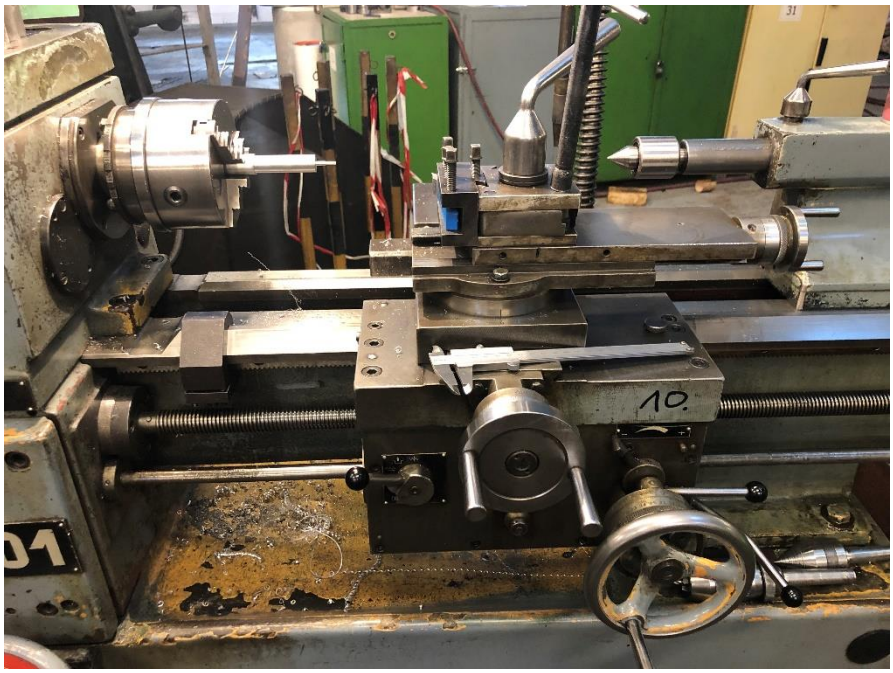
Önálló laboratórium
Dokumentáció

Név, neptun kód: Jánoki Csaba, AELQUH
Téma megnevezése: IoT - Adatgyűjtés, adatelemzés ipari környezetben
Konzulensek: Kovács László, Fábián István

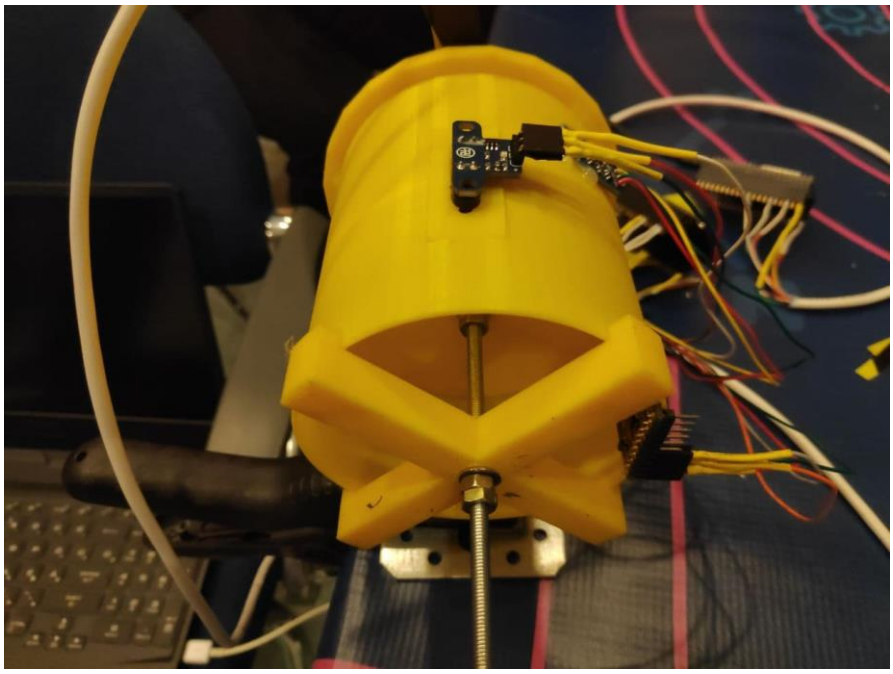
1. Feladat bemutatása

A feladat lényege az volt, hogy ipari folyamatokat tudjunk megfigyelni, azokról adatot gyűjtsünk, majd wifi kapcsolaton keresztül elküldjük egy adatbázisba. Ez több szempontból is hasznos lehet, egyrész közel valós időben (kis késleltetéssel) tudjuk figyelni az eszköz állapotát, működésének adatait. Másrészt az adatokat az adatbázisban elmentve az adatok elemezhetőek, és ezek alapján például javaslatokat lehet tenni a gép átalakítására/ továbbfejlesztésére az esetleges működési hibák kijavítására, vagy hatékonyabb működés elérésére.

Az elkészült mérőrendszer eredeti tesztalanya az *1. ábrán* látható, BME gépészmérnöki karához tartozó CNC esztergagép lett volna, viszont a koronavírus járvány miatt nem volt lehetőségünk hozzáférni ehhez, ezért a tényleges tesztelés egy 3D nyomtatási technológiával készült „eszterga szimulátoron" történt (*2. ábra*).



1. ábra



2. ábra

2. Szenzorok

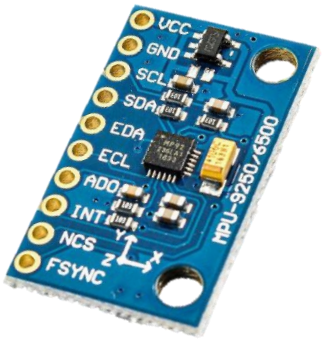
A projekt egyik legfontosabb alkotóelemei maguk az elhelyezett szenzorok, ugyanis ezek segítségével tudjuk elvégezni az egyes méréseket, a környezetünk különféle paramétereit villamos jelekké alakítva, majd ezeket mikrokontroller segítségével feldolgozva.

A projektben szerepet kapott egy hőmérő, mikrofon, fordulatszámmérő, árammérő, valamint egy rezgésmérő. Ezek közül én a rezgésmérő elkészítésével foglalkoztam behatóbban.

2.1 Rezgésmérő

2.1.1 Szenzor adatai

A feladatra egy MPU9250-es eszköz került kiválasztásra. Ez egy multifunkciós szenzor, képes 3 tengelyes gyorsulásmérésre, található benne ezen kívül giroszkóp, magnetométer, és még hőmérő is. Ezek közül a funkciók közül a gyorsulásmérő került használatra a rezgésmérés megvalósításához. Maga a szenzor a *3. számú ábrán* látható.



3. ábra

Az eredeti cél az lett volna, hogy 10 kHz-es mintavételi frekvenciával minta vételezzük az aktuális gyorsulásértéket, vagyis 100 µs-ként lett volna új adatunk. Ez azért lett volna kedvező, mivel a mikrofon is hasonló jelleggel kell kezelni, mint a gyorsulásmérőt, és azt 10 kHz-el minta vételezzük. Mivel a használt mikrofon analóg kimenettel rendelkezik, ezért minden időpillanatban rendelkezésre áll a kimenetén új adat, korlátot csak a mikrokontroller AD alakítója szabhat, ami viszont maximum 10 kHz-ig üzemképes.

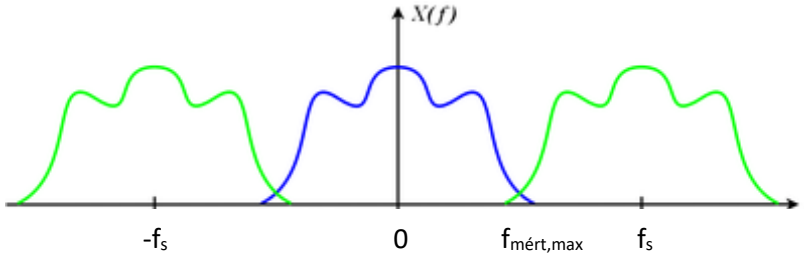
Sajnálatos módon a gyorsulásmérő szenzor nem volt képes ilyen gyakran mintákat szolgáltatni, az adatlapi értékek szerint a maximális adatszolgáltatási sebessége (Data output rate) csupán 4 kHz. Viszont az eszköz ezt a mintavételi frekvenciát csak egy speciális üzemmódban képes elérni. Több GitHub-os könyvtár is kipróbálásra került, amelyek konkrétan ehhez a szenzorhoz íródtak, viszont az összes maximum az 1 kHz-es mintavételezést tette lehetővé.

Mélyebben elmerülve az eszköz regiszter kiosztásában az volt látható, hogy a 4 kHz-es üzemmód abban az esetben érhető el, ha az eszköz 29-es számú regiszterének 3-as bitjébe 0 érték kerül beállításra. Ez megpróbálásra került, különféle bitkombinációkkal, de az eszköz továbbra sem szolgáltatott 1 kHz-nél gyakrabban adatokat. Internetes fórumok esetén azt is írták, hogy a 4 kHz csak akkor érhető el, ha minden egyéb funkció (pl. giroszkóp, magnetométer) kikapcsolásra kerül, viszont ez sem hozta a kívánt eredményt, így rengeteg próbálkozás után végül a szenzor 1 kHz-es gyakorisággal szolgáltat a kimenetén új, hasznos gyorsulásadatot.

A Nyquist-féle frekvencia feltétel értelmében a mintavételi frekvenciának minimum kétszeresének kell lennie, mint a jel sávszélessége, különben spektrumszivárgás alakulhat ki, és az meghamisíthatja a tényleges mérési eredményeket. Ennek értelmében, mivel jelenleg a mintavételi frekvencia 1 kHz, ezért a tényleges mérési jeltartomány csupán ennek a feléig, 500 Hz-ig tart. Annak érdekében, hogy ne következhessen be spektrum átlapolódás, az eszköz

rendelkezik egy változtatható törésponti frekvenciájú aluláteresztő szűrővel, ezzel megszabadulva a nagyobb frekvenciás, nem kívánatos spektrumösszetevőktől. Mint az a *4. ábrán* is látható, ha a mérési tartományt kiterjesztenénk az f_s felélél nagyobbra, akkor bekövetkezne a nemkívánt spektrumátlapolódás.

$$f_s = 1000\ Hz \implies f_{mért,max} = \frac{f_s}{2} = \frac{1000}{2} = 500\ Hz$$



4. ábra

Az eszköz rendelkezik többféle, programozással állítható méréshatárral. Ezek a $\pm 2g,\ \pm 4g,\ \pm 8g,\ \pm 16g$. Elsőként az utóbbi beállításon került tesztelésre az eszköz, viszont mivel úgy tűnt, hogy különféle mérési esetek közül egyikben sem került mérésre 2 g-nél nagyobb abszolútértékű érték, ezért a $\pm 2g$ konfiguráció célravezetőnek tűnt, ugyanis így sokkal jobban kihasznált a mérési tartomány, és nagyobb pontossággal lehet a méréseket elvégezni.

Megjegyzésként annyit hozzáfűznék, hogy kézben erőteljesen rázva a mért érték megközelítette a 16 g-t, viszont a tényleges tesztelési körülmények között ez az érték sosem állt fent, ugyanis a mérendő berendezések rezgése sok esetben meglehetősen intenzív, viszont az amplitúdójuk nem annyira, éppen ezért felesleges a nagyon széles méréshatár.

2.1.2 Kommunikáció a mikrokontrollerrel

Az eszköz kétféle kommunikációs protokollt támogat, az I2C-t és az SPI-t. Mindkét megoldás működőképes lehet, és tesztelésre is kerültek, mielőtt a véglegesen használ ki nem lett választva.

Végül az I2C lett kiválasztva. Első sorban azért, mivel kevesebb vezetéket igényel a kapcsolat a szenzor és a mikrokontroller között, más jelentős különbség pedig a projekt elején nem volt tapasztalható, és éppen ezért a célnak az előzetes tesztek alapján tökéletesen megfelelőnek tűnt. A szenzor adatlapja szerint képes támogatni a 400 kHz-es adatátviteli sebességet, ez viszont nagyságrendekkel nagyobb, mint amire a feladat szempontjából egyáltalán szükség van.

Később viszont kisebb probléma adódott a kommunikációval. Mivel eredetileg az eszterga gépre lett szabva a mérővezeték, ezért a hossza kb. 2 méter volt. Ez viszont sajnos azt eredményezte, hogy a mérési adatok hibásan érkeztek meg,

és 20% körüli mérési pontatlanság volt tapasztalható. Mivel az I2C kommunikáció nem ilyen hosszú távolságokra lett kifejlesztve, ezért nem meglepő, hogy ezzel kapcsolatban problémák adódtak. Egyes források szerint az I2C maximálisan néhány méter vezetékkel használható, viszont a mi esetünkben a 2 méter is már jelentős hibát okozott. Mivel ez a projekt elején még nem volt látható, ezért már korábban beforrasztásra kerültek az I2C vezetékek.

A végleges tesztelés jelen esetben nem az eredeti gépen történt, hanem a kis 3D nyomtatott eszközön, ezért a kábelhossz megkötések nélkül csökkenthető volt, ezzel kiküszöbölve a problémát. Végül a mérővezeték hossza 30 cm körüli érték lett, és ezzel a módszerrel ugyanazok az eredmények voltak mérhetőek, mint a néhány centiméter hosszú jumper vezetékekkel.

Célszerű lett volna az SPI kapcsolatot is megpróbálni a hosszú mérővezetékeken keresztül, viszont mivel ez a probléma csak a félév vége felé derült ki, ezért már nem volt lehetőség mindent újra forrasztani, és letesztelni, főleg annak fényében, hogy nem teljesen biztos, hogy ez megoldotta volna a problémát. Bizonyos források szerint az SPI kapcsolat nagyjából maximálisan 10 méteres távolságra használható, más források szerint viszont az I2C még nagyobb távolságra képes, mint az SPI. Nyilván ezen kérdés eldönthető a tényleges mérések elvégzésével.

Megjegyzésként annyit tennék hozzá, hogy a vezetékcsökkentés az alkalmazott hőmérőszenzor esetében a hosszcsökkentés elkerülhetetlen lett volna, ugyanis az csak és kizárólag I2C kommunikációra volt képes.

2.1.3 A rezgésmérérő szoftverének implementációja

A szoftver elsőként egy ESP8266 (NodeMCU v3) mikrokontrollerre került megírásra, viszont a végleges projektben egy erőteljesebb mikrokontrollerre, egy ESP32-re került áthelyezésre. Szerencsére ez minimális mértékben érintette a projektet, vagyis jó volt a kód hordozhatósága.

2.1.3.1 MPU9250 könyvtár

Az alkalmazott szenzorhoz rendelkezésre áll több, előre megírt függvénykönyvtár, amelyek segítségével lebonyolíthatók a különféle szükséges műveletek a mikrokontroller és a szenzor között (pl. adatfogadás), így nem kell ezeket regiszter szinten közvetlenül elvégezni. Ezek közül több is kipróbálásra került, és végül az asukiaaa nevű GitHub felhasználó MPU9250 könyvtára került kiválasztásra. Sok függvény áll rendelkezésre, amelyek gördülékennyé teszik a használatot, például közvetlenül megkapható mindhárom tengelyen mért gyorsulásértékekből kiszámított eredő gyorsulás.

A setup ciklusban inicializálásra kerül a gyorsulásmérő funkció. A mySensor változó tartozik magához a szenzorhoz, és ezen keresztül indul el itt a gyorsulásmérés. Paraméterként megkapja az ACC_FULL_SCALE_2_G macro-t, ami a korábban tárgyalt 2g-s méréshatárt állítja be. Ezen kívül az aluláteresztő szűrő törésponti frekvenciáját és a mintavételi frekvenciát nem szükséges explicit beállítani, ugyanis azok alapból is a maximális beállításon kerülnek inicializálásra, és pontosan erre van jelenleg szükség.

2.1.3.2 FFT könyvtár

A szenzor aktuális értékét kiolvasva egy adott időpillanathoz tartozó mérési eredményt kapunk. Ebből egy időtartománybeli jel kapható meg. Jelen esetben szükség van arra, hogy ezt a frekvenciatartományba átültetésre kerüljön, ugyanis a cél az, hogy meghatározzuk az aktuálisan mérendő rezgés frekvenciakomponenseit (spektrumát).

Ezen feladat elvégzéséhez jól használható az arduinoFFT nevű, szintén GitHub-on elérhető függvénykönyvtár. Mint ahogy a nevéből is látható, ez kifejezetten mikrokontrolleres felhasználásra készült, így a jelenlegi projektben is jól használható. A szenzor könyvtárához hasonlóan itt is nagyon sok hasznos függvény található, amik segítségével kiszámítható egy adott adathalmaz Fourier transzformáltja, vagy például megkereshető a legnagyobb csúcs (legdominánsabb frekvencia komponens).

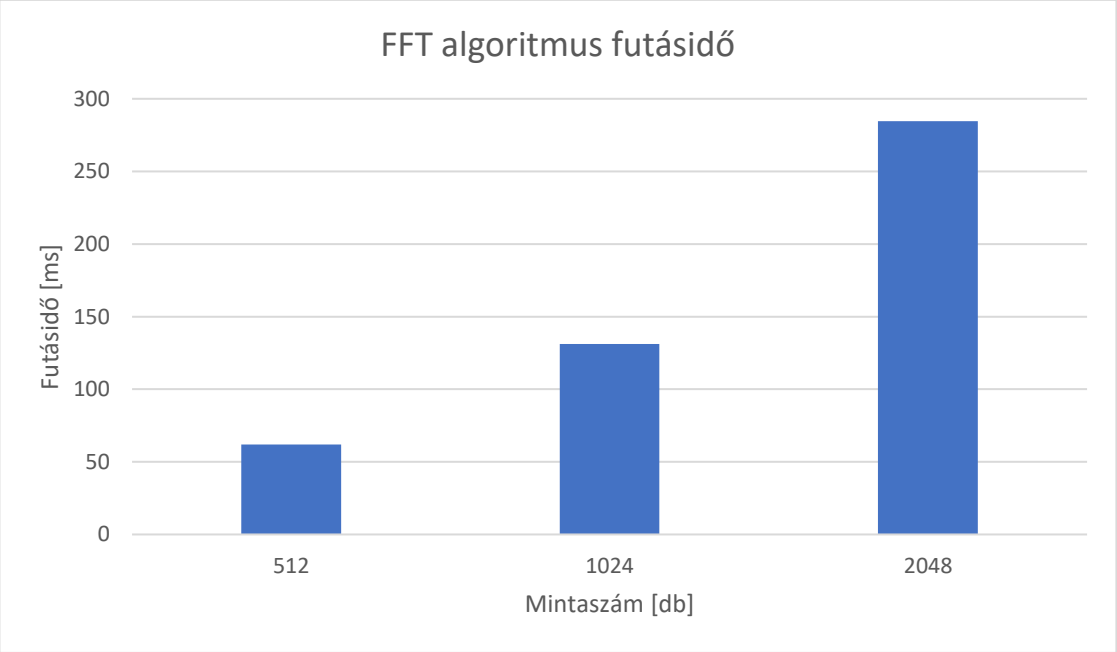
Fontos kitétel, hogy a mintaszámnak 2 egész számú hatványának kell lennie, ugyanis az FFT algoritmus csak ilyen esetekben képes hibátlanul működni. E mellett arra is szükség van, hogy a teljes adattömb rendelkezésre álljon a transzformáció kezdetekor, az nem megfelelő, hogy elindítjuk az algoritmust és a folyamat közben még adatokat töltünk a bufferbe.

2.1.3.3 Mintavételezés megvalósítása

Fontos kérdés elsőként azt eldönteni, hogy mennyi mintát vegyünk. Ez egy meglehetősen összetett kérdés, sok mindentől függ, hogy mit érdemes választani. A mikrokontroller memóriája nem túl nagy, éppen ezért nem lehet akármekkora mintaszámot választani, ugyanis ezeket az értékeket szükséges eltárolni. Másrészt, ha túl sok mintát veszünk, akkor azon nagyon sokáig fog futni az FFT algoritmus, ezzel pedig csökken a végleges adatkiadás gyakorisága. Ezzel szemben minél több mintát veszünk, annál precízebben, nagyobb felbontással tudjuk a frekvenciaértékeket meghatározni, ugyanis az 1 kHz erre az N db mintára képződik le, és kevesebb minta esetén nagyobb lesz köztük a lépték, ezzel pontatlanabbá téve a mérést.

Jelen esetben 1024 minta került kiválasztásra, ugyanis ez tűnt optimális megoldásnak, ekkor ugyanis nem fut tolerálhatatlanul sokáig az FFT algoritmus,

és ennyi adat belefér a mikrokontroller memóriájába is. A *5-ös ábrán* látható egy diagram, ami az ESP8266-on történő FFT algoritmus futásidejeit ábrázolja különböző mintaszámok esetén.



5. ábra

Az eszköz rendelkezik beépített flash memóriával, amit EEPROM-ként lehet használni. Ebben jelentősen nagyobb tárhely áll rendelkezésre, ezért kipróbálásra került, hogy ide írjuk a mérési eredményeket, ezzel sokkal nagyobb adathalmazt előállítva. Sajnos az EEPROM írási művelet nagyon lassú, ezért jelen feladat során nem alkalmazható ez a megközelítés.

Korábban már szó esett arról, hogy a kiválasztott mintavételi frekvencia 1 kHz, ez alapján a mintavételi időköz 1 ms, ami azt jelenti, hogy ilyen gyakran kell kiolvasni a szenzorból az aktuális gyorsulásértéket. Addig kerülnek kiolvasásra és a tömbbe betöltésre az adatok, amíg mind az 1024 hely feltöltésre nem kerül.

A micros() függvény visszatérési értéke megadja a mikrokontroller egy speciális memóriaszegmensének értékét, ami az MCU indítása óta eltelt minden mikroszekundumban megnöveli a számláló értékét 1-gyel. Kezdetben lementjük az aktuális értékét a previousMicros változóba. Ezek után akkor vesszük az első mintát, ha a micros() aktuális értéke nagyobb, mint a previousMicros változó és a samplingTime konstans összege. Ha teljesül a felvétel, akkor kiolvasásra kerül az új gyorsulásérték a szenzorból, majd eltárolásra kerül a tömbben. Ezek után a previousMicros változó értéke megnövelésre kerül a mintavételi idő értékével, és innentől kezdve ez ismétlődik a tömb feltöltődéséig.

2.1.3.4 FFT elvégzése

Miután rendelkezésre áll a teljes adathalmaz, elvégezhető rajta az FFT algoritmus. Elsőként egy ablakozási műveletet kell végrehajtani az adathalmazon. Erre azért van szükség, mert az FFT algoritmus az időtartományban megtalálható jelet az első mintától az utolsóig vizsgálja, és úgy tekinti, mintha ez a mintasorozat a végtelenségig ismétlődne. Éppen ezért a tömbhatáron, ha nem egész számú periódust vettünk a jelből, akkor lesz benne egy ugrás (törés), és ez a végeredményben egy valójában nem létező nagyfrekvenciás zavarként jelenne meg. Megfelelő ablakozással ez a hatás enyhíthető. Többféle típusú ablakozás érhető el, ezek közül a legcélravezetőbb a Hamming ablakozás lehet, ugyanis ez az egyik lehető legjobb kompromisszumos megoldást tudja nyújtani koherens és nemkoherens mintavételezés során egyaránt.

Az ablakozás elkezdődhet a tényleges algoritmus elvégzése az adathalmazon. Miután kiszámításra kerültek az egyes komponensek, azok az eredeti tömbbe kerülnek visszatöltésre. Így a függvényből kilépve a bemeneti buffer tartalmazza a végleges frekvenciaspektrumot. Minden tömbelem $\frac{1000\,[Hz]}{1024-1} \cdot n$ frekvencia értékhez tartozik, ahol n jelöli az aktuális indexet, vagyis a legelső a DC komponens, míg az utolsó tartozik az 1 kHz-hez. A korábban tárgyalt okok miatt csak a mintavételi frekvencia feléig tudunk értékeket hasznosítani, így a feldolgozás során csupán az 512. indexű mintáig kell elhaladni.

A Fourier transzformáció a bonyolult időfüggvényeket elemi szinuszhullámok összegére bontja fel, jelen esetben is ez történik. A tömbben adott indexen tárolt értékek a fent leírt frekvenciához tartozó gyorsulás amplitúdó értékét tárolja. Illetve ez nem teljesen igaz, ugyanis az algoritmus sajátossága, hogy a tömb elemeit szükséges a mintaszámmal lenormálni (vagyis leosztani 1024-gyel), és akkor kapható meg a tényleges, adott frekvenciához tartozó gyorsulásérték. Ennek a legnagyobb csúcs megtalálása szempontjából nincs jelentősége, mert egymáshoz való viszonyulása nem fog megváltozni az összetevőknek, viszont a spektrum ábrázolásakor (lásd később) fontos a skálázás.

2.1.3.5 Legnagyobb csúcsok megkeresése

Azért van szükség erre a funkcióra, mivel az adatokat szeretnénk egy MQTT szerverre felküldeni. Nyilván a felküldés is hosszabb időt venne igénybe nagyon sok adat továbbítása esetén, viszont ezek után még az is problémát jelentene, hogy ezt a rengeteg adatot el kell tárolni valahol. A teljes spektrum felküldése nagyon pazarló megoldás lenne, sok esetben értelmetlen is, mert általában sok olyan frekvencia van, ahol nem mérhető semmilyen érdemi érték.

Az alkalmazott FFT könyvtár tartalmaz beépített függvényt arra az esetre, ha a legnagyobb spektrumértéket kell megkeresni, viszont olyan, amely az N db

legnagyobb csúcsot keresné meg nincs, ezért ez külön lett implementálva a topPeaks függvényben.

Ez a függvény bemenetként vár egy tömböt, amiben meg kell keresnie a 10 legnagyobb értéket, valamint egy olyan tömböt, amibe el tudja tárolni ezeket a megtalált értékeket. Ezen kívül a mintavételi frekvenciára is szükség van. A függvény nem feltétlenül csak a 10 legnagyobb értéket tudja megkeresni, nyilván ez az érték a topPeakNum változó értékének átírásával tetszés szerűén módosítható, egészen a mintavételi szám feléig (ugyanis nincs több hasznos adat).

Végig lépkedve a tömb elemein az algoritmus le ellenőrzi, hogy az aktuális tömbelem nagyobb-e, mint a korábban megtalált legnagyobb. Ha újabb legnagyobb értéket talál, akkor magát az értéket is elmenti, hogy legyen mihez viszonyítania, de elmenti az indexét is, hogy tudja hol kell keresni a tömbben. Ha végig ment a tömbön, akkor az IndexOfMaxY változó tartalmazza a jelenleg tömbben található legnagyobb amplitúdójú elem indexét.

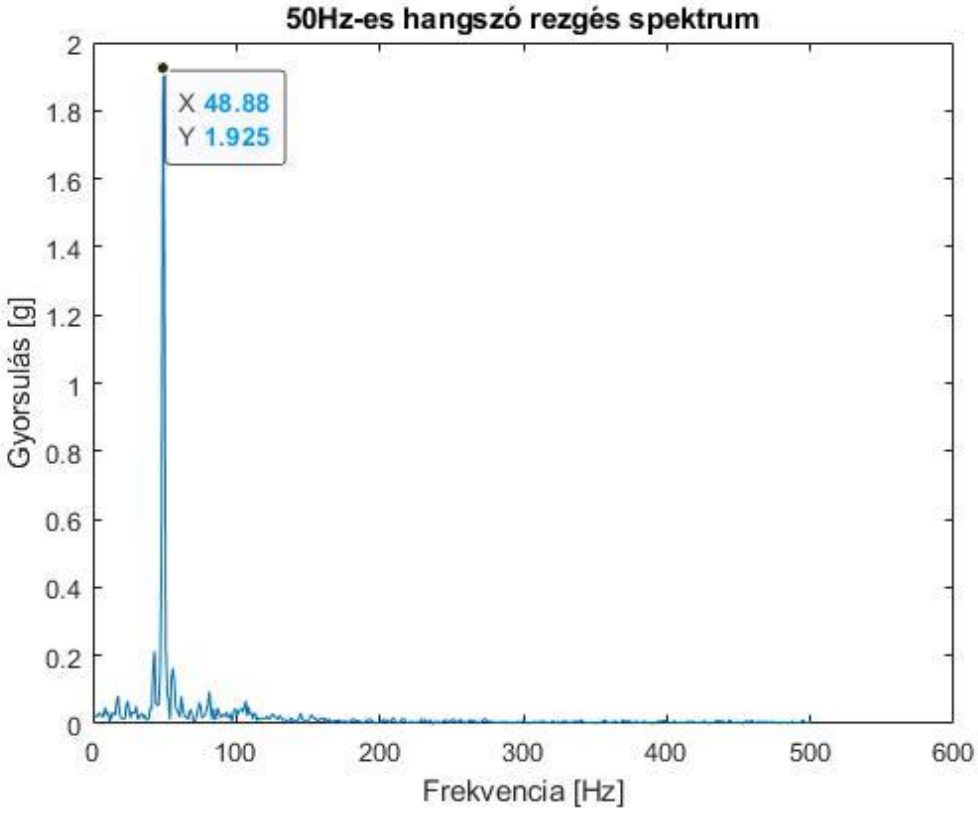
Mivel diszkrét mintavételi pontjaink állnak rendelkezésre, ezért valószínű, hogy a tényleges csúcs nem ezekben a pontokban van jelen valójában, hanem valahol közöttük. Éppen ezért, hogy ezt a hatást minél jobban csökkenteni lehessen lineáris interpolációt lehet használni, ami egyenest illeszt a szomszédos mintavételi pontokban található értékek összege és az aktuális érték kétszerese közé, majd ezt végül felezi. Ezáltal próbálja megbecsülni azt, hogy hol helyezkedhet el a tényleges legnagyobb csúcs a ±fél mintavételi pont távolságon belül. Ha pontosan a mintavételi frekvencia felénél található az aktuális legnagyobb komponens, akkor a pontosabb számítás érdekében sampleNum szerint skálázunk és nem sampleNum-1 szerint, mint az egyéb esetekben.

Legvégül, miután az interpoláció megtörtént betöltjük a legnagyobb csúcsokat csoportosító tömbünkbe az újonnan kiszámított frekvencia értéket, az eredeti tömbben pedig kinullázzuk az aktuálisan legnagyobb csúcs értékét, hogy a következő alkalommal már ne azt találjuk meg, mint legnagyobb csúcs. Ez a folyamat addig ismétlődik, amíg a topPeak tömb fel nem töltődik a legdominánsabb spektrumösszetevőkkel.

Miután a topPeaks függvény is lefutott az alkalmazás lényegi része lezárult, és a topPeak tömbben megtalálható a topPeakNum számú legdominánsabb frekvenciakomponen, hertzben kifejezve. Ezzel gyakorlatilag készen áll arra, hogy wifi-n keresztül felküldésre kerüljön az adatbázisba.

2.1.4 Az elkészült feldolgozó program tesztje

Nyilván szükség van tesztekre ahhoz, hogy igazoljuk a szoftver helyes működését. Ehhez valami konzisztens rengést kellene mérni, aminek ismert a rezgési frekvenciája. Ehhez megfelelőnek bizonyult egy hangszóró, amely rendelkezik mélynyomóval. Erre ráhelyezve a rezgésmérőt, majd a hangszórón egy tökéletes szinuszos hangot lejátszva ezt a frekvencia értéket kellene a rezgésmérővel is mérni. A *6-os számú ábrán* látható egy 50 Hz-es szinuszos hang lejátszásakor mérhető rezgés spektruma.



6. ábra

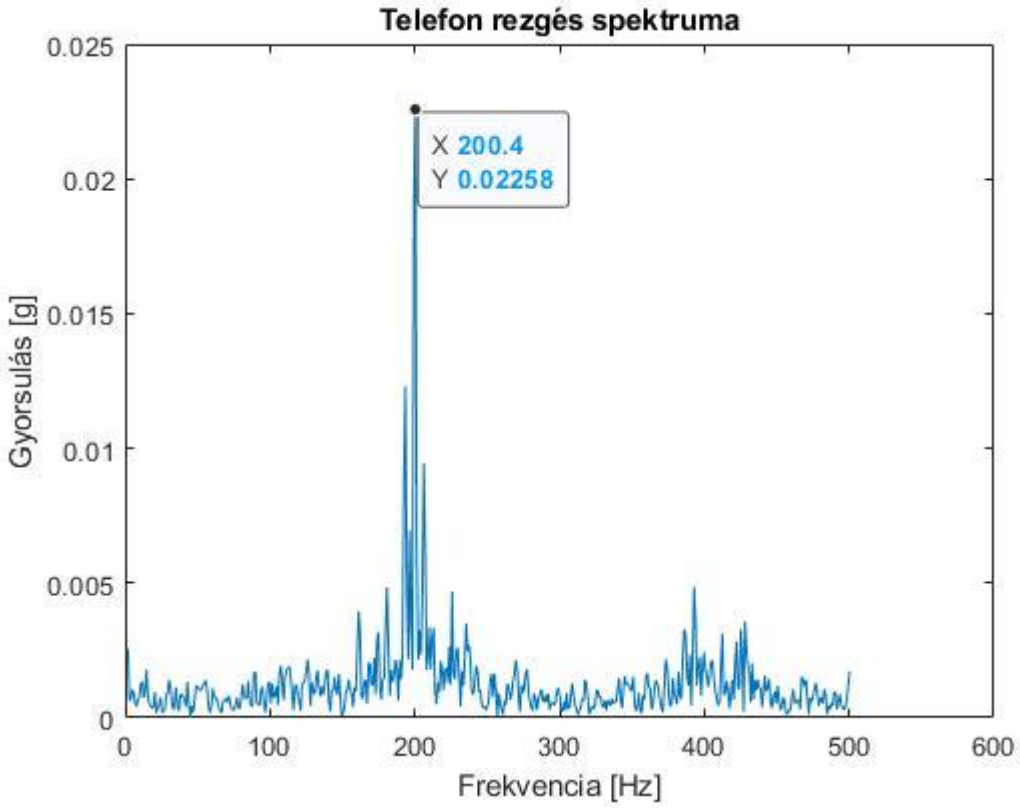
A tisztán szinuszos jelek frekvenciatartománybeli képe vonalas, ez ebben az esetben is jól látható. Meglehetősen közeli mérési eredmény az 50 Hz-es elvárt értékhez, főleg, hogy a hangszóró sem hitelesítetten 50 Hz-es rezgést biztosít. Látható, hogy ezt leszámítva minden egyéb frekvencia összetevő elhanyagolható, éppen ezért szükséges a top 10 érték kiszűrése, mert innen is látható, hogy a teljes spektrum elküldése nagyon pazarló lenne, pl. 400 Hz esetén valószínűleg semmi hasznos adat nincs, csak valami minimális zaj.

A *7-es számú táblázatban* különféle frekvenciájú hangokra található meg a 10 legdominánsabb spektrumösszetevő. Látható, hogy a legdominánsabb mindig nagyon jól közelíti az elvárt értéket, és az egyes összetevők is nagyon közel vannak egymáshoz, mivel tisztán szinuszos esetet vizsgálunk, éppen ezért nem is számítunk egyéb összetevőkre.

f (Hz)	1. (Hz)	2. (Hz)	3. (Hz)	4. (Hz)	5. (Hz)	6. (Hz)	7. (Hz)	8. (Hz)	9. (Hz)	10. (Hz)
40	39.3	38.77	39.26	39.1	39.11	39.38	38.8	39.27	38.82	39.26
50	48.88	48.55	49.17	49.58	48.9	48.94	49.38	48.49	49.24	49.15
70	69.14	68.13	68.91	68.38	68.55	69.26	67.82	69.09	67.76	68.96
100	98.7	98.0	98.1	98.72	97.09	98.69	97.14	98.67	97.88	98.35
150	148.02	145.24	147.69	145.59	147.65	146.86	147.89	145.38	147.71	147.52

7. ábra

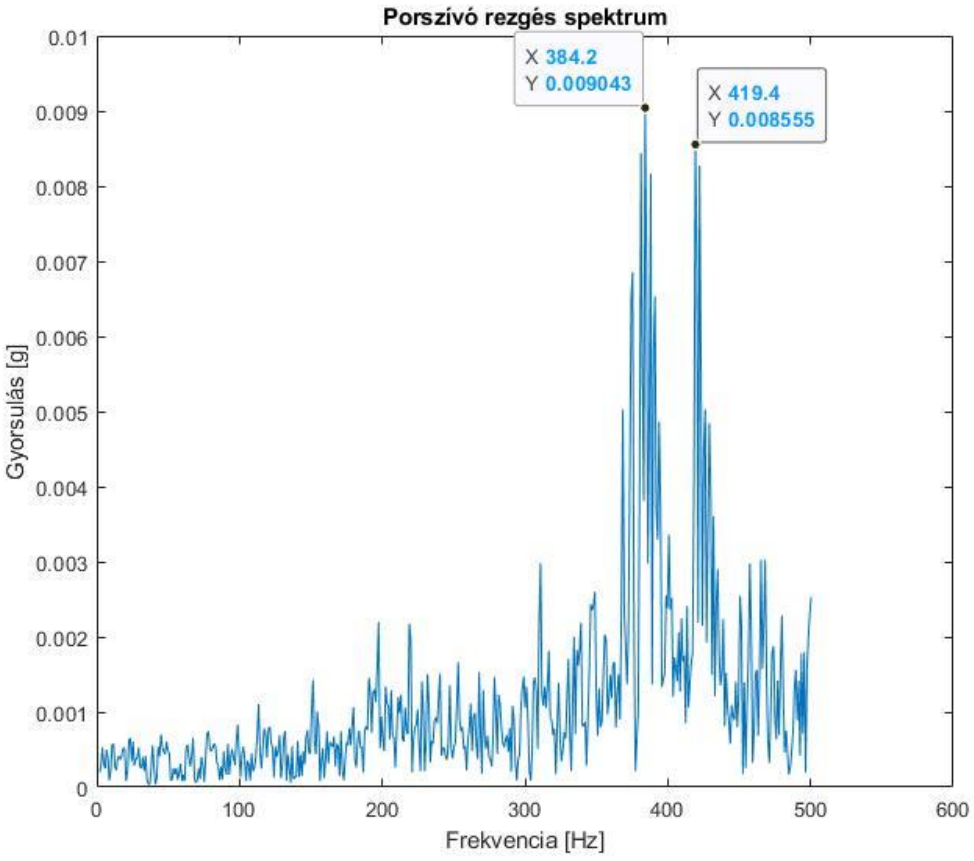
A tesztelés ezen kívül még többféle eszközzel is megtörtént. Telefon rezgésének a spektruma látható a *8-as számú ábrán*. Ez is egy meglehetősen adott frekvencián domináns értéket ad, de itt már megjelennek ugyan kisebb, de egyéb összetevők is. Mobiltelefonos spektrum analizátor alkalmazással vizsgálatra került hang alapján is a telefon rezgése, és az is 200 Hz körüli értéket jelzett csúcsnak.



8. ábra

Ezen kívül a projektben szereplő fordulatszám merővel is tesztelésre került. Egy DC motorral megforgatva egy tárcsát mérésre került a tárcsa fordulatszáma, valamint a rezgése. Ezen teszt alapján a két szenzor kölcsönösen alátámasztotta egymás megfelelő működését, ugyanis a motor kapocsfeszültségét változtatva minden esetben az RPM érték átszámítva hertz-be nagyon jó közelítéssel megegyezett.

Ezen kívül egy porszívó spektruma is felvételre került, a *9-es ábrán* látható, hogy itt már sokkal több frekvenciakomponens megjelenik, több egymástól távolabb eső jelentős csúcs is mérhető.



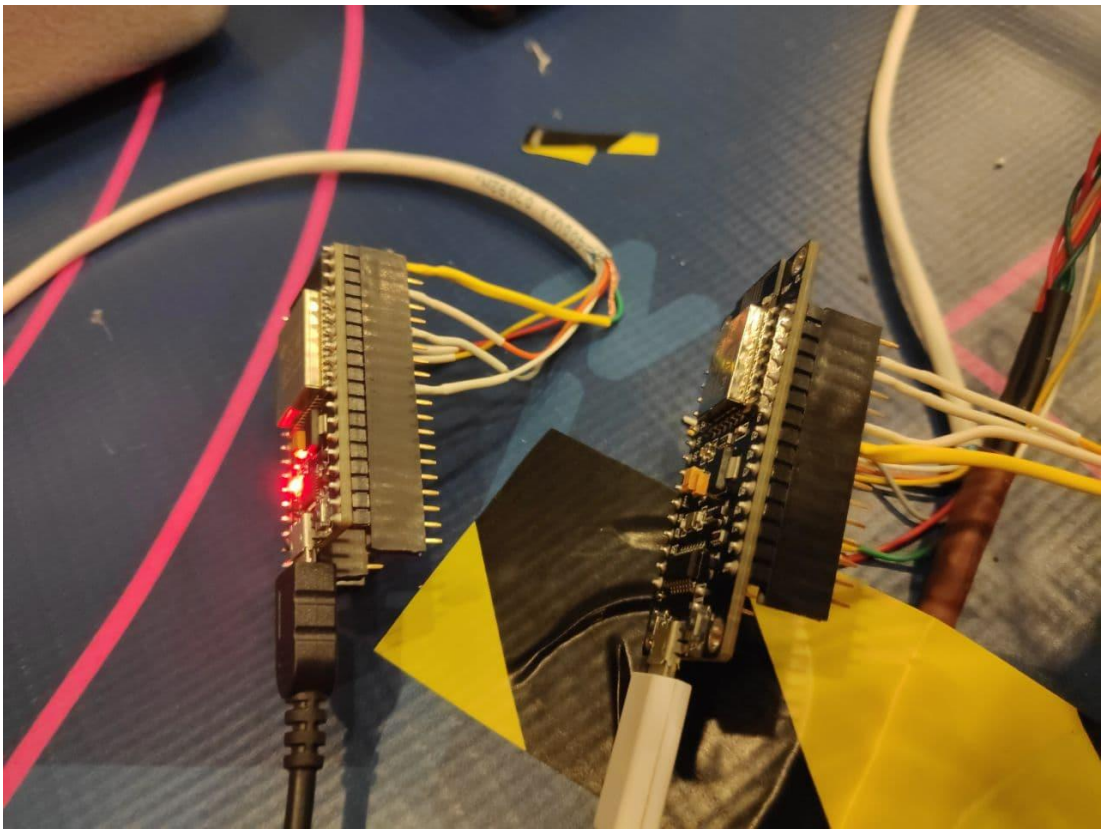
9. ábra

2.2 Egyéb szenzorok

A projektben szereplő egyéb szenzorok szoftverjének megírásával nem én foglalkoztam, viszont a rendszer összeállításával és a teszteléssel kapcsolatban ezekkel is foglalkoztam. Erről részletesebben a későbbiekben még lesz szó.

3. Rendszer összeállítása

Miután az összes szenzorhoz tartozó adatfeldolgozó szoftver megírásra és letesztelésre került, elkezdődhetett a végleges rendszer összeállítása. Ez tartalmaz összesen a már korábban említett 5 szenzort, valamint 2 mikrokontrollert. Mikrokontrollerek kiválasztásánál fontos szempont volt az, hogy rendelkezzenek beépített wifi modullal, így az adatok küldése könnyen megoldható, és nem szükséges külön periféria modul a wifi funkció megvalósításához. Ezért végül egy ESP32 és egy ESP8266 típusú mikrokontroller lett kiválasztva a feladatra.

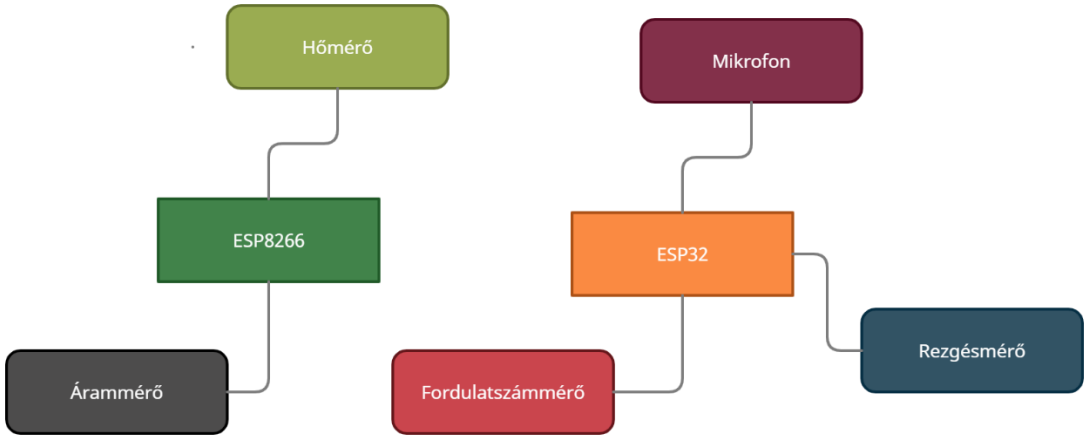


10. ábra

A *10-es ábrán* látható a végleges rendszerben a két mikrokontroller. Látható, hogy egyik esetében sem közvetlenül az MCU lábára lett ráforrasztva a mérővezeték, hanem egy külön pin sor került elhelyezésre, és erre lettek ráforrasztva a megfelelő vezetékek. Ezzel a rendszer moduláris marad, és a külön pin sor bármikor lehúzható az eszközről és akár azonnal is használható valamilyen más projekthez.

Maga a forrasztás sok időt vett idénybe, viszont végül sikeresnek mondható, mert az összeépített rendszer megfelelően funkcionált. A mikrokontroller oldal mellett természetesen a szenzor oldalakon is szükség volt forrasztásra, ez mind elvégzésre került. A megfelelő szigetelés érdekében (mint ahogy az a képen is látszik) zsugorcsövek kerültek elhelyezésre minden beforrasztott kontaktus köré.

A végleges rendszer blokkdiagramja a *11-es számú ábrán* látható. Ebben megfigyelhető minden korábban említett komponens. Azért lett ez az elrendezés kiválasztva, mivel a mikrofon, fordulatszámmérő és rezgésmérő meglehetősen gyakori kiszolgálást igénylő eszközök, ugyanis ezredmásodpercenként, vagy még gyakrabban igényelnek kiszolgálást. Erre a feladatra nyilván egy erősebb mikrokontrollerre van szükség, az ESP32 pedig megfelelőnek bizonyult a feladatra. Ezzel szemben az ESP8266 egy valamelyest gyengébb mikrokontroller, ez képes kezelni a lassabb kiszolgálást is toleráló hőmérő, valamint árammérő modulokat.



11. ábra

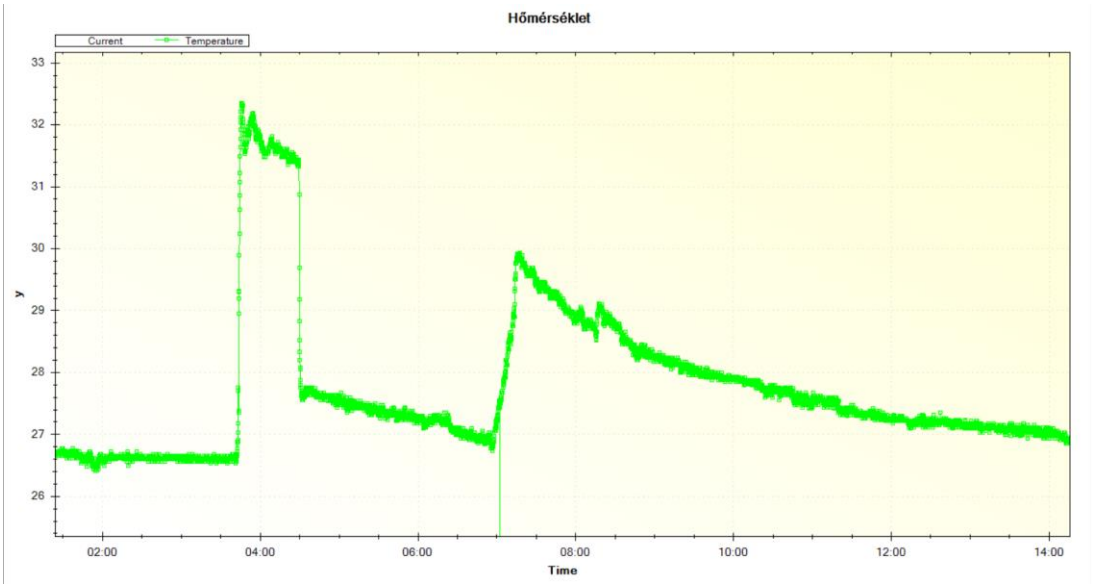
4. Tesztelés a végleges rendszeren

Miután összeállításra került a végleges rendszer elkezdődhettek vele az „eszterga szimulátoron" a végleges tesztek, ahol minden szenzor működésének vizsgálata egyszerre történik.

4.1 Hőmérő teszt

12. ábra

A hőmérő a *12-es ábrán* jelzett módon van az eszközre helyezve, a forgástengely aktuális hőmérsékletét méri. A *13-as ábra* mutatja a hozzá kapcsolódó időfüggvényt. Az látható, hogy kezdetben 26 °C körüli értékeket mér, majd található benne egy nagyon hirtelen ugrás, és 32 °C-ig is felmegy a mért érték. Ez akkor történt, amikor még nyugalmi állapotban az ujjunkat a szenzor elé helyeztük. Látható, hogy az ujjat elvéve ismét ugrásszerűen lecsökken a mért hőmérséklet, ami minimálisan nagyobb, mint korábban feltehetően azért, mert az érintéstől a csavar kissé felmelegedett. Ezek után a következő felfutás akkor volt tapasztalható, amikor bekapcsoltuk a forgatást. Látható, hogy itt is hirtelen, de az előző esethez képest viszonylag lassabban emelkedett a hőmérséklet, majd kikapcsolva szép lassan hűlve beállt a kiindulási hőmérsékletértékre.



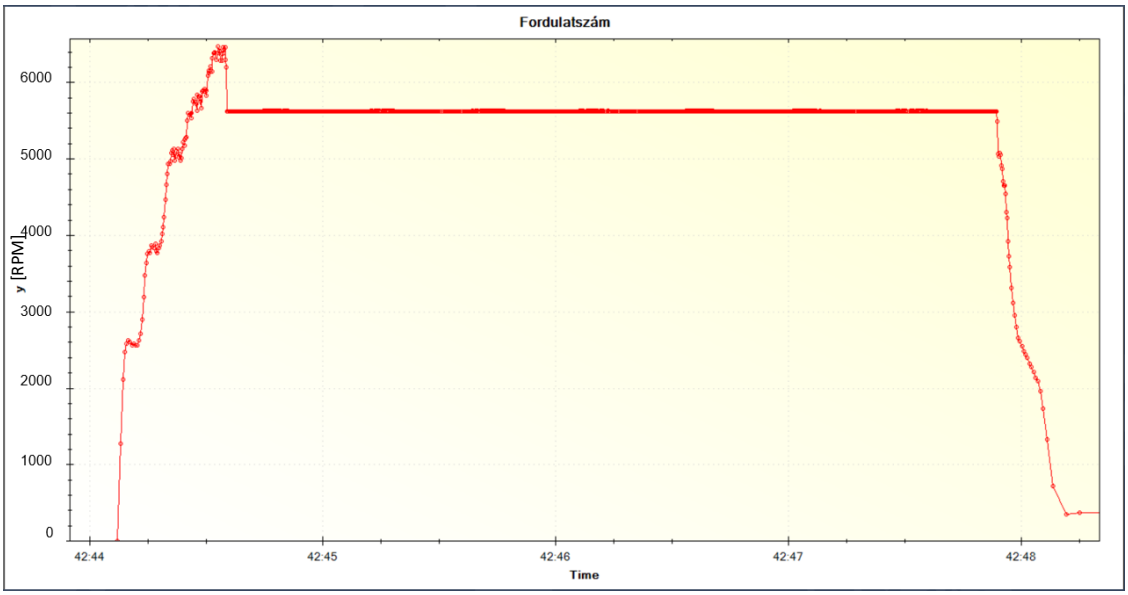
13. ábra

4.2 Fordulatszámmérő teszt

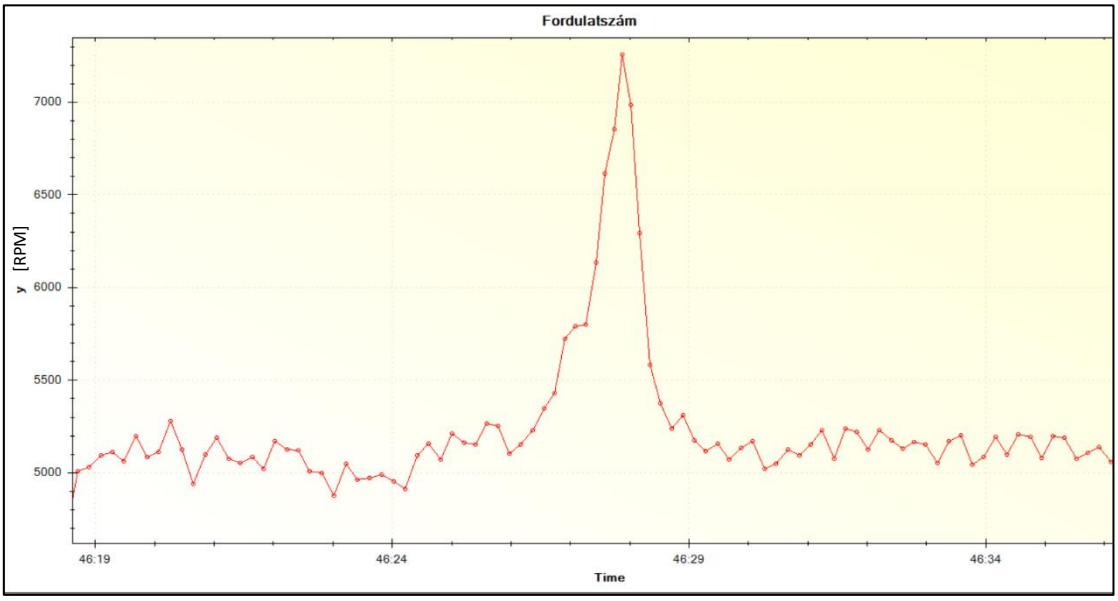
A fordulatszámmérő elhelyezkedése a *14. ábrán* látható. Felül található egy nyílás, és a fénysorompó ott nyúlik az eszközbe, mérve a tárcsa fordulatszámát. A *15. ábrán* látható egy bekapcsolási folyamat, majd a rendszer beáll egy konstans fordulatszámértékre, a végén pedig a kikapcsolásnál is megfigyelhető a hirtelen lelassulás.

14. ábra

Ezen kívül a 16. ábrán pedig az látható, hogy a rendszert már működtetjük egy állandó, körülbelül 5000 RPM-es fordulatszámmal, majd a forgató eszköz sebességét növelve ez felmegy 7000-es érték fölé, majd visszacsökkentve az eredeti fokozatra beáll a kezdeti, közel állandó fordulatszámértékre.



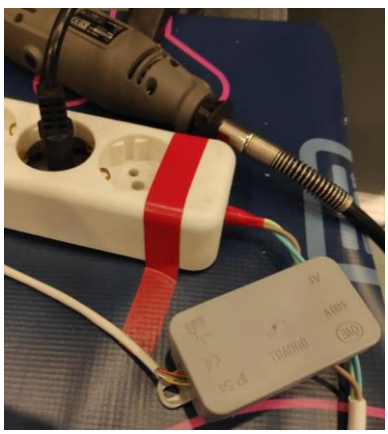
15. ábra



16. ábra

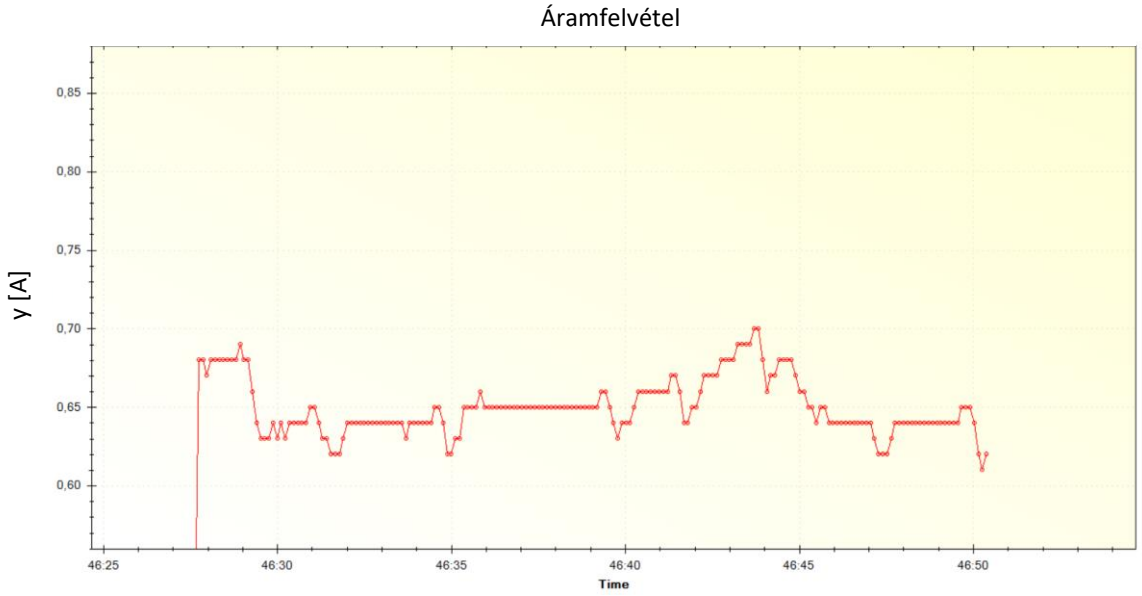
4.3 Árammérő teszt

Az árammérő képes akár 3 fázisú rendszer áramának megmérésére is. Jelen esetben a forgást keltő eszköz csupán 1 fázisú, ezért csak ez az egy került megmérésre. A *17. ábrán* látható a mérési elrendezés. Egy megbontott konnektor fázisvezetéke átvezetésre kerül a szenzoron, ami a szürke dobozon belül található. Maga a forgást előidéző eszköz a kép tetején látható, és az áramfelvétele a konnektoron keresztül történik, amelyen átfolyó áramot mérjük.



17. ábra

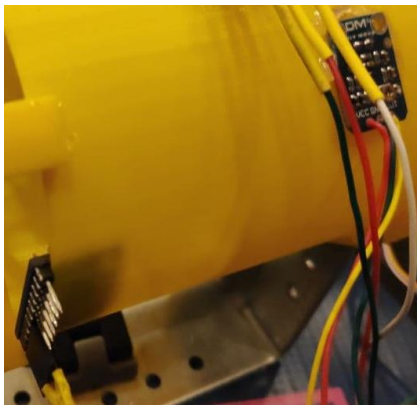
A teszt során egy idődiagram került felvételre, ami a *18-as ábrán* látható. Bekapcsolás után látható egy nagyobb áramfelvétel, feltehetően az indulás miatt. Ezután visszaesik, és egy közel állandó, 650 mA körüli értékre áll be. Majd látható egy újabb kiugrás, ekkor a forgató visszafogásra került, és ennek következtében nagyobb áramfelvételre van szüksége, ekkor 700 mA-es értéket is eléri.



18. ábra

4.4 Mikrofon és rezgésmérő teszt

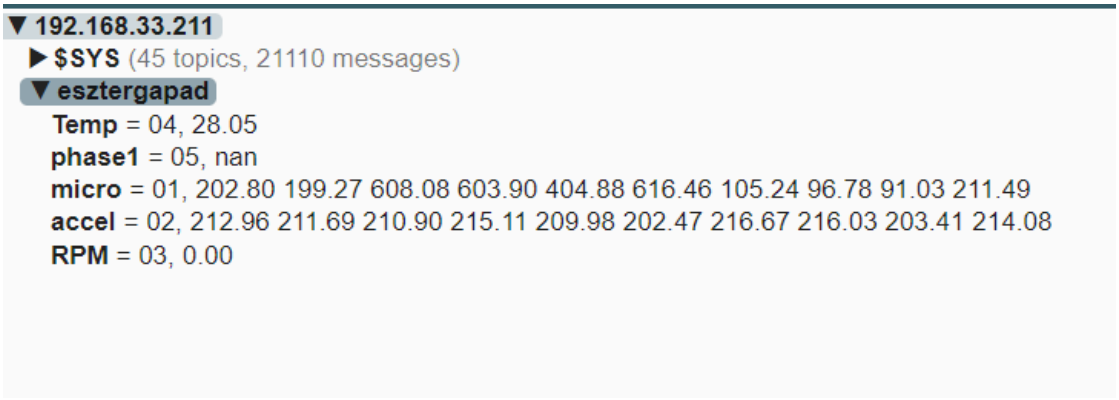
A rezgésmérő részletes tesztelése már korábban ismertetve lett. A mikrofon nagyon hasonló jellemzőt mér, és a tesztelése is hasonló. A mérési elrendezés a *19-es ábrán* látható, baloldalt a rezgésmérő, jobb oldalt pedig a mikrofon. A forgató eszközzel gerjesztve a rendszert azt tapasztaltuk, hogy a mikrofon teszi a dolgát megfelelően, 500 és 700 Hz körüli spektrum maximumokat adott eredményül. Sajnos mivel maximum 500 Hz-ig működőképes a gyorsulásmérő szenzor, ezért ezeket a rezgéseket nem képes detektálni.



19. ábra

Próbaként telefonnal megrezgetésre került az eszterga szimulátor váza, és itt az volt tapasztalható, hogy mindkét szenzor jól működik, a csúcsértéket közel azonosan ismerik fel. A *20. árbrán* már a szerverre felküldött adatok láthatók ezek az idő múltával folyamatosan frissülnek, és kiírásra kerül minden egyéb szenzor adatai mellett a rezgésmérés és a mikrofon által detektált top 10

frekvenciakomponens. Az is látható, hogy a mikrofon a kevésbé domináns spektrumkomponensek között talál nagyobb frekvenciás (pl. 600 Hz-es összetevőket) is. Nyilván sokkal jobb lett volna, ha a rezgésmérő is képes erre, viszont a rendelkezésre álló szenzor sajnos nem volt többre képes.



20. ábra

5. Továbbfejlesztési lehetőségek

Elsőként mindenképpen jó továbbfejlesztési lehetőség lenne egy másik gyorsulásmérő szenzor beszerzése, amely nagyobb mintavételi frekvencia elérésére is képes. Ezzel ugyanis nagyobb frekvenciatartományt lennénk képesek vizsgálni, és valósabb kép adódhatna a megfigyelt rendszerről. Keresgélések alapján az adatlapi értékek szerint az Adafruit ADXL345 szenzor alkalmas lehetne a feladatra, de nyilván beszerzés előtt érdemesebb ennek alaposabban utána járni.

Ezen kívül célravezető lehet más típusú kommunikációs protokoll tesztelése, a korábban már említett indokok miatt. Teszteléssel pontosan megállapítható lehet, hogy esetleg az SPI kommunikáció alkalmasabb-e a feladatra vagy sem.

További szenzorok projektbe vonásával lehetőség lenne alaposabban megfigyelni a különféle ipari folyamatokat. Például lehetne kamerát használni, és a kamera képéből információt kinyerni valamilyen képfeldolgozási eljárás segítségével.

Végül az is hasznos lenne, ha nagyobb teljesítményű mikrokontrollert használnánk. Ekkor ugyanis gyakrabban szolgáltathatna a rendszer adatokat (pl. az időigényes FFT algoritmus sokkal gyorsabban futhatna). Másrészről, ha nagyobb tárhellyel is rendelkezne, akkor még több adatot tudna tárolni, ezzel pontosabbá téve a mérést.

6. Összegzés

Úgy gondolom, hogy a projekt sikeresnek mondható. A félév elején kitűzött célokat sikerült elérni, a rendszer teljesen összeállt és működőképes. Sokat

tanultam a félév során a beágyazott rendszertervezésről, mikrokontrollerek programozásáról, kommunikációs megoldásokról, forrasztásról, valamint a csapatban történő munkavégzésről. A jövőben szívesen folytatnám az itt megkezdett munkát ezen a projekten.

7. Források

- https://github.com/kosme/arduinoFFT
- https://github.com/asukiaaa/MPU9250_asukiaaa
- https://3cfeqx1hf82y3xcoull08ihx-wpengine.netdna-ssl.com/wp-content/uploads/2015/02/PS-MPU-9250A-01-v1.1.pdf
- https://invensense.tdk.com/wp-content/uploads/2015/02/RM-MPU-9250A-00-v1.6.pdf