengengrad	LaengengradAusrichtung	Breitengrad	BreitengradAusrichtung

In diesen Tabellen werden vom Messprogramm die Messwerte und Zeitstempel eingetragen. Messwerte haben hierbei das DECIMAL-Format[19], Einsatz-IDs das INT-Format[19] und Zeitstempel haben das DATETIME-Format[20] der SQL-Datenbanksprache.

5.2.4 Das Messprogramm

Das Messprogramm mit dem Namen “Pruefstation“ ist das Herzstück dieses Projektes und war Teil der Entwicklungsarbeit dieses Projektes. Das Programm wurde in C geschrieben. Ausschlaggebend hierfür war einerseits die Mächtigkeit von C, andererseits sind bei allen externen Programmen (SQLite) oder Geräten (serieller GPSSensor, DrDAQ-Platine) Schnittstellen für C vorhanden.
Im Folgenden soll seine Funktionsweise und sein Aufbau grob dargestellt werden, die Quellenverweise sind die Beispiele, an denen sich für diese Entwicklung orientiert wurde. Genauere Daten zur Funktionsweise sind dem Quellcode des Programmes zu entnehmen, welcher auf der CD mit der Bachelorarbeit beiliegt.

Das Programm enthält im Quellcode folgende Funktionen:

static int callback(void *NotUsed, int argc, char **argv, char **azColName) Diese Funktion ist nötig bei dem Aufruf von Datenbankzugriffen[26]. Wenn bei diesem Zugriff Werte zurückgegeben werden, werden sie durch diese Funktion auf dem Bildschirm ausgegeben.

void ledGruen(void) Diese Funktion ändert die Farbe der LED auf der DrDAQ-Platine zu grün. Sie bedient sich dabei der C-Funktionen aus der Bibliothek der DrDAQ-Platine.

void ledBlau(void) Diese Funktion ändert die Farbe der LED auf der DrDAQ-Platine zu grün. Sie bedient sich dabei der C-Funktionen aus der Bibliothek der DrDAQ-Platine.

void temperaturEintrag(int EID) Dieser Funktion wird eine Einsatz-ID übergeben. Zuerst wird hier auf dem initialisierten *mutex* ein lock ausgeführt[25], damit zu einer bestimmten Zeit nur diese Funktion ausgefüht werden kann. Danach wird der Temperaturwert über eine Funktion der Bibliothek der Messplatine gemessen (*UsbDrDaqGetSingle*[5]). Danach wird die aktuelle Zeit an ein *time_t*-Objekt[31] übergeben und in ein entsprechendes *struct tm*-Objekt umgewandelt. Mit der Hilfe dieses Objektes wird dann mit Hilfe der *sprintf*-Funktion ein SQL-konformer INSERT-Anfragestring für die Datenbank erstellt. Dieser String enthält dann die Einsatz-ID, den Messzeitpunkt und den gemessenen Temperaturwert. Danach wird mit Hilfe eines Befehls der SQLite-Bibliothek[27] dieser Anfrage-String ausgeführt und die Werte in die Tabelle TempSensor eingetragen. Danach wird der lock auf dem mutex wieder aufgehoben.

void pHEintrag(int EID) Diese Funktion funktioniert ähnlich wie die Funktion *temperatur-Eintrag*. Eine Einsatz-ID wird übergeben, auf dem initialisierten *mutex* ein lock ausgeführt, der pH-Wert gemessen und beides wird zusammen mit einem Zeitstempel in einen Anfragestring kopiert. Dieser trägt dann die Werte in die Tabelle pHSensor ein. Danach wird auch hier der lock auf dem *mutex* aufgehoben.

void voltEintrag(int EID) Auch diese Funktion funktioniert ähnlich wie die Funktionen *temperaturEintrag* und *pHEintrag*. Eine Einsatz-ID wird übergeben, auf dem initialisierten *mutex* ein lock ausgeführt, der Spannungs-Wert gemessen und beides wird zusammen mit einem Zeitstempel in einen Anfragestring kopiert. Dieser trägt dann die Werte in die Tabelle SpannungsSensor ein. Danach wird auch hier der lock auf dem *mutex* aufgehoben.

void GPSEintrag(int EID) Auch dieser Funktion wird eine Einsatz-ID übergeben. Danach läuft der Zugriff auf das Gerät anders ab als in den vorherigen Messfunktionen, da *GPSEintrag* auf den GPS-Sensor und nicht auf die DrDAQ-Platine zugreift. Da es sich bei dem GPS-Sensor um ein serielles Gerät handelt, wird es erst einmal initialisiert und danach den Spezifikationen des Herstellers gemäß eingestellt[6] (Baudrate etc.[22]). Danach wird mit einem char-Array ein String erstellt und mit Hilfe eines *read* auf dem GPS-Sensor werden dann die aktuellen Positionsdaten in diesen String eingefügt.Danach wird das Gerät wieder geschlossen.
Danach muss der Rückgabe analysiert werden. Mit Hilfe der Funktionen *strstr()* und *strtok()* wird in diesem String geprüft, ob die Positionsdaten im GGA-Protokoll[23] vorhanden sind und danach werden diese Daten in jeweils einen String für Längen- und Breitengrad sowie deren Ausrichtungen eingefügt.
Danach wird auch hier wieder mit Einsatz-ID und aktueller Zeit ein Anfragestring erstellt, welcher die Werte in die Tabelle GPSSensor einfügt.

void miss(int param1, int param2) Diese Funktion, ruft je nachdem welche Parameter mit der Hilfe von Einsatz-ID und Messgeräte-ID übergeben wurden, die entsprechende Mess-Funktion auf und übergibt ihr dabei die Einsatz-ID.

void* funktionsHandle(void *arg) Diese Funktion wird an die in der *main*-Funiktion erstellten *threads*[24], zusammen mit einem *struct* übergeben, welches die Werte zu Einsatz-IDs, Messgeräten und Schedules enthält. Danach werden mit Hilfe der Schedule-ID und einer SQLite-Funktion zum lesen der Datenbank die nötigen Daten eines Schedules kopiert. Mit Hilfe dieser Daten wird daach geprüft, ob der Startzeitpunkt für die Messungen bereits da ist. Wenn nicht, wird gewartet und immer wieder geprüft, wenn ja, werden die Messungen des mit der Messgeräte-ID festgelegten Gerätes eingeleitet und im Intervall

wiederholt. Dabei wird immer wieder geprüft, ob der Endzeitpunkt erreicht wurde, sobald dieser erreicht ist, werden die Messungen abgebrochen.

int main (int argc, char *argv[]) Diese Funktion steuert steuert wie in C üblich alle anderen Funktionen im Programm. Über den Start des Programms mit Parametern wird im *argv[]-Array die Einsatz-ID übergeben. Am Anfang wird die DrDAQ-Platine, die Datenbankdatei sowie ein *mutex*-Objekt initialisiert[25]. Danach wird die korrekte Initialisierung überprüft. Ist alles korrekt initialisiert wird die LED auf der DrDAQ-Platine auf grün umgeschaltet (so könnte man auch ohne Bildschirm bei automatischem Programmstart am Gewässer erkennen, dass alles korrekt läuft), wenn nicht, wird das Programm geschlossen. Danach werden die Zuordnungen von Schedule-ID, Einsatz-ID und Messgeräte-ID anhand der übergebenen Einsatz-ID aus der Datenbank gelesen[28][29]. Danach wird für jede Zuordnung ein *thread* erstellt[24], welcher dann mit den übergebenen IDs und der *funktionsHandle*-Funktion gestartet wird. Wenn alle *threads* beendet sind, wird das Programm beendet.

5.3 Bilanz

Nachfolgend soll kurz zusammengefasst werden, welche der formulierten Anforderungen an die Station erfüllt wurden. Auch sollen die bis jetzt angefallenen Materialkosten aufgezählt werden, um eine kleine Prognose erstellen zu können, wie sich die Kosten für die Station entwickeln.

5.3.1 Erfüllte Anforderungen

Folgende Anforderungen an die Station wurden erfüllt:

1 funktionale Anforderungen

- die Messtation soll mobil und tarnbar sein, daher muss
  - die Messtation möglichst klein sein
  - die Messtation tragbar sein
- die Parameter für die Messungen (Zeitpunkte, Zeitabstände, Messwerteauswahl) sollen einstellbar sein
- die Messwerte sollen persistent, z.B. in einer Datenbank abgelegt werden
- der Prototyp soll zumindest pH-Wert, Temperatur und die spezifische Leitfähigkeit messen können
- bei Einschalten oder Abschalten der Station soll es eine Rückmeldung (z.B. über eine LED) an den Nutzer geben

2 technische Anforderungen

- die Station soll auf Hardware- und Softwareebene bezüglich weiterer Sensoren und deren Messwerte leicht erweiterbar sein
- der Computer der Station soll (nach Aufgabenstellung) ein Raspberry Pi sein
- das Betriebsystem der Station muss (durch den Einsatz des Raspberry Pis) Linux sein

3 Benutzungsschnittstellen

- der Raspberry Pi soll per Anschluss an Bildschirm, sowie Maus und Tastatur konfigurierbar sein
- eine Rückmeldemöglichkeit (z.B. eine LED) soll beim Ein/Auschalten gegeben sein

4 Qualitätsanforderungen

- die Station soll die Werte zuverlässig und mit möglichst geringen Abweichungen und mit möglichst geringen Messfehlern messen

Diese Aufzählung zeigt, dass nicht alle Anforderungen umgesetzt werden konnten. Die Energieversorgung, ein wasserdichtes Behältnis, sowie das automatische Starten des Messprogrammes wurden nicht umgesetzt. Gründe dafür liegen einerseits im Zeitaufwand, den diese Arbeit hatte, andererseits auch in den Problemen, die bei der Entwicklung dieser Station aufgetreten sind. Informationen dazu werden im Abschnitt *Während der Implementierung aufgetrene Probleme* näher erläutert.
Allerdings wurden die wesentlichen Funktionalitäten und Anforderungen umgesetzt. Die Station kann jetzt, nach vorheriger Konfiguration, selbstständig und nach vorgegebenen Zeitplänen, Messungen in einer Flüssigkeit durchführen und aufzeichnen. Wird das Messprogramm ausgeführt, ist mit Hilfe der LED auf der DrDAQ-Platine auch ohne Bildschirm sichtbar, ob das Programm noch korrekt ausgeführt wird oder nicht. Sind die Messungen durchgeführt, können die Datenbank-Dateien auch auf anderen Rechnern ausgewertet werden.

5.3.2 Kosten

Folgende Kosten sind bei der Entwicklung der Station bis zum aktuellen Entwicklungspunkt angefallen:

- das pico DrDAQ pH Logger Set: 200,15 €[4]
- ein RaspberryPi Modell A: 31,79 €[32]
- ein Holux GR-213 GPS-Empfänger: 59,95 €[33]
- ein D-Link DWL-G122 G WLAN 54 MBit/Sek USB Adapter: 14,28 €[7]

Hinzu kommen etwa 20,00 € für einen aktiven USB-Hub und etwa 15,00 € für eine SD-Karte. Die Kosten für Eingabegeräte und Bildschirm werden nicht mit aufgeführt, da sie nicht in die Station verbaut werden sollen. Die Kosten für die Software sind im Falle der DrDAQ-Platine bereits im Paket inbegriffen, die restliche Software (Betriebssystem, Datenbanksystem, Messprogramm) sind entweder frei verfügbar oder wurde im Rahmen der Arbeit selbst entwickelt.
Somit liegen die Gesamtkosten (ohne Entwicklungskosten) bis jetzt bei 341,17 €, um es rund zu machen etwa bei 350 €. Somit ist auch die Anforderung, die Kosten gering zu halten, bis zu diesem Punkt der Entwicklung erfüllt.

5.4 Während der Implementierung aufgetrene Probleme

An dieser Stelle sollen einmal die größten Probleme aufgezeigt werden, welche bei der Entwicklung und Implementierung aufgetreten sind.

5.4.1 Treiberprobleme

Anfangs waren für das Projekt ein Raspberry Pi Model B und das aktuelle Standardbetriebsystem *Raspbian*[14], welches auf Debian 7 “Wheezy“ basiert, vorgesehen. Da der Raspberry Pi vorgegeben war, war die DrDAQ-Platine die ideale Ergänzung. Bei allen Händlern sowie auf der Herstellerseite wurde die Kompatibilität der Platine mit dem Raspberry Pi beworben. Auch ein Blog hatte bereits gezeigt, wie leicht man mit dem Raspberry Pi und dieser Platine arbeiten könnte[40]. Auch die C-Schnittstelle machte diese Platine ideal.
Bei der Treiberinstallation kam jedoch die Ernüchterung. Die Treiber ließen sich nicht installieren. Nach einiger Recherche war klar, dass die Treiber nur mit Debian 6 “Squeeze“, der Betriebssytemversion vor *Raspbian*, kompatibel waren. Die Gründe hierfür liegen in der hard-float-Berechnung die in *Raspbian* neu implementiert wurde. Auch die Anfragen an den Hersteller ergaben nur, dass es in naher Zukunft keine Treiber für Raspbian geben werde. Somit musste also Debian 6 verwendet werden, was zu weiteren Problemen führte.

5.4.2 Betriebsystemprobleme

Nachdem klar war, dass Debian 6 verwendet werden musste, war die nächste Frage:„Woher bekommt man das alte System noch?“ Das Problem hier war die Verfügbarkeit. Alle Webseiten die einen Download für dieses Betriebssytem anboten, verlinkten immer wieder auf die Homepage der Raspberry Pi Foundation[14]. Diese jedoch bietet dieses Betriebssystem seit dem Erscheinen von *Raspbian* nicht mehr an. Daher verging auch hier wieder einige Zeit, bis nach einiger Suche eine passende Version bei einem Filehoster[30] gefunden wurde. Danach ging es gleich mit den nächsten Problemen weiter.

5.4.3 Probleme mit dem Raspberry Pi Modell

Das erste Modell des Rasberry Pi war ein Modell B mit 512 Megabyte Arbeitsspeicher. Leider ließ sich mit diesem Modell das neue Betriebssystem auf Debian 6-Basis nicht starten. Nach einem Nachmittag des Ausprobierens wurde festgestellt, dass dieses Betriebssytem auf einem Modell A Raspberry Pi funktioniert. Somit musste dann auch der Raspberry Pi gegen ein Modell A ausgetauscht werden.

5.4.4 Probleme mit der Uhr

Da der Raspberry Pi keine Echtzeituhr hat, muss entweder die Zeit bei jedem Start aktuell gesetzt werden, oder es wird eine externe Uhr angebaut. Da die erste Möglichkeit im freien Feld unpraktisch ist, sollte eine externe Echtzeit-Uhr eingebaut werden.
Auch hier gab es eine Reihe von Uhren, die man verwenden konnte. Am Ende wurde eine Uhr ausgewählt welche bereits in einem Issue des MagPi - Onlinemagazins empfohlen wurde[36]. Die Uhr sollte auf den GPIO-Pins des Raspberry Pi angebracht werden und mit Treibern installiert werden. Jedoch waren hier die Treiber nicht mit Debian 6, sondern nur mit *Raspbian* kompatibel. Weitere Recherche im Internet ergab, dass es für Debian 6 keine Möglichkeit gibt, eine Uhr einzubauen, welche nicht mit einem viel zu hohen Zeitaufwand verbunden wäre. Der Grund dafür liegt in den I2C-Controllern, welche für jede Echtzeit-Uhr benötigt werden. Für diese sind unter Debian 6 keine Treiber vorhanden, erst mit *Raspbian* sind diese Treiber nativ verfügbar. Daher wurde die Uhr als nebensächlicher Teil des Projektes auf Eis gelegt.

6 Test

Nachdem das Messprogramm fertig geschrieben war, konnte mit einem größer angelegten Test begonnen werden. Im Folgenden werden die beiden durchgeführten Tests dargestellt und ausgewertet. Dabei werden auch Schwächen, die die Messtation noch hat, aufgezeigt und mögliche Lösungsvorschläge aufgezeigt.

6.1 Testaufbau

Folgendes Bild zeigt einmal den Testaufbau:

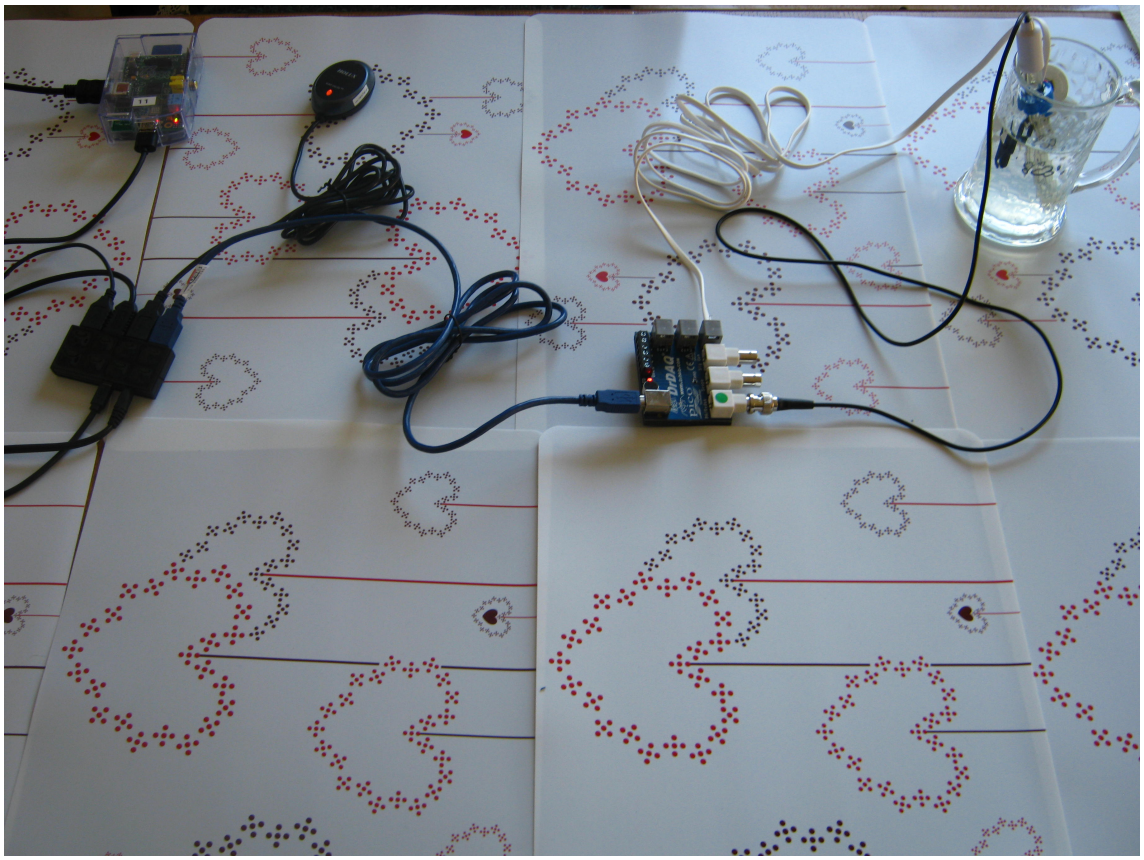

Abbildung 12: *Testaufbau*

Wie auf dem Bild zu sehen, sind für diesen Aufbau der Raspbery Pi, die GPS-Maus, die DrDAQ-Platine sowie Maus und Tastatur an dem aktuven USB-Hub angeschlossen. Dieser versorgt die Geräte auch mit Energie. Der USB-Hub selbst ist zusammen mit einem Strommessgerät an eine Steckdose angeschlossen. Dieses Strommessgerät soll später Aufschluss über den Energieverbrauch der Station geben. Der externe Temperatursensor sowie die pH-Elektrode sind an der DrDAQ-Platine angeschlossen und befinden sich in einem Glas mit Wasser.

Dieser Testaufbau wird auf folgender Grafik noch einmal schematisch dargestellt:

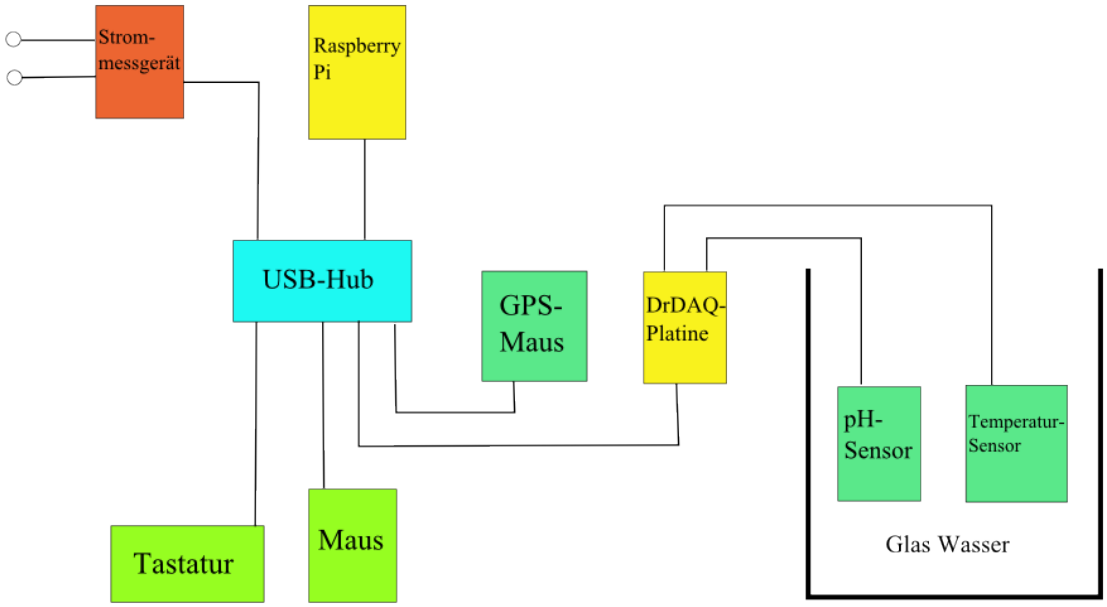

Abbildung 13: *schematischer Testaufbau*

6.2 Softwarekonfiguration

Nachfolgend wird dargestellt, wie die Datenbank für die beiden Testläufe konfiguriert wurde. Die nachfolgenden Tabellen zeigen die Einsätze, die Schedules und die Zuordnungen von Einsätzen, Schedules und Messgeräten.
Tabellen für Testlaufbau Eins:

Einsaetze:

EID	Ort	Auftraggeber
1	Testaufbau	Eike Bockhorst

Schedules:

SID	von	bis	Intervall
1	2013-11-30 16:00:00	2013-12-05 15:00:00	60
2	2013-11-30 16:15:00	2013-12-05 15:00:00	300
3	2013-12-01 02:00:00	2013-12-05 15:00:00	90
4	2013-12-03 02:00:00	2013-12-05 15:00:00	30
5	2013-12-01 00:15:00	2013-12-05 15:00:00	120
6	2013-11-30 22:20:00	2013-12-04 23:00:00	180

SIDtoEIDtoMgID:

EID	MgID	SID
1	1	1
1	2	1
1	3	1
1	4	2
1	2	3
1	2	4
1	1	5
1	3	6
1	1	6

Tabellen für Testlaufbau Zwei:

Einsaetze:

EID	Ort	Auftraggeber
2	Testaufbau2	Eike Bockhorst

Schedules:

SID	von	bis	Intervall
1	2013-12-01 16:00:00	2013-12-05 15:00:00	60
2	2013-12-01 16:15:00	2013-12-05 15:00:00	300
3	2013-12-02 02:00:00	2013-12-05 15:00:00	90
4	2013-12-03 02:00:00	2013-12-05 15:00:00	30
5	2013-12-02 00:15:00	2013-12-05 15:00:00	120
6	2013-12-01 22:20:00	2013-12-04 23:00:00	15

SIDtoEIDtoMgID:

EID	MgID	SID
2	1	1
2	2	1
2	3	1
2	4	2
2	2	3
2	2	4
2	1	5
2	3	6
2	1	6

6.3 Testablauf Test Eins

Nachdem Testaufbau und Datenbankkonfiguration fertig gestellt waren, konnte der Test gestartet werden. Folgende Schritte wurden zum Start des Test unternommen:

- der Zähler für den Energieverbrauch am Strommessgerät wird auf 0 zurückgesetzt
- der Raspberry Pi wird gestartet, es wird auf den User pi eingeloggt
- mit dem Befehl *sudo date –set= "2013-11-30 15:30"* wird das Datum des Raspberry Pi auf das aktuelle Datum eingestellt
- das Messprogramm wird mit *./Pruefstation 1* um 15:30 Uhr am 30.11.2013 gestartet.
- sobald das Programm gestartet ist, wird abgewartet und von Zeit zu Zeit kontrolliert

Testanalyse Test Eins Nach dem Start des Versuchs lief, den Testausgaben auf dem Bildschirm zur Folge, alles normal. Auch nach einer Kontrolle 3 Stunden später lief immer noch alles nach Plan. Aber bei einer kurzen Kontrolle um 00:45 Uhr des 01.12.2013 musste festgestellt werden, dass das Programm aufgrund eines Segmentation faults (Schutz- oder Zugriffverletzung) abgebrochen worden war. Den Ausgaben des Bildschirms zur Folge, war das Programm um 21:15 Uhr in Begriff eine geplante Messung der aktuellen GPS-Position vorzunehmen. Dabei trat diese Zugriffsverletzung auf und das Programm wurde beendet. Die Testausgaben auf dem Bildschirm wiesen darauf hin, dass das Lesen der Daten der GPS-Maus fehlerhaft gewesen sein muss und bei der Erstellung des Anfrage-Strings für die Datenbank wurde die Zugriffsverletzung ausgelöst.
Dieser Fehler wurde dann mit einer kleinen Überprüfungsfunktion behoben, so dass ein zweiter Testlauf gestartet werden konnte.

6.4 Testablauf Test Zwei

Nachdem der Fehler, der bei Testablauf Eins festgestellt worden war, behoben war und ein neuer Testauftrag in die Datenbank eingetragen wurde, konnte ein neuer Testversuch unternommen werden.

- der Zähler für den Energieverbrauch am Strommessgerät wird auf 0 zurückgesetzt
- der Raspberry Pi wird gestartet, es wird auf den User pi eingeloggt
- mit dem Befehl *sudo date –set= "2013-12-01 14:50"* wird das Datum des Raspberry Pi auf das aktuelle Datum eingestellt
- das Messprogramm wird mit *./Pruefstation 2* um 14:51 Uhr am 01.12.2013 gestartet.

- sobald das Programm korrekt gestartet ist, wird von Zeit zu Zeit kontrolliert
- am 02.12.2013 um 16:30 werden dem Wasser im Glas ein paar Spritzer Zitronensaft hinzugefügt
- am 03.12.2013 um 12:45 werden dem Wasser im Glas weitere Spritzer Zitronensaft hinzugefügt
- bei der letzten Kotrolle am 05.12.2013 um 23:45 wird festgestellt, dass das Programm alle in der Datenbank vorgesehenen Zeitpläne abgearbeitet hat und das Programm daraufhin daraufhin planmäßig beendet wurde.
- danach konnte die Datenbank für die Analyse entnommen werden, der Stromzähler zeigte einen Verbrauch von 0,31kWh (Kilowattstunden) an.

6.5 Testanalyse Test Zwei

Nach dem Abschluss des Tests wurde dieser, aufgrund der in der Datenbank vorhandenen Einträge, analysiert. Die Datenbankdateien für beide Testläufe befinden sich auf der Bachelorarbeits-CD und können unter Windows mit der *Mozilla Firefox*-Erweiterung *SQLite-Manager*[34] oder mit dem Programm *SQLite Administrator*[35] ausgelesen werden. Unter Linux kann auch der SQLitebrowser, der hier im Projket auf dem Raspberry Pi zum Einsatz kam, verwendet werden.
Die Analyse wurde mit der *Mozilla Firefox*-Erweiterung *SQLite-Manager* durchgeführt. Im Wesentlichen ergab die Analyse Folgendes:

- Alle Messungen wurden wie geplant ausgeführt.
- Die Intervalle zwischen den Messungen sind immer 2 bis 5 Sekunden Länger als angegeben.
- Die Sensoren reagieren genau auf Temperatur- oder pH-Wertänderungen.
- In der Temperaturtabelle gab es zwischen den Werten kleine Schwankungen um 2 °C.
- In der pH-Tabelle gab es anfangs nur die temperaturbedingten Schwankungen. Der pH-Wert lag bei 7,93 pH, mit Schwankungen von 0,03 pH.
- nachdem der Zitronensaft zugegeben worden war, änderte sich der pH-Wert um knapp 5 pH auf 2,96 pH mit den gleichen Schwankungen wie vorher.
- nach der zweiten Zugabe von Zitronensanft, sank der pH-Wert um weitere 0,2 pH ab, diesmal erhöhten sich jedoch die Schwankungen auf einen Bereich von 0,08 pH.
- Die Spannungswerte, welche durch die pH-Elektrode gemessen wurden, verhalten sich den ganzen Test über ähnlich wie die pH-Werte.

- Weiterhin wurde festgestellt, dass die Messzeitpunkte in allen Tabellen, nicht gemäß des “YYYY-MM-DD HH:MM:SS“-Eingabeformates des DATETIME-SQL-Formates eingetragen wurden. Statt zum Beispiel “05“ in dem “DD“-Feld einzutragen, wurde nur “5“ eingetragen. Dies passierte immer bei Tages-, Minuten-, Stunden-, oder Sekundenwerten unter 10. Dies führte dazu, das diese Einträge im Programm *SQLite Administrator*[35] nicht angezeigt wurden.
- Das Strommessgerät zeigte für den zweiten Versuch einen Verbrauch von 0,31kWh an.

6.6 Schlussfolgerungen zur Analyse

Die Analyse lässt nun folgende Schlüsse zu:

- Die Temperaturschwankungen im Wasser wurden durch die Änderungen der Raumtemperatur des Versuchsraumes verursacht, welche die Wassertemperatur beeinflussen.
- Die kleinen Schwankungen im pH-Wert sind verursacht durch die Temperaturschwankungen, da die Temperatur den pH-Wert ebenfalls beeinflusst[37]. Dies gilt auch für die Spannungsschwankungen.
- Die Fehler bei der Eingabe des Datums in die Datenbank ist ein Programmierfehler. Da bei der Erstellung der Anfragestrings int-Werte aus dem *struct tm* Objekt die Minuten darstellen, fehlt bei Werten unter 10 die 0, die dem Wert vorangestellt werden muss, um dem DATETIME-SQL-Format zu entsprechen.
- Die Verlängerungen der Intervalle sind darin begründet, dass die Messungen an sich auch etwas Zeit in Anspruch nehmen, dadurch kommt bei dem momentan verwendeten Algorithmus zur Wartezeit noch die Messzeit hinzu. Auch kommt Wartezeit hinzu, da die Messungen nur möglich sind, wenn gerade kein anderer Thread Messungen durchführt und damit alle anderen mit dem *mutex*-lock sperrt.
- Die Messung des Strommessgerätes hat gezeigt, dass die Station sehr wenig Energie verbraucht hat. Somit lässt sich damit schon schließen, dass die Station, sollte sie einmal für einen Außeneinsatz verbaut werden, auch über längere Zeit (ca. eine Woche) z.B. mit einer kleinen Autobatterie betrieben werden kann. Rein rechnerisch wäre dies möglich, wenn man zum Beispiel eine Batterie mit 100Ah (Amperestunden) hat. Der Raspberry Pi , welcher eine Leistungsaufnahme von etwa 500 mA hat, könnte damit etwa 200 Stunden lang betrieben werden.

6.7 Mögliche Verbesserungen

Die Schlussfolgerungen zur Analyse lassen Raum für Verbesserungen, die an der Station vorgenommen werden können. Diese bestehen im Wesentlichen aus Verbesserungen im

Messprogramm.

Messzeitpunkt-Einträge Da in der Analyse festgestellt wurde, dass die Messzeitpunkte bei Einträgen in die Datenbank nicht immer dem DATETIME-SQL-Format entsprechen, müsste für alle Sekunden-, Minuten-, Stunden-, Tages- und Monatswerte eine Überprüfung eingebaut werden. Diese prüft, ob einer dieser Werte unter 10 ist. Wenn dem so ist, muss bei der Erstellung des SQL-Anfragestrings eine 0 vor den entsprechenden Wert geschrieben werden.

Intervallverlängerungen Da sich die Intervalle durch die Messzeiten und die Wartezeiten, welche durch andere Threads verursacht werden, immer etwas länger sind als geplant, verschieben sich auch immer die Messzeitpunkte um ein paar Sekunden. Auf lange Sicht könnte dies dazu führen, dass sich Messzeitpunkte um ganze Minuten verschieben. Je nachdem wie genau diese Messzeitpunkte eingehalten werden müssen, könnte dies zu Problemen führen. Dies könnte behoben werden indem man ein anderes Konzept für die Messungen verwendet.
Ein solches Konzept könnte zwei *threads* und eine *Struktur* (*struct*) beinhalten. Eine solche *Struktur* könnte wieder wie im getesteten Programm, alle Daten wie Einsatz-ID, Schedule-ID, Messgeräte-ID enthalten. Mit Hilfe einer solchen *Struktur* könnte man dann zwei *threads* starten. Der erste *thread* baut mit Hilfe der *Struktur* eine Liste, in welchem die genauen Zeitpunkte für die Messungen stehen. Der zweite *thread* könnte aus diesee Liste dann die Zeitpunkte entnehmen und dann immer genau zum passenden Zeitpunkt messen. Der erste *thread* könnte währenddessen immer weiter neue Zeitpunkte in der Liste nachschieben, bis keine mehr vorhanden sind. Hat der zweite Thread alle Zeitpunkte abgearbeitet, ist diese Zuorndung beendet.
Auf diese Weise könnte es für jede Zuordnung von Messgerät zu Schedule zwei *threads* geben, die diese Aufgaben übernehmen. Noch besser wäre nur noch, wenn der erste *thread* aus allen Zuordnungen von Schedule und Messgerät eine einzige Liste mit Zeitpunkt der Messung und gefordertem Messwert erzeugt, welches der zweite *thread* abarbeitet. So könnte man die Arbeit auch von nur zwei *threads* bewerkstellen lassen.

Anhand folgender Grafik wird diese Idee noch einmal verdeutlicht:

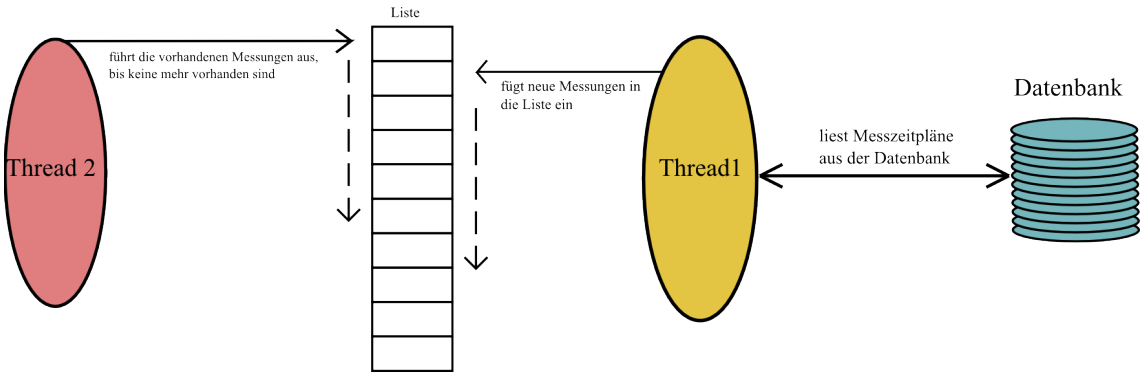

Abbildung 14: *alternative Softwarearchitektur schematisch*

7 Fazit/Ausblick

Diese Arbeit hat gezeigt, das es möglich ist, mit recht wenig Geld eine kleine mobile Wasserqualitätsprüfstation zu entwerfen. Diese Station misst nach vorgegebenen Zeitplänen die vorgeschriebenen Werte und trägt diese Messungen in eine Datenbank ein. Auch der Energieverbrauch der Station ist gering genug, um mobil über eine Batterie betrieben zu werden.
Die nächsten Schritte wären nun zum einen eine passende Energieversorgung für diese Station zu entwerfen, mit welcher die Station auch über einen Zeitraum von etwa einer Woche selbstständig Messungen durchführen kann. Zum anderen muss die Station auch wie in der Zielsetzung beschrieben, in ein wasserdichtes Behältnis verbaut werden, aus welchem nur die Sensoren herausragen. Weiterhin müssten Funktionen zum automatischen Start des Messprogrammes implementiert werden, eventuell ein Start- und Stoppknopf zum Ein- und Ausschalten der Station. Auch die Möglichtkeiten des Einbaus einer Echtzeit-Uhr sowie die Möglichkeit die Datenbank auf einem austauschbaren USB-Stick abzulegen, sollten jetzt genauer erarbeitet werden.

Weiteres Potential dieser Station liegt in ihrer Erweiterbarkeit. Es könnten leicht Sensoren ergänzt werden, der Code um entsprechende Funtkionen erweitert werden und in die Datenbank müssten diese neuen Sensoren dann eingetragen werden.

8 Anhang

Die genutzte Software (Betriebssystem, Treiber, etc.) sowie der Quellcode des Messprogramms ist auf der Bachelorarbeits-CD mitsamt einer Bedienungsanleitung für die Messtation enthalten. Ebenfalls darauf sind die Datenbankdateien, welche in den Versuchen als Messergebnisse fungierten.
Grafiken wurden in dieser Arbeit mit Visual Paradigm[38], Inkscape[39] oder Xerdi[18] erstellt.

Literatur

[1] USB DrDAQ Data Logger Specifications http://www.drdaq.com/specifications.html, Zugriff 08.12.2013

[2] DrDAQ Data Logger – Sensors and Accessories http://www.drdaq.com/sensors.html, Zugriff 08.12.2013

[3] USB DrDAQ Kit http://www.drdaq.com/products.html, Zugriff 08.12.2013

[4] pico DrDAQ pH Logger Set http://www.conrad.de/ce/de/product/128667/pico-DrDAQ-pH-Logger-Set-USB-Datenaufzeichnungsgeraet-Oszilloskop-Vorsatz-Data-Logge Zugriff 08.12.2013

[5] USB DrDAQ Data Logger Programmers Guide http://www.picotech.com/document/pdf/usbdrdaqpg.en-4.pdf, Zugriff 08.12.2013

[6] GR-213u GPS Receiver Specifications http://www.holux.com/JCore/en/products/products_spec.jsp?pno=273, Zugriff 08.12.2013

[7] Produktbeschreibung D-Link DWL-G122 G WIRELESS LAN 54 MBit/Sek USB ADAPTER http://www.amazon.de/gp/product/B000EXDQDQ/ref=oh_details_o04_s00_i00?ie=UTF8&psc=1#productDescription, Zugriff 08.12.2013

[8] RaspberryPi Spezifikationen http://de.wikipedia.org/wiki/Raspberry_Pi#Spezifikationen, Zugriff 08.12.2013

[9] Debian 6 “Squeeze“ Grundinstallation http://raspberrycenter.de/handbuch/debian-6-squeeze-grundinstallation, Zugriff 08.12.2013

[10] Raspberry Pi – Partition an größere Sd-Karte anpassen http://sparky0815.de/2012/05/raspberry-pi-partition-an-grosere-sd-karte-anpassen/, Zugriff 08.12.2013

[11] Raspberry Pi: WLAN einrichten http://www.tacticalcode.de/2013/02/raspberry-pi-wlan-einrichten.html, Zugriff 08.12.2013

[12] Package firmware-ralink http://packages.debian.org/search?suite=squeeze&searchon=names&keywords=ralink, Zugriff 08.12.2013

[13] Win32 Disk Image http://sourceforge.net/projects/win32diskimager/, Zugriff 08.12.2013

[14] Raw Images http://www.raspberrypi.org/downloads, Zugriff 08.12.2013

[15] Drivers for the Raspberry Pi http://www.picotech.com/support/topic10959.html, Zugriff 08.12.2013

[16] Package sqlite3 und Package libsqlite3-dev http://packages.debian.org/search?suite=squeeze&searchon=names&keywords=sqlite3, Zugriff 08.12.2013

[17] Package sqlitebrowser http://packages.debian.org/search?suite=squeeze&section=all&arch=any&searchon=names&keywords=sqlitebrowser, Zugriff 08.12.2013

[18] Xerdi http://www.informatik.uni-oldenburg.de/~nemesis/xerdi/, Zugriff 08.12.2013

[19] SQL Numerische Datentypen http://dev.mysql.com/doc/refman/5.1/de/numeric-types.html, Zugriff 08.12.2013

[20] SQL Datums- und Zeittypen http://dev.mysql.com/doc/refman/5.1/de/date-and-time-types.html, Zugriff 08.12.2013

[21] Die CHAR- und VARCHAR-Typen http://dev.mysql.com/doc/refman/5.1/de/char.html, Zugriff 08.12.2013

[22] Die serielle Schnittstelle http://www.netzmafia.de/skripten/hardware/Seriell/, Zugriff 08.12.2013

[23] Global Positioning System Fix Data (GGA) http://de.wikipedia.org/wiki/NMEA_0183#Global_Positioning_System_Fix_Data_.28GGA.29, Zugriff 08.12.2013

[24] Mit den POSIX-Threads programmieren http://openbook.galileocomputing.de/c_von_a_bis_z/026_c_paralleles_rechnen_004.htm, Zugriff 08.12.2013

[25] How to Use C Mutex Lock Examples for Linux Thread Synchronization http://www.thegeekstuff.com/2012/05/c-mutex-examples/, Zugriff 08.12.2013

[26] SQLite C/C++ Tutorial http://www.tutorialspoint.com/sqlite/sqlite_c_cpp.htm, Zugriff 08.12.2013

[27] One-Step Query Execution Interface http://www.sqlite.org/capi3ref.html#sqlite3_exec, Zugriff 08.12.2013

[28] sqlite3_get_table Tutorial http://www.proggen.org/doku.php?id=dbs:sqlite:libsqlite3:communicate:sqlite3_get_table, Zugriff 08.12.2013

[29] Convenience Routines For Running Queries http://www.sqlite.org/capi3ref.html#sqlite3_free_table, Zugriff 08.12.2013

[30] Raspberry Pi OS Debian (squeeze) Download http://www.netzwelt.de/software-chooser/27159_2-raspberry-pi-os.html, Zugriff 08.12.2013

[31] Zeitroutinen http://openbook.galileocomputing.de/c_von_a_bis_z/019_c_zeitroutinen_001.htm#mj0a8102e96a1297bd767ac7274e069f0f, Zugriff 08.12.2013

[32] RaspberryPi Modell Ahttp://www.amazon.de/Raspberry-Pi-Typ-A-256MB/dp/B00CFKPDAU/ref=sr_1_2?s=ce-de&ie=UTF8&qid=1386607084&sr=1-2&keywords=raspberry+pi+modell+a, Zugriff 09.12.2013

[33] Holux GR-213 GPS-Empfänger Conrad http://www.conrad.de/ce/de/product/378793/Holux-GR-213-GPS-Empfaenger, Zugriff 08.12.2013

[34] SQLite Manager Firefox https://addons.mozilla.org/de/firefox/addon/sqlite-manager/, Zugriff 09.12.2013

[35] SQLite Administrator http://www.heise.de/download/sqlite-administrator-1159579.html, Zugriff 09.12.2013

[36] The MagPi Issue 15 http://magpi.finalart.hu/The-MagPi-issue-15-en.pdf, Zugriff 10-12-2013

[37] Temperaturabhängigkeit pH-Wert http://de.wikipedia.org/wiki/PH-Wert#Temperaturabh.C3.A4ngigkeit, Zugriff 10.12.2013

[38] Visual Paradigm http://www.visual-paradigm.com/, Zugriff 10.12.2013

[39] Inkscape http://inkscape.org/en/, Zugriff 10.12.2013

[40] Using the DrDAQ with a Raspberry Pi http://www.designspark.com/blog/using-the-drdaq-with-a-raspberry-pi, Zugriff 11.12.2013

[Bild1] http://press.picotech.com/images-hi/usb-drdaq.jpg, Zugriff 08.12.2013

[Bild2] http://www.drdaq.com/images/drdaq-schematic.png, Zugriff 08.12.2013

[Bild3] http://www.conrad.de/medias/global/ce/1000_1999/1200/1280/1286/128668_BB_00_FB.EPS_1000.jpg, Zugriff 08.12.2013

[Bild4] http://www.conrad.de/medias/global/ce/1000_1999/1200/1280/1286/128670_BB_00_FB.EPS_1000.jpg, Zugriff 08.12.2013

[Bild5] http://www.pecino.com/blog/wp-content/uploads/2013/03/RS128-Raspberry_Pi_Model_A.jpg, Zugriff 08.12.2013

[Bild6] http://www.nudatech.com/image/cache/data/products/Raspberry_Pi/Raspberry_Pi_A-scheme-640x480-500x500.png, Zugriff 08.12.2013

[Bild7] http://ecx.images-amazon.com/images/I/41JNAua4t2L.jpg, Zugriff 08.12.2013

[Bild8] http://ecx.images-amazon.com/images/I/31UEDrTkWJL.jpg, Zugriff 08.12.2013