KOMPETENZZENTRUM FÜR MATHEMATISCHE MODELLIERUNG IN MINT-PROJEKTEN IN DER SCHULE

KOMMS Reports Nr. 12 (2020)

Reports zur Mathematischen Modellierung
in MINT-Projekten in der Schule

KOMMS
Kompetenzzentrum für Mathematische Modellierung in MINT-Projekten in der Schule

Produktorientierte mathematische Modellierung am Beispiel eines Schrittzählers

Patrick Capraro

EUROPÄISCHER SOZIALFONDS
Mehr Chancen für Rheinland-Pfalz

EUROPÄISCHE UNION
Europäischer Sozialfonds

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Zusammenfassung:
Die Konstruktion eines Schrittzählers mit einem Arduino-Mikrocontroller und einem Bewegungssensor ist ein spannendes Technikprojekt

- Es gibt einen Einblick in eine alltagsnahe Technologie
- Technische Hürden lassen sich durch entsprechende Vorbereitung des Materials reduzieren; Schülerinnen und Schüler können dann ohne tiefe Programmier- und Elektronikkenntnisse einfache Ergebnisse erzielen
- Verschiedene Aspekte eines modernen Entwicklungsprozesses werden integriert: Programmieren, Arbeiten mit Hardware, Problemlösen, Design
- Die Aufgabe lässt sich beliebig weit ergänzen, indem zusätzliche Funktionen implementiert werden

Wir erläutern den Grundgedanken hinter der produktorientierten Modellierung und die vielfältigen Möglichkeiten, die Fragestellung zu bearbeiten. Darüberhinaus werden die technischen Details der verwendeten Hardware diskutiert, um einen schnellen Einstieg ins Thema zu ermöglichen.

Inhaltsverzeichnis

1 Der Grundgedanke der Produktorientierung

1.1 Grundidee

Mathematik steckt in allen Bereichen unseres Lebens, in denen komplexe Vorgänge gesteuert werden. Ob das die Optimierung von Fahrplänen bei der Deutschen Bahn sind, die GPS-Daten im Smartphone oder die Verschlüsselung bei einem Login im Onlineportal. Wenn wir uns mit mathematischer Modellierung beschäftigen, können wir in diese Zusammenhänge eintauchen und erkennen, wie mächtig die Mathematik beim Lösen von Problemen tatsächlich ist.

In der Schule geschehen diese Lösungsversuche – wenn überhaupt – oft nur theoretisch. Die Schülerinnen und Schüler bearbeiten ein Modellierungsprojekt, entwickeln eine Lösung und präsentieren diese möglicherweise in einem kurzen Vortrag. Danach ist das Kapitel abgehakt und man wendet sich einem anderen Thema zu.

Manche interessierte Schülerinnen und Schüler stellen jedoch zurecht die Frage: „Wie gut ist die Lösung denn? Wie würde man die Idee in die Praxis umsetzen? Kann man damit etwas anfangen?“ Manche Lösungen zu einem realen, offen gestellten Problem sehen in der Theorie gut aus, doch es bleibt offen, ob sie dem Praxistest standhalten.

Eines der Grundprinzipien der mathematischen Modellierung ist die Reduktion des Problems auf eine überschaubare Anzahl von (mathematischen) Sachverhalten. Dabei geht in aller Regel Information verloren. Das ist eine der wesentlichen Eigenschaften von Modellen (nämlich das Abbildungsmerkmal): Sie erfassen nicht alle Attribute des Originals und haben daher einen eingeschränkten Gültigkeitsbereich [Sta73, S. 131].

Daher ist es eine lohnenswerte Investition, wenn eine Lösungsidee – soweit möglich – in die Realität umgesetzt und getestet wird. Dann erhält man zusätzliche Informationen über die Qualität der Lösung und kann überprüfen, ob die Modellreduktionen gerechtfertigt sind. Gegebenenfalls kann anhand dieser Informationen das Modell überarbeitet werden.

Bei den Modellierungsaktivitäten des KOMMS ist im Laufe der Zeit der Fokus auf solche Modellierungsprojekte gewachsen, bei denen die Schülerinnen und Schüler die Möglichkeit haben, nicht nur eine abstakte Lösungsidee zu entwickeln, sondern diese auch auf reale Daten anzuwenden und zu testen: ein fertiges Produkt. Wir sprechen daher in diesem Zusammenhang von **produktorientierter Modellierung** [BBC17].

Besonders deutlich wird dieser Ansatz bei Projekten, in denen ein technisches Gerät entwickelt werden soll. Ein Bauplan, der in der Theorie sinnvoll aussieht, mag an einfachen Hindernissen scheitern, die bei der Planung zu banal erschienen, um sie im Modell zu berücksichtigen. Technische Geräte, wie Sensoren oder Motoren, funktionieren möglicherweise nicht so reibungsfrei, wie die physikalische Theorie das voraussagt. Auf solche Gegebenheiten muss das Entwicklerteam reagieren und entsprechende Workarounds liefern. Dies fügt dem Modellierungsprozess eine ganz neue Dimension hinzu.

1.2 Die verschiedenen MINT-Disziplinen

In mehreren Durchgängen wurden bei den Modellierungsveranstaltungen des KOMMS und der Arbeitsgruppe Technomathematik der TU Kaiserslautern bereits technische Geräte entwickelt. Oft wurden neue Technologien einbezogen, die speziell für den Bildungssektor entwickelt wurden. Namhaft sind hier der Mikrocontroller Arduino und der Einplatinencomputer Raspberry Pi. Projektbeispiele sind die Entwicklung von 3D-Scannern, Lichtor-

geln und Schrittzählern.

Auch wenn die mathematische Modellierung generell eine starke Tendenz dazu hat, Fachwissen aus unterschiedlichen Disziplinen zu erfordern, so ist das bei der produktorientierten Modellierung oft stärker ausgeprägt. Die technischen Projekte erfordern dabei eine besondere Vielfalt an Kenntnissen aus den MINT-Disziplinen. Beim Schrittzähler könnten das folgende Inhalte sein:

- Mathematik
  - Arbeiten mit großen Datenmengen und deren statistische Auswertung
  - Numerische Verfahren (z.B. numerische Integration)
  - Koordinatensysteme: Das Koordinatensystem des Beschleunigungssensors ist – relativ zum umgebenden Raum – ständig in Bewegung
- Informatik
  - Programmieren
  - Komplexitätstheorie, Rechenzeit und Speicherplatzmanagement: Manche Geräte (vor allem der Arduino) haben stark reduzierte Ressourcen
  - Einbeziehung von Methoden des maschinellen Lernens
- Naturwissenschaften
  - Physik: Geschwindigkeit und Beschleunigung, Stromkreise (zum Lesen und Erstellen von Schaltplänen)
  - Biologie und Gesundheit: Kalorienrechner als Erweiterung der Fragestellung
- Technik
  - analoge und digitale Signale verstehen und nutzen
  - Steuerung von Sensoren
  - Steuerung von Ein- und Ausgabegeräten

Ergänzt wird die Fülle an fachlichen Anforderungen noch um die Komponente des Designs[1]. Zwar kann man mit dem Anspruch an die Aufgabe herantreten, dass es weniger um die Funktionalität geht, sondern eher darum, ein Grundverständnis für die technischen Schwierigkeiten zu entwickeln. Aber selbstverständlich besteht die Option, auch Designkomponenten zum Teil der Aufgabenstellung zu machen.

Dies können handwerkliche Aufgaben sein, wie die Anfertigung einer Tragevorrichtung für den Schrittzähler und einer Box für die unterschiedlichen Bauteile. Es kann um die optische Gestaltung der Ausgabe (z.B. auf einem Display) gehen, oder um Tragekomfort und Benutzerfreundlichkeit der Funktionen. Gerade diese Aufgaben sind teilweise losgelöst von den mathematischen Problemen und erlauben es Schülerinnen und Schülern, die kreative Talente besitzen, sich einzubringen. Andererseits zeigt gerade die Steuerung eines TFT-Displays, dass in der heutigen Zeit auch in kreativen Berufen ein Grundverständnis der mathematischen Darstellung von Designelementen hilfreich ist.

Schließlich kann es auch wichtig sein, das Produkt gezielten Tests zu unterziehen, um Schwachstellen in der Entwicklung zu erkennen. Auch das Testen ist ein anspruchsvolles Element von Problemlösestrategien.

[1] Der Autor ist der Meinung, dass bei produktorientierter Modellierung der Sammelbegriff MINT zu MINDT erweitert werden sollte.

1.3 Parallelen zu einem modernen Entwicklerteam

Die Relevanz der produktorientierten Modellierung zeigt sich auch in Parallelen zu modernen Organisationsformen von Entwicklerteams. Oft zeigt sich bei den Modellierungsveranstaltungen, dass sich die Gruppen (bei geeingeter Gruppenstärke von beispielsweise 5 oder 6 Personen) in Projektteams aufteilen: Eher technik- und programmieraffine Schülerinnen und Schüler beschäftigen sich mit der Hardware und der Datenverarbeitung, andere möglicherweise mit dem mathematischen Modell und der grundsätzlichen Frage, wie die vorliegenden Daten genutzt werden können, andere stürzen sich auf die Designkomponenten und machen sich Gedanken darüber, wie der Benutzer das Gerät steuern kann.

Wir vergleichen diese Aufgaben einmal mit typischen Berufsbezeichnungen aus der Welt der Softwareentwicklung (vgl. beispielsweise [Sta]):

- Frontend-Entwickler: Arbeitet an der Schnittstelle zum Benutzer
- User-Experience-Entwickler: Optimiert das Design hin zu besserer Benutzerfreundlichkeit
- Backend-Entwickler: Entwickelt die Vorgänge, die im Hintergrund geschehen (von denen der Benutzer nichts mitbekommt)
- Solution-Engineer: Erarbeitet allgemeine Lösungsstrategien für die Problemstellung

Auch die Methoden agiler Produktentwicklung können sich in der Projektarbeit widerspiegeln: Hier ist es üblich, dass die unterschiedlichen Entwicklerteams unabhängig an verschiedenen Aufgaben arbeiten. Es wird jedoch durch regelmäßige Teambesprechungen gewährleistet, dass erstens die verschiedenen Teile des Produkts tatsächlich zusammenpassen; und dass zweitens die Zielvorgabe ständig an neue Erkenntnisse der Entwicklerteams angepasst wird.

2 Schrittzähler – ein vielfältiges MINT-Projekt

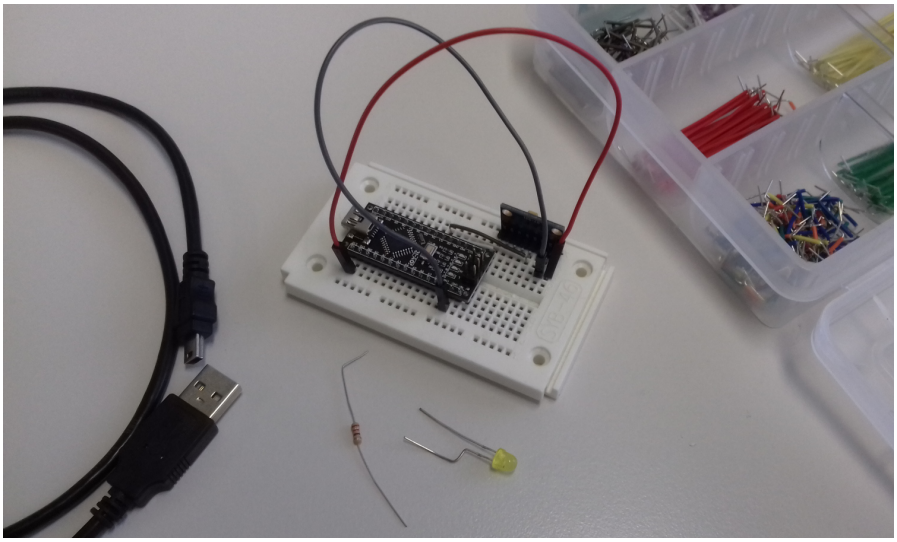


Abbildung 1: Minimalmodell eines Schrittzählers (*Foto: Patrick Capraro*)

2.1 Aufgabenstellung

Ein Schrittzähler kann anhand von bestimmten Bewegungsmustern erkennen, wie viele Schritte der Träger macht. Dazu sind selbstverständlich Sensoren notwendig, welche die Bewegung des Geräts im dreidimensionalen Raum analysieren können. Doch diese Daten alleine reichen noch nicht aus, um die Schritte zu zählen, schließlich müssen verschiedene Bewegungsarten, wie Laufen und Autofahren, unterschieden werden. Außerdem muss die Fortbewegung auch von kleineren Bewegungen unterschieden werden, wie dem Zittern des Beins oder anderen Störungssignalen.

In dem Projekt soll mit Hilfe eines Arduino-Mikrokontrollers und eines Beschleuniggungssensors ein Schrittzähler konstruiert werden. Der Fokus liegt dabei auf der Auswertung der Sensordaten sowie der Frage, welche Bewegungsmuster als Schritt verstanden werden sollen und welche nicht. Je nach dem, wie die Aufgabe vorbereitet wird, kann der technische Aspekt (Aufbau und Steuerung der Elektronik) thematisiert werden, oder auch nicht.

2.2 Technische Ausstattung

Aus der Aufgabenstellung und der Art und Weise, wie das Material zur Verfügung gestellt wurde, war eine Umsetzung des Projektes mit Hilfe von einem Arduino Nano und einem Beschleunigungssensor mit der Bezeichnung GY 521 vorgesehen. Der Mikrocontroller stand im mehrfacher Ausführung zur Verfügung, damit mehrere Teammitglieder gleichzeitig entwickeln konnten. Der Beschleunigungssensor war nur in zweifacher Ausführung vorhanden.

Zusätzlich standen mehrere Breadboards und zugehörige Steckverbindungen (Drahtbrücken und Jumperkabel) zur Verfügung, um die Schaltungen zu realisieren. LEDs und Widerstände dienten unter anderem als Arbeitsmaterial für eine Kurzeinführung in das Arbeiten mit dem Arduino.

Weiterhin gab es verschiedene Bauteile, die Weiterentwicklungen des Geräts ermöglichen sollten (ohne dass solche Erweiterungen explizit gefordert waren). Dabei handelte es sich um verschiedene Sensoren aus dem Sensorkit X40 von joy-it, sowie einem TFT-Display, Tastern, Kondensatoren, Transistoren und der Möglichkeit, Batterien anzuschließen.

Raspberry Pis standen ebenfalls zur Verfügung, wurden aber in bisherigen Durchführen nicht von den Schülerinnen und Schülern genutzt.

2.3 Vorbereitung

In der Vorbereitung auf die Modellierungsprojekte wurde in aller Regel getestet, ob die Arduino-Mikrocontroller und die Sensoren einwandfrei funktionierten. Insbesondere wurde dabei ein Code getestet, mit dem die Sensoren ausgelesen werden konnten. Bisher waren die Schülerinnen und Schüler jedoch stets selbst in der Lage, einen geeigneten Code online zu finden. Im Abschnitt 3.1 zeigen wir, wie der Sensor ausgelesen werden kann.

Da die Mehrheit der Schülerinnen und Schüler mit dem Arduino nicht vertraut war, bot es sich an, eine kurze Einführung in das Arbeiten mit dem Gerät durchzuführen. Dabei können auch die Grundlagen des Programmierens, wie die Nutzung von Schleifen und Kontrollstrukturen, sowie die Besonderheiten der Sprache C vermittelt werden. Hier lohnt es sich, einen kleinen Ablauf vorzubereiten oder die Schülerinnen und Schüler in Einzelarbeit mit entsprechenden Arbeitsblättern lernen zu lassen. Das Ansteuern einer LED und das Einlesen von Signalen an einem Taster eignen sich gut, um die wichtigsten Ideen zu vermitteln.

3 Hilfestellungen für die Umsetzung

3.1 Der Beschleunigungssensor GY 521

3.1.1 Verkabelung

Am Beschleunigungssensor werden 4 Anschlüsse belegt:

VCC	5 V-Pin oder äußere Spannungsquelle
GND	GND-Pin
SCL	A5
SDA	A4

3.1.2 Der Arduino-Code

```
#include<Wire.h>
const int MPU=0x68;
int16_t AcX,AcY,AcZ,Tmp,GyX,GyY,GyZ;

void setup(){
  Wire.begin();
  Wire.beginTransmission(MPU);
  Wire.write(0x6B);
  Wire.write(0);
  Wire.endTransmission(true);
}
void loop(){
  Wire.beginTransmission(MPU);
  Wire.write(0x3B);
  Wire.endTransmission(false);
  Wire.requestFrom(MPU,14,true);
  AcX=Wire.read()<<8|Wire.read();
  AcY=Wire.read()<<8|Wire.read();
  AcZ=Wire.read()<<8|Wire.read();
  Tmp=Wire.read()<<8|Wire.read();
  GyX=Wire.read()<<8|Wire.read();
  GyY=Wire.read()<<8|Wire.read();
  GyZ=Wire.read()<<8|Wire.read();
  delay(1000);
}
```

Listing 1: Auslesen des Beschleunigungssensors GY 521.

Der Befehl `#include<Wire.h>` sorgt dafür, dass die Bibliothek eingebunden wird, welche die Kommunikation mit dem Sensor ermöglicht. Die Angaben der Form `0x6B` sind Speicheradressen, wobei `0x` anzeigt, dass es sich um eine Zahl in hexadezimaler Darstellung handelt, die nachfolgenden beiden Stellen beschreiben dann den Zahlenwert.

Die Befehle `Wire.xxx` dienen dazu, das Auslesen zu initialisieren. Dabei sind folgende Angaben wichtig: `Wire.write(0x3B);` gibt an, wo mit dem Auslesen begonnen wird, und `Wire.requestFrom(MPU,14,true);` gibt an, wie viele Bytes gelesen werden (nämlich 14).

In manchen Codebeispielen, die man im Internet findet, ist der Wert auf 12 gesetzt und es werden daher zu wenige Informationen gelesen.

Der Befehl `Wire.read()<<8|Wire.read();` liest 2 Bytes aus, und zwar in einer ganz besonderen Art und Weise: Zunächst liest `Wire.read()` ein Byte als Integerwert in Binärdarstellung (z.B. 251=$(11111011)_2$). Dann werden durch den Befehl `<<8` an die Binärzahl 8 Nullen angehängt (Multiplikation mit 256=$(100000000)_2$). Schließlich wird noch ein Byte gelesen und mit dem Befehl `|` auf das vorherige Byte addiert. Man liest somit einen Integerwert ein, der einen Speicherplatz von 2 Byte umfasst.

Die Sensorwerte sind damit in den Variablen `AcX,AcY,AcZ,Tmp,GyX,GyY,GyZ` gespeichert, dabei stehen die ersten drei für die Beschleunigungswerte in 3 Raumrichtungen, die vierte für die Temperatur, und die letzten drei für die Winkelgeschwindigkeiten für die 3 Raumachsen.

Achtung: Die Reihenfolge, in der die Daten mit dem Befehl `Wire.read()<<8|Wire.read();` eingelesen werden, ist nicht beliebig. Man sollte diese Reihenfolge nicht ändern.

3.1.3 Interpetation der Daten

Die Daten lassen sich bequem über den seriellen Monitor oder den Seriellen Plotter betrachten. Dazu kann man im Setupteil des Codes den Befehl `Serial.begin(9600);` einfügen, um die serielle Schnittstelle zu aktivieren. Nach dem Auslesen der Daten im Loopteil des Codes kann man dann Befehle der Form `Serial.println(AcX);` verwenden, um Zahlenwerte (oder auch Zeichenketten) auf dem seriellen Monitor auszugeben. Die Zahlenwerte können auch über den seriellen Plotter in einem Koordinatensystem visualisiert werden.

Die gemessenen Rohwerte müssen natürlich noch in physikalische Einheiten umgerechnet werden. Hier sind die Umrechnungsfaktoren bei der Beschleunigungsmessung 16384=1 g und bei der Winkelgeschwindigkeit 131=1 Grad pro Sekunde.

Achtung: Man sollte sich von den gemessenen Werten nicht irritieren lassen: Auch in völliger Ruhe misst der Beschleunigungssensor große Werte, denn auch die Erdbeschleunigung wird erfasst!

3.2 Das Problem mit dem Koordinatensystem

Will man die Messdaten nutzen, um beispielsweise die Beschleunigung im dreidimensionalen Raum zu erfassen, muss man die Erdbeschleunigung subtrahieren. Das ist allerdings nicht trivial, denn das Koordinatensystem des Sensors ist relativ zum Koordinatensystem des Raumes in ständiger Bewegung. Insbesondere ist die relative Lage der Achsen beider Koordinatensysteme zeitabhängig. Daher weiß man bei einer Sensormessung nicht, wie groß der Anteil der Erdbeschleunigung auf jeder Achse des Raumes ist.

3.2.1 Vollständige Herangehensweise an das Problem

Wir können die Messungen der Winkelgeschwindigkeiten nutzen, um ständig das Koordinatensystem des Sensors anzupassen. Letztendlich nutzt man hier eine numerische Integration, um die absolute Winkeländerung vom Startzeitpunkt t_0 bis zum Zeitpunkt t, zu berechnen.

Wir müssen dazu zu Beginn den Sensor kalibrieren. Dazu erfasst man, wie die Koordinatensysteme relativ zueinander liegen. Das kann beispielsweise geschehen, indem der Arduino in einer Situation angeschaltet wird, in der sich der Sensor in Ruhe befindet (in diesem Fall misst man lediglich die Erdbeschleunigung und kann anhand der Daten erkennen, wie die z-Achse ausgerichtet ist). Danach initialisieren wir die drei Winkel

```
int alphaX=0,alphaY=0,alphaZ=0;
```

im Programmkopf und aktualisieren bei jeder Sensormessung nach einem Zeitschritt `dt`

```
alphaX+=GyX*dt;
alphaY+=GyY*dt;
alphaZ+=GyZ*dt;
```

Aus diesen Winkeln können nun die Koordinaten des Sensors in Koordinaten des umgebenden Raums umgerechnet werden. Um den Zeitschritt `dt` zu erhalten, ist die Funktion `mikros()` nützlich, die einen Zeitwert in Mikrosekunden zurückgibt. Mit dem Code aus Listing 2 können wir `dt` berechnen.

```
long t,dt;
t=mikros();
//tu etwas
dt=mikros()-t;
```

Listing 2: Berechnen einer Zeitdifferenz.

3.2.2 Reduzierte Herangehensweise an das Problem

Das Problem lässt sich mit auch mit weniger Aufwand beheben, indem man zwei vereinfachende Annahmen hinzunimmt:

1. Wir interessieren uns nur für den Betrag der Beschleunigung.
2. Wir gehen davon aus, dass die Beschleunigung ausschließlich in der x-y-Ebene (des umgebenden Raums) stattfindet.

Punkt 2 ist für die meisten Anwendungen nicht richtig, aber unter Umständen mag der Fehler, den man begeht, akzeptabel sein.

In diesem Fall können wir folgendermaßen vorgehen: Die gemessene Beschleunigung $\vec{a}_{\text{sensor}}$ besteht aus zwei Komponenten: Die Beschleunigung $\vec{a}$, die wir messen wollen, und die Erdbeschleunigung $\vec{g}$. Es gilt

$$\vec{a}_{\text{sensor}} = \vec{a} + \vec{g}, \qquad \vec{a} \perp \vec{g},$$

und wegen Pythagoras

$$|\vec{a}|^2 = |\vec{a}_{\text{sensor}}|^2 - |\vec{g}|^2.$$

3.3 Arbeiten mit der seriellen Schnittstelle

Mit den Befehlen

```
Serial.print(<variable>);
```

und

```
Serial.println(<variable>);
```

Abbildung 2: Mit dem seriellen Plotter lassen sich die Messwerte anschaulich darstellen.

können Variablen auf dem seriellen Monitor ausgegeben werden. Um die Ausgabe sichtbar zu machen, muss man den seriellen Monitor unter `Werkzeuge` aktivieren.

Werden Zahlen über die serielle Schnittstelle geschrieben, dann kann man sich die Werte auch über den seriellen Plotter betrachten: übersichtlich in einem Schaubild dargestellt und gegen die Zeit geplottet (siehe Abbildung 2).

Die serielle Schnittstelle ist bestens für Experimente geeignet, wenn man sich auf Fehlersuche im Code begibt oder wenn das Arbeitsergebnis getestet werden soll.

3.4 Beispiel einer Realisierung

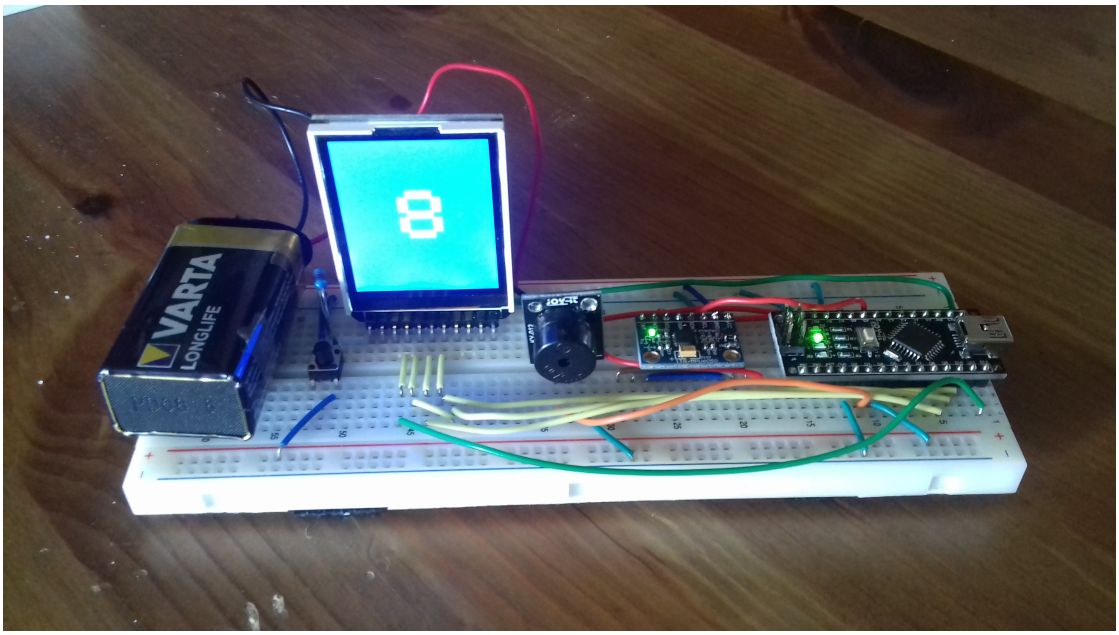


Abbildung 3: Der fertigte Schrittzähler bei der Fraunhofer Talentschool. Die verwendeten Bauteile sind (von links nach rechts): Batterie, Taster, TFT-Display, Piezo-Buzzer, Beschleunigungssensor, Arduino.

Bei der Fraunhofer Math-Talent-School im März 2020 wurde das Projekt mit einer Gruppe von 5 Teilnehmern aus den Klassenstufen 11 und 12 realisiert. Die Schülerinnen und Schüler kamen von Schulen des MINT-EC-Netzwerks aus ganz Deutschland. Die meisten verfügten über Grundkenntnisse im Programmieren und eine Schülerin hatte schon einmal mit einem Arduino gearbeitet.

An der Talentschool nahmen 23 Schülerinnen und Schüler teil, die an 4 Projekten arbeiteten. Start war montags abends mit der Projekteinteilung. Von Dienstag bis Donnerstag wurde ca 7 Stunden pro Tag an den Projekten gearbeitet. Freitags wurden die Ergebnisse im Plenum präsentiert.

Aufgrund der Vorkenntnisse wurden der Gruppe nur sehr wenige Hilfestellungen an die Hand gegeben (z.B. kein vorgefertigter Code). Die Teilnehmer organisierten sich gemäß ihrer Vorkenntnisse grob in zwei Teams: zwei Schülerinnen arbeiteten an der Benutzerschnittstelle (Steuerung des Schrittzählers über einen Taster, Steuerung des TFT-Displays), ein weiteres Zweierteam arbeitete mit dem Sensor und entwickelte das mathematische Modell des Schrittzählers. Eine Schülerin arbeitete mit beiden Teams zusammen und konnte daher die Teilprojekte gut zusammenführen. Die Tests des Geräts wurden oft mit der gesamten Gruppe besprochen.

Das Endergebnis (siehe Abbildung 3) verfügte über folgende Funktionen

- Die aktuelle Anzahl an Schritten wird auf dem Display ausgegeben
- Es ertönt ein Piepton nach einer gewissen Anzahl an Schritten und beim Reset
- Ein Taster ermöglicht einen Reset und kann erweiterte Anzeigeoptionen auf dem Display freischalten
- Durch ein Klettband kann der Schrittzähler am Bein befestigt werden

4 Messwerte Exportieren – vom technischen Problem zum Data-Science-Problem

Beim Arduino ermöglicht uns die serielle Schnittstelle, die Messwerte einzusehen, um notwendige Tests und Datenanalysen durchzuführen. Noch besser ist es jedoch, wenn wir die Messwerte auf einem Computer analysieren könnten, um fortgeschrittene mathematische Methoden darauf anzuwenden oder mit Hilfe einer höheren Programmiersprache bzw. eines Tabellenkalkulationsprogramms die Daten elektronisch auszuwerten.

Ein Datenexport ist über den Arduino nicht ohne weiteres möglich, doch ein Raspberry Pi kann diese Aufgabe meistern. Wir erläutern die wichtigen technischen Kniffe und wagen einen Ausblick, was wir mit den Daten anstellen können.

4.1 Sensor in Betrieb nehmen und Daten auslesen

Wir folgen einem Online-Tutorial auf `tutorials-raspberrypi.com` [ras].

- Zunächst wird der Sensor über den Raspberry mit Spannung versorgt (3.3 V oder 5 V) und die SDA- sowie SCL-Pins werden verbunden (SDA: GPIO2, SCL: GPIO3).
- Über die Konfiguration (`sudo raspi-config`) aktivieren wir SPI und I2C.
- Sofern noch nicht vorhanden, schreiben wir in die Datei `/etc/modules` die beiden Zeilen
  ```
  i2c-bcm2708
  i2c-dev
  ```
- Wir installieren zwei Pakete
  ```
  sudo apt-get install i2c-tools python-smbus
  ```

Jetzt können wir die Daten über ein Pythonskript auslesen (siehe Listing 3).

```
#!/usr/bin/python
import smbus
import math

# Register
power_mgmt_1 = 0x6b
power_mgmt_2 = 0x6c

def read_byte(reg):
    return bus.read_byte_data(address, reg)

def read_word(reg):
    h = bus.read_byte_data(address, reg)
    l = bus.read_byte_data(address, reg+1)
    value = (h << 8) + l
    return value

def read_word_2c(reg):
    val = read_word(reg)
    if (val >= 0x8000):
        return -((65535 - val) + 1)
    else:
        return val

bus = smbus.SMBus(1) # bus = smbus.SMBus(0) fuer Revision 1
address = 0x68       # via i2cdetect

# Aktivieren, um das Modul ansprechen zu koennen
bus.write_byte_data(address, power_mgmt_1, 0)

gyX = read_word_2c(0x43)
gyY = read_word_2c(0x45)
gyZ = read_word_2c(0x47)

acX = read_word_2c(0x3b)
acY = read_word_2c(0x3d)
acZ = read_word_2c(0x3f)
```

Listing 3: Auslesen des Beschleunigungssensors GY 521 in Python.

4.2 Daten exportieren und analysieren

Nun müssen die Daten in eine Datei geschrieben werden. Packt man das Auslesen und Schreiben der Daten in eine Schleife, lassen sich Messergebnisse über einen längeren Zeitraum dokumentieren (siehe Listing 4).

```
import time
t0=time.time()
f = open("daten.csv","a")
```

```
while(1):
  t=time.time()-t0
  gyX = read_word_2c(0x43)
  gyY = read_word_2c(0x45)
  gyZ = read_word_2c(0x47)

  acX = read_word_2c(0x3b)
  acY = read_word_2c(0x3d)
  acZ = read_word_2c(0x3f)

  f.write(str(t)+',')
  f.write(str(acX)+','+str(acY)+','+str(acZ)+',')
  f.write(str(gyX)+','+str(gyY)+','+str(gyZ)+'\n')
  time.sleep(0.1)
```

Listing 4: Ergänzung zu Listing 3: Schreiben in eine Textdatei.

Die csv-Datei kann nun direkt auf dem Pi weiterverarbeitet werden, oder man kopiert sie auf einen anderen Rechner und arbeitet dort weiter. Das csv-Format kann beispielsweise mit einem Tabellenkalkulationsprogramm geöffnet werden (siehe Abbildung 4).

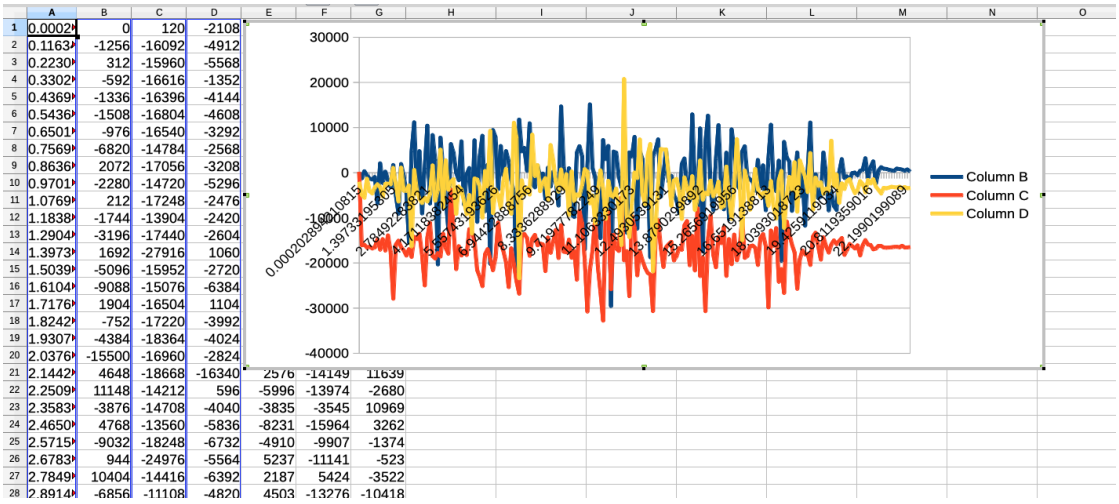


Abbildung 4: Analyse der Messdaten über ein Tabellenkalkulationsprogramm.

4.3 Trainingsdaten erzeugen

Will man die Daten mit statistischen Methoden oder durch maschinelles Lernen auswerten, kann es nützlich sein, den Zeitpunkt zu markieren, zu denen ein Schritt erfolgt ist. Dazu kann man den Messaufbau um einen Taster erweitern, der bei jedem Schritt gedrückt wird. In diesem Fall würde man bei jedem Schleifendurchgang in Listing 4 überprüfen, ob an dem Taster ein Signal anliegt und würde dann eine weitere Spalte in die `csv`-Tabelle schreiben (z.B. 1 für *Schritt* und 0 für *kein Schritt*).

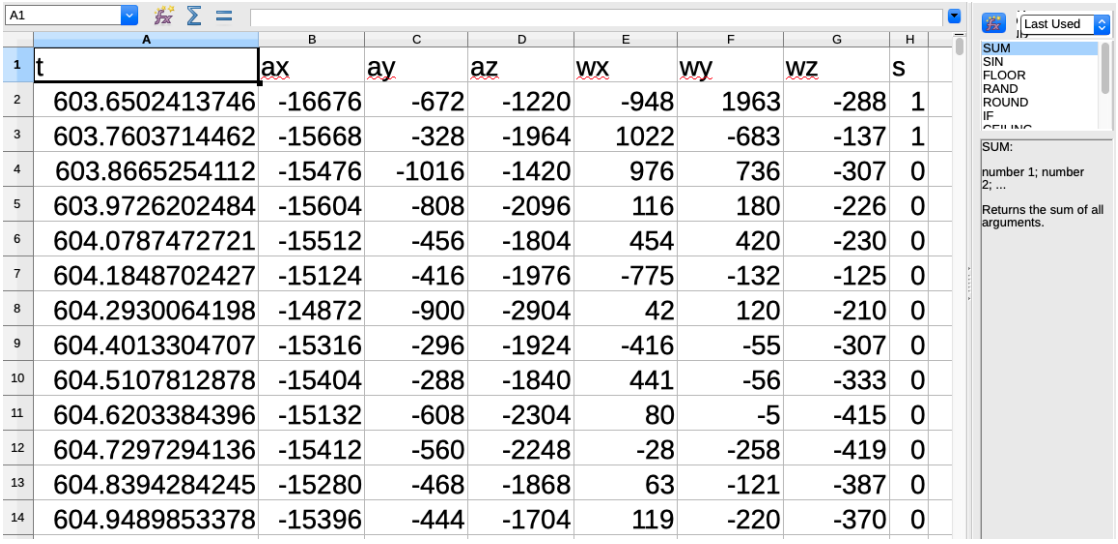
4.4 Beispiele

Wir wollen in diesem Abschnitt ein paar Denkanstöße geben, wie die Auswertung eines Trainingsdatensatzes aussehen könnte. Die hier verwendeten Daten sind online frei verfügbar[2]. Der Sensor befand sich in der Nähe des rechten Kniegelenks. Die markierten Zeitpunkte haben die Ereignisse gemessen, wenn der rechte Fuß auf den Boden aufgesetzt wurde (d.h. nach jeder Markierung sollte der Schrittzähler 2 Schritte gezählt haben).

4.4.1 Daten sichten und bereinigen

In Abbildung 5 sehen wir ein Beispiel, wie ein solcher Trainingsdatensatz aussehen kann. In der letzten Spalte wurde der Zeitpunkt erfasst, zu dem ein Schritt erfolgt ist. Man sieht in der Tabelle, dass dieser manchmal doppelt gelistet wird. Diese und andere Fehler in der Tabelle müssen zunächst gesichtet und korrigiert werden.

In Listing 5 wird diese Aufgabe (ebenso wie das Löschen einer unvollständigen Zeile) mit Python automatisiert.



	A	B	C	D	E	F	G	H
1	t	ax	ay	az	wx	wy	wz	s
2	603.6502413746	-16676	-672	-1220	-948	1963	-288	1
3	603.7603714462	-15668	-328	-1964	1022	-683	-137	1
4	603.8665254112	-15476	-1016	-1420	976	736	-307	0
5	603.9726202484	-15604	-808	-2096	116	180	-226	0
6	604.0787472721	-15512	-456	-1804	454	420	-230	0
7	604.1848702427	-15124	-416	-1976	-775	-132	-125	0
8	604.2930064198	-14872	-900	-2904	42	120	-210	0
9	604.4013304707	-15316	-296	-1924	-416	-55	-307	0
10	604.5107812878	-15404	-288	-1840	441	-56	-333	0
11	604.6203384396	-15132	-608	-2304	80	-5	-415	0
12	604.7297294136	-15412	-560	-2248	-28	-258	-419	0
13	604.8394284245	-15280	-468	-1868	63	-121	-387	0
14	604.9489853378	-15396	-444	-1704	119	-220	-370	0

Abbildung 5: Beispiel für einen Trainingsdatensatz. In der rechten Spalte **s** wurde markiert, zu welchem Zeitpunkt ein Schritt gezählt werden soll.

```
import pandas as pd
import numpy as np

data=pd.read_csv('daten2.csv')

#unvollstaendige letzte Zeile loeschen
data=data.drop(2634)
#doppelte Einsen in der Spalte 's' loeschen
mask=(data['s']==1) & (data.shift(1)['s']==1)
data.at[mask,'s']=0
```

Listing 5: Die Daten werden mit der Pythonbibliothek Pandas eingelesen und bereinigt.

[2] Erhältlich bei Kaggle über den Link `https://www.kaggle.com/patricklc/pedometer-school-project`.

4.4.2 Daten aufbereiten

Nun kann man die Tabelle ein wenig mehr unter die Lupe nehmen. Dazu kann es sinnvoll sein, die Daten weiter zu bearbeiten. Im Folgenden wurden die Marker in der Spalte **s** verwendet, um die Daten der Schritte voneinander zu trennen. Wenn wir jeden Schritt auf eine Zeitskala von 0 bis 1 umskalieren und die Schritte in ein gemeinsames Koordinatensystem plotten, bekommen wir eine Darstellung wie in Abbildung 6. Diese können wir nutzen, um erste Vermutungen anzustellen.

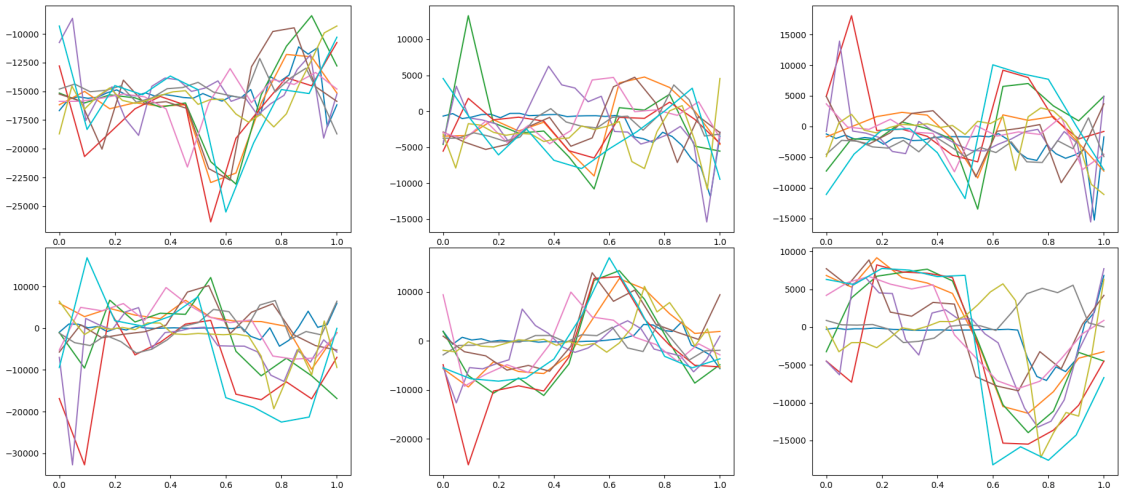


Abbildung 6: Plots von den Daten der ersten 10 Schritte; Oben die Beschleunigungsdaten in x-, y- und z-Richtung (von links nach rechts); Unten die Winkelgeschwindigkeitsdaten in x-, y- und z-Richtung (von links nach rechts). Die Zeitskalen wurden so transformiert, dass der Schritt bei 0 beginnt und bei 1 endet.

Auffallend ist etwa, dass die Winkelgeschwindigkeitsdaten dazu neigen, in der Mitte der Zeitskala einen Vorzeichenwechsel zu haben. Wir könnten uns also dazu entscheiden, den Vorzeichenwechsel zu verwenden, um einen Schritt vorherzusagen. Wir untersuchen die Daten nach folgendem Schema:

- Wir ergänzen die Datentabelle um eine weitere Spalte **h**, in der wir die Zeitpunkte markieren, die genau zwischen zwei Zeitpunkten liegen, die in der Spalte **s** markiert worden sind.
- Für alle Zeiten t in unserer Datentabelle vergleichen wir in der Spalte **wz** den Eintrag eine Zeile vor dem Zeitpunkt t und eine Zeile danach. Wenn ein Vorzeichenwechsel vorliegt, wird für den Zeitpunkt t angenommen, dass wir uns in der Hälfte zwischen zwei Schritten befinden, die in der Spalte **s** markiert worden sind.
- Wir vergleichen unsere Vorhersagen mit der Spalte **h**, wobei wir von einem Erfolg ausgehen, auch wenn wir bei der Vorhersage um einen Zeitschritt daneben liegen.

Die Ergebnisse dieser Analyse stellen wir in einer Vierfeldertafel (Abbildung 7) dar.

Wir können mit unserer Herangehensweise alle Schritte korrekt vorhersagen, machen jedoch auch viele Vorhersagen zu Zeitpunkten, an denen kein Schritt erfolgt ist. Wir sehen also, dass wir unser Modell erweitern müssen. Mit zusätzlichen Bedingungen können wir die Vorhersagegenauigkeit sicherlich verbessern.

	Schritt vorhergesagt	kein Schritt vorhergesagt
Schritt	211	0
kein Schritt	613	1810

Abbildung 7: Vierfeldertafel für die Vorhersage *Vorzeichenwechsel in Spalte* ***wz***.

4.4.3 Auswertung durch ein neuronales Netz

Selbstverständlich könnte man auch ein neuronales Netz darauf trainieren, die Schritte zu zählen. Hier ist es wichtig, dass ein möglichst großer Satz an Trainingsdaten existiert. Da bei den Daten eine zeitliche Komponente wesentlich ist, bieten sich sogenannte Rekurrente Neuronale Netze (RNN) an, z.B. ein Long Short-Term Memory (LSTM). Wir sehen ein Beispiel dafür in Listing 6.

Das Modell, das mit diesem Code trainiert wird, lässt sich exportieren und zu einem späteren Zeitpunkt nutzen, um bei eingehenden Daten Schritte zu erkennen. Wenn der Schrittzähler mit einem Raspberry Pi realisiert wird, kann das Modell in der Sprache Python geladen und genutzt werden.

Diese Herangehensweise eignet sich, um neuronale Netze kennen zu lernen und ihre Eigenschaften experimentell zu erforschen. Beispielsweise könnte man sich anschauen, was passiert, wenn man das Netz auf die Bewegung einer Person trainiert und dann eine andere Person den Schrittzähler verwendet. Aus mathematischer Sicht geht allerdings recht viel Potential verloren, da die Auswertung der Daten vollständig automatisiert wird.

```
import pandas as pd
import numpy as np

data=pd.read_csv('gy521.csv')

#remove incomplete last line
data=data.drop(2634)
#remove double ones in col 's'
mask=(data['s']==1) & (data.shift(1)['s']==1)
data.at[mask,'s']=0

predictors=['ax','ay','az','wx','wy','wz']

#build training data as a list of
#sequences of 12 consecutive timeframes
seqLength=12
X=[]
for i in range(len(data['s'])-seqLength):
   X.append(data.loc[i:i+seqLength-1,predictors])

#shape of X should now be:
#(numb. of timeframes-seqLength, seqLength, numb. of predictors)
X=np.array(X)
Y=data.loc[0:2634-seqLength-1,'s']
```

```
from keras.models import Sequential
from keras.layers import Dense, LSTM

model = Sequential()
model.add(LSTM(32, input_shape=(seqLength,6)))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='rmsprop', loss='binary_crossentropy',
    metrics=['accuracy'])
model.fit(X, Y, batch_size=32, epochs=30)
```

Listing 6: Ein rekurrentes neuronales Netz, das aus den Daten von 12 Zeitschritten eine Vorhersage macht, zu welchem Zeitpunkt ein Schritt erfolgt ist. Die Genauigkeit lag bei dem vorliegenden Datensatz bei ca. 92%. Es ist möglich, dass bei größeren Trainingsdatensätzen die Genauigkeit erhöht werden kann.

Literatur

[BBC17] Bock, W. ; Bracke, M. ; Capraro, P.: Product orientation in modeling tasks. In: *CERME Proceedings* 10 (2017), S. 1073–1075

[ras] raspberrypi.com tutorials: *Measuring Rotation and acceleration with the Raspberry Pi*. https://tutorials-raspberrypi.com/measuring-rotation-and-acceleration-raspberry-pi/, . – Accessed: 2020-03-18

[Sta] Stackoverflow: *Basiswissen Entwicklertypen – Die 12 häufigsten Entwicklertypen und ihre Skillsets im Überblick*. https://www.stackoverflowbusiness.com/hubfs/content/de/Basiswissen%20Entwicklertypen.pdf, . – Accessed: 2020-03-18

[Sta73] Stachowiak, H.: *Allgemeine Modelltheorie*. Springer-Verlag, 1973