Schützenvermessung als DIY-Version

Ein Projekt von Antonio Rehwinkel (16 Jahre)

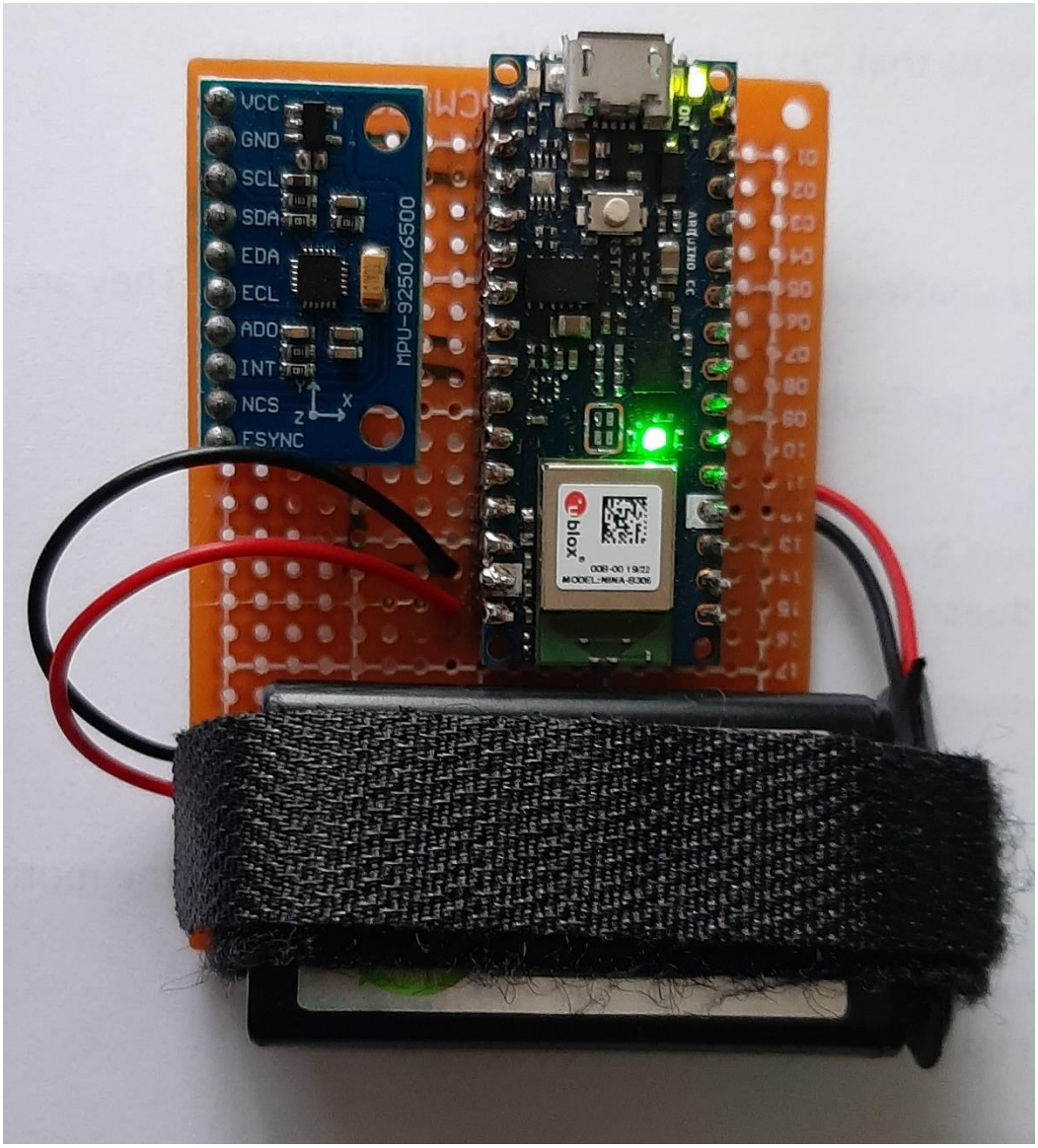
Kurzfassung

Um erfolgreich zu sein müssen Bogenschützen ihren Aufbau roboterhaft wiederholen. Um sogar kleinste Fehler in der Körperhaltung zu finden, kann es sinnvoll sein die Bewegungen des Schützen nachzuverfolgen, zu „tracken“.

Das Ziel meines Projekts ist, diese Trainingsmethode für jeden zugänglich zu machen. Mein Augenmerk liegt dabei auf Größe, Preis und Genauigkeit.

Verwendet wurden:

- MPU9250
- Arduino Nano 33 BLE
- ein Smartphone



Inhaltsverzeichnis

2. Einleitung

2.1 Problemlage

Als Bogenschütze bekommt man schnell mit, auf welch hohem Niveau andere Schützen es schaffen, immer wieder dasselbe zu tun. Dies bezieht sich nicht nur auf den allgemeinen Aufbau, bei dem diese Eigenschaft sogar von Nöten ist, sondern auch auf Fehler im Aufbau.

Da ich selbst schieße und die Möglichkeit hatte in zwei unterschiedlichen Vereinen zu trainieren ergab sich schnell das Problem, das meine Schussform drohte schlechter zu werden. Ohne Trainer, der meinen persönlichen Aufbau kannte, konnte ich nicht feststellen wann ich Fehler wiederholte oder sogar neue einbaute. Eine Lösung musste her, die es mir erlaubte meinen Schussablauf selbst nachzuverfolgen und Unterschiede oder Fehler selbst zu finden.

Bei Profi-Schützen wird schon seit Langem ähnliches getan. Mit Hilfe von mindestens 2 Hochgeschwindigkeitskameras und Punkten am Schützen ist man in der Lage den Bewegungsablauf eines Schützen zu analysieren und 3D darzustellen.

Die wohl bekanntesten Aufnahmen werden derzeit vorrangig zum Perfektionieren der mechanischen Ausrüstung und als Lehrmaterial verwendet. Da Profi-Schützen häufig einen nahezu perfekten Aufbau haben, werden die Daten selten zum Korrigieren von Aufbaufehlern eingesetzt.

Zur Auswertung der Daten ist eine weitere Person von Nöten, die speziell darauf geschult wurde die erzeugten Daten auszuwerten.

Die Hochgeschwindigkeitskameras, welche bei den ersten Aufnahmen von Bogenschützen benutzt wurden, sind in der Lage 6000 bis 8000 Fotos in der Sekunde zu schießen. Kameras dieser Qualität kosten selbst heute noch über 100 000 €.

Der Raum, den diese Messmethode in Beschlag nimmt, ist ebenfalls nicht zu unterschätzen.

2.2 Anforderungen

Wenn man es nun schafft den Kostenpunkt zu drücken und weniger Platz zu benötigen, ergeben sich völlig neue Möglichkeiten. Dies würde vor allem den kleinen Vereinen zu Gunsten kommen. Sie wären endlich in der Lage den Schussaufbau aller Schützen auf kleinste Fehler zu prüfen und eintrainierte Fehler einfach festzustellen.

Die Anforderungen an mein Projekt wären 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.

Ein weiterer Punkt der zu beachten war ist, dass der Schütze auf keinen Fall gestört werden darf. Dies wirkte sich bei meiner Idee vor allem auf die Größe, das Gewicht und die Datenübertragung aus.

Entwickelt und gebaut wurde dieses Projekt im Schuljahr 2020/21 von zuhause aus.

3.Vorgehensweise, Materialien und Methoden

3.1 Materialsuche

Es galt nun, die genannten Kriterien zur Hardware zu erfüllen. 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.

3.1.1 Möglichkeiten

Auf meiner Suche hatte ich die Idee, einen Ultraschallsensor am Schützen zu befestigen. Der Ultraschallsensor hätte am Boden befestigt werden können und um die Höhe der einzelnen Punkte des Schützen bestimmen. Ebenfalls wäre eine Kabelführung für schnellere Datenübertragung und ein günstigerer Preis möglich gewesen. Dieses System muss allerdings sehr genau auf den Schützen eingestellt werden und liefert wenig verwendbare Daten.

Auch die Verwendung einer günstigeren Kamera wäre möglich gewesen. Dabei vereint man allerdings alle Nachteile, die das jetzige System hat. Man braucht viel Platz und trotz sehr günstiger Kameras übersteigt der Preis sehr schnell den Kostenpunkt sowohl meiner Ersten als auch meiner folgenden Idee.

Meine finale Idee war, einen Beschleunigungssensor und einen BLE-Chip (Erklärung: siehe 3.1.3) zu kombinieren.

3.1.2 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. Alle Sensoren beinhalten eine Kalibrierung innerhalb der Firma und einen Selbsttest bei Benutzung. Der Sensor benötigt nur 2,4 bis 3,6 Volt für die Inbetriebnahme. In der Tabelle sind die Datenpins und die Genauigkeit der Sensoren notiert.

Sensorik	*Datenübertragung (**a**nalog-to-**d**igital-**c**onverters)*	*Empfindlichkeit (einstellbar)*
Gyroskop	3 * 16 bit ADCs	±250, ±500, ±1000, and ±2000°/sec (dps)
Beschleunigungssensor	3 * 16 bit ADCs	±2g, ±4g, ±8g, ±16g
Magnetometer	3 * 16 bit ADCs	full-scale range of ±4800µT
Übetragung	I²C, SPI	

Tabelle 1: Daten zum verwendeten Sensor

Die Daten werden über den I²C-Bus vom Arduino abgefragt. Die Abtastrate beträgt hierbei stolze mögliche 400kHz. Dabei werden alle Sensoren abgefragt und die Daten versendet.

Die geringe Größe des Chips (150 mm * 250mm), der geringe Energieverbrauch (3,5mA wenn alle Sensoren ausgelesen werden), die hohe Genauigkeit und Geschwindigkeit der Datenübertragung lassen diesen Sensor perfekt für dieses Projekt werden.

3.1.3 Arduino Nano 33 BLE

Der Arduino Nano 33 BLE ist wie der Name schon sagt 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 genannt Bluetooth-Low-Energy, mit allen anderen Bluetooth-Geräten ab der Bluetooth Version 4.0 kabellos zu kommunizieren.

Clock (Geschwindigkeit des Arduinos)	*64 MHz*
Memory	1 MB Flash | 256 KB SRAM
Interfaces	I²C ...
Volt	Input: 4,5 – 21 Volt Output: 3,3 Volt
Größe	18 mm * 45 mm

Tabelle 2: Daten zum verwendeten Arduino

Exkursion zum Thema BLE

Die BLE-Technik wurde entwickelt, um bei möglichst großen Datenübertragungen möglichst wenig Energie zu verbrauchen. Dies ermöglicht es mir, den gesamten Sensor mit einer deutlich kleineren Stromversorgung auszurüsten als mit dem Standard Bluetooth.

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

3.2 Vorgehensweise

Die gemessenen Daten müssen zur Verarbeitung an das Smartphone geschickt werden. Da dies über BLE passiert, müssen bestimmte Limits eingehalten werden, was wiederrum eine spezielle Routine auf dem Handy erfordert.

3.2.1 Überlegungen zu Datenübertragung

In der Programmierung von BLE-Geräten spricht man von „Services“ und „Characteristics“. Einfach gesagt sind „Services“ Ordner mir „Characteristics“ als Dateien. Der Dateipfad sind die UUIDs.

Für eine möglichst hohe Übertragungsgeschwindigkeit sollte der Client nicht zu viele „Characteristics“ abfragen, denn jeder Wechsel kostet Rechenzeit. Um mehrere Daten in eine „Characteristic“ zu legen benutzte ich die String()-Klasse von Arduino. Mit diesen kann man seine Daten in eine Text-Variable zusammenfügen und beim Client wieder entschlüsseln.

Dafür werden die Daten des Sensors (Datentyp: Float) über eine Arduino eigene Funktion in String umgewandelt. Diese Strings werden dann mit Trennzeichen in einen größeren String konkateniert, welcher dann über BLE „versendet“ wird.

Bei der Umwandlung in einen String kann ich der String()-Klasse einen Parameter für die umzuwandelnden Dezimalstellen mitgeben. Dieser Parameter wurde von mir auf 2 Dezimalstellen gesetzt. Ich brauche zwar genaue Werte, jedoch muss ich ebenfalls so wenig Daten wie möglich schicken, damit die Datengrenze nicht überschritten wird.

Zum Datenlimit sehen Sie bitte „4.2 Rechnung zum Datenlimit“.

3.2.2 Theoretisches Bogenschießen

Für die Darstellung des Bewegungsablaufs werden nicht alle Daten des Sensors benötigt. Um herauszufinden welche Daten ich wirklich brauche, befestigte ich den Sensorkasten am Schützen und beobachtete die Daten. Dabei hatte der Gyroskop Sensor und das Magnetometer wenige Veränderungen zu messen. Nach einigen Überlegungen und habe ich mich dafür entschieden nur die Beschleunigungsdaten des Sensors zu übertragen. So kann sogar die Übertragungsgeschwindigkeit erhöht werden.

3.3 Entwicklungsumgebungen

Für die Funktionsfähigkeit meiner Entwicklung musste ich noch zwei Programme entwickeln. Die übertragen Daten müssen vom Arduino ausgelesen und verarbeitet werden und dann zur Auswertung an das Empfangsgerät (z.B. das Handy) übertragen werden. Dieses muss die Daten dann sinnvoll darstellen, damit der Schütze eine Auswertung machen kann.

Verwendet wurden die Arduino IDE und die Website M.I.T App Inventor (https://appinventor.mit.edu).

Die Programme habe ich eigenständig geschrieben. Ideen und Anregungen habe ich mir in einigen Foren gesucht. Kleine Programmausschnitte konnte ich sogar direkt übernehmen. Hier musste ich jedoch den Code gut nachvollziehen, damit er in meine Programme integriert werden konnte.

3.3.1 Arduino Code

Der Arduino liest den Sensor aus, formt die Daten in einen großen String um und schickt diesen dann per BLE an das Handy/Tablet...

Je weniger der Arduino rechnen muss, desto kleiner kann die Batterie sein.

3.3.2 Android Code

Der Code auf dem Handy soll nicht nur die Daten empfangen, sondern diese auch anzeigen. Dazu müssen die Werte des Beschleunigungssensors (m/s²) in Strecken (m) umgerechnet werden. Diese Daten können dann unterschiedlich dargestellt werden:

1. Einfaches Diagramm (kann auch mit den „Rohdaten“ erzeugt werden)
2. 2D-Schaubild für Bewegungsdaten der Höhe oder Breite
3. 3D-Schaubild für umfassende Sicht

Bei den Schaubildern ist es wichtig, die Bewegungsdaten erst in eine Distanz und dann in Pixel auf dem Bildschirm zu übersetzten. Die Formeln zur Umrechnung der Daten stehen in „4.4 Umrechnung der Daten“. Leider habe ich bis jetzt keine Daten oder Tests zur 3D-Umgebung.

Die Übertragung und Darstellung der Rohdaten funktioniert schon und kann zur Auswertung benutzt werden (siehe Abbildung 1).

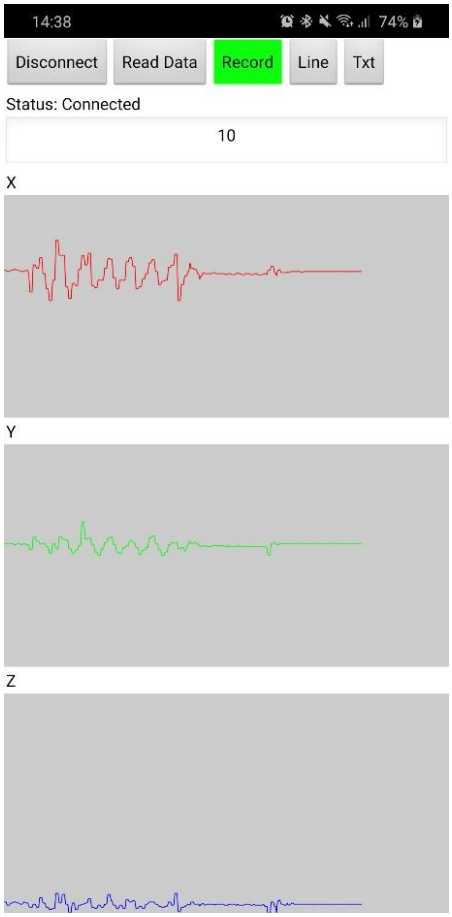


Abbildung 1: Darstellung der Echtzeit-Daten

Außerdem besteht die Möglichkeit die empfangenen Daten zu speichern und später wieder auszugeben.

Das Format der Datei ist „.txt“ und wird im internen Speicher des Telefons gespeichert. Die Auswahl der gespeicherten Daten erfolgt über Eingabe des Dateinamens.

Die Datei muss sich dabei im internen Speicher befinden. Das Anzeigen der gespeicherten Daten (siehe Abbildung 2) erfolgt in der getesteten Übertragungsgeschwindigkeit (Kapitel 4.3).

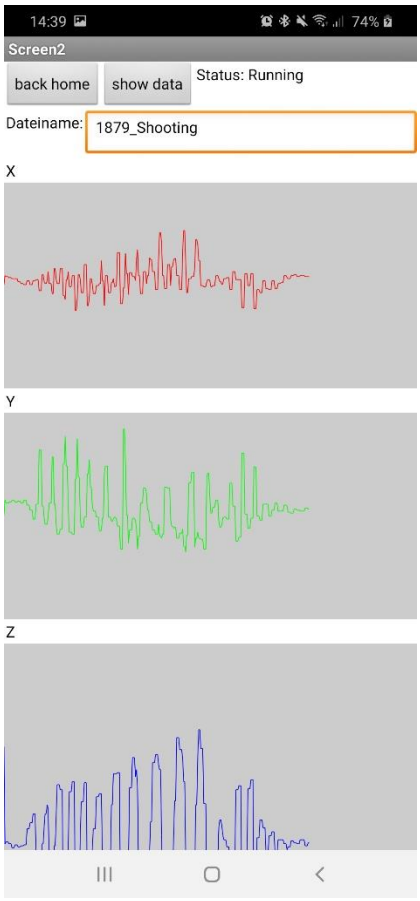


Abbildung 2: Darstellung der gespeicherten Daten

3.3.3 Details zur Darstellung

Das Diagramm besteht aus einem Canvas welcher sich automatisch an die Größe des Displays anpasst. Die Visualisierung der Daten funktioniert wie folgt:

1. Die empfangenen/gespeicherten Daten werden als y-Achse interpretiert. Sie werden mit 20 multipliziert wodurch gewährleistet wird, dass die maximalen Standartwerte nur den Rand des Canvas berühren, während die erwarteten Werte noch gut zu erkennen sind.

2. Als nächstes wird der neue Wert von der Höhe des Canvas subtrahiert. So wird die gezeichnete Linie in die Mitte des Canvas gesetzt und die Darstellung negativer Werte ermöglicht.

3. Der Wert der x-Achse wird an der Anzahl der Durchläufe ermittelt. Bei jedem Durchlauf wird der Wert der Variable eins größer als ihr vorangegangener Wert. Wenn der x-Achsen-Wert größer wird als der Canvas, wird die Variable wieder auf null gesetzt und der Canvas gelöscht, um wieder neu bezeichnet zu werden.

Dieses System wird sowohl bei der Echtzeit-Darstellung verwendet als auch bei der Darstellung der gespeicherten Daten.

4. Test zur Datenübertragung

Die folgenden Tests und Rechnungen überprüfen, wie groß die erwarteten erzeugten Daten sind, ob ich diese in einem Datenpaket versenden kann und wie schnell das eigentliche Lesen und Senden der Daten von statten geht.

Danach wird die Theorie hinter der 2D/3D-Animation erläutert.

4.1 Test auf Stellengröße der Sensordaten

Für eine genau Berechnung ist es nötig herauszufinden, wie viele Stellen die ausgegebenen Daten des Sensors haben. Um die Stellenzahl herauszufinden ließ ich alle Daten auf dem Seriellen Monitor der Arduino IDE anzeigen und habe durch Bewegung des Sensors für möglichst große Werte gesorgt. Das Ergebnis waren 2-3 Vorkommastellen (inkl. Minuszeichen) gefolgt von 6-7 Dezimalstellen. Der Output des Sensors ist somit 9-10 Stellen groß.

4.2 Rechnung zum Datenlimit

So hat ein Sensor-Wert die Größe von 7 Char's oder auch 7 Byte (siehe Kapitel 5.1).

Rechnung:
(Vorkommastellen) 3 + (Komma) 1 + (Dezimalstellen) 2 + (Null-terminierung) 1 = 7 (Char)

Um nun die komplette zu bringende Datenmenge zu errechnen muss man nun 9 Werte aus den Sensoren, 8 Trennzeichen und eine weitere Null-Terminierung einberechnen.

Rechnung:
(Sensoren)9 * (Byte)7 + (Null-Terminierung) 1 = 64 (Byte)

Dass diese Datenmenge in das Datenlimit passen, ist mit dieser letzten Rechnung bewiesen:

Rechnung:
(Maximale Daten) 512 – (benötige Daten) 64 = 448 (Byte)

448(Byte) >= 0 (Byte)

4.3 Test zur Übertragungsgeschwindigkeit

Sowohl beim Arduino als auch auf dem Empfangsgerät wird die Zeit gemessen, die beide Geräte brauchen, um das gesamte Programm zu durchlaufen. Die Zeit wurde in Millisekunden auf dem Bildschirm ausgegeben.

Arduino: 46 (22 Hz)

Android: 19 (52 Hz)

Wie sich die fast doppelt so schnelle Laufzeit des Handys auf die Datenverarbeitung auswirkt kann ich leider noch nicht sagen.

4.4 Umrechnung der Daten

Beim Test auf die Stellengröße der Sensordaten fiel mir auf, dass die Achse des Sensors die gerade nach oben zeigt, durchgehend etwa 9.81 m/s² anzeigt. Da die Erdanziehungskraft etwa 9.81 m/s² beträgt, schließe ich, dass der Sensor die Werte statt in „g“ (wie in der Dokumentation beschrieben) in m/s² ausgibt. Dadurch entfällt die Umrechnung in m/s².

Die folgenden Formeln werden benötigt, um die Beschleunigungsdaten in eine erfassbare Strecke umzurechnen:

Formeln:

a * t = v

s = 0,5 * a * t² + v *t

Für das Programmieren bedeutet dies:

[Daten] * [Erfassungszeit] = v

s = 0,5 * [Daten] * [Erfassungszeit]² + v * [Erfassungszeit]

5. Modell

Das fertige Modell ist 900 * 200 * 750 mm groß und kann damit fast überall am Schützen angebracht werden. Die Box wiegt samt Akku 73 Gramm. Da dieses Gewicht jedoch fast allein durch den Batterietyp bestimmt wird kann man das Gewicht einfach verringern. Die Datenübertragung des Beschleunigungssensors funktioniert problemlos. Bei dem Versuch, alle Daten des Multi-Chips zu übertragen schien der String das Datenlimit zu überschreiten und es wurden maximal 4 Werte übertragen. Alle Daten zu senden ist also mit dem jetzigen Code nicht möglich, allerdings auch nicht nötig.

Sowohl das Diagramm als auch die geplante Animation funktionieren auch mit den einzelnen Daten des Beschleunigungssensors.

Die erwähnte Darstellung als Diagramm ist als erste Auswertungsmethode für den Laien schnell zu erlernen und ermöglicht eine einfache Fehlersuche. Durch die Trennung der X, Y und Z-Achsen hat man ebenfalls den wohl bis jetzt genauesten Überblick über alle Daten.

Die Übertragungsgeschwindigkeit von 46 Millisekunden pro Datenstrang entspricht einer Kamera mit 22 Bildern pro Sekunde.

Der Bau-Preis von etwa 25€ erfüllt die Anforderung für alle Vereine eine Alternative zu finden.

6 Zusammenfassung

Aufgrund meiner Vorüberlegungen wusste ich exakt was ich benötigen würde und konnte so die Materialien schnell zusammenstellen.

Anfängliche Schwierigkeiten bei der BLE-Verbindung mit dem Test-Code wurden schnell behoben und der Schritt zum eigentlichen Projekt wurde möglich. Der Fehler war ein noch nicht behobener Bug in einer der offiziellen BLE-Bibliotheken.

Die visuelle Darstellung war für mich das spannendste am Projekt und lief gut von der Hand. Etwaige Probleme wurden schnell erkannt und gelöst. Die einfache Darstellung ermöglicht es jedem, nachzuvollziehen wo seine Fehler liegen und wie überall, ist dies der erste Schritt zur Verbesserung.

Das Ziel, eine kleinere und günstigere Methode zum genauen Vermessen des Schussaufbaus zu erfinden wurde erreicht. Leichte Abstriche bei Datenrate und Genauigkeit müssen im Vergleich zu den heutzutage benutzten Hochgeschwindigkeitskameras zwar gemacht werden, aufgrund unterschiedlicher Ziele und Aufgaben wird es allerdings kompliziert mein Projekt mit dieser Methode zu vergleichen.

Die Hochgeschwindigkeitskameras benötigen die hohe Datenrate, um auch noch einen gerade abgeschossenen Pfeil sehen zu können. Der Schussaufbau eines Schützen zieht sich jedoch über mindestens drei Sekunden und ermöglicht es, die Datenrate bei gleichbleibender Qualität herunter zusetzten.

Empfehlen würde ich den neuen Sensor für Schützen, die entweder in einem Zweitverein schießen oder ohne Trainer zu Hause schießen. Beide Schützen brauchen Hilfe dabei, unabhängig vom Umfeld ihren Schussaufbau sicher zu wiederholen und Veränderungen sofort zu bemerken.

Mit der richtigen Software wäre es sogar möglich, Anfängern eine Trainingsmöglichkeit zu geben die nicht sofort einen Trainer benötigt. Der grobe Aufbau ist bei jedem Schützen gleich. Wenn man diesen zu Anfang spielerisch an neue Schützen heranführt erspart man den Trainern viel Arbeit.

Auch ein günstiges und genaues „Full-Body-Tracking“ für diverse VirtualReality-Spiele ist mit meiner Methode realisierbar. Die Möglichkeiten sind endlos.

Jetzige Probleme sind vor allem Hardware bezogen. So ist es möglich, die jetzt benutze 9V-Batterie gegen eine leichtere Batterie auszutauschen. Dies würde das hohe Gewicht drastisch senken. Ebenfalls kann man die Größe der Platine so verkleinern.

Zukunftspläne sind die 3D-Darstellung des Schützen mit Indikatoren, die Unterschiede zum vorherigen Schussaufbau zeigen. Selbst Trainingspläne mit gewollten und im Programm hervorgehobenen Änderungen am Aufbau sollen möglich werden.

Quellen

Alle Quellen wurden am 4.1.2021 auf Aktualität geprüft.

- Q1 : http://www.wernerbeiter.com/de/produkte/videos/video.php
- Q2 : https://www.novelbits.io/bluetooth-low-energy-advertisements-part-1/
- Q3 : https://www.youtube.com/watch?v=5YovICoYCLg
- Q4 : https://community.appinventor.mit.edu/t/ble-esp32-bluetooth-send-receive-arduino-ide/1980/2
- Q5 : https://www.electronoobs.com/eng_arduino_tut20_1.php
- Q6 : https://invensense.tdk.com/wp-content/uploads/2015/02/PS-MPU-9250A-01-v1.1.pdf

Unterstützungsleistungen

- Daniel Jenkner (Schiller-Gymnasium, Korrekturlesen der schriftlichen Arbeit)
- Marek Czernohous (Schiller-Gymnasium, Korrekturlesen der schriftlichen Arbeit, Kauf der Materialien)