

---

**UNIVERSITATEA „SAPIENTIA” DIN CLUJ-NAPOCA**  
**FACULTATEA DE ȘTIINȚE TEHNICE ȘI UMANISTE,**  
**TÎRGU-MUREȘ**  
**SPECIALIZAREA CALCULATOARE**

**Indicator inteligent pentru roller**  
**PROIECT DE DIPLOMĂ**

**Coordonator științific:**

**Ș.l.dr.ing. Turos László-Zsolt**

**Absolvent:**

**Ambrus Barna**

**2023**

UNIVERSITATEA „SAPIENTIA” din CLUJ-NAPOCA  
Facultatea de Științe Tehnice și Umaniste din Târgu Mureș  
Specializarea: **Calculatoare**

Viza facultății:



**LUCRARE DE DIPLOMĂ**

Coordonator științific:  
**ș.l. dr. ing. Turos László-Zsolt**

Candidat: **Ambrus Barna**  
Anul absolvirii: **2023**

**a) Tema lucrării de licență:**

INDICATOR INTELIGENT PENTRU ROLLER

**b) Problemele principale tratate:**

- Studiu bibliografic privind sisteme similare realizate
- Proiectarea sistemului
- Realizarea aplicației mobile web pentru monitorizare și configurarea modului de funcționare
- Realizarea aplicației pe sistemul încorporat ESP32 responsabil pentru colectarea și filtrarea datelor de la senzor, respectiv pentru comunicația pe WiFi
- Realizarea aplicației webserver și de control pe sistemul încorporat ESP8266

**c) Desene obligatorii:**

- Schema bloc al aplicației
- Diagrame de timp privind datele măsurate și filtrate
- Imagini cu software-ul realizat
- Imagini cu prototipul realizat

**d) Softuri obligatorii:**

- Aplicația pe sistemul încorporat 1 pentru colectarea și filtrarea datelor de la senzori
- Aplicația pe sistemul încorporat 2 pentru controlul LED-urilor

**e) Bibliografia recomandată:**

- Túros László-Zsolt, Székely Gyula: Érzékelők és mérőhálózatok, Scientia kiadó, 2022
- C. Howard, S. Mark és F. Nathan, Gyroscope Vector Magnitude: A proposed measure for accurately measuring angular velocities, 2022.

**f) Termene obligatorii de consultații: săptămânal**

**g) Locul și durata practicii:** Universitatea „Sapientia” din Cluj-Napoca,

Facultatea de Științe Tehnice și Umaniste din Târgu Mureș

Primit tema la data de: 31.03.2022

Termen de

predare:

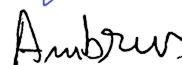


27.06.2023

Semnătura Director Departament

Semnătura coordonatorului

Semnătura responsabilului  
programului de studiu

Semnătura candidatului



---

## Declarație

Subsemnata/ul Ambrus Barna absolvent(ă) al/a specializării Calculatoare, promoția 2023 cunoscând prevederile Legii Educației Naționale 1/2011 și a Codului de etică și deontologie profesională a Universității Sapiientia cu privire la furt intelectual declar pe propria răspundere că prezenta lucrare de licență/proiect de diplomă/disertație se bazează pe activitatea personală, cercetarea/proiectarea este efectuată de mine, informațiile și datele preluate din literatura de specialitate sunt citate în mod corespunzător.

Localitatea, Tg-Mures

Data: 27.06.2023

Absolvent Ambrus Barna

Semnătura 

---

# Indicator inteligent pentru roller

## Extras

Lucrarea descrie în detaliu un sistem și arhitectura acestuia pentru gestionarea semnalelor de schimbare a direcției și de frânare pe scutere și biciclete, folosind un senzor cu accelerometru și giroscop pe spatele mâinii. Proiectul urmărește să ofere o soluție pentru semnalele de direcție pe volan fără a îndepărta mâna de pe volan, evitând astfel accidente care rezultă. Utilizatorul este capabil să indice o schimbare de direcție prin gesturi precum întoarcerea mâinii într-o parte, astfel încât nu este necesar să ia mâna complet de pe volan, menținând astfel echilibrul vehiculului. Semnalele de frânare sunt automate ca urmare a detectării accelerației. Sistemul utilizează două microcontrolere (ESP32 și ESP8266), dintre care unul procesează datele senzorilor, în timp ce celălalt rulează un server web și aprinde LED-urile corespunzătoare. Serverul web este responsabil pentru comunicarea dintre microcontrolere și, de asemenea, rulează o pagină web care acționează ca interfață pentru utilizator. Pe pagina web sunt afișate semnalele active, cum ar fi semnalele de viraj și semnalele de frânare, care pot fi activate și dezactivate. Comunicarea între diferitele componente se realizează fără fir, utilizând tehnologia WiFi.

**Cuvinte cheie:** microcontroler, WiFi, giroscop, accelerometru

**SAPIENTIA ERDÉLYI MAGYAR  
TUDOMÁNYEGYETEM  
MAROSVÁSÁRHELYI KAR  
SZÁMÍTÁSTECHNIKA SZAK**

**Intelligens hajtányjelző  
DIPLOMADOLGOZAT**

**Témavezető:**

**Dr. Turos László-Zsolt,  
egyetemi adjunktus**

**Végzős hallgató:**

**Ambrus Barna**

**2023**

---

# Kivonat

A dolgozat egy olyan rendszert és annak felépítését mutatja be részletesen, amely a különböző hajtányokon történő irányváltásra és fékezésre vonatkozó jelzéseket kezeli, egy gyorsulásmérőt és giroszkópot tartalmazó szenzor segítségével, ami a kézfejen kap helyet. A projekt megoldást próbál nyújtani a hajtányokon történő irányjelzésekre a kéz kormányról való levétele nélkül, ezzel elkerülve az ebből fakadó baleseteket. Az irányváltásra vonatkozó jelzéseket a felhasználó képes jelezni különböző gesztusok segítségével, mint a kéz oldal irányú forgatása, így nem szükséges levennie a kezét a kormányról teljesen, ezzel megtartva a jármű egyensúlyát. A fékjelzések automatikusan történik a gyorsulás érzékelése következtében. A rendszer két mikrovezérlőt használ (ESP32 és ESP8266) amik közül az egyik a szenzor adatokat dolgozza fel, míg a másik futtat egy webszervert és megvilágítja a megfelelő led-eket. A webszerver a kommunikációért felel a mikrovezérlők között, valamint egy felhasználói felületként funkcionáló weboldalt is üzemeltet. A weboldalon láthatóak az aktív jelzések, mint a kanyarjelzések és a fékjelzések, valamint ki- és bekapcsolhatjuk ezeknek a működését. A különböző komponensek közötti kommunikáció vezeték nélkül valósul meg WiFi technológiát használva.

**Kulcsszavak:** mikrovezérlő, WiFi, giroszkóp, gyorsulásmérő

---

# Abstract

This paper describes in detail a system and its architecture for managing direction change and braking signals on scooters and bicycles, using a sensor with an accelerometer and gyroscope on the back of the hand. The project aims to provide a solution for direction signals on the crank without removing the hand from the handlebars, thus avoiding the resulting accidents. The user is able to indicate a change of direction by gestures such as turning the hand sideways, so that it is not necessary to take the hand completely off the steering wheel, thus maintaining the balance of the vehicle. Brake signals are automatic as a result of acceleration sensing. The system uses two microcontrollers (ESP32 and ESP8266), one of which processes the sensor data, while the other runs a web server and lights the corresponding LEDs. The web server is responsible for the communication between the microcontrollers and also runs a web page that acts as a user interface. On the web page, active signals such as turn signals and brake signals are displayed and can be switched on and off. Communication between the different components is wireless using WiFi technology.

**Keywords:** microcontroller, WiFi, gyroscope, accelerometer

## Tartalomjegyzék

1. Bevezető .....	10
1.1. Háttérinformáció.....	10
1.2. A probléma felvetése .....	10
1.3. Célkitűzés és motiváció .....	11
2. Elméleti megalapozás és szakirodalmi tanulmány .....	11
2.1. Szakirodalmi tanulmány .....	11
2.2. Általános működés .....	13
2.3. Elméleti alapok .....	14
2.3.1. IoT .....	14
2.3.2. DIY .....	15
2.3.3. Mikrovezérlők.....	15
2.3.4. Gyorsulásmérő és giroszkóp .....	16
2.3.5. Webszerver .....	19
2.4. Felhasznált eszközök .....	19
2.4.1. ESP32 és ESP8266 .....	19
2.4.2. MPU6050 .....	21
2.4.3. LED.....	22
2.5. Felhasznált technológiák .....	23
2.5.1. I2C.....	23
2.5.2. HTTP.....	23
2.5.3. Websocket.....	24
2.5.4. LittleFS .....	24
3. A rendszer specifikációi és architektúrája.....	24
3.1. Rendszer specifikációi.....	25
3.1.1. Funkcionális követelmények .....	25
3.1.2. Nem-funkcionális követelmények .....	26
3.2. Rendszer architektúra .....	27
3.2.1. Rendszerkomponensek .....	29
3.2.2. Komponensek közötti kapcsolat és kommunikáció .....	29
4. Részletes tervezés és gyakorlati megvalósítás .....	29
4.1. A szenzor adatokat feldolgozó mikrovezérlő szoftvere .....	30
4.1.1. Inicializálás .....	30
4.1.2. Szenzor adatok olvasása és feldolgozása .....	30
4.1.3. Irányjelzés .....	31
4.1.4. Fékjelzés .....	35
4.1.5. Összegzés .....	38
4.2. A webszervert futtató mikrovezérlő szoftvere .....	39
4.2.1. Inicializálás .....	39
4.2.2. Webszerver .....	39
4.3. Webalkalmazás.....	42
4.4. Prototípus bemutatása.....	44
5. Továbbfejlesztési lehetőségek.....	46
5.1. Hardver fejlesztése .....	46
5.2. Szoftver fejlesztése.....	47
6. Hivatkozások.....	48



## Ábrajegyzék

1. ábra Lightvest.....	12
2. ábra Fésűs gyorsulásmérő [5].....	17
3. ábra Szabad forgó giroszkóp.....	17
4. ábra Roll, Pitch, Yaw .....	18
5. ábra ESP32 .....	20
6. ábra ESP8266 .....	20
7. ábra MPU6050 tengelyek.....	21
8. ábra MPU6050 .....	22
9. ábra Mpu6000 és MPU6050 összehasonlítása [11] .....	22
10. ábra A rendszer egyszerű vázlata I.....	28
11. ábra A rendszer egyszerű vázlata II. ....	28
12. ábra Bal irány jelzése .....	32
13. ábra Jobb irány jelzése .....	33
14. ábra Hamis jelzés .....	34
15. ábra X és Z tengelyek felcserélése .....	35
16. ábra Mért gravitációs gyorsulás .....	36
17. ábra Számított lineáris gyorsulás szemléltetése .....	37
18. ábra Fékezés jelzése .....	38
19. ábra WiFi Station (STA) Mode [17] .....	40
20. ábra WiFi Access Point (AP) Mode [17] .....	41
21. ábra Kanyarjelzés .....	43
22. ábra Fékjelzés.....	43
23. ábra Jelzés bekapcsolva.....	44
24. ábra Jelzés kikapcsolv .....	44
25. ábra Prototípus ábra.....	45
26. ábra Prototípus a valóságban.....	46

# 1. Bevezető

A projekt bemutatja egy olyan rendszer kiépítését és működését, amely lehetővé teszi az irányváltoztatás és fékezés jelzését egy hajtányról, a kéz kormányról való levétele nélkül egy szenzorokkal ellátott úgy nevezett okos kesztyű segítségével. Az irányjelzés és a fékezés időben és hatékonyan történő jelzése rendkívül fontos a közúti közlekedésbiztonság szempontjából. A projekt célja, hogy olyan innovatív megoldást nyújtson, amely minimalizálja a vezetők egyensúlyvesztéséből eredő balesetek lehetőségét, és ez által növeli a közlekedésbiztonságot a hajtányokon.

## 1.1. Háttérinformáció

A hajtányok, mint például biciklik, rollerek és más hasonló közlekedési eszközök, napjainkban egyre népszerűbbé válnak az emberek körében. Ennek számos oka van. Először is, a hajtányok kiváló alternatívát nyújtanak a rövid távolságok megtételére akár városi, akár falusi környezetben is. Az emberek egyre inkább igénylik a környezetbarát és fenntartható közlekedési lehetőségeket, és a hajtányok kiválóan illeszkednek ebbe a kategóriába. Nemcsak környezetbarát, hanem gazdaságos is valamilyen hajtány használata az autókkal szemben, főleg ha az üzemanyag és az energiafogyasztást tekintjük, ugyanis ezek a járművek határozottan kevesebbet fogyasztanak, vagy pedig nem is fogyasztanak, mint a hagyományos kerékpárok és rollerek, amiket az utas hajt meg.

Emellett fontos megemlíteni, hogy a hajtányok rugalmasabbá és gyorsabbá teszik a közlekedést zsúfolt városi területeken. A dugók és a parkolási nehézségek elkerülésével a hajtányok lehetővé teszik az emberek számára, hogy könnyedén átjussanak a forgalmon. Ez jelentős időmegtakarítást és kényelmet nyújt a felhasználóknak.

Az említett tényezők miatt a hajtányok egyre fontosabb szerepet töltenek be a közlekedésben. Ahhoz azonban, hogy ezek az eszközök valóban biztonságosak legyenek, fontos, hogy a vezetők hatékonyan és időben jelezzék az irányváltoztatást és a fékezést.

## 1.2. A probléma felvetése

A hajtányok vezetői az irányváltoztatásukról általában nem szokták figyelmeztetni a forgalomban lévő többi embert, ugyanis ezek a járművek nem rendelkeznek erre a célra kifejlesztett eszközzel, vagy csak nagyon ritkán. Ez értelemszerűen rengeteg veszélyt hordoz magában és kifejezetten balesetveszélyes tud lenni. Azok az emberek, akik viszont igyekeznek betartani a forgalmi szabályokat a hajtányuk vezetése közben is, azok általában a megfelelő

irányba kitért karral igyekeznek jelezni, hogy éppen irányváltoztatásra készülnek. Ez határozottan felelősegteljesebb magatartás, mint a korábban bemutatott, viszont még ez is tartogat veszélyeket, ugyanis így a vezető kénytelen levenni a kezét a kormányról és ebből kifolyólag elveszítheti az egyensúlyát. Igaz, hogy a tapasztaltabb hajtányvezetőknek semmiképpen nem okoz gondot ez a művelet, sőt az emberek többségének sem okoz gondot néhány másodperc erejéig fél kézzel irányítani a járművet. A baj viszont az, hogy nem minden ember mondható tapasztalt hajtányvezetőnek és nem minden ember rendelkezik ennek megfelelő egyensúlyérzéssel. Amikor a vezető elveszíti az egyensúlyát, nem tudja megfelelően irányítani a hajtányt. Ez azzal a kockázattal jár, hogy összeütközik más járművekkel, gyalogosokkal vagy tárgyakkal az úton. Az ilyen típusú ütközések súlyos sérüléseket okozhatnak mind a vezetőnek, mind pedig a többi közlekedőnek.

### **1.3. Célkitűzés és motiváció**

Akárhogy is vesszük a korábban említetteket, a jelenlegi kézkinyújtáson alapuló jelzésrendszer veszélyeket hordoz magában, amelyek számos közúti baleset forrásaként is szolgálhatnak, így akár emberek élete is veszélybe kerülhet. Ezért elengedhetetlen egy olyan megoldás kifejlesztése, amely lehetővé teszi a vezetők számára, hogy egyszerűen jelezzék az irányváltoztatást és a fékezést, anélkül hogy veszélyeztetnék saját biztonságukat vagy másokét. A projekt célja, hogy olyan rendszert hozzon létre, amely lehetővé teszi a hajtányok vezetői számára a biztonságosabb és kényelmesebb közlekedést.

A cél az, hogy a hajtányt vezetők a kezeiket ne kelljen, hogy levegyék a kormányról, így elkerülve az egyensúlyvesztést, valamint hogy ez által egy kényelmesebb alternatívát nyújtsunk az irányjelzésre. Ezen kívül fontos megjegyezni a jobb észlelhetőséget, ugyanis a beépített LED égőknek köszönhetően a rendszer határozottan a többi forgalomban lévő ember tudtára adja az épp aktuális jelzést még sötétben is, így téve biztonságosabbá a közlekedést.

## **2. Elméleti megalapozás és szakirodalmi tanulmány**

### **2.1. Szakirodalmi tanulmány**

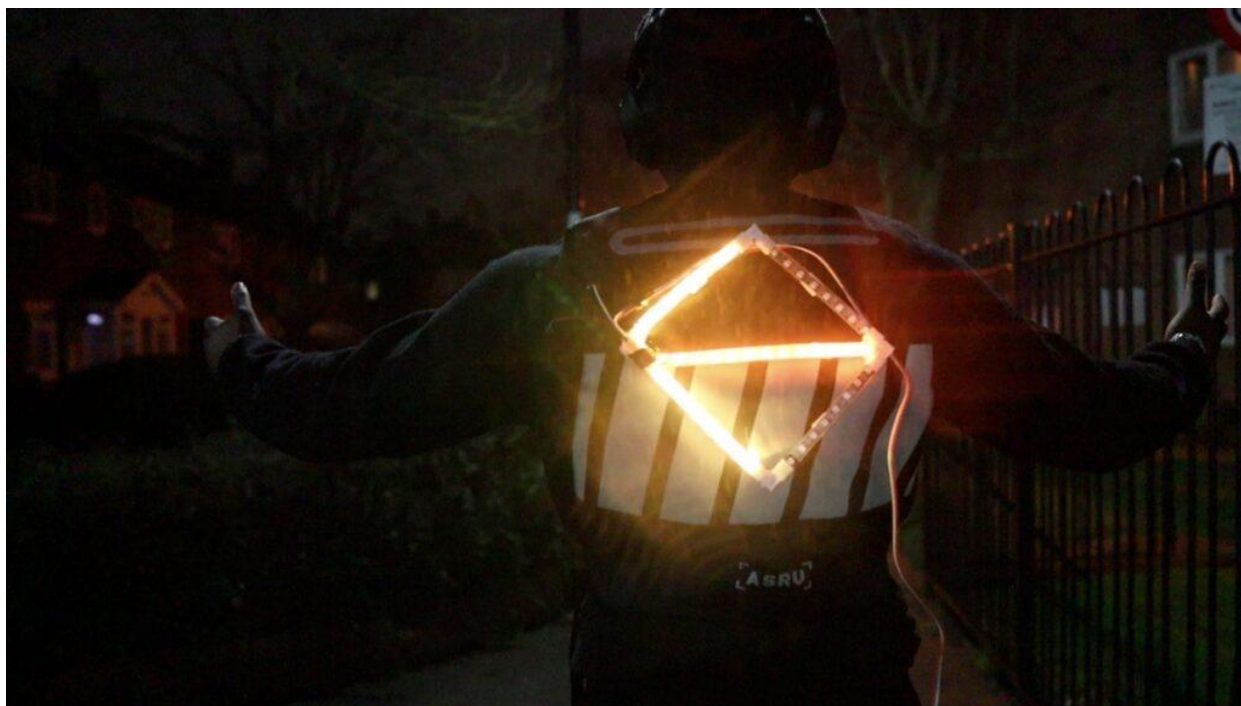
A szakirodalmi tanulmányt azzal kezdtem, hogy utánanéztem milyen kisebb méretű mikrovezérlők léteznek, amelyek el tudnak férni a kézen is. Így találtam rá kezdetben az Arduino Nano lapkákra és a tervezés kezdeti fázisaiban még azt szerettem volna használni. A döntés végül

mégis az ESP alapú mikrovezérlőkre esett, ugyanis ez olcsóbb is és rendelkezik beépített WiFi és Bluetooth modulokkal, így a vezeték nélküli kommunikációhoz nincs szükség más kiegészítőkre.

Ez után következett a szenzor kérdése, amit rövid kutakodás során meg is találtam. Ez egy MPU6050, ami tartalmaz egy gyorsulásmérőt és egy giroszkópot. Ennek a segítségével lehet figyelni a kéz mozgását az irányjelzésekhez, valamint a gyorsulást a fékezés jelzéséhez.

Mivel fontos a vezeték nélküli kommunikáció, ezért a kutatás során megpróbáltam minél több lehetőséget is figyelembe venni. Ilyen volt a rádiófrekvenciás (RF) kommunikáció, amihez szükség lett volna RF adó és vevő modulokra, valamint a bluetooth is. Kezdetekben a bluetooth-ot részesítettem előnyben, ugyanis ennek a technológiának a segítségével lehet kommunikálni akár mobiltelefonokkal is. Ennek előnye az lett volna, hogy így egy okostelefonon meg lehet jeleníteni a rendszer adatait egy felületen és akár irányítani is lehet azt. A döntés végül mégis a WiFi-re esett. Ennek oka, hogy az adatokat egy webszerver segítségével jelenítem meg egy weboldalon, így kézenfekvő, hogy a mikrovezérlők is WiFi-n keresztül kommunikáljanak egymással.

Ezt követően megpróbáltam keresni az interneten, hátha találok ilyen vagy ehhez hasonló projekteket inspiráció céljából. Viszonylag gyorsan találtam is egy startup projektet, ami nagyban hasonlít az enyémre, néhány különbséggel. A projekt címe „LightVest – Build an Arduino Turn Signal Bike Safety Vest”, amit egy *Eben Kouao* nevű ember csinált, aki a DIY megközelítéssel különböző eszközöket hoz létre és ezekről videókat is készít. [1]



1. ábra Lightvest

Az egyik nagyon szembetűnő különbség az én projektem és a korábban említett startup projekt között az, hogy ebben a projektben a kormány elfordításának a hatására történnek meg a jelzések, míg nálam egy kézfejen lévő szenzor dőlésének a hatására. A kormány elfordítás hatására létrejött jelzéssel azonban van egy hatalmas probléma. Ez nem más, mint az, hogy ha csak akkor jelez a rendszer irányváltást, ha már a kormány elfordult, vagyis az irányváltás folyamatban van, vagy már meg is történt, akkor a forgalomban levő többi ember túl későn szerez csak tudomást erről és így ez balesethez is vezethet. A jelzésnek az irányváltást előre kellene jeleznie a többi forgalomban levő ember számára, hogy ne érje váratlanul őket, nem pedig az irányváltással egyidőben kell megtörténnie.

Mindenesetre a kutatás hatására sikerült kiválasztanom, hogy milyen komponensekkel és technológiákkal szeretnék dolgozni, valamint betekintést nyertem egy hasonló rendszernek a felépítésébe is. Ezt a projektet tanulmányozva kaptam képet arról, hogyan is kellene kinéznie egy kész projektnek, valamint a hibáiból tanultakat igyekeztem beleépíteni a saját projektembe.

## **2.2. Általános működés**

A projektben a szenzor és a mikrovezérlő együttműködése jelenti a rendszer alapját. A gyorsulásmérőt és giroszkópot is tartalmazó szenzor a kézfejen helyezkedik el egy mikrovezérlővel. A mikrovezérlő figyeli és feldolgozza a szenzor adatait, majd ennek alapján meghatározza, milyen jelzést kell küldenie a másik mikrovezérlőnek, ami a megjelenítésért felelős.

Az alapállapotból kiinduló bal vagy jobb dőlés esetén a rendszer a megfelelő irányba történő irányváltást jelzi. Az alapállapot a kéz pozícióját jelenti a kormányon, amikor közel vízszintes helyzetben van a szenzor. A mikrovezérlő a kéz dőlését figyeli, és csak akkor tekinti az adott mozgást jelzésnek, ha az abszolút dőlésszög és a relatív dőlésszög (az alapállapothoz viszonyított szög) is meghaladja a meghatározott küszöbértéket. Ez a feltételrendszer kizárja a hamis jelzéseket, lehetővé téve a felhasználó számára, hogy a kezét más tevékenységekre is használja közben. A jelzés csak akkor történik meg, ha ezek a feltételek teljesülnek a jobb vagy a bal irányban.

A fékezés jelzése az X tengely gyorsulását vizsgálja és egy erre megszabott feltételrendszernek kell teljesülnie a fékezés jelzéséhez. Nagyon leegyszerűsítve, ha a gyorsulás meghalad egy bizonyos előre beállított negatív értéket, akkor a rendszer jelzést ad erre. Természetesen további feltételeket is tartalmaz ez a funkció, amikről bővebben lesz szó a továbbiakban. Ez a funkció biztosítja, hogy a forgalomban lévők is értesüljenek arról, ha a jármű fékezik.

A mikrovezérlő ezeket a jelzéseket továbbítja a webszervert futtató másik mikrovezérlőnek, ami megjeleníti az adatokat egy weboldalon. Emellett a jelzéseknek megfelelő LED-ek is felgyulladnak, így a felhasználó és a többi utas is könnyen észlelheti az éppen aktuális jelzéseket.

## **2.3. Elméleti alapok**

### **2.3.1. IoT**

Az IoT (Internet of Things) vagyis a dolgok internete, olyan elektronikai eszközöket jelent, amelyek képesek információkat felismerni valamilyen szenzor segítségével és ezt egy internet alapú hálózaton keresztül megosztani más eszközökkel. Általában ezeket a rendszereket ellátják az „okos” vagy „intelligens” jelzőkkel, mint például az okosóra, vagy okosotthon.

Az IoT egy manapság népszerű és rohamosan fejlődő technológia, aminek rengeteg felhasználási területe van, legyen szó akár ipari, akár otthoni felhasználásról. Ezek a rendszerek arra hivatottak, hogy a technológia segítségével hatékonyabbá és biztonságosabbá tegyék a mindennapi életünket.

Egyik legnépszerűbb otthoni felhasználása az IoT rendszereknek az úgynevezett okosotthon, aminek a lényege, hogy a különböző otthoni tárgyakba és eszközökbe beágyazott érzékelők és mikrovezérlők egy hálózaton keresztül kapcsolatban állnak egymással és a működtető személlyel. Így a rendszer működhet automatikusan is, de lehetőség van a vezérlésre is. Legyen szó akár automatikus virágöntözésről, vagy automatikus redőny le- és felengedésről, vagy hőmérsékletszabályozásról, ezek a rendszerek szinte bármilyen feladatot elláthatnak, amik segítik a mindennapjainkat.

Egy másik, igen csak ígéretes ágazata az IoT-nek az ipari felhasználás, vagyis az IIoT (Industrial Internet of Things). Az otthoni felhasználáshoz hasonlóan itt is különböző szenzorok mérési adataival dolgoznak egy hálózaton keresztül, viszont itt jellemzően érzékenyebb szenzorokat használnak, és nagyobb adattömegeket is kell feldolgozzanak. Az IIoT-t széleskörben lehet használni az iparban, legyen szó akár gyártásról, akár szállításról. [2]

Összességében ez egy gyors ütemben fejlődő technológia, ami rengeteg helyen képes megkönnyíteni vagy éppen hatékonyabbá tenni az életünket. Jelen projekt is beleillik ebbe a témakörbe, ugyanis szenzorok és vezérlők kommunikálnak egymással egy hálózaton keresztül, valamilyen „okos” funkciót ellátva.

### 2.3.2. DIY

A "DIY" (Do-It-Yourself) rövidítés azt jelenti, hogy "csináld magad". Ez egy olyan viselkedési forma, amely során az emberek saját maguk készítenek, módosítanak vagy javítanak meg dolgokat, ahelyett, hogy az adott témában jártas személy vagy szakértő segítségét vennék igénybe. Rengeteg tényező motiválhat erre, mint például az alkotás vágya, a tanulás, de akár az is, hogy a saját magunk által elkészített eszköz olcsóbb, megbízhatóbb, minőségibb mint az aktuálisan megvásárolható termékek.

A DIY filozófiája alapvetően arra ösztönözi az embereket, hogy használják kreativitásukat, képességeiket és ismereteiket ezeknek a projektek megvalósítására. Ez a tevékenység számos területen elterjedt, beleértve az otthoni javításokat, a barkácsolást, az elektronikai projekteket, a kertészkedést, az ékszerkészítést és sok más hobbit vagy alkotói területet.

A DIY projektjeik során az emberek általában saját készleteket, eszközöket és anyagokat használnak, és a szükséges lépéseket és eljárásokat önállóan tanulmányozzák és alkalmazzák. Az interneten számos forrás található, amelyek segítséget nyújtanak az embereknek a DIY projektek megvalósításához, beleértve az útmutatókat, videókat, fórumokat és közösségi platformokat. [3]

Összességében ez tulajdonképpen egy mentalitás, ami arra vonatkozik, hogy amiket önerőből otthon is képesek vagyunk elvégezni, azt végezzük is el mi magunk. Ezzel valószínűleg nagymértékben fejlesszük a saját képességeinket is, miközben valamilyen hasznos eszközt vagy rendszert építünk ki. Jelen projekt is egy DIY projekt, mivel egy ilyen irányjelző rendszert akár meg is lehet vásárolni minden bizonnyal, viszont a saját rendszer valószínűleg olcsóbb is, és megfelelő befektetett energiával még akár hatékonyabb is lehet.

### 2.3.3. Mikrovezérlők

A mikrovezérlő (MCU – *microcontroller unit*) egy kisméretű számítógép, ami tartalmaz egy vagy több processzort (vagy processzormagot), memóriát és programozható ki- és bemeneteket. Ezek mellett gyakran található FRAM, NOR flash vagy OTP ROM formájú programmemória, valamint kisebb mennyiségű RAM is.

A mikrovezérlők programozhatóak, ami azt jelenti, hogy teljesen testreszabható a működésük, így rengeteg területen fel lehet őket használni, legyen szó ipari felhasználásról vagy valamilyen magán célú DIY projektről. Alacsony energiafogyasztásuk és kompakt méretük miatt ideálisak szinte bármilyen rendszernek a vezérlésére és automatizálására. Ezek a kisméretű eszközök rengeteg lehetőséget rejtenek magukban. Napjainkban rengeteg fajta mikrovezérlő létezik, csak meg kell találni azt, amelyik leginkább kielégíti az adott projekt igényeit. [4]

Ebben a projektben, amikor ki kellett választanom a megfelelő mikrovezérlőt, a legfontosabb szempont az volt, hogy minél kisebb legyen a mérete, valamint, hogy képes legyen valamilyen vezeték nélküli kommunikációra külső modul nélkül is.

#### 2.3.4. Gyorsulásmérő és giroszkóp

A giroszkóp és a gyorsulásmérő olyan szenzorok, amelyek képesek érzékelni és mérni a készülék mozgását és orientációját a térben. Ezek a szenzorok fontos szerepet játszanak a beágyazott rendszerekben, az okostelefonokban, a drónokban, az önvezető járművekben és más olyan alkalmazásokban, ahol a mozgásérzékelés és a térbeli orientáció meghatározása szükséges. A szenzorok működésének részletesebb leírása megtalálható az *Érzékelők és mérőhálózatok* [5] című könyvben.

A **gyorsulásmérő** a készülék gyorsulását érzékeli és méri a térbeli irányokban (X, Y, Z). A mért értéket  $g$ -ben fejezzük ki ( $g = 9,81 \text{ m/s}^2$ ) ami a Föld gravitációs gyorsulása. A gyorsulás mérésének két alapelve van:

- A tehetetlenségi erő mérése ( $F = m \cdot a$ , ahol  $m$  – tömeg és  $a$  - gyorsulás)
- Felület vagy pont kilengésének a mérése ( $x$ )

Az általam használt gyorsulásmérő közvetett erőméréssel működik. Ennek egyik fajtája, miszerint egy  $m$  tömegű test rá van kötve egy  $k$  rugóállandójú rugóra, amelynek a másik vége rögzített. Gyorsulás hatására a test egy  $x$  elmozdulást ér el. Adott a következő képlet:

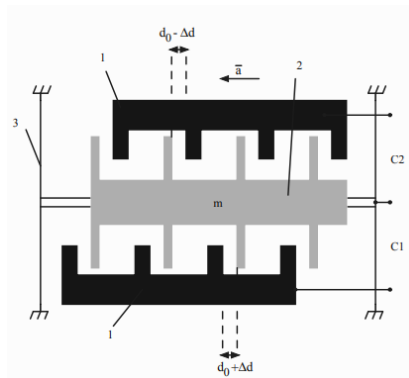
$$F = m \cdot a = k \cdot x$$

Ebből kifejezhető a gyorsulás:

$$a = \frac{k}{m} \cdot x$$

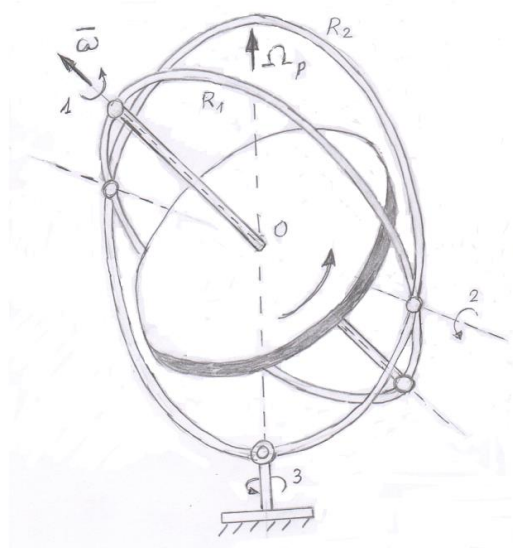
Ezt az  $x$  elmozdulást, ha megmérjük, akkor kiszámíthatjuk a test gyorsulását. Ezt induktív vagy kapacitív elmozdulásérzékelővel lehet mérni. Ebből a legelterjedtebb típus a kapacitív gyorsulásmérő. A projektben használt gyorsulásmérő szintén egy kapacitív gyorsulásmérő, abból is az úgynevezett *Fésűs gyorsulásmérő*.





2. ábra Fésűs gyorsulásmérő [5]

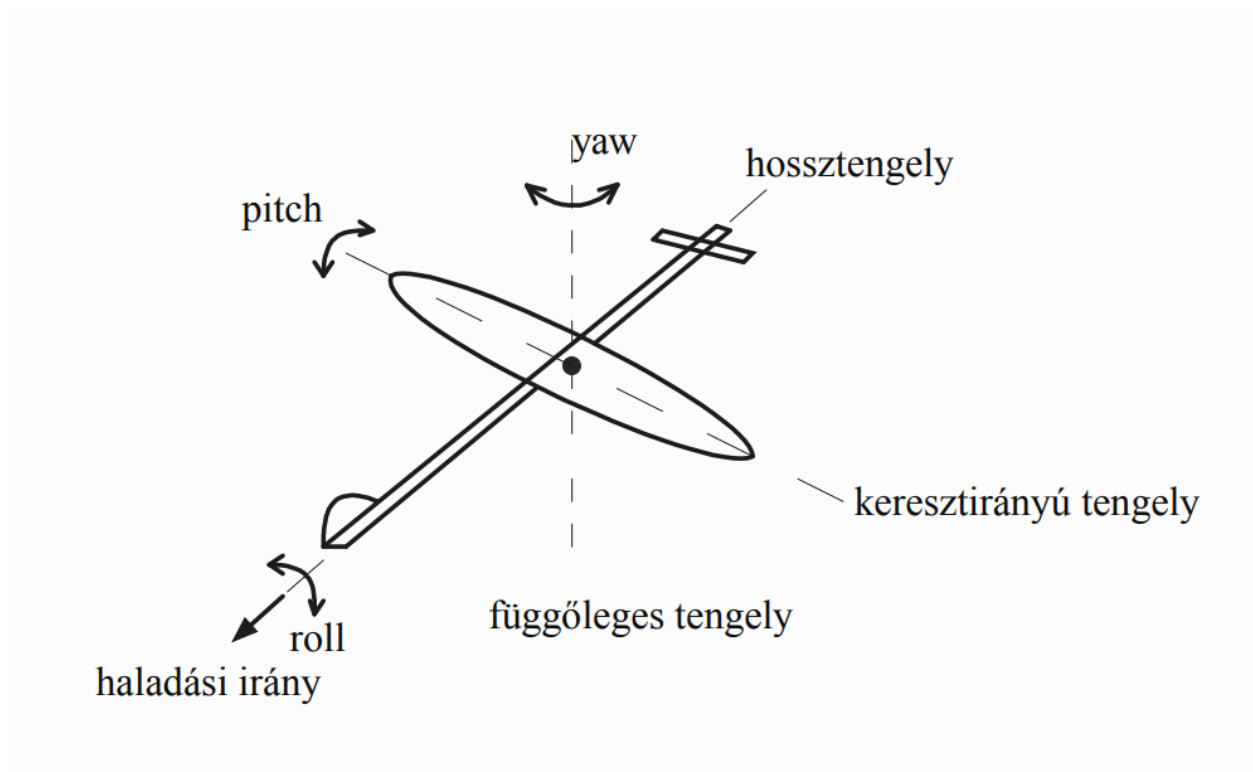
A **giroszkóp** a forgás mértékét és sebességét méri az adott tengely körül. Jelen esetben az én szenzorom egy rezgő giroszkópot tartalmaz, ami a Coriolis-erőn alapul. A szenzor MEMS (Micro-Electro-Mechanical Systems) technológiát használ, vagyis nagyon kis méretű mechanikus és elektromechanikus elemekből áll, amelyek a forgás hatására megváltoztatják a rezgés frekvenciáját, ezáltal a giroszkóp képes érzékelni és mérni a forgatás mértékét és irányát.



### 3. ábra Szabad forgó giroszkóp

Egy jármű mozgásakor három egymásra merőleges tengely menti elmozdulást határozunk meg:

- forgás (yaw) – függőleges tengely körüli forgás
- hosszdzölés vagy bólíntás (pitch) – a keresztirányú tengely körüli forgás
- orsózás (roll) – a hossztengey körüli forgás



4. ábra Roll, Pitch, Yaw

Ezekkel a szögekkel (4. ábra) pontosan meghatározható egy objektum pozíciója a térben. Leggyakrabban a repülőgépek és drónok irányítása esetén használják ezeket a szögeket, de más járművek esetén is használatos.

A giroszkóp és a gyorsulásmérő kombinálása lehetővé teszi a pontosabb és megbízhatóbb mozgásérzékelést és térbeli orientációt. Általában ez a két érzékelő egy készüléken belül helyezkedik el (mint például az általam használt szenzorban is) és így az eszköz 6 úgynevezett szabadságfokkal rendelkezik (DOF – Degrees Of Freedom), vagyis a gyorsulásmérő által mért 3 tengely (X, Y és Z), valamint a giroszkóp által mért 3 tengely összessége. Lehetőség van kiegészíteni a rendszert még egy magnetométerrel is, ami a Föld mágneses mezőjét felhasználva érzékeli az eszköz irányát, így ezzel 9 szabadságfokot is el lehet érni a pontosabb mérési eredmények érdekében. Ezt az eljárást, amely során több szenzor által mért adatokat kombinálunk a pontosabb mérések érdekében, úgy nevezzük, hogy *szenzorfüzió* (*sensor fusion*). [6]

Összefoglalva ezeknek a szenzoroknak fontos szerepük van a repülőgépek és más járművek irányításában, mint a hajók vagy az önvezető autók, de használatosak bármilyen projektben ahol fontos a térbeli orientáció meghatározása. Ebben a projektben a kéz és az azon

levő szenzor orientációját kell meghatározzuk, ahhoz hogy képesek legyünk ez alapján jelzéseket küldeni. A giroszkóp és a gyorsulásmérő elengedhetetlen részét képezik a projektnek.

### **2.3.5. Webszerver**

Egy webszerver egy olyan számítógépes rendszer vagy alkalmazás, amely HTTP-n (HyperText Transfer Protocol) vagy annak biztonságos változatán a HTTPS-en fogad kéréseket. A kliens (általában webböngésző) kezdeményezi a kommunikációt azzal, hogy kérést küld a szervernek HTTP-n keresztül és a szerver válaszol a kért tartalommal, vagy egy hibaüzenettel, amennyiben nem tudta teljesíteni a kérést.

Egy webszerverhez szükséges hardver változhat, annak függvényében, hogy mennyi kérést kell kezeljen. Ez alapján a kis beágyazott rendszerektől, mint egy router ami egy kis webszervert futtat, egészen a nagy forgalommal rendelkező webhelyekig, amik akár több száz szerverrel is dolgozhatnak és ezek is rengeteg nagy teljesítményű számítógépekből állnak, mindenféle webszerver megtalálható manapság. [7]

Jelen projekt esetében a fentebb említett skála alsó részéről beszélhetünk, ugyanis egy kis mikrovezérlő futtatja a webszervert, aminek minimális számú kérést kell feldolgoznia, így nem is szükségeltetik erősebb hardver erre a feladatra.

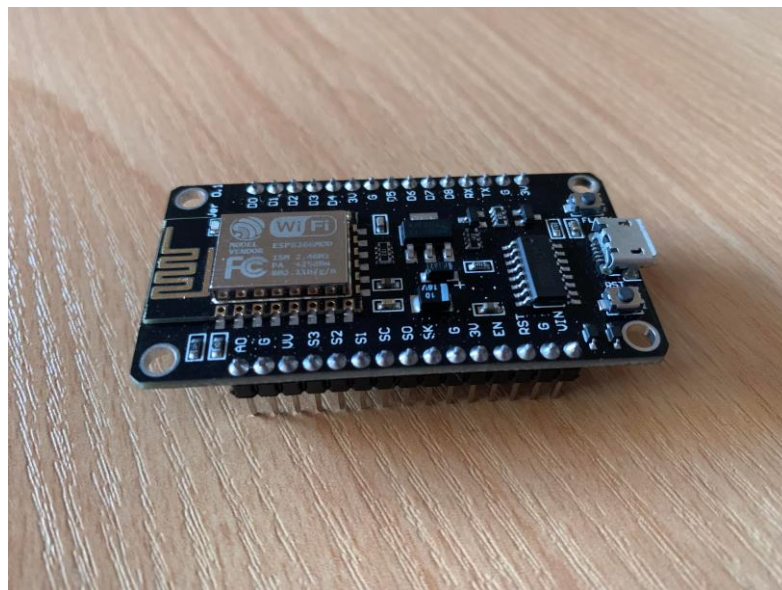
## **2.4. Felhasznált eszközök**

### **2.4.1. ESP32 és ESP8266**

Az általam használt két mikrovezérlő közül az egyik egy ESP32 alapú (5. ábra), a másik egy ESP8266 alapú (6. ábra) mikrovezérlő. Ezek a mikrovezérlők hasonló funkciókat kínálnak, néhány kisebb-nagyobb eltérés mutatkozik csak a kettő között. A projektek követelményei alapján kell kiválasztani azt, amelyik teljesíti az adott igényeket.



5. ábra ESP32



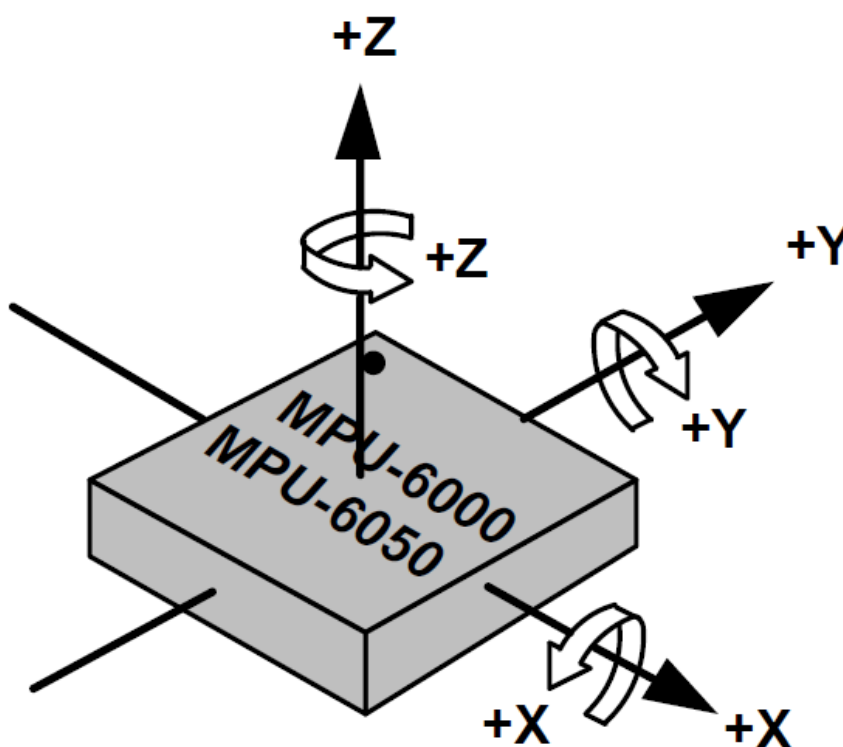
6. ábra ESP8266

Mindkét mikrovezérlő rendelkezik beépített WiFi modulokkal, ezért kiválóak bármilyen IoT projekt megvalósítására. A mikrovezérlők adatlapja szerint [8] [9] [10] specifikációban nagyrészt hasonlítanak egymásra, csupán néhány különbség mutatkozik. Ilyen különbség például, hogy az ESP32 rendelkezik Bluetooth modullal is, de az ESP8266 nem. Ezen kívül is megfigyelhetők különbségek a két mikrovezérlő között, mint a beépített processzorok típusa és órajele, valamint az eszközökbe épített memóriák típusa és mérete is különbözik egymástól.

A projekt során nem volt számottevő a két mikrovezérlő közötti különbség, ugyanis a feladataikat tökéletesen ellátták.

#### 2.4.2. MPU6050

Az MPU-6050 az adatlap szerint [11] egy integrált 6-tengelyes mozgásérzékelő eszköz, amely a világ első olyan eszköze, amely egyetlen kis 4x4x0,9 mm-es kiserelésben kombinál egy 3-tengelyes giroszkópot, egy 3-tengelyes gyorsulásmérőt, valamint egy DMP-t (Digital Motion Processor).



7. ábra MPU6050 tengelyek

Maga a szenzor egy 6 tengelyes MotionFusion kimenetet biztosít, viszont a beépített I2C busszal képes közvetlenül fogadni egy külső szenzor adatait is, így biztosítva egy 9 tengelyes MotionFusion kimenetet is akár.



8. ábra MPU6050

Az MPU6050 biztosít három-három 16 bites ADC-t (Analog-to-Digital Converter) külön a giroszkóp és külön a gyorsulásmérő kimeneteinek a digitalizálásához. Mind a giroszkóp, mind a gyorsulásmérő érzékenysége programozható, így testre lehet szabni, hogy mekkora skálán szeretnénk mérni, legyen szó így akár kisebb, akár nagyobb mozgások észleléséről.

Alább látható néhány technikai adat, összehasonlítva az MPU6000-al ami azonos családba tartozik mint az MPU6050, csupán néhány eltérés van a kettő között:

Part / Item	MPU-6000	MPU-6050
VDD	2.375V-3.46V	2.375V-3.46V
VLOGIC	n/a	1.71V to VDD
Serial Interfaces Supported	I <sup>2</sup> C, SPI	I <sup>2</sup> C
Pin 8	/CS	VLOGIC
Pin 9	AD0/SDO	AD0
Pin 23	SCL/SCLK	SCL
Pin 24	SDA/SDI	SDA

9. ábra Mpu6000 és MPU6050 összehasonlítása [11]

### 2.4.3. LED

A LED (light-emitting diode) vagyis fénykibocsátó dióda olyan félvezető eszköz, ami képes fényt kibocsájtani, ha áramot vezetnek át rajta. A LED-ek széles körben használatosak a világítástechnikában, az elektronikában és más iparágakban, de még a mindennapi életben is elég elterjedtek manapság köszönhetően számos előnyüknek az izzókkal szemben. Ilyen előny a kisebb



fogyasztás, a hosszabb élettartam, a kisebb méret és még sok más olyan tulajdonság, ami miatt sokkal kifizetődőbb ezeknek az eszközöknek a használata.

A LED-ek az elektroluminiscencia elvére épülnek. Az elektroluminiscencia egy olyan fizikai jelenség, amelyben az elektronok és az elektronlyukak rekombinációja következtében fotonok (fényrészecskék) keletkeznek a félvezető anyagban. Amikor áram folyik keresztül a LED-en, fényt bocsájt ki magából, legyen az látható vagy éppen nem látható tartományban (mint az UV). [12]

## **2.5. Felhasznált technológiák**

### **2.5.1. I2C**

Az I2C (Inter-Integrated Circuit) egy mérőinterfész, amit a Philips Semiconductor fejlesztett ki 1982-ben. Több mester- (master) és több szolga (slave) eszköz közötti soros kommunikációra képes. Nagyon elterjedt kommunikációs mód a mikrovezérlők és az érzékelők között. Egyik legnagyobb előnye a konkurenciához képest, hogy csupán két vezetékét használ az adatok közvetítésére. Az egyik vezeték az SDA (Serial Data Line) ami az adatok továbbításáért felel, a másik vezeték az SCL (Serial Clock Line) ami az órajelet biztosítja. [5] [13]

Ebben a projektben a szenzor és a mikrovezérlő közötti kommunikáció valósul meg ennek a technológiának a segítségével.

### **2.5.2. HTTP**

A HTTP (HyperText Transfer Protocol) egy alkalmazásrétegű protokoll hipermédiás dokumentumok, mint a HTML továbbítására, amit webböngészők és webszerverek közötti kommunikációra terveztek, de használható más célokra is. A protokoll a klasszikus kliens-szerver modellt követi, vagyis a kliens kezdeményezi a kapcsolatot a szerver felé egy kérés formájában, amire a szervernek kell válaszolnia.

Az HTTP kommunikáció során a kliens (általában egy webböngésző) HTTP kérést küld a szervernek, amelyben meghatározza, hogy milyen adatokat vagy műveleteket szeretne elérni. A kérés tartalmazza a típust, ami lehet GET, POST, PUT, DELETE, stb., az erőforrást (URL) és még tartalmazhat opcionális fejléceket és adatokat is. A szerver fogadja a kérést, feldolgozza és választ küld a kliensnek. Ez a válasz tartalmaz egy kódot, ebből a legismertebb a 200, ami az OK, valamint a 404, ami a Not Found, vagyis nem található jelentéseket hordoznak magukban. Ezen kívül a válasz tartalmazhat még opcionális fejléceket is. [14]

A HTTP-t a webszerver létrehozásánál és futtatásánál használtam. Ezt a szerveret a mikrovezérlő futtatja, és ez felel a webes felhasználói felület létrehozásáért.

### **2.5.3. WebSocket**

A WebSocket egy kommunikációs protokoll, ami kétirányú duplex kommunikációt tesz lehetővé egyetlen TCP protokollon keresztül. Magát a protokollt az IETF (Internet Engineering Task Force) standardizálta 2011-ben RFC 6555 néven.

A WebSocket lehetőséget ad a kliens, mint a webböngésző, vagy bármilyen alkalmazás és a webszerver közötti kétirányú valós idejű kommunikációt, anélkül, hogy a kliensnek kérést kellene intéznie a szerver felé. [15]

A projektben ezt a kommunikációs protokollt használtam a valós idejű adattovábbításra a két mikrovezérlő között, valamint a szerver és a böngésző között is.

### **2.5.4. LittleFS**

A LittleFS egy mikrovezérlőkre tervezett fájlrendszer, ami a flash memóriát használja. Fontos megemlíteni az SPIFFS fájlrendszert is, ami ennek az alapját képezi, viszont a LittleFS egy újabb és gyorsabb fájlrendszer, ami aktív fejlesztés alatt áll, így ajánlatos a címben is szereplő fájlrendszert használni a régivel szemben. [16]

Ezt a fájlrendszert használom arra, hogy a webalkalmazáshoz tartozó HTML, CSS és JavaScript fájlokat a mikrovezérlő flash memóriájában tároljam, így amikor egy böngészőtől kérést kap, akkor el tudja küldeni ezeket a kért adatokat és megjeleníteni a weboldalt a felhasználók számára.

## **3. A rendszer specifikációi és architektúrája**

A projekt célja egy hajtányt ellátni különböző okos funkciókkal, amiknek a segítségével képes lesz a vezető csupán kézmozdulatok segítségével jelezni a forgalom számára az irányváltoztatás szándékát. Emellett a rendszer képes kell, hogy legyen észlelni a jármű gyorsulását és automatikusan jelezni a fékezést is.

Ezekkel a funkciókkal a rendszer nagyban elősegítené a biztonságos közlekedést a különböző hajtányok vezetőinek a számára, ugyanis a megfelelő kommunikáció a forgalomban levők között elengedhetetlen része a biztonságos közlekedésnek. Ezt a kommunikációt szeretné a projekt megvalósítani a hajtányok esetében.



### 3.1. Rendszer specifikációi

A rendszer alapja a két mikrovezérlő, amik egyenként ellátják a maguk fontos feladatát. Az első mikrovezérlő felelős a szenzor adatainak az olvasásáért és az adatok feldolgozásáért, valamint a feldolgozott információ továbbításáért. A második mikrovezérlő felelős a webszerver futtatásáért, valamint azért, hogy a kapott információk alapján a megfelelő led-eket világítsa meg. Maga a webszerver mint egy információ közvetítői réteg szerepel a két mikrovezérlő között, valamint ez szolgálja ki a klienseket is, akik el szeretnék érni a weboldalt. Így a rendszer fontos szereplője egy vagy akár több kliens is, akik a felhasználói felületnek szánt weboldalon követik nyomon a rendszer állapotát, és bele is avatkozhatnak a rendszer működésébe bizonyos interakciók segítségével.

#### 3.1.1. Funkcionális követelmények

A rendszernek rengeteg olyan folyamatot el kell látnia, amik közül, ha akár egy is kiesik, akkor az egész működésképtelenné válhat.

Az első és egyben az egyik legfontosabb feladat az MPU6050 szenzor által mért adatoknak az olvasása és feldolgozása. Erre a feladatra az első számú mikrovezérlőt használtam (NodeMCU-32s). A mikrovezérlőt összekötve a szenzorral, valamint a megfelelő könyvtárcsomag használatával viszonylag egyszerűen le lehet kérdezni a szenzor által mért aktuális adatokat. A szenzortól lekérdezhetők a gyorsulásmérő által mért 3 koordináta szerinti és a giroszkóp által mért 3 koordináta szerinti adatok, valamint egy hőmérséklet is. Ez utóbbi nem a környezeti hőmérséklet mérésére szolgál, hanem a szenzor kalibrációjához szükséges a pontosabb mérési eredmények érdekében. Ezekből a szenzor által mért nyers adatokból a mikrovezérlő kiszámolja a roll illetve pitch szögeket. Ezeket az adatokat a mikrovezérlő folyamatosan figyeli és egy szigorú feltételrendszer segítségével eldönti, hogy mikor történt jelzés, amit továbbítania kell.

A következő fontos feladat a második mikrovezérlőhöz (Node MCU V3 - LoLin (Wi-Fi)) tartozik, mégpedig a webszerver futtatása. Ez ezért nagyon fontos, mert nem csupán az adatok megjelenítéséért felel ez a webserver, hanem ez szolgál kommunikációs csatornaként is a két mikrovezérlő és a kliens (egy a szerverhez csatlakozott böngésző) között is. Így elengedhetetlen ez a funkció a rendszer működése szempontjából. Ugyanez a mikrovezérlő vezérli a jelzésre szánt led-eket is, ami szintén egy nagyon fontos funkcionalitás, hiszen ez a leglátványosabb része az egész rendszernek.

A harmadik funkcionalitás a felhasználói felület, ami egy weboldal. Ezen keresztül figyelhetik a felhasználók a rendszer állapotát, vagyis láthatják az aktuális jelzéseket ezen a felületen is. Emellett interakcióba is léphetnek a rendszerrel, ugyanis a felhasználói felület erre is biztosít lehetőséget. A felhasználók egy gomb segítségével képesek engedélyezni és tiltani a jelzéseket.

Összefoglalva a rendszernek több funkcionalitást is el kell látnia, amik kölcsönhatásban vannak egymással. Mindegyik funkcionalitás kellően fontos a rendszer működése szempontjából. A szenzor adatok olvasása és feldolgozása úgyszintén az alapvető funkciókhoz tartozik, mint a webservert és a felhasználói felület. Ezeknek a segítségével áll össze a rendszer nagy egészként.

### **3.1.2. Nem-funkcionális követelmények**

#### **3.1.2.1 Használhatóság és rendszerfüggetlenség**

A rendszer teljeskörű használata érdekében szükség van egy okostelefonra, vagy valamilyen olyan okos eszközre, ami rendelkezik WiFi modullal és képes használni egy webböngészőt. Ez az okos eszköz rendelkezhet bármilyen operációs rendszerrel, ami képes szolgáltatni az előbb felsorolt funkciókat.

Az okoseszköz használatára azért van szükség, hogy a felhasználó képes legyen az erre a célra szánt felületen nyomon követni a rendszer állapotát, valamint beleszólni a működésébe. Ezzel lehet kihasználni teljes mértékben a rendszer kínálta lehetőségeket, azonban ez nem egy kötelező követelmény. E nélkül is működőképes marad a rendszer, csak lemond a felhasználó arról, hogy megfigyelje az állapotokat és beleszóljon a működésbe.

#### **3.1.2.2 Adatvédelem és biztonság**

Adatvédelem szempontjából a rendszer nincs kitéve támadásnak, ugyanis lokális hálózaton történik a kommunikáció, így ameddig csak a felhasználó birtokolja a mikrovezérlő WiFi hálózatának a jelszavát, ami feltehetően egy komplex jelszó, akkor nem érdemes attól félni, hogy illetéktelenek hozzáférnek a rendszer információihoz és beleszólnak a működésébe.

Mivel a rendszer nem tárol és nem is használ semmilyen kényes információt, mint például felhasználói adatokat és jelszavakat, így nem szükséges semmilyen adatvédelmi intézkedés, mint a titkosítás a kommunikáció során.

#### **3.1.2.3 Környezeti tényezők**

Fontos környezeti tényező tud lenni a nedvesség és a por, ugyanis ezek az elektronikai eszközök nem rendelkeznek semmilyen víz- és porállósági tanúsítvánnyal, ezért fontos odafigyelni ezekre.

Ugyanilyen környezeti tényező lehet a hőmérséklet is, ami beleszólhat a szenzor mérési eredményeibe, viszont azért, hogy nem szükségesek a pontos mérések, így a szenzor szempontjából ez nem okozhat nagy problémát. A mikrovezérlők üzemi hőmérsékletének a maximuma körülbelül 85 Celsius fok, így ha az eszköz huzamosabb ideig valamilyen hőforrásnak van kitéve, akkor könnyen túlmelegedhet. Emellett fontos kiemelni az akkumulátorokat is, amik sokkal érzékenyebbek a szélsőséges hőmérsékleti viszonyokra, mint a többi komponens, így azoknál különösen kell figyelni, hogy ne melegedjenek túl.

#### **3.1.2.4 Felhasználói felület**

Fontos követelmény továbbá, hogy a felhasználói felület átlátható és intuitív legyen. A felhasználóknak első perctől legyen világos, hogy az oldalon megjelenő elemek, milyen funkcionalitással rendelkeznek. Az ikonoknak és a gomboknak megfelelő kell, hogy legyen a mérete, hogy egy telefon kijelzőjén is átlátható és kényelmesen használható legyen.

#### **3.1.2.5 Teljesítmény**

A teljesítmény szempontjából a rendszer nincs kitéve a túlterhelésnek, ugyanis a megfelelő számításokat el tudja végezni, valamint az általános használat során csupán egy szerverre csatlakoztatott eszközt kell kiszolgálnia, így nem valószínű, hogy túlterhelődik a rendszer.

#### **3.1.2.6 Mobilitás**

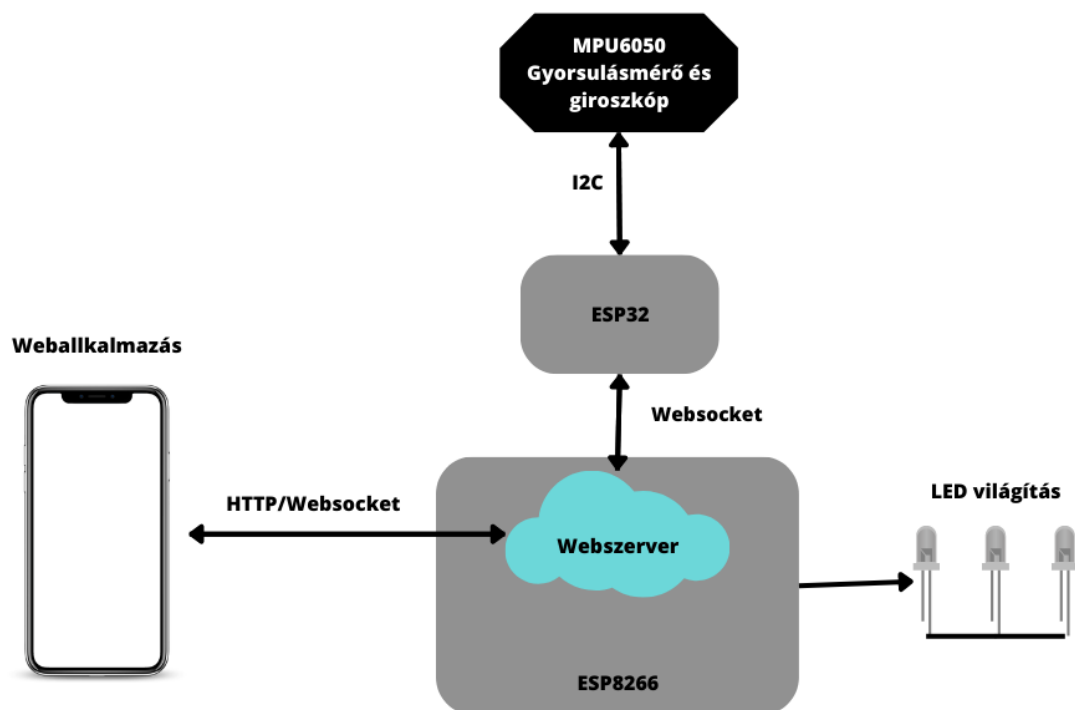
A rendszernek teljes mértékben mobilisnak kell lennie, mivel a használata egy jármű használata során történik. A rendszer kell, hogy biztosítson magának egy saját kommunikációs csatornát (WiFi hálózatot) ami mindig elérhető és a rendszerkomponensek közötti adatcserét biztosítja. Emellett fontos az áramellátás is, így a rendszer kell, hogy rendelkezzen megfelelő méretű akkumulátorokkal egy esetleges több órás üzemidő elérése érdekében.

#### **3.1.2.7 Skálázhatóság**

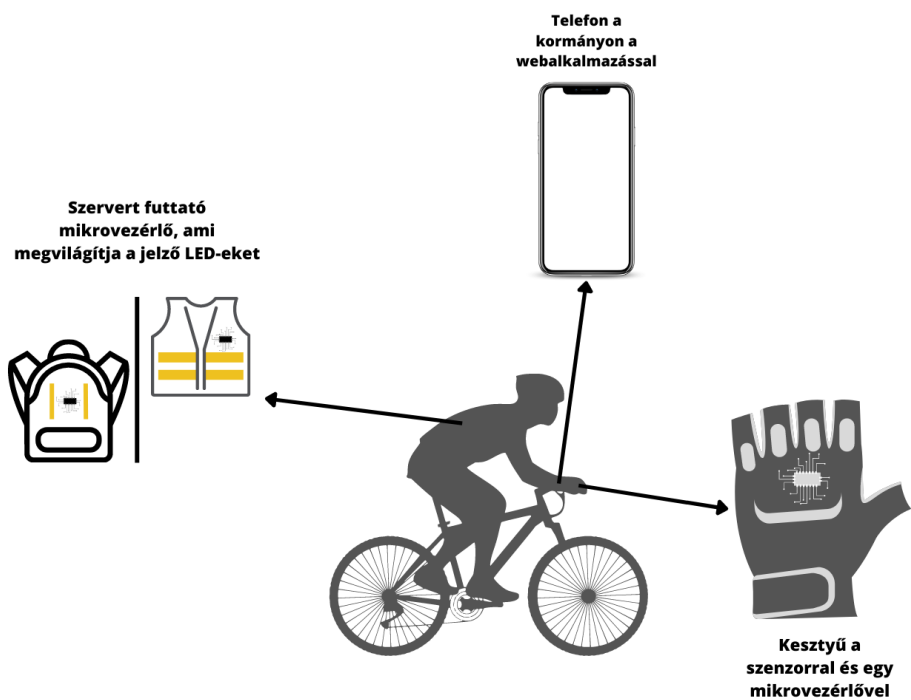
Lehetőség kell legyen arra, hogy a rendszert további eszközökkel lássuk el. Ilyen eszköz lehet egy vagy több szenzor, amiknek a segítségével pontosíthatjuk a meglévő méréseket, valamint további funkcionalitásokkal láthatjuk el a rendszert.

### **3.2. Rendszer architektúra**

A rendszer architektúra szempontjából leírható a különböző komponensekkel és az ezek közötti kapcsolattal. Ez a struktúra az alábbi ábrán látható.



10. ábra A rendszer egyszerű vázlata I.



11. ábra A rendszer egyszerű vázlata II.

### 3.2.1. Rendszerkomponensek

A rendszer komponensei közé tartozik a két mikrovezérlő, a webalkalmazás, a szenzor és a led-ek, valamint a középen elhelyezkedő webszerver. Jól látható az ábrán (*10. ábra A rendszer egyszerű vázlata*), hogy a webszerver az a tényező, amely összekapcsolja a meglévő komponenseket.

Az ESP32 mikrovezérlő felel a szenzor adatok olvasásáért és azért, hogy ezeket a szervernek továbbítsa. A másik mikrovezérlő, az ESP8266 felel az adatok fogadásáért és a led-ek megvilágításáért, valamint ezen a mikrovezérlőn fut a webszerver is. A webalkalmazás azért felel, hogy az adatok meg legyenek jelenítve, valamint, hogy interakciót biztosítson a felhasználó számára. Jól látható, hogy mindegyik komponensnek a kapcsolata kétirányú a webszerverrel, vagyis a kommunikáció oda-vissza történik.

### 3.2.2. Komponensek közötti kapcsolat és kommunikáció

Az ESP32 mikrovezérlő a szenzorral egyirányú kommunikációt folytat nagyrészt. Azért csak nagyrészt, mert a szenzor inicializálásakor be lehet állítani a szenzor mérési skáláját, de ez opcionális. Ezen kívül a szenzor és a mikrovezérlő között egyirányú kommunikáció folyik I2C protokollt használva. Ez az elem a webszerverrel kétirányú kapcsolatban van és websocket-en keresztül küldi, valamint fogadja az adatokat.

Az ESP8266 mikrovezérlő közvetlen kapcsolatban áll a led-ekkel, mivel megvezéri azokat. Továbbá ez a mikrovezérlő felel a webszerver futtatásáért is.

A webalkalmazás szintén kétirányú kapcsolatban van a webszerverrel. A statikus elemeket, amik a weboldal megjelenéséért felelnek azokat HTTP-n keresztül kapja, a folyamatosan változó jelzésekre vonatkozó adatokat pedig websocket-en fogadja és megjeleníti azokat, valamint a felhasználói interakciókat visszaküldi a webservernek szinté websocket-en keresztül.

A webszerver a kommunikáció középpontjában áll. Folyamatosan fenntartja a websocket kommunikációt a különböző elemek között, valamint figyeli a beérkező HTTP kéréseket is, amiket a csatlakoztatott eszközöktől kap a webalkalmazás elérésére vonatkozóan.

## 4. Részletes tervezés és gyakorlati megvalósítás

A projektet tervezés szempontjából 3 fő csoportra lehet osztani a szoftverek szerint:

- A szenzor adatokat feldolgozó mikrovezérlő (ESP32) szoftvere
- A webszervert futtató mikrovezérlő (ESP8266) szoftvere
- Felhasználói felület

## 4.1. A szenzor adatokat feldolgozó mikrovezérlő szoftvere

Ez egy ESP32 alapú mikrovezérlő, ami össze van kötve egy MPU6050 szenzorral. A továbbiakban a szoftver fontosabb részeit mutatom be.

### 4.1.1. Inicializálás

A mikrovezérlő első feladata a szenzor, valamint a vezeték nélküli kommunikáció inicializálása a `setup` függvényben. Előbbi szükséges ahhoz, hogy a szenzorból érkező adatokat a továbbiakban tudjuk olvasni, utóbbi pedig azért fontos, hogy a feldolgozott adatokat vezeték nélkül tudjuk továbbítani a webszervernek.

A szenzor inicializálása úgy történik, hogy kezdetben létrehozunk egy szenzor objektumot a következő módon:

```
Adafruit_MPU6050 mpu;
```

Ezt követően az `mpu.begin()` parancs segítségével inicializáljuk a szenzort, ezt követően készen áll a használatra. A szenzor mérési skáláját is itt érdemes beállítani, ezt a következő példa alapján lehet elvégezni:

```
mpu.setAccelerometerRange(MPU6050_RANGE_8_G);  
mpu.setGyroRange(MPU6050_RANGE_250_DEG);  
mpu.setFilterBandwidth(MPU6050_BAND_21_HZ);
```

A következő feladat a vezeték nélküli kommunikáció inicializálása. Első lépésként beállítjuk a WiFi modult STA módba (erről a későbbiekben bővebben lesz szó), majd beállítjuk a websocket-et is a megfelelő címre és hozzáadunk egy eseménykezelőt amin keresztül majd fogadni fogjuk az adatokat a websocket-en keresztül. Ezt az eseménykezelőt arra használjuk, hogy ha valamilyen websocket üzenet érkezik, akkor azt annak megfelelő módon tudjuk feldolgozni. Tartalmaz egy tesztelést a szöveges üzenetekre vonatkozóan, majd ezen belül vizsgáljuk a kapott üzenetet és elvégezzük az annak megfelelő műveletet.

### 4.1.2. Szenzor adatok olvasása és feldolgozása

Ez egy függvényen belül valósul meg, aminek a feladata olvasni az aktuális adatokat a szenzorból, valamint ezek alapján kiszámolni a pitch, illetve roll szögeket, ugyanis a yaw szög pontosabb számításához szükség lenne egy magnetométerre is. Ez a két szög elég ahhoz, hogy a rendszer megfelelően működjön, viszont egy olyan projekt esetén ahol szükséges a még pontosabb térbeli pozíció meghatározása ajánlott egy 9 szabadságfokú szenzorfüzió használata (lásd: 2.3.4)

Jelen esetben a két rendelkezésre álló szenzor segítségével (giroszkóp és gyorsulásmérő) a roll, illetve pitch szögeket számolom ki. A gyorsulásmérő és a giroszkóp adatait egy Complimentary Filter segítségével kombinálom, hogy pontosabban megkapjam a roll és pitch szögeket. Ennek a szűrőnek a lényege az, hogy az egyik szenzor erősségeivel kompenzálja a másik szenzor gyengeségeit és fordítva. Súlyokat használ, amiknek a segítségével, míg az egyik szenzor mérését egy alul áteresztő szűrővel szűrjük, addig a másik szenzor méréseit egy felül áteresztő szűrővel.

A roll és a pitch szögek kiszámítása a következőképpen történik Complimentary Filter használatával C++ nyelven: [6]

1. Kiszámoljuk a gyorsulásmérő adatai alapján a két szöget:

```
rollAcc = atan2(AccY, AccZ) * 180.0 / PI;
```

```
pitchAcc = atan2(-AccX, sqrt(AccY * AccY + AccZ * AccZ)) * 180.0 / PI;
```

2. Kiszámoljuk a giroszkóp adatai alapján is a két szöget

```
dt = 0.01;
```

```
roll += GyroX * dt;
```

```
pitch += GyroY * dt;
```

3. Egy Complimentary Filter segítségével kombináljuk a két szenzor méréseit:

```
alpha = 0.9;
```

```
roll = alpha * roll + (1 - alpha) * rollAcc;
```

```
pitch = alpha * pitch + (1 - alpha) * pitchAcc;
```

Ezek alapján fogjuk majd tudni a szenzor térbeli orientációját és így meghatározni az irányváltásra vonatkozó jelzéseket.

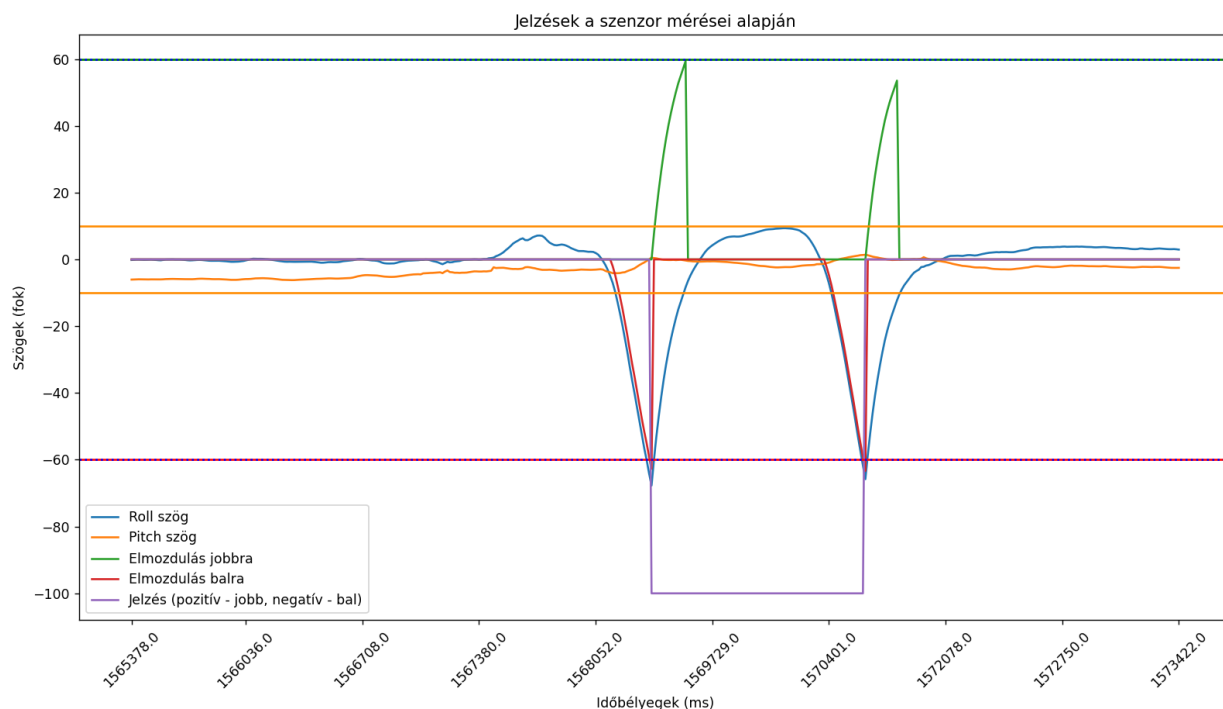
#### **4.1.3. Irányjelzés**

Az irányjelzések vizsgálatát több függvény végzi összedolgozva egymással a korábban kiszámolt két szög alapján. Ha ezek az értékek megfelelnek egy bizonyos feltételrendszernek, akkor beszélünk valamilyen irányváltoztatásra vonatkozó jelzésről.

A kód ezen része folyamatosan figyeli a szenzor aktuális értékeit, és ezeket összehasonlítja a korábbi mért értékekkel, így meghatározza, hogy az adott irányba mekkora az elmozdulás. A feltételrendszer magába foglalja, hogy csak akkor történik jelzés, ha a roll szög a kiinduló

állapothoz viszonyítva is meghalad egy bizonyos küszöbértéket (jelen esetben  $\pm 60$  fok), valamint az abszolút mért szög is meghalad egy értéket (ez is  $\pm 60$  fok), valamint a pitch szög közel van a 0-hoz ( $\pm 10$  fok) tehát majdnem vízszintesen van. Ezeket a küszöbértékeket kísérleti úton határoztam meg.

### Bal irány jelzése:

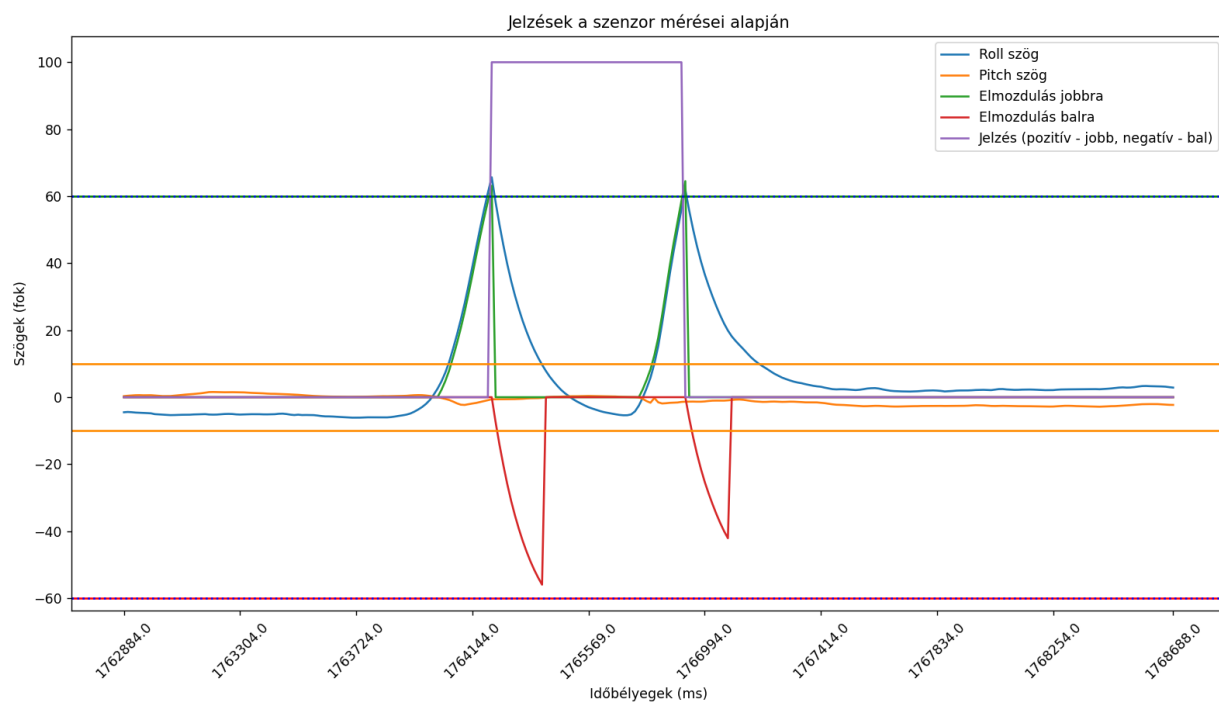


12. ábra Bal irány jelzése

Az ábrán látható, hogy a roll szög és a bal oldali relatív elmozdulás is meghaladja a küszöbértéket, valamint a pitch szög a megszabott határokon belül van. Ennek a hatására megtörténik a jelzés, amit az ábrán egy jól látható kiugró negatív érték jelez, ami a balra jelzésnek felel meg. A következő alkalommal amikor szintén teljesülnek a feltételek, akkor a jelzést megszüntetjük és az állapota visszatér az alapértékre. Miután a szenzor elmozdult balra a feltételeknek megfelelően és ez után visszatér az alapállapotba, látható hogy az ellentétes, vagyis a jobb irányú relatív elmozdulás egy kiugró értéket mutat, viszont ez nem számít jelzésnek, addig ameddig a roll szöggel nem egyszerre haladják ezt a küszöböt meg.

### Jobb irány jelzése:

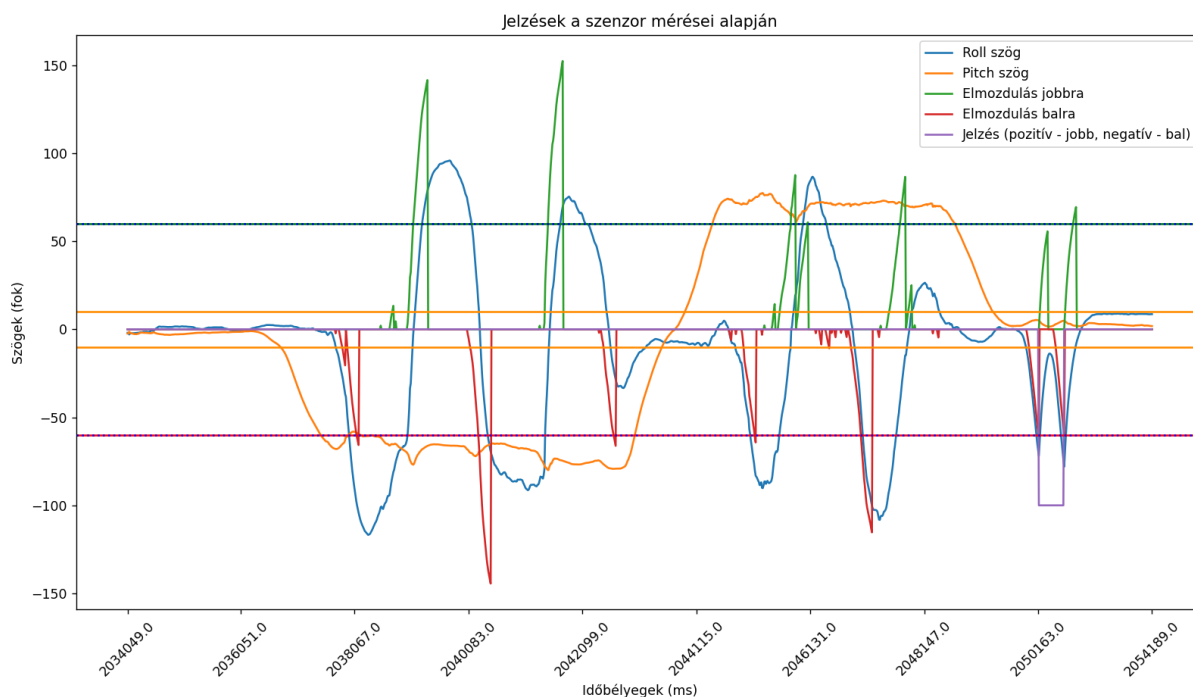




13. ábra Jobb irány jelzése

Az ábrán, az előző jelzéssel ellentétes jelzés, vagyis egy jobb oldali jelzés látható. Itt a roll szög és a jobb oldali relatív elmozdulás is meghaladja egyszerre a megszabott küszöbértéket, valamint a pitch szög is a megfelelő korlátok között marad, így megtörténik a jobb oldali jelzés, amit egy pozitív kiugró érték mutat az ábrán. Megfigyelhető itt is az a jelenség, hogy az ellentétes oldali relatív elmozdulás értéke megnő amikor a szenzor alapállapotba áll vissza, de itt sem okoz ez gondot.

**Hamis jelzés:**



14. ábra Hamis jelzés

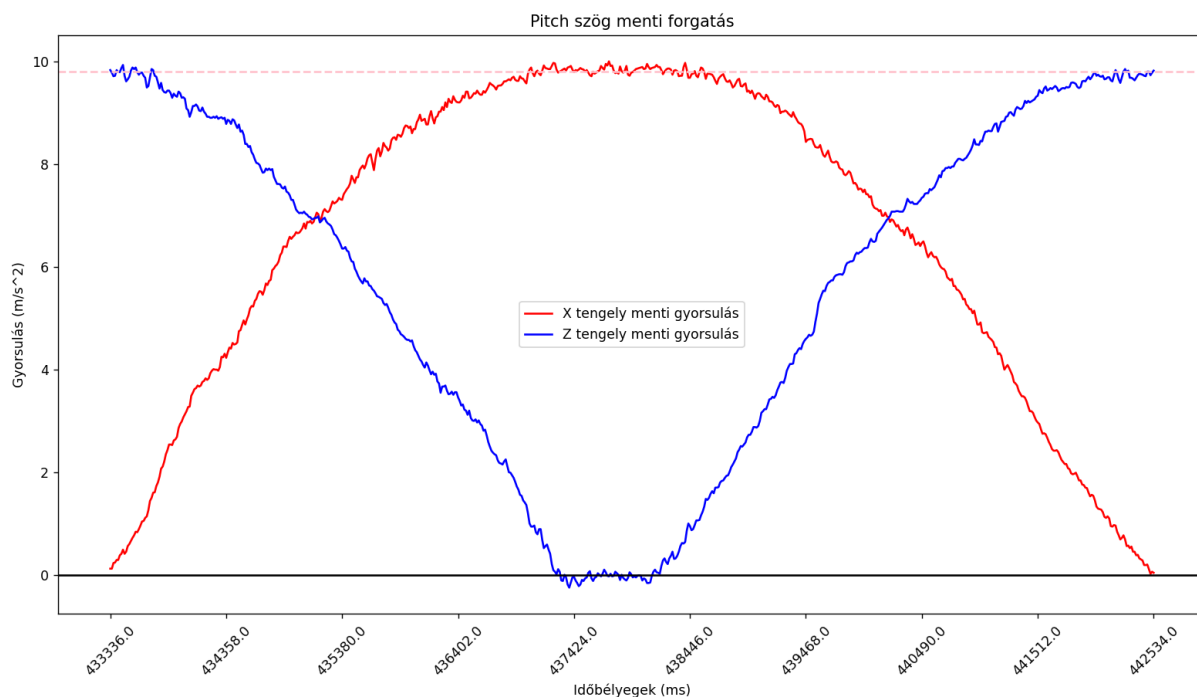
Az ábra megmutatja, hogy mi történik abban az esetben, ha a szenzor mozgása nem felel meg annak a feltételnek, hogy a pitch szög a megszabott keretek között legyen. Mivel a kormányon levő kézen a szenzor pitch szöge egy hasonló értéket venne fel, mint a megszabott korlát, ezért ez is egy feltétel. Így ha a felhasználó másra használja a kezét, akkor kiküszöbölhetők a hamis jelzések. Az ábra mutatja, hogy a roll szög esetén teljesülhetnek a feltételek egy jelzésre, viszont ameddig a pitch szög nem kerül a kiszabott korlátok közé, addig nem történik meg a jelzés. Az ábrán több próbálkozás is látható jobb és bal jelzésekre egyaránt, viszont csak a legutolsónál történik meg a tényleges jelzés, ahol minden feltétel teljesül.

Ezeknek a szigorú feltételeknek a hatására ténylegesen csak akkor jön létre egy jelzés, hogyha kell, így elkerülve a hamis jelzéseket. Az irányváltásnak egy változó tárolja az értékét, ami hogyha nincs jelzés, akkor egy annak megfelelő értéket tárol (például: None). Amennyiben a megfelelő feltételrendszer teljesül, ezt a változót megváltoztatjuk egy olyan értékre, ami az adott jelzést jelenti (például: Left és Right). Ezután ezt az értéket elküldjük a webszervernek websocket-en keresztül, feltéve, hogy az érték nem egyezik meg a változó korábbi értékével. Így csak akkor küldünk jelzést a szervernek, hogyha az adott jelzést tároló változó értéke megváltozott, vagyis ha huzamosabb ideig nem történik jelzés, akkor nem küldünk semmit. Ezzel az eljárással elkerülhető, hogy a kommunikációs csatorna túlterhelődjön és valamilyen kommunikációs hiba lépjen fel.

Amennyiben teljesülnek a feltételek és a jelzést tároló változó értéket vált, egyből el is küldjük ezt websocket-en keresztül a webszervernek és addig marad a változó abban az értékben, ameddig nem történik még egyszer meg, hogy valamelyik feltétel teljesülésével jelzés generálódik. Így az első jelzést generáló mozdulatsor jelzést generál, az azt követő mozdulatsor, ami a feltételrendszernek megfelel, pedig kioltja a jelzést. Ezzel a módszerrel, ha jelezünk egy valamilyen irányú kanyart, az a jelzés addig élni fog, ameddig nem jelzünk egy azonos, vagy ellentétes irányú kanyart, amivel kiolthatjuk az előzőt.

#### 4.1.4. Fékjelzés

A fékjelzéshez kezdetben csupán az X tengelyen mért gyorsulást vizsgálta a rendszer, viszont gyorsan rá lehetett jönni, hogy ez nem megfelelő, ugyanis amint a szenzor elfordul valamilyen mértékben a Föld irányába, rögtön elkezd mérni a gravitációs gyorsulásnak az X tengelyre eső komponensét, aminek a nagysága függ a szenzor pozíciójától. Ez egy eléggé gyakori hibát okozott a jelzéseknél. Ennek az oka az, hogy a szenzorban levő irányok változnak a szenzor orientációjától függően. Ebből következik, hogy nem csak akkor mérhetek egy fékezésnek megfelelő mértékű gyorsulást az X tengelyen, hogyha ténylegesen valamilyen gyorsulás történik, hanem akkor is, hogyha a szenzor orientációja miatt az X tengely már nem merőleges a gravitációs vonzásra.



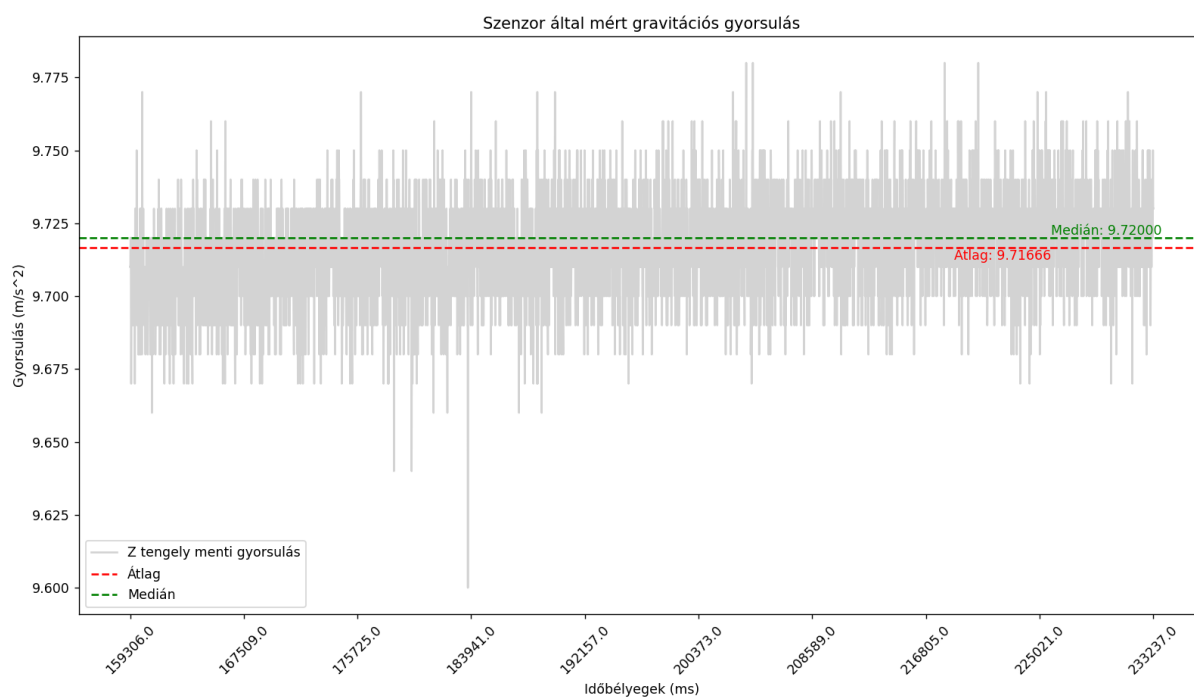
15. ábra X és Z tengelyek felcserélése

A fenti ábrán (15. ábra) megfigyelhető a folyamat, ami során felcserélődik az X és a Z tengely a pitch szög menti forgatásnak köszönhetően.

Ennek a hibának a kiküszöbölésére igyekeztem megoldást találni. Egy hasonló témát feldolgozó könyvben [17] találtam a gravitációs gyorsulásra egy kiküszöbölési módszert, ami sokkal jobb, mint a korábbi, viszont ez sem tökéletes. A könyv szerint a következő képlettel lehet kiszámolni egy adott irány lineáris gyorsulását:

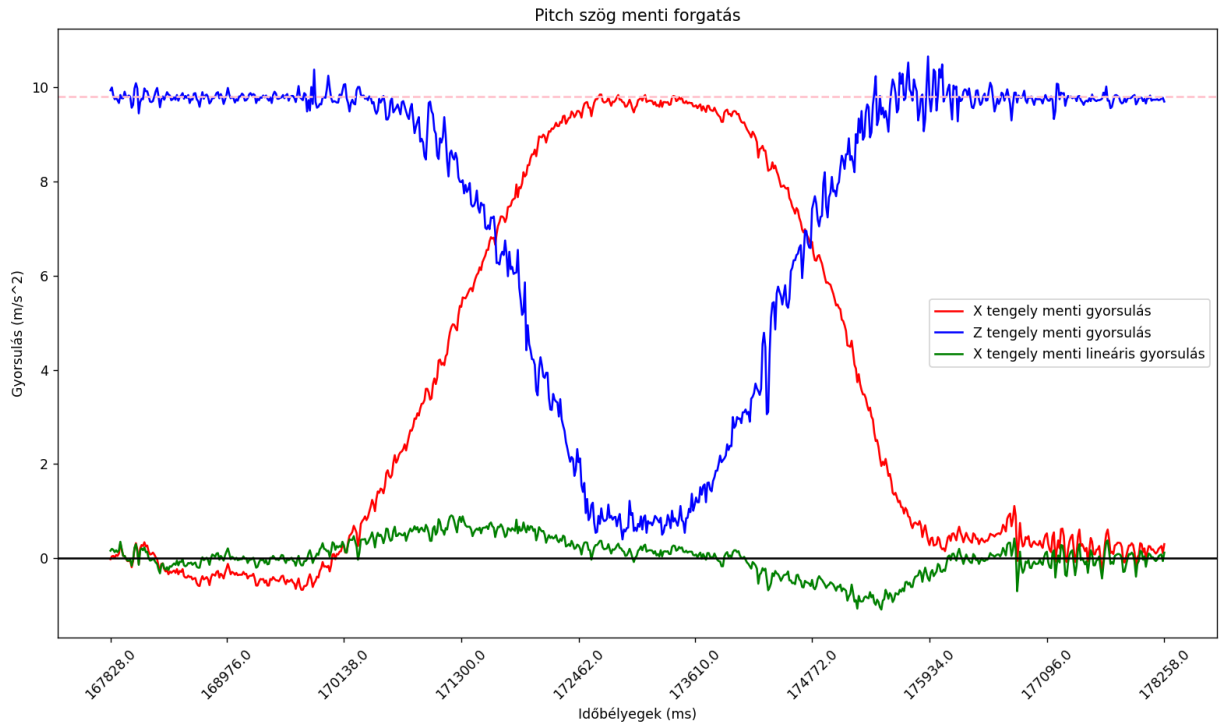
$$LinAcc_X = Acc_X - G_{total} * \cos\left(\frac{\pi}{2} + \theta_X\right)$$

A képletben az  $Acc_X$  a mért gyorsulás az X tengely mentén, a  $G_{total}$  a szenzor által mért gravitációs gyorsulás, és  $\theta_X$  az X tengellyel bezárt szög, vagyis a pitch. A  $G_{total}$  kiszámításához vízszintes felületre helyeztem a szenzort és több ezer mérés alapján (pontosabban 5688) kiszámoltam az értékek átlagát és a mediánt.



16. ábra Mért gravitációs gyorsulás

A fenti ábrán (16. ábra) láthatóak a mérési eredmények, valamint hogy a medián is és az átlag is kerekítve  $9,72 \text{ m/s}^2$ , így ezzel a gyakorlati értékkel végeztem a számításokat. Ezzel a módszerrel már a szenzor nem jelez akkora mértékű gyorsulást, hogyha elkezdem forgatni a korábbi példa szerint.



17. ábra Számított lineáris gyorsulás szemléltetése

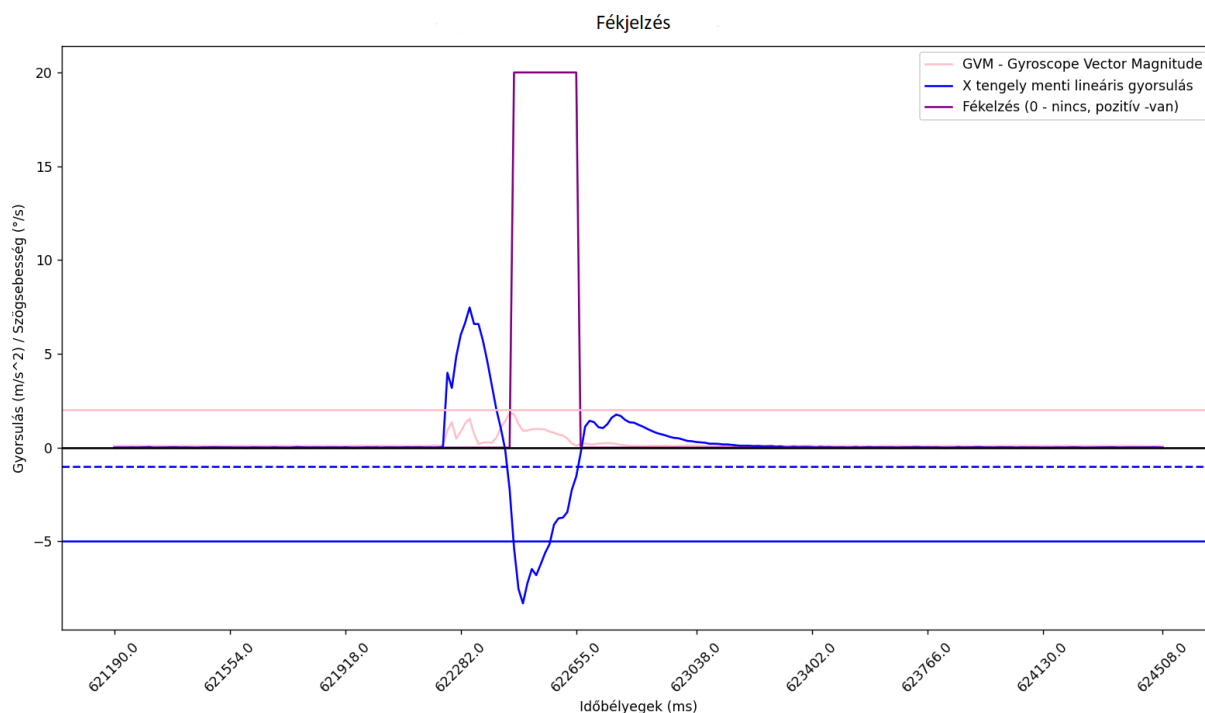
Az ábrán (17. ábra) látható, hogy ezzel a számított lineáris gyorsulással számolva kiküszöböltük a pitch szög menti forgatásból származó hibákat.

További szűréseket is használtam a fékezés jelzéséhez, ilyen érték a GVM vagyis Gyroscope Vector Magnitude [18] ami a giroszkóp által mért 3 értékből képzett vektornak a mérete és a következő képlettel számoljuk ki:

$$GVM = \sqrt{G_X^2 + G_Y^2 + G_Z^2}$$

Ez az érték megmutatja a giroszkóp által mért szögsebességek mértékét függetlenül az irányoktól. Ez azért hasznos, mert így a forgásokból eredő gyorsulásokat is ki lehet szűrni.

A fékezés tényleges jelzéséhez a feltételek közé tartozik az, hogy a GVM értéke alacsony legyen, így ki lehet küszöbölni az olyan hamis jelzéseket, amik mondjuk a rázás vagy összevissza mozgatásból fakadnak. Továbbá az X tengely menti lineáris gyorsuláshoz használt 2 küszöbértéket is figyeli a rendszer a fékezésnél. Az egyik egy alacsonyabb negatív gyorsulási érték, amit el kell érjen ahhoz, hogy elkezdjen a rendszer fékezést jelezni, valamint egy második küszöb, ami még mindig egy negatív érték, viszont már nem akkora abszolútértékű felel azért, hogyha az az érték fölé esik a gyorsulás, akkor kikapcsolja a jelzést.



18. ábra Fékjelzés jelzése

Fontos megemlíteni, hogy ezek a küszöbértékek a szemléltetést szolgálják és a gyakorlati használathoz ezeket újra felül kell vizsgálni.

A fenti feltételek függvényében a rendszer egy változóban folyamatosan tárolja az aktuális fékjelzést egy erre vonatkozó szöveggént, amit majd továbbítani tud. Létezik egy másik változó is, ami mindig az előző értéket tárolja a fékezésnek, így a kettőt összehasonlítva itt is csak akkor küldünk tovább jelzést, ha az érték változik, így megkímélve a kommunikációs csatornát attól, hogy a túl sok adat miatt torlódás következzen be, viszont mégis valós időben tudjuk a változásokat közvetíteni.

#### 4.1.5. Összegzés

Az ESP32 mikrovezérlő szoftvere rengeteg kulcsfontosságú feladatot lát el, a különböző komponensek inicializálásától kezdve egészen a szögek számításáig és a jelzések vizsgálatáig egyaránt. Fontos feladata az is, hogy fogadja a websocket-en érkező adatokat és feldolgozza azokat. Mivel a felhasználói felület tartalmaz egy jelzés ki- és bekapcsoló gombot, ami generál egy jelzést a szerveren keresztül, amit ez a mikrovezérlő is megkap, így ezt is le kell kezelje. Egy változó tárolja, hogy aktívak a jelzések vagy sem és csak akkor végzi el a fentebb felsorolt

funkcionalitásokat, ha az engedélyezve van. Ez a változó pedig aszerint változtatja az értékét, hogy websocket-en milyen üzenetet kapott.

## **4.2. A webszerver futtató mikrovezérlő szoftvere**

A második mikrovezérlő egy ESP8266 és a hozzá tartozó rész feladata a webszerver futtatása és a beérkező adatoknak megfelelő led-ek megvilágítása.

### **4.2.1. Inicializálás**

A mikrovezérlő első feladata a setup függvényben levő inicializálási műveletek megvalósítása, vagyis beállítani a led-eknek megfelelő kimeneti pin-eket, valamint inicializálni a webszervert.

A led-ek megvilágításához szükséges kimeneti pin-eket a következő módon állítjuk be:

```
pinMode(led_pin_left, OUTPUT);  
pinMode(led_pin_right, OUTPUT);  
pinMode(led_pin_stop, OUTPUT);
```

### **4.2.2. Webszerver**

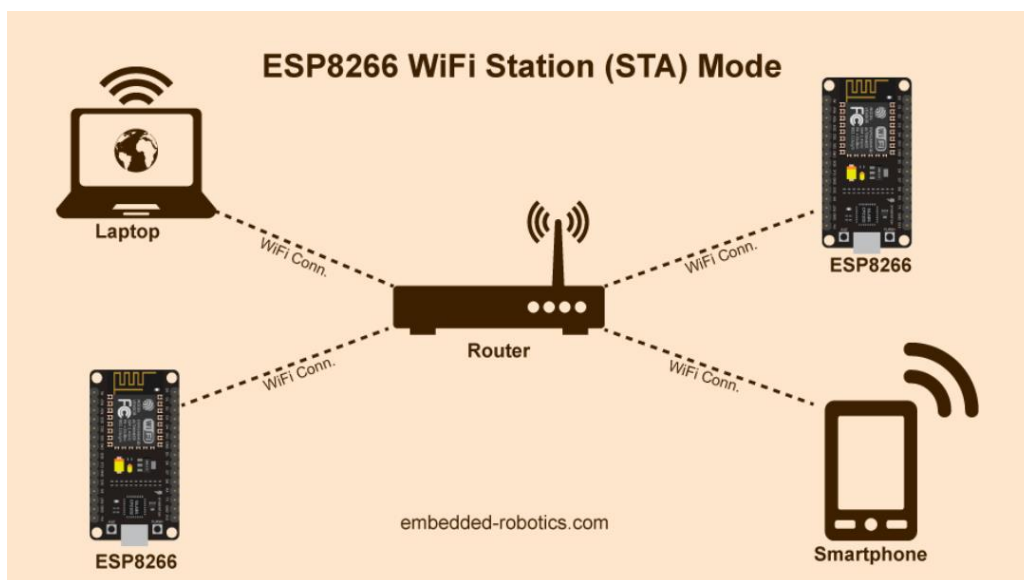
A webszervert felel az eszközök közötti vezeték nélküli kommunikációért. Az első mikrovezérlő egység az olvasott és feldolgozott szenzor adatokat a webszervernek küldi el, ami a kapott adatokat továbbítja a weboldalra csatlakozott eszközöknek (böngészők). Tehát egy kommunikációs csatornaként szolgál a webszerver az eszközök között.

A webszerver tulajdonképpen egy HTTP protokollt használó szerver, ami kiegészül websocket-el is. A HTTP biztosítja a statikus elemeit az oldalnak, mint a HTML, CSS és JavaScript kód, valamint a képek betöltése. A websocket-re azért van szükség, mert ennek a technológiának a segítségével anélkül is tudunk adatokat küldeni és fogadni, hogy kérést kellene intéznünk a szervernek, így létre lehet hozni egy olyan kétirányú kommunikációt a kliens és a szerver között, ami valós időben képes szolgáltatni az adatokat. Így ezzel a módszerrel amint megtörténik a jelzés, szinte egyből, minimális késéssel meg is jelenik a weboldalon a megfelelő információ, valamint a led-ek is ezzel egyidőben kigyúlnak.

A webszerver létrehozásához szükség van a mikrovezérlő WiFi moduljára. Egy mikrovezérlő WiFi modulját két féle üzemmódban tudjuk beprogramozni, hogy egy ilyen vagy ehhez hasonló hálózatot tudjunk kiépíteni.

#### 4.2.2.1 Station (STA) Mode:

Az első mód a Station (STA) Mode. Ezzel a móddal beprogramozott mikrovezérlő úgy fog viselkedni, mint bármilyen más WiFi modullal rendelkező eszköz, mint egy okostelefon vagy egy laptop. Ennek a módnak a segítségével a mikrovezérlő képes rácsatlakozni egy, már létező WiFi hálózatra.



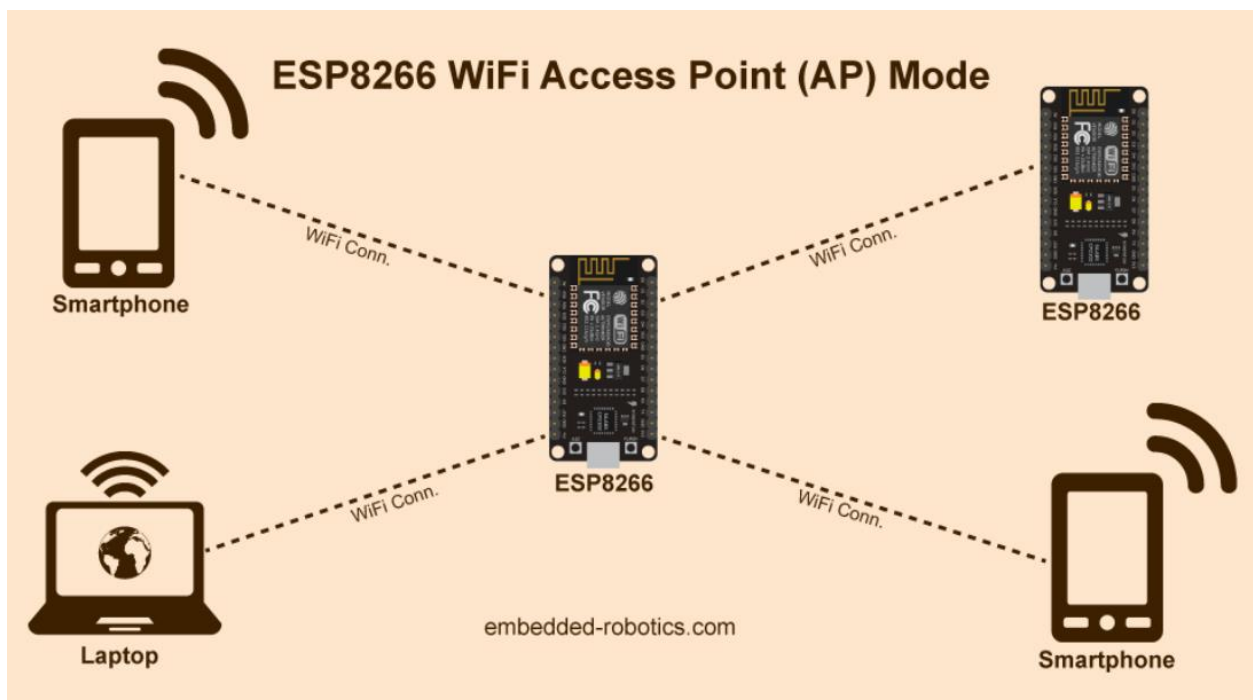
19. ábra WiFi Station (STA) Mode [17]

Az STA mód használatához mindössze arra van szükség, hogy a mikrovezérlő kódjában, a `setup` függvényen belül létrehozzuk a kapcsolatot a WiFi hálózattal, a következő sor segítségével: `WiFi.begin(ssid,password);`, ahol az `ssid` a WiFi hálózat nevét jelenti, a `password` paraméter pedig a hálózat jelszavát.

#### 4.2.2.2 Access Point (AP) Mode:

A másik mód az Access Point (AP) Mode. Ennek a módnak a segítségével a mikrovezérlő WiFi modulja létrehoz egy saját WiFi hálózatot a megadott névvel (`ssid`-val) és jelszóval. Így ha egy eszköz el szeretné érni a mikrovezérlő webszerverét, akkor rá kell, hogy csatlakozzon a mikrovezérlő WiFi hálózatára. Ehhez mindössze azt kell tenni, hogy a `setup` függvényben létrehozzuk a WiFi hálózatot a következő sorral: `WiFi.softAP(ssid,password);`. Ugyanilyen Access Point-ként szolgálnak a mindennapokban is használt WiFi router-ek is, valamint az okostelefonokon használt hotspot-ok is.





20. ábra WiFi Access Point (AP) Mode [17]

Ezt követően a szerver beállítása ugyanúgy történik, mint az előző módban.

#### 4.2.2.3 Szerver létrehozása:

Amennyiben eldöntöttük, hogy a mikrovezérlő WiFi modulját milyen módban szeretnénk használni, be kell konfigurálnunk a szerverünket. Ehhez szükségünk van egy *server* változóra, amit inicializálunk valamelyik port-on. Erre többféle változat is van, ugyanis rengeteg könyvtár áll rendelkezésünkre. Néhány példa erre:

- `#include <WiFi.h>` és `WiFiServer server(80);`
- `#include <WebServer.h>` és `WebServer server(80);`
- `#include <ESPAsyncWebServer.h>` és `AsyncWebServer server(80);`

Ezek az opciók nagyrészt megegyeznek egymással, a különbség talán az, hogy van, amelyik könyvtárcsomag több funkcionalitást kínál, mint a másik. Meg kell találni az adott projektnek legmegfelelőbb opciót. Én az utolsót használtam a projekt során, ugyanis az általam kapott információk alapján az `AsyncWebServer` kínálja a legtöbb funkcionalitást és a használata is egyszerű. Ezt követően a `server.begin();` sorral el is indíthatjuk a webszervert, amit azok az eszközök érnek el, akik a szerverrel azonos WiFi hálózatra vannak csatlakozva.

#### 4.2.2.4 Összegzés:

Ez a felsorolt két mód létezik, amivel a mikrovezérlőnk WiFi modulját bekonfigurálhatjuk. A két mód közötti különbség az általános használat szempontjából elég szembetűnő, ugyanis míg

a STA mód esetén elég egy hálózaton lenni a mikrovezérlővel, hogy elérjük a webszervert és ez által kommunikálni tudunk a WiFi hálózaton keresztül, addig az AP módban rá kell csatlakozni a mikrovezérlő által létrehozott hálózatra.

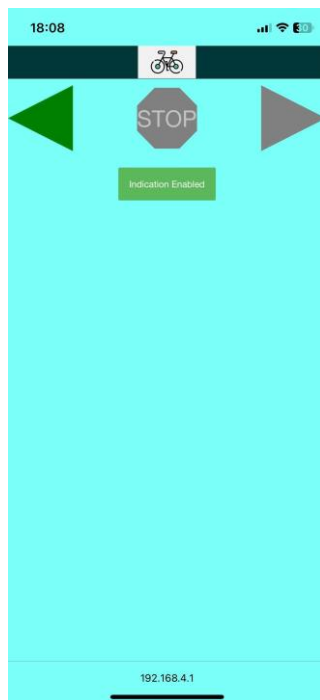
A projekt tervezése során kipróbáltam mindkét lehetséges opciót. Kezdetben a szerver tesztelése során STA módot használtam, ugyanis így könnyebb volt elérni a szervert, mert nem kellett folyamatosan lecsatlakoztassam valamelyik okoseszközöm az általam használt WiFi hálózatról, hogy a mikrovezérlő hálózatára csatlakozzak és vissza. De mivel ez a projekt megköveteli a mobilitást, ugyanis egy jármű jelzésrendszerét valósítja meg, így ez a mód nem kézenfekvő, mert ahhoz az szükségeltetik, hogy a jármű bármerre is van, mindenhol legyen egy elérhető és ismert WiFi hálózat. Erre talán megoldást jelenthetne az, hogy mivel az okostelefonunk mindig nálunk van, így annak a hotspot-ját használja a rendszer a kommunikációra. Ezzel a módszerrel az a probléma, hogy amennyiben az okostelefon mobilhálózatának a jelerőssége legyengül, esetleg olyan környéken halad át a jármű, ahol nagyon terhelve van a mobilhálózat, vagy pedig egyáltalán nincs is jel, ott a mobil hotspot megszűnik és ezzel a rendszer kommunikációja megszakad. Ezzel a módszerrel már a jelzések sem működnének megfelelően és így a rendszer túl instabil lenne. Ennek kiküszöbölése érdekében döntöttem úgy, hogy a webszervert futtató mikrovezérlő az AP módot használja, így a felhasználónak szánt weboldal elérhető, amennyiben egy okostelefonnal rácsatlakozunk a mikrovezérlő hálózatára.

Az a mikrovezérlő, amelyik a szenzor adatokat olvassa és feldolgozza, STA üzemmódba rácsatlakozik az előző mikrovezérlő által létrehozott hálózatra és azon keresztül kommunikálnak egymással. Ezzel a módszerrel a jármű szinte bárhol is legyen, a rendszer kommunikációja stabil marad. Igaz, hogy így az okos eszköz, mondjuk egy mobiltelefon, amit használunk arra, hogy a rendszer állapotát kövessük a weboldalon, már nem lesz képes kapcsolódni az internetre mobilhálózaton keresztül, de cserébe a kommunikáció a rendszeren belül stabil marad. A tervezés során a fenntartható stabil kommunikáció volt a fő cél, így erre a megoldásra esett a választásom.

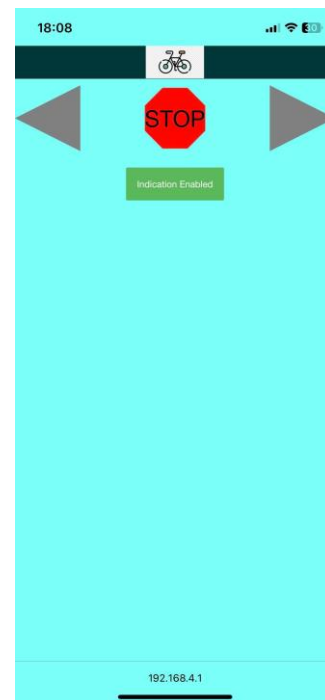
### **4.3. Webalkalmazás**

A webalkalmazás egy egyszerű weboldal, amit a mikrovezérlő által futtatott szerveren lehet elérni, amennyiben egy okoseszköz rácsatlakozik a szervert futtató mikrovezérlő hálózatára és egy böngésző segítségével felkeresi a megfelelő címet. A webalkalmazás célja ismertetni a rendszer állapotát a felhasználóval, valamint megengedni a felhasználónak, hogy szabályozni tudja a rendszert. Ez egy egyszerű felület, ahol felhasználóbarát módon megjelennek az aktuális

jelzések, legyen szó az irányváltásra vonatkozó jelzésekről, amiket egy-egy nyíl jelez (21. ábra), valamint a fékezésre vonatkozó jelzések, amiket egy stop tábla jelez (22. ábra).



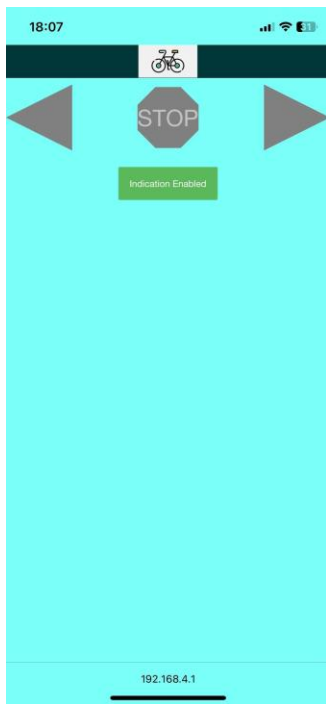
21. ábra Kanyarjelzés



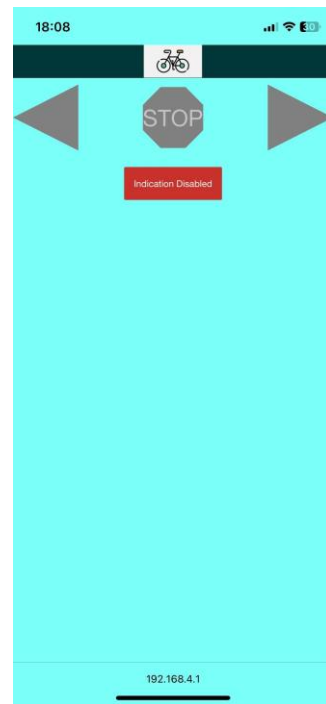
22. ábra Fékjelzés

Ezeknek a segítségével folyamatosan meg lehet figyelni az aktuális jelzéseket vizuálisan is.

Emellett a webalkalmazás interakciót is biztosít a felhasználó számára. Erre szolgál egy gomb, aminek a megnyomásával ki- (24. ábra) és be (23. ábra) lehet kapcsolni a rendszer működését. Így ha netán nincs szükség jelzésekre, akkor innen le lehet tiltani őket, valamint újra bekapcsolni, ha megint szükség lenne rájuk.



23. ábra Jelzés bekapcsolva

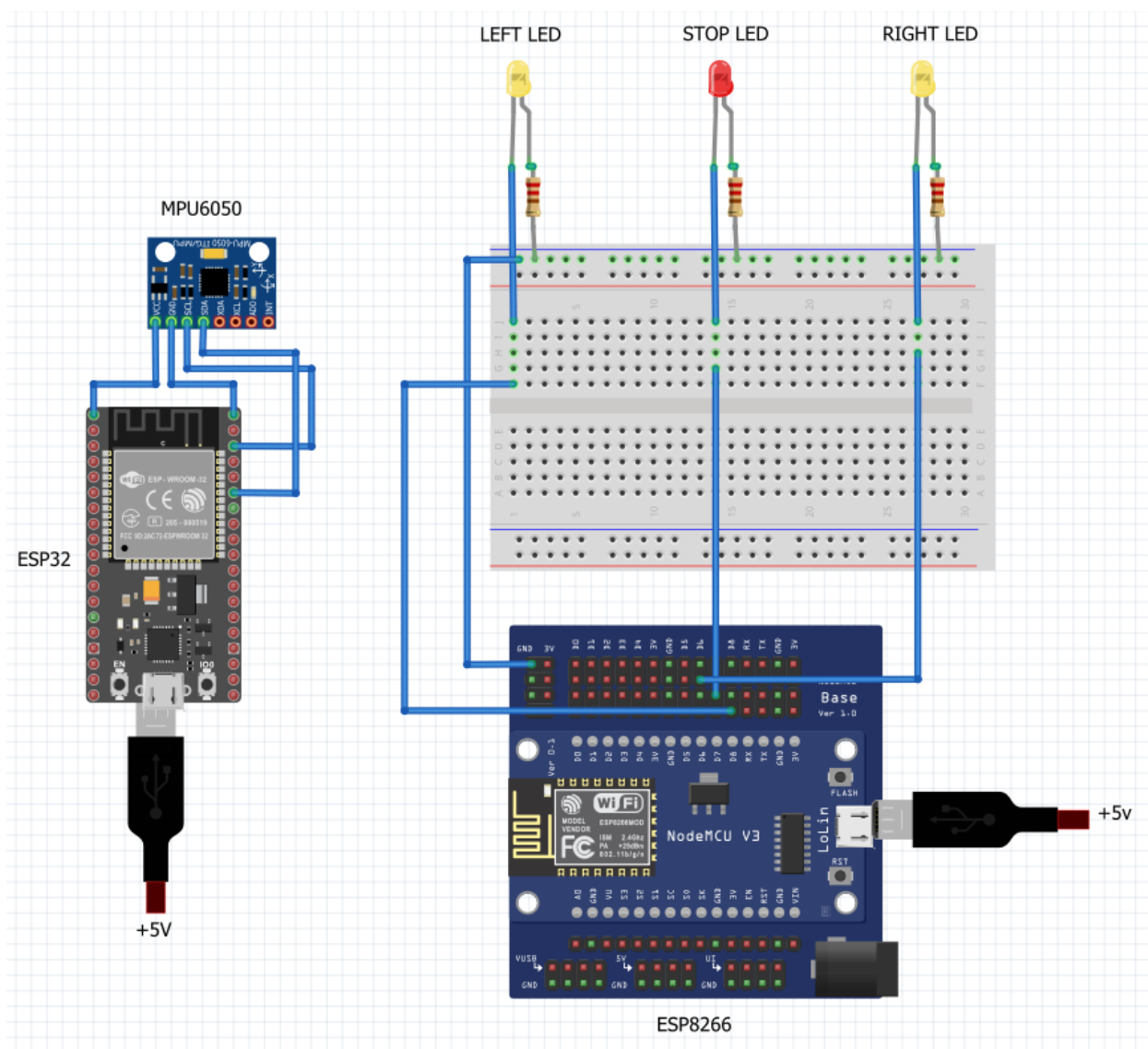


24. ábra Jelzés kikapcsolva

A weboldal JavaScript kódjában van lekezelve a WebSocket kommunikáció, tehát az üzenetek fogadása, valamint adott esetben adatok küldése a szervernek. A JavaScript kódban vannak lekezelve az üzenetekre vonatkozó válaszreakciók is, vagyis ha fék- vagy kanyarjelzések történnek, akkor a megfelelő elemeknek a színét változtassa meg.

#### 4.4. Prototípus bemutatása

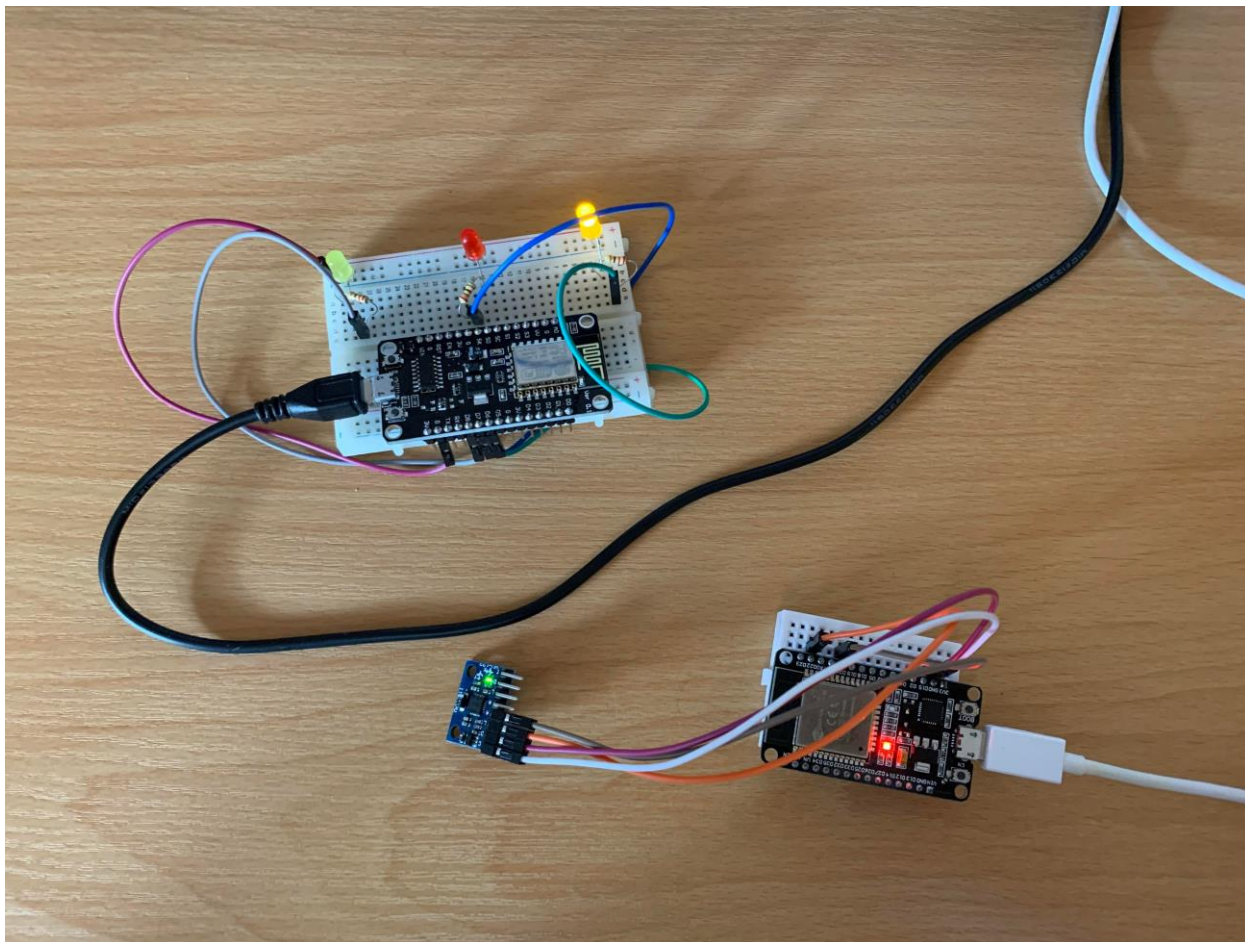
A projekt gyakorlati megvalósításához fontos egy prototípus létrehozása. A prototípus tartalmaz minden olyan alkatrészt, amely nélkülözhetetlen a végtermék működése esetén, viszont gyakorlatban nem használható, mint egy végleges termék. Célja a rendszer funkcionalitásainak a letesztelése, az esetleges hibák kiküszöbölése és hogy bemutassa a tervezett rendszer működését.



25. ábra Prototípus ábra

A fenti ábrán látható a két mikrovezérlő, az ESP32 amelyik az MPU6050 szenzorhoz van kötve, ez szolgáltatja az adatokat, valamint látható az ESP8266 ami a LED-ek megvilágításáért felel,

valamint a webszerver futtatásáért, ami biztosítja a rendszer számára a kommunikációt.



*26. ábra Prototípus a valóságban*

## 5. Továbbfejlesztési lehetőségek

### 5.1. Hardver fejlesztése

A hardver fejlesztésének az egyik lehetősége a meglévő prototípus továbbfejlesztése lenne, egy gyakorlatban is jól használható eszközzé. Ehhez szükséges lenne valamilyen tároló eszköz, hogy a kézen helyet kapó MPU6050 szenzort, a hozzá tartozó ESP32 mikrovezérlőt, valamint egy kisebb méretű akkumulátort kompakt formában lehessen tárolni és védeni a környezeti behatásoktól. Az ESP8266 mikrovezérlőt szintén egy áramforrással kellene tárolni, valamint nagyobb led-eklet kötni hozzá a jobb láthatóság miatt.

Egy másik fejlesztési lehetőség lenne a rendszert kiegészíteni más szenzorokkal is, mint sebességmérő szenzorral, vagy egy hőmérséklet érzékelő szenzorral, ami méri a mikrovezérlő, a szenzorok és az akkumulátor hőmérsékletét az esetleges túlhevülés elkerülése végett.

## 5.2. Szoftver fejlesztése

A projekt szoftverének rengeteg fejlesztési lehetősége van, ugyanis bármikor ki lehet egészíteni plusz funkciókkal, valamint a kinézetet is lehet fejleszteni a felhasználói felületnél. A hardver fejlesztése, amennyiben plusz eszközöket ad hozzá a rendszerhez, magával vonja a szoftver fejlesztését is, hogy biztosítva legyenek az új funkciók.

Egy további fejlesztési lehetőség lenne egy komplexebb szűrő használata, ugyanis a Complimentary filter fix értékeket használ súlyként, így idővel az értékek eltolódhatnak és kissé pontatlanná tehetik a méréseket. Erre megoldást jelenthet a Kalman filter használata. A Kalman filter egy komplexebb szűrő, ami dinamikusan számolja ki a súlyokat és ezzel egy pontosabb eredményt biztosít, mintha fix értékű súlyokkal dolgoznánk.

## 6. Hivatkozások

- [1] „LightVest,” [Online]. Available: <https://smartbuilds.io/lightvest-arduino-led-bike-vest-tutorial/>. [Hozzáférés dátuma: 2023].
- [2] „Internet of things,” 2023. [Online]. Available: [https://en.wikipedia.org/wiki/Internet\\_of\\_things](https://en.wikipedia.org/wiki/Internet_of_things).
- [3] „Do it yourself,” 2023. [Online]. Available: [https://en.wikipedia.org/wiki/Do\\_it\\_yourself](https://en.wikipedia.org/wiki/Do_it_yourself).
- [4] „Microcontroller,” 2023. [Online]. Available: <https://en.wikipedia.org/wiki/Microcontroller>.
- [5] L.-Z. Túrós és G. Székely, Érzékelők és mérőhálózatok, Kolozsvár: Scientia Kiadó, 2022.
- [6] T. Long, Data Fusion with 9 Degrees of Freedom Inertial Measurement Unit To, San Luis Obispo, 2017.
- [7] „Web server,” 2023. [Online]. Available: [https://en.wikipedia.org/wiki/Web\\_server](https://en.wikipedia.org/wiki/Web_server).
- [8] „ESP8266,” [Online]. Available: <https://datasheetspdf.com/pdf-file/994356/EspressifSystems/ESP8266EX/1>. [Hozzáférés dátuma: 2023].
- [9] „ESP32 Series Datasheet,” [Online]. Available: [https://www.espressif.com/sites/default/files/documentation/esp32\\_datasheet\\_en.pdf](https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf). [Hozzáférés dátuma: 2023].
- [10] „ESP-Wroom-32,” [Online]. Available: <https://pdf1.alldatasheet.com/datasheet-pdf/view/1179101/ESPRESSIF/ESP-WROOM-32.html>. [Hozzáférés dátuma: 2023].
- [11] I. Inc., „MPU-6000 and MPU-6050 Product Specification Revision 3.4,” 2013.
- [12] „Light-emitting diode,” [Online]. Available: [https://en.wikipedia.org/wiki/Light-emitting\\_diode](https://en.wikipedia.org/wiki/Light-emitting_diode). [Hozzáférés dátuma: 2023].
- [13] „I2C,” [Online]. Available: <https://en.wikipedia.org/wiki/I%C2%B2C>. [Hozzáférés dátuma: 2023].
- [14] „HTTP,” Mozilla, [Online]. Available: <https://developer.mozilla.org/en-US/docs/Web/HTTP>. [Hozzáférés dátuma: 2023].
- [15] „WebSocket,” [Online]. Available: <https://en.wikipedia.org/wiki/WebSocket>. [Hozzáférés dátuma: 2023].



- [16] „readthedocs.org,” [Online]. Available: <https://arduino-esp8266.readthedocs.io/en/latest/filesystem.html#flash-layout>. [Hozzáférés dátuma: 2023].
- [17] S. Ubejd és R. Angel, Indoor Positioning using Sensor-fusion in Android Devices, 2011.
- [18] C. Howard, S. Mark és F. Nathan, Gyroscope Vector Magnitude: A proposed measure for accurately measuring angular velocities, 2022.
- [19] „Embedded-Robotics,” [Online]. Available: <https://www.embedded-robotics.com/esp8266-wifi/>. [Hozzáférés dátuma: 2023].

UNIVERSITATEA SAPIENTIA DIN CLUJ-NAPOCA  
FACULTATEA DE ȘTIINȚE TEHNICE ȘI UMANISTE, TÎRGU-MUREȘ  
SPECIALIZAREA CALCULATOARE

Vizat decan  
Conf. dr. ing. Domokos József

Vizat director departament  
Ș.l. dr. ing. Szabó László Zsolt