Inhaltsverzeichnis

Einleitung:

Ziel

Funktionierender Beschleunigungssensor mit entsprechender Software.

Kurzbeschreibung

Das Ziel dieser Jugend Forscht Arbeit ist es, einen funktionierendes LowBudget Beschleunigungs-sensorsystem zu entwickeln, dieses mithilfe eines Programms auf einem Laptop oder einem RaspberryPI im Physikunterricht eingesetzt werden kann. Der Sensor soll bei linearen und bei kreisförmigen Beschleunigungen zum Einsatz kommen.

Idee:

Warum ich das Beschleunigungssensorsystem bauen wollte:

Das System wollte ich aus verschiedenen Gründen bauen:

1. Weil ich mich sehr für Technik interessiere,

2. Weil ich im Physikunterricht mitbekommen habe,dass ein Großteil der Schüler das Thema „Beschleunigungen“ nicht versteht. Ich überlegte, wie man Schülern auf interessante Weise Beschleunigungen näher bringen und darstellen kann. In Gesprächen mit meinem Physik Lehrer, konnte ich meine Vermutung bestätigen. So entschied ich mich, dieses Thema näher zu betrachten. Durch die Unterstützung meines Physiklehrers, Herr Droll, kamen wir auf die Idee einen Beschleunigungssensor zu verwenden. Fertige Systeme sind sehr teuer und meine Schule hat nicht Unmengen an Geld, für so ein System über. Aus diesem Grund musste eine „Low Budget“ Lösung her. Mein Physik-Lehrer empfahl mir daher, ein Beschleunigungssensorsystem als Jugend-forscht-Projekt zu entwickeln. Die Idee war geboren, doch an der Umsetzung haperte es, da ein Beschleunigungssensor allein, keine verwendbaren Werte liefert. Es fehlt ein Gerät, welches den Sensor ausliest, bzw. die Werte darstellt. Ich hatte die Wahl zwischen zwei Geräten: ich hätte einen RaspberryPI verwenden können oder ein Arduino Mikrocontrollerboard. Ich entschloss mich für den Arduino, da ich schon Erfahrungen mit diesem hatte und schon einige private Sachen damit gebaut habe (z.B. einen Tisch inklusive Discobeleuchtung). Der Arduino ist in der Lage die Daten auszulesen und zu bearbeiten. Aber er bietet keine Möglichkeit, diese anzuzeigen; dafür benötige ich noch einen Laptop und ein Bluetooth-Modul, welches das Kabel zur seriellen Datenübertragung ersetzt, damit dieses nicht bei den Messungen stört. Zusätzlich soll das ganze System sehr kompakt sein, damit man es vielseitig einsetzen kann.

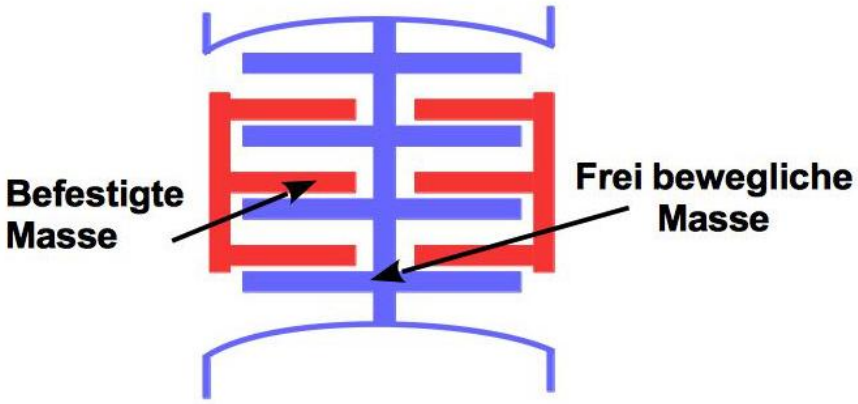
3. Zukunftsorientiert möchte ich mich mit dem Thema Software und Softwareentwicklung und Gamedevelopment befassen, da diese Themen meinen Berufs- und Studienwünschen entsprechen.

Umsetzung:

Schritt 1:

Wie ein Beschleunigungssensor funktioniert (wie misst er Beschleunigungen?):

Grundlegend funktioniert jeder Beschleunigungssensor nach folgendem Prinzip: Sie haben alle eine frei bewegliche Masse und eine befestigte Masse, die sich nicht bewegen kann. Diese Konstruktion ist meist kammförmig.



Zeichnung nach Robert Bosch GmbH: Funktionsprinzip Beschleunigungssensor

Bei einer Beschleunigung, bewegt sich die frei bewegliche Masse in die entgegengesetzte Richtung, in die der Sensor bewegt wurde. Dadurch, dass diese beiden Massen aus Metall bestehen, ändert sich die elektrische Kapazität zwischen den beiden Massen. Diese Änderung wird von der Messeinheit erfasst, umgerechnet und in Volt ausgegeben. Diese Werte werden an den analogen Ausgängen des Sensors bereit gestellt. Der Sensor verfügt über drei dieser analogen Ausgänge: für jede Achse einen (x, y, z). Diese analogen Werte, die an den Achsen ausgegeben werden, können nun über einen analogen Eingang am Arduino eingelesen werden.

Wie binde ich den Beschleunigungssensor an den Arduino an:

Ein Arduino ist ein Mikrocontroller, dieser funktioniert wie jeder normale Mikrochip. Er verfügt über einen Mikrochip, sozusagen das Gehirn des Arduino. Dort werden die Programmcodes verarbeitet und in Elektroimpulse verwandelt. Mit diesen Impulsen steuert der Mikrochip die restlichen Bestandteile des Boards, wie z.B. einen digitalen Pin (Digitalpin)

Der Arduino ist wie folgt aufgebaut:
Er verfügt über 13 digitale und 5 analoge Pins. Diese Pins werden verwendet, um passende Bauteile anzuschließen, z.B. LEDs oder, wie in diesem Projekt, einen Beschleunigungssensor.
Es gibt spezielle Schnittstellen, die für andere Dinge verwendet werden:

- 2 Ground Ausgänge, um Bauteile mit negativer Spannung zu versorgen
- einen 5 Volt und einen 3,3 Volt Ausgang, um Bauteile mit positiver Spannung zu versorgen
- einen Resetpin
- einen Arefpin
- 3 serielle Schnittstellen, 1x USB-Kabel Anschluss, 1x Tx-Pin, 1x Rx-Pin, der Tx- und der Rx-Pin sind ebenfalls serielle Schnittstellen, an denen man externe Geräte anschließen kann um Daten zu übermitteln. Der Tx-Pin ist der Transmitter, also der Sender und der Rx-Pin der Receiver, also der

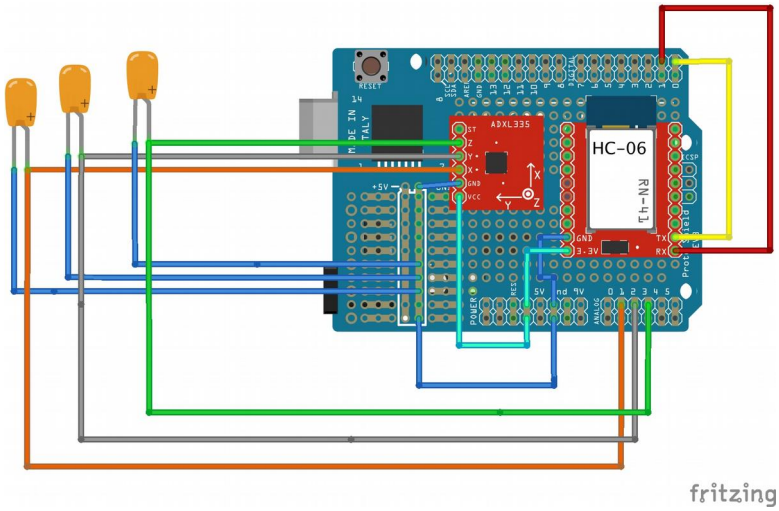
Empfänger.

- eine Schnittstelle zum Anschließen eines Netzteils

Die Teile die ich verwende, um den Arduino (Arduino Uno, Atmega328P) anzuschließen sind:

- Ein Prototypingboard. Dieses ist ein spezielles shield (ein Aufsatz, der auf den Arduino aufgesteckt wird) mit einem Breadboard in der Mitte, dieses wird verwendet um Bauteile zu verdrahten.
- einen Laptop
- den Beschleunigungssensor ADXL 335
- mehrere Drähte
- 3 Kondensatoren
- ein Bluetooth-Modul
- ein Batterie betriebenes Netzteil

Die Kabel habe ich wie unten in der Skizze zu sehen, am Arduino angeschlossen.
3 Drähte stecken an 3 Analogenpins, ein Draht steckt im Tx-Pin, ein Draht steckt im Rx-Pin, ein weiteres steckt am 3,3 Volt Ausgang und eines am „Ground“. Die Drähte, die an den Analogpins angeschlossen sind, sind mit dem anderem Ende am jeweiligen Pin der Achse des Sensors angeschlossen. Das Kabel welches am „Ground“ angeschlossen ist, ist mit dem anderem Ende an der generellen negativen Spannungsverteilung des Breadboards angeschlossen. Dasselbe habe ich mit einem weiteren Kabel und dem 3,3 Volt Ausgang gemacht. Die 3 Kondensatoren stecken an den Ausgängen der 3 Achsen des Sensors. Mit dem anderen Ende stecken diese an der negativen Spannungsverteilung des Breadboards. Der Draht des Tx-Pins führt in den Rx-Pin des Bluetooth-Moduls, da der Transmitter des Arduinos die Daten an den Receiver des Bluetooth-Moduls senden muss, dementsprechend führt der Draht des Rx-Pin des Arduinos zum Tx-Pin der Bluetooth-Moduls. In diesem Bild ist das alles besser zu sehen:



Agentur nicht vorhanden, Leon Knorr

Schritt 2:

Das erste Testprogramm:

Als zweiten Schritt habe ich ein Testprogramm für den Arduino entwickelt, damit dieser den Sensor ausliest und die ausgelesenen Daten an den PC weiterleitet (siehe Anhang 1). Dort können die Werte im sogenannten seriellen Monitor (Bestandteil der Arduino Software) gelesen werden. In diesem Programm habe ich den x-Pin (der Pin wo die x-Achse angeschlossen ist) ausgelesen und die Daten ausgeben lassen. Später habe ich noch die restlichen Pins ausgelesen (y und z). Der Arduino lag dabei ruhig auf dem Tisch, x und y lagen parallel und z lag senkrecht zum Tisch. Dadurch misst diese Achse (z) alle Beschleunigungen die senkrecht und x und y die parallel zum Tisch ausgeübt werden. Dementsprechend habe ich auf der z-Achse eine gerade Linie erwartet, die höher liegt als die Linien von x und y, da diese die Erdbeschleunigung messen müsste. Auf der x- und der y-Achse habe ich gerade Linien erwartet, die auf der selben Höhe liegen. Das Ergebnis hat jedoch meine Erwartungen nicht erfüllt. Ich hatte auf allen Achsen die selbe Ausgabe. Diese ergaben auch keine gerade Linie, sondern hatten periodische Ausschläge (siehe Anhang 2). In diesem Programmcode lief die Datenübertragung über Kabel.

Schritt 3:

Fehlersuche:

Nachdem ich mir die Ergebnisse nochmal genau angesehen, den Programmcode und die Schaltung überprüft hatte, ging ich zu meinem Vater und fragte ihn, ob er sich mit mir zusammen das Ganze mal anschauen könnte. Wir fanden den Fehler sehr schnell: der Sensor steckte falsch herum auf dem Breadboard, wodurch die Achsen nicht ausgelesen worden sind. Die Werte die angezeigt wurden waren die 3,3V die ursprünglich an den Sensor angeschlossen sein sollten. Dieser Fehler ist mir unterlaufen, weil ich beim Verdrahten des Sensors mit dem Arduino, die Verschaltung des Breadboards nicht beachtet hatte. Zusätzlich zu diesem Fehler habe ich noch vergessen, die Kondensatoren zu verwenden.

Schritt 4:

Verbesserung des 1. Programms

Wir schauten uns sofort danach das Programm für den Arduino an und auch die damit verbundenen Ausgaben. Das erste, was meinem Vater auffiel, war, dass in dem Programm die Ausgabe des Sensors nicht umgerechnet wurde. Mein Vater erklärte mir, das ein Mikrochip in einer Einheit rechnet. Diese Einheit wird auch Digits genannt. Die Menge an Digits die ein Mikrochip berechnen kann, ist je nach Modell unterschiedlich. In diesem Fall sind es 1024 Digits. Der Arduino kann maximal 5V ausgeben und einlesen, diese 5V entsprechen in der Skala des Arduinos 1024 Digits. Der Sensor kann aber nur 3V ausgeben. Allerdings kann man diesen auch mit 3,3V Spannung versorgen. Da der Sensor nur eine Ein- und Ausgabe von 3V hat und ich ihn mit einer Spannung von 3,3V versorge, wird die Ausgabe von dem Sensor zum Arduino ungenau. Zusätzlich kommt hinzu, dass der analoge Eingang am Arduino eine Lesespanne von 0 bis 5V hat, vom Sensor kommen aber nur 3,3V. Dies führt dazu, dass die Werte, die am Ende im Laptop

ankommen, noch ungenauer sind, als sie ohnehin schon waren.
Aber nun zur Rechnung:
Legende:

UG = Untere Grenze der erfassbaren Messwerte ($-3\,g$)

OG = Obere Grenze der erfassbaren Messwerte ($+3\,g$)

MWE = Messwert-Ende (mehr „Digits“ liefert der Sensor nicht an den Arduino)

MW = Der gemessene Wert (in Digits)

Es ist gegeben:
1x Beschleunigungssensor Typ: ADXL 335.
3x Sensorausgang: 1x Beschleunigung für x-Achse, 1x Beschleunigung für y-Achse, 1x Beschleunigung z-Achse.
Die Ausgabe der Sensorausgänge: 0 - 3,3 Volt entsprechen einer Beschleunigung von $\pm 3\,g$.
Besonderheit: 1,65 Volt entsprechen einer Beschleunigung von $0\,g$.
Erkenntnis: Der Messbereich teilt sich in einen positiven und einen negativen Messbereich auf.

Das zur Messwerterfassung dienende Arduino-Board besitzt 3 analog Eingänge. Diese werden zum Anschluss des Sensor benötigt.
Jeder dieser Analogeingänge hat einen Messbereich von 0 - 5 Volt. Das entspricht in der Software einem Messbereich von 0 - 1024 digits.

Aufgabenstellung:
Die vom Sensor übermittelten Beschleunigungswerte sollen im Seriellen Monitor als reale *g*-Werte angezeigt werden.

Problem:
Im seriellen Monitor werden die Werte in „digits“ angezeigt.

Lösung:
Entwicklung einer Formel zur Umrechnung der in „digits“ ausgegebenen Beschleunigungswerte in reale *g*-Werte.

Entwicklung der Formel:

$$\begin{aligned} 5\,V &\triangleq 1024\ \text{digits} \\ 3{,}3\,V &\triangleq MWE\ \text{digits} \end{aligned} :$$

$\overset{\text{Dreisatz}}{\Rightarrow}$ vom Arduino maximal empfangen: $MWE = \frac{1024\ \text{digits}}{5\,V} \cdot 3{,}3\text{V} = 675{,}84\ \text{digits}$

Mit einem zweiten Dreisatz ermittle ich eine Skala zur Ermittlung der Beschleunigung aus den Messwerten.

$$\begin{array}{ccc} OG-UG & \hat{=} & MWE \\ a^* & \hat{=} & MW \end{array}$$

$$\overset{\text{Dreisatz}}{\Rightarrow} \quad a^* = \frac{OG-UG}{MWE} \cdot MW \ .$$

Diese Skala fängt bei 0*g* an und endet bei 6*g*. Da der Sensor allerdings von -3*g* bis +3*g* misst, subtrahiere ich von dem Ergebnis die obere Grenze oder addiere die negative untere Grenze. Damit liegt die *g*-Skala korrekt zwischen -3*g* und +3*g.*

Ergebnis:

$$a = a^* + UG = \left[\left(\frac{OG-UG}{MWE}\right) \cdot MW\right] + UG$$

in Worten:

$$\text{Beschleunigung} = \left[\left(\frac{\text{Obere Grenze} - \text{Untere Grenze}}{\text{Messwert Ende}}\right) \cdot \text{Messwert}\right] + \text{Untere Grenze}$$

Diese Formel habe ich dann in das Programm einprogrammiert (siehe Anhang 3). Daraufhin habe ich noch einmal den Arduino angeschlossen, das neue Programm auf den Arduino geladen und noch einmal alles getestet.
Die Ergebnisse entsprachen ganz meinen Erwartungen.

Verbesserung des 2. Programms:

Da in den 2 vorherigen Programmcodes die Datenübertragung noch über Kabel funktionierte, habe ich in diesem Programmcode die serielle Datenübertragung von Kabel auf Bluetooth umgestellt. Dies funktionierte reibungslos.

Schritt 5:

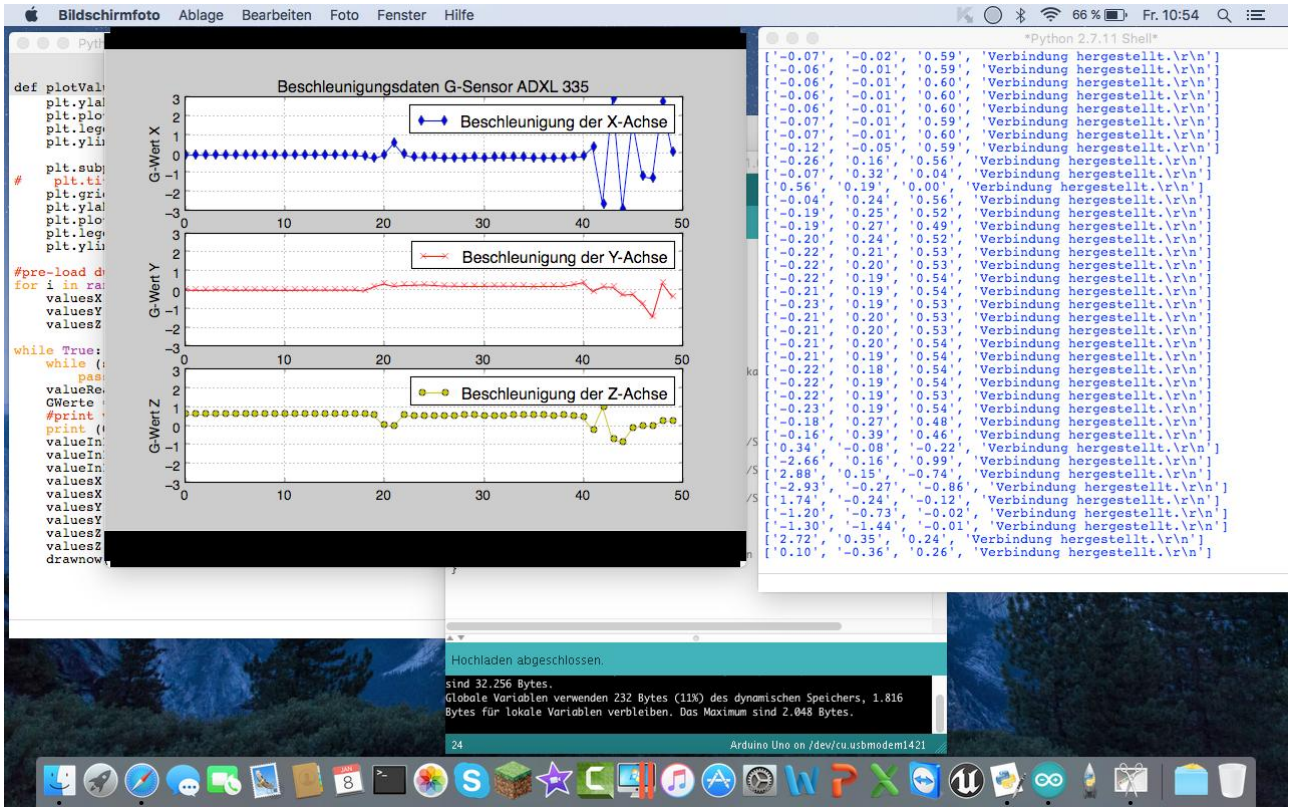
Visualisierung:

Nachdem die Software und die Hardware lief, machte ich mir Gedanken, wie man die Daten visualisieren kann. Dafür habe ich mit Unterstützung meines Vaters etwas im Internet recherchiert. Wir sind dabei auf zwei Erweiterungen für die Programmiersprache „Python“ gestoßen. Mit Hilfe dieser Erweiterungen ist es möglich, Graphen zu zeichnen, die in Echtzeit aktualisiert werden. Erst hatten wir die Idee, alle Kurven pro Achse am Sensor in ein Koordinatensystem zu schreiben und zeichnen zu lassen. Doch dies erwies sich als schwierig und sehr unübersichtlich. Nach weiterer Recherche sind wir dann auf einen Befehl gestoßen, der es ermöglicht, mehrere Koordinatensysteme gleichzeitig anzeigen zu lassen. Nach mehreren Versuchen gelang es uns, mehrere Graphen zu zeichnen. Doch es gab ein Problem mit der Übertragung der Daten. Die Daten, die der Arduino sendet, werden in eine Zeile geschrieben und mit einem Semikolon getrennt. Die Erweiterungen für Python können aber nur einen Wert aus einer Zeile herauslesen.

Nach weiterer Recherche fanden wir einen Befehl der es möglich macht, die Zeile in einzelne Teile zu unterteilen und in eine Liste zu schreiben. Aus dieser kann dann das Programm, die einzelnen benötigten Werte herauslesen. Das bedeutet, dass das Python-Programm (siehe Anhang 4) die Daten verarbeiten kann und die passenden Graphen zeichnen kann. Dieses Programm ist nicht für den RaspberryPI geeignet, da der Raspberry es leistungstechnisch nicht in einer adäquaten Geschwindigkeit bewerkstelligen kann, die Daten zu verarbeiten und zu zeichnen. Für den RaspberryPi ist ein weiteres Programm namens „Processing“ in Entwicklung. Dieses ist in der Programmiersprache Java geschrieben und darauf spezialisiert, mithilfe eines selbstgeschriebenem Programmcodes des Users, graphische Darstellungen zu erzeugen.

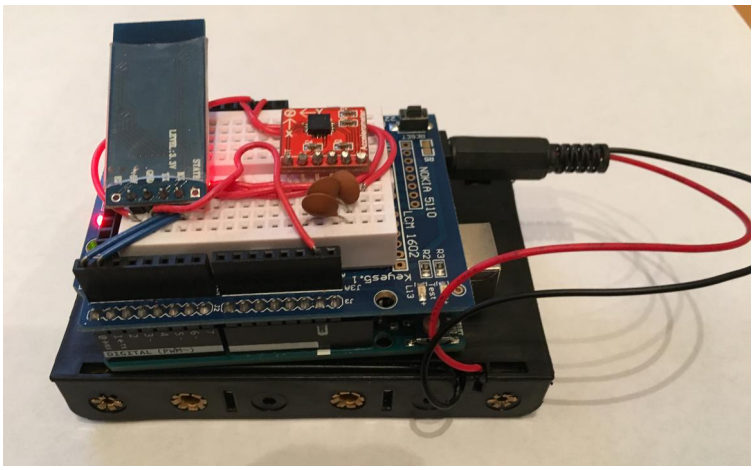
Ergebnisse:

Nach Fertigstellung beider Programme sieht das Endergebnis folgendermaßen aus:



Links in dem kleinen Fenster werden die drei Graphen zu den jeweiligen Achsen gezeichnet und rechts werden die *g*-Daten noch einmal ausgegeben, damit man überprüfen kann ob alles richtig funktioniert. Unter anderem kann man rechts sehen, ob eine Verbindung aufgebaut ist oder nicht. Sollten keine Meldungen und Daten mehr angezeigt werden, besteht keine Verbindung mehr zum Arduino.

Der Hardwareaufbau stellt sich so dar:



Agentur nicht vorhanden, Leon Knorr

Was habe ich selbst herausgefunden?

Wie man oben sicherlich herauslesen kann, wurde ich von einigen Personen unterstützt. Komplett alleine habe ich den Test-Code geschrieben, sowie den endgültigen Programmcode inklusive der Datenübertragung via Bluetooth. Bei dem Rest wurde ich unterstützt, vor allem von meinem Vater.

Ergebnisdiskussion

Ich bin sehr zufrieden mit dem Endergebnis dieser Jugend-Forscht Arbeit. Mir hat das Bauen und das Programmieren des Ganzen sehr viel Spaß gemacht. Ich finde, das ist das Wichtigste an der ganzen Sache. Ich habe auch gemerkt, das eine Jugend-Forscht-Arbeit sehr viel Zeit beansprucht und ich mehr Zeit hätte investieren sollen, um das Ergebnis noch besser machen zu können. Was sehr gut gelaufen ist, war die Zusammenarbeit und die Unterstützung mit all denen, die mich unterstützt haben.

Zusammenfassung:

Mein Ziel habe ich erreicht und übertroffen. Mein eigentliches Ziel war es die Daten auf dem Laptop in der Form von Zahlen anzeigen zu lassen, die Visualisierung ist ein Extra, um das nächste Ziel zu erreichen: Dieses war es, Beschleunigungen für Schüler visuell darzustellen, damit diese das Thema besser verstehen. Außerdem wollte ich den Schulen im Physikunterricht ein Komplettsystem bieten, welches günstig und benutzerfreundlich ist, damit jeder mit diesem System arbeiten kann. Dies soll mit einem RaspberryPI realisiert werden.

Quellen und Unterstützungsleistungen

Unterstützungsleistungen:

Andy Knorr, staatlich geprüfter Techniker in der Fachrichtung Elektrotechnik, Schwerpunkt: Energietechnik und Prozessautomatisierung, Bilfinger Watertechnologys GmbH.

Unterstützungsleistung: Lieferant weiterführender Informationen, Hilfe bei Programmtest, Hilfe bei der Fehlersuche, Hilfe bei der Verbesserung der Software, Beratung bei der Software, Wahl und Korrektur der Software und der Verdrahtung der Hardware.

Edgar Droll, Lehrer für die Fächer: Mathematik, Physik, Informatik, Marion-Dönhoff-Gymnasium Lahnstein. Unterstützungsleistung: Korrektur von Formeln und Bereitstellung der Hardware.

Paul Lambrecht, Schüler im freiwilligem Sozialen Jahr, Marion-Dönhoff-Gymnasium, Lahnstein. Unterstützungsleistung: Hilfe bei der Wahl der Programmiersprache des Visualiesierungsprogramms, Korrektur von Texten, Lieferant von Informationen und Lehrer der Programmiersprache.

Quellen:

Internetlinks

https://www.sparkfun.com/pages/accel_gyro_guide , 4.9.15, Sparkfun Electronics Inc, weiterführende Informationen.

https://www.arduino.cc/en/Reference/HomePage , letzter Aufruf: 8.1.16, Arduino LLC, Syntax und der Programmiersprache für den Arduino.

https://www.youtube.com/watch?v=KZ4Zcc74RHo, letzter Aufruf: 8.1.16, Google Inc, weiterführende Informationen.

https://www.youtube.com/watch?v=KZVgKu6v808, letzter Aufruf: 8.1.16, Google Inc, weiterführende Informationen.

http://matplotlib.org/api/pyplot_summary.html, letzter Aufruf: 8.1.16, The matplotlib development team, Syntax und Befehle für die Erweiterung Matplotlib der Programmiersprache Python.

https://www.python.org, letzter Aufruf 8.1.16,

http://www.dotnetperls.com/split-python, letzter Aufruf: 8.1.16, Sam Allen, weiterführende Informationen.

https://www.youtube.com/watch?v=8eM0qpSuIlw, 8.10.15, Google Inc, Idee für die Visualisierung.

https://processing.org, letzer Aufruf: 5.3.16, Processing, Grundlage für die Visualisierung der Daten auf dem RaspberryPI.

PDF-Dateien:

ADXL335_v10.pdf, 4.9.15, Datasheet.
Runtergeladen von: https://lms.bildung- rp.de/lahnstein/mod/wiki/view.php?pageid=171

ADXL335.pdf, 4.9.15, Datasheet.
Runtergeladen von: https://lms.bildung-rp.de/lahnstein/mod/wiki/view.php?pageid=171

Beschleunigungssensoren_PdN.pdf,4.9.15, Informationen über die Funktionsweise eines Beschleunigungssensors.
Runtergeladen von: https://lms.bildung-rp.de/lahnstein/mod/wiki/view.php?pageid=171

Wiring.pdf, 4.9.15, Informationen über das Anschließen eines Beschleunigungssensors. Runtergeladen von: https://lms.bildung-rp.de/lahnstein/mod/wiki/view.php?pageid=171

Anhang 1

Programmcode des ersten Testprogramms

Legende:

„//“ ist eine Textzeile, das bedeutet dass das Programm den Teil hinter dem „//“ ignoriert

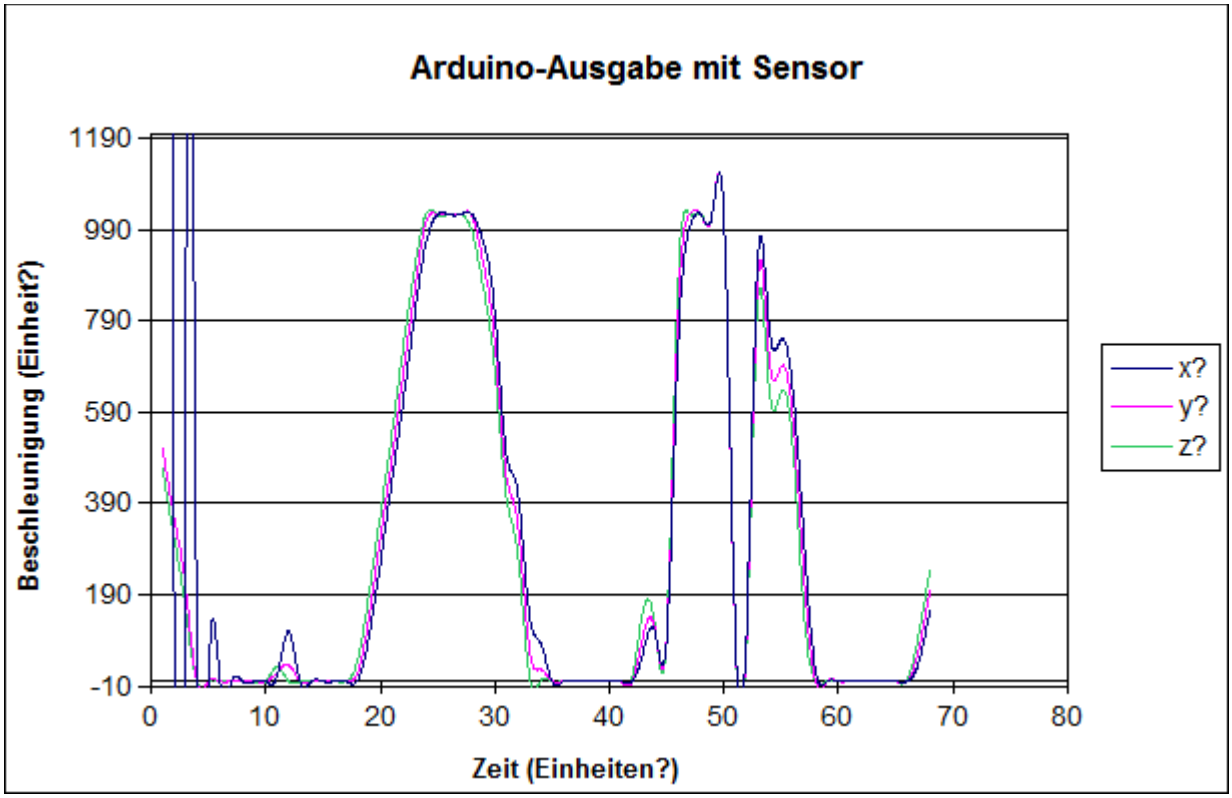
```
const int x = A3; //X-Pin am Sensor
const int y = A2; //Y-Pin am Sensor
const int z = A1; //Z-Pin am Sensor

void setup() {
Serial.begin(9600); //Stellt die Serielle Kommunikation her
}

void loop() {
Serial.print(analogRead(x)); //sendet die Sensor Ausgaben des x-Pin an den
Computer
Serial.print("\t"); //lässt eine Lücke
Serial.print(analogRead(y)); // sendet die Sensor Ausgaben des Y-Pin an den
Computer
Serial.print("\t"); //lässt eine Lücke
Serial.print(analogRead(z)); // sendet die Sensor Ausgaben des Z-Pin an den
Computer
Serial.println(); //Fängt eine neue Zeile an
delay(500); // Zeit zwischen dem nächsten Auslesen
}
```

Anhang 2

Graphische Darstellung der ersten Messung (EXCEL)



Anhang 3

Programmcode mit Berechnung der g-Werte:

```
const int x = A3; //X-Pin am Sensor
const int y = A2; //Y-Pin am Sensor
const int z = A1; //Z-Pin am Sensor
float UG = (-3.0);
float OG = 3.0;
int MWE = 675;

void setup() {
Serial.begin(9600); //Stellt die Serielle Kommunikation her
}

void loop() {
Serial.print((((OG-UG)/MWE)*analogRead(x))+UG); // ließt den X-Pin aus, rechnet
die ausgelesenen Messwerte in reale G-Werte um und sendet diese an den Computer.
Serial.print(";"); //sendet ein „;“ an den Computer
Serial.print((((OG-UG)/MWE)*analogRead(y))+UG); // ließt den Y-Pin aus, rechnet
die ausgelesenen Messwerte in reale G-Werte um und sendet diese an den Computer.
Serial.print(";");
Serial.print((((OG-UG)/MWE)*analogRead(z))+UG); // ließt den Z-Pin aus, rechnet
die ausgelesenen Messwerte in reale G-Werte um und sendet diese an den Computer.
Serial.print(";");
Serial.print("Verbindung hergestellt."); //sendet "Verbindung hergestellt." an
den Computer.
Serial.println(); //Fängt eine neue Zeile an
delay(500); // Zeit zwischen dem nächsten Auslesen
}
```

Anhang4:

3. Arduino Programm:

```
#include <SoftwareSerial.h> // importiert die Benötigte Erweiterung

#define rxPin 0 //definiert die seriellen Pins
#define txPin 1

const int x = A1; //X-Pin am Sensor
const int y = A2; //Y-Pin am Sensor
```

```
const int z = A3; //Z-Pin am Sensor
float UG = (-3.0);
float OG = 3.0;
int MWE = 675;

SoftwareSerial btSerial(rxPin, txPin); //definiert die seriellen Pins als Aus- und Eingang

void setup() {
  // put your setup code here, to run once:
btSerial.begin(9600); //Stellt die Serielle Kommunikation her
btSerial.println("connected");
}

void loop() {
  // put your main code here, to run repeatedly:
btSerial.print((((OG-UG)/MWE)*analogRead(x))+UG); //Schreibt die Sensor Ausgaben des x-Pin in Zahlen
und sendet sie via Bluetooth an den Laptop/Raspberry.
btSerial.print(";"); //schreibt ein Simicolon
btSerial.print((((OG-UG)/MWE)*analogRead(y))+UG); //Schreibt die Sensor Ausgaben des Y-Pin in Zahlen
btSerial.print(";");
btSerial.print((((OG-UG)/MWE)*analogRead(z))+UG); //Schreibt die Sensor Ausgaben des z-Pin in Zahlen
btSerial.print(";");
btSerial.print("Verbindung hergestellt.");
btSerial.println(); //Faengt eine neue Zeile an
delay(500); // Zeit zwischen dem nechsten Auslesen
}
```

Anhang 5

Legende:

„#“ das „#“ definiert den Anfang einer Textzeile, alles was hinter einem „#“ steht wird von dem Programm ignoriert.

Python-Programmcode zur graphischen Darstellung der g-Werte:

```
import serial #Importiert die Erweiterung mit dem Namen „serial“
import matplotlib.pyplot as plt #Importiert die „matplotlib.pyplot“ Erweiterung
als plt
from drawnow import * #Importiert die „drawnow“ Erweiterung

valuesX = []
```

```
valuesY = []
valuesZ = []

plt.ion()
cnt=0

serialArduino = serial.Serial('/dev/cu.usbmodem1421', 9600) #Definiert die Übertragungsrate der Daten und die Schnittstelle an die der Arduino angeschlossen ist (bei jedem unterschiedlich).

def plotValues(): #Definiert eine Variable mit dem Namen plotValues
    plt.subplot(3,1,1) #Durch diesen Befehl weis der Computer in wie vielen Reihen und Spalten die Graphen später stehen und für den Wievielten Graph die unteren Einstellungen ,bis zu dem Absatz, gelten.
    plt.title('Beschleunigungsdaten G-Sensor ADXL 335') #In dieser Zeile wird der Name des Graphen definiert.
    plt.grid(True) #Zeichnet im späteren Graphen ein Raster.
    plt.ylabel('G-Wert X') #beschriftet die Y-Achse.
    plt.plot(valuesX, 'bd-', label='Beschleunigung der X-Achse') #Einstellungen für die Kurve.
    plt.legend(loc='upper right') #Definiert an welcher Position später die Legende stehen wird.
    plt.ylim(-3.0, 3.0) #skalliert die X-Achse

    plt.subplot(3,1,2)
    plt.grid(True)
    plt.ylabel('G-Wert Y')
    plt.plot(valuesY, 'rx-', label='Beschleunigung der Y-Achse')
    plt.legend(loc='upper right')
    plt.ylim(-3.0, 3.0) #skalliert die Y-Achse

    plt.subplot(3,1,3)
    plt.grid(True)
    plt.ylabel('G-Wert Z')
    plt.plot(valuesZ, 'yo-', label='Beschleunigung der Z-Achse')
    plt.legend(loc='upper right')
    plt.ylim(-3.0, 3.0) #skalliert die Y-Achse

#Läedt die Daten für die X-Achse des Koordinatensystems neu und definiert das immer die letzten 50 Werte im Graphen angezeigt werden.
for i in range(0,50):
```

```
    valuesX.append(0)
    valuesY.append(0)
    valuesZ.append(0)

while True: #Anfang einer unendlichen Schleife
    while (serialArduino.inWaiting()==0): #sobald der Arduino keine Daten mehr
sendet wird die Schleife unterbrochen..
        pass
    valueRead = serialArduino.readline() #Ließt die serielle Schnittstelle aus.
    GWerte = valueRead.split(";") #Teilt die Daten die der Arduino sendet am
Semikolon und schreibt die geteilten Daten in eine Liste.
    print (GWerte) #zeigt die Daten in der Liste an.
    valueInIntX = float(GWerte[0]) #definiert welcher Wert die Kurve des Graphen
der X-Achse hat.
    valueInIntY = float(GWerte[1]) #definiert welcher Wert die Kurve des Graphen
der Y-Achse hat.
    valueInIntZ = float(GWerte[2]) #definiert welcher Wert die Kurve des Graphen
der Z-Achse hat.
    valuesX.append(valueInIntX) #zeichnet den Wert im Koordinatensystem ein.
    valuesX.pop(0) #verhindert das der oberste Wert dauerhaft in das Koordina-
tensystem gezeichnet wird.
    valuesY.append(valueInIntY)
    valuesY.pop(0)
    valuesZ.append(valueInIntZ)
    valuesZ.pop(0)
    drawnow(plotValues) #zeichnet die Graphen.
```