Produktrealisierung Spot-your-Squat

Dokumentation

Lehrveranstaltung:

Produktrealisierung

720.036

Student*innen

Anna Dopona

Christoph Kölly

Denise Rabenseifner

Merle Rustler

Raphael Roupec

Created on:

28.12.2024

Inhaltsverzeichnis

1 Hardware:

Das Produkt wurde mithilfe von Hardwarekomponenten, welche ohne Software keine Funktion besitzen, realisiert. Diese werden in Punkt 1.1 dokumentiert. Wenn im Punkt 1 von Software die Rede ist, bezieht sich dies auf Punkt 2. Die elektronischen Geräte wurden, nach Zusammenbau in ein Gehäuse verpackt (siehe Punkt 1.2).
Zuständigkeiten (Hardware):

- Testung und Ankauf: Raphael Roupec
- Übersicht und Koordination / Projektleitung: Raphael Roupec
- Entwicklung und Zusammenbau: Raphael Roupec, Denise Rabenseifner, Merle Rustler, Anna Dopona, Christoph Kölly
- Gehäuse: Christoph Kölly
- Dokumentation Zusammenfassung: Raphael Roupec, Christoph Kölly

Das gesamte Projekt wurde in 3 Phasen realisiert. In der ersten Phase wurden die einzelnen Komponenten festgelegt, gekauft und getestet (September – November). In der zweiten Phase wurde das Innenleben des Produktes mit seinen Kernfunktionalitäten zusammengestellt (Dezember). In der dritten Phase wurde das Produkt in sein Gehäuse verpackt und finalisiert (Jänner).

1.1 Elektronische Komponenten

Das Produkt setzt sich aus folgenden Komponenten zusammen (Abbildung 1):

- Gehäuse (3D-Druck)
- Arduino Rev3 (Arduino Starter Kit)
- 9V – Batterie (Amazon Product Line)
- IMU-Sensor MPU9250 (Amazon)
- Adafruit 7 Segment Anzeige Display (LED) – HT16K33 (Amazon)
- Knightbright RGB Elektrode (Amazon Starter Kit)
- 2x 10 k Ω Widerstand (Arduino Starter Kit)
- 21 x Spannungsleiterkabel (Ardunio Starter Kit) (verschiedene Farben)
- SPDT Switch (Arduino Starter Kit)

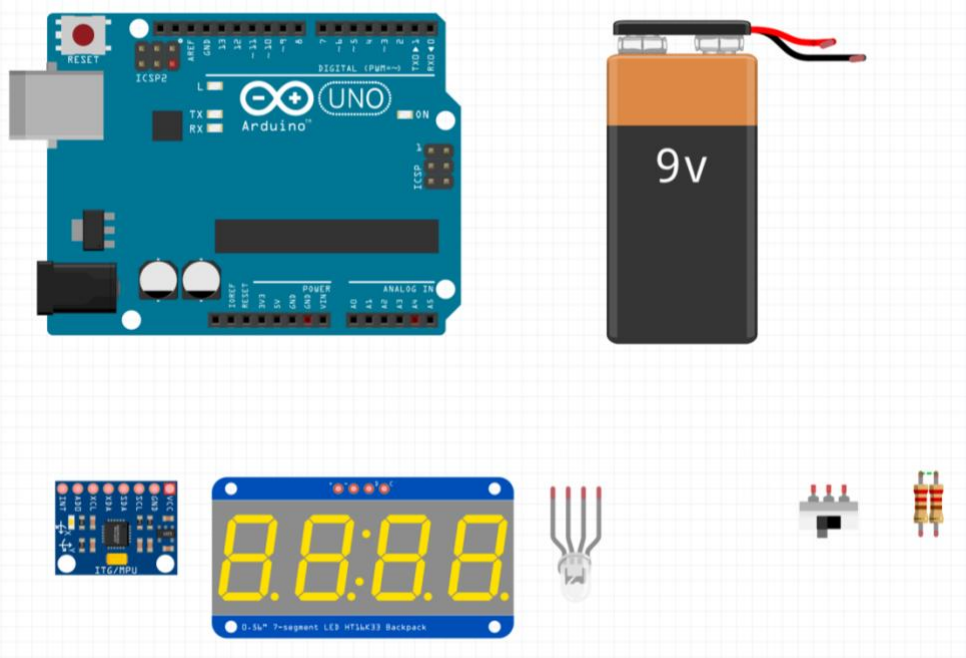

Abbildung 1: Hardwarekomponenten eigene Darstellung

1.1.1 Kommunikationsprotokoll:

Um das Kommunizieren von den einzelnen Komponenten zu ermöglichen, wurde auf die bereits bestehende Funktionalität des I2C – Protokolls aufgebaut. I2C ermöglicht es mehrere Komponenten über ein Master / Slave System anzusprechen (Abbildung 2). Bei einzelnen Komponenten ist kein Pull Up Widerstand notwendig, bei mehreren Komponenten ist es dies aber verpflichtend. Der Pull Up Widerstand wurde mit zwei 10 k Ω umgesetzt. Software wurde entwickelt, um Werte aus dem Sensor auszulesen und diese interpretierbar zu machen (Überfunktion Lesen). Für das Display und RGB-Diode wurde Software entwickelt, um die interpretierte Information auszugeben (Überfunktion Schreiben).

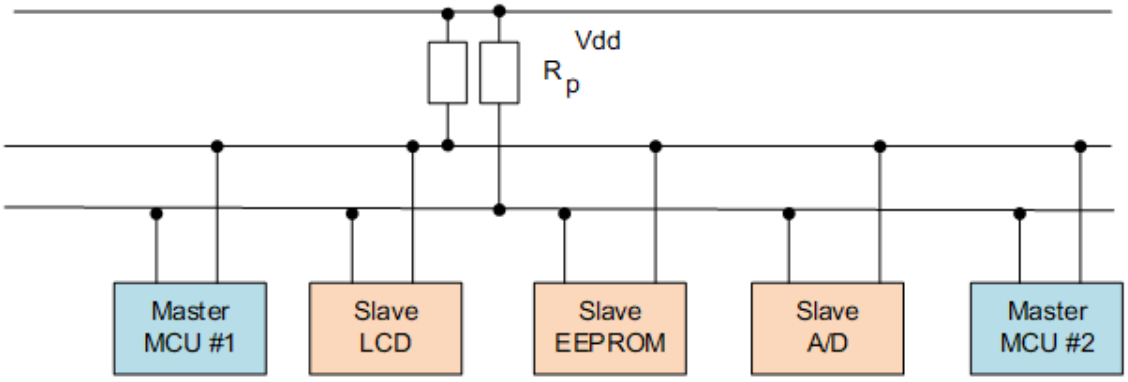

Abbildung 2: I2C Protokoll Übersicht - von: https://www.rt-thread.io/document/site/programming-manual/device/i2c/i2c/

1.1.2 Ablauf:

Nach dem Kauf der Komponenten wurden diese in der ersten Phase getestet. Begonnen wurde mit dem IMU-Sensor, dieser wurde in den vorherigen Semestern schon für ein Microcontroller Projekt verwendet. Die Komponente wurde über einen Pull Up Widerstand verbunden und lieferte Daten, funktionierte somit einwandfrei. Als nächster Schritt wurde das Display in den Schaltkreis eingebaut. Der Aufbau der Testung ist in Abbildung 3 gezeigt. Für diesen Aufbau wurde ein kleines Programm geschrieben, welches die gemessenen Werte des Gyroskops in einer Achse auf dem Display anzeigen lässt. Dieses Programm wurde leider nicht abgespeichert oder wurde im Verlauf des Projektes überschrieben. Daher ist es derzeit nicht genauer dokumentiert. Der Test erwies sich aber als Erfolg, da die Werte des gemessenen Sensors einwandfrei angezeigt werden konnten.

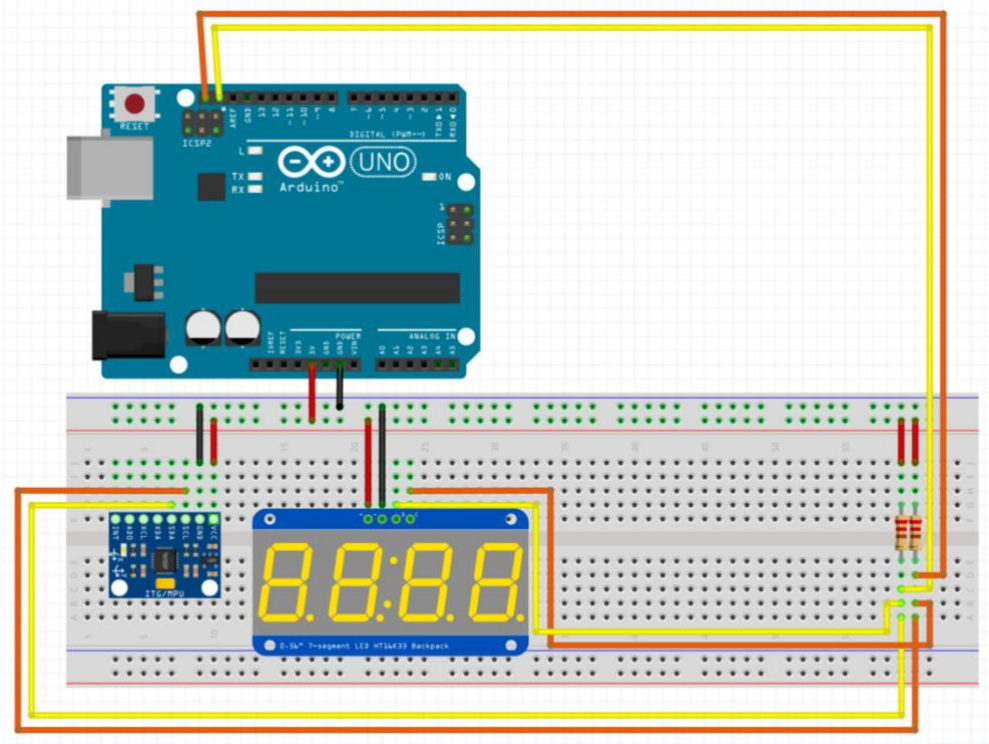

Abbildung 3: Testaufbau - IMU und Display eigene Darstellung

In der zweiten Phase wurden alle Komponenten gleichzeitig auf einem Steckbrett zusammengebaut. Abgesehen von Problemen dokumentiert in 1.1.4 gab es keine groben Zwischenfälle. Das noch ungeschützte Innenleben des Produktes wurde daraufhin getestet. Die Hauptanforderungen an das Innenleben konnten als erfolgreich implementiert festgehalten werden. Einerseits zählte das Display die erfolgreichen Kniebeugen und andererseits wurde über eine LED die korrekte Farbe

ausgegeben. Der Aufbau auf dem Steckbrett ist in Abbildung 4 dargestellt. Allerdings gab es noch einige unausgereifte Passagen im Code die einige Fehler/Bugs produzierten, außerdem kam es zu einigen Wackelkontakten. Deshalb wurde beschlossen in der dritten Phase die Kontakte zu verlöten, bzw. durch Steck Systeme zu ersetzen. Es gehört angemerkt, dass der Switch in dieser Variante noch nicht eingebaut war und die RGB - Diode durch einen LED-Streifen ersetzt war (der LED-Streifen hatte 2 Eingänge (PIN 6 & 15) und einen Stromversorgungseingang).

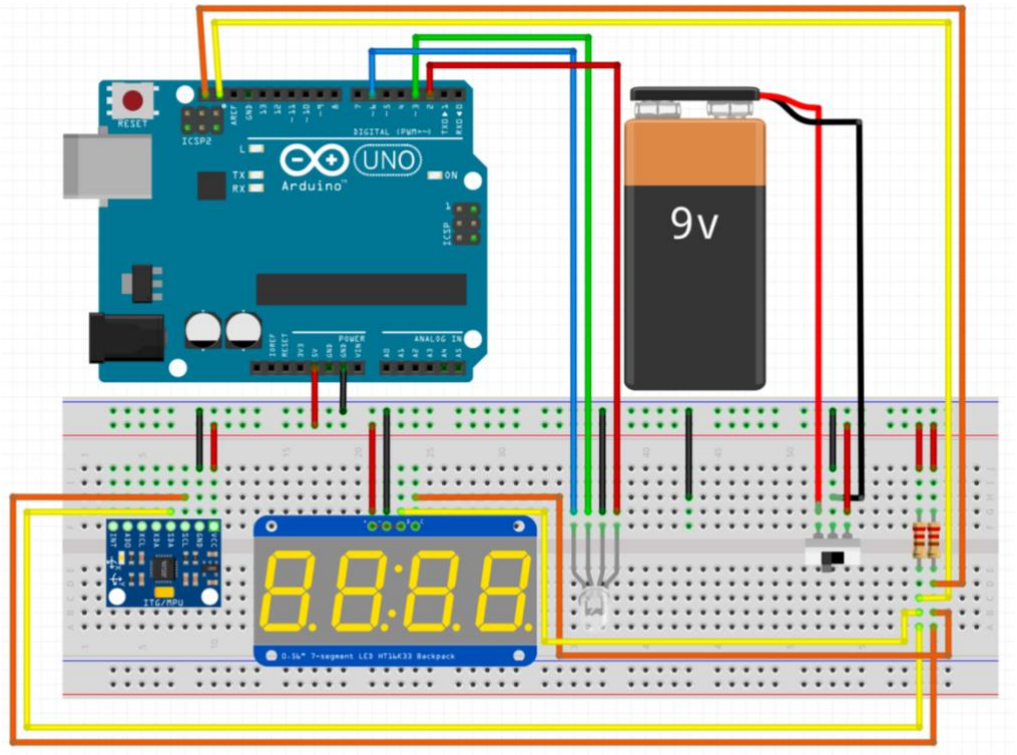

Abbildung 4: Darstellung des Produktes in der zweiten Phase eigene Darstellung

In der dritten Phase war geplant das Innenleben in das Gehäuse zu passen und alle Verbindungen/Komponenten zu verlöten. Hier kam es dann zu einigen Problemen, in Punkt 1.1.4 wird darauf genauer eingegangen. Nach zwei Versuchen zu unterschiedlichen Terminen war das Produkt fertig. Das Innenleben war miteinander verlötet und es gab keinerlei Wackelkontakte. Alle geforderten Funktionen abgesehen von der LED wurden implementiert. Leider funktionierte zu diesem Zeitpunkt die eingefügte RGB-Diode nicht vollständig (leuchtete ununterbrochen ROT). Für eine Fehlerbehebung und ausreichende Nachtestung, fehlte dann leider die Zeit. Das fertige Produkt ist im Anhang 1 zu sehen.

1.1.3 Schaltdiagram:

In der letzten implementierten Version des Produktes wurde folgendes Schaltdiagram angewandt, um das Produkt zu realisieren. Das Produkt erfüllt in dieser Funktion alle MUSS-Kriterien, außer die Ausgabe eines Lichtsignals als Feedback für die Bewegung. Ob dies an diesem Aufbau liegt oder nicht wird in Punkt 1.1.4 (LED-RGB Diode) diskutiert.

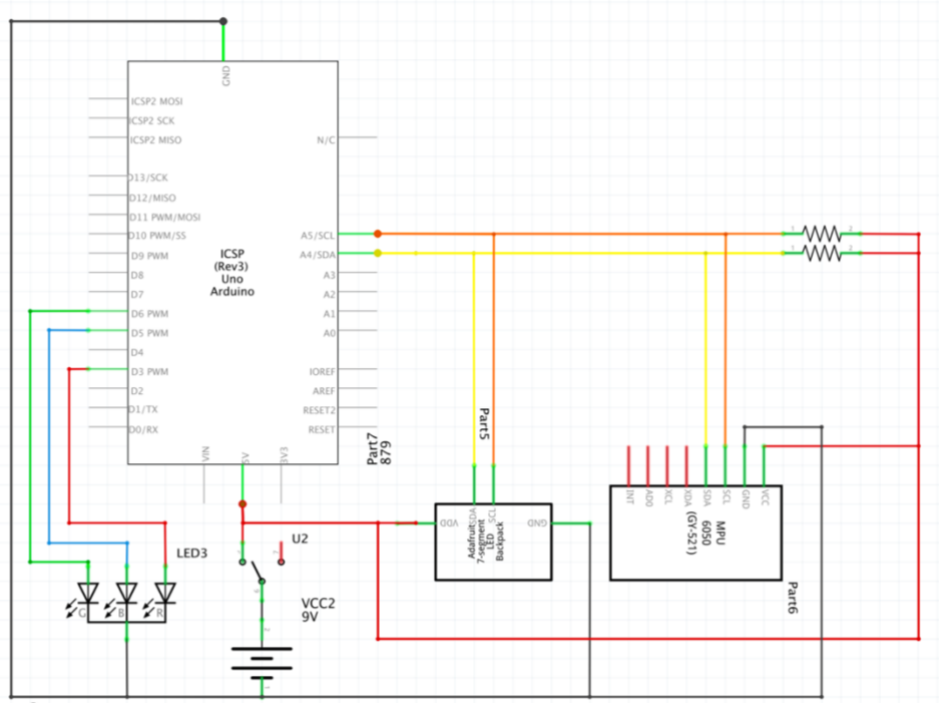

Abbildung 5: Schaltdiagramm eigene Darstellung (konzipiert von einem nicht Elektrotechniker)

1.1.4 Probleme/Herausforderungen

Allgemeine Herausforderung – Verkabelung (Phase 2)

Das korrekte Zusammenfügen und Verbinden der einzelnen Komponente, stellte sich als eine doch anspruchsvolle Herausforderung dar. Allerdings gab es einige hilfreiche Dokumentationen von Produkthersteller*innen und Personen die ähnlichen Probleme in der Vergangenheit in diversen Internetforen besprochen haben, auf die zurückgegriffen werden konnte. Auch das Kabelmanagement und das Behalten des Überblicks erwies sich anfänglich als schwer.

LED / RGB – Diode (Phase 3)

Grundsätzlich war geplant gewesen, den LED-Streifen durch eine einzelne LED zu ersetzen. Im Zuge des Verarbeitungsprozesses beim ersten Termin wurde die LED beschädigt. Es wurde daraufhin beschlossen, sie durch eine im Arduino Starter Kit enthaltene RGB-Diode zu ersetzen.

Es gibt mehrere Möglichkeiten wieso die LED/RGB – Diode in der letzten Iteration nicht ordnungsgemäß funktioniert hat. Drei Möglichkeiten wurden identifiziert:

A. Beschädigte Komponente – RGB-Diode
B. Fehlerhafter Code – Implementierung falsch/Bug
C. Fehlerhafte Verkabelung bzw. fehlender Pull Up Widerstand

Option B ist am unwahrscheinlichsten. Die RGB-Diode wurde vor Einbau getestet und funktionierte einwandfrei. Allerdings könnte theoretisch eine falsche Code-Version auf dem MCU-Prozessor vorhanden gewesen. Dies wurde am Donnerstag jedoch widerlegt als der korrekte Code hochgeladen wurde und die LED immer noch keine korrekte Farbe lieferte, obwohl eine korrekte Kniebeuge erkannt wurde. Deshalb erscheint entweder Option C, Option D oder Option A wahrscheinlicher. Es ist möglich, dass beim zweifachen Zusammenlöten der Komponente Hitze übertragen wurde und diese dadurch geschädigt wurde. Auch ist es möglich, dass die Verkabelung nicht ordnungsgemäß funktionierte. Durch das Anschließen an ein Spannungsmessgerät konnte aber gezeigt werden, dass die Kabel selbst nicht beschädigt waren und theoretisch Spannung übertragen wird. Auch das Fehlen eines Pull Up Widerstandes wurde diskutiert, angeblich scheint dies öfters ein Problem bei RGB-Dioden zu sein. Allerdings wurde sie beim Testen nicht an einen Pull Up Widerstand angeschlossen und die RGB - Diode funktionierte trotzdem.

Problem beim ersten Termin – Fehlender 3D Druck/ Beschädigte LED (Phase 3)

Phase 3 war ursprünglich in zwei Termine aufgeteilt. Es war geplant das Produkt, am 12.01.2024 (erster Termin) fertig zu stellen. Dazu war vereinbart, das fertige Gehäuse zu überprüfen und die einzelnen Komponenten zu verlöten. Der zweite Termin, das Zeitfenster zwischen 22.01.-23.01.2024 war als extra Termin für etwaige Behebung von Problemen eingeplant gewesen. Dies wurde auch so von allen Teilnehmer*innen zur Kenntnis genommen und mit allen Parteien besprochen.

Am 12.01.2024 war allerdings der 3D-Druck noch nicht fertig. Zum Zeitpunkt des Treffens war dieser in zwei Orten, einmal vor Ort und einmal bei einem anderen

Kollegen mit einem anderen Druckverfahren gerade am Drucken. Der Druck vor Ort wurde nach 3 Stunden fertig. Die Zwischenzeit wurde genutzt, um die Komponenten zu verlöten. Im Zuge dessen kam es zur Beschädigung der einzubauenden LED. Es wurde beschlossen dies beim nächsten Termin mit einer entsprechenden Abänderung der RGB-Diode und deren Implementation in den Projektcode nachzuholen. Nach Beendigung des Druckes stellte sich heraus, dass die Maße des 3D-Drucks nicht passten. Im Detail konnte die Anzeige nicht in das Gehäuse gepasst werden und die Halterung für den IMU-Sensor war fehlerhaft. Dies wurde auf einen Kommunikationsfehler bei der Übermittelung der Maße des IMU – Sensors und fehlerhaftem Auslesen eines Datenblattes zurückgeführt. Das zu eng Anliegen des MCU an das Display machte eine Verbindung mit der Stromversorgung außerdem unmöglich. Da die Komponente nicht in das Gehäuse gepasst werden konnten, wurde das Innenleben nach erfolgreicher Verlötung kurz erneut geprüft. Die erfolgreichen Wiederholungen wurden gezählt und die Streifen LED gab die korrekte Farbe aus. Aufgrund der späten Stunde wurde vereinbart entweder am 22.01.2024 oder 23.01.2024 weiterzumachen.

Probleme beim zweiten Termin – Terminkoordination/Verfehlung (Phase 3)

Wegen persönlicher Gründe (Krebs im näheren Bekanntenkreis) war der Projektleiter nicht in der Lage den Termin am Montag wahrzunehmen. Dies wurde allerdings kommuniziert und der Vorschlag übermittelt den am Dienstag eingeräumten Termin am 23.01.2024 zu nutzen. Von 5 Personen bejahten 4 an diesem Tag teilnehmen zu können. Die einzige Person, die an diesem Tag keine Zeit hatte, war die Partei zuständig für den 3D Druck. Mehrere Lösungsstrategien wurden angeboten. Doch noch am Montag das Meeting zu zweit abzuhalten (Projektleiter und 3D-Druck Zuständiger). Dies wurde verneint die Partei sei zu diesem Zeitpunkt arbeiten. Dienstag sei keine Zeit aber Mittwoch, der 24.01.2024 wäre möglich. Auf erneute Anfrage an alle Teilnehmer konnte niemand den Termin wahrnehmen, auch der Projektleiter war verhindert. Das Abliefern des 3D-Drucks wurde angeboten, dies sei nicht möglich. Schließlich wurde über Telefon vereinbart, dass die Komponenten von dem Zuständigen für 3D-Druck abgeholt werden und am Mittwoch allein zusammengebaut werden.

Am 23.01.2024 um 19:00 wurden die Komponenten abgeholt (verspätet - 18:00 vereinbart). Es gab dann am 24.01.2024 einige Probleme. Unter anderem hatte der Teilnehmer **selbstständig** beschlossen alle Komponente neu zu verlöten wegen ästhetischen Gründen und die Funktion des Produktes war nicht mehr gegeben. Im Detail wurden alle Komponenten initialisiert, aber danach zählte das Produkt auch bei keinem Input kontinuierlich nach oben. Dies ist auf einen Error Code zurückzuführen (I2C-Error). Dieser Error – Code erscheint, wenn das I2C Protokoll nicht richtig initialisiert wird und/oder die Verbindung zum Sensor/Display fehlerhaft ist. Im Zuge von Testungen trat dieses Problem ebenfalls auf (mehr dazu in Punkt 2.4.1) und konnte deshalb auch ohne Programminterface als sehr wahrscheinlich identifiziert werden.

Der Partei wurde daraufhin zweimal telefonisch um 20:00 Beistand geleistet. Es wurde kommuniziert, was die wahrscheinlichste Fehlerquelle ist, und potenzielle Lösungsstrategien übergeben. Zu dem Zeitpunkt war es sehr wahrscheinlich, dass:

A. Verkabelung fehlerhaft
B. Sensor beschädigt
C. Programmcode fehlerhaft (unwahrscheinlich – keine Veränderung bis auf RGB-Implementierung)

Es wurde zur Durchtestung aller Komponenten und der Serial Ausgabe – Überprüfung in der Arduino IDE geraten. Letzteres konnte nicht getan werden, da kein Anschlusskabel für den PC /MCU vorhanden war.

Dies war die letzte Kommunikation mit der Partei, keine weitere Meldung erfolgte danach. Am nächsten Tag wurde um 02:13 ein Video in unsere WhatsApp – Gruppe hochgeladen. Das Produkt sei nach erneutem Verlöten wieder funktionstüchtig allerdings funktionierte die RGB nicht mehr richtig.

Keine weitere Dokumentation ist vorhanden….

1.2 Gehäuse

Die Maße der Komponenten wurde im Dezember an die zuständige Partei vermittelt.

Die zuständige Partei wurde am 27.01.2024 aufgefordert die Dokumentation des Gehäuses bis 31.01.2024 zu übermitteln. (Chatverlauf)

Abbildung 6: Projektgruppen - Chatverlauf von 27.01-01.02.2024

Ein Dokument wurde an den Projektleiter übermittelt und dem Wunsch wurde nachgegangen.

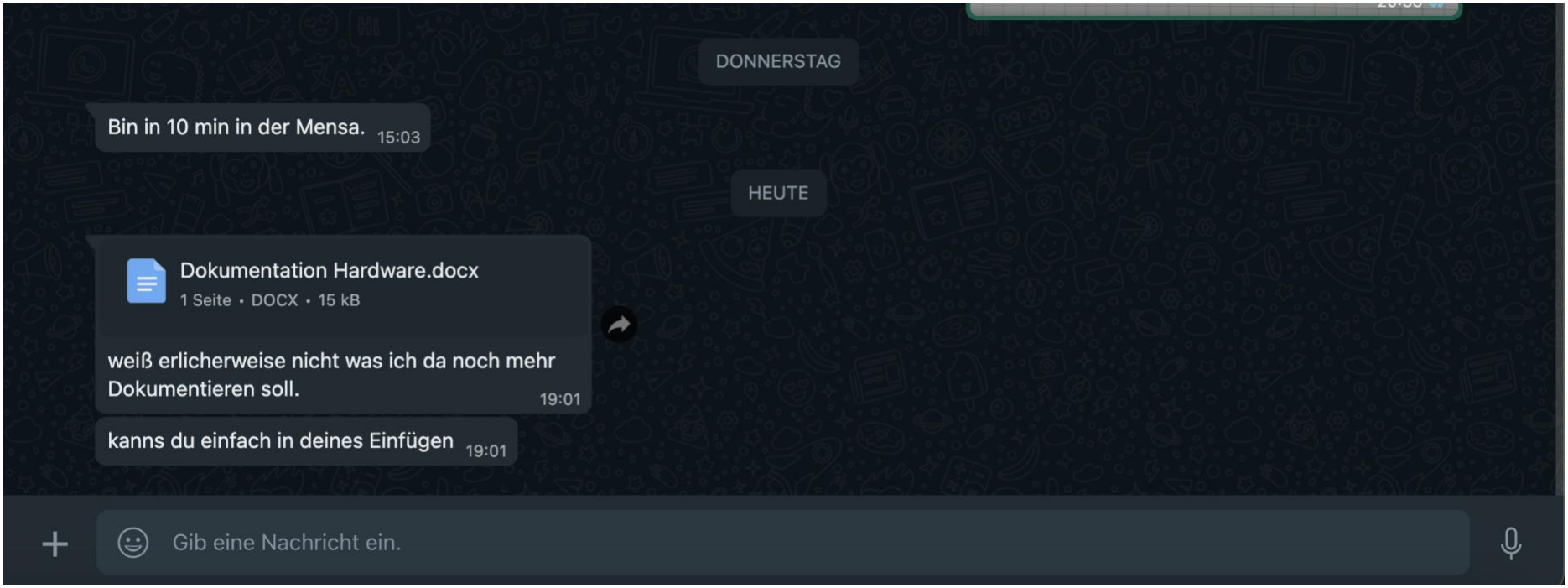

Abbildung 7: Chatverlauf Raphael Roupec - Christoph Kölly 31.01.2024

Inhalt des Dokumentes:

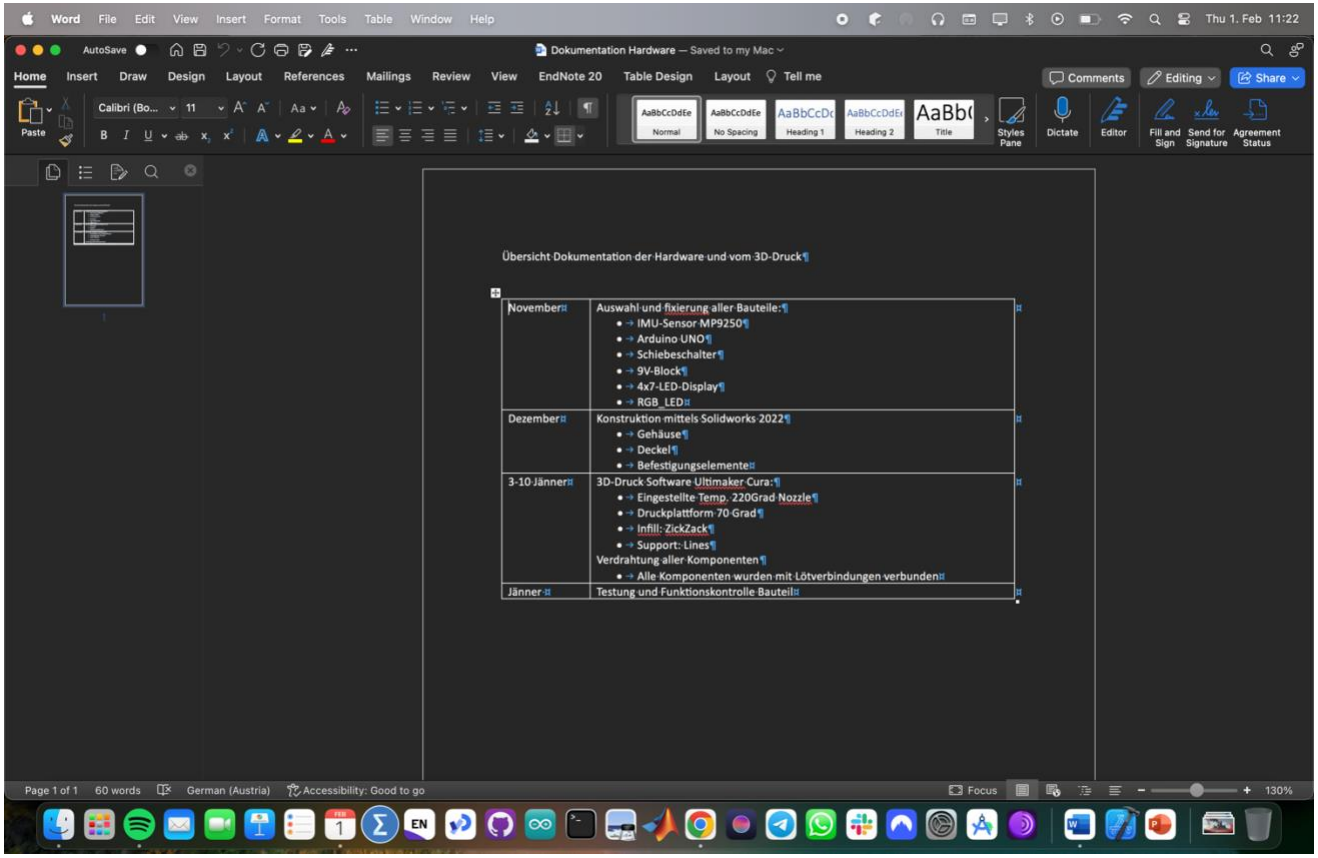

Abbildung 8: Inhalt des übermittelten Dokumentes aufgerufen am 01.02.2024

Dieses Vorgehen begründet sich in: Nicht Wahrnehmen / Probleme in der Terminkoordination und nicht Erbringen von Leistung,

Bei Fragen bitte den Verfasser des Dokuments: Christoph Kölly kontaktieren. christoph.koelly@stud.fh-campuswien.ac.at

Die Vorgehensweise wurde Teamintern abgesprochen und Konsens wurde eingeholt. Die betroffene Partei wurde darüber informiert.

Erneute Anfrage um 18:00 bis 22:00 keine Übermittlung weiteren Materials

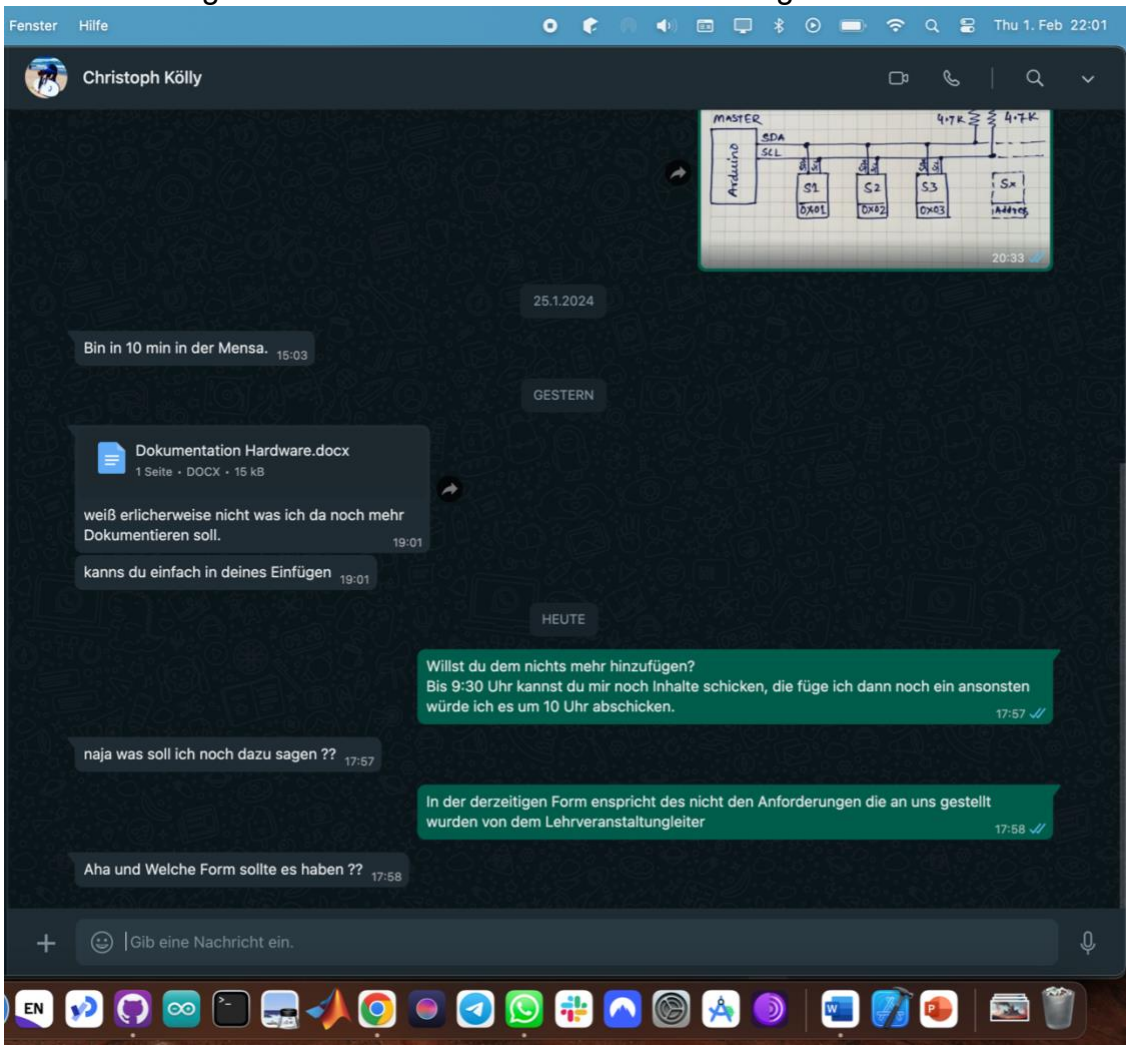

Abbildung 9: Screenshot Whatsapp Chatverlauf Raphael - Roupec 01.02.2024 um 10:00 PM

2 Software:

Für die Entwicklung der Software wurde die Arduino IDE benutzt. Der Syntax des Codes folgt den Anforderungen der C/C++ ähnlichen Programmiersprache des Arduino Systems. Im weiteren Dokument sind alle Funktionen mit „void function eigenname“ gekennzeichnet. Grundsätzlich müssen zwei Methoden benutzt werden:

- void setup (wird nur einmal aufgerufen zu Beginn)
- void loop (endlosschleife wird nach void setup kontinuierlich, bis es zu einem interrupt kommt aufgerufen)

Es wurden 5 Überkategorische Funktionen implementiert, um den Anforderungen an das Produkt gerecht zu werden. Die erste Funktion initialisiert die IMU-Sensoren und etabliert die I2C Verbindung. Die zweite Funktion initialisiert alle Feedbackelemente (Display / LED / RGB) und setzt sie auf ihre Anfangswerte/ausgaben. Die dritte Funktion ermöglicht das Auslesen von Daten. Damit sie dann in der vierten Funktion interpretiert werden können. Im Detail geht es hier um die Erkennung der Tiefe eines Squats. Wenn eine Kniebeuge tief genug ist, soll dies festgehalten werden. Die fünfte Funktion gibt das Signal bei einer erfolgreichen Wiederholung dann aus. Einerseits wird ein Zählstand am Display angezeigt, andererseits leuchtet die RGB / LED (im Ampelsystem) je nach Tiefe der Kniebeuge.

Das gesamte Projekt war in 3 Phasen geplannt. In der ersten Phase wurden die einzelnen Komponenten via Softwarebibliotheken getestet. Der dazugehörige Code wurde nicht abgespeichert, bzw. überschrieben. (September – November). In der zweiten Phase wurde das Innenleben des Produktes mit seinen Kernfunktionalitäten zusammengestellt (Dezember). Hier war es wichtig, dass alle Funktionen implementiert werden und im Zusammenspiel die MUSS Kriterien erfüllen kann. In der dritten Phase wurde das Produkt in sein Gehäuse verpackt und finalisiert (Jänner). Dies hatte bis auf die Abänderung der LED zu einer RGB - Dioden Implementierung keine große Auswirkung auf den Code. Die letzte verwendete Code Implementierung ist in diesem Dokument genauer aufgeschlüsselt.

Zuständigkeiten (Software):

- Übersicht und Koordination / Projektleitung: Raphael Roupec
- Entwicklung des Codes: Raphael Roupec, Denise Rabenseifner, Merle Rustler, Anna Dopona
- Dokumentation Zusammenfassung: Raphael Roupec

Der Sourcecode ist auf der GitHub Seite einsehbar, er wurde auch im Code selbst dokumentiert. Dort wurde ein Repository angelegt, um den Code abzusichern und ihn für alle frei zugänglich zur Verfügung zu stellen.
GITHUB: https://github.com/roupecraphael/Arduino_Protype

2.1 Libraries

Es wurden 4 externe Libraries benutzt, um das Projekt zu realisieren. Eine der hier angeführten Libraries ist derzeit nicht in Verwendung, aufgrund des Wechsels der LED zu RGB – Diode, wurde aber im Code belassen. (Links im ReadMe.gh auf GitHub)

- <MPU9250.h> by hideakitai (IMU Sensor)
- <Adafruit_NeoPixel.h> by Adafruit (Display Library)
- <Wire.h> by Arduino (I2C – Protokoll Implementer von Arduino)
- <Adafruit_GFX.h> (Adafruit- Teil von „Adafruit_LEDBackpack.h")
- „Adafruit_LEDBackpack.h" (Sammelgruppe/ Library für alle Adafruit Segment LEDs - Adafruit)

<Adafruit_NeoPixel.h> ist eine der ehemalig verwendete Library und wurde für den Adafruit - LED Streifen genutzt. Es wurden aufgrund der wichtigeren Probleme mit dem Projekt, den Rahmenbedingungen und weil es nicht notwendig war, nicht primär darauf geachtet den Code perfekt und „sauber" zu gestallten.

2.2 Void Setup

Im „void setup()“ finden sich die Funktionen zur Initialisierung aller Komponenten und dem Setzen der Anfangsbedingungen / Einstellungen.

Progammhierarchie:

- initialization:
    - Serial.begin()
    - feedbackinit()
    - imuinit()

Serial.begin() wurde vor allem zum Zweck des Debuggings benötigt, wurde aber trotzdem im Code belassen.

2.2.1 Funktion Feedback - Initialisierung

Die Funktion im genannt code „void function feedbackinit()“ initialisiert die 7-Segmentanzeige, die Pins für die RGB – Diode und ebenfalls noch inkludiert ist die alte Version welche noch LEDs von Adafruit benutzt hat. Außerdem wird das Display auf „init“ und die Farbe der LED / RGB auf Blau gesetzt. Diese Funktion war am einfachsten zu implementieren und verursachte keine Probleme

2.2.2 Funktion IMU-Initialisierung

Diese Funktion initialisiert den IMU-Sensor über die Klasse der IMU – Library und bestimmt die Einstellungen des Sensors. Die Standardeinstellungen wurden größtenteils übernommen, außer der Magnetische Inklination. Diese wurde angepasst an Wien (im Endeffekt keine Auswirkung). Diese Funktion hat die größte Herausforderung zum Implementieren dargestellt. Mehr dazu in Punkt 2.4.1

2.3 Void Loop

Nach erfolgreichem Durchlaufen der Setup Methode beginnt die Endlosschleife der Loop-Methode zu greifen. In der Loop-Methode sind die restlichen drei Funktionen des Produktes implementiert: Lesen, Interpretieren und Schreiben. Im code heißen die „void functions“ anders, die Programhierachie entschlüsselt welche void functions gemeint sind.

Programmhierachie:

- timer() (nicht funktional – Debug)
- lesen() (LESEN)
- reward() (INTERPRETIEREN)
    - orientation() (Bestimmt Sensor Ausrichtung)
        - rewardfunction (ax) (SCHREIBEN A – if rewardmetrix < 0.5)
        - rewardfunction (ay) (SCHREIBEN B – else if rewardmetrix > 0.8)
        - rewardfunction (az) (SCHREIBEN C – else (rewardmetrix < 0.8))

2.3.1 Funktion Lesen

Die Funktion Lesen wird in der „void function lesen()“ implementiert. Diese speichert den ersten Wert des Sensors, als die Ausrichtung des Sensors ab. In der finalen Version des Produktes wurden schlussendlich nur die Beschleunigungssensor-Daten verarbeitet. Die x, y und z Koordinate wird gespeichert, um in der Funktion Interpretation verwendet zu werden. Alle darauffolgende Werte werden nicht mehr explizit abgespeichert, sondern dienen zur Bewertung der Bewegung. Diese Funktion zu implementieren war nicht problematisch, nachdem die Funktionsweise der Library verstanden wurde.

2.3.2 Funktion Interpretation

Diese Funktion war am aufwendigsten zu Implementieren und brauchte am längsten bis sie endlich funktional war. Mehr dazu in Punkt 2.4.2. Grundsätzlich ist die Funktion im code als „void function reward()“ bezeichnet. Sie beinhaltet einen kurzen Algorithmus („void function orientation()“), der selbst erfunden wurde, um zu bestimmen auf welche Achse die Gravitation wirkt. Wenn der Wert einer Variable über ein bestimmtes Threshold geht, kann die Achse zur weiteren Verwendung genutzt werden. Die ersten abgespeicherten Werte des Sensors werden hier angewendet. Wenn die Achse bestimmt wurde, wird der ausgelesen ax, ay oder az Wert an ein switch Statement übergeben und durchläuft die „void function rewardfunction()“. Hier wird der absolute Wert der Sensorachse gegen 1 differenziert. Der resultierende Wert ist so ein Merkmal welche neue Position der Sensor im Vergleich zur Ausgansposition eingenommen hat. Je höher der Wert gegen 1 geht (90 Grad) desto tiefer ist die Kniebeuge.

Entwickelt wurde dieser Algorithmus, indem zuerst die Daten ausgelesen wurden, dann Hintergrundinformation zu der Funktionsweise eines Beschleunigungssensor gesammelt und schließlich mit einer Menge Trial-and-Error so angepasst wurde, bis das eine kontinuierliche Erkennung des Winkels/Auslenkung möglich waren. Warum dies aber genau funktioniert ist mir selbst ein Rätsel, keine vollständige Erklärung ist für mich vorhanden.

2.3.3 Funktion Schreiben

In der Funktion Schreiben wird in der „void function rewardfunction()“, über ein if/elseif/else (Switch) Statement bestimmt welche Anzeige für den jeweilig gemessenen Wert ausgegeben werden soll. Ist die Auslenkung zu gering (unter 0,5) wird ein rotes Licht angezeigt und der interne Zähler für erfolgreiche Wiederholungen erhöht sich nicht. Ist die Auslenkung gut aber noch nicht tief genug (unter 0.8) wird ein gelbes Licht angezeigt und der interne Zähler verändert sich nicht. Ist die Auslenkung hoch genug (über 0.8) dann zählt die Kniebeuge zu den erfolgreichen Wiederholungen der Zähler erhöht sich um 1 und verzögert die Loop schleife um 4 Sekunden, damit bei Ausharren auf 90 Grad nicht kontinuierlich hochzählt wird. Dieser Wert könnte theoretisch noch niedriger gesetzt werden. Es wurde auch der Versuch gestartet über die void function timer ein Zeitfenster von circa 15 Sekunden zu setzen. Sobald der Wert über 0,5 ist, soll dann nur dann einmalige das Erreichen der Schwelle des Wertes über 0.8 festgehalten werden. Dies konnte aber leider aus zeitlichen Gründen und Fehlern/Bugs der algorithmischen Logic nicht umgesetzt werden. Deshalb wird diese Passage im Code auch als Dead Code bezeichnet (void function definescunum). Diese Passage erwies sich als eher problemlos zu implementieren.

2.4 Herausforderungen

2.4.1 Externe Libraries: (Phase 1-2)

Jedes Mal wenn externer Code im eigenen Projekt benutzt wird, kann dies entweder Fluch oder Segen sein. Es gibt Einzelne gut ausgearbeitete und dokumentierte Libraries und es gibt das Gegenteil. Die externe Library (/Klasse) welche in dem Projekt für den Kommunikationsaufbau und das Datenauslesen des MPU9250 zur Anwendung kamen, hat gute Funktionalitäten. Leider sind keine davon zum Zeitpunkt der Einarbeitung ordentlich dokumentiert gewesen. Deshalb hat es signifikante Zeit gebraucht mit der Klasse zu experimentieren und die korrekte Implementierung herauszufinden. Vor allem deswegen, weil keine Erfahrung mit dem Datenprotokoll

I2C vorhanden war. Die tatsächliche Implementierung ist sehr einfach, nur war dies nicht bekannt und es gab immer wieder Probleme. Einerseits durch falsches Anschließen der Komponente, Setzen von falschen Adressen etc.. Es dauerte insgesamt sicher 10 h bis, die Komponente endlich angesprochen werden konnte und dann auch Daten ausgab. In Zuge dessen wurde auch der I2C Error Code und dessen Verhalten auf den Code/Zusammenspiel des Codes mit den Komponenten bekannt. Aufgrund der extensiven Beschäftigung mit der Library war dies eine der wahrscheinlichsten Gründe für das nicht Funktionieren des Produktes in der dritten Phase, zum zweiten Termin. (Sensor wurde nicht richtig erkannt wegen falscher Verkabelung.)

2.4.2 Implementierung des Algorithmus: Bewegungsbewertung (Phase 3)

Es gab leider keine, bzw. wurden keine Anhaltspunkte für dieses Problem gefunden. Im Endeffekt musste alles von Grund auf verstanden und implementiert werden. Es hat sehr viel Hintergrundwissen zu IMU-Sensoren benötigt, um zu verstehen, welche Daten wie ausgelesen werden. Vor allem die Suche nach hilfreichen Quellen hat sich als zeitaufwendig gestaltet. Was zum Erfolg geholfen hat war das auslesen ein Sensor-Achsenwert während einer Bewegung ausgelesen. Anschließend wurde nach erkennbaren Mustern gesucht. Gemeinsam mit dem vorher erarbeiteten Hintergrundwissen, wurde der Algorithmus teils experimentell, teils mit Annahmen, teils mit der Datenauslese realisiert. Im Nachhinein wurde der Algorithmus und dessen Funktionsweise immer weniger verstanden (Es wurde kein Korrektheitsbeweis gemacht).

3 Lessons Learned

3.1 Fehlende Lichtausgabe: RGB-Diode – LED-Streifen

Es ist nicht eindeutig, warum die Lichtausgabe des Produktes fehlerhaft war/ist. Dennoch können Lektionen aus dem Projekt gezogen werden. Wäre das Problem früher erkannt worden oder hätte man eine LED-Einheit in Reserve gehabt, wäre es einfacher zu beheben gewesen. Außerdem ist es in Zukunft wichtig, bei dem Zusammenbau von Komponenten, diese direkt danach ordentlich und ausführlich zu testen. Weiters wäre die Zuweisung einer eindeutigen Zuständigkeit für das Testen und genügend Zeit für diese hilfreich. Außerdem werden in Zukunft vorsorglich mehrere Stückzahlen von zu verbauenden Komponenten besorgt, um Defekte einfacher austauschen zu können.

3.2 Terminkoordination/Verfehlung/ Fehlende Leistung

Das gesamte Projekt wurde im September besprochen und durchgeplant, die Zeiträume zwischen den einzelnen Terminen waren großzügig angelegt um viel Spielraum für Fehler zu lassen. Weiters wurde auch darauf geachtet nicht alles zum Schluss zu erledigen. Warum es trotzdem zu Problemen kam, ist kein Rätsel. Ein Faktor der bestimmt eine Rolle gespielt hat, war das der Projektleiter in der letzten Woche durch einen Notfall (Krebs im näheren Bekanntenkreis) die FH depriorisiert hat und sich nicht mehr um alles kümmerte. Allerdings wurde dieser Zwischenfall offen und rechtzeitig an alle Teilnehmer*innen kommuniziert und von diesen auch zur Kenntnis genommen. Denn dies zeichnete unser Team bisher aus, dass es zu einer äußerst guten Kommunikation zwischen allen Parteien kam. Was allerdings über die gesamte Kooperation (1-3.Semester) auffällig war, dass am Ende des zweiten und dritten Semesters ein Partei des Öfteren, Veränderungen oder Versäumnisse nicht rechtzeitig kommunizierte.

Das Verfehlen von vereinbarten Terminen, ohne scheinbaren Grund (nicht eruierten Grund) und das nicht Erbringen von Leistung kann und soll auch nicht kompensiert werden. Dies ist etwas, was nicht durch die Planung verhindert werden kann. Vor allem dann, wenn trotz eindeutiger Versuche dies zu kompensieren, es nicht möglich gemacht wird, bzw. keine Initiative von dieser Partei gezeigt wird.

Trotzdem hätte eine frühere Kontaktaufnahme (im Rückblick) mit dem Lehrveranstaltungsleiter einige Probleme minimieren bzw. vielleicht auch lösen

können. Leider benötigt es immer Zeit und mehrere Vorkommnisse, um Schlüsse aus den Ereignissen ziehen zu können. In diesem Fall war dies erst am Schluss möglich. Es ist immer darauf zu achten eine klare Kommunikation zwischen allen Parteien aufrechtzuerhalten. Dies wurde auch hier getan.

Anhang 1

Abbildung 10: Ansicht Nr. 1

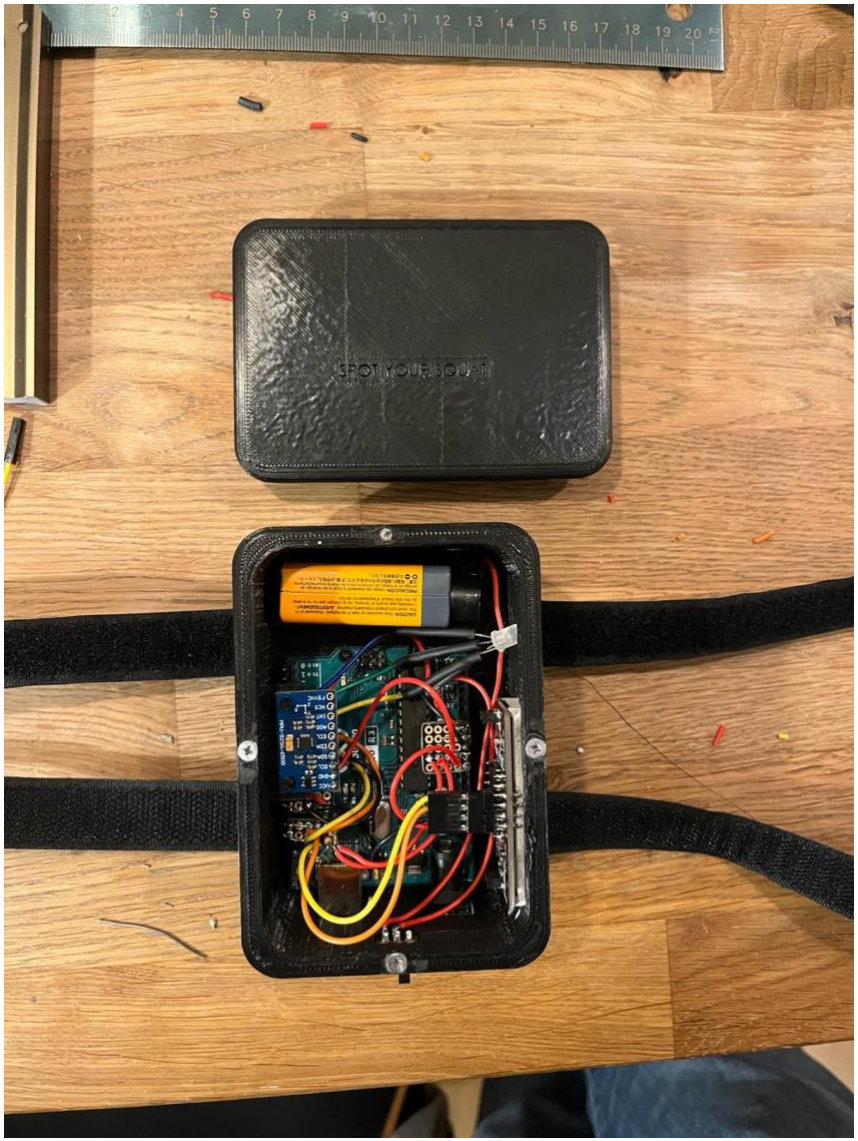

Abbildung 11: Ansicht Nr. 2