Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Méréstechnika és Információs Rendszerek Tanszék

GYORSULÁSMÉRŐ-ALAPÚ VIRTUÁLIS DOBSZETT MEGVALÓSÍTÁSA

Készítette
Bányay Dániel

Konzulens
Dr. Bank Balázs Lajos

2017

TARTALOMJEGYZÉK

HALLGATÓI NYILATKOZAT

Alulírott Bányay Dániel, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy hitelesített felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Kelt: Budapest, 2017. 05. 16.

.....

Bányay Dániel

Összefoglaló

Szakdolgozatom témájának kiválasztásánál a legfőbb motivációm az volt, hogy megvalósítsam egy korábbi projektötletemet. Így a fejlesztés során végig megvolt bennem a lendület, hogy tényleg egy olyan eszközt hozzak létre, amely teljesíti az általam támasztott igényeket, és amelyből később egy piacképes terméket lehet fejleszteni.

Mivel sokféle hangszeren játszom, és a szabadidőm nagy részét is a zenével töltöm, ezért biztos voltam benne, hogy ebben a témakörben találok majd egy olyan területet, ahol még lehet fejleszteni a piacon elérhető eszközökön. Utazás közben is általában zenét hallgatok, és gyakran dobolok az kezeimmel, követve az adott szám ritmusát. Innen jött az ötlet: szeretnék létrehozni egy olyan eszközt, amellyel utazás közben is lehet a zeneszámok ütemére dobolni, miközben a felhasználó hallja a megütött dobok hangját is. Az ötletet továbbgondolva rájöttem, hogy ez az eszköz alkalmas lehet arra, hogy kezdő dobosok utazás közben is tudjanak gyakorolni. Dobolásnál az egyik legnehezebb feladat, hogy a felhasználó megtanuljon különféle ritmusmintákat játszani a kezeivel és a lábaival, miközben tartja a tempót. Ezt a folyamatot függetlenítésnek hívják, és csak sok gyakorlással fejleszthető. Nagy segítség lenne a felhasználónak, ha utazás közben is lehetne gyakorolni az alap ritmusokat, miközben vissza tudja hallgatni a virtuális térben megütött dobokat.

A rendszer 4 hordozható egységet tartalmaz, amelyből 2 db a kezekre húzható kesztyűként, a másik 2 db-ot pedig a cipőre lehet felrakni. Ezen felül tartalmaz még egy központi feldolgozó egységet, amely a hordozható egységeken található gyorsulásmérő szenzorok adatait dolgozza fel. A szakdolgozatom témája a feldolgozó algoritmusok megírása és alkalmazása kész hardveren, így ez a változat még nem fogja teljesíteni azt a követelményt, hogy az eszköz hordozható legyen.

A prototípust egy LPC 1768 fejlesztőkártyán valósítom meg, ez lesz a feldolgozó egység a rendszerben. Ehhez vezetékes kapcsolattal csatlakoznak a gyorsulásmérők, amelyek digitális kimeneti adatait dolgozza fel a kártyán található mikrokontroller. A fejlesztőkártya USB kapcsolaton keresztül MIDI üzeneteket küld egy PC-nek. A számítógépen futó VSTi program feladata, hogy a MIDI üzeneteket hanggá alakítsa és visszajátssza a felhasználónak.

Fontos, hogy a dobolás élménye természetes legyen, ezért ügyelni kell arra, hogy a lehető legkisebb idő teljen el az ütés érzékelése és a hang visszajátszása között. Szintén fontos szempont, hogy a feldolgozó algoritmus képes legyen megkülönböztetni az ütéseket erősség szerint. Ezen felül a rendszernek meg kell tudnia különböztetni az ütéseket aszerint, hogy a virtuális térben hol történtek, így a felhasználó többféle dobot is meg tud szólaltatni, ha változtatja az ütések helyét.

Természetesen egy virtuális eszközön való játék nem adhatja vissza az igazi dobszetten való dobolás élményét, de ezt nem is tartom fő szempontnak. Arra viszont alkalmas, hogy egy kezdő dobos gyakorolni tudjon, illetve hogy utazás közben rájátsszon a kedvenc számaira, ha épp úgy tartja kedve.

Abstract

The primary motivation at selecting the subject of my thesis was realizing an innovation idea. All along the development I had the drive to make a device real that meets my expectations, and also which serves as a base for further development of a marketable product.

I play several kinds of instruments and spend most of my spare time dealing with music. That's the reason why I was sure that I would find a subject for development in this field. I usually listen to music during travelling as well, and I tend to drum with my hands, following the rhythm of the given music. So came the idea that I would like to develop a device that enables playing the rhythm of tracks during travelling with an audible feedback of the virtual drums. When I started to think the idea over, I realized that this device would be appropriate also for practicing drums for beginners during travelling. One of the most difficult parts of drumming is to play different patterns with both hands and legs while keeping the given tempo. This can only be improved with a lot of training. It would be a great help for the users to exercise the basic rhythm patterns during travelling, while listening to actual drum sounds through their earphones.

The system consists of 4 portable units, two of them can be put on the hands as gloves, and the other two units are placed on the shoes. The system includes a central processing unit that processes data from the accelerometer sensors of the portable units. The subject of my thesis is to write the algorithms for data processing and test them on a ready-to-use hardware. This version of the system will not meet the requirement of portability yet.

The prototype is based on a LPC 1768 development board as the processing unit of the system. The accelerometers are connected via wired communication. The digital output of the sensors data are processed by the microcontroller built on the card. The processing unit sends MIDI messages via USB port to a PC. The VSTi program on the computer will convert MIDI messages into sound and play it back to the user.

One of the most important aspects that the user should feel the drumming itself natural, so the delay between the actual hit and the audible sound should be minimised. It is also important that the processing algorithm should be capable of making difference between the drumbeats according their strength. The system should also differentiate the beats based on their location in the virtual space to enable the user play on different virtual drums.

I know it for sure that playing a virtual drum cannot replace the feeling of playing a real drumset, but that is not the point of it. The device is designed for beginners to practice and /or to enjoy drumming with their favourite soundtracks during travelling.

1 Bevezetés

A témám kiválasztásánál a legfőbb célom az volt, hogy megvalósítsak egy olyan dobgyakorlást segítő eszközt, amely akkor még nem létezett a piacon. Az ötlet onnan jött, hogy napjainkban egyre elterjedtebbek a felhasználó mozgásán alapuló játékok (Pl. Wii, PlayStation Move), viszont ezek az eszközök csak otthon használhatóak.

Fiatal korom óta többféle hangszeren zenélek, többek között dobon is játszom. Bár soha nem jártam dobtanárhoz, tudom, hogy az oktatást nem rögtön a dobfelszerelésen kezdik, hanem gumilapon. Itt be lehet gyakorolni az alapritmusokat és a megfelelő technikákat. Ebből látszik, hogy egy kezdőnek nincs szüksége a teljes felszereléshez ahhoz, hogy gyakorolni tudjon. Amikor egy ember úgy dönt, hogy dobolni szeretne tanulni, az egyik legnehezebb lépés a végtagok egymástól való függetlenítése. Ez gyakorlatban azt jelenti, hogy az embernek alapvetően nehezére esik más ritmusmintát játszani a kezével, mint a lábával. A legelső feladat a kezek lábaktól való független mozgatása, és ezt lehet fokozni odáig, amit csak a legprofibb dobosok tudnak: a négy végtag független egymástól, képes különféle ritmusokat játszani úgy, hogy közben a tempó megmarad. Ez a szint nagyon sok koncentrációt és gyakorlást igényel.

Jelenleg nincs a forgalomban a piacon olyan eszköz, amellyel utazás közben fejleszteni lehetne ezeket a képességet. Amikor elterveztem, hogy milyen igényeknek kell megfelelnie az általam fejlesztett eszköznek, nem az lebegett a szemem előtt, hogy ez a hordozható eszköz helyettesítené a dobszetten való gyakorlást, hanem hogy a kezdőknek olyan lehetőséget biztosítson, amellyel hasznosan tudják felhasználni az utazással eltöltött idejüket és fejleszteni tudják a doboláshoz szükséges képességeiket.

A cél tehát egy olyan rendszert létrehozása, amely 4 hordozható egységből és egy feldolgozó központi egységből áll. A 4 hordozható egységből 2 kerül a kezekre, 2 pedig a lábakra. Az egységek a rajtuk lévő inerciális mozgásérzékelő szenzorokkal figyelik az eszközök orientációját és gyorsulását, és továbbküldik az adatokat a feldolgozó egységnek. A felhasználó fejhallgatón vagy hangszórón tudja visszahallgatni a megszólaltatott dobok hangját. A szenzorokból kinyert adatokból a feldolgozó egység képes meghatározni a hordozható egységek pozícióját, és ez lehetővé teszi, hogy a felhasználó nem csak egyféle dobon tud játszani, hanem a kezek virtuális térben való pozíciójától függően képes legyen megszólaltatni a dobszett több elemét.

A végső cél az, hogy a 4 egység vezeték nélkül kommunikáljon a központi egységgel, amely a felhasználó okostelefonja lenne. A BSc szakdolgozatom még nem ezt a megoldást alkalmazza. A főbb területek, amelyekre nagyobb hangsúlyt fektetek a dolgozat keretein belül, az a szenzor adatok feldolgozása, és ebből a pozíció és az ütés erősségének meghatározása, majd az eszköz prototípusának elkészítése mbed NXP LPC 1768 fejlesztőkártyán, vezetékes kapcsolattal.

2 Piacon forgalomban lévő megoldások

Amikor kiválasztottam a szakdolgozatom témáját, még nem volt a piacon az én ötletemhez hasonló eszköz. A következőkben bemutatott első két eszköz piacon már elérhető termék, a harmadik egyelőre csak koncepció, amely a kickstarter.com közösségi finanszírozású projektekkel foglalkozó honlapon elérhető, mint fejlesztés alatt álló termék. Ezeket a megoldásokat fogom összehasonlítani ebben a fejezetben.

2.1 RockJam Electronic Roll Up MIDI Drum Kit

A terméknek nincsen honlapja, így a róla elérhető információkat egy internetes áruház termékleírásából tudtam megszerezni. Ez alapján az eszköz 9 drum padet tartalmaz, amelyeket igazi vagy játék dobverővel lehet ütni, illetve 2 pedált. Könnyen szállítható, mivel kis méretűre össze lehet hajtogatni. USB-n keresztül PC-hez kapcsolható, MIDI adatot küld ki magából. Ez az eszköz azonban nem teljesíti azt az általam támasztott követelményt, hogy utazás közben is használható legyen.

Ár: 60$

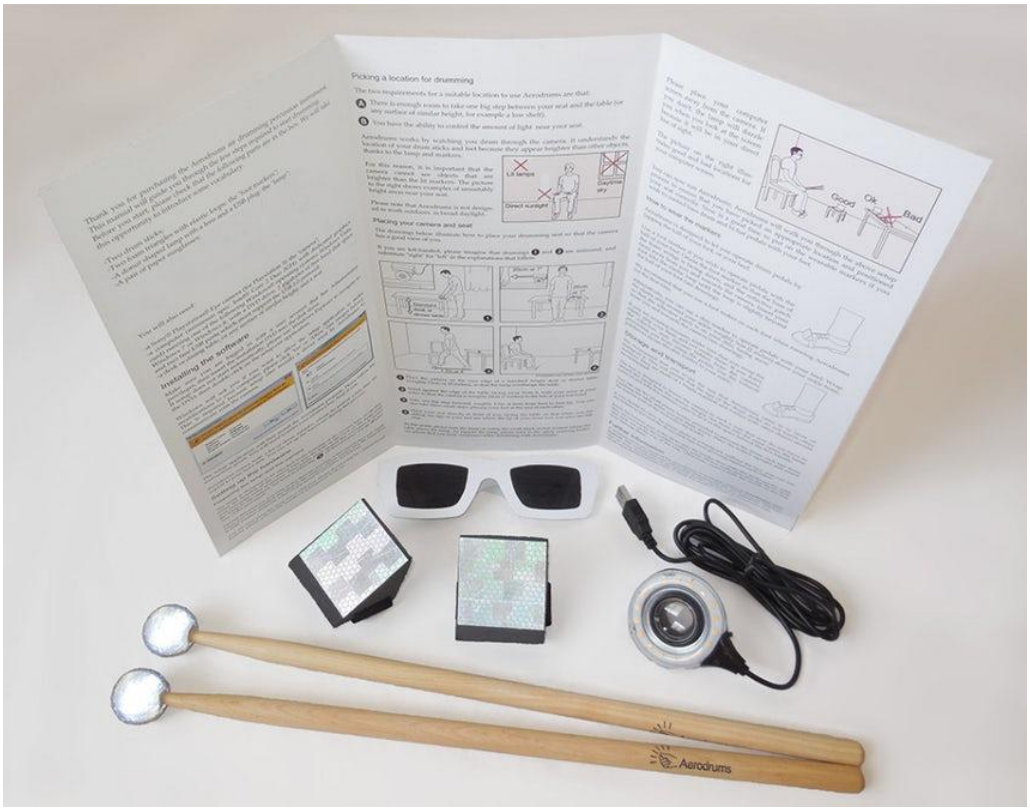
2.1. ábra: Rockjam hordozható dobszett

2.2 Aerodrums

A második termék már közelebb áll az én elképzelésemhez. Nem kell hozzá felület, a felhasználó a levegőben játszik. A végtagokra fényvisszaverő felületek vannak ráapplikálva, a kamera az ezekről való visszaverődésekből számolja ki a végtagok aktuális helyzetét. Ez a megoldás nem használható zárt szobán kívül, mivel a kamerának kell egy minimális távolság ahonnan látja az összes visszaverő felületet, illetve napfényben szignifikánsan megnő a hamis észlelések száma. Előnye, hogy az optikai érzékelés útján centiméter pontosan képes meghatározni a dobverők és a lábak pozícióját.

Honlap: http://aerodrums.com/aerodrums-product-page/

Ár: 200$



2.2. ábra: Aerodrums virtuális dobszett

2.3 Freedrum

A Freedrum koncepciója majdnem teljesen megegyezik azzal a gondolatmenettel, amelyet én követtem. A kickstarter.com honlapon futó közösségi finanszírozású kampányra egy online hirdetés hívta fel a figyelmem. Állításuk szerint ez az eszköz át fogja formálni a virtuális dobok piacát. Azóta sikerült elérni a kitűzött célösszeget, a gyártás 2017 augusztusában fog kezdődni. A 4 végtagra 4 eszközt használnak, amelyek akkumulátorral rendelkeznek, és vezeték nélkül Bluetooth MIDI szabványon keresztül

kommunikálnak egy telefonos applikációval. Az applikáció tetszőleges MIDI lejátszó lehet, amely átalakítja a jeleket megfelelő hangokká.

Különbség az én elképzelésemhez képest, hogy én a céleszközt egy kesztyűként képzeltem el, illetve hogy a két leggyakrabban használt dobot (pergő, lábcin) a két térden való ütéssel lehet megszólaltatni, a Freedrum pedig kifejezetten írja, hogy levegőben való ütésekre van bekalibrálva, és ajánlott a dobverők használata. Ez véleményem szerint erősen korlátozza az eszköz szabad felhasználását utazás közben.

Honlap: https://www.freedrum.rocks/

Ár: 160$

2.3. ábra: Freedrum inerciális szenzor alapú virtuális dobszett

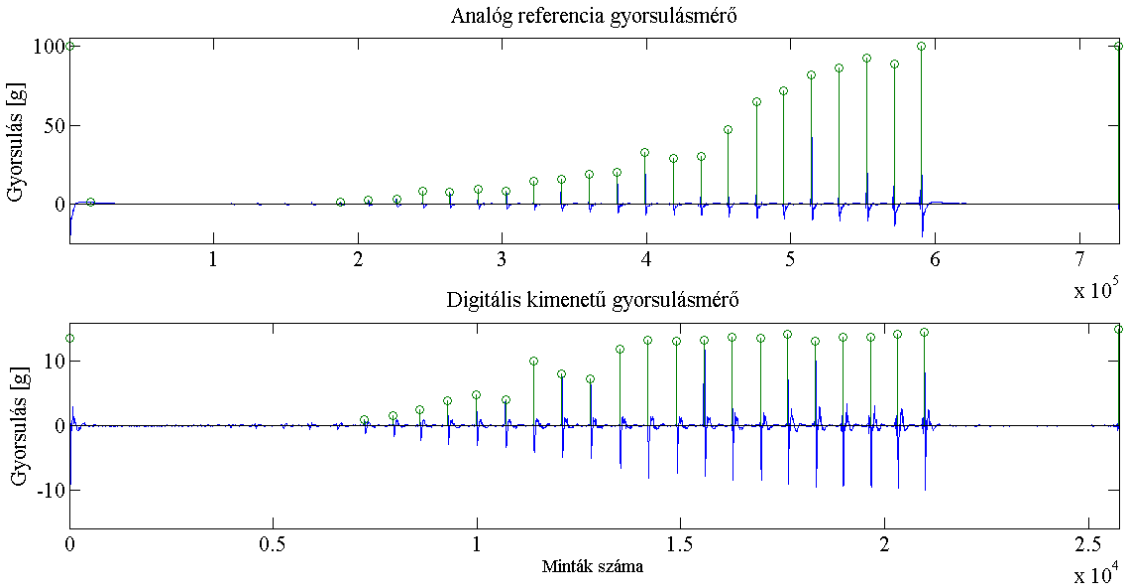
3 Előzetes vizsgálatok

A fejlesztés legelején, amikor még nem volt biztos, hogy az eszköz megvalósítható, egy Pololu MMA7341LC analóg gyorsulásmérő kimeneti adatait rögzítettem hangkártyán keresztül. Ezután kirajzoltattam MATLAB környezetben, és így vizsgáltam a jelalakot. Az IC méréstartománya ±9 g volt, és az ábrákból hamar kiderült, hogy az erősebb ütések detektálására ez a tartomány kevés.

Az adatokat digitalizálni kell a MATLAB tesztek elvégzéséhez és később a jel mikrokontrollerrel való feldolgozásához is, ezért úgy döntöttem, hogy áttérek a digitális gyorsulásmérő használatára. A LIS3DH háromtengelyű digitális kimenetű gyorsulásmérő IC megismerése, regisztereinek beállítása és a megfelelő adatok kinyerése jelentős munkabefektetést igényelt, mivel korábban nem foglalkoztam mélyebben ezzel a témakörrel.

Ezután labor környezetben végzett mérésekkel vizsgáltam a dobütések jelalakjait. Elsődleges cél volt az olcsó (~1500 Ft) digitális gyorsulásmérő adatainak összehasonlítása a DSP laborban található Brüel&Kjar referencia gyorsulásmérővel. A teszt mérési elrendezése a **Függelék 5. ábráján** látható. Másodlagos cél pedig jelalakban lévő mintázatok felismerése, amelyek észlelésére később algoritmus írható.

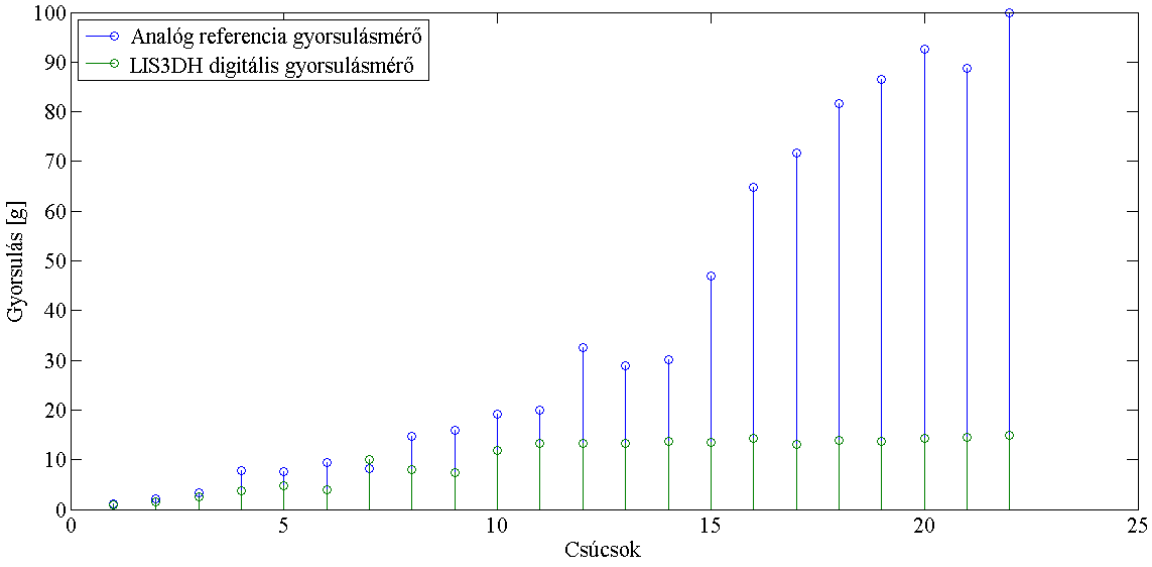
3.1 Összehasonlítás referencia gyorsulásmérővel



3.1. ábra: Analóg referencia gyorsulásmérő és LIS3DH digitális consumer gyorsulásmérő Z irányú adatainak összehasonlítása

Hogy a két eszközről felvett ütések megfeleltethetőek legyenek egymásnak, a mérés elején és végén található egy-egy erősebb ütés (**3.1 ábra**). Ezeket az ütéseket arra használtam, hogy szinkronba tudjam állítani a két csatornát, amelyek a nagyságrendekben eltérő mintavételi frekvencia miatt nem egyenlő hosszúak. A többi csúcsot szintén saját szkripttel kerestem meg, amely végül alapjául szolgált a valós idejű szintérzékelő algoritmusnak is, amelyről részletesen a 3.3 pontban írok.

A LIS3DH méréshatára ±16 g, amely nem sokkal több, mint az előző IC ±9 g méréshatára, de ebben az árkategóriában ez a legszélesebb, ami elérhető. A szemléltetés kedvéért közös ábrán (**3.2 ábra**) is megjelenítettem az eredményt, így jól látszik, hogy 16 g felett a digitális gyorsulásmérő "telítődik", nem képes különbséget tenni az ezen érték feletti ütések között. Érdekes tény még, hogy a legerősebb ütés eléri a 100 g-t is. Eleinte törekedtem rá, hogy a 16 g feletti gyorsulás értékeket is meg tudjuk különböztetni, vagy az ütés görbe alatti területből, vagy az ütés előtti negatív gyorsulás maximális amplitúdójából. Azonban a tesztek során kiderült, hogy a 16 g-nél nagyobb térden/combon történő ütések már fájdalommal járnak, így nem jelentene hozzáadott értéket a kész termékhez ezeknek a csúcsok megkülönböztetése.



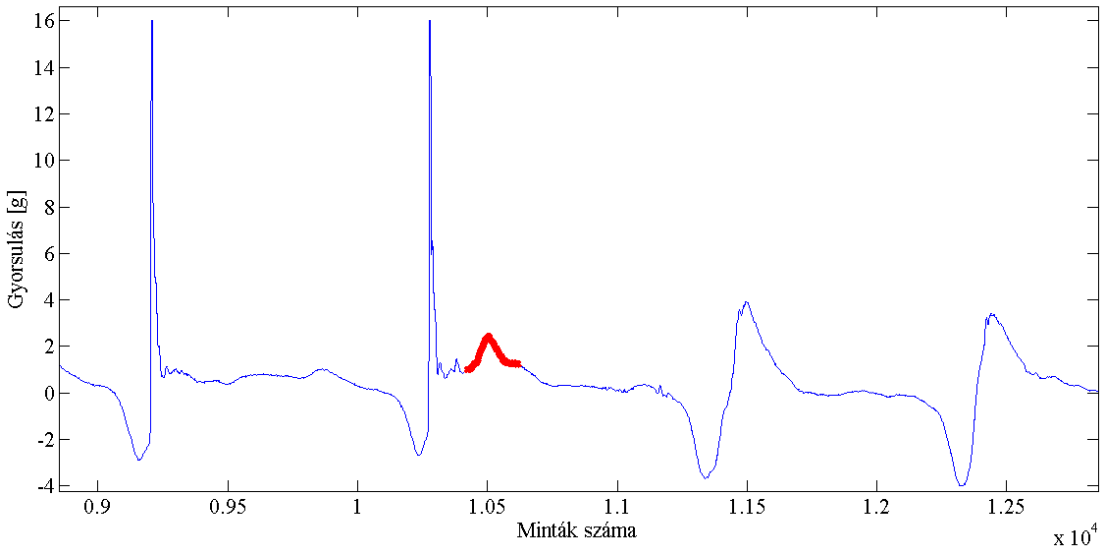
3.2. ábra: Analóg és digitális gyorsulásmérő csúcsainak összehasonlítása

3.2 Pozíció számítása gyorsulási adatokból

A következő tesztek arra irányultak, hogy miként tudom majd a későbbiekben megkülönböztetni a különböző dobokat a virtuális térben. Először a gyorsulás adatokat vizsgáltam egy olyan mozgásnál, amelynél a felhasználó egy térden való pergődob

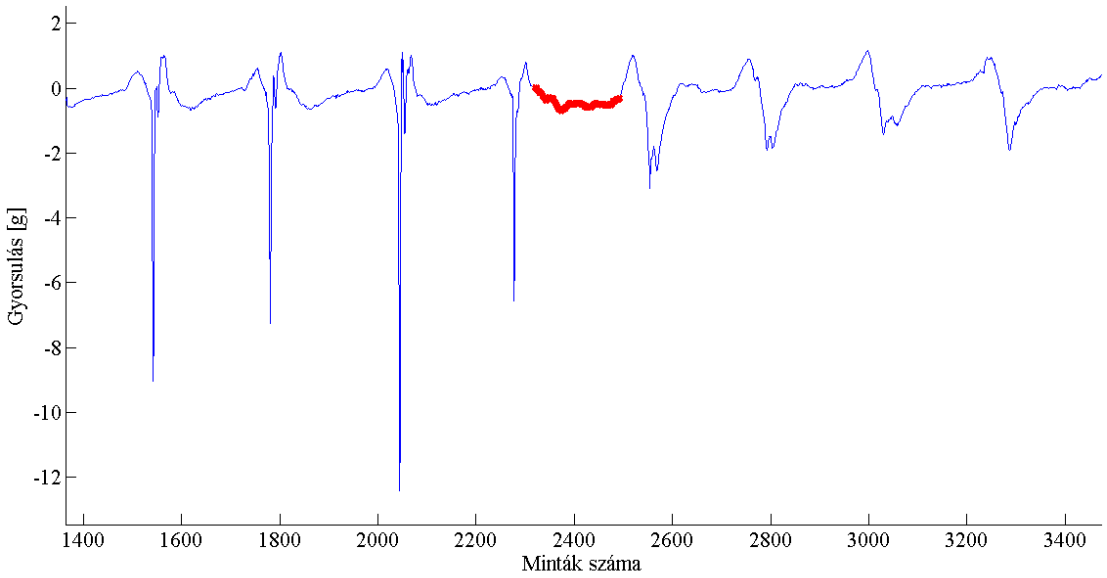
megütése után a felső cintányérokhoz nyúl. A továbbiakban referenciairányok megadásánál ezt a helyzetet fogom alappozíciónak venni. Ehhez képest a pozitív Z irány vektora a Föld közepe fele mutat, a pozitív X irány jobbra, a pozitív Y irány pedig előre.

Ennél a mozdulatnál a gyorsulás a Z tengely irányában változik a legtöbbet, ezért a **3.3 ábrán** csak a Z tengelyt ábrázoltam. A teszt során kiderült, hogy az ütés gyorsulási görbéje sokkal nagyobb amplitúdójú, mint a pozícióváltásé. A váltást a **3.3 ábrán** pirossal jelöltem. Mivel az ütésekhez képest nehéz detektálni az ilyen kis változásokat, ezért nem lehetséges hatékony detektáló algoritmust írni rá.



3.3. ábra: Pozíció detektálása gyorsulási adatokból, Z tengelyű mozgás

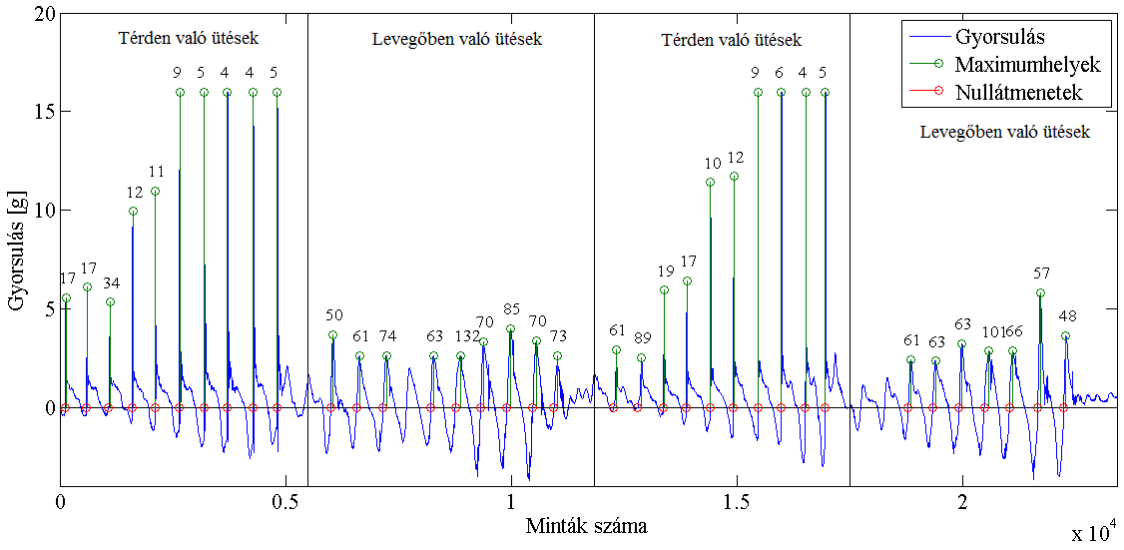
A következő mérés ugyanezen az elven alapult, azzal a különbséggel, hogy egy oldal irányban történő ütést vizsgáltam. Ez a mozgás megfeleltethető egy álló tam vagy egy kísérő cintányér megütésének. Így arra számítottam mérés előtt, hogy az X irányú mozgásban jól megfigyelhető lesz a pozícióváltás.



3.4. ábra: Pozíció detektálása gyorsulási adatokból, X tengelyű mozgás

A **3.4 ábrán** látható eredmény nem az előre vártakat hozta. Először is az tűnt fel, hogy bár a térden való ütések Z irányúak, a gyorsulás akkora, hogy az X (és az Y) tengely is majdnem a maximális 16 g kitérést mutatta. A pirossal jelölt szakasz megint az átmeneti rész a térden ütött pergődob és a levegőben ütött cintányér/tam között. A görbe még annyira se jellegzetes, mint a **3.3 ábránál**, így erre se lehet hatékony algoritmust írni.

Egy dolog viszont mindkét vizsgálatnál feltűnt. A térden való ütések és a levegőben történő ütések görbéi szemre jól elkülönülnek. Amennyiben megelégszünk azzal, hogy csak 2 pozíciót különböztetünk meg az alapján, hogy térden vagy levegőben történt, érdemes megvizsgálni a kétféle jelalakot. A **3.5 ábrán** egymás után láthatóak különböző amplitúdójú térden elvégzett ütések, majd levegőben való ütések, és ugyanez még egyszer megismételve.



3.5. ábra: ütések megkülönböztetése hullámforma szerint

A MATLAB-ban írt algoritmus megkeresi a csúcsokat, majd attól visszafelé számol addig, ameddig nullátmenetet nem ér. A csúcs és a nullátmenet között mintavételezett minták száma a csúcsok felett olvasható, így megkapjuk az ütés szélességének a felét. Azért nem érdemes a teljes szélességet nézni, mert valós idejű megoldásnál a csúcs elérésének pillanatához minél közelebb kell visszajátszanunk a felhasználónak a megfelelő dobütés hangját, és a csúcson túli minták számolásával további késleltetést vinnénk a rendszerbe.

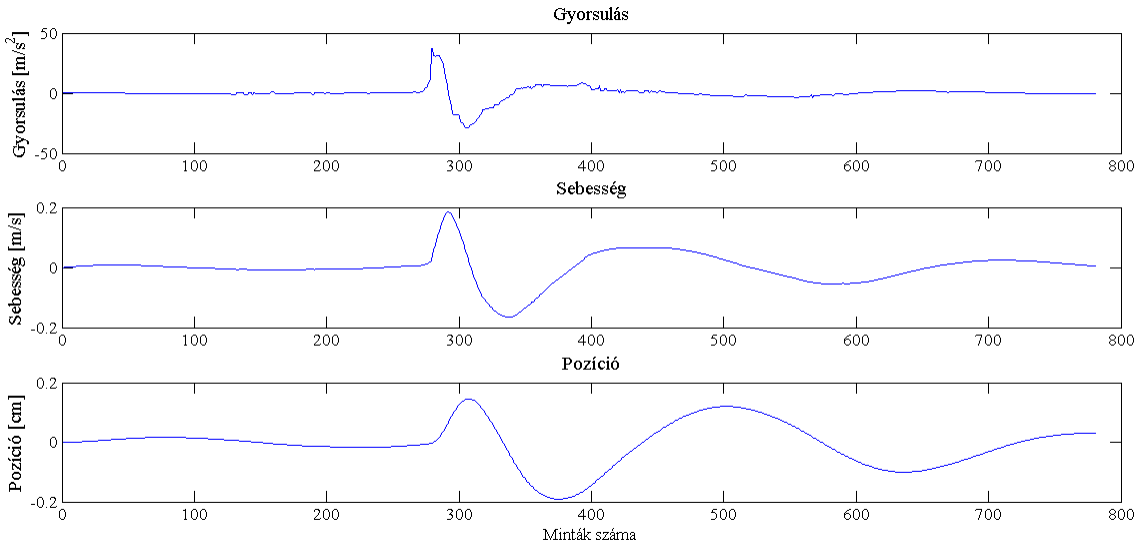
A kép elemzése után az derül ki, hogy a nagyobb ütéseknél a nullátmenet és a csúcs közötti eltelt idő nagyon rövid, jól elkülöníthető a levegőben történő ütésektől. Azonban a kisebb ütések félszélessége nagyon hasonló a levegőben való ütésekéhez: vegyük példának az első levegőben való ütések blokk első ütését 50 minta szélességgel, és a második térden való ütések blokk első ütését 61 minta szélességgel. Így ez a módszer önmagában nem használható, legfeljebb kiegészítő algoritmusként.

Következő teszt a gyorsulási adatokból kettős integrálással megkapott pozíciószámítás volt. A sebesség és a pozíció meghatározásánál szivárgó integrátort alkalmaztam, hogy a Föld gravitációja által létrehozott DC offset ne okozzon “elszállást”. A mellékelt MATLAB kódban a 0.99-el való szorzás a szivárgó intergrátor erősítését jelenti, az *Fs* pedig a gyorsulásmérő mintavételi frekvenciáját.

```
for i=2:1:length(acceleration)
    velocity(i)=0.99*velocity(i-1)+(acceleration(i-1)*1/Fs);
end

for i=2:1:length(velocity)
    position(i)=0.99*position(i-1)+(velocity(i-1)*1/Fs);
end
```

Ahogy az a **3.6 ábrán** is látszik, a szivárgó integrátor aluláteresztő szűrőként is funkcionál, kisimítja a gyorsulási adatok kisebb rendellenességeit. A kétszeres integrálás megváltoztatta a függvény alakját (ütés utáni görbe megnövekedett), és nem a valóságnak megfelelő számokat adott vissza.

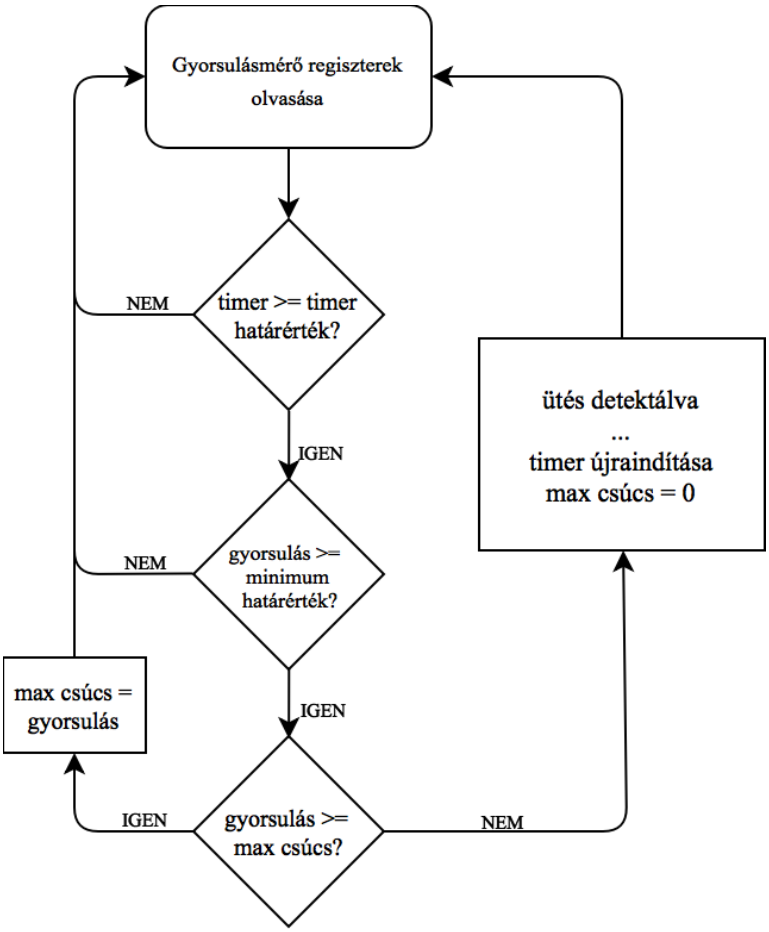


3.6. ábra: Pozíció számítása szivárgó integrátorral

A teszt során elég információt gyűjtöttem ahhoz, hogy belássam, hogy az én megoldásomhoz más módon kell kiszámolni a végtagok aktuális helyzetét. A megoldást a **4. fejezetben** írom le részletesen.

3.3 Szintérzékelő algoritmus

A mozdulatfelismerő algoritmus fejlesztése során az első lépés a szintérzékenység megvalósítása volt. A valós dobolási élményt sokkal közelebb lehet hozni azzal, ha az eszköz egy gyenge ütésre válaszként egy gyengén megütött dob hangját, míg egy erősebb ütésnél az erősebben megütött hangját adja vissza. Ehhez jóval többre van szükség, mint egy egyszerű küszöbszint és aktuális gyorsulási érték közötti összehasonlításra. Az algoritmus működésének diagramja a **3.7 ábrán** látható.



3.7. ábra: Szintérzékelő algoritmus diagramja

A függvény tartalmaz egy időmérőt, amely az előző detektált ütés óta eltelt időt méri. Ez azért fontos, mert a csúcsok utáni szakaszok nem mindig szigorúan monoton csökkenőek. Ha az ütés nagy erejű volt, akkor gyakran előfordulnak kettős csúcsok, nekünk ezt azonban egy ütésnek kell érzékelni. Emiatt megadtam egy minimális időtartományt, amely alatt az algoritmus nem detektálhat csúcsot. Ezt az értéket tapasztalati úton választottam ki, az általam reálisnak tartott leggyorsabb ritmust véve alapul.

A gyorsulásmérő regisztereinek kiolvasása után a program egy összetett IF állításhoz érkezik. Ha az előző ütés óta eltelt idő megfelelő, akkor az algoritmus megvizsgálja, hogy a gyorsulás adat átlépi-e a minimális küszöbszintet. Ha ez megtörtént, akkor két opció mehet végbe: az első, hogy az aktuális gyorsulási érték nagyobb, mint az előző maximum érték. Ez esetben a frissen beolvasott maximális értékünk lesz az új maximális érték.

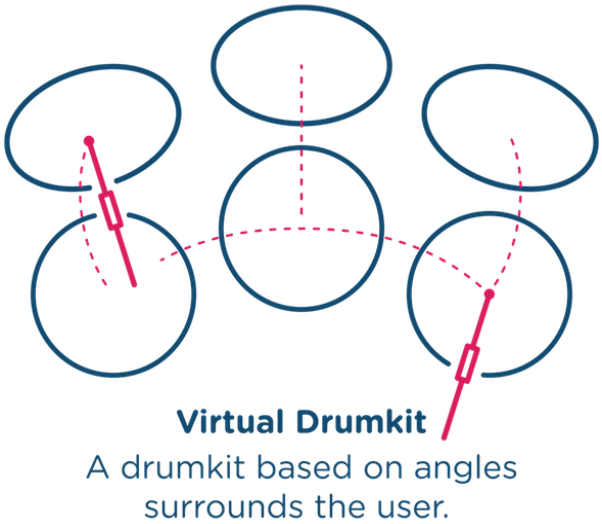
Ha az új érték viszont kisebb, mint a maximális érték, akkor az előző érték volt a legnagyobb amplitúdóval rendelkező adat ennél a dobütésnél. Ez azt jelenti, hogy dobütést detektáltunk. A ”…” rész a szövegdobozban azt jelzi, hogy ezen az ágon még több esemény történik, a részletes leírás a **Függelék 6. ábráján** látható. A timer és a segédváltozók nullázódnak, és a friss gyorsulásmérő adatok újból beolvasásra kerülnek.

4 Felhasznált külső algoritmusok

4.1 Orientáció meghatározás MARG szenzorral

Az előző fejezetben részletezett okokból kiderült számomra, hogy egyedül a gyorsulásmérő adataiból pozíciót számolni az én általam fejlesztett eszköz számára nem lehetséges. Emiatt új koncepció után kellett néznem. A megoldást a korábban említett, az internetes Kickstarter kampányban szereplő félkész eszköz leírásában találtam meg. Bár a szöveg nem említ semmilyen technikai részletet, egy lentebb mellékelt képről kiderült számomra, hogy a dobverők pozícióját nem az elmozdulásból határozzák meg, hanem az eszköz relatív orientációjából. Ha sikerül meghatározni a kezünk vízszintes és a függőleges szögelfordulását egy referenciaponthoz képest, akkor egyértelműen meghatározhatjuk a pozícióját is.

A korábban alkalmazott LIS3DH IC-t emiatt egy másik IC-re kellett cserélni, amely a gyorsulásmérőn kívül tartalmaz giroszkópot, amely a szögelfordulást érzékeli, illetve magnetométert, ami a mágneses teret. Az így rendelkezésre álló 9 szabadságfokból már meghatározhatók azok az adatok, amelyek a pontos orientáció érzékeléséhez szükségesek, melyből kiszámolható az adott végtag pozíciója. Az új IC ST LSM9DS0 inerciális modul, melynek gyártója és regiszter felépítése megegyezik a korábban használt LIS3DH-jéval, tehát a regisztercímek újradefiniálása után gond nélkül használhatóak vele a korábban megírt algoritmusok.



4.1. ábra: Szög alapú pozíció meghatározás, https://www.freedrum.rocks/

A **MARG** a **M**agnetic **A**ngular **R**ate és **G**ravitiaion szavakból összeállított betűszó. A MARG szenzorokat használó algoritmusok a gyorsulásmérő adataiból kiszámolnak egy relatív gravitációs vektort a mérési síkhoz képest (esetünkben a szenzor vízszintes helyzetéhez képest). Nagy dinamikájú mozdulatoknál, mint amilyen a dobütés is, nem lehet egyedül a gravitációs vektorral számolni, mivel az a mozgás során folyamatosan változik. A mágneses szenzor érzéketlen a dinamikai változásokra. A giroszkóp pedig kiegészítő szenzorként funkcionál, amely a nagyfrekvenciájú változásokat érzékeli [1]. Az így kapott 9 szabadságfokból számomra megfelelő pontossággal meghatározható az eszköz orientációja.

Ezek után a következő feladat a megfelelő algoritmus megtalálása volt. Az interneten először az Android eszközök által használt módszert találtam meg. Az Android API-ban [2] lévő elfordulást számoló függvények (pl. `getAngleChange()`) leírásai arra utalnak, hogy az algoritmus rotációs mátrixokkal operál, illetve csak mágneses és gyorsulás adatokat használ, szögelfordulást nem.

Tesztelés után kiderült, hogy számomra nem alkalmas, mivel nem kompenzálja megfelelően a magas dinamikájú mozgásokból fakadó gravitációs vektor változásokat. Mivel az én eszközömben a gyorsulás folyamatosan dinamikusan változik 1 g és 16 g között, ezért új algoritmus után kellett néznem.

4.2 A Madgwick algoritmus

A megoldást Sebastian Madgwick AHRS (Attitude and Heading Reference System) algoritmusában találtam meg. A továbbiakban röviden bemutatom ennek működését és felhasználását a saját megvalósításomban. Az algoritmus alapja a kvaterniók elméletére épül, melyet röviden összefoglalok a következő bekezdésben. Bár a matematikai háttér megértése nem volt szükséges az algoritmus használatához, én fontosnak tartottam, hogy legalább az alapokról legyen fogalmam, így magabiztosabban tudom használni az algoritmushoz mellékelt MATLAB és C kódot, illetve módosítani tudom a saját felhasználásomhoz.

Definíció: $q = a + bi + cj + dk$ alakú formális kifejezésekből, az úgynevezett kvaterniókból áll, ahol a,b,c,d valós számok, i,j,k pedig a Q kvaterniócsoport elemei, ahol $i^2 = j^2 = k^2 = ijk = -1$ [3].

A kvaterniók a komplex számok 4 dimenzióra való kiterjesztése, amellyel kifejezhető egy merev test helyzete a 3 dimenziós térben. Segítségükkel könnyen leírhatóak pont és tengely körüli forgatások, amelyek a valós számok körében csak bonyolult mátrixok szorzásával tehetőek meg.

Egy valós idejű rendszerben kritikus pont az algoritmusok futási ideje. A Madgwick algoritmus legújabb C nyelvű implementációja optimalizálva lett CPU felhasználás szempontjából, ezért alkalmas arra, hogy 10-50 Hz-el meghívva kiszámolja a felhasználó végtagjainak pozícióját, és közben ne befolyásolja a szintérzékelő algoritmus működését. 183 szorzást, 64 összeadást, 40 kivonást, 24 osztást és 5 négyzetgyököt tartalmaz [4].

Az algoritmushoz mellékelt C forrásfile bemenete a 9 adat, amely az ICből érkezik (3 tengelyű gyorsulás, 3 tengelyű mágneses és 3 tengelyű giroszkóp adat), kimenete kvaternió. Ebből még meg kell határozni az Euler-szögeket, hogy alkalmazni tudjam őket pozíció számítására. A szögeket a **4.1 egyenletek** adják meg, ahol a MadgwickAHRS algoritmus kimenete Q = (q1,q2,q3,q4). Az `Atan2(x,y)` függvény abban különbözik az `atan(x,y)` függvénytől, hogy megvizsgálja a megadott x és y koordinátákat, és ezek előjelétől függően adja vissza a kiszámolt szög előjelét [5].

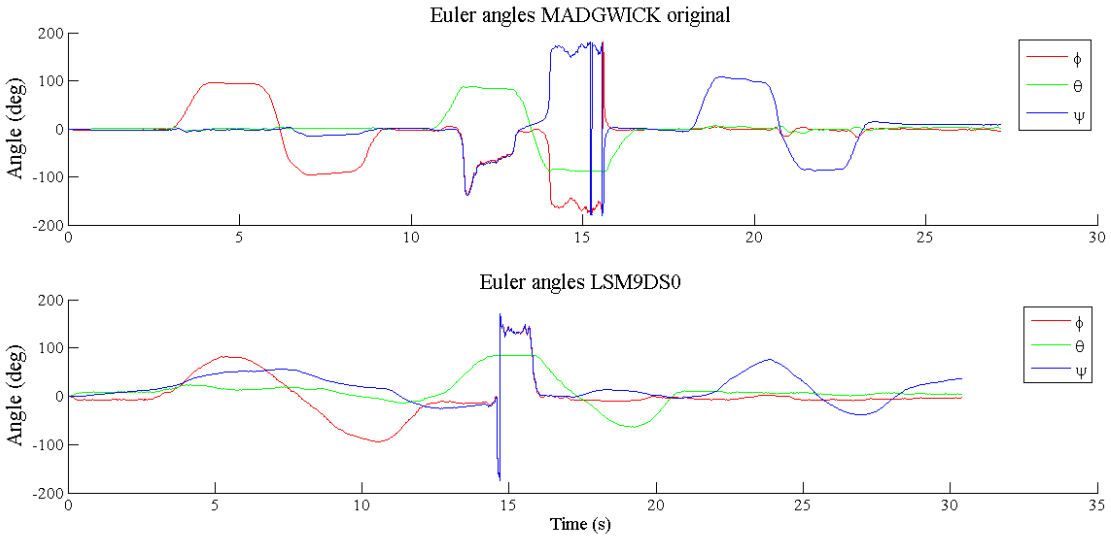
$$\psi = \mathrm{Atan2}\left(2q_2q_3 - 2q_1q_4, 2q_1^2 + 2q_2^2 - 1\right) \tag{7}$$

$$\theta = -\sin^{-1}\left(2q_2q_4 + 2q_1q_3\right) \tag{8}$$

$$\phi = \mathrm{Atan2}\left(2q_3q_4 - 2q_1q_2, 2q_1^2 + 2q_4^2 - 1\right) \tag{9}$$

4.1. egyenlet: Euler-szögekre való áttérés

Ezen összefüggések lekódolása után a Madgwick algoritmus már számomra megfelelő szögeket adott ki. A **4.2 ábrán** a felső grafikonon a Madgwick MATLAB kódhoz mellékelt tesztadatokból futtatott MATLAB algoritmus eredménye, alatta pedig a mikrokontrolleren futó Madgwick AHRS algoritmus és az általam hozzáírt `calculateEulerAngles()` függvény soros porton kiolvasott eredménye látható. A teszt során ugyanazt a mozdulatsort végeztem el, amit a Madgwick tanulmányban is leírtak: lassú előre majd hátra döntés, ez után baloldalra majd jobb oldalra billentés, végül vízszintes tengelyű forgatás. Mindegyik mozgatás -90° és +90° között történt. Mivel a felső ábrán lévő teszt 3 tengelyű mozgató keretben készült, ezért sokkal egyenletesebb, mint az általam kézzel véghezvitt mozgás. De a lényeg így is látszik, a két ábra nagyon hasonló, tehát az algoritmus működik valós időben is.



4.2. ábra: Madgwick algoritmus működése teszt adatokon és saját adatokon [10]

Mindkét ábrán 15 másodpercnél anomália látható a *phi* (X tengely körüli elfordulás) és a *pszi* (Z tengely körüli elfordulás) reprezentációjában. Ezt a hibát *Gimbal lock*-nak nevezik. Akkor jön létre, ha a *theta* szög (Y tengely körüli elfordulás) túllépi a ±90°-ot. Ez egy szingularitás miatt jön létre, amely az Euler szögekre való áttérés során kerül be a rendszerbe. Mivel a virtuális dobon végzett ütések nem vagy csak minimálisan mozdulnak el *theta* irányban, ezért ez a probléma nem kerül elő az én felhasználásomban.

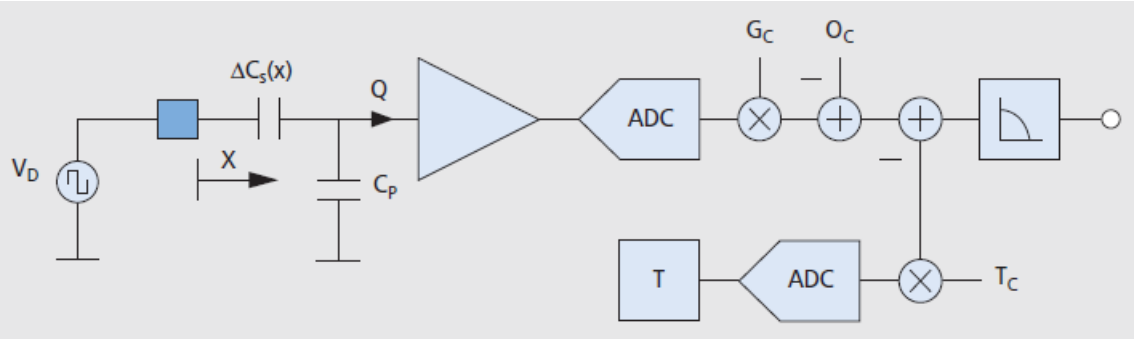
Magának a Madgwick algoritmusnak szüksége van egy kis időre, amíg az algoritmus által számolt orientáció eléri a tényleges orientációt. A kalibráció kiértékelést a felhasználóra bíztam. Sebastian Madgwick megvalósításban ez úgy nézett ki, hogy a számítógép képernyőjének jobb oldalán volt egy test a kívánt orientációban, bal oldalt pedig másik test az algoritmus által számolt orientációban [6]. A két test orientációja folyamatosan közelített egymáshoz. A kalibráció sikeres volt, ha a folyamat végén a két test orientációja megegyezett. Ez az én megoldásomban ez úgy történik, hogy a program folyamatosan kiírja a képernyőre az Euler-szögeket, és ha a felhasználó úgy látja, hogy a kalibrációs folyamat véget ért, tehát az értékek már nem változnak jelentős mértékben, akkor egy ütéssel jelzi a program számára, hogy a kalibráció befejeződött.

5 LSM9DS0 inerciális szenzor

Bár a feladatkiírásban még ST LIS3DH gyorsulásmérő szerepel, fejlesztés közben kiderült, hogy a pozíció számításhoz ez az IC nem megfelelő, így a kézi egységnél áttértem az ST LSM9DS0 inerciális modulra. Mivel ugyanaz a gyártó, a regiszterek felépítése nagyjából megfelel egymásnak. Az LSM9DS0 nem csak gyorsulásmérőt tartalmaz, hanem mágneses szenzort és giroszkópot is, ezért egy fokkal bonyolultabb a kezelése. Ebben a fejezetben ennek a chipnek a működését fogom ismertetni.

5.1 Áttekintés a MEMS eszközök működéséről

A MEMS (Microelectromechanical systems) szenzorok működése egyszerűen modellezhető egy mechanikai keretben felfüggesztett *m* tömegű testtel és a függesztésre használt *km* rugóállandójú rugóval. Gyorsulásmérő esetén a testre ható erő, giroszkóp estén a Coriolis-erő hozza létre a rugó megnyúlását ebben a rendszerben [7]. A mechanikai keret kondenzátor fegyverzeteként is funkcionál. Ha a test helyzete megváltozik a kereten belül, akkor megváltozik a rendszer kapacitása. A kapacitásváltozás arányos a feszültség változásával, melyet számszerűsíteni lehet egy analóg/digitális átalakítás után. Az LSM9DS0 (és a többi digitális kimenetű MEMS eszköz is) ezeket a számokat küldi tovább a kiválasztott kommunikációs módon keresztül.



5.1. ábra: MEMS eszköz sematikus ábrája [7]

5.2 Az LSM9DS0 tulajdonságai és regiszter beállításai

Az LSM9DS0 iNemo inerciális modul az ST Microelentronics SiP (System-in-Package) megoldása, amely 3 szabadságfokú gyorsulásmérőt, 3 szabadságfokú szögsebesség szenzort és 3 szabadságfokú mágneses szenzort tartalmaz [8]. A System-in-Package azt

jelenti, hogy a tokozáson belül több szilíciumchipet is magába foglal, illetve tartalmazza a működéshez szükséges passzív és aktív alkatrészeket is.

Az adatok kiolvasása I2C és SPI protokollokon keresztül lehetséges. A rendszer 2 programozható kimenet lábat is magába foglal, amelyek különböző események bekövetkezése esetén interruptok generálására használhatóak.

A fejlesztés során jelentős időt fordítottam a regiszterek számomra megfelelő értékeinek megtalálására és tesztelésére, emiatt fontosnak tartom ezen beállítások közlését a hozzájuk tartozó magyarázattal.

5.2.1 A giroszkóp subchip regiszter beállításai

WHO_AM_I_G:

Read-only regiszter. Jelentősége a LIS3DH és az LSM9DS0 beüzemelésénél volt. Ha olvasásnál ugyanazt az eredményt adja vissza, amely az adatlapon is szerepel, akkor a kommunikáció megfelelően működik, és a kiolvasás sikeres volt. Amennyiben nem sikerül visszaolvasni a megfelelő értéket, a program hibát jelez, és leáll a működése.

CTRL_REG_1_G:

A giroszkóp tengelyeinek engedélyezése és a mintavételi frekvencia beállítása. Mivel a szögsebesség-szenzor adatainak kiolvasása interrupttal történik, amelyet a mágneses szenzor vezérel 50 Hz-es mintavételével, ezért fontos szempont, hogy a mintavétel ennél gyorsabb legyen, ennek megfelelően 95 Hz-et állítottam be.

CTRL_REG_2_G:

High-pass filter beállítások. A tesztek során az derült ki számomra, hogy csökkenteni lehet a tengelyek driftjét, ha felüláteresztő szűrőt használok. A vágási frekvencia növelésével javultak az eredmények. 95 Hz-es mintavételezésnél a legmagasabb választható vágási frekvencia 7.2 Hz, ezt az értéket választottam. Mivel a kézzel történő pozíció váltások ennél jóval gyorsabbak, ezért a vágással nem veszítünk értékes információt.

CTRL_REG_4_G:

A szenzor érzékenységének beállítása. Bár a szenzor maximális beállítása 200 rad/s értéket is megenged, a tesztek során hamar kiderült, hogy az 500 rad/s elégséges a gyors mozgások érzékelésére.

5.2.2 A mágneses- és gyorsulásmérő szenzort tartalmazó subchip beállításai

CTRL_REG_1_XM:

Gyorsulásmérő mintavételezési frekvenciájának beállítása. Ez különösen fontos, hogy a maximális értéken legyen, mivel így lehetünk biztosak abban, hogy elkapjuk a detektált ütések csúcsértékét. A LIS3DH 5 kHz maximális frekvenciájához képest az LSM9DS0 maximális beállítása 1600 Hz, de ez a mintavételezési frekvencia is elégségesnek bizonyult a csúcsok érzékeléséhez.

CTRL_REG_2_XM:

Gyorsulásmérő érzékenysége. A **3.1 fejezetben** végzett tesztek alapján jól látszik, hogy egy nagyobb ütés esetén a gyorsulás akár a 100 g-t is elérheti. Ezért az érzékenységet az elérhető maximális ±16 g-re kell állítani

CTRL_REG_3_XM:

Interrupt generátor beállításai az INT1 lábon. Mivel a mágneses kiolvasás frekvenciája fogja megadni az interruptok kiadásának gyakoriságát, ezért a P1_DRDYM regiszter értéket 1-re kell állítani. Az interrupt pin leköveti a DATA READY jelet (**6.2 ábra**), amely 1-re vált, ha van új elérhető adat, és akkor tér vissza 0-ba, ha az adatot kiolvasták.

CTRL_REG_5_XM:

Mágneses szenzor mintavételezési frekvenciája. Ahogy korábban írtam, ez adja meg az INT1 pin kiadott interrupt frekvenciáját is.

A Madgwick tanulmány szerint az általuk fejlesztett algoritmus akkor működik megfelelően, ha alacsony frekvencián, 10 Hz körüli mintavételezéssel használják [10]. Ehhez képest magasabb frekvencián pontosabb és gyorsabb működést tapasztaltam az én eszközömmel. Végül azért választottam az 50 Hz-es mintavételezést, mert a 100 Hz-es interruptok már nem érkeznek meg maradéktalanul az 1 méter hosszú szalagkábel és a breakout board pinjeire ráhúzott csatlakozók kontaktus hibái miatt. A breakout board oldalon így is forrasztással biztosítottam az elektromos összeköttetést, azonban ezt a mikroprocesszor oldalon ezt nem tehettem meg.

CTRL_REG_6_XM:

Mágneses érzékenység kiválasztása. Mivel számomra a föld mágneses mezeje (0.5 Gauss egyenlítőnél) hordoz információt, ezért a legkisebb, ±2 Gauss-os méréshatárt választottam.

INT_CTRL_REG_M:

Itt lehet engedélyezni, hogy a mágneses szenzor interruptot generálhasson.

5.3 Kommunikáció az mbed LPC1768 és a szenzorok között

Az mbed fejlesztőkártya többféle üzemmódot biztosít a perifériákkal való kommunikációra. A LIS3DH (és az LSM9DS0) I2C és SPI kommunikációt támogat. Én az I2C-vel kezdtem a fejlesztést, mivel ez tűnt a legegyszerűbbnek és ez használja a legkevesebb vezetéket. A protokollt a 90-es évek végén fejlesztették ki, és azóta is rengeteg elektronikai eszköz használja egymás közötti soros kommunikációra. Az I2C (IIC, Inter-Integrated Circuit) kommunikáció egyik tulajdonsága, hogy elég hozzá 3 vezeték: egy a földnek (GND), egy az órajelnek (SCL) és egy az adatoknak (SDA). Az I2C koncepciója azon alapul, hogy minden adattal kapcsolatos jel az SDA vezetéken fut, nincsenek külön címvezetékek vagy Read/Write műveleteket megkülönböztető vezetékek.

A kommunikációban részt kell vennie legalább egy MASTER eszköznek, amely a folyamatot elindítja és adja az órajelet, illetve maximum a címzés bitszélességének megfelelő számú SLAVE eszköznek. A SLAVE eszközök a címük alapján lesznek beazonosíthatóak. Az I2C kommunikáció maximális sebessége fast módban 400 kHz [11].

5.3.1 I2C kommunikáció az LSM9DS0 inerciális szenzorral

Ahogy az adatlap is írja, az SCL és az SDA vonalnak szüksége van arra, hogy alapállapotban V_{dd}-re legyenek kötve. Ezt a LIS3DH breakout boardján egy 1 kOhm-os ellenállással oldottam meg, az LSM9DS0 boardjába pedig eleve be van építve a felhúzó ellenállás.

Table 12. Transfer when master is writing multiple bytes to slave

Master	ST	SAD + W		SUB		DATA		DATA		SP
Slave			SAK		SAK		SAK		SAK	

5.1. táblázat: Több byte-os adat olvasása a slave eszközről

Table 14. Transfer when master is receiving (reading) multiple bytes of data from slave

Master	ST	SAD+W		SUB		SR	SAD+R			MAK		MAK		NMAK	SP
Slave			SAK		SAK			SAK	DATA		DAT A		DAT A		

5.2. táblázat: Több byte-os adat írása a slave eszközre

Az üzenetek (**5.1 és 5.2 táblázat**) az UART-hoz hasonlóan start jellel (ST) kezdődnek és stop jellel (SP) zárulnak. A start után következik a SLAVE eszköz 7 bites címe (SAD), a 8. bit pedig azt határozza meg, hogy írás (W) vagy olvasás (R) történik. A SLAVE eszköz egy nyugtát (SAK) küld vissza, ha helyes volt a cím. Ezután a MASTER eszköz elküldi a 8 bites alcímet (SUB), amellyel az eszköz regisztereit lehet címezni. Nyugtázás után bekövetkezik a kért adatok (DATA) írása/olvasása. Oszcilloszkóp képe az I2C kommunikációról a **Függelék 7. ábráján** látható.

Bár a 7 bites címzés sokféle SLAVE eszköz azonosítását tenné lehetővé, az általam használt IC-knél ez nem lehetséges. Az összes LIS3DH és LSM9DS0 IC-nek fix címe van, egyedül a cím utolsó bitjét lehet állítani földre vagy V_{dd} -re kötéssel. Így egy I2C vonalon 2 eszközt tudunk megcímezni. Az LSM9DS0-nak annyi különbsége van a LIS3DH-hoz képest, hogy külön szubchip kezeli a giroszkóp adatait, így annak külön írás és olvasási címe van.

Az mbed saját könyvtára I2C adatok írására és olvasására beépített függvényeket tartalmaz, amelyet egy I2C objektum létrehozása után lehet használni.

Több byte írása LSM9DS0 IC-n:

```
i2ccomm[0] = CTRL_REG1; // makró a CTRL1 regiszter címére
i2ccomm[1] = 0x08; // írni kívánt adat
i2c.write(WRITE_ADDR, i2ccomm , 2); // 2 byte adat írása
```

Több byte olvasása LSM9DS0 IC-n:

```
reg1data = CTRL_REG1;
i2c.write(WRITE_ADDR, &reg1data, 1);
i2c.read(READ_ADDR,&reg1data, 1);
```

Megfigyelhető, hogy előbb egy írás művelettel el kell elküldeni az IC számára, hogy melyik regiszter tartalmát szeretnénk olvasni. Az IC ezután átmásolja az olvasni kívánt regiszter tartalmát a READ_ADDR makróval ellátott regiszter címre, innen érjük el a kívánt adatot.

5.3.2 SPI kommunikáció az LIS3DH szenzorral

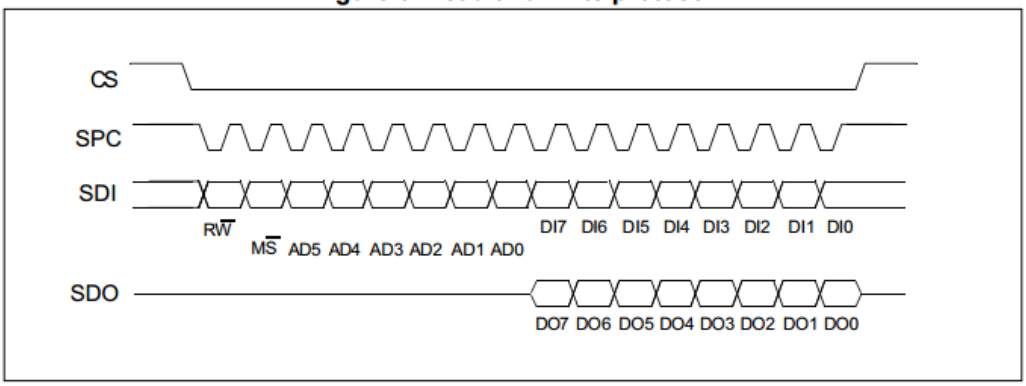
A fejlesztés végén a rendszerhez csatlakoztattam a kéz egység mellé egy láb egységet is. Ez nem más, mint a korábbi tesztekhez használt LIS3DH digitális kimenetű gyorsulásmérő szenzor. Ahogy az LSM9DS0, ez is támogatja az I2C-t és az SPI-t. Bár az LPC 1768 pinoutja szerint 2 darab I2C kommunikációt tud kezelni, gyakorlatban ezek

használata vagylagos. Ettől függetlenül meg tudtam volna oldani, hogy a már meglévő I2C kommunikáció adat- és órajelvonalát megosztom Y elágazó kábelekkel. Azonban ez még több csatlakozók közötti kapcsolatot kívánt volna, és tapasztalataim szerint ez a csatlakozótípus nem biztonságos, így adatkiesést okozhat.

Ezért az SPI-t választottam a LIS3DH láb egység kommunikációs protokolljának. Másik szempont volt, hogy ha a későbbiek folyamán 4 egység csatlakozik a mikrokontrollerhez, akkor szükség van a gyorsabb kommunikációra. Az SPI frekvenciája akár 1 Mhz is lehet, szemben az I2C maximális 400 kHz-es értékével. Az eszközök kiválasztása itt nem címzéssel történik, mint az I2C esetében, hanem *Chip Select* jellel, tehát tetszőleges számú eszközt lehet rá csatlakoztatni. Hátránya, hogy legalább 3, de normál módban 4 vezeték kell hozzá (*MOSI, MISO, SCK, CS*).

Az SPI kommunikáció Master eszköze, esetünkben a mikrokontroller szabja meg a kommunikáció tulajdonságait. Az mbed SPI könyvtár `spi.format(16,3)` parancs első paraméterével a bitek számát lehet beállítani, második paraméterével az üzemmódot. Az SPI 4 féle üzemmódot tartalmaz, amelyek 2 tulajdonság kombinációiból jönnek létre. Az első tulajdonság az órajel polaritás, a második az adatok olvasási és írási helye. Mivel a LIS3DH szenzor adatlapja low active órajel polaritást és felfutó élre történő olvasást követel meg, ezért a 3. módot választottam.

Figure 6. Read and write protocol



5.2. ábra: SPI kommunikáció LIS3DH szenzorral

Ahogy **5.2 ábrán** látható, a LIS3DH szenzornak 16 bitre van szüksége a kommunikációhoz. Az SDI (*MOSI*) első bitje megadja, hogy írás vagy olvasás történik. A második bit engedélyezi az automatikus cím inkrementálást. Ezután következik a 6 bites regiszter cím. Ha olvasás történik, akkor a további bitek nem relevánsak, ha írás, akkor a megadott regiszterbe az utolsó 8 bit fog beleíródni. Ha olvasás történt, akkor az SDO (*MISO*) vonalon érkezik a válasz az utolsó 8 bit ideje alatt [8]. A **Függelék 8. és 9. ábrája** az írás és az olvasás hullámformáját mutatja be.

6 Beágyozott szoftver

6.1 NXP 1768 és az mbed fejlesztőkörnyezet

Bár az általam fejlesztett eszköz végleges, forgalomba hozható változatához egyértelműen saját terv alapján készült nyomtatott áramkört használnék, a hardvertervezés a szakdolgozatomnak nem része. Emiatt olyan eszköz után néztem, amely egyszerűsíti a prototípus elkészítését és segít abban, hogy a szakdolgozat keretében a jelfeldolgozó algoritmusokra koncentrálhassak. A tanszéken kaptam egy NXP LPC1768 fejlesztőkártyát, amely tökéletesen megfelel erre a célra. Mivel korábban nem dolgoztam beágyazott környezetben, ezért első lépés az eszköz megismerése volt.

Az mbed fejlesztőkörnyezet egy böngészőből futtatható alkalmazás, amely beépített könyvtárainak és egyéb funkciónak köszönhetően egyszerűvé teszi a programozást. A könyvtár a kártyán található kommunikációs protokollokhoz használható osztályokat tartalmazza, amellyel a kommunikáció típusának megfelelő objektumokat hozhatunk létre, és az osztály függvényei segítségével vezérelhetjük az adatcserét. Ezen kívül az alap C könyvtárakat (pl. math.h) is magában foglalja, illetve könnyen importálhatóak bele mások által írt projektek is. Ilyen például az általam használt *USBMIDI* [9] könyvtár is.

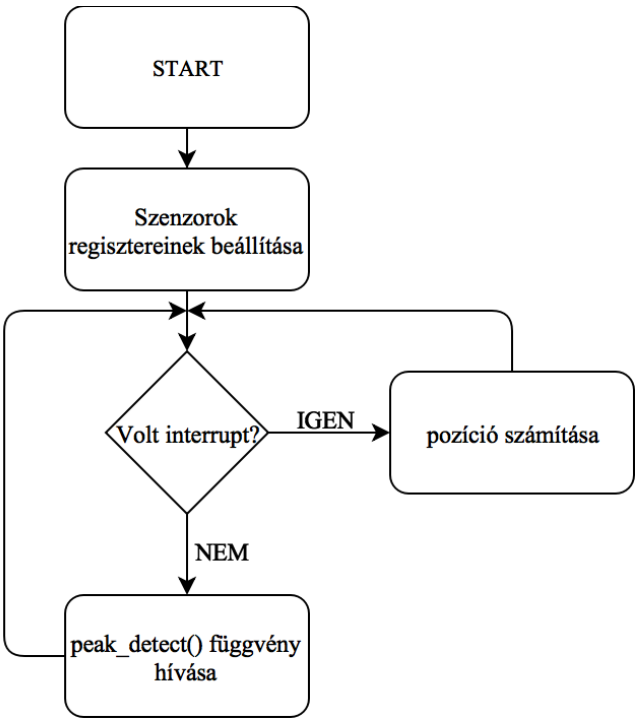
A Drag'n'Drop programozás azt jelenti, hogy miután a compiler sikeresen lefutott, egy bináris file töltődik le a felhasználó számítógépére. A PC-hez USB-n csatlakoztatott eszköz (amely csatlakozót a soros kommunikáció is használja) külső flash memóriaként jelenik meg a számítógép eszközkezelőjében. Ha erre az eszközre "áthúzzuk" a bináris file-t, akkor a hardveren a reset gomb megnyomása után a bináris file áttöltődik a RAM-ba, és a legújabb programverzió fog futni az eszközön.

Az, hogy a környezet online elérhető, egy bizonyos fokú biztonságérzetet ad, mivel a fileok a fejlesztő számítógép meghibásodása esetén se vesznek el, a forráskód bármilyen böngészővel rendelkező eszközről elérhető. Ezen felül az applikációban beépített alapszintű verziókezelés is megtalálható, amely lehetővé teszi, hogy szükség esetén a felhasználó vissza tudja állítani a szoftvert egy korábbi állapotra.

Azért vannak hiányosságai is az mbed környezetnek: az, hogy a fejlesztés online folyik, az egyben hátrány is, mivel internetelérés nélkül a fejlesztés nem folytatható. Ezen felül nincs benne autocomplete funkció, amely megkönnyítené a gépelést, illetve nem tartalmaz debuggert, így a program esetleges hibáira a fejlesztőkártyán található LEDek visszajelzéséből és soros porton keresztüli konzolra kiíratásból lehet következtetni.

6.2 A mikrokontrolleren futó program

A mikrokontrolleren található szoftver 3 fő részből áll (**6.1 ábra**): az inicializációhoz tartozó függvényekből, a csúcsdetektáláshoz tartozó függvényekből és az orientáció detektáláshoz tartozó függvényekből.



6.1. ábra: main() függvény működése a beágyazott szoftverben

A RESTART gomb megnyomása után az inicializáció az I2C kommunikáció helyes működésének ellenőrzésével indul, amelyben megvizsgálja a `hello_world()`, hogy a *WHO_I_AM* regiszterek a megfelelő értékeket adják-e vissza. Amennyiben az értékek helyesek, megtörténik az inerciális szenzor regisztereinek beállítása a `set_gyro()` és a `set_accelerometer_magnetometer()` függvényekkel az **5.2.1 és 5.2.2 fejezetben** olvasottak szerint.

Az inicializálás után felfutó interrupt élre aktiválom az orientáció érzékelő függvényt a `m_int.rise(&read_alldata_interrupt)` paranccsal. A szoftver ezután egy végtelen ciklusba ér, amelyben a `peak_detect()` függvény hívódik meg folyamatosan.

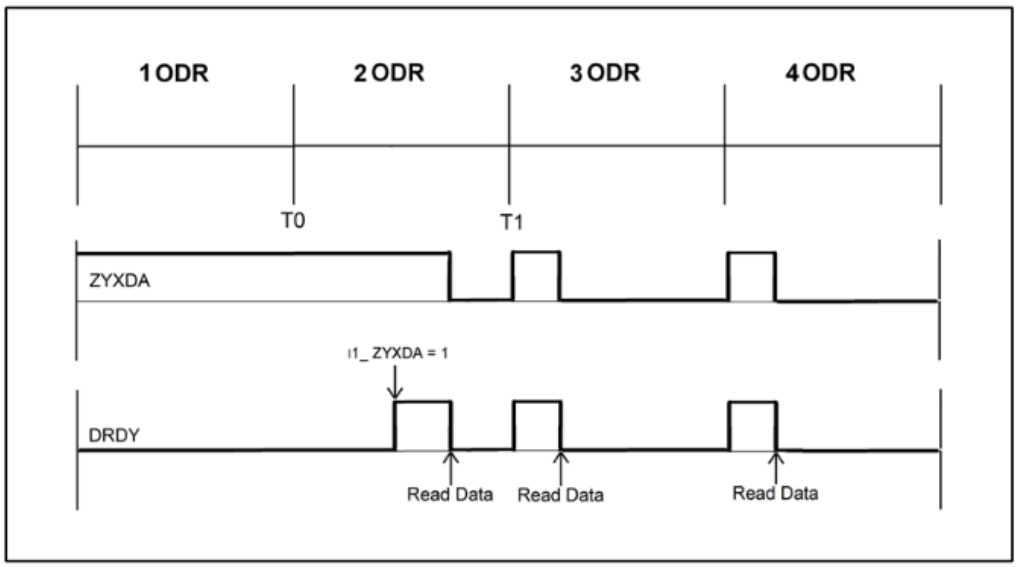
Amennyiben nem érkezett interrupt, bekerülünk a `peak_detect()` függvénybe. A függvény diagramja a **Függelék 6. ábrán** látható. A függvény csúcs detektáló algoritmusáról a **3.3 fejezetben** írtam. Amennyiben a rendszer csúcsot észlel, először meghívódik a `midi_scale()` függvény, amely a MIDI üzenet *velocity* paraméterét fogja visszaadni, majd meghívódik a `choose_sector()`. A függvény bemeneti paraméterei között vannak az előző orientáció meghatározásból származó adatok. A virtuális teret összesen 6 szektorra osztottam: 3 vízszintes és 2 függőleges virtuális térszeletre. A jelenlegi prototípusban a **8.2 fejezetben** említett okokból kifolyólag ebből csak 2 üzemel.

Miután a rendszer megállapította a szektort, a `choose_instrument()` függvény visszaadja a megfelelő hangszer MIDI *Note* paraméterét. Ezután a `midi_play()` függvény megkapja a megfelelő paramétereket, és kiküldi az erre dedikált USB kábelen a MIDI üzenetet, majd visszatér a végtelen ciklus elejére.

Amennyiben érkezett interrupt, az orientációdetektáló függvény hívódik meg. Beolvasásra kerül mind a 9 szenzor adat, amely a Madgwick algoritmus bemenete lesz. A kvaternió kimenetből az `euler_angles()` függvény számol szögeket.

6.3 Interruptok működése

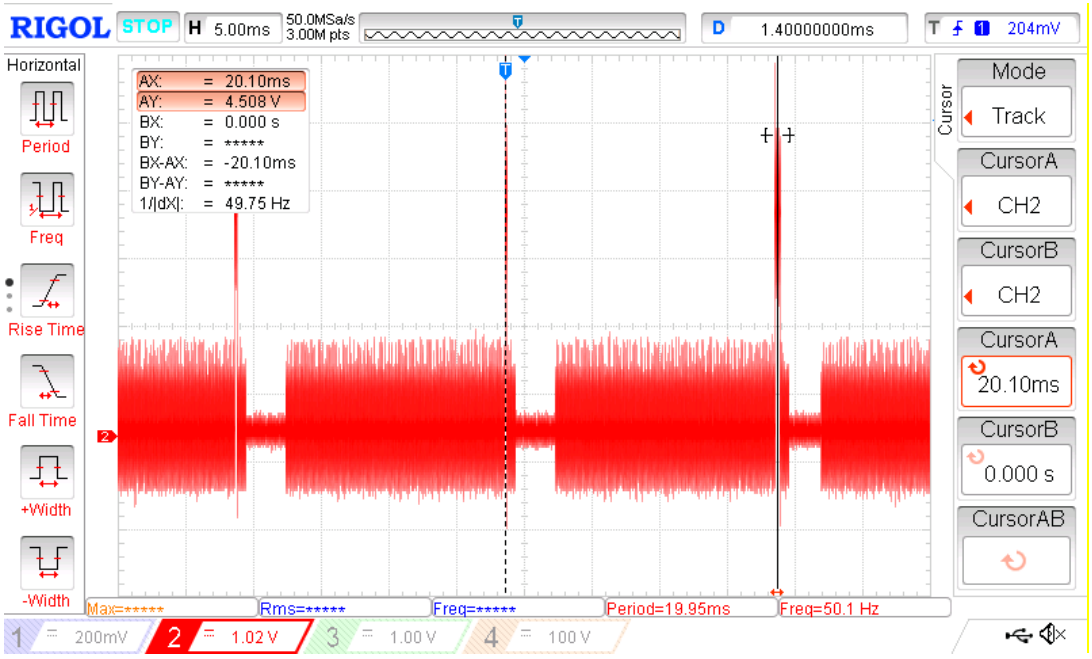
Az LSM9DS0 3db programozható interrupt lábbal rendelezik (ebből 2 a mágneses és gyorsulásmérő szubchipé, 1 pedig a giroszkópé). A szenzort programozni lehet, hogy különféle események bekövetkezésekor megváltoztassa a lábak állapotát. Ilyen eseménynek számít egy bizonyos gyorsulási határérték átlépése, klikkelő mozdulat észlelése, vagy az általam is használt DATA READY jel.



6.2. ábra: LSM9DS0 DATA READY jel

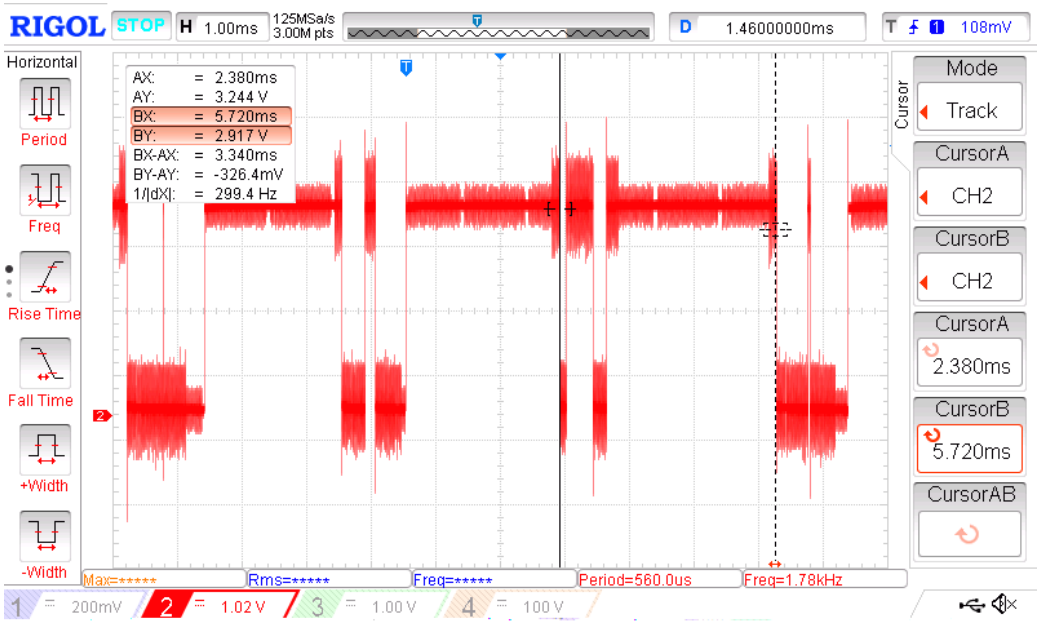
A DRDY jel alaphelyzetben 0-ból indul. Ha a kiválasztott szenzorban új adat elérhető az összes aktív tengelyre, akkor a jel 1-be vált. Kiolvasás esetén tér vissza 0-ba. Ez a jelalak normál polaritásra vonatkozik, a regiszterekből állítható, ha low-active működést szeretnénk elérni.

Én a mágneses szenzor DATA READY jelét vezettem ki az INT2 interrupt lábra. Mintavételi frekvenciának 50 Hz-et állítottam be, tehát az interrupt is ilyen gyakorisággal ugrik 1-be, majd vissza 0-ba. Ez a jel triggereli az orientáció kiszámoló függvényt. Fontos itt megjegyeznem, hogy a DRDY jel csak akkor ugrik vissza 0-ba, ha a kiolvasás megtörtént. Tehát még az interruptot aktiváló függvény előtt kell lennie egy regiszterolvasásnak, máskülönben a DRDY jel 1-ben fog maradni. Ha az interruptvonalon valamilyen csatlakozási probléma miatt kimarad egy impulzus, akkor nem történik meg a következő kiolvasás, így az orientációszámító függvény működése is. Ezt a problémát egy watchdog elven működő függvénnyel lehet orvosolni, amely ha bizonyos ideig nem érzékel interruptot, akkor kiolvassa a megfelelő regisztereket, így a DATA READY jel megint megfelelően tud működni.



6.3. ábra: Oszcilloszkóp ábra 50 Hz-es DATA READY interruptról

Eredeti ötlet az volt, hogy nem csak az orientációszámító függvény működik interruptként, hanem a gyorsulási csúcsokat figyelő algoritmus is. Ezt a függvényt a gyorsulás szenzor DRDY jele triggerelte volna a mintavételezési frekvenciával, amely 1.6 kHz ennél a szenzornál. Azonban ez a megoldás nem működött. A **6.4 oszcilloszkóp ábra** mutatja, hogy a jelek nem folytonosan érkeznek, mint az 50 Hz esetében, hanem párosával, majd egy kis szünet kihagyásával újból 2 interrupt generálódik. Ez a működés nem helyes, arra tudok következtetni, hogy az eszköz nem képes ilyen magas frekvenciájú interruptjel kiadására. A gyorsulásmérő szenzor frekvenciáját nem csökkenthetem, mivel ezzel azt kockáztatnám, hogy egy ütéscsúcs kimaradna a mintavételezésből. A végső megoldás szerint a pozíciót számoló függvény továbbra is interrupttal hívódik meg, míg a csúcsdetektáló függvényt egy végtelen ciklussal olvasom be. A végtelen ciklusban olvasás sajnos nem a legjobb megoldás, mivel folyamatosan terheli a processzort.



6.4. ábra: Oszcilloszkóp képe a rosszul működő 1.6 kHz-es interruptról

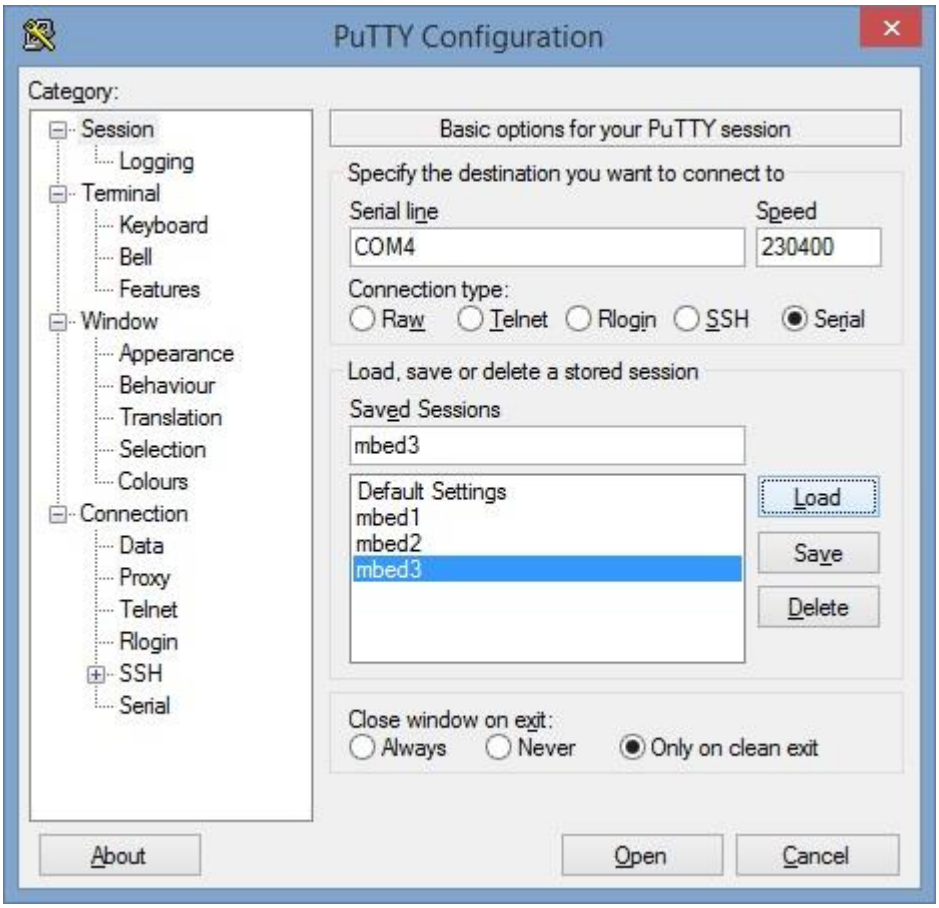
A helyes működés érdekében használni kellett egy - egy `disable_interrupt()` parancsot a függvények elején. Ezen parancsok nélkül a rendszer összeomlott. Ennek következményeként addig nem tud érvényre jutni semmilyen interrupt, ameddig azt újra engedélyeztük `enable_interrupt()` paranccsal. Emiatt kisebb csúszások jöhetnek létre az 50 Hz-es interrupt meghívásban. A Madgwick algoritmusnak szüksége van arra, hogy tudja, hogy mennyi idő telt el az előző mintavételezés óta. Ezt a függvény elején a SAMPLE FREQUENCY konstans beállításával lehet megadni. Az én esetemben ez nem megoldás, ezért úgy módosítottam az algoritmust, hogy egy timerrel mérem az előző függvényhívás óta eltelt időt, és ezt a paramétert használom fel a konstans helyett. Az algoritmus így érzéketlen lett a kisebb interrupt meghívási frekvencia változásokra.

7 Számítógépes környezet

Ahhoz, hogy a szenzorokból eljusson a jel a számítógépre és s PC hang formájában megjelenítse a megfelelő hangszert, sokféle programon kell keresztülmennie. Ebben a fejezetben a számítógépes környezet alkalmazásaival és azok megfelelő beállításával foglalkozom, illetve megismertetem az általuk használt protokollokat.

7.1 PuTTY SSH kliens

A PuTTY egy nyílt forráskódú Telnet és SSH és kliens, amellyel COM portokat is lehet olvasni. Ezt a funkcióját használtam a tesztelések során és a kész megoldásban is. Az mbed LPC1768 USB-n keresztül csatlakozik a számítógéphez, amelyet egy driver telepítése után a PC virtuális COM portnak észlel. Az eszközkezelőből kikeresve megtalálható, hogy pontosan melyik COM porton keresztül folyik a kommunikáció. Ezt, illetve a beágyazott szoftverben beállított baud rate-et megadva indítható a beolvasás.



7.1. ábra: a PUTTY SSH kliens soros port olvasásra konfigurálva

A beágyazott szoftverben a `pc.printf()` paranccsal lehet kiíratni erre a terminálra a kívánt adatokat. A PuTTY rendelkezik Logging funkcióval, amely a konzolba kiíratott

szöveges kimenetet menti ki tetszőleges formátumú file-ba. Nálam a kimeneti file a MATLAB által olvasható .m kiterjesztést kapta, így könnyen meg tudtam oldani azt, hogy a szenzoradatokat kiküldöm soros porton, majd egyből használhatom őket tesztelésre a MATLAB környezetben.

7.2 Reaper Digital Audio Workstation

A Reaper egy DAW (Digital Audio Workstation) szoftver. Hanganyagok felvételére, keverésére és masterelésére szokták használni. Diákok széles körében elterjedt, mivel megfizethető az ára, és funkciókban nem sokkal marad el a professzionális szintű, sokkal drágább versenytársaitól.

Én arra használom, hogy a Toontrack EZ Drummer nevű VSTi-t futtassam rajta. A VST-k (Virtual Studio Technology) DAW-ba beépülő pluginek, amelyek kiegészítik az alap funkciókat. Ennek egy külön csoportja a VSTi, melyben az i Instruments-et jelöl. Ezek a pluginek valódi hangszerek hangját szintetizálják.

7.3 Toontrack EZ Drummer VSTi

A Toontrack EZ Drummer [12] egy virtuális dob hangszer. A dobok hangját nem szintetizálással hozza létre, hanem különböző módon és eszközökön felvett mintákból lehet kiválasztani a számunkra megfelelő dobhangot. Számomra legfontosabb tulajdonsága, hogy MIDI parancsokkal triggerelhető. Ez a megoldás jelentősen leegyszerűsítette a szakdolgozatom működését. A korábbi elképzelés szerint én hoztam volna létre szintetizálással a különböző hangokat, de MIDIn keresztül triggerelve az EZ Drummer sokkal jobb minőségű dobhangot képes előállítani, mint amilyet létre tudtam hozni, és így több időt fordíthattam a mozgások feldolgozására.

A professzionális VSTi-kre jellemző, hogy élethűen próbálják leutánozni a valóságos hangszerek akusztikai tulajdonságait. A doboknál megfigyelhető jelenség, hogy egy gyengén megütött hangszernek nem csak dinamikában van különbsége az erősen megütötthez képest, hanem hangszínében és lecsengésében is. A Toontrack terméke is ilyen. A korábban említett MIDI skálázó függvény végzi el a beérkező gyorsulásmérő adatokból az “erősség” (*velocity*) kiszámolását, és ezt az adatot továbbítja a számítógép fele. Innentől a VSTi feladata, hogy lejátssza a neki megfelelő dob hangot.

.

7.2. ábra: Toontrack EZ drummer VST [12]

7.4 ASIO4ALL

Az betűszó ASIO jelentése Audio Stream Input/Output, amely feladata a hangkártya összekötése a DAW szoftverrel. Alap esetben ezt a feladatot elvégzi az operációs rendszerhez tartozó hangkártya kezelő alkalmazás. Azonban a Windows saját hangillesztő szoftverének nem prioritása az alacsony késleltetés, ezért nem lett optimalizálva a futási ideje. Az én felhasználásomban kritikus a gyors válaszidő, ezért használom ezt az ingyenesen letölthető illesztőszoftvert a Reaper DAW szoftverrel együtt.

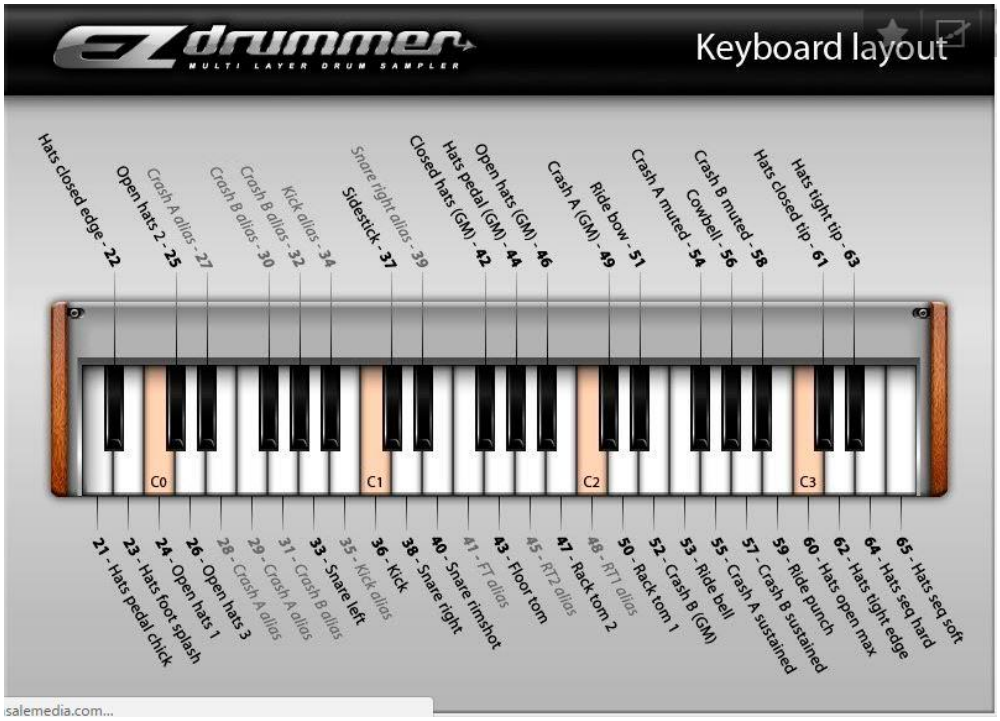
7.5 A MIDI protokoll

A MIDI szabványról szintén több oldalt lehetne írni, emiatt csak a témámmal kapcsolatos elemeket fogom kifejteni. A MIDI (Musical Instrument Digital Interface) egy 1982-ben létrehozott szabvány, amely a MIDI szabvány által használt 5 pólusú kábelt, a rajta futó jel fizikai tulajdonságait és a MIDI üzenetek tartalmát definiálja. Az én felhasználásomban a MIDI kábel nem szerepel, egy USB csatlakozón keresztül csatlakozik a mikroprocesszor a számítógéphez. A PC ezen az USB csatlakozón keresztül külső hangeszközként ismeri fel a fejlesztőkártyát, és MIDI üzeneteket vár.

7.5.1 MIDI üzenet

A vezérlő hangszer (vagy más néven az adó) az információt bájtok sorozataként küldi a vevő, vagyis a vezérelt eszköz felé. Egy MIDI-üzenet legalább egy bájt hosszú, maximális hossza pedig nincs meghatározva [13].

A MIDI szabvány 4 féle üzemmódot különböztet meg aszerint, hogyan értelmezik a bejövő üzeneteket. Az üzemmódot a fogadó eszköz felé közölni kell egy-egy folyamat során, amelyet RESET után az USBMIDI könyvtár `set_mode()` függvénye végez el. Én a 3. üzemmódot használom: OMNI ON, POLY. Az OMNI azt jelenti, hogy a fogadó eszköz csak egy csatornát figyel. A virtuális dob eszközöm egy csatornát használ, azon belül a különböző dobhangokat a *Note* paraméter változtatásával lehet elérni. A POLY pedig azt jelenti, hogy egy csatornán többféle hang (*Note*) jöhet.

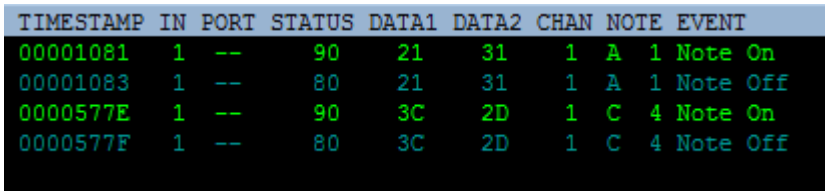


7.3. ábra: EZ drummer VST note-jai [12]

7.5.2 Hang jellegű üzenetek

Az lentebb látható **7.4-es ábra azt mutatja, ahogy** a MIDI-OX nevű szoftverrel figyeltem a külső hangeszközről érkező üzeneteket. Két különböző pozícióban hajtottam végre ütést, amely négy üzenetet eredményezett. A hang jellegű üzenetek státusz bittel kezdődnek. Ez megmondja a fogadó eszköznek, hogy az utána lévő DATA biteket hogyan értelmezze. A PC szoftver kiírja státuszokhoz tartozó eseményeket: a `0x90` kódhoz tartozó státusz *NoteOn* parancsot jelöl, a `0x80` pedig *NoteOff*-ot. A `9`-es és a `8`-as utáni `0` ez esetben a 1-es csatornát jelöli, mivel a *STATUS* biteknél a számozás 0-tól, míg a csatornák számozása 1-től kezdődik. Ezután következnek az adatbitek. A *DATA1* 8

biten ábrázolja a *Note* paramétert, utána a *Velocity* következik, melynek 128 különböző értéke lehet.



TIMESTAMP	IN PORT	STATUS	DATA1	DATA2	CHAN	NOTE	EVENT
00001081	1 --	90	21	31	1	A 1	Note On
00001083	1 --	80	21	31	1	A 1	Note Off
0000577E	1 --	90	3C	2D	1	C 4	Note On
0000577F	1 --	80	3C	2D	1	C 4	Note Off

7.4. ábra: MIDI üzenetek detektált ütés esetén

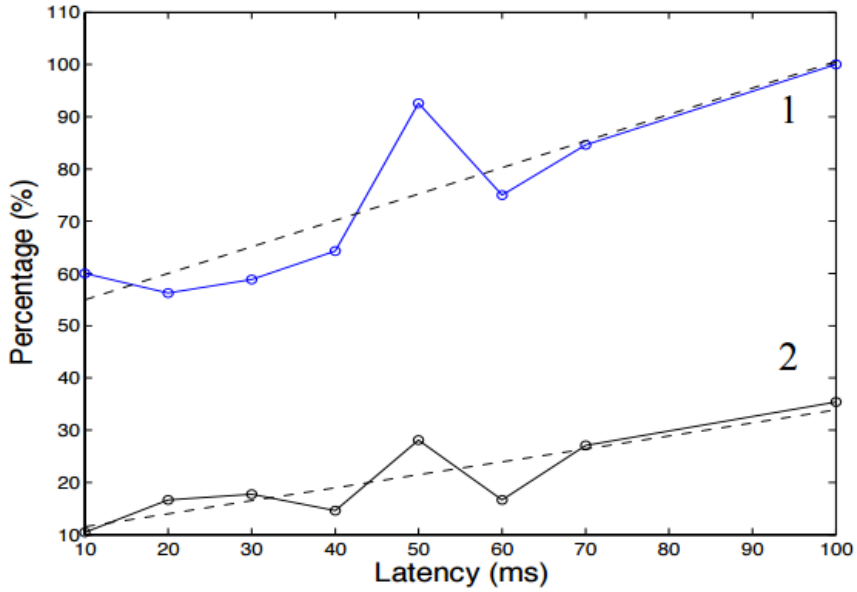
A dobhangok megszólaltatására kétféle üzenetet használok az USBMIDI könyvtárból: A `midi.write(MIDIMessage::NoteOn(note,velocity))` és a `midi.write(MIDIMessage::NoteOff(note,velocity))`-t. Paraméterként a fentebb mellékelt *Note*-oknak megfelelő számokat várják, illetve a *velocity*-t, amelyet az általam írt `midiscale()` függvény számol ki a pillanatnyi maximális gyorsulásból. Mivel a doboknál a hangszer érintése impulzusszerű, ezért a *NoteOff* üzenet egyből a *NoteOn* üzenet után következik.

8 A prototípus tesztelése

8.1 Rendszer késleltetése

Bármilyen virtuális zenei megoldásnál rendkívül fontos szempont a rendszer késleltetése. Képzeljük el a szituációt, amikor lenyomjuk a szintetizátor billentyűjét, és a hang tizedmásodpercekkel később szólal csak meg. Egy dob eszköznél, amelynél a legfőbb szempont az, hogy ritmusban tudjunk játszani, az észrevehető késleltetés teljesen el tudja rontani a játékélményt.

Teemu Mäki-Patola a hangeszközök késleltetésével foglalkozó tanulmányában [14] azt írja, hogy a késleltetés észlelése sok dologtól függhet. Ilyen például a felhasználó zenei háttere, kora, illetve a hangszer és a játékstílus is. Egy olyan hangszernél, amelynél hosszú kitartott hangokat játszanak (Pl. Themerin) akár a 100 ms is elfogadható.



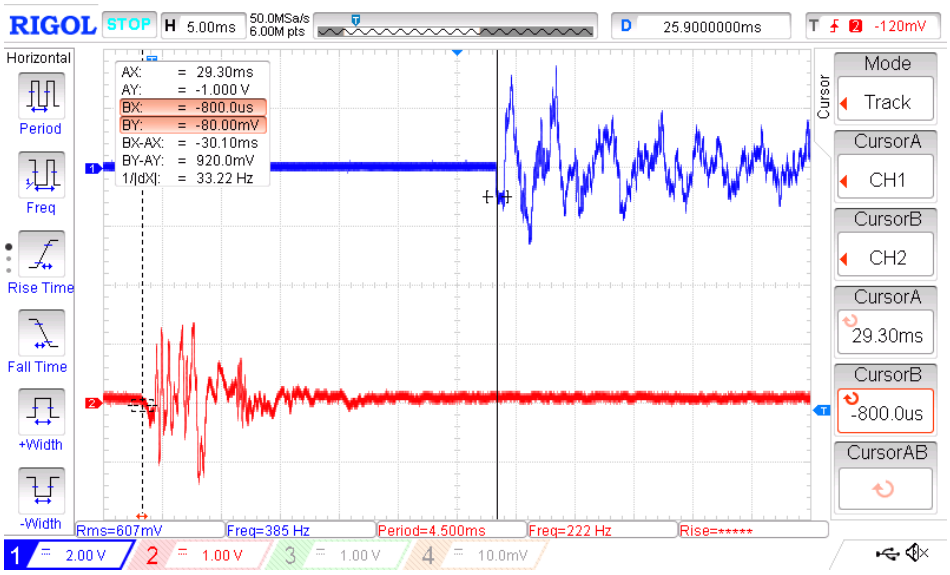
8.1. ábra: Teemu Mäki-Patola tanulmány a legkisebb érzékelhető késésekről [14]

A Teemu Mäki-Patola tanulmány kísérlete abból állt, hogy egy hangszórón különböző szinusz hangokat játszottak le, és a résztvevő embereknek Theremin segítségével reprodukálni kellett azokat úgy, hogy közben a rendszerbe különböző késleltetéseket vittek be. Az alanyoknak feltették azt a kérdést, hogy a 2 beállítás közül melyiknek volt nagyobb a késleltetése.

A **8.1 ábra** 1-essel jelölt grafikonja azt mutatja, hogy a kísérletben résztvevő emberek hány százaléka találta zavarónak az adott késleltetést. Itt csak azokat a méréseket vették figyelembe, ahol helyesen válaszoltak a feltett kérdésre. A 2-essel jelölt görbe szintén azt mutatja, hogy a résztvevők hány százaléka tartotta zavarónak az adott késleltetést, de itt beleszámolták azokat a méréseket is, ahol rossz válasz érkezett a feltett kérdésre.

A tanulmány alapján zenei megoldásoknál törekedni kell arra, hogy a késleltetés 40 ms alatt legyen, efelett az érték felett a késésből adódó kognitív disszonancia már jelentősen befolyásolja a játékélményt.

A saját eszközöm létrehozásánál is törekedtem arra, hogy kevésbé számításigényes algoritmusokat használjak, illetve minimálisra csökkentsem a rendszer egyéb elemeiből származó késleltetést.



8.2. ábra: A rendszer késleltetése oszcilloszkóppal kimérve

A mérés során az oszcilloszkóp 1-es csatornájára kötöttem a számítógép fejhallgató kimenetéből érkező jelet, a 2-es csatornára pedig egy mikrofont, amely rögzítette a kezem térdemhez való csapódásának hangját. A két esemény között 30 ms telt el.

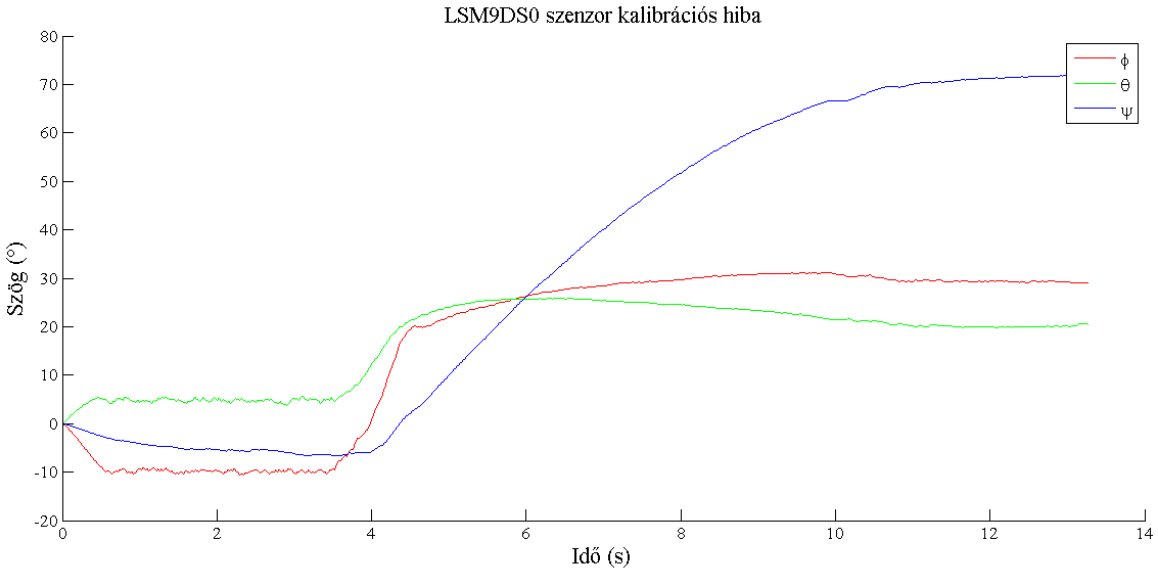
Ez az érték önmagában soknak tűnhet, azonban az előző részben kifejtett okok miatt nem észrevehető. Mivel a dobütés egy éles hangot eredményez, ehhez képest a térden történő tompa puffanást nem is lehet érzékelni. A nem térden történő ütések (pl. cintányér) pedig szintén nem okoz késésből adódó kényelmetlen érzést, mivel ott előbb jön a hangjel, és ennek az érzékelése után állítjuk meg a kezünket.

8.2 Kalibráció hiányából származó hiba

Jelenleg az eszköz kétféle pozíciót képes detektálni, amelyet a kéz X tengely körüli elfordulás Euler-szögéből számol ki. A beágyazott szoftver alapján ennél többre is képes az eszköz. Ha a Z tengely körüli elfordulást is bevesszük a számításba, akkor a 3 vízszintes és 2 függőleges pozíció összesen 6 szektort ad ki, amellyel 6 különböző dob szólaltatható meg.

A **4.3 ábra** alapján majdnem tökéletesen működik a Madgwick algoritmus teszt közben a három Euler szög meghatározására, azonban ez folyamatos használat mellett sajnos nem

igaz. Ha a kéznek van X vagy Y tengely körüli szögelfordulása, akkor a Z tengely körüli szögnél drift figyelhető meg. Ez azt a hibás működést okozza, hogy az eszköz folyamatos forgást érzékel a Z tengely körül.



8.3. ábra: Kalibrációs hibából fakadó drift a Z tengely körül

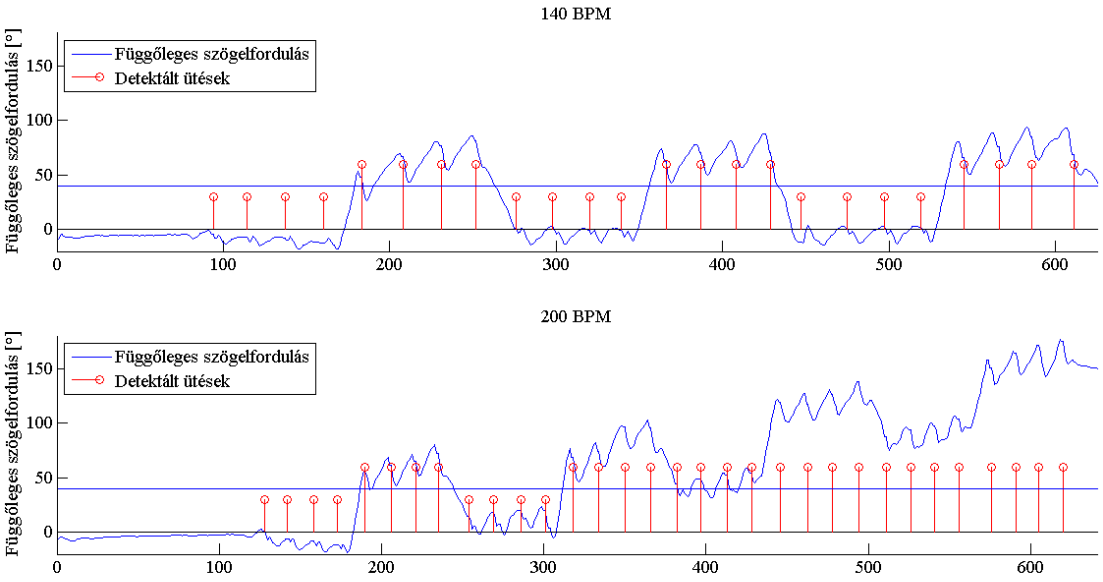
A **8.3 ábrán látható** ez a probléma. A teszt során megdöntöttem az eszközt az X (*pszi*) és az Y tengely (*theta*) körül, majd ebben a helyzetben hagytam. Ez a két tengely kis idő elteltével beáll a 30°-os megdöntéseknek megfelelően, viszont a Z tengely (*phi*) folyamatosan változik.

ST szenzorokkal foglalkozó internetes fórumokon kutatva [15], [16], [17] felfedeztem, hogy másnak is volt hasonló problémája ezzel az eszközzel és a Madgwick algoritmussal. A fórumon azt írták, hogy a problémát a mágneses szenzor hibája okozza. Az X (*pszi*) és az Y tengelyű (*theta*) elfordulásoknál a Madgwick algoritmus nem használja a mágneses adatokat, csak a Z tengely (*phi*) körülieknél.

8.3 Kétállapotú pozícióérzékelés pontosságának vizsgálata

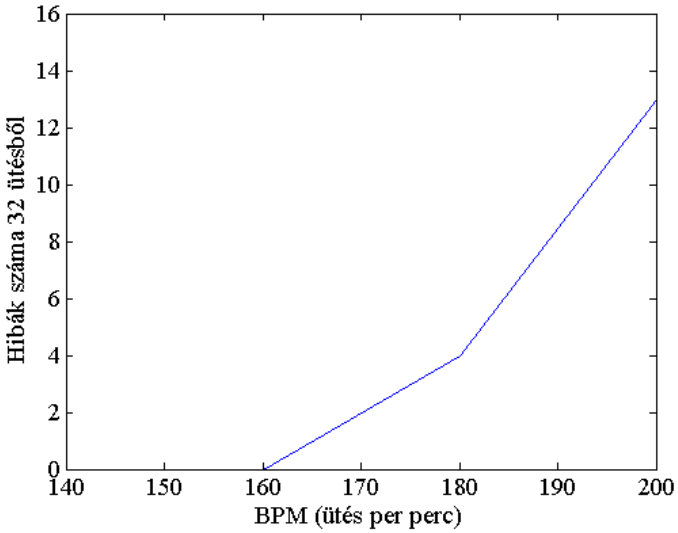
A **8.2 fejezetben** tárgyalt okokból a prototípus jelenlegi változata 2 pozíciót képes megkülönböztetni. Ebben a fejezetben a pozíciómeghatározás pontosságát fogom tesztelni.

A teszt két részből áll. Az első felében térd-térd-térd-térd levegő-levegő-levegő-levegő ritmusmintákat játszottam. A tempót egy metronóm diktálta, egy ütés egy metronómra érkezett.



8.4. ábra: Pozíciódetektálás pontossága 140 BPM és 200 BPM értékeknél, első teszt

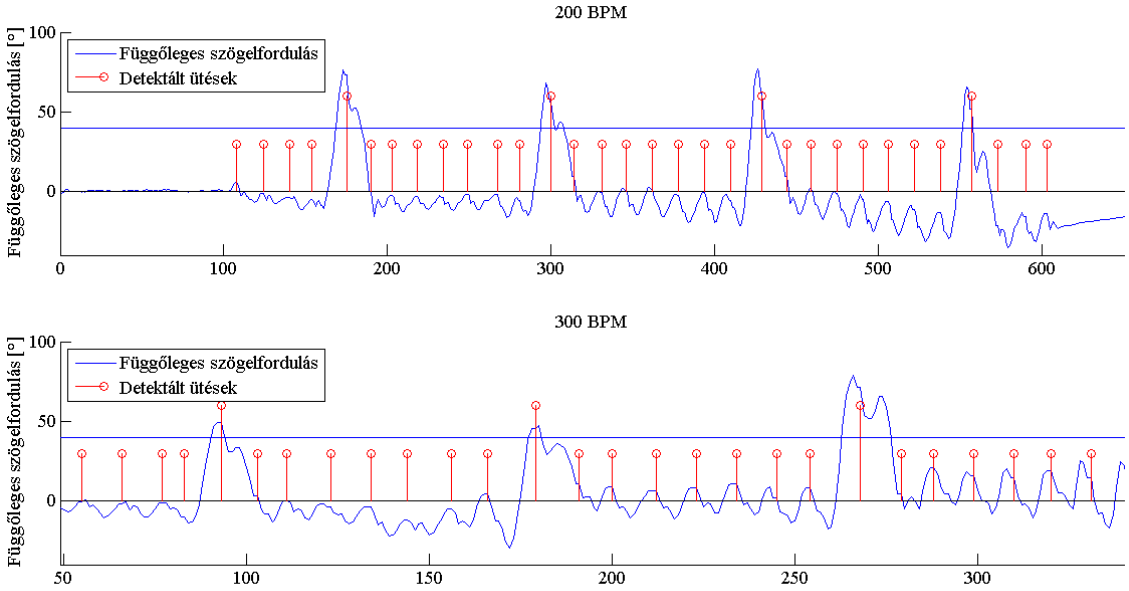
A teszt során 140 BPM (Beats Per Minute, percenkénti ütésszám) volt a legalacsonyabb beállított tempó, innen 200 BPM-ig növeltem a tempót 20-as lépésközzel. A **8.4 ábrán** a két szélsőséges tempóval készített teszt látható. Kék színnel a Madgwick függvény által számolt függőleges szögelfordulás van feltűntetve. A kék vízszintes csík a komparálási szintet jelenti, ha azt ütés pillanatában a függőleges szögelfordulás 40° felett van, akkor az algoritmus felső ütést érzékel, ez alatt alsót. Pirossal pedig azt jelöltem, hogy az ütés alsó vagy felső pozícióban történt. A pirossal jelölt függvény értéke nem felel meg az Y tengely skálázásának, azt szemlélteti, hogy alsó vagy felső ütést detektált a rendszer. 140 BPM mellett a pozíció detektálás 100%-ban pontos, 200 BPM mellett viszont már nem megbízható. Jól látszik, hogy a hibajelenség időfüggő. A 140 BPM-es tempó mellett a függvénynek van ideje visszaállni a 0° függőleges szögelforduláshoz, ezzel szemben 200 BPM mellett már nincs idő a nullpozícióba való visszatérésre.



8.5. ábra: Hibák száma a tempó függvényében

A **8.5 ábra** a köztes tempók tesztjét is figyelembe veszi. A 160 BMP-es tempó még szintén nem okozott hibát,180 BPM-nél már jelentkeztek a problémák, és a hibák száma a tempó növelésével növekszik.

A teszt második felében térd-térd-térd-térd levegő-térd-térd-térd ütésmintát vizsgáltam. Az előző teszttel szemben az a különbség, hogy a kéz sokkal több időt tölt alsó pozícióban, és a levegőben történt ütés után az algoritmusnak van ideje visszaszivárogni a 0° helyzetbe.



8.6. ábra: Pozíciódetektálás pontossága 200 BPM és 300 BPM értékeknél, második teszt

A **8.6 ábra** alátámasztja az előző bekezdésben leírtakat. Bár egy negatív irányú drift itt is jelen van, a függvénynek van ideje visszatérni az alaphelyzetbe, így még 300 BPM tempó mellett is megfelelően adja vissza a pozíciót.

A tesztből kiderült számomra, hogy a pontosság erősen függ attól, hogy a felhasználó mennyi ideig üt a levegőbe. Természetesen nem lehet megmondani neki, hogy csak minden harmadik térd ütés után engedélyezett a levegőbe ütés, a függvényt kell felgyorsítani úgy, hogy a térdre való visszatérés is ugyanolyan sebességgel történjen, mint a kéz felemelése. További feltevésem, hogy ha az előző fejezetben felvetett kalibrációs problémát is megoldanám, akkor javulna a függőleges irányú pozícióérzékelés pontossága is.

9 Összefoglalás és további fejlesztési lehetőségek

Ezen szakdolgozat megírása számomra rendkívül hasznos volt. A legfontosabb szempont, hogy kiderült, hogy az általam kitalált ötlet megvalósítható, működik a való életben is. Ezen kívül használható tudást szereztem olyan területeken, amelyekkel korábban nem vagy csak érintőlegesen foglalkoztam az egyetemi tanulmányaim során. A fejlesztés természetesen nem volt zökkenőmentes. Ahogy a szakdolgozatomból kiderül, volt több olyan mellékvágány, amely végül nem vezetett megoldásra, de segített a prototípus megalkotásához.

Bár a dolgozatban nem kapott jelentős részt, de sok időt töltöttem el az MMA7341LC analóg gyorsulásmérő jeleinek vizsgálatával, még a 2017-es őszi szemeszter elkezdése előtt. A tesztekre megírt MATLAB vizsgálófüggvényeket és egy kezdetleges ütésdetektáló algoritmust fel tudtam használni, amelyeket a későbbiek során. Felismertem a problémát, hogy ennek az IC-nek a méréstartománya kevés lesz az én felhasználásomra, illetve azt, hogy ha hordozható eszközt szeretnék csinálni, akkor digitális kimenetű gyorsulásmérőt kell majd alkalmaznom.

A kísérletezést a LIS3DH szenzorral folytattam. Először meg kellett ismernem az LPC 1768 fejlesztőkártyát, majd összepárosítanom a gyorsulásmérő szenzorral, hogy abból értékes adatokat tudjak kinyerni. Ehhez meg kellett tanulnom az I2C kommunikáció alapjait, illetve kipróbálni az összes lehetséges beállítást, hogy megtaláljam a számomra megfelelő kombinációt. Ennek a részfolyamatnak a végén elkészült a végleges szintérzékelő algoritmus, amely már úgy működött, ahogy szerettem volna. A termék ebben az állapotában is jelentős felhasználói élményt tudott nyújtani, szimplán azzal, hogy fejhallgatón meg tudtam hallgatni egy virtuális dob hangját, amely természetesnek ható módon érzékelte az ütések erősségét.

Azonban az célkitűzésem az volt, hogy a kezeken lévő eszközök többféle virtuális dobot tudjanak megszólaltatni, ezért folytattam a fejlesztést. A gyorsulásmérő szenzor adataiból különféle módszerekkel számítottam sebességet és pozíciót, de ezek a módszerek nem bizonyultak használhatónak, ezért tovább kutattam az interneten már meglévő megoldások után.

Egy félkész termék adatlapjáról jött az ötlet, amely megtalálható a szakdolgozatban. Ez a felfedezés a pozíciószámítást teljesen új alapokra helyezte. A pozíciót nem közvetlenül a gyorsulási adatokból számítottam, hanem az eszköz vízszintes és függőleges szögelfordulásából, amely egyértelműen meghatározza az eszköz pozícióját is. Ehhez kellett egy új IC, amely a gyorsulásmérőn kívül mágneses szenzort és giroszkópot is tartalmaz. Az LSM9DS0 megfelelt ezeknek a követelményeknek. Mivel ugyanaz a gyártója, mint a korábban használt LIS3DH-nak, és mivel a korábban használt IC-vel tapasztalatot szereztem a regiszterek beállítása terén, az eszköz üzembe helyezése a korábbi szenzor több hetes üzembe helyezési ideje helyett néhány napba telt.

A következő lépés a megfelelő algoritmus megtalálása volt, amely ezekből az adatokból meg tudja határozni a vízszintes és a függőleges szögelfordulást. Hamar megtaláltam

Sebastian Madgwick nyílt forráskódú algoritmusát, amely jelenleg a legelterjedtebbnek számít orientáció meghatározás terén. Ezután az volt a feladat, hogy összerakjam a rendszert az alkotóelemekből, és futtassam a fejlesztőkártyán a korábban megírt és az új algoritmusokat. Ahhoz, hogy használhatóvá tegyem, meg kellett ismerkednem a Madgwick algoritmus matematikai hátterével, és kiegészítenem a kódot, hogy a számomra megfelelő értékeket adja vissza.

A fejlesztés vége felé felmerült egy probléma, amely driftet okozott az eszköz vízszintes irányú szögelfordulásában. Bár a megoldást elméletileg megtaláltam hasonló problémákkal foglalkozó internetes fórumokon, gyakorlatban már nem tudtam tesztelni. A rendszer így is működőképes és ebben az állapotában is használható az általam kitűzött célra, bár ahhoz, hogy piacképes termék legyen belőle, még sok fejlesztés van hátra.

A prototípusról kép a **Függelék 1. ábráján** látható, a tesztelésről készült rövid videó pedig megtekinthető a **Mellékletben**.

Az LPC1768 fejlesztőkártya jelenleg egy darab kézre és egy darab lábra csatolható Slave eszköz fogadására és jeleinek feldolgozására képes, a kéz eszköz pedig többnyire megbízhatóan érzékeli a függőleges szögelfordulást. A cél természetesen az lenne, hogy mind a négy végtaggal dobokat lehessen megszólaltatni, és működjön a pontos függőleges és vízszintes szögelfordulás meghatározás is. A lábakra LIS3DH gyorsulásmérő szenzort használnék, ahogy a mostani megoldásban is van, a kezekre pedig LSM9DS0 inerciális szenzor kerülne.

A végső megoldás vezeték nélküli kommunikációt használna. Az egyes egységek tartalmaznák a megfelelő szenzort, kommunikációs egységet, illetve egy feldolgozó egységet is. A hordozható kivitelhez ezeken kívül szükség van az akkumulátorról való tápellátás biztosítására.

A feldolgozást azért helyezném át az egységekre, mert így csökkenteni lehet a vezeték nélküli kommunikáción elküldött adatok számát. Egyrészt minimalizálni lehet a kommunikáció hibájából adódó adatvesztést és csökkenteni lehet a késleltetést, másrészt a fogadó eszköz a felhasználó okostelefonja lenne, és így nem vonna el erőforrást a processzortól. A Bluetooth kommunikáción csak a MIDI üzenetek kerülnének kiküldésre. A piacon fellelhető okostelefonok nagy része támogatja ezt a szabványt, így a virtuális dob eszköz könnyen párosítható lenne már meglévő MIDI lejátszó alkalmazásokhoz.

A jövőben ezen gondolatok mentén fogom folytatni az eszköz fejlesztését addig, ameddig úgy érzem, hogy sikerült egy piacképes eszközt létrehoznom.

Köszönetnyilvánítás

Szeretném itt megköszönni a konzulensem, Dr. Bank Balázs munkáját, aki a folyamat során legalább annyira lelkes volt, mint én, és nem engedte, hogy egy-egy akadálynál feladjam az eredeti elvárásaimat, hogy félkész munkát adjak be.

Illetve szeretném megköszönni Vadász Istvánnak a beágyazott program megírásával kapcsolatos segítségét, nélküle sokkal lassabb ütemben tudott volna haladni a fejlesztés és nem jutott volna el olyan szintre, amelyen most van.

Irodalomjegyzék

[1] Xiaoping Yun, Eric R. Bachmann Robert B. McGhee, ”A Simplified Quaternion-Based Algorithm for Orientation Estimation From Earth Gravity and Magnetic Field Measurements”, *IEEE transactions on instrumentation and measurement*, vol. 57, no. 3, pp.638 – 642,March 2008.

[2] Android Developers: Sensor Manager Tools, API level 1, URL: https://developer.android.com/reference/android/hardware/SensorManager, Letöltve: 2017. máj.

[3] Kovács Adél: Kvaterniók és forgatások, Szakdolgozat, Eötvös Lóránd Tudományegyetem Természettudományi kar, pp. 4-15, 2014.

[4] Hugo Costelha, Pedro A. Amado Assunção, Luis Conde Bento*:* ”Low-complexity MARG Algorithms for Increased Accuracy in Space Pointing Devices”, *IEEE 1st International Workshop on Consumer Electronics*, pp. 1-5, Portugal, March 2015.

[5] C++ Algorithms, URL: http://www.cplusplus.com/reference/cmath/atan2/, Letöltve: 2017 máj.

[6] Sebastinan Madgwick: Open Source IMU and AHRS Algorithm with x-IMU video, URL: https://www.youtube.com/watch?v=BXsGWoOMtmU

[7] K. Shaeffer: ”MEMS Inertial Sensors: A Tutorial Overview”, *IEEE Communications Magazine*, pp. 100-103, April 2013.

[8] LSM9DS0 iNEMO inertial module datasheet, URL: http://www.st.com/en/mems-and-sensors/lsm9ds0.html, Letöltve: 2017 máj.

[9] ARM mbed Handbook of Algorithms, URL: https://developer.mbed.org/handbook/USBMIDI, Letöltve: 2017 máj.

[10] Sebastian O.H. Madgwick, ”An efficient orientation filter for inertial and inertial/magnetic sensor arrays”, Ph.D., University of Bristol, pp. 2-13, April 2010.

[11] Jean-Marc Irazabal, Steve Blozis*:* I2C Communication Application Note, pp. 2-17, March 2003.

[12] EZ Drummer VSTi hivatalos honlap, URL: http://www.toontrack.com/ezdrummer-line/, Letöltve: 2017 május

[13] Sík – Gerényi: MIDI Alapozás és protokoll, p.61, 2006.

[14] Teemu Mäki-Patola, Perttu Hämäläinen, ”Latency Tolerance for Gesture Controlled Continuous Sound Instrument without Tactile Feedback”, Project ALMA, Helsinki University of Technology, 2004.

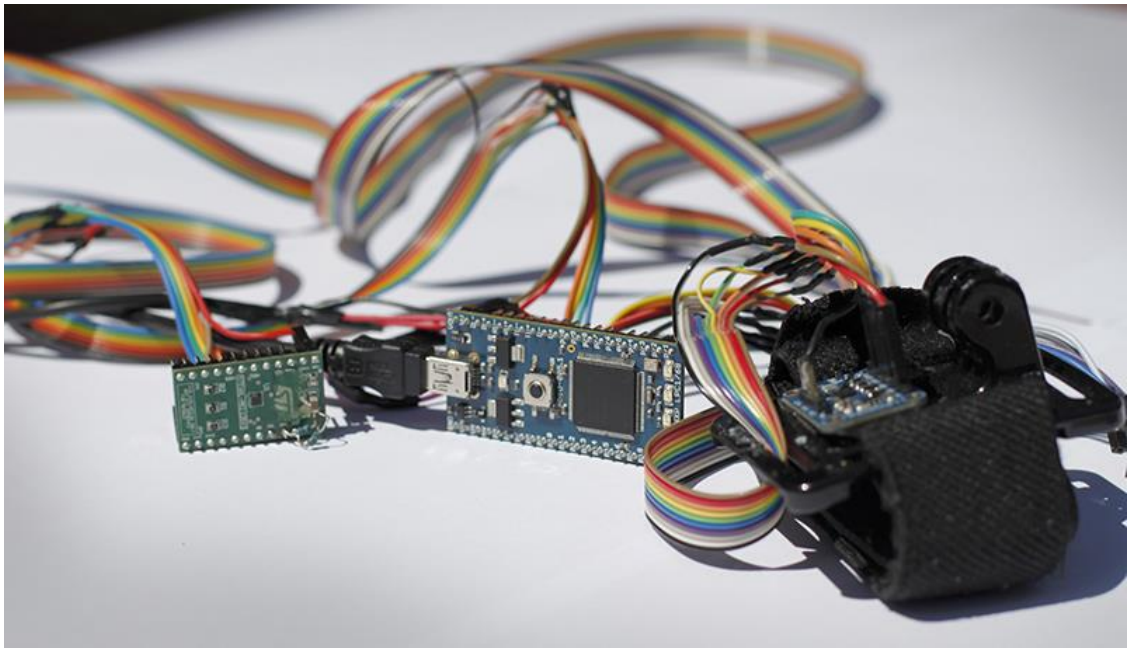
[15] Stack Exchange Electrical Engineer Forum, Drift of Madgwick Algorithm,
URL: https://electronics.stackexchange.com/questions/186315/why-am-i-seeing-drift-when-using-madgwick-algorithm-to-correct-for-orientation-w,
Letöltve: 2017 május

[16] DIY Drones Community, Yaw drift with MadgwickAHRS,
URL: http://diydrones.com/forum/topics/yaw-drift-with-madgwickahrs,
Letöltve: 2017 május

[17] Arduino Forum, Madgwick filter algorithm for IMU,
URL: https://forum.arduino.cc/index.php?topic=225591.0,
Letöltve: 2017 május

Függelék

A rendszer tartalma



Függelék 1. ábra: A rendszer tartalma: láb egység, feldolgozó egység, kézi egység

- NXP LPC1768 mikrokontroller (feldolgozó egység)
- ST LSM9DS0 inerciális modul IC hozzá tartozó breakout boarddal (kézi egység)
- ST LIS3DH gyorsulásmérő IC hozzá tartozó breakout boarddal (láb egység)

Vezetékezés az LSM9DS0 inerciális szenzor és a mikrokontroller között

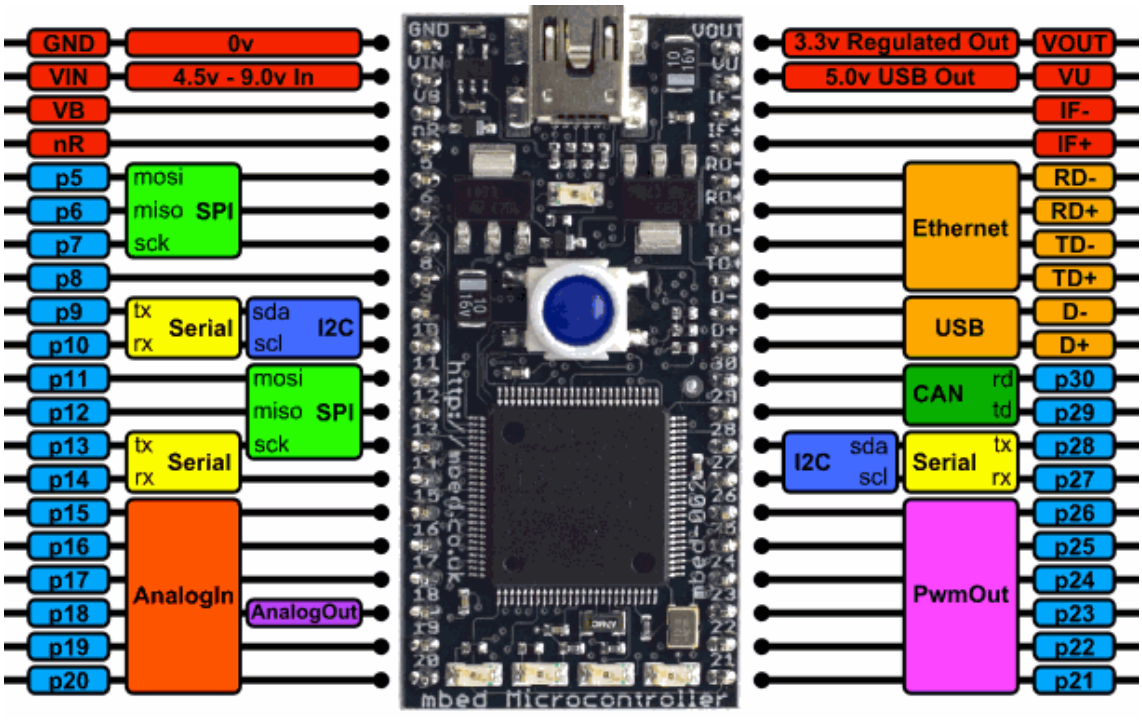
LSM9DS	kontroller	szerep
VIN	VOUT	tápellátás
GND	GND	föld
SDA	p9	I2C adat vezeték
SCL	p10	I2C órajel
INT1	p11	DRDY interrupt

Vezetékezés az LIS3DH gyorsulásmérő szenzor és a mikrokontroller között

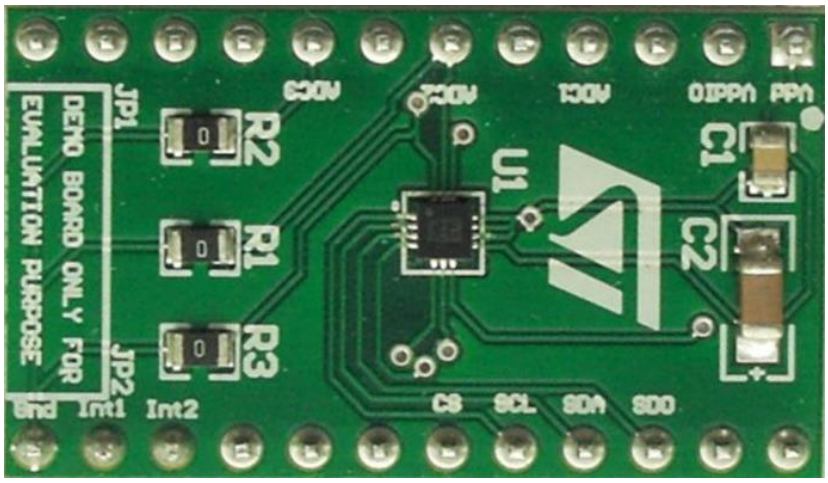
LIS3DH	kontroller	szerep
VIN	3.3V OUT	tápellátás
GND	GND	föld
SDI	p5	SPI MOSI jel
SDO	p6	SPI MISO jel
SCL	p7	SPI órajel
CS	p8	SPI CS jel

További vezetékek

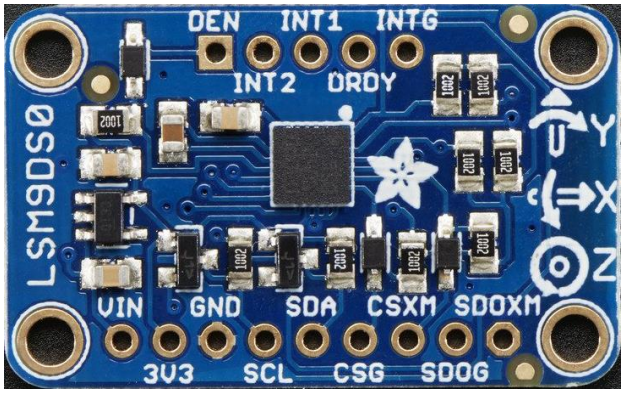
mikrokontroller	számítógép	szerep
USB csatlakozó	USB csatlakozó	soros kommunikáció
D+,D-	USB csatlakozó	MIDI jelek küldése



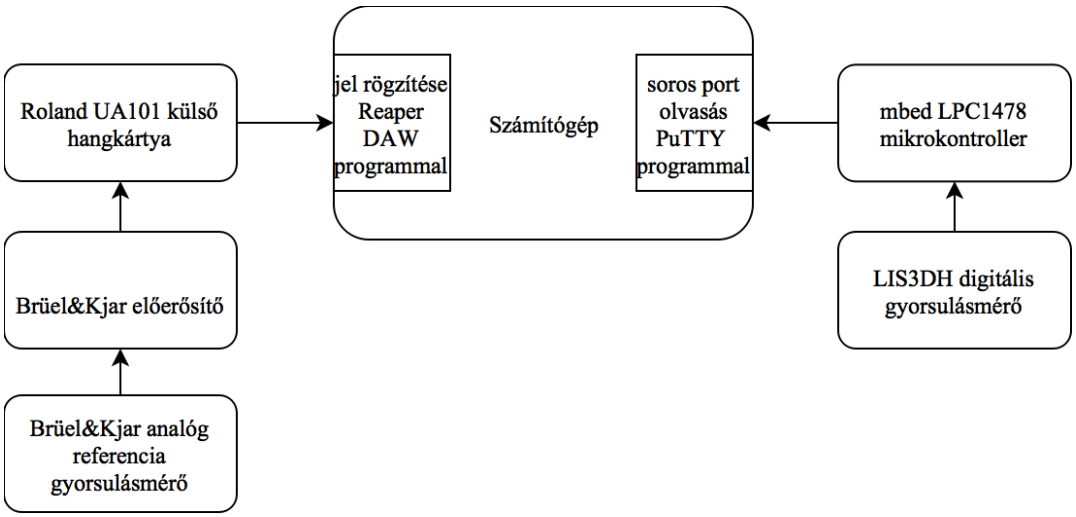
Függelék 2. ábra: Az mbed LPC1768 fejlesztőkártya ki-és bemeneti lehetőségi



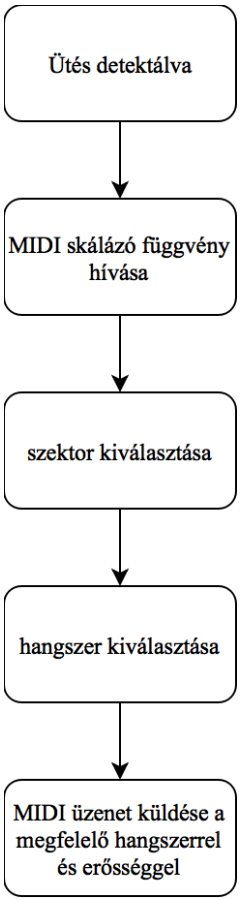
Függelék 3. ábra: LIS3DH és a hozzá tartozó breakout board



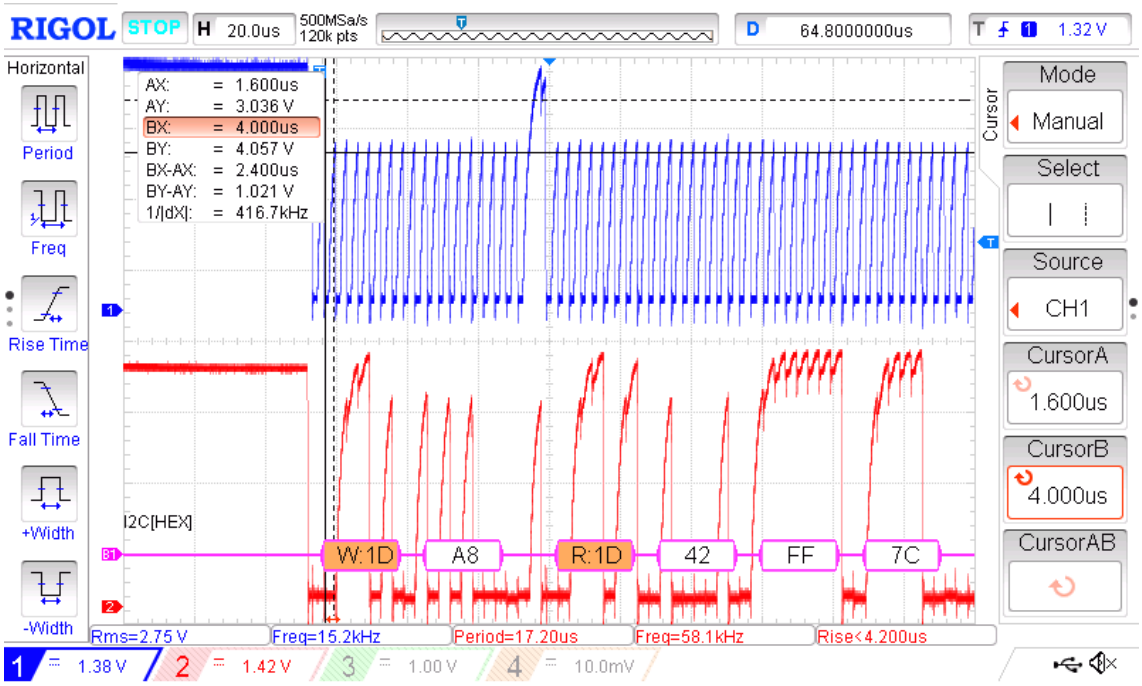
Függelék 4. ábra: LSM9DS és a hozzá tartozó breakout board



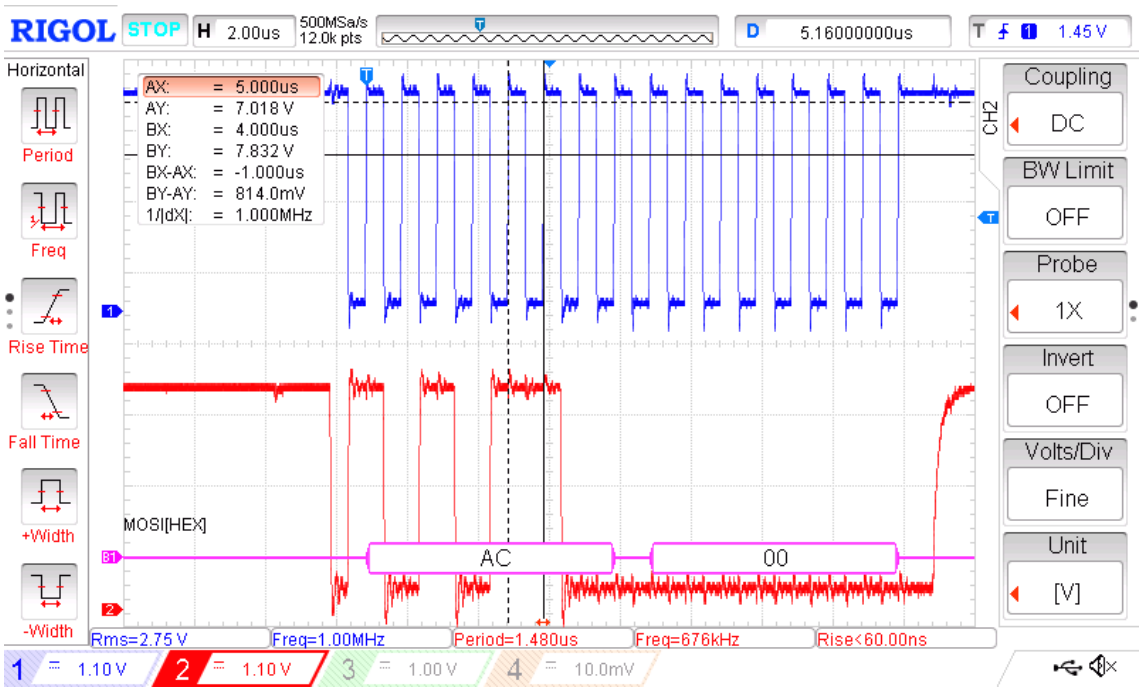
Függelék 5. ábra: Mérési elrendezés a 3.1 és 3.2 tesztjeihez



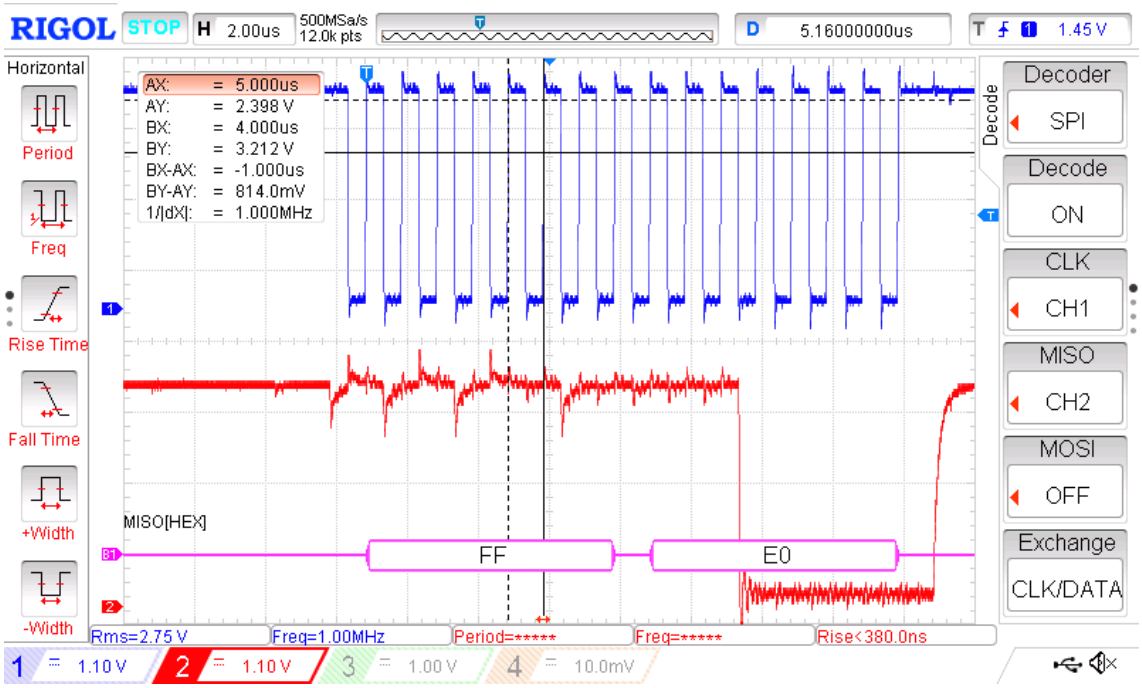
Függelék 6. ábra: Szintérzékelő algoritmus részletezve



Függelék 7. ábra: Írás és több byte olvasása I2C kommunikáción keresztül



Függelék 8. ábra: Adat kérése SPI kommunikáción keresztül, MOSI vonal



Függelék 9. ábra: Adat olvasása SPI kommunikáción keresztül, MISO vonal