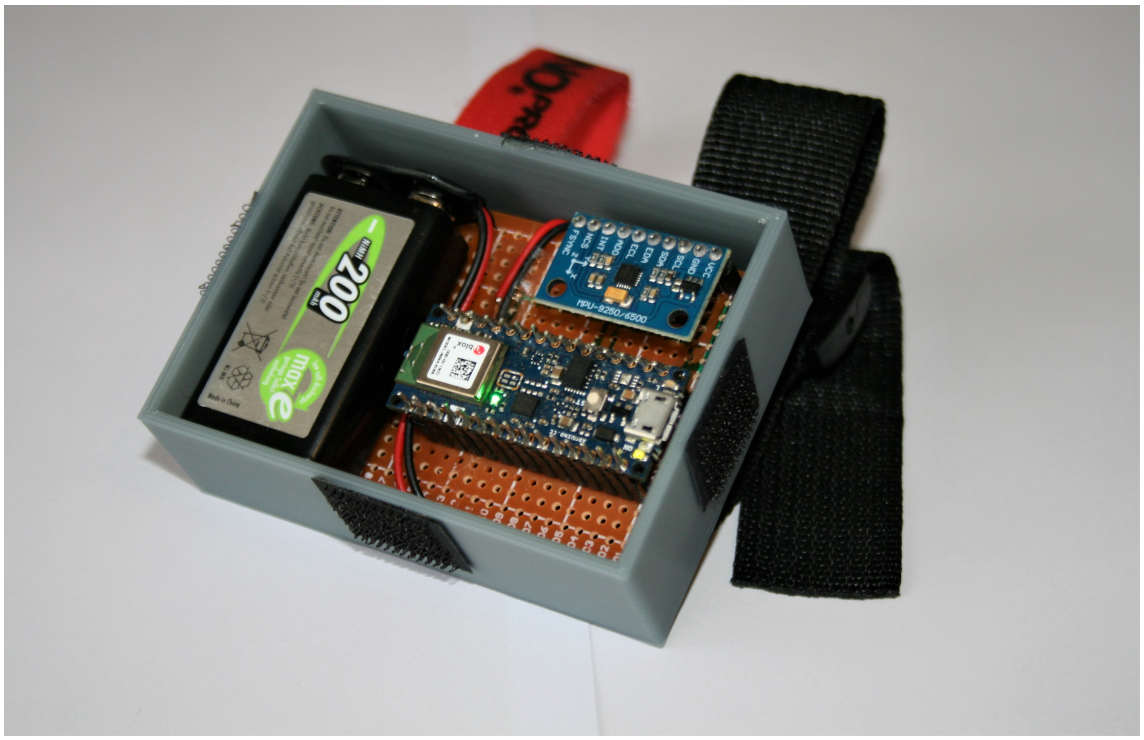

Archer-Tracking

Bewegungstracking für Sportler

Antonio Rehwinkel

31. Dezember 2021

Unterstützende Lehrer: Herr Czernohous
Herr Dierle
`Emails?`

und Professoren: Hochschule1
Hochschule2
`Emails?`

Inhaltsverzeichnis

1 Vorwort

Mithilfe eines Beschleunigungssensoren kann man viele Bewegungen erforschen und vermessen. Mit meinem System sollen mehrere Beschleunigungssensoren dazu eingesetzt werden können, Bewegungsabläufe aufzunehmen, miteinander zu vergleichen und zu erkennen.

Die Daten werden über Bluetooth-Low-Energy an ein Handy geschickt, wo Sie sowohl gespeichert als auch verwertet werden können. Als Beispiel gilt hier für mich das Bogenschießen, bei dem selbst kleine Bewegungen immer wieder auf gleiche Weise ausgeführt werden müssen. Mit meinen Sensor sollen hier teure Kamerasysteme abgeschafft werden und es so jeden ermöglichen, selbst ohne Bogen oder Trainer bei sich Zuhause zu den Bewegungsablauf zu trainieren.

1.1 IMU vs Kameratracking

Mein Projekt überschneidet sich in seinen Zielen häufig mit Tracking das bei VR-Brillen eingesetzt wird. Hier wird zur Feststellung der Position des Spielers häufig eine Kombination aus Kameratracking und Infrarot-LED.
Hierbei muss der Spieler die Fernbedienungen festhalten die die Infrarot-LEDs beinhalten, während die Kameras im Raum so verteilt werden müssen das der Spieler immer erkannt wird.
Die Neigung des Kopfes und der Hände werden auch hier häufig Mithilfe eines IMU bestimmt.

Da diese Systeme viel Platz benötigen, viel Geld kosten und für die Bildverarbeitung häufig eine große Rechenkraft benötigen ist dieses System nicht für viele Privatnutzer sinnvoll oder bieten einen Bewegungsfreiraum der Sport zu lässt.

Die IMU-Sensoren bestehen mindestens aus einem Gyroskop und einem Beschleunigungssensor, manche bieten sogar ein Magnetometer an. Somit sollte es möglich sein, über die Beschleunigung die Distantz die ein Körper mit diesem Sensor zurück legt zu messen. Die Neigung und Orientation sind über das Gyroskop und Magnetometer sehr genau messbar.

Die Vorteile der IMU liegen auf der Hand, sie sind günstig, klein und leicht. Aus diesen Gründen trägt fast jeder heutzutage so einen Sensor bei sich, die meisten Handys haben ihn schon eingebaut.

Für mein Projekt benutze ich dennoch einen eigenen IMU, um die Qualität der Messdaten sicher zu stellen.

1.2 Bewegungen und Analyse

Eine Interessante Bewegung stellt vor allem der Zugarm des Schüzten dar. Der Auszug verläuft nahezu linear, der häufigere Fehler an dieser Stelle versteckt sich allerdings in der Höhe des Zugarms. Um diese zu messen muss man die Erdanziehungskraft der Z-Achse herausrechnen.

Da der Schussablauf eines Bogenschützen viele Stationen mit verschiedenen Bewegungen beinhaltet fällt es häufig sogar den Trainern schwer zwischen einem technisch guten oder schlechtem Schuss zu unterscheiden.
Die Datenlage aus dem 9DOF-System des verwendeten MPU erzeugt eine Datenwolke, die diesen Vorgang fürs erste verkompliziert.
Die Daten sind allerdings sehr gut zu vergleichen und zu mitteln. Mit diesen Eigenschaften kann man über künstliche Intelligenz, genauer, Machine Learning die Schüsse klassifiezieren. So kann jeder Schütze seine eigene Datenbasis erstellen, nach der sein Schuss klassifiziert und so eingeordnet werden können.
Ein Vergleich mit einer deutlich größeren Datenbasis als einem einzelnen Schützen ist denkbar.

Für dieses System ist es nahezu unwichtig wie viele Körperteile überwacht werden. Mit mehr Sensoren (oder Vergleichspunkten) wird es lediglich schwieriger für den Schützen einen guten Wert bei der klassifiezierung zu erreichen.
Die Verwendung setzt natürlich vor allem korrekte, genau und viele Daten vorraus, dies gilt für das Training des Modells so wie für die Live-Daten des Schützen.

2 Bluetooth-Low-Energy

Mit Bluetooth 5.0 wurde eine neue übertragunsweise zu Bluetooth hinzugefügt. Diese nennt sich Bluetooth-Low-Energy und zeichnet sich durch einen geringen Stromverbrauch und damit einem höherem Datendurchsatz aus.

Bluetooth sendet Daten in Paketen. Hierbei ist bei Bluetooth-Low-Energy (zukünftig BLE) der Sender als Server ausgewiesen und der Empfänger als Client.

Hierdurch kann der Client bis zu??? Server abfragen ohne sich mit diesen verbinden zu müssen. Jeder Server kann unbegrenzt Characteristiken anbieten, diese Stellen verschieden Datensätze dar, die vom Client abgefragt werden können.

Laut Dokumentation beträgt der Maximale Datensatz 244 Bytes pro Paket bei aktiviertem DLE. Diese Funktion ließ ich ausgeschaltet, wodurch ich Maximal 27 Bytes pro Paket versenden kann.

2.1 Datendurchsatz per BLE

2Mbps, steht in Prozessor Doku, lieber nochmal in BLELib nachlesen!
Das Sendeprotokoll von Bluetooth schreibt vor, das ein Datenpaket von leeren Datenpakten eingepackt wird, somit beträgt die Sendezeit pro Datenpaket:

$$\text{Zeit} = \text{Leer} + \text{IFS} + \text{Data} + \text{IFS Leer} = \text{leerespacket(größe)} \;/\; \text{datarate} \tag{2.1}$$

Für mich heißt das:

$$\text{leerGröße} = 2 + 4 + 2 + 3 = 11 \text{ Bytes} == 88 \text{ bits} \tag{2.2}$$

und die Sendezeit für das leere Paket beträgt damit:

$$\text{leerZeit} = 88/2\text{Mpbs} = 44 \text{ Mikro Sekunden.} \tag{2.3}$$

Für ein volles Datenpaket brauche ich:

Voll = 44 + 2*150+ 2+4+2+4+20+3 = 44+300+35 = 379Bytes *8 ==3032 Bits Vollzeit = 3032 / 2 = 1,516 Mikrosekunden

(2.4)

Für ein gesaamtes Datenpaket brauche ich damit mindestens:

Zeit = 44 + 2*150 + 1,516 = 345,516 Mikrosekunden 88/2 + 2*150 + (2 + 4 + 2 + 4 + 20 + 3)*8/2

(2.5)

2.2 Datengröse

Die Daten werden als String versendet, diese werden von Arduino mit einer Null Terminiert. Die Größer der Sensordaten beträgt:

Vorkommastellen (3) + Komma (1) + Dezimalstellen (2) + Terminierung (1) = 7 Char

(2.6)

1 Char entspricht 1 Byte, somit gilt:

9 Sensoren * 7 Byte = 63 Byte 63 Byte / 27 Byte = 2,3 Datenpakete pro alle Sensoren

(2.7)

Somit brauche ich für das Senden aller Sensoren mindestens 3 Characteristics.

2.3 Tatsächliche Übetragunsgeschwindigkeit

3 Hardware

3.1 Arduino Nano 33 BLE

Der Arduino Nano 33 BLE wurde mit einem M4-ARM-Processor und einem 5.0 Bluetooth Modul ausgestattet. Damit ist er BLE-Fähig und liefert einen starken Prozessor für Kommarechnungen.

Der Arduino eignet sich durch seine BLE-Fähigkeit und I2C beziehungsweise SPI-Anschlüsse zur Verwendung mit dem Verwendetem MPU9250. Die geringe Größe und Stromverbrauch sind weitere Pluspunkte.

3.2 MPU9250

Der benutze IMU in diesem Projekt ist ein Multi-Chip mit einem 3-Achsen Gyroskop, 3-Achsen Beschleunigungssensor und einem 3-Achsen Magnetometer. Alle Sensoren wurden noch in der Firma kalibriert. Der Chip bietet einen eingebauten Low-Pass-Filter der eine erste Bearbeitung der Daten vornimmt und so den Hauptprozessor entlastet.
In der folgenden Tablle ist die Empfindlichkeit und Genauigkeit der einzelnen Sensoren notiert, sowie die Übetragunsart und Geschwindigkeit.

Für eine einfache Handhabung benutze ich die I2C Verbindung zwischen Arduino und MPU9250.

3.3 Stromverbrauch

Der Stromverbrauch wurde mit einem Multimeter am Batterieanschluss in verschiedenen Modi gemessen. Die Ergebnisse stehen in der Tabelle:

Der Stromverbrauch lässt so berechnen und erleichter die Korrekte Batterie-Wahl.
Die Verwendeten Formeln: P = U / I
Ah / V = Wh
Wh / W = t –¿ (U*I*t) / U*I = t

So verbraucht der Arduino
0,005 A * 7,4 V = 0,037 W

Die Verschiedenen Batterien-Typen stehen in der folgenden Tabelle:

Um eine Stromversorgung des Arduinos sicher zu stellen benötigt man 2 Knopfzellen des Typs CR2025.Dennoch hat die Knopfzelle CR2025 nicht nur eine bessere Laufzeit sondern ebenfalls weniger Gewicht, weniger Platz und eine bessere Differenz zwischen Cut-Off-Spannung zu angenotener Spannung.

Gegen das umrüsten auf die Knopfbatterie spricht einzig der Umweltschutz. Denn im Gegensatz zu 9-Volt-Batterien gibt es keine Akkus für Knopfzellen.

4 Software

Die Anzeige für Daten erfolgt am Handy, hier habe ich genug Rechenpower nicht nur Nachkommastellen genau zu berechnen, sondern kann ebenfalls visuelle Darstellungen anzeigen. Die Daten werden hierfür über das bereits erklärte BLE an das Handy gesendet.

4.1 Datenübertragung

Die Daten werden wie in Kapitel zur BLE-Datenübetragung beschrieben in 3 verschiedenen Charakteristiken gesendet und empfangen.
Eine Instanz muss hierbei sicherstellen das Daten nicht doppelt gesendet oder empfangen werden.
Dies wird beim Arduino durch Abfrage einer Bibliothekseigenen Funktion sichergestellt, die erst auslöst wenn neue Daten des MPU9250 erzeugt wurden.
Sobald diese Daten gesendet wurden, bekommt das Handy ein Signal und liest daraufhin die neuen Daten.

4.2 Android-App

Zur Erstellung der Android-App wurde AppInventor und die BLE-Extension verwendet.

Beim Start der App wird man aufgefordert Bluetooth und GPS anzuschalten, das GPS ist nach Android-Richtlinien zu aktivieren. Danach kann man nach verschiedenen Geräten scannen und sich mit diesen zu verbinden. Erfolgt die Verbindung mit einem falschen Gerät schließt sich die App.
Nach der Verbindung wird sofort die Übertragung gestartet

Das empfangene Datenpaket muss vor der Verarbeitung in die einzelnen Daten aufgespalten und von String zu mindestens Float-Werten gepaarst werden. Die Split-Funktion von AppInventor sucht nach "—" als Trennzeichen und spaltet hier die Werte. Diese werden in ein Array gespeichert welches später die einzelnen aktuellen Werte ausgeben kann. Es gibt insgesamt 3 Möglichkeiten die Daten anzeigen zu lassen.

4.2.1 Rohdaten

Die gelesenen Daten werden direkt im Textformat auf dem Bildschirm ausgegeben. Der Zeitunterschied zwischen Datenpaketen wird in Millisekunden auf dem Bildschirm angezeigt. Es ist möglich die Daten gleichzeitig aufzuzeichnen.

4.2.2 Graph

Die Daten werden in einem Graph dargestellt, hierzu werden sie zuerst in ein Array geschrieben. Aus diesem Array erzeugt das Programm in einem vorgegebene Bereich die

Datenpunkte, die aufgrund ihrer Masse wie ein Liniendiagramm aussehen.
Neue Daten werden rechts geschrieben, während die alten Daten nach Links aus dem Bildschirm verschwinden.
Der Zeitunterschied zwischen Datenpaketen wird in Millisekunden auf dem Bildschirm angezeigt. Es ist möglich die Daten gleichzeitig aufzuzeichnen.

4.2.3 Aufgezeichnete Daten

Die von den anderen Funktionen aufgezeichneten Funktionen können hier ausgegeben werden. Hierzu benötigt der Schütze den Namen der Datei. Die Dateien werden seit kurzem unter Android in einem App-Eigenem Ordner gespeichert. Diesen muss der Schütze momentan auslesen um den zufälligen Namen der neuen Datei zu kennen.
Die Daten werden als Graph dargestellt. Die Beschleunigungsdaten werden außerdem in Rohform über dem Graphen ausgegeben.

4.3 Arduino Firmware

Das Programm auf dem Arduino Nano 33 Ble stellt zu Beginn eine I^2C Verbindung mit dem MPU9250 her. Diese Verbindung wird über die Wire-Bibiliothek im Fast-Mode (Frequenz:400 000) hergestellt. Das BLE-Objekt zur kommunikation mit dem Boardeigenem Chip wird ebenfalls initialisiert.
Sobald eine BLE-VErbindung steht, fragt der Arduino den Sensor ab. Sollte dieser neue Daten bereitgestellt haben, werden die Daten in einem String verbunden und über die zu den Daten gehörende Characteristic veröffentlicht. Dabei ist die Characteristic so eingestellt das verbundene Geräte eine Nachricht bekommen, sobald der Wert der Characteristic sich verändert.

Der Motion-Processor wird von der Bibliothek verwendet und kann so die Roh-Daten des Sensors mit einem Tief-Pass-Filter vorverarbeiten. Weitere Datenverarbeitung übernimmt der Arduino nicht, um eine möglichst hohe Datenrate zu garantieren.

4.4 Gleichmäßige und ungleichmäßige Beschleunigung

Um die Distanz aus der gemessenen Beschleunigung zu berechnen gibt es 2 mögliche Wege.

Der erste Weg berechnet die Distanz mittels der Formel für gleichmäsig Beschleunigte Bewegungen. Die gemessene Bewegung ist allerdings nicht gleichmäßig, womit dieser Ansatz prinzipeill falsch ist.
Allerdings lieferte diese Formel die besseren Ergebnisse bei Tests.

Der zweite Weg verwendet Integrale, ein Ansatz der auch von verschiedenen anderen Forschern auf diesem Gebiet verfolgt wird. Um von Beschleunigung auf Distanz zu kommen muss man zwei mal integrieren.
Das Problem das schon beim ersten Integral auftritt ist, das Mess-Fehler von sogennanten Rauschen zu Drift wechseln. Dieser Fehler verstärkt sich beim zweiten integrieren weiter und liefert so schnell ungenaue Werte.

Als Zeit wird die Frequenz mit der der Sensor Daten misst genommen. Hierfür wird die Frequenz in Zeitabschnitte——Zeitperioden umgerechnet.

4.4.1 Gleichmäßige Formel

Die Formel für gleichmäßige Beschleunigung berechnete die Distanz in meinen Versuchen mit einer Genauigkeit von +-10 cm auf 30cm Teststrecke. Wurden hierbei ebenfalls negative Beschleunigungen gemessen wirkten sich diese direkt auf die Distanz aus. Dies stellte ein Problem beim abbremsen am Ende der Teststrecke dar. Die Werte sanken wieder auf null.
Aus diesem Grund sind nur positive Werte für diese Funktion zugelassen. Hier der Code-Aussschnitt aus der Berechnung.
Für die Tests wurde dieser Code statt der BLE-Übetragung auf dem Arduino ausgeführt.

```
if (acc > 0) {
t = (freq / 1000); //hz is not time but frequenzy

distance = (distance) /*+ (velocity * t) */ + (acc * (t * t) * 0.5);
velocity = acc * t;
}
```

4.4.2 Integral

Die Berechnung der Distanz über Integrale ist “The way to go” in der Wissenschaft, obwohl die Fehler die sie mit sich bringt bekannt sind. Eben diese Fehler wirkten sich bei meinem Sensor stark aus.
Die Testergebnisse ergaben einen Fehler von +70cm auf einer Strecke von 30cm. Wurde der Sensor zurückbewegt an seinen Startpunkt sank der Wert jedoch wieder auf Null ab. Somit kann man schließen, dass das Ergebniss nur falsch skaliert ist.
Dieser Fehler wurde noch nicht behoben.
Ein klarer Vorteil dieser Rechnung zeigt sich schon beim Test, die Formel funktioniert auch für negative Beschleunigungen. Der Wert sinkt am Ende der Teststrecke nicht auf, sondern bleibt auf seinem hohen Wert.
Folgend der Code-Ausschnitt der Integral-Rechnung:

```
t = (freq / 1000); // hz to time

velocity = t * ((acc + accOld) / 2) + velocity;
accOld = acc;
distance = t * ((velocity + velocityOld) / 2) + distance;
velocityOld = velocity;
```

4.4.3 Tests zu den Messungen

Einer der insgesamt 4 Tests zur Distanz mit der gleichmäßigen Formel:
Einer der insgesamt 5 Tests zur Distanz mit der Integral-Formel:

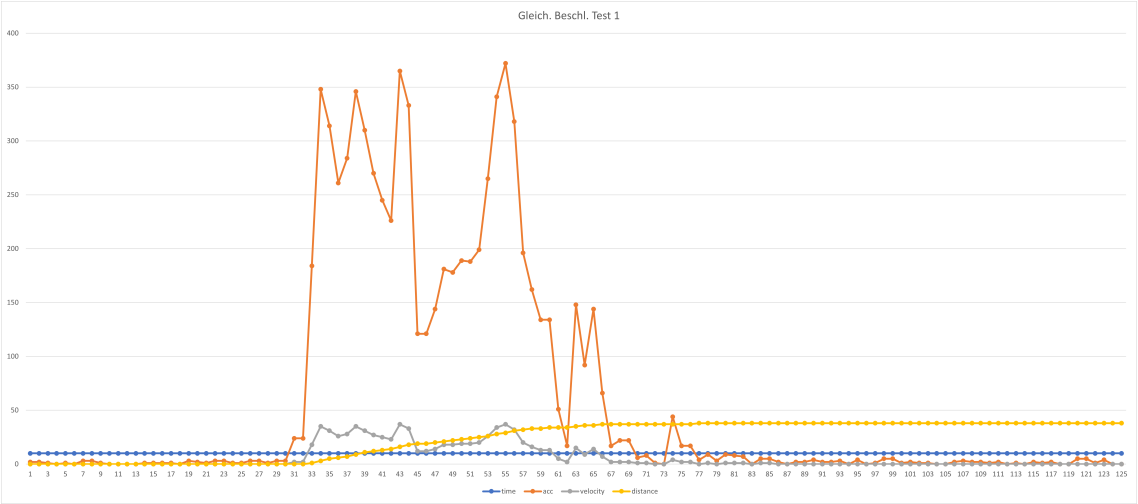

Abbildung 4.1: Gleichmäßige Formel — Test 1

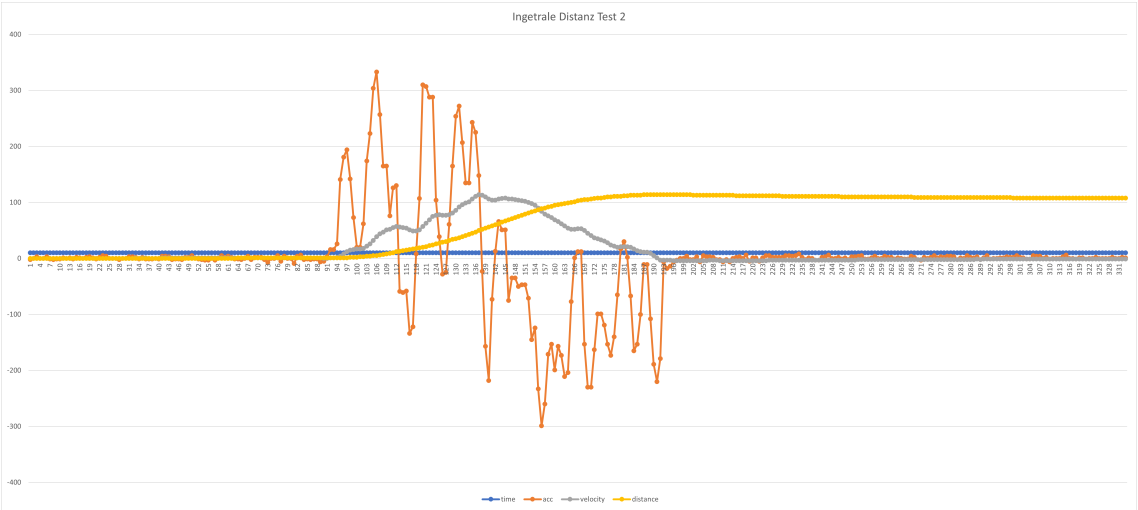

Abbildung 4.2: Integrale Formel — Test 2

5 Machine Learning

5.1 Datenauswahl und Dimensionen

5.2 Modell und Funktion

6 Fazit