Entwicklung eines Low-Cost-USB-Datenloggers zur Nachbereitung naturwissenschaftlicher Praktika

Tobias Gutzler, Robert Kastl und Volkhard Nordmeier

Freie Universität, Fachbereich Physik, Arnimallee 14, 14195 Berlin
tobias.gutzler@fu-berlin.de, robert.kastl@fu-berlin.de, volkhard.nordmeier@fu-berlin.de

Kurzfassung

Auf Grundlage eines Mikrocontrollers wurde ein USB-Datenlogger entwickelt, der es auf einfachste Weise ermöglicht, Messdaten aus Praktikumsversuchen auf einem USB-Stick mit in die Nachbereitung zu nehmen. Dafür sind weder Rechner im Praktikum noch teure Interface-Systeme und die zugehörige Software nötig. Praktikumsversuche sind in er Regel ‚Interventionen', bei denen das Experiment im Fokus steht. Um eine Langzeitwirkung zu erzielen, werden Praktika sowohl ausführlich vor- als auch nachbereitet. Als Erweiterung bisher bekannter Interaktiver Bildschirmexperimente (IBE) wurde nun eine Möglichkeit geschaffen, die im Praktikum aufgenommenen Messdaten mitzunehmen und in entsprechende IBE einzubinden. So kann das Experiment – genau wie es im Praktikum abgelaufen ist – auf dem Bildschirm wiederholt bzw. nachvollzogen werden. Außerdem können die Messdaten zur Auswertung in Heimarbeit mit entsprechenden Programmen genutzt werden.
Im Rahmen dieses Beitrags soll der entwickelte USB-Datenlogger vorgestellt und an einem Beispielexperiment die Einbindung der Messdaten in ein IBE illustriert werden.

1. Motivation

Das Experimentieren hatte schon immer eine besondere Bedeutung in der Naturwissenschaft. Schaut man in die Physik, so sieht man auch heute noch, wie wichtig Experimente für die Erkenntnisgewinnung sind. So werden aufwendige Forschungseinrichtungen, auch auf internationaler Ebene, unterhalten, um mit Hilfe physikalischer Experimente Theorien zu stützen. Ein Beispiel für eine solche Einrichtung ist der Teilchenbeschleuniger CERN[1]. Hier wurde letztes Jahr die Existenz des Higgs-Bosons mit großer Wahrscheinlichkeit experimentell bestätigt. Auch heute werden also noch fundamentale Erkenntnisse experimentell gewonnen. Die Auswertung dieser Experimente ist ohne Computer gar nicht denkbar. Zu einer naturwissenschaftlichen Kompetenz zählt daher heute nicht mehr das Experimentieren allein, sondern auch die computergestützte Messwerterfassung und Weiterverarbeitung.

Nicht ohne Grund nehmen Experimente eine besondere Stellung in der naturwissenschaftlichen Ausbildung ein. Sie sollen eigene Erfahrungen ermöglichen, naturwissenschaftliche Denkweisen fördern und dabei einen Zugang zu naturwissenschaftlichen Arbeitsweise schaffen. Nicht zuletzt sollen sie aber die Möglichkeit bieten, naturwissenschaftliche Vorgehensweisen exemplarisch zu durchlaufen und Handlungsmuster eben dieser Domäne kennenzulernen [8, S. 4]. Da diese Handlungen heute jedoch weitgehend am Computer stattfinden, sollte auch in der Schule und besonders im naturwissenschaftlichen Praktikum der Hochschulen der Computer zur Auswertung mit einbezogen werden.
Da der Einsatz von Experimenten über dies auch noch motivieren soll [8, S. 4], und der Computer heutzutage zum integrativen Bestandteil des alltäglichen Lebens gehört, sollte dieser eben auch hier zum Einsatz kommen und den Lernenden die Gelegenheit geben, seine Vorteile zu nutzen.

Natürlich gibt es bereits Konzepte, die genau dies ermöglichen. Zu diesen gehören jedoch Interface-Systeme, die oft teuer und daher nicht immer zugänglich sind. Vor allem bleibt festzuhalten, dass diese Geräte in der Regel nicht in ausreichender Anzahl angeschafft werden können, damit jede Schülerin / jeder Schüler sie nutzen kann. Soll auch in der Schule ein zeitgemäßes Bild der Naturwissenschaft vermittelt werden, besteht folglich der Bedarf an kostengünstigen *Computergestützten Messwerterfassungssystemen (CMS)*, welche die Messdaten digital zur Auswertung am Computer zur Verfügung stellen.

Betrachtet man den Verlauf von Praktika oder auch Experimenten in der Schule, so kann man feststellen, dass es neben der Durchführung des Experiments auch eine Phase der Vorbereitung und eine der Auswertung und Nachbereitung gibt. Diese Phasen sind dabei keineswegs unwichtig, denn nur wenn auch diese adäquat umgesetzt werden, können sich die Funktionen von Experimenten voll entfalten [7, S. 203–264]. So ist für das Verständnis die Phase der Nachbereitung besonders wichtig, denn hier kann die Verbindung vom gesehenen Experiment zum abstrakten Symbolsystem der Physik hergestellt werden [7, S. 203–264]. Da der Versuch aber oft

[1] CERN: Conseil Européen pour la Recherche Nucléaire

schon vor einigen Tagen durchgeführt wurde, wenn Studierende oder Schüler/innen ihn zu Hause auswerten, wäre es sinnvoll, wenn sie diesen, so wie er in der Realität abgelaufen ist, noch einmal erleben oder anschauen könnten.

So gesehen besteht ein Bedarf an einem Low-Cost-Messsystem, welches es Lernenden ermöglicht, Messdaten in Experimenten selbst aufzunehmen, um diese anschließend auszuwerten und Experimente währenddessen wiederholend zu betrachten. Dies beschreibt das Vorhaben, das in diesem Beitrag dargestellt wird. Dazu wurden zunächst anhand didaktischer Vorüberlegungen Anforderungen ermittelt, nach denen darauf ein solches Messsystem (den USB-Datenlogger) entwickelt wurde. Um ein wiederholendes Ablaufen des Experiments zu ermöglichen, musste zusätzlich eine Möglichkeit gefunden werden, die Messdaten in ein Interaktives Bildschirmexperiment (IBE) einzubinden und es so wie im realen Fall ablaufen zu lassen. Die folgende Schemadarstellung zeigt die Grundidee, die dieser Arbeit zugrunde liegt:

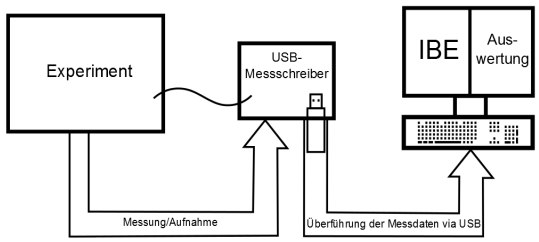

Abb. 1: Schematische Darstellung des Vorhabens

Wie in Abb. 1 zu sehen ist, kann ermöglicht werden, dass Daten aus einem Experiment aufgenommen und anschließend in einen Computer überführt werden. Im Computer stehen die Daten dann zur Auswertung zur Verfügung oder können in einem IBE zum wiederholenden Betrachten des Experiments genutzt werden. Da ein Computer für jeden Messplatz dem Low-Cost-Gedanken widersprechen würde, sollte die Datenaufzeichnung ohne Computer geschehen und die Daten entsprechend auf einem externen Datenträger gespeichert werden, der wiederum den Transport der Messdaten ermöglicht. Hier fiel die Wahl auf einen USB-Stick, da davon auszugehen ist, dass jeder Lernende über einen solchen verfügt.

2. Ziele und Anforderungen zur Entwicklung

Folgende Zielsetzungen liegen der Entwicklungsarbeit zu Grunde:

Ziel 1
: Schaffen eines kostengünstigen Angebots zur digitalen Messwerterfassung.

Ziel 2
: Es soll den Lernenden ermöglicht werden, selbstständig Messdaten aufzunehmen, auszuwerten und zu interpretieren.

Ziel 3
: Damit sollen die förderlichen Aspekte von MBLs nach Hucke [8] auch hier Ziele darstellen:
  a. Einfachstes Aufnehmen von Messdaten.
  b. Synchrones Erzeugen von Messdaten.
  c. Sofortiges Bereitstellen von Messdaten.

Ziel 4
: Ein weiteres Ziel ist das Einsparen von Zeit, die sonst bei der Aufnahme der Daten und bei der Eingabe dieser für eine Auswertung verloren gegangen wäre.

Ziel 5
: Damit verbunden ist das Ziel, zu fördern, Erfahrenes (also durch Experimente Erlebtes) mit theoretischem Wissen zu verbinden.

Ziel 6
: Es sollte eine Erleichterung des Übergangs von der Durchführungs- zur Auswertungsphase geschaffen werden.

Ziel 7
: Es ist sinnvoll, eine Auswertung am Computer zu ermöglichen, um so auch den Umgang mit entsprechenden Programmen zu lernen.

Ziel 8
: Es sollten damit einhergehend die Möglichkeiten verbessert werden, Gesetze aus Experimentierdaten selbst abzuleiten (auch in der Nachbereitung zu Hause).

Ziel 9
: Es sollen Möglichkeiten zur umfassenden Nachbereitung (auch zu Hause) verbessert werden, was besonders wichtig für ein nachhaltiges Lernen ist.

Ziel 10
: Es ist sinnvoll, die Intensivierungsfunktion von Experimenten durch das Ermöglichen vielfältiger Darstellungsformen zu verstärken.

Ziel 11
: Das Experiment soll auch in der Auswertung bzw. Nachbereitung in den Mittelpunkt gerückt werden.

Ziel 12
: Lernende sollen sich Ihren eigenen Experimentverlauf durch Einbindung in IBE wiederholend anschauen können.

Zum Teil direkt aus fachdidaktischen Überlegungen, vor allem aber aus den genannten Zielen ergeben sich folgende Anforderungen für die Entwicklung des Datenloggers:

Anforderung 1
: Aus dem ersten Ziel ergibt sich die Anforderung, dass es sich um ein Low-Cost-Gerät handeln sollte.

Anforderung 2
: Aus dem zweiten, dritten und vierten Ziel ergibt sich die Anforderung nach der Einfachheit in der Anwendung. Das Gerät muss also so gestaltet sein, dass eine Bedienung in kürzester Zeit erlernt werden kann und auch hinterher keinen besonderen Aufwand erfordert.

Anforderung 3
: Des Weiteren ergibt sich aus dem dritten Ziel, dass die Messdaten synchron aufgenommen und sofort zur Verfügung stehen sollten.

Anforderung 4
: Aus den Zielen vier bis zehn lässt sich ableiten, dass die Daten in einer Form zur Verfügung stehen müssen, die sie leicht transportierbar macht.

Anforderung 5
: Außerdem müssen die Messdaten aufgrund dieser Ziele einfach in andere Computerprogramme zu importieren sein.

Anforderung 6
: Aus den Zielen zehn bis zwölf ergibt sich, dass die Messdaten nicht nur so zur Verfügung stehen müssen, dass sie sich einfach in Auswertungsprogramme integrieren lassen, sondern auch in einem IBE zu verwenden sind.

Anforderung 7
: Daraus folgt als weitere Anforderung, dass die Aufnahmegeschwindigkeit so groß sein muss, dass ein flüssiges Bild entstehen kann.

Anforderung 8
: Da das Gerät nur einen geringen Mehrwert hätte, wenn es nur für einen Versuch zu nutzen wäre, ist eine weitere Anforderung, die universelle Einsetzbarkeit. Dazu gehört auch die Möglichkeit, mehrere Messwerte gleichzeitig aufnehmen zu können.

Anforderung 9
: Die Anwendung sollte unabhängig vom Internet sein [11].

Anforderung 10
: Es muss eine Wandlung analoger Messwerte in digitale Repräsentationen stattfinden.

Anforderung 11
: Da das Messgerät im schulischen Alltag oder auch im Praktikum einsetzbar sein soll, und dort durch viele Hände geht, ist die Robustheit eine weitere Anforderung.

Diese Anforderungen gehen damit weit über die von Bernshausen [3] vorgestellten Anforderungen an CMS hinaus:

1. Erweiterung der Möglichkeiten bei Aufnahme, Darstellung und Auswertung.
2. Die Benutzerfreundlichkeit, da ein CMS bei der Durchführung von Experimenten unterstützen soll.
3. Die Abstimmung auf unterrichtliche Aspekte [3, S. 14–15].

3. Verwendete Geräte

Physical Computing ist das Schlagwort, unter dem man die Verwendung von Elektronik zur Erstellung von Prototypen – so, wie es auch hier geschehen ist – versteht. Dazu gehört dann auch die Gestaltung interaktiver Objekte, die zur Steuerung und Regelung genutzt werden. Dabei werden Sensoren abgefragt und Aktoren angesteuert. Die Steuerung wird dabei innerhalb einer Software realisiert, deren Programm in einem Mikrokontroller abläuft [2, S. 3]. Diese Form der Steuerung ist (nach DIN 19226 T.5/02.94) als Speicherprogrammierbare Steuerung (SPS) zu charakterisieren [4, S. 250].
Die Erstellung von Prototypen war früher sehr aufwendig und erforderte nicht selten ein Wissen, über das nur Ingenieure verfügten. Dabei musste Elektronik eingesetzt und aufwendige Schaltkreise aufgebaut werden. Mikrocontroller bieten heute die Möglichkeit diesen Prozess zu vereinfachen und somit zu beschleunigen. Eine weit verbreitete Open-Source-Plattform für *Physical Computing*, basierend auf einem einfachen Input/Output-Board und einer die Sprache *Processing* implementierenden Entwicklungsumgebung, ist *Arduino*. Das Open-Source-Konzept ermöglicht das Selberbauen eines Mikrocontrollerboards sowie den preiswerten Kauf eines fertigen Boards.

Eine *Integrierte Entwicklungsumgebung* (*Integrated Development Environment – IDE*) steht kostenlos zum Download unter www.arduino.cc bereit [2, S. 1–3].

Damit sind bereits die zwei wesentlichen Komponenten eines *Arduinos* benannt: die Hardware und die IDE, also die Software. Daher wird im Folgenden ein kurzer Überblick über die Hardware, die bei einem *Arduino UNO* zur Verfügung steht, sowie die IDE gegeben. Außerdem wird für das Vorhaben eine Schnittstelle zu einem USB-Stick benötigt. Daher wird anschließend kurz das hier verwendete USB-Host vorgestellt.

Das Mikrocontrollerboard – Arduino UNO

Die Wahl fiel bei der hier vorgestellten Entwicklung auf das Mikrocontrollerboard *Arduino UNO*. Diese Wahl ist zum einen darin zu begründen, dass *Arduino* sehr gut zum hier vertretenen Open-Source-Gedanken passt und speziell das *UNO* ein besonders preiswertes Board ist. Die Hardware des hier verwendeten *Arduino UNO* ist somit als Mikrocontrollerboard, also einer bestückten Platine mit einem Schaltkreis, zu beschreiben, dessen zentrales Element ein Mikrocontroller ist. Speziell beim *Arduino UNO* ist dies ein *ATMEL ATMEGA 328* [2, S. 17–19].

Als die wichtigsten Elemente das Boards sind folgende zu nennen:

- Der *Mikrocontroller* ist in diesem Fall ein 28-beiniger *ATMEL ATMEGA 328*, der durch einfaches Herausziehen und Einstecken problemlos und kostengünstig bei Defekt ausgetauscht werden kann.
- *14 digitale Ein- und Ausgänge (Pins 0 – 13)*, von denen *6 (Pins 3, 5, 6, 9, 10 und 11)* auch als *analoge Ausgänge* genutzt werden können [2, S. 18].

- *6 analoge Eingänge (Pins A0 – A5)*, hinter denen sich AD-Wandler mit *Successiven Approximations Registern* (SAR) verbergen. Sie haben eine Auflösung von n=10bit, also 1024 Digitalschritte bzw. einen digitalen Wertebereich von Z=0 bis Z_{max}=1023 [2, S. 18; 1, S. 125]. Da die höchste anzulegende Spannung hier mit $U_{max}=U_{FSR}$[2]=5V angegeben ist, ergibt sich eine kleinste unterscheidbare Spannung von U_{LSB}[3]=4,88mV. Daraus ergibt sich dann ein Quantelungsfehler von ½U_{LSB}=2,44mV. Die Geschwindigkeit dieser SAR wird mit 76,9 kSPS[4] bzw. maximal mit 15 kSPS angegeben [1, S. 250].
- *Weitere Pins*, an denen Spannungen von 5V oder 3,3V und das zugehöriger Bezugspotential (Masse – ground – GND) abgegriffen werden kann.
- *Ein Reset-Taster*, der wie der Name schon sagt den Arduino zurücksetzt und das Programm des Mikrocontrollers neu startet (und so ein eventuell implementiertes Setup neu durchlaufen lässt).
- *Ein USB-Anschluss*, über den die Kommunikation mit einem Computer stattfindet, oder auch die Spannungsversorgung realisiert werden kann [2, S. 18].
- *Ein Gleichstromanschluss* für DC-Hohlstecker zur Spannungsversorgung. Wird diese Spannungsversorgung verwendet, schaltet der Arduino automatisch auf sie um. Wird sie nicht verwendet, so nutzt der Arduino die Spannungsversorgung via USB [2, S. 18].
- *Mehrere LEDs*, von denen eine die Betriebsbereitschaft des *Arduinos* anzeigt, zwei eine serielle Kommunikation anzeigen (RX und TX) und eine frei ansteuerbar ist.

Neben einer übersichtlichen, gut dokumentierten und kompakten Hardware bietet *Arduino* eine plattformübergreifende integrierte Entwicklungsumgebung (siehe Abb. 2), mit der auf einfachste Weise Programme geschrieben und auf den Mikrocontroller übertragen werden können. Bei der *Arduino* IDE handelt es sich um ein Programm, das auf einem Computer ausgeführt wird und so das Erstellen eines Programms für den Mikrocontroller ermöglicht [2, S. 203–264]. Dabei ist die IDE als „*Cross-Plattform Java Applikation die als ein Programmcode Editor und Compiler dient*“ [5, S. 3] zu charakterisieren. Da die *Arduino* Entwicklungsumgebung auf *Processing* basiert, – einer IDE, die für Künstler etc. entwickelt wurde, um so das Programmieren besonders einfach zu gestalten – ist die Programmierung schnell und einfach zu erlernen. Die Programmiersprache basiert dabei auf *Wiring*, einer Sprache, die der verbreiteten Programmiersprache C (und C++) ähnelt [2, S. 20; 5, S. 3].

In der Entwicklungsumgebung programmiert man sogenannte *Sketches*, die zunächst in die komplexere Sprache C übersetzt und anschließend zu einem avr-gcc-Compiler übertragen werden. Dieser kompiliert den Code dann in die Sprache des Mikrocontrollers (Maschinencode), in der dann das Programm im Controller abläuft [2, S. 20; 9, S. 10–11].

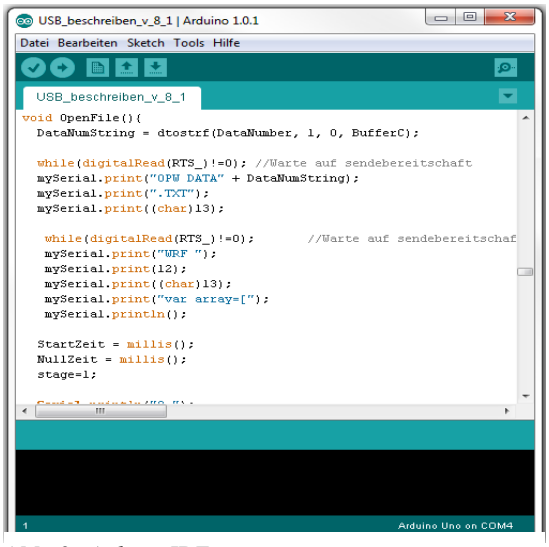

Abb. 2: *Arduino* IDE

Die *Arduino* IDE bietet Buttons mit denen der Quellcode des aufgerufenen Sketches zu kompilieren, oder auf angeschlossene Hardware (z.B. einen *Arduino UNO*) zu übertragen ist. Weitere Buttons gibt es, um einen neuen Sketch anzulegen, einen vorhandenen aufzurufen oder einen offenen Sketch zu speichern. Außerdem bietet die IDE einen seriellen Monitor, auf den man Daten vom Arduino ausgeben kann [9, S. 10–13].

Die Entwicklung eines Programms geschieht durch das Schreiben eines Sketches im dafür vorgesehenen Editor. „Sketch“ steht bei Arduino für das Programm[5]. Der Begriff bedeutet so viel wie Skizze oder Entwurf und wurde aufgrund der Zielgruppe (Künstler und Designer) eingeführt [9, S. 24].

Der USB Host

Innerhalb unseres Vorhabens sollen Experimentierdaten von dem zu entwickelnden Datenlogger auf einen Computer zur Verfügung gestellt werden. Unter Berücksichtigung der vierten Anforderung, dass die Daten in einer Form zur Verfügung stehen müssen, die sie leicht transportierbar macht, und der neunten Anforderung, der Unabhängigkeit vom Internet, musste ein Weg ermöglicht werden, der diesen Transport unkompliziert ermöglicht. Hier kamen zwei Möglichkeiten in Betracht: eine direkte Übertragung auf einen Computer am Versuchsort oder eine Übertragung auf einen Datenträger, der

[2] Full Scale Range (FSR)
[3] Least Significant Bit (LSB)
[4] kilo samples per second (kSPS)

[5] Die beiden Begriffe „Sketch“ und „Programm“ werden daher im Folgenden synonym verwendet.

anschließend die Daten für einen beliebigen, lokal nicht gebundenen, Computer bereitstellt.

Der Ansatz des direkten Übertragens auf einen Computer (ohne Internet) würde bedeuten, dass an jedem Experimentierplatz ein Computer zur Verfügung stehen müsste, was wiederum einen erhöhten Kostenaufwand mit sich bringen würde und damit der ersten Anforderung, nach einer Low-Cost-Lösung, nicht gerecht werden würde. Daher wurde hier nach einem geeigneten Speichermedium gesucht, welches mit einem Arduino zu beschreiben ist.

Zur Auswahl stand daher das Beschreiben von SD- bzw. Mikro-SD-Karten oder USB-Sticks. Das Schreiben auf SD-Karten ist dabei die einfachere Lösung, da es für Arduino auch ein *Ethernet Shield* gibt, welches eine SD-Schnittstelle enthält. Auch Bibliotheken und geeignete Treiber sind zu diesem Zweck leicht zu finden. Zwei triftige Gründe sprechen jedoch für eine Verwendung von USB-Sticks als Speichermedien:

1. Es ist davon auszugehen, dass mehr SchülerInnen und Studierende einen eigenen USB-Stick alltäglich mit sich führen als eine SD-Karte.
2. Es ist nicht zwangsläufig davon auszugehen, dass die Lernenden an ihrem Computer zu Hause einen Kartenleser haben. Sehr wahrscheinlich ist aber, dass diese USB-Ports besitzen.

Aus diesen beiden Gründen – die auch durch die Anforderung vier gestärkt werden – fiel die Entscheidung zugunsten des USB-Sticks aus, wenngleich die Umsetzung dadurch erschwert wurde. Der USB-Stick soll dabei – mit Blick auf Anforderung sechs und neun – gleichzeitig Träger des IBE werden.

Es musste nun also eine geeignete Schnittstelle zwischen Arduino und USB-Stick gefunden werden, die ein Beschreiben des Sticks ermöglicht.

Nach Recherche bei verschiedenen Anbietern fiel die Wahl zunächst auf das *USB Host Shield für Arduino* von *ITEAD*. Dabei handelt es sich um einen Kontroller, der bereits als Stapelshield ausgeführt ist und somit direkt auf das *Arduino UNO* Board aufgesteckt werden kann. Da der Hersteller sein Produkt als vollwertiges USB-Host mit umfangreicher Bibliothek und Treibern beworben hat, wurde dieses Produkt für einen Einkaufspreis von ca. 20,- € angeschafft. Leider stellte sich heraus, dass die Treiber zum Zeitpunkt der Entwicklung nicht weit genug entwickelt wurden, um einen USB-Stick mit diesem Gerät zu beschreiben (der heutige Stand ermöglicht dies jedoch).

Daher wurde als Alternative der *Memory Stick Datalogger* vom Hersteller *Parallax* bestellt (Kosten ca. 40,- €), der einen FTDI[6]-Chip als Herzstück besitzt. Für diesen Kontroller werden tatsächlich eine Bibliothek sowie geeignete Treiber für die Anwendung mit einem *Arduino* zur Beschreibung eines USB-Sticks bereitgestellt. Nachteil dieses USB-Kontrollers ist, dass die Verbindung zwischen Arduino und dem *Memory Stick Datalogger* hergestellt werden muss. Ein einfaches Stapeln bzw. aufstecken ist nicht ohne weiteren Aufwand möglich.

4. Erstellung eines Stapelshields

Nachdem der USB-Stick als Speichermedium und ein USB-Kontroller als Schnittstelle ausgewählt wurde, musste dieser Kontroller noch mit dem *Arduino UNO* verbunden werden. Für die ersten Versuche ist dies mittels einer Steckplatine bzw. eines Steckbrettes, mehreren Kunststoff-Fassungsadern, zwei Tastern, einem *Arduino UNO* und dem *Memory Stick Datalogger* von *Parallax* geschehen.
Nachdem die ersten Versuche glückten, sollte diese Verbindung in einer Form umgesetzt werden, die beständiger ist. Dazu wurde eine Platine in Form eines Stapelshield erstellt, die nun anhand der Beschreibung des Herstellungsprozesses erläutert werden soll.

Erstellung der Schaltung

Um das Stapelshield dem *Arduino UNO* anzupassen, musste zunächst die richtige Form erstellt werden. Da es bereits viele Shields für den *Arduino UNO* im Open-Source-Bereich gibt, wurde hier einfach ein passendes Protoshield herausgesucht. Gewählt wurde das *Proto Shield* von Fried [6]. Dabei handelt es sich um eine *eagle-schematic*- und eine *eagle-board*-Datei. Die wesentlichen Elemente dieser Vorlage sind dabei die passend angeordneten Pin-Leisten und die direkt über den *Arduino UNO* passende äußere Form. Zum einen sind die Pinleisten so gestaltet, dass die Pins direkt in den *Arduino UNO* passen und so das Shield auf diesen gestapelt werden kann. Zum anderen sind diese mit einer Reihe Pads verbunden, damit die Pins mit anderen Bauelementen verbunden werden können. Weitere Elemente, die hier von Bedeutung sind, sind der durchgeführte und schon richtig verbundene Reset-Taster, sowie zwei LEDs mit zugehörigen Vorwiderständen. Sinnvolle Ergänzungen stellen die Padleisten, die auf Masse (GND) oder +5V liegen, dar. Da es sich um ein Protoshield handelt, sind zusätzlich viele Pads vorgesehen, auf denen weitere Bauelemente zu Versuchszwecken untergebracht werden könnten.
Um das Shield auf den hier vorliegenden Bedarf anzupassen, wurden zunächst einige Elemente aus dem Schaltplan entfernt. Der hier entstandene Schaltplan ist in Abb. 3 zu sehen.

[6] FTDI: Future Technology Devices International

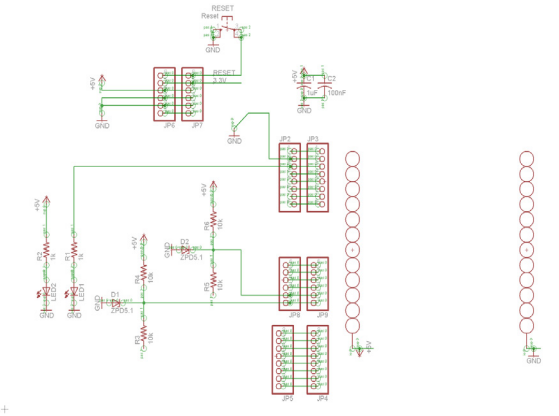

Abb. 3: eagle-schematic für das Stapelshield

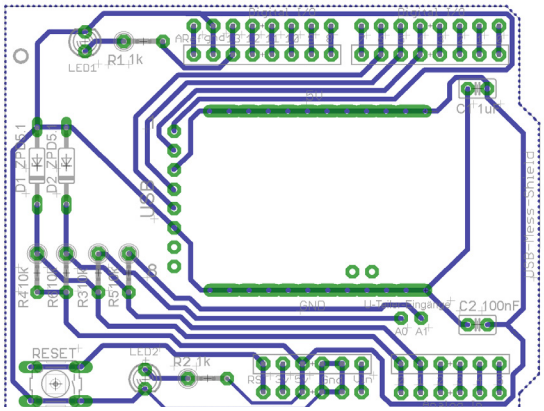

Abb. 4: eagle-board für das Stapelshield

Für den Einsatz hier wurden die Pinleisten, die das Stapeln sowie das Anschließen weiterer Komponenten ermöglichen, beibehalten. Außerdem wurden der Reset-Taster, die beiden Kondensatoren, die Schutz vor Störspannungen unterschiedlicher Frequenzen bieten sollen, je eine Padleiste auf GND und +5V und die beiden LEDs mit ihren Vorwiderständen übernommen. Die LED2 zeigt dabei die Spannungsversorgung und damit die Betriebsbereitschaft des Gerätes an, während die LED1 über den Pin 13 ansprechbar ist. Da die analogen Eingänge des *Arduino UNO* für Spannungen von 0 bis +5V geeignet sind, wurde hier für zwei Eingänge je eine Schaltung mit zwei Widerständen vorgesehen. Grundsätzlich ist dieses Vorgehen üblich, um Spannungen über +5V zu messen. Dann werden die beiden Widerstände einfach als Spannungsteiler gegen GND genutzt und das mittlere Potential abgegriffen. In diesem Fall wird der Spannungsteiler jedoch mit einer Seite an +5V gelegt. Wird nun in der Mitte abgegriffen, so wird ein außen angelegter Spannungsbereich von -5V bis +5V auf einen vom Arduino abgegriffenen Bereich von 0 bis +5V verändert, da es sich um zwei gleiche Widerstände handelt. Weil der Arduino bei angelegten Spannungen über +5V und unter 0V leicht beschädigt werden kann, ist hier zusätzlich je eine Z-Diode (D1 & D2) geschaltet, die ein Unter- oder Überschreiten des vorgesehenen Spannungsbereich verhindert. Da das Shield nicht mehr als typisches Protoshield gedacht ist, wurden hier die Pinleisten auf je eine für GND bzw. +5V reduziert.

Design des Boards

Aus dem Schaltplan der eagle-schematic-Datei wird direkt der Entwurf der Platine, die eagle-board-Datei, erzeugt. Das Layout muss jedoch manuell erstellt werden. Von der Vorlage Fried [6] konnten die äußeren Maße sowie die Pinleisten übernommen werden. Dadurch konnte ein passgenaues Stapeln des Shields ermöglicht werden. Das fertige Layout ist in Abb. 4 zu sehen.

Auch der Reset-Taster konnte am selben Platz gelassen werden, an dem er zuvor war. Alle anderen Elemente mussten versetzt werden. Zusätzlich zum Schaltplan wurde hier, direkt im Boardlayout, ein Anschluss für den *Memory Stick Datalogger* vorgesehen.

Dazu wurden zunächst sechs Pads für die Pins vorgesehen und entsprechend angeschlossen. Pin 1 und Pin 8 sind dabei beschriftet. Der Hersteller Parallax stellt dazu in seinem Datenblatt ein Beschaltungsschema zur Verfügung (siehe Abb. 5)

Pin Definitions (UART Mode)

Pin	Name	Description
1	Vss	Connects to System Ground
2	RTS#	Request To Send (Connects to MCU CTS)
3	Vdd	Connects to +5V (Regulated)
4	RXD	Receive Data (Connects to MCU TXD)
5	TXD	Transmit Data (Connects to MCU RXD)
6	CTS#	Clear To Send (Connects to MCU RTS)
7	NC	No Connection
8	RI#	Ring Indicator (Making this input low resumes from Suspend)

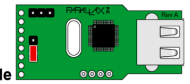

Abb. 5: Beschaltungsschema des Memory Stick Dataloggers [10]

Gemäß diesem Beschaltungsschema wurden die Pins 4 und 5 als serielle Schnittstellen mit den digitalen Ein- und Ausgängen 5 und 6 des *Arduino UNO* verbunden. Die Spannungsversorgung geschieht über die Pins 1 und 3 des USB-Kontrollers, die mit GND und +5V verbunden wurden. Die Pins 2 und 6 dienen einem Handshake, der hier nur einseitig genutzt wird, daher wurde der CTS[7]-Pin (Pin 6) auf GND gelegt und lediglich der RTS[8]-Pin abgefragt, um nur zum USB-Kontroller zu senden, wenn dieser auch empfangsbereit ist.

Bei näherem Betrachten des Boards fällt auf, dass sich direkt über der GND-Padleiste zwei weitere Pads befinden, die lediglich den Zweck erfüllen, mithilfe von zwei Stiften den *Memory Stick Datalogger* zu stützen, also aus statischen Gründen vorhanden sind.

[7] CTS: Clear To Send
[8] RTS: Request To Send

Durch ein Polygon, das durch die gepunkteten Linien auf den Außenlinien angedeutet ist, wird dafür gesorgt, dass nahezu alle ungenutzten Kupferflächen auf GND liegen.

Fertigung des Shields

Nachdem nun das Layout gestaltet war, musste das Gerät als solches gefertigt werden. Der Herstellungsprozess hat dabei einige Zeit in Anspruch genommen.

Der erste Schritt der Herstellung war das Gestalten der Shieldplatine. Dies kann durch Ätzen oder Fräsen einer Platine geschehen. Da zu diesem Zweck eine (kleine) CNC-Fräse zur Verfügung stand, wurde dieses Herstellungsverfahren gewählt. Dazu wurde mit Hilfe einer Outline-Funktion aus der eagle-board-Datei zunächst eine HPGL-Datei erzeugt. Diese konnte dann vom Programm WinPC-NC eingelesen und die Fräse nach der Vorlage gesteuert werden.

Eine Beschriftung der Platine wurde mit Hilfe eines Laserdruckers und einer Overhead-Folie auf die Platine kopiert. Anschließend wurde die Platine gemäß Schaltplan manuell bestückt.

An dieser Stelle sei auf die Anforderung 8, die universelle Einsetzbarkeit verwiesen. Diese Anforderung führte zu der Entscheidung, gleich vier analoge Eingänge zur Nutzung heraus (an Gehäusebuchsen) zu führen und so bereitzustellen. Dabei sind zwei dieser Eingänge für den Bereich 0 bis +5V und zwei der Eingänge für den Bereich -5V bis +5V (denn hier kommen die Spannungsteiler gegen +5V zum Einsatz) zu verwenden. Des Weiteren liegt die Begründung für zwei Taster (die ebenfalls am Gehäuse plaziert werden müssen) in der Anforderung 2, der Einfachheit der Anwendung, denn die Messung soll durch einfaches Tasten eines Start-Tasters begonnen und durch Tasten eines Stopp-Tasters gestoppt werden.
Zum Schluss wurde der *Memory Stick Datalogger* aufgesteckt und mit zwei Drähten fixiert. Das fertige aufgesteckte Shield ist in Abb. 6 zu sehen.

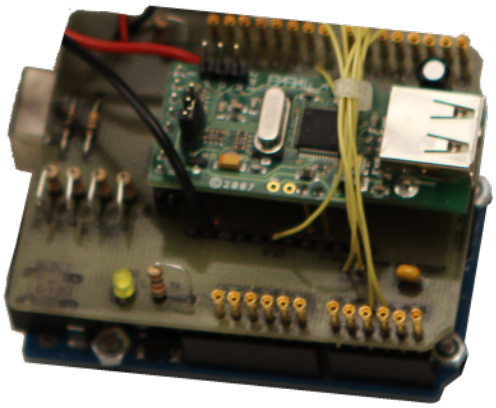

Abb. 6: Fertiges Stapel-shield

5. Bau des Gehäuses

Dem Bau eines Gehäuses lagen zwei Elementare Anforderungen zu Grunde: Anforderung 1, dass das Endprodukt ein Low-Coast-Gerät werden sollte und Anforderung 11, die Robustheit des Endproduktes.

Aufgrund von Anforderung 1 wurde ein Grundkonzept gewählt, bei der Eckprofilschienen zurechtgeschnitten, Platten in diese eingeschoben und mit Schrauben und Endplatten zusammengehalten werden. Als Material für die Platten wurde 1,5mm starkes Aluminium, um auch der Anforderung 11 gerecht zu werden. Die Platten wurden zunächst mit dem Programm Inkscape und später mit dem Programm Frontplattendesigner gestaltet. Aus den Layoutdateien wurden dann wieder HPGL-Dateien erzeugt, die vom Programm WinPC-NC zur Steuerung der Fräse genutzt werden konnten. Nach dem Zuschnitt wurden alle Bauelemente platziert und das Gehäuse zusammengebaut, sodass das *Arduino UNO* Board samt Stapelshield einfach eingeschoben werden konnte (siehe **Fehler! Verweisquelle konnte nicht gefunden werden.**).

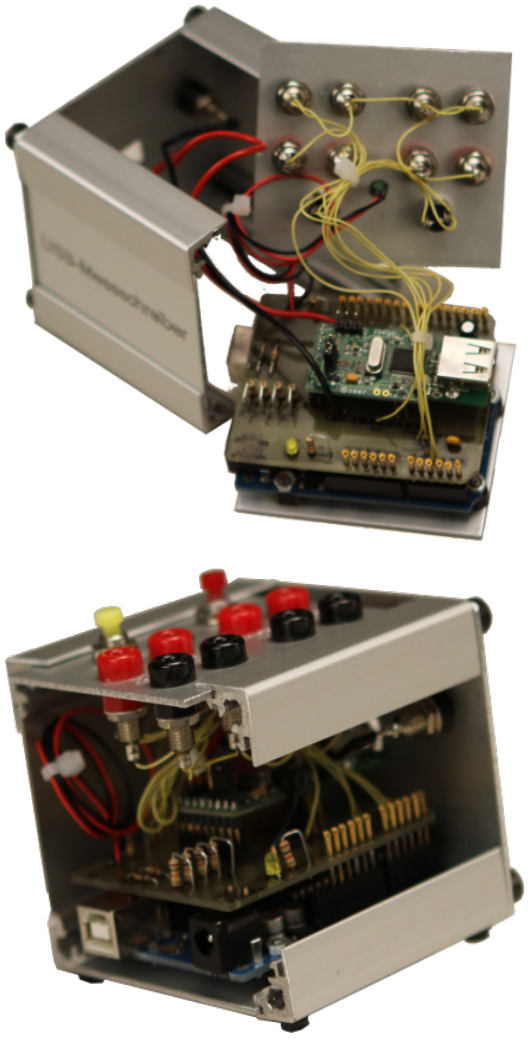

Abb. 7: *Arduino UNO* mit Stapelshield (oben) und Anschlüssen am Gehäuse (unten)

6. Das fertige Gerät

Da nun alle Teilkomponenten anhand ihres Herstellungsprozesses beschrieben wurden, bleibt noch das fertige Produkt, den USB-Datenlogger, zu beschreiben. Das fertige Gerät (siehe Abb. 8) hat dabei beinahe die Form eines Würfels und ist mit den Maßen 8,5cm x 8cm x 8,5cm kompakt gehalten. An der Oberseite befinden sich vier Messeingänge in Form von roten Bananenbuchsen, von denen die linken beiden für Spannungen von 0 bis +5V und die rechten beiden für Spannungen von -5V bis +5V geeignet sind. Als zweiter Anschluss steht je eine schwarze Bananenbuchse zur Verfügung, die auf GND des Boards liegt und das Bezugspotential vorgibt. Durch das Programm können ein Eingang aber auch mehrere abgefragt werden. Um weitere Möglichkeiten zu schaffen, befinden sich an der rechten Seite des Gerätes zwei Bananenbuchsen, über die die Boardspannung von +5V nach außen geführt und so für weitere Geräte (z.B. einen kleinen Messverstärker) zur Verfügung gestellt wird. Weiterhin befinden sich auf der Oberseite zwei Taster, von denen einer zum Starten und einer zum Stoppen der Messung dient. Außerdem ist auf der Oberseite eine LED zu finden, die den Betrieb, also das Messen bzw. das Schreiben auf den USB-Stick anzeigt.

Abb. 8: USB-Datenlogger

Der USB-Stick, wird auf der rechten Seite des Gerätes eingesteckt (Abb. 8, re.), während sich auf der linken Seite ein USB-Anschluss zur Verbindung mit einem Computer befindet (Abb. 8, li.), der zur Programmierung und Überwachung oder auch zur Spannungsversorgung genutzt werden kann. Für den Betrieb muss dieser Anschluss jedoch nicht genutzt werden, wenn die Spannungsversorgung über die hierfür vorgesehene Buchse direkt daneben geschieht.

Das Betreiben des USB-Datenloggers gestaltet sich denkbar einfach: Zunächst schließt man den Datenlogger an ein Netzgerät an oder versorgt ihn via USB-Schnittstelle mit der nötigen Betriebsspannung. Nun kann das Gerät wie ein herkömmliches Spannungsmessgerät (unter Berücksichtigung der Bezugsmasse) angeschlossen werden. Dabei können gleichzeitig mehrere Spannungen erfasst werden. Die Messung wird durch Betätigen des Start-Tasters begonnen. Der Betrieb wird durch leuchten der LED angezeigt. Die Messung stoppt bei einem langsamen USB-Stick, wenn der Ring-Puffer voll ist, automatisch oder, wenn dies nicht geschieht, durch Betätigung des Stopp-Tasters. Dann werden vor Beendigung des gesamten Prozesses noch die übrigen Daten aus dem Puffer auf den USB-Stick geschrieben. Erst wenn auch dies geschehen und damit der gesamte Prozess beendet ist, erlischt die LED.

Die Daten werden je Aufnahme in eine txt-Datei geschrieben. Diese Daten werden einfach durchnummeriert. Dabei ist die txt-Datei als Array gestaltet, sodass sie vom IBE – also in einem Java-Skript – das sich auf dem USB-Strick befinden kann, aufgerufen und in dieses integriert werden kann. Auch dies wurde im Rahmen der Arbeit erfolgreich durchgeführt.

Wird der USB-Stick während der Messung abgezogen, ist davon auszugehen, dass die Datei zerstört wird und so nicht erhalten bleibt. Sollte dies jedoch in einem Moment geschehen, in dem gerade keine Daten geschrieben werden, so kann die Datei mit Glück bestehen bleiben. Die Daten dieser Messung werden so bis zu diesem Zeitpunkt gesichert. Falls dies nicht der Fall ist, muss eine neue Messung vorgenommen werden. Die Nummerierung der Dateien wird einfach fortgeführt.

Nach Anforderung 1 sollte es sich bei dem Endprodukt um ein Low-Cost-Gerät handeln. Tabelle 1 zeigt eine Überschlagsrechnung zur Ermittlung der Gesamtkosten, die zur Herstellung des USB-Datenloggers aufgewendet werden mussten.

Bauteile	Preise (ca.)
Arduino UNO	50,-€
USB Host	40,-€
Platine & Bestückung	10,-€
Gehäusematerial & Bestückung	10,-€
Summe	**110,-€**

Tab. 1: Überschlagsrechnung zur Ermittlung der Gesamtkosten

Als Ergebnis dieser Überschlagsrechnung gemäß Tabelle 1 ergeben sich Kosten in Höhe von ca. 110,-€. Vergleicht man diese Kosten mit aktuellen Preisen bekannter Lehrmittelhersteller, wird rasch deutlich, warum hier von einem Low-Cost-Gerät gesprochen werden kann.

Das Gerät wurde in verschiedenen Versuchsreihen getestet und ist in seiner Funktion einwandfrei. Eine Verdopplung des Messtaktes ist bei derzeitigem Programm problemlos möglich, sodass Daten für ein flüssiges Bild mit einer Frequenz von 50Hz erzeugbar wären. Für höhere Frequenzen wäre jedoch eine vollständige Zwischenspeicherung im Puffer und damit eine starke Begrenzung der Aufnahmezeit verbunden.

7. Einbindung in ein IBE

Neben der Entwicklung des USB-Datenloggers ist es Bestandteil dieser Beitrags, zu zeigen, dass die aufgenommenen Messdaten auch in ein IBE eingebunden werden können. Aus diesem Grund wurde bereits berichtet, dass die Datei in Form eines Arrays geschrieben wurde, das in ein Java-Skript integrierbar ist.

Nach Anforderung 8 soll die hier vorgestellte Anwendung universell einsetzbar, d.h. auf unterschiedliche Experimente anwendbar sein. Da das Gerät Spannungen aufnehmen kann, erschien es zu einfach, einen Versuch aus der Elektrodynamik als Beispiel zu wählen. In Hinblick auf die Verwertung in einem IBE, gerieten der Aspekt der Anschaulichkeit sowie bewegte Gegenstände in den Vordergrund, wodurch nahe lag, ein Experiment aus der Mechanik zu wählen. Darauf wurden Versuchsbeschreibungen in Praktikumsskripten der physikalischen Praktika verschiedener Universitäten gesichtet und das Pohlsche Rad als möglicher Versuch ausgewählt, da es an vielen Standorten eingesetzt wird. Der Versuchsaufbau wurde in einem IBE umgesetzt und diente als Demonstrator für die Einbindung.

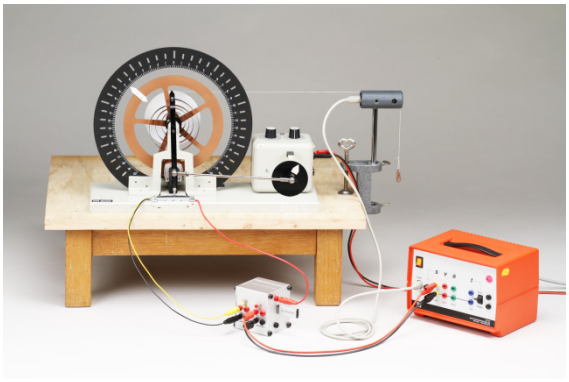

Abb. 9: Versuchsaufbau für das IBE mit dem Pohlschen Rad

Erfasst wurde über ein Potentiometer und mit Hilfe des Spannungsausgangs des USB-Datenloggers die Position des Erregers, sowie über einen Bewegungsmesswandler der Firma Leybold, der ebenfalls an den USB-Datenlogger angeschlossen war, die Position des Schwungrades. Diese Positionen wurden dann in Form eines Arrays in eine txt-Datei auf den USB-Stick geschrieben. Die in der Datei befindlichen Messwerte wurden im Anschluss vom IBE aufgerufen und das IBE lieferte eine exakte Repräsentation des Realexperiments.

8. Fazit & Ausblick

Dem hier vorgestellten Projekt lag die Idee zugrunde, eine kostengünstige Alternative zur digitalen Messwerterfassung zu schaffen, die eine besonders einfache Anwendung ermöglicht und Messdaten leicht ‚transportierbar' macht. Außerdem sollte die Idee berücksichtigt werden, die Messdaten so in ein IBE zu integrieren, dass dieses wie das echte, von Lernenden durchgeführte, Experiment abläuft. Durch diese Entwicklung sollte eine engere Verzahnung der Durchführungs- und Auswertungsphase ermöglicht werden.

Die Ergebnisse der ersten Erprobungsphase zeigen, dass das Vorhaben erfolgreich umgesetzt werden konnte. Dazu soll im Folgenden einer kurzer Abgleich in Bezug auf die eingangs aufgestellten Anforderungen erfolgen:

Tatsächlich ist es gelungen, eine kostengünstige Alternative zu den angebotenen Interfacesystemen bekannter Lehrmittelhersteller zu schaffen (vgl. Anf. 1).
Da das Messgerät genau wie ein übliches Spannungsmessgerät verwendet werden kann und die Aufnahme der Messwerte durch einfaches Drücken eines Tasters gestartet und gestoppt werden kann, ist die Anforderung nach Einfachheit in der Anwendung grundsätzlich erfüllt (vgl. Anf. 2).
Leider wird dieser Punkt jedoch davon eingeschränkt, dass nur Spannungen von -5V bis +5V bzw. von 0 bis +5V gemessen werden dürfen.
Weiter ist zu erwähnen, dass die Messung bei vollem Puffer automatisch beendet wird, was ebenfalls zur Vereinfachung der Anwendung beiträgt.
Auch die Verwendung in einem IBE ist zunächst einfach. Die erste Messdatei wird völlig automatisch vom IBE aufgerufen.
Sowohl die synchrone Aufnahme als auch das sofortige Bereitstellen der Messdaten konnte umgesetzt werden (vgl. Anf. 3).
Es kommt lediglich bei zu langsamen USB-Sticks zu einer Verzögerung von wenigen Sekunden, wenn der Puffer noch Daten enthält, die noch nicht auf den Stick geschrieben wurden.
Die Daten werden in Form einer txt-Datei direkt auf einem USB-Stick bereitgestellt. Dadurch sind die Messdaten leicht transportierbar (vgl. Anf. 4).
Die Daten werden in einer .txt-Datei als Tabelle geschrieben. Die einzelnen Werte innerhalb einer Zeile sind durch Kommata separiert, was einen Import in gängige Programme ermöglicht (vgl. Anf. 5). Dies wurde auch getestet.
Der IBE-Demonstrator lässt sich über die Messdaten ansteuern (vgl. Anf. 6).
Das IBE läuft flüssig. Das leicht ‚ruckelige' Erscheinungsbild des IBE-Demonstrators bleibt dem IBE und der Aufnahmequalität der Fotos geschuldet und nicht der Aufnahmegeschwindigkeit des USB-Datenloggers (vgl. Anf. 7).
Da es bereits viele Messwandler gibt, die Messwerte in proportionale Spannungen wandeln, ist das Gerät in vielen Versuchen einsetzbar (vgl. Anf. 8).
Der USB-Datenlogger ist unabhängig vom Internet einsetzbar, er benötigt lediglich eine Span-

nungsversorgung (vgl. Anf. 9). Die Daten können dann zusammen mit dem IBE auf einen USB-Stick transferiert werden.
Die AD-Wandlung wird von den analogen Eingängen des Arduino UNO mit ihren *Successiven Approximations* Registern übernommen (vgl. Anf. 10).
Da das Gerät mit einem stabilen Aluminiumgehäuse ausgestattet wurde, übersteht es sogar Stürze aus geringer Höhe. Die Bezeichnung als ‚robustes Gerät' ist daher durchaus gerechtfertigt (vgl. Anf. 11). (Einschränkend muss hier jedoch erwähnt werden, dass das Anlegen von zu großen Spannungen oder negativen Spannungen an den einfachen Eingängen leicht zu Beschädigungen führen kann.)

In Bezug auf die Erreichung der Ziele gilt:

Es wurde ein kostengünstiges Angebot zur digitalen Messwerterfassung geschaffen (vgl. Ziel 1).
Mit dem USB-Datenlogger kann den Lernenden ermöglicht werden, selbstständig Messdaten aufzunehmen, auszuwerten und zu interpretieren (vgl. Ziel 2).
Die förderlichen Aspekte von MBLs nach Hucke [8] werden durch die hier vorgestellte Anwendung unterstützt (vgl. Ziel 3):

a. Einfachstes Aufnehmen von Messdaten.
b. Synchrones Erzeugen von Messdaten.
c. Sofortiges Bereitstellen von Messdaten.

Zu a) muss jedoch einschränkend erwähnt werden, dass Hucke [8] auch von der synchronen Erzeugung von Graphen während des Experiments spricht, was mit dem USB-Datenlogger nicht möglich ist. In diesem Fall müsste ein Computer am Messplatz zur Verfügung stehen oder eine komplexe Erweiterung des Gerätes geschehen.
Durch die Verwendung des USB-Datenloggers kann im Vergleich zur manuellen Messwerterfassung wertvolle Zeit bei der Aufnahme von Messwerten eingespart werden (vgl. Ziel 4).
Durch die Anwendung werden die Möglichkeiten erweitert, Erfahrenes (also das Experiment) mit Theoretischem zu verbinden (vgl. Ziel 5). Dabei unterstützt dies sowohl die Möglichkeit die Daten zur Auswertung zu verwenden, als auch die, das Experiment währenddessen wiederholend anschauen zu können. An dieser Stelle ist jedoch festzuhalten, dass die Aufgabenstellung (im Praktikum oder in der Schule) entsprechend angepasst werden muss.
Durch den vereinfachten Prozess der Datenaufnahme und die engere Verzahnung von Durchführung und Auswertung, die durch die Anwendung im IBE gestützt wird, wird der Übergang dieser Phasen erleichtert (vgl. Ziel 6).
Es wird eine Auswertung am Computer ermöglicht, um so auch den Umgang mit entsprechenden Programmen zu lernen (vgl. Ziel 7).
Es werden damit einhergehend die Möglichkeiten verbessert, Gesetze aus Experimentierdaten selbst abzuleiten (auch in der Nachbereitung zu Hause) (vgl. Ziel 8). Auch hier ist natürlich eine entsprechende Aufgabenstellung erforderlich.
Die Möglichkeiten zur umfassenden Nachbereitung (auch zu Hause) sind verbessert worden (vgl. Ziel 9).
Es wird die Möglichkeit geschaffen, die Intensivierungsfunktion von Experimenten durch die Ermöglichung vielfältiger Darstellungsformen zu verstärken (vgl. Ziel 10).
Durch die Einbindung der Messdaten in ein IBE kann das Experiment auch in der Auswertung bzw. Nachbereitung stärker in den Mittelpunkt gerückt werden (vgl. Ziel 11).
Lerner können sich ihren eigenen Experimentverlauf durch Einbindung in IBE durch diese Anwendung wiederholend anschauen (vgl. Ziel 12).

Die Ziele, die diesem Beitrag zu Grunde liegen, können damit als erreicht angesehen werden. Dennoch gibt es bereits zu diesem Zeitpunkt Ideen und Vorschläge zur Erweiterung und Verbesserung der Entwicklung:

- Eine sinnvolle Ergänzung, die bei der Einbindung der Daten in ein IBE nach dem hier vorgestellten IBE-Demonstrator vorgenommen werden könnte, ist die automatische Bestimmung der Grenzwerte. Hier ist dies manuell durch experimentell bestimmte Grenzwerte geschehen. Eine Automatisierung dieses Prozesses würde jedoch die Anwendung vor allem beim Wechseln der Geräte im Experiment vereinfachen.
- Um die universelle Einsetzbarkeit des USB-Datenloggers weiter auszubauen, wäre es sinnvoll, weitere Eingänge mit unterschiedlicher Ausstattung vorzusehen. So ist hier ein integrierter Operationsverstärker oder weitere Spannungsteiler denkbar, die auch andere Messbereiche ermöglichen.

Darüber könnten ein Datenlogger auch dafür genutzt werden, Messdaten online direkt über ein Netzwerk zur Verfügung zu stellen. Dieses Herangehen wäre besonders bei Demonstrationsexperimenten sinnvoll. Außerdem würde eine solche Anwendung den hier vernachlässigten förderlichen Aspekt, Graphen synchron zu Experimenten entstehen zu lassen, berücksichtigen. Im Rahmen eines anderen Projektes (TET: www.tetfolio.de) wird derzeit ein vergleichbares Messinterface entwickelt, das es sogar ermöglicht, dass jeder Lerner den Graphen auf einem eigenen mobilen Endgerät (z.B. Tablet-PC oder Smartphone) entstehen sieht und dort auch Messdaten bereitgestellt werden.

Ein anderes Beispiel für eine Weiterentwicklung wäre, Möglichkeiten zu schaffen, die Aufnahmegeschwindigkeit zu steigern. Hierzu könnte z.B. eine SD-Karte statt eines USB-Sticks verwendet werden,

auf die schneller geschrieben werden kann. Wenn nur weniger Daten (ca. 450 Messwerte) einer höheren Frequenz, als sie momentan mit dem USB-Datenlogger gemessen werden kann, benötigt werden, kann für dieses Gerät auch ein neues Programm geschrieben werden, welches alle Daten während des Messprozesses in einen Pufferspeicher schreibt. Das Wandeln und Schreiben der Daten auf den USB-Stick müsste in diesem Fall im Anschluss an die Messung geschehen. Mit neu zur Verfügung gestellten Bibliotheken sind nach ersten Versuchen der Autoren beim Schreiben der Daten auf USB-Sticks bereits Datenraten von 100Mbyte/s möglich. Dadurch eröffnen sich neue Perspektiven für die hier beschriebene Entwicklung bzw. Weiterentwicklungen, in denen ebenfalls auf einen USB-Stick geschrieben wird.

Insgesamt ist festzuhalten, dass durch den USB-Datenlogger ein Low-Cost-Gerät zur digitalen Messwerterfassung zur Verfügung gestellt wird, mit dem bei richtiger Aufgabenstellung eine Verzahnung der Durchführungs- und Auswertungsphase gefördert werden kann. Es ergeben sich so vielfältige Möglichkeiten für den Einsatz im naturwissenschaftlichen Praktikum oder im schulischen Unterricht.

Bei Interesse am USB-Datenlogger, wird darum gebeten sich direkt an die Autoren zu wenden, die gerne notwendige Dateien, Schaltpläne und Programme zur Verfügung stellen oder bei der Fertigung behilflich sind.

9. Literaturverzeichnis

[1] ATMEL: Datenblatt: 8-bit AVR Mikrocontroller with 4/8/16/32K Bytes In-System Programmable Flash. ATmega48PA, ATmega88PA, ATmega168PA, ATmega328P.
[2] Banzi, Massimo (2012): Arduino für Einsteiger. [die Open-Source-Elektronik-Prototyping-Plattform für Arduino 1.0]. Beijing, Cambridge, Farnham, Köln, Sebastopol, Tokyo. O'Reilly.
[3] Bernshausen, Henrik (2008): Vergleichende Analyse von Computergestützten Messwerterfassungssystemen für den Physikunterricht. Dissertation. Siegen.
[4] Brechmann, Gerhard (2002): Elektrotechnik. Tabellen Energieelektronik. Braunschweig. Westermann.
[5] Evans, Brian W. (2009): Arduino Programmier-Handbuch.
[6] Fried, Limor (2011): Proto Shield. Arduino prototyping.
[7] Girwidz, Raimund (2009): Medien im Physikunterricht. In: Kircher, Ernst; Girwidz, Raimund; Häußler, Peter(Hrsg.): Physikdidaktik, S. 203–264. Berlin, Heidelberg. Springer.
[8] Hucke, Lorenz (1999): Handlungsregulation und Wissenserwerb in traditionellen und Computergestützten Experimenten des physikalischen Praktikums. Dissertation. Dortmund.
[9] Margolis, Michael (2012): Arduino-Kochbuch. [Arduino für Fortgeschrittene ; behandelt Arduino 1.0]. Beijing, Cambridge, Farnham, Köln, Sebastopol, Tokyo. O'Reilly.
[10] Parallax Inc: Datenblatt. Memory Stick Datalogger (#27937): http://www.parallax.com/Portals/0/Downloads/docs/prod/comm/MemoryStickDataloggerV1.1.pdf. 05.13.
[11] Wilhelm, Thomas; Trefzger, Thomas (2010): Erhebung zum Computereinsatz bei Physik-Gymnasialleherern. In: Nordmeier, Volkhard; Grötzebauch, Helmut(Hrsg.): PhyDidB: http://www.phydid.de/index.php/phydid-b/article/view/109/119. 04.13.