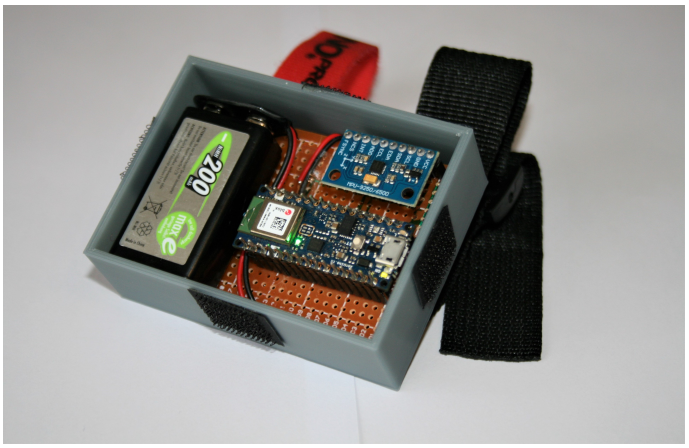


Archer-Tracking

Bewegungstracking für Bogenschützen

Projekt von Antonio Rehwinkel

17. März 2022

Angefertigt am Schiller-Gymnasium Offenburg

Unterstützende Lehrer:
Herr Czernohous, Herr Dierle

und Mitarbeiter der Hochschule Offenburg:
R. Hilterhaus, R. Echle

Kurzfassung:

Mithilfe eines Beschleunigungssensors kann man viele Bewegungen erforschen und vermessen. Mit meinem System sollen ”IMU” dazu eingesetzt werden, Bewegungsabläufe aufzunehmen, zu erkennen und miteinander zu vergleichen.
Die Daten werden über Bluetooth-Low-Energy an ein Handy geschickt, wo sie angesehen und gespeichert werden können. Im Unterschied zum Vorjahr ist es mir nun auch möglich, die Bewegung dreidimensional darzustellen und somit die Auswertung benutzerfreundlich zu gestalten. Die Visualisierung findet an einem PC statt.
Mein System soll die Verwendung von Kameras bei Bewegungserfassungen ersetzen oder erweitern, da diese Systeme nicht nur viel Platz benötigen, sondern sehr teuer sein können und tote Winkel besitzen. Durch einen niedrigen Preis, kleine Formate und eine einfache Handhabung soll die Bewegungsanalyse so in den Alltag rücken. Für mich steht als Anwendungsfall ein wiederholungsträchtiger Sport wie Bogenschießen im Mittelpunkt. Mein System soll es jedem*ermöglichen den Bewegungsablauf ohne Trainer zu trainieren.
Auch das Erkennen von Krankheiten im Bewegungsapparat stellt einen Anwendungsbereich dar, auf den ich mich jedoch nicht konzentrierte.

*Aus Gründen der Lesbarkeit wird das Gendern weggelassen

Inhaltsverzeichnis

1 Einleitung

1.1 Problemlage

Bogenschießen, der Sport der perfekten Wiederholung. Je besser ein Schütze seinen Schussablauf wiederholen kann, desto einfacher ist es für den Trainer Fehler zu finden und das Visier einzustellen. Sehr gute Trefferbilder sind die Folge.
Doch was, wenn der Trainer durch Corona-Regelungen oder Solo-Training nicht dabei sein kann? Der Schütze muss sich selbst kontrollieren, um keine Gewöhnungsfehler einzutrainieren und schlechter zu werden. Ohne Spiegel oder andere Hilfsmittel ist dies jedoch kaum möglich. In genau dieser Situation befand ich mich als Schütze im Frühling 2020.

Bei professionellen Schützen wird der Bewegungsablauf mittels Hochgeschwindigkeitskameras überprüft und selbst kleinste Fehler werden entdeckt. Dabei liegt der Fokus meistens auf dem Lösvorgang. Dieser ist nur eine kleine Bewegung und dauert nur den Bruchteil einer Sekunde. Benötigt wird dazu jedoch neben Kamera, Licht und Platz häufig auch eine dritte Person, die die Datenmenge auswerten kann. All dies macht es kleinen Vereinen unmöglich auch nur an ein ähnliches System zu denken.
Das Ziel meiner Arbeit ist es, ein System zu entwickeln, das es dem Schützen ermöglicht, seinen eigenen Schussaufbau zu sehen und zuverlässig zu bewerten. Damit unterscheidet sich das Ziel meines Systems, ein Full-Body-Tracking, von der Punktgenauen beobachtung, wie die Kamera es im obigen System beabsichtigt.

1.2 Anforderungen

Der Schütze soll seinen Schussablauf auf Grund der Daten und Visualisierung auswerten und bewerten können. Da hierbei der gesamte Bewegungsablauf beobachtet wird, benötigt man keine zentrierte, sehr hohe Genauigkeit wie bei der Beobachtung des Lösens des Schützen. Somit muss die Genauigkeit der Hochgeschwindigkeitskameras nicht erreicht werden.
Die Anforderungen an mein Projekt sind damit, dass eine kostengünstige Alternative geschaffen wird, die wenig Platz benötigt und schnelle Ergebnisse liefert. Dabei müssen diese Ergebnisse genau und für jeden verständlich sein.
Des Weiteren darf der Schütze auf keinen Fall gestört werden. Dies wirkte sich bei meiner Idee vor allem auf die Größe, das Gewicht und die Datenübertragung aus.

Durch Größe und Preis soll es selbst kleinen Vereinen möglich sein, den Bogenschützen zu vermessen und gezielte Hilfe auch ohne Trainer zu bieten. Auch Privatpersonen könnten sich so ein System sinnvoll zulegen.

Entwickelt und gebaut wurde dieses Projekt im Schuljahr 2020/21 von Zuhause aus und wurde im Schuljahr 2021/22 weitergeführt.

2 Vorgehensweise

2.1 Möglichkeiten

Auf meiner Suche hatte ich die Idee, einen Ultraschallsensor am Schützen zu befestigen. Der Ultraschallsensor hätte am Boden befestigt werden können, um die Höhe der einzelnen Punkte des Schützen zu bestimmen. Ebenfalls wäre eine Kabelführung für schnellere Datenübertragung

und damit ein günstigerer Preis möglich gewesen. Dieses statische System hätte allerdings auf einen sich bewegenden Schützen eingestellt werden müssen. Ein großer Widerspruch, lösbar nur durch weitere Technik, die zu kaufen gewesen wäre. Auch die Verwendung einer günstigeren Kamera wäre möglich gewesen. Dabei vereint man allerdings alle Nachteile, die das Kamera-Tracking hat. Man braucht viel Platz und trotz sehr günstiger Kameras treibt man die Kosten in die Höhe. Meine finale Idee war, eine IMU [1] und einen BLE-Chip[2] zu kombinieren.
Der IMU sollte mir die Beschleunigsdaten in alle Richtungen liefern, aus dehnen ich die Distanz berechnen wollte. Als weitere Möglichkeit bot sich an, mittels des IMU die Orientierung des Sensors zu berechnen.

2.2 Materialsuche

Es galt nun, zu den genannten Kriterien die passende Hardware zu finden. Um die Bewegungen des Schützen nachzuverfolgen, muss ich wissen, wo die einzelnen wichtigen Punkte des Aufbaus sind. Dies betrifft beide Arme und die Schultern. Da sich die Arme viel bewegen, schien es mir möglich, mithilfe eines günstigen Beschleunigungssensors die Änderungen festzustellen. Bei hoher Genauigkeit könnte dieser vielleicht sogar die Bewegungen der Schultern messen. Um den Schützen nicht zu behindern, ist eine kabellose Verbindung von Vorteil.

[1] Näheres hierzu in *Kapitel 3.2 MPU9250*
[2] Näheres hierzu in *Kapitel 4 Bluetooth-Low-Energy*

3 Hardware

3.1 Arduino Nano 33 BLE

Der Arduino Nano 33 BLE ist ein Prozessor aus dem Hause Arduino der Nano Reihe. Was diesen von der normalen Nano-Reihe unterscheidet ist der BLE-Chip NINA-b3(nRF52840) auf seinem Rücken. Dieser Chip ermöglicht es dem Arduino über Bluetooth 5.0, auch Bluetooth-Low-Energy genannt, mit allen anderen Bluetooth-Geräten ab der Bluetooth Version 4.0 kabellos zu kommunizieren.

Eigenschaft	Daten
Memory	1MB Flash | 256 KB SRAM
Interfaces	I^2C, SPI, …
Volt	Input: 4,5 - 21 V | Output: 3,3 V

Der Arduino ist aufgrund seines BLE-Chips, der I²C-Verbindungsmöglichkeit und der Output-Volt Zahl von 3,3 Volt der richtige Prozessor für dieses Projekt. Auch die kleinen Maße und das geringe Gewicht bringen ihm nur Pluspunkte ein.

3.2 MPU9250

Für eine genaue Datenlage sorgt in meinem Projekt der Multi-Chip MPU9250. Dieser ist mit einem 3-Achsen Beschleunigungssensor, einem 3-Achsen Gyroskop Sensor und einem 3-Achsen Magnetometer ausgerüstet. Damit bietet er neun Freihtsgrade (9 Degrees of Freedom) und gehört zu der Gruppe der Inertialen Messeinheiten[1]. Alle Sensoren erhielten eine Kalibrierung innerhalb der Firma und können einen Selbsttest bei Benutzung vollziehen. Der Sensor benötigt nur 2,4 bis 3,6 Volt während des Betriebs. In der Tabelle sind die Datenpins und die Genauigkeit der Sensoren notiert.

Sensoren	Datenübertragung	Empfindlichkeit
Gyroskop	3 * 16bit ADCs	$\pm 250/\,\text{sec}$, $\pm 500\,\text{sec}$, $\pm 1000/\,\text{sec}$, $\pm 2000/\,\text{sec}$
Beschleunigungssensor	3 * 16bit ADCs	$\pm 2g$, $\pm 4g$, $\pm 8g$, $\pm 16g$
Magnetometer	3 * 16bit ADCs	full-scale range of $\pm 4800\,\text{mT}\,\text{mT}$
Übertragung	I^2C, SPI, …	

Die Daten werden über den I²C-Bus vom Arduino abgefragt. Die Abtastrate beträgt hierbei mögliche 400kHz. Dabei werden alle Sensoren abgefragt und die Daten versendet. Durch die geringe Größe des Chips (15mm*25mm), dem geringem Energieverbrauch (3,5 mA wenn alle Sensoren ausgelesen werden), der hohen Genauigkeit und der Geschwindigkeit der Datenübertragung ist dieser Sensor perfekt für dieses Projekt. Der verbaute DMP (Digital Motion Processor) wird ebenfalls verwendet und filtert die Daten mit einem Low-Pass-Filter.

4 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 einem höheren Datendurchsatz aus.

[1] IMU bzw. Inertial Measurement Unit

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

Die Server bieten *Services* an, die mit *Characteristics* befüllt sind. So bietet mein Arduino den Service *MPU9250* an, mit dem Characteristis *Accl*, *Gyro* und *Mag*.
Der Nachteil dieser Verteilung der einzelnen Daten besteht hierbei in der Zeit, die für die Abfrage gebraucht wird. Jede *Characteristic* muss einzeln abgefragt werden, hierbei kann ein Großteil des Datendurchsatzes des MPU9250 verloren gehen.

Laut Dokumentation beträgt der maximale Datensatz von BLE 244 Bytes pro Paket bei aktiviertem DLE. Diese Funktion ließ ich ausgeschaltet, wodurch ich maximal 27 Bytes pro Paket versenden kann. Dieses Problem erklärt ebenfalls weshalb die Sensor-Daten auf verschiedene *Characteristics* aufgeteilt werden. Alle Daten passen nicht in ein einzelnes zu versendendes Paket.

4.1 Datengröße

Die Daten werden als String versendet, diese werden von Arduino mit einer Null terminiert.

Die Größe der Sensordaten beträgt:
Vorkommastellen (3) + Komma (1) + Dezimalstellen (2) + Terminierung (1) = 7 Char

1 Char entspricht 1 Byte, somit gilt:
*9 Sensoren * 7 Byte = 63 Byte*
63 Byte / 27 Byte = 2,3 Datenpakete pro alle Sensoren

Somit brauche ich für das Senden aller Sensoren mindestens drei *Characteristics*.

4.2 Datendurchsatz per BLE

Das Sendeprotokoll von Bluetooth schreibt vor, dass ein Datenpaket von leeren Datenpaketen eingepackt wird. Ebenso ist eine kurze Wartezeit vorgeschrieben. Diese beträgt 150 Mikrosekunden und wird abgekürzt mit *IFS*. Der Arduino Nano unterstützt *2 Mbps* bei der BLE-Übertragung, dies ist also die Datenrate. Des Weiteren wird nicht auf eine Antwort des *Clients* gewartet, was die Übertragungsgeschwindigkeit weiter erhöht.

Somit beträgt die optimale Sendezeit pro Datenpaket:
Zeit = Sendedauer[Leer] + IFS + Sendedauer[Voll] + IFS
Sendedauer[Leer] = LeeresPacket / Datarate

Für mich heißt das:
LeeresPacket = 2 + 4 + 2 + 3 = 11 Bytes = 88 bit

und die Sendezeit für das leere Paket beträgt damit:
Sendedauer[Leer] = 88 bit / 2 Mbps = 44 Mikrosekunden

Für ein volles Datenpaket brauche ich:
2+4+2+4+27+3 = 42 Byte (8 Umrechnung in Bit) = 336 bit*

Sendedauer[Voll] = 336 bit / 2 Mbps = 168 Mikrosekunden

Für ein gesamtes Datenpaket brauche ich somit mindestens:
Zeit = 44 + 150 + 168 + 150 = 512 Mikrosekunden

beziehungsweise 0,512 Millisekunden. Die maximal erreichbare Datenübetragungs-Frequenz liegt bei 1,95 kHz.

4.3 Tatsächliche Übetragunsgeschwindigkeit

Die ausgerechnete Datenrate kann in der Praxis kaum erreicht werden, weshalb ein Test zur tatsächlichen Datenrate Pflicht ist. Für den Test wurde der auch später in der Praxis verwendete Code verwendet. Die gemessene Datenrate entspricht 50 Hz, was der eingestellten Aktualisierungsrate des MPU9250 entspricht.

5 Analyse und Visualisierung

5.1 Analyse

Da der Schussablauf eines Bogenschützen viele Stationen mit verschiedenen Bewegungen beinhaltet, fällt es häufig sogar den Trainern schwer, zwischen einem technisch guten oder schlechten Schuss zu unterscheiden.
Somit galt es einen Punkt zu finden, bei dem Fehler auffällig sind, so dass ich die Daten auch in der Praxis nachvollziehen kann.

Eine interessante Bewegung stellt vor allem der Zugarm des Schützen dar. Der Auszug verläuft nahezu linear, der häufigere Fehler an dieser Stelle versteckt sich in der Höhe des Zugarms. Um diese zu messen, muss man die Erdanziehungskraft der Z-Achse herausrechnen.
Möglich ist ebenso, die Neigung des Armes über die Euler-Winkel zu berechnen. Sollte der Arm zu hoch sein, wird sich die Ausrichtung des Sensors stark verändern, so wie die zurückgelegte Distanz in Z-Richtung.

Um den vollen Bewegungsablauf eines Schützen zu tracken, wird an jedem Körperteil ein Sensor benötigt. Dies ermöglicht nicht nur das gegenseitige Überprüfen auf Sinnhaftigkeit der Orientierung (menschliches Skelett hat beschränkte Bewegungsfreiräume), sondern auch die Klassifizierung der einzelnen Schuss-Abläufe könnte, auf Grund von mehr Daten, genauer ablaufen.

Der MPU9250 bietet 9 Freiheitsgrade (Degrees Of Freedom - DOF). Aus welchen diese bestehen, wird im Kapitel zum MPU9250 erläutert. Diese Freiheitsgrade lassen es zu, die Distanz die der Sensor sich in alle Richtungen bewegt, zumindest in der Theorie, zu berechnen. Die Orientierung des Sensors und damit die Orientierung des Körpers an der er befestigt ist, kann man genau messen und berechnen.

So ist eine Darstellung als 3D-Modell möglich, an der der Schütze seine Fehler sieht und Unterschiede zu vorangegangen Schüssen hervorgehoben werden.
Denkbar wäre eine Klassifizierung der einzelnen Schüsse, um, wie in einem Videospiel, die Genauigkeit der Wiederholung in Prozent anzugeben. Interessant wird es, sobald mehrere Schützen Datensätze ihrer Schüsse vergleichen. Unterschiede werden ersichtlich, genau wie Gemeinsamkeiten.

Dieses Kapitel beschäftigt sich im Folgenden mit der dreidimensionalen Visualisierung.

5.2 Integrale

Die Distanz, berechenbar aus den Werten des Beschleunigungssensors, kann uns die Differenz zwischen einem Startpunkt und dem jetzigen Punkt des Sensors berechnen. Durch die drei Achsen meines Beschleunigungssensors ginge dies sogar in alle Richtungen. Damit könnte man ein Modell erzeugen, das den betreffenden Punkt in die einzelnen Richtungen verschiebt, bis der finale neue Standort des Sensors erreicht wird.

Um die Distanz aus der gemessenen Beschleunigung zu berechnen, fand ich zwei verschiedene Formeln. Die wohl bekannteste Umrechnung benutzt Integrale, die zweite Formel ist die der gleichmäßigen Beschleunigung.

Die Integration wird von allen mir bekannten Forschungen verwendet. Man muss eine Doppelintegration ausführen um von Beschleunigung auf Distanz zu kommen, hierbei verwandelt sich das Rauschen des Sensors in Drift, so entsteht einen exponentiell steigender Fehler.
Die gleichmäßige Formel kann, im Gegensatz zum Integral, nur positive Beschleunigungen verwerten, hat in den Tests allerdings deutlich genauere Werte und einen geringeren Fehler bei Stillstand aufgezeigt.

Als Zeit wird die Frequenz, mit der der Sensor Daten misst, genommen. Hierfür wird die Frequenz in Zeitabschnitte umgerechnet, mit der die Formeln letzendlich arbeiten. Der Testaufbau und die Durchführung ist im *7.1 Distanz-Testaufbau* zu sehen. Die Formel entspricht hierbei der Umrechnung in mHz, da ein Skalierungsfehler bei der Verwendung der normalen *1/f*-Formel entsteht.

5.2.1 Gleichmäßige Formel

Die angepasste Formel für gleichmäßige Beschleunigung berechnete die Distanz in meinen Versuchen mit einer Genauigkeit von ± 8 cm auf 30cm Teststrecke.
Die physikalische Formel für Bewegungen mit einer Anfangsgeschwindigkeit lautet $s + v * t + a * t^2 * 0.5$, allerdings wurden mit dem leicht verändertem Weg-Zeit-Gesetz $s + a * t^2 * 0.5$ bessere Testergebnisse erzielt.
Da die negative Beschleunigung den Wert der Distanz wieder nullierte, wurden zu dieser Formel nur positive Distanzmessungen zugelassen.
Hier der Code-Ausschnitt:

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

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

Für die Tests wurde dieser Code statt der BLE-Übetragung auf dem Arduino ausgeführt.

5.2.2 Integral

Die Berechnung der Distanz über Integrale ist der Standard in der Wissenschaft. Beim integrieren meiner Sensordaten ist zu beachten, das beim mathematischen Integrieren die zu berechnenden Werte gegen den Limes laufen können. Da ich nur begrenzete Datenmengen zur Verfügung habe, ist dies nicht möglich, wodurch allein das Integrieren Fehler erzeugt. Ebenfalls wird das Rauschen meines Sensors nicht beachtet und aufaddiert. Da ich zwei mal integrieren muss, ist dieser Fehler in der Distanz exponentiell.

Die Testergebnisse ergaben einen Fehler von ±15cm auf einer Strecke von 30cm. Ebenso war das Ergebnis sofort in Zentimetern, statt wie erwartet in Metern. 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 ab, 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;
```

5.3 Orientierungen

Um die Orientierung des Sensors zu berechnen, geht man auf die Grundaussagen der Sensoren zurück. Der Beschleunigungssensor gibt die G-Kräfte an, daraus lassen sich Rückschlüsse über die Orientierung der Pitch- und Roll-Achse ziehen. Das Gyroskop gibt Drehmoment-Werte zurück, worüber sich der Beschleunigungssensor überprüfen lässt und eine Drehung um die Yaw-Achse messbar macht. Das Magnetometer setzt schließlich alle Werte in Relation zum Erdmagnetfeld und kann sowohl das Gyroskop als auch den Beschleunigungssensor überprüfen. Diese Überprüfung findet über Filter statt, in meinem Projekt verwende ich hierfür den AHRS-Filter[1]. Der AHRS-Filter fusioniert die Daten in eine verwendbar Orientierung.
Für die Darstellung der Orientierung im dreidimensionalen Raum gibt es mehrere Ansätze, die zwei bekanntesten sind wohl die Euler-Winkel und die Quaternionen. Die Vor- und Nachteile werden im Folgenden erläutert.

5.3.1 Euler-Winkel

Bei Euler-Winkeln wird das Objekt um die einzelnen Achsen in einer festgelegten Reihenfolge gedreht. Sollten hierbei nach einer Drehung zwei Achsen aufeinander liegen, entsteht der sogenannte *Gimbal-Lock* und die Darstellung der Bewegung findet fehlerhaft statt. Für reele Systeme, wie Gimbal an einer Drohne, ist dieser Fehler vernachlässigbar, da er nur selten vorkommt und die Euler-Winkel den tatsächlich benötigten Drehwinkeln der Motoren entsprechen. Da ich jedoch keine Motoren ansteuern muss, steht vor allem der Fehler im Mittelpunkt und ist der Grund dafür, das ich nicht die Euler-Winkel verwende.

[1] Der AHRS-Filter ist ein erweiterter Kalman-Filter. Er passt seine Fehlerwerte über Zeit dem Drift der Sensorik an und bietet so auch auf lange Zeit gute Ergebnisse.

5.3.2 Quaternionen

Um die Position eines Körpers im dreidimensionalen Raum darzustellen, reichen drei Koordinaten, die Möglichkeit der Orientierung verlangt jedoch nach mindestens einer vierten Koordinate. Quaternionen beschreiben ein solches vierdimensionales System. Die Formel der Quaternionen sieht eine Reelle und drei imaginäre Zahlen vor. Die imaginären Zahlen im Verhältnis zueinander stehen und ihrere Werte, vereinfach gesagt, skaliert werden können, kann ein "Richtungsvektor" berechnet werden. Dieser kann in alle Richtungen zeigen. Das Objekt wird an diesem Vektor orientiert. Ein Überprüfen der Rohdaten fällt durch die vierte Dimension für mich aus. Allerdings schaffen es die Quaternionen durch diese weitere Dimension jegliche Orientierung im Raum darzustellen ohne etwas ähnliches wie einen *Gimbal-Lock* zu erzeugen. Die Vollständige Erklärung der Quaternionen sprengt momentan leider meinen Wissenstand. Die einfache Eingabe meiner Daten in fast alle bekannte 3D-Visualisierungsprogramme und der fehlende *Gimbal-Lock* sind der Grund für mich, zukünftig die Quaternionen zu verwenden.

6 Software

Die Anzeige der Daten erfolgt am Handy, hier habe ich genug Rechenpower, nicht nur Nachkommastellen genau zu berechnen, sondern auch um visuelle Darstellungen zu erzeugen. Die Daten werden hierfür über das bereits erklärte BLE an das Handy gesendet. Somit brauchte ich sowohl eine Android-App als auch ein Programm für den Arduino.
Momentan werden ebenfalls Computer-Programme wie Matlab und OpenSim zur Visualisierung benötigt.

6.1 3D-Visualisierung

Die Darstellung erfolgt momentan über Matlab und OpenSim. In Matlab werden die IMU-Daten fusioniert, die Orientierung in Quaternionen berechnet und in das korrekte Daten-Format geschrieben. OpenSim schreibt den Sensor-Daten dann den angegeben Knochen eines Skellets zu und orientiert diese anhand der Daten neu.

6.1.1 Matlab und OpenSim

Matlab ist ein kostenpflichtiges Programm und wurde mir von meiner Schule im Rahmen der MakerSpace-AG zur Verfügung gestellt. Ich benutze es, um einerseits Live-Daten des Sensors zu verwerten als auch um die gespeicherten Handy-Daten in das favorisierte Datei-Format für OpenSim umzuformen und damit für spätere Benutzung vorzubereiten. Die Verwertung und Umformung findet mittels eigenem Code statt. Matlab stellt mir bei der Umformung eine einfache Verwendung des AHRS-Filters bereit. Der AHRS-Filter berechnet hierbei die Orientierung in Quaternionen.
OpenSim ist eine OpenSource-Software, die verwendet wird für biomechanische Modellierung, Simulation und Analyse. Die Sensorwerte werden für OpenSim in das Datenformat *.sto* umgeschrieben. OpenSim ließt aus dieser Datei die manuell zugewiesenen Messpunkte wie Ober-/Unterarm aus und weist die folgenden Orientierungen mit zeitlicher Kennnummer den verschiedenen Körperteilen zu. Mithilfe der inversiven Kinematik berechnet OpenSim den aufgenommenen Bewegungsablauf. Neben den Sensoren zur Bewegungsaufnahme empfiehlt es sich, einen Sensor an der Hüfte zu platzieren. Dieser dient durch wenig Bewegung in jegliche Richtung als Null-Punkt und kann als Relationspunkt für die anderen Sensoren verwendet werden.

Ausflug: Inversive Kinematik

Inversive Kinematik löst das Problem, einen bekannten End-Punkt zu haben, jedoch die richtige Bewegung zu diesem End-Punkt nicht zu kennen. Da die berechneten Orientierungen solche End-Punkte darstellen, verwendet OpenSim Inversive Kinematik, um die Animation zwischen den End-Punkten zu erzeugen. Da dies vollautomatisch passiert, kenne ich auch hier nur die grobe Umsetzung.

6.2 Datenübertragung

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

6.3 Arduino Firmware

Das selbst geschriebene Programm auf dem Arduino Nano 33 BLE stellt zu Beginn eine I^2C Verbindung mit dem MPU9250 her und startet das BLE-Modul.
Sobald ein Gerät sich mit dem Arduino verbindet, beginnt dieser mit der Datenübertragung.
Hierbei wurden die Filter vom Digital-Motion-Processor des MPU9250 schon mit einem Low-Pass-Filter verarbeitet.

6.4 Android-App

Zur Erstellung der Android-App wurde AppInventor und die BLE-Extension verwendet.
Programmiert wurde das Programm von mir. Ideen und Anregungen wurden in verschiedensten Foren gefunden.

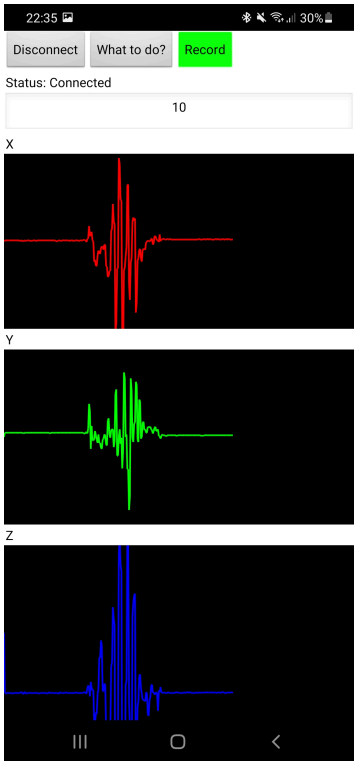
Beim Start der App wird man aufgefordert Bluetooth und GPS anzuschalten, das GPS ist nach Android-Richtlinien zu aktivieren. Danach kann man nach verschiedenen Geräten scannen und sich mit diesen verbinden. Erfolgt die Verbindung mit einem falschen Gerät, schließt sich die App. Nach der Verbindung wird sofort die Übertragung gestartet.
Das empfangene Datenpaket muss vor der Verarbeitung in die einzelnen Daten aufgespalten und von String zu Float-Werten, besser Double-Werten, geparst werden. Diese werden in ein Array gespeichert, welches später die einzelnen aktuellen Werte ausgeben kann.

Es gibt insgesamt drei Möglichkeiten die Daten anzeigen zu lassen.

6.4.1 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.

6.4.2 Graph

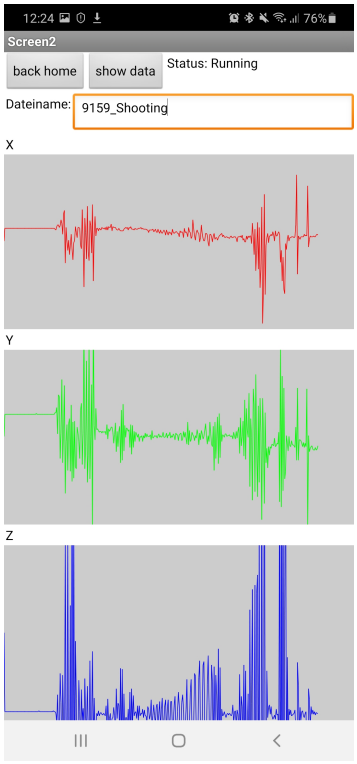


Die Beschleunigungs-Daten werden in einem Graph dargestellt, hierzu werden sie zuerst in ein Array geschrieben. Aus diesem Array erzeugt das Programm in einem vorgegebenen Bereich die Datenpunkte, die aufgrund ihrer Masse wie ein Liniendiagramm aussehen.
Neue Daten werden rechts geschrieben, während 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.

6.4.3 Animation

Hier wird die Distanz aus den Beschleunigungsdaten berechnet und mithilfe eines Punktes visualisiert. Benutzt werden hierfür die X- und Y-Achse. Die Rechnung ist in Kapitel *5.2.1 Gleichmäßige Formel* beschrieben.

6.4.4 Aufgezeichnete Daten



Die von den anderen Funktionen aufgezeichneten Graphen können hier ausgegeben werden. Hierzu benötigt der Schütze den Namen der Datei. Die Dateien werden seit kurzem unter Android in einem App-eigenen Ordner gespeichert. Von dort muss die Datei in den öffentlichen Gerätespeicher verschoben werden. Den Datei-Namen 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.

7 Benötigte Leistung

Da dieses Jahr fest steht, welche Daten gesendet werden können, lohnt es sich nun auch, die benötigte Leistung zu ermitteln. Diese wurde mit einem Multimeter am Batterieanschluss in verschiedenen Modi gemessen. Die Ergebnisse stehen in der Tabelle:

Modul	An/Aus	Verbrauch(in mA)
BLE	aus	5
BLE	an	10.2
BLE	an und verbunden	13
MPU9250	an	5

Der Stromverbrauch lässt sich so berechnen und erleichtert die korrekte Batterie-Wahl. Die verwendeten Formeln:

$P = U * I$
Für die Einheiten gilt nach Multiplikation mit der Zeit:
$Ah * V = Wh$
$Wh / W = h \rightarrow (U*I*t) / U*I = t$

(7.1)

So hat der Arduino eine Leistungsaufnahme von:

$0{,}005\ A * 7{,}4\ V = 0{,}037\ W$

(7.2)

Die verschiedenen Batterien-Typen stehen in der folgenden Tabelle:

Typ	Laufzeit(in Stunden)	Gewicht	Cut-Off-Spannung	Differenz[1]
9V	40	50g	7.2V	$9 - 7.2 = 2,8$
CR2025	12	2,5g	2V	$3 - 2 = 1$
2 * CR2025	48,6	5g	2V	$6 - 2 = 4$

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, braucht weniger Platz und liefert eine bessere Differenz zwischen Cut-Off-Spannung zu angebotener Spannung.
Gegen das Umrüsten auf die Knopfbatterie spricht einzig der Umweltschutz. Denn im Gegensatz zu 9-Volt-Batterien gibt es keine Akkus für Knopfzellen. Beruhigend ist die geringe Leistungsaufnahme, was größte Argument für die genutzte Hard- und Software war.

8 Tests

8.1 Distanz-Testaufbau

Der Sensor wurde auf einem Breadboard mit einem Arduino Uno verbunden und übertrug die Werte via USB-Kabel an den Seriellen Monitor der Arduino IDE. Um eine mögliche Schieflage auf dem Breadboard abzufedern, wurden aus 200 Messungen der Durchschnitt berechnet und von den Werten vor der Verarbeitung subtrahiert. So liegt der Sensor mathematisch flach auf.

8.1.1 Distanzmessungen

Insgesamt wurden 10 Tests pro Formel gemacht, die Ergebnisse sind nachfolgend zu sehen.

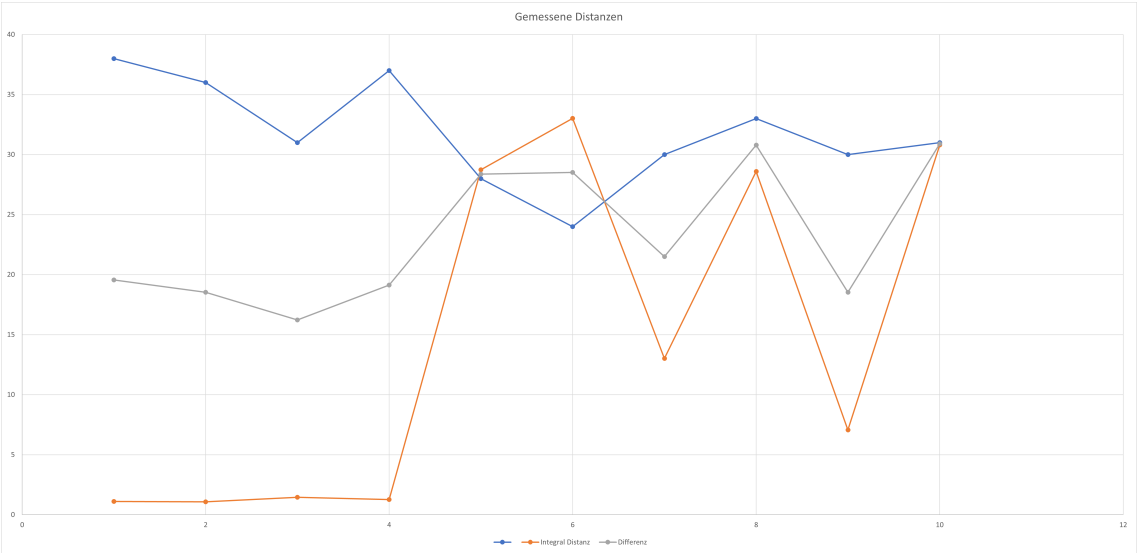


Abbildung 8.1: Distanzen | Alle Tests

[1] zwischen Cut-Off-Spannung und angebotener Spannung

8.1.2 Fehler

Die zwei weiteren Graphen zeigen die Fehler bei 10 sekündigem Stillstand. Der exponentielle Drift bei der Integralen Formel ist typisch.

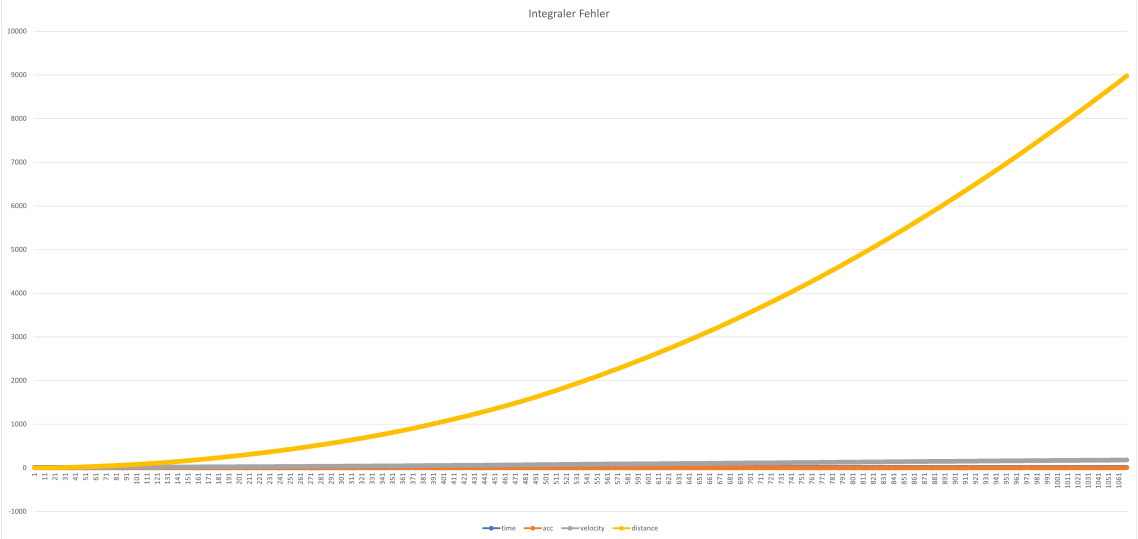


Abbildung 8.2: Integrale Formel | Drift

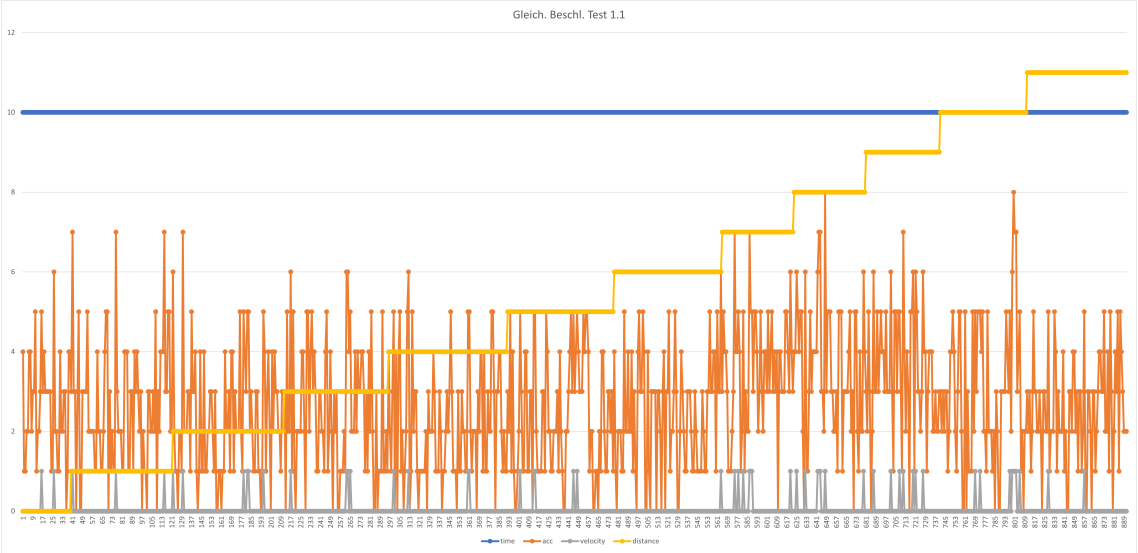


Abbildung 8.3: Gleichförmige Beschl. Formel | Drift

Auch die Fehler in den Distanzmessungen sind zu groß, als dass eine zuverlässige, über auch nur Minuten gehende Distanzberechnung für mein Projektziel sinnvoll wäre.

9 Ergebnissdiskussion

Die Genauigkeit der gemessenen Distanz des MPU9250 hat zu hohe Fehlerwerte, als dass sie momentan ohne Vorverarbeitung im dreidimensionalen Tracking verwendet werden könnten. Allerdings sind momentan keine Filter oder Fusionen der Sensoren des MPU9250 implementiert, weshalb eine Steigerung der Messgenauigkeit noch möglich ist. Es gibt Forschungen auf diesem Gebiet, die akzeptable Ergebnisse erreichten. Im einher zu diesen Erfolgen kamen jedoch immer Einschränkungen die Benutzung betreffend, die mich schließlich von der Idee abbrachten,

in eine ähnliche Richtung, im Rahmen dieses Projekts, zu forschen.
Die Orientierung hingegen kann genau berechnet werden, was die Visualisierung ermöglicht.
Die Berechnung der Distanz, wie anfangs geplant, ist nicht ohne erheblichen Aufwand möglich. Somit fällt die Idee darüber die Gliedmaßen zu tracken aus. Stattdessen wird durch Drittsoftware die Orientierung berechnet und verwendet.
Hierbei trägt die Wahl der Sportart der Funktionnstüchtigkeit des Projekts bei. Beim Bogenschießen findet wenig Bewegung statt, die nicht über eine Neu-Orientierung dargestellt werden kann, weshalb die Berechnung der Distanz für dieses Projekt nicht nötig ist.

10 Fazit

Das fertige Modell ist 80 * 60 * 27 mm groß und wiegt mit den neuen Batterietypen ungefähr 30 Gramm. Die Größe und das Gewicht erfüllen die erste Anforderung. Man kann die Box überall am Schützen befestigen ohne das dieser durch zusätzliches Gewicht stark beeinflusst würde.
Die zweite Anforderung, ein geringerer Preis als Kamera, wurde erfüllt, die Kosten belaufen sich auf 25 € pro Sensor und 150 € für ein Oberkörper-Tracking, bestehend aus je zwei Ober- und Unterarm sowie Brust- und Hüftsensor. Die letze Anforderung, die leichte Handhabung nähert sich an, es ist mir nun möglich, Distanzen und Winkel in eine bestimmte Richtung zu berechnen und darzustellen. Die Auswertung der Graphen ist nach wie vor möglich.

Die BLE-Übertragung funktioniert gut, das Problem der überlaufenden Characteristics wurde behoben. Somit kann man alle Daten an das Smartphone senden und empfangen. Die Datenpackete bieten jedoch noch Platz für Verbesserung.

Die Darstellungen der Daten funktioniert weiterhin, seit Android 11 müssen die gespeicherten Schüsse manuell in den Gerätespeicher verschoben werden. Android richtete eine Datenschutzmaßnahme ein, nach der Apps nicht mehr in den Gerätespeicher, sondern in einen appeigenen Unterordner schreiben müssen. Das Lesen stellt jedoch kein Problem dar.

Mit den neuen mathematischen Formeln stellt sich nun ein weiteres Problem, die kabellose Übertragung. Je mehr Nachkommastellen übertragen werden, desto genauer erfolgt die Berechnung der Winkel und vor allem der Distanz. Der Multi-Chip liefert 16 Nachkommastellen, was pro Achse eine Characteristic benötigen würde, vorraussetzung, das die DLE-Funktion nicht aktiviert wurde.
Momentan sind die berechneten Ergebnisse genau genug mit nur 2 Nachkommastellen, weshalb ich dieses Problem vorerst nicht weiter bearbeitete.

Die Distanz stellte sich im Lauf der diesjährigen Arbeit als interessant, aber nicht zielführend für die Animation aus. Ersetzt wurde diese durch die funktionierende Orientierungsberechnung. Die Berechnung und Visualisierung findet momentan mithilfe von Dritt-Software statt. Ein Umstieg von AppInventor auf tatsächlichen Code wird dringend benötigt, da AppInventor weder 3D-Animationen noch die von mir benötigen Berechnungen anbieten oder durchführen kann.
Die Darstellung kann momenten nicht live angesehen werden, ein Feature, an dem noch gearbeitet wird.

Die berechneten Winkel und Distanzen haben einen Fehler-Wert, der im Vergleich zu momentan benutzten Hochgeschwindigkeitskameras enorm ist, jedoch hat die angesprochene Kundschaft andere Wünsche.

Geplant ist die vollständige Darstellung eines Schützen in einer Android-App und die Klassifizierung der Schüsse via Machine-Learning. Hierzu steht ein erstes Modell, die Datenbasis ist momentan allerdings zu gering, um belastbare Ergebnisse zu berechnen.

11 Danksagung

Danke an

- Herr Czernohous (Korrekturlesen der Arbeit und technische Hilfe, Kauf der Materialien) [Schiller Gymnasium Offenburg]
- Herr Dierle (Hilfe bei physikalischen Zusammenhängen und Formeln, Korrekturlesen der Arbeit) [Schiller Gymnasium Offenburg]
- R. Echle und R. Hilterhaus (für Arbeitsideen und Hilfestellung) [Hochschule Offenburg]

12 Quellen

- Zuletzt besucht am 12.1.2022 | *HowToMechatronics* https://howtomechatronics.com/how-it-works/electrical-engineering/mems-accelerometer-gy
- Zuletzt besucht am 12.1.2022 | *NovelBits* https://www.novelbits.io/bluetooth-5-speed-maximum-throughput/
- Zuletzt besucht am 12.1.2022 | *Journal of Mechanical Engineering and Sciences (JMES)* http://jmes.ump.edu.my/images/Volume_6/4_Ahmad%20et%20al.pdf
- Zuletzt besucht am 12.1.2022 | *BatteryEquivalents* https://www.batteryequivalents.com/lithium-cr2025-dl2025-e-cr2025-sb-t14-5003lc-br2025 html
- Zuletzt besucht am 12.1.2022 | *akkuline.de* https://www.akkuline.de/test/varta-high-energy-9v-batterie-test-messung
- Zuletzt besucht am 12.1.2022 | *Youtube | Google TechTalks* https://www.youtube.com/watch?v=C7JQ7Rpwn2k
- Zuletzt besucht am 2.3.2022 | *Didaktik Mathematik HU Berlin | Quaternionen: von Hamilton, Basketbällen und anderen Katastrophen* http://didaktik.mathematik.hu-berlin.de/files/2011_quaternionen.pdf