Járműkövetés 3D virtuális környezetben

Vehicle tracking in 3D virtual environment

HORVÁTH Botond[1]

[1]Mechatronika, Optika és Gépészeti nformatikaTanszék, Budapesti Műszaki és Gazdaságtudományi Egyetem, Gépészmérnöki Kar, 1111 Budapest, Műegyetem rakpart 3.
Email: horvath.botond@mogi.bme.hu

Abstract

Technology is developing rapidly, the presence of cyber-physical systems is becoming more and more significant, and the role of virtual reality is increasing greatly in the daily lives of both industrial and private users. Simulations, digital shadows, and digital twins help us in many areas of life, be it planning, commissioning, testing, optimization, or even video games. The topic of this project is the implementation of vehicle tracking in real time using the three-dimensional software called MaxWhere. This requires retrieving and transmitting location data with a GNSS receiver, refining them using a dynamic filtering algorithm, and displaying the vehicle in virtual space based on the received position. Such an application can be a great help in industry, especially in the case of tracking vehicles moving in factories, warehouses or sites, because with this technology even all tracked objects within a well-defined area can be displayed in a single virtual space. Roaming in the space, the operator can get anywhere in seconds and observe what he needs, he can even intervene in the various processes through the interface, thus saving time and energy.

Keywords: : GNSS, digital twin, Kalman-filter, mechatronics, industry 4.0, IoT

Kivonat

A technológia rohamosan fejlődik, a kiberfizikai rendszerek jelenléte egyre jelentősebb, a virtuális valóság szerepe nagy mértékben növekszik mind az ipari, mind a magán felhasználók mindennapjaiban. A szimulációk, digitális árnyékok, valamint digitális ikrek az élet rengeteg területén jelentenek segítséget számunkra, legyen szó tervezésről, beüzemelésről, tesztelésről, optimalizálásról, vagy akár videójátékokról. Ezen projekt témája járműkövetés megvalósítása a MaxWhere nevű háromdimenziós szoftver segítségével valós időben. Ehhez szükséges a helyadatok lekérése és továbbítása GNSS vevővel, azok finomítása dinamikus szűrési algoritmus felhasználásával, valamint a jármű megjelenítése a virtuális térben a kapott pozíció alapján. Egy ilyen alkalmazás nagy segítség lehet az iparban, kiváltképp a gyárakban, raktárakban vagy telephelyeken közlekedő járművek követése esetén, ugyanis ezzel a technológiával egy jól behatárolt területen belül akár minden követett objektum egyetlen virtuális térben megjeleníthető. Az operátor a térben barangolva pillanatok alatt bárhová eljuthat, és figyelheti meg azt, amire éppen szüksége van, a felületen keresztül akár be is avatkozhat a különböző folyamatokba, ezzel időt és energiát megtakarítva.

Kulcsszavak: GNSS, digitális iker, Kálmán-szűrő, mechatronika, ipar 4.0, IoT

1. BEVEZETÉS

A fejlesztés célja mozgó jármű digitális árnyékának pályakövetése virtuális térben, ezáltal az alkalmazás alapjául szolgálhat egy - a jövőben szabadon fejleszthető - virtuális objektumkövető platformnak, melyben tetszőleges ipari környezet monitorozása valósítható meg. A valós járműbe helyezett GNSS helymeghatározó eszköz szolgáltatja a követéshez szükséges adatokat, melyek MQTT szerveren keresztül kerülnek továbbításra az általam már egyéb szimulációk elkészítésére használt, de elsősorban interaktív, térbeli munkakörnyezetek és adatelemzések létrehozására kifejlesztett MaxWhere nevű háromdimenziós programnak, ami valós időben jeleníti meg a térképen a vizsgált jármű állapotát. A pontosabb helyzet meghatározásának érdekében célszerű dinamikus szűrési eljárást is használni.

2. DINAMIKUS SZŰRÉSI ELJÁRÁSOK

Kálmán Rudolf Emil magyar villamosmérnök, matematikus, feltaláló, a modern rendszerelmélet kiemelkedő alakja volt. Munkájával számos hazai és nemzetközi elismerést is elnyert, az ő nevéhez fűződik többek között a Kálmán-szűrő elvének kidolgozása [1]. A Kálmán-szűrő alapú algoritmusok zajjal terhelt lineáris dinamikai rendszerek állapotára adnak optimális becslést mérések, illetve az előző becsült állapot figyelembevételével, ennek eredményeképp sokkal pontosabb információt kaphatunk a vizsgált rendszer állapotáról, mintha csak a mérési eredményt vennénk figyelembe. A Kálmán-szűrő alapú algoritmusok a bemenő adatok sorozatos mérésével zajjal terhelt lineáris dinamikai rendszerek állapotára adnak optimális becslést, mely rendszerint pontosabb, mintha csak a mérést végeztük volna el. Ez rekurzív módon működik, vagyis elegendő mindig csak az utolsó eredményt figyelembe venni, nincs szükség memóriajellegre. A becsült adatokat átlagolja a szűrő, majd az új méréseket is figyelembe véve súlyozott átlagolást végez a kovariancia alapján, mely a rendszer állapotainak becsült bizonytalanságából adódik ki. A súlyozott átlag eredményeként megkaphatjuk a következő állapotot [2].

A mai napig előszeretettel alkalmazzák a Kálmán-szűrőt és kiterjesztéseit navigációs rendszereknél, irányításvezérléseknél (repülőgépek, űrhajók, robotrepülőgépek), légi járművek és rakéták radar követésénél, tengeralattjárók és földi járművek akusztikai követésénél és jelfeldolgozó rendszereknél. Más területeken is találkozhatunk azonban vele, hiszen a szűrő bármely rendszerre alkalmazható, ami folytonos állapotváltozókkal és zajos megfigyelésekkel leírható. Ily módon fellelhető nukleáris reaktorok vagy vegyipari üzemek irányításánál, növényi ökoszisztémák vizsgálata során, vagy akár az ökonometria területén is [3].

3. GNSS ADATOK

3.1. Weboldal

Ahhoz, hogy a jármű mozgását le lehessen írni, szükséges volt bizonyos adatok mérése. Kézenfekvőnek tűnt egy egyszerű okostelefon használata erre a célra, hiszen ez könnyen hozzáférhető, és van benne helymeghatározás, egy weboldal futtatásának segítségével lekérhető a pozíció.

A webfejlesztés HTML nyelven történt, a helyadatok lekérdezésében a Geolocation API interfész nyújtott segítséget, mely IP cím, MAC cím, RFID és GPS koordináta alapján küld pozíciót a weboldalra [4].

Meg kellett még oldani az adatok továbbítását, ezt MQTT szerveren keresztül tettem meg. Mivel a Geolocation API csak https protokoll alatt fut, a https protokoll pedig csak a WebSocket Secure kapcsolaton keresztül tud kommunikálni, így a hagyományos mqtt szerver helyett ws szervert kellett igénybe vennem. Az ily módon publikált adatokat a szimulációs programomban egyszerűen le tudtam kérni (1. ábra).

Annak érdekében, hogy bármelyik okostelefonon meg lehessen nyitni a weboldalt, hosztolásra volt szükség. Erre a Firebase platformját használtam, így szabadon elérhetővé vált a weboldalam az interneten [5]. Megnyitás után nincs más teendő, mint az okostelefonunkon bekapcsolni a helyzetmeghatározást, majd engedélyezni azt a böngészőben. Amint lekérdezésre került az első geokoordináta pár, az kiíratásra kerül, és az oldal már küldi is a szerverre az adott hosszúsági- és szélességi fokot json objektum formájában, így ezeket majd a programban egyszerűen fel lehet használni.

1. ábra. *Helyadatok kinyerése okostelefonnal*

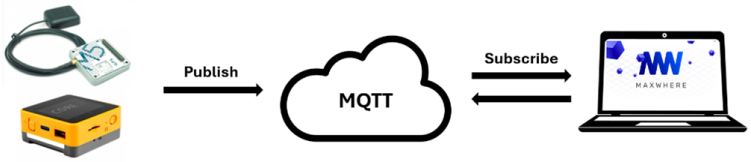
Noha ez a módszer is megoldotta a koordináták lekérését és továbbítását, és kezdetben ennek segítségével valósítottam meg a szimulációt a térben, a végső verzióban mégsem az adatok ilyen módon történő megszerzését alkalmaztam, mivel ezzel csak a hosszúsági- és szélességi fokot lehet mérni, sebességet és irányt azonban nem. A weblap mellett szól viszont az a tény, hogy épületen belül is sikeresen le tudja kérni a telefon helyzetét, még ha nem is olyan frekvenciával, mint szabad téren, de sok esetben ez előnyt jelenthet, például csarnokokban vagy raktárakban.

3.2. Mikrovezérlő

Mivel a fejlesztés során felvetődtek más lehetőségek is, célszerűnek tűnt egy olyan megoldás, mely nemcsak a pozícióra, hanem a mozgás irányára és aktuális sebességére is ad értékeket. A piacon fellelhető mikrovezérlők közül számos példány megfelelhet erre a célra, választásom ez esetben az M5Stack Core2-re

esett, mivel ez ESP32 modullal van ellátva, vagyis képes csatlakozni WiFi hálózatra, így az adatok továbbítása az MQTT szerverre könnyedén megvalósítható, emellett a termékcsalád tartalmaz ezzel kompatibilis GPS modult és hozzá tartozó antennát is [6]. Ez az M5Stack NEO-M8N, mely rendelkezik belső, illetve külső antennával is [7]. Mivel a tesztelés során a belső antenna rosszabb időjárási vagy domborzati körülmények között meglehetősen gyengének bizonyult, a külső antennát használtam a továbbiakban.

A mikrovezérlő programozása Arduino IDE fejlesztőkörnyezetben történt C nyelven. A GPS modul működtetéséhez szükség volt a TinyGPSPlus könyvtár használatára [8]. Ez a könyvtár olyan függvényeket tartalmaz, melyekkel le lehet kérdezni a különböző mért adatokat. Ezeket az adatokat az MQTT szerverre küldtem ki (2. ábra), ehhez pedig a PubSubClient könyvtár teremtette meg a lehetőséget [9]. A publikált értékeket ebben az esetben is egy json objektumban tároltam, ez tartalmazta a sebességet, irányt, hosszúságot és szélességet.



2. ábra. *Helyadatok kinyerése mikrovezérlővel*

4. KOMMUNIKÁCIÓ

A mért adatok továbbítása MQTT szerveren keresztül történt. Az MQTT-t már húsz évvel ezelőtt is használták gép-gép közötti (M2M - Machine to Machine) kommunikációra, és napjainkban is rendkívül elterjedt az IoT eszközök közti kapcsolat megteremtésében, hiszen ez egy nyílt és ingyenes lehetőség a lehető legegyszerűbb formátumú üzenetek kezelésére. Ez a módszer akár kényszerekkel terhelt hálózatok esetében is ideális megoldás lehet. Az MQTT szerveren egy-egy topicra üzenet küldhető, ezekre a topicokra fel lehet iratkozni, így lekérhetőek a beérkező üzenetek egy másik eszköz által. Ezen fejlesztés esetén a pozíció adatokat kezdetben az okostelefon, majd később a mikrokontroller küldte, és a szimulációt futtató számítógép fogadta.

A kliensre való feliratkozás és a szerverről a publikált adatok lekérdezése a főprogramban az MQTT.js könyvtár függvényeinek alkalmazásával történt [10]. A weboldalas és a mikrovezérlős megoldás között itt csak a kliens definiálásánál volt különbség. Ezt követően feliratkoztattam a programot a topicra, és amint új adat érkezett a szerverre, az máris feldolgozásra kerülhetett.

5. KÖRNYEZET KIALAKÍTÁSA

5.1. MaxWhere tér

A virtuális teret, benne a mozgó jármű digitális árnyékával a MaxWhere alkalmazásban valósítottam meg. A MaxWhere Engine az a platform, mely egyesíti a 3D grafikát a webalapú tartalmakkal. A MaxWhere terekben a magasszintű programozás egy JavaScript API-n keresztül (Node.js) történik. Egy alap MaxWhere térre volt csak szükségem, ebbe illesztettem be egy már a cégnél meglévő osm nevű komponenst. Ez egy olyan komponens, melybe tetszőleges térképből generált GeoJSON fájlt betölthetünk, az megjeleníti a síkban.

5.2. Térkép

A pontos járműkövetéshez szükség volt az adott környezet hiteles mására. Ehhez a feladathoz egy térkép megjelenítése jelentette a legjobb megoldást. A térkép kinyerésére az OpenStreetMap alkalmazást használtam [11]. Ez egy egész világra kiterjedő, bárki által szabadon fejleszthető térkép. Az OpenStreetMap felületén szabadon kijelölhető egy téglalap alakú terület, melyet ezután exportálni lehet. Az itt kapott fájl OSM kiterjesztésű, a programunk azonban csak GeoJSON kiterjesztésű térképet tud lekezelni, így ezt szükséges átkonvertálni, erre található webes alkalmazás [12]. Az így létrejött fájlt már beemelhettem a programomba, és meg is jelent a MaxWhere térben a térkép.

6. ADATOK DINAMIKUS SZŰRÉSE

A pontosabb pálya elérése érdekében sokszor ajánlott dinamikus szűrési eljárást használni. Jelen esetben a Kálmán-szűrő megfelelő megoldásnak tűnt, hiszen mindig elegendő az előző állapot ismerete és az új mért adatok beérkezése a következő állapot kiszámításához.

Az értékek finomításához a Kálmán-szűrőt két dimenzióban alkalmaztam, mivel ezen példában síkban történő mozgásról van szó. Ez esetben az *xy* síkban mozgó jármű pozícióját, sebességét, illetve gyorsulását vizsgáltam *x* és *y* irányban. A bemenő adatok a pozíció - melyet a geokoordinátákból számoltam -, a sebesség, és a mozgás iránya voltak, melyeket a mikrovezérlő segítségével mértem. Gyorsulást nem tudtam mérni a rendelkezésemre álló eszközökkel, így annak nagyságára csak becslést adtam.

Minden iteráció végén egy jóslást kellett végezni a következő időpillanatra, mivel itt még nincs birtokunkban a következő mérési eredmény. A mérési jelvektorból, illetve az előző iterációban jósolt állapotvektorból lehet meghatározni az aktuális állapotvektort.

7. JÁRMŰ MOZGÁSA

7.1. Valós idejű megjelenítés

A következő lépés a jármű mozgatása volt az MQTT szerverről érkező adatok feldolgozásával. Mindig akkor történt a következő pozíció és orientáció kiszámítása, amikor új adatcsomag érkezett. Amíg nem alkalmaztam szűrést, ilyenkor egy az egyben a mért geokoordinátákból számoltam a pozíciót, illetve a mért orientációt adtam a járműnek (3. ábra). A Kálmán-szűrő használatával azonban pontosabb pályakövetést tudtam megvalósítani. A program minden egyes iterációban elvégzi a szükséges mátrixműveleteket, ezzel becslést adva a jármű aktuális pozíciójára.

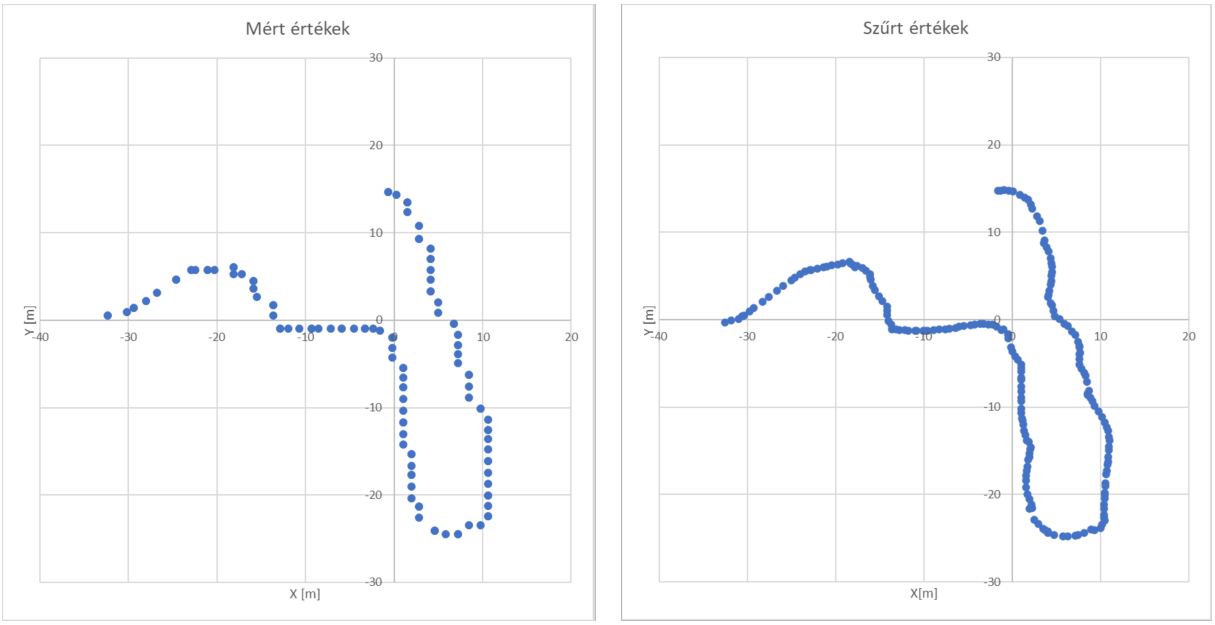
3. ábra. *Jármű megjelenítése virtuális térben*

7.2. Mozgás reprodukálása

Annak érdekében, hogy egy mérés ugyanúgy megvalósulhasson még egyszer, ezáltal megkönnyítse a tesztelési folyamatot, a tér minden egyes futtatása során elmentettem egy Node-RED programmal az MQTT szerverre érkező adatcsomagokat egy text fájlba soronként. Egy ilyen dokumentált mozgást bármikor le lehet játszani a térben, így a szimuláció működésének vizsgálata egyszerűbbé válik. Ehhez egy Node.js alkalmazást írtam JavaScript nyelven, mely beolvassa a megadott fájlt soronként, majd elmenti a teljes adatcsomagot, illetve az előző adatcsomaghoz viszonyított időkésést is egymás után egy-egy tömbbe. Ezt követően a program MQTT szerverre publikálja az adatcsomagokat az annak megfelelő tömbből kinyerve azokat, az egyes sorokhoz tartozó időkésleltetésnek megfelelő ütemben. Az MQTT-ről lekérve a szimuláció a mérésekkel megegyező módon felhasználja az adatokat.

8. EREDMÉNYEK

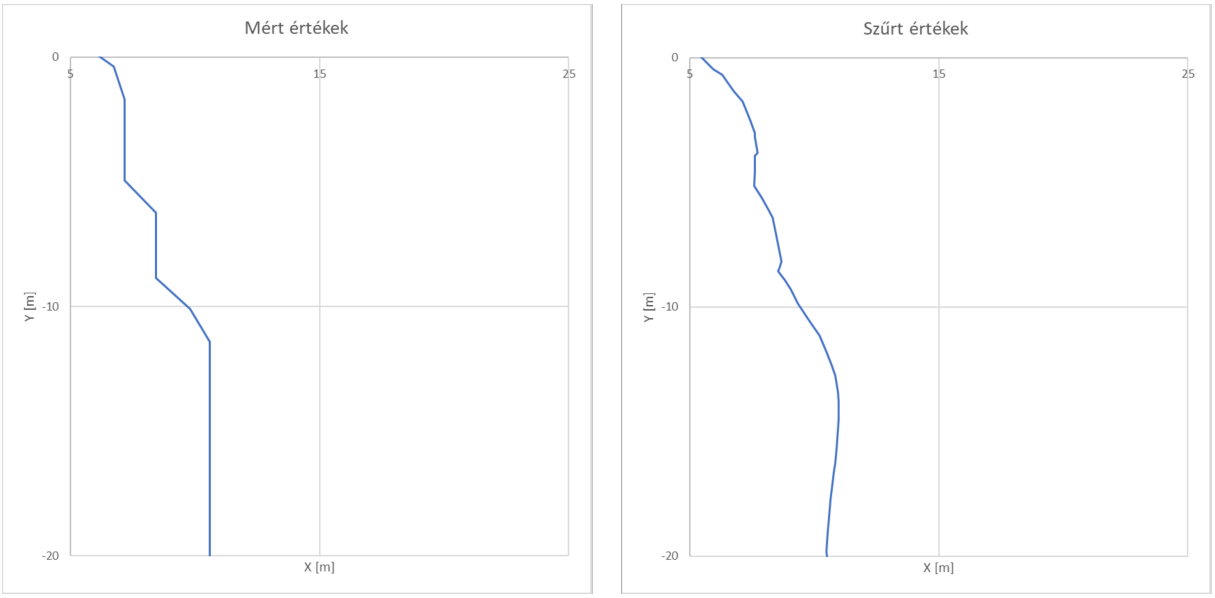
A szűrés lényegének szemléltetése érdekében grafikonokat készítettem a referenciának felvett ponthoz viszonyított pozíció x és y értékeire mind a mért, mind a szűrt adatok esetén. Az adatok felvétele egy parkolóban történő lassú mozgás során készült, pontokkal jelenítettem meg az egyes pozíciókat (4. ábra). Megfigyelhető, hogy szűrt pálya sokal több pontot tartalmaz. Ez abból adódik, hogy vannak esetek, mikor a mikrovezérlő új adatokat küld, de a geokoordináták az előző üzenetben foglaltaktól nem térnek el, így a nyers pozíció változatlan volna, a szűrővel mégis tudunk egy következő értéket becsülni a sebesség figyelembevételével.



4. ábra. *A mért és szűrt értékek ábrázolása pontdiagramon*

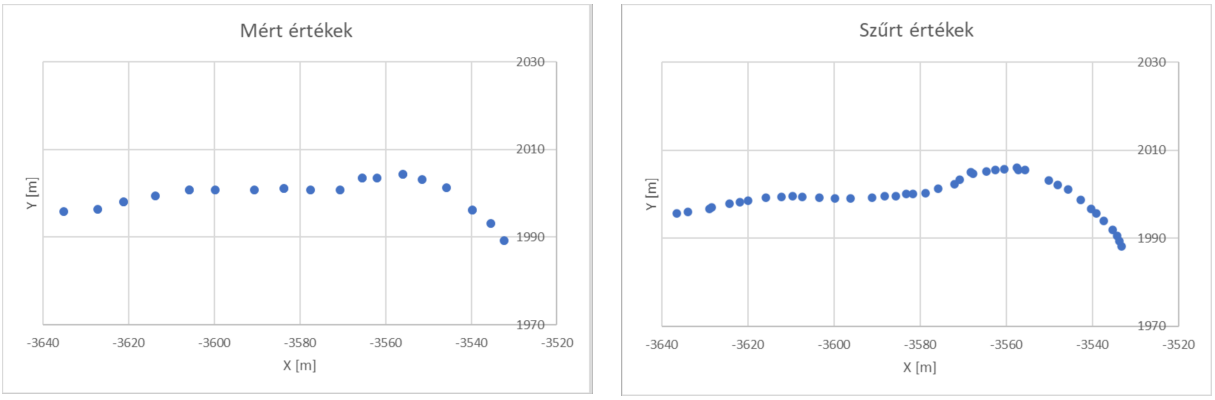
Az egyes pontok közötti lineáris mozgást feltételező modellnek megfelelő vonaldiagramot készítettem a mozgás egy leszűkített intervallumán, így jobban megfigyelhető a trajektória finomodása (5. ábra).

Mint a mellékelt ábra mutatja, a nyers pozíciókhoz képest a szűrt adatok sokkal simább mozgáspályát mutatnak, mivel egy-egy pontatlanabb mérés nem befolyásolja annyira az értékeket. Fontos továbbá, hogy a szűrt pálya sok esetben folytonosabb is, mivel a számításoknak köszönhetően több pontból tevődik össze. Ez az egyenes pályán annyira nem lényeges, de így a kanyarok lekövetése a szűrő alkalmazásával jóval gördülékenyebb.



5. ábra. *A mért és szűrt értékek ábrázolása vonaldiagramon*

Városi közlekedésben nagyobb sebességgel elvégezve a mérést hasonló eredmény tapasztalható (6. ábra), ezek is alátámasztják az előzőleg tett megállapításokat.



6. ábra. *A mért és szűrt értékek ábrázolása nagy sebességnél*

9. KONKLÚZIÓ

A fejlesztés végére a dinamikus szűrés alkalmazásával egy elég pontos objektumkövetési modellt sikerült alkotnom, amely használható a mindennapi életben járművek monitorozására, de akár ipari környezetben is, legyen szó egy telephelyről, raktárépületekről, vagy egy település tömegközlekedéséről. Különböző mintavételezési gyakorisággal végeztem a teszteket, és azt tapasztaltam, hogy a másodpercenként történő helyzetlekéréssel gyalogosok és lassabb járművek mozgása már egészen jól modellezhető, egy autó követéséhez azonban már szükséges a másodpercenként 3-5 mintavétel is. A mozgáspályák vizsgálata után megállapítható, hogy a Kálmán-szűrő használata jelentős javulást okozott, a mozgás szimulációja kevésbé darabos, és a jármű digitális árnyéka kevésbé tér le az elvárt pályáról.

KÖSZÖNETNYILVÁNÍTÁS

A Doktoranduszi Kiválósági Ösztöndíj Program (DKÖP) által támogatott projekt a Kulturális és Innovációs Minisztérium Nemzeti Kutatási Fejlesztési és Innovációs Alapból nyújtott, valamint a Budapesti Műszaki és Gazdaságtudományi Egyetem támogatása alapján valósult meg.

Külön köszönettel tartozom a MaxWhere Solutions Zrt. munkatársainak a fejlesztés feltételeinek biztosításáért, Dr. Galambos Péternek a munka irányításáért, és Dr. Czmerk András Józsefnek az útmutatásért.

IRODALMI HIVATKOZÁSOK

[1] *Rudolf E. Kalman-ETHW*, http://ethw.org/Rudolf_E._Kalman (Utolsó letöltés: 2021. 12.16).
[2] *Richard J. Meinhold–Nozer D. Singpurwalla: Understanding the Kalman Filter.*, The American Statistician, 37. évf. (1983) 2. sz., 123–127. p. ISSN 0003-1305, 1537-2731. http://www.tandfonline.com/doi/abs/10.1080/00031305.1983.10482723
[3] *Qiang Li–Ranyang Li–Kaifan Ji–Wei Dai: Kalman Filter and Its Application*, n 2015 8th International Conference on Intelligent Networks and Intelligent Systems (ICINIS) (konferenciaanyag). Tianjin, China, 2015, IEEE, 74–77. p. ISBN 978-1-4673-8221-2. http://ieeexplore.ieee.org/document/7528889/.
[4] *Geolocation API -Web APIs*, https://developer.mozilla.org/en-US/docs/Web/API/Geolocation_API (Utolsó letöltés: 2021. 12.08).
[5] *GPS*, https://gps-mw-watch-3000.web.app/ (Utolsó letöltés: 2021. 12.08).
[6] *M5Stack Core2 ESP32 IoT Development Kit for AWS IoT EduKit*, https://shop.m5stack.com/products/m5stack-core2-esp32-iot-development-kit-for-aws-iot-edukit (Utolsó letöltés: 2021. 12.08).
[7] *GPS Module with Internal & External Antenna (u-blox NEO-M8N)*, https://shop.m5stack.com/products/gps-module (Utolsó letöltés: 2021. 12.08).
[8] Mikal Hart: *TinyGPSPlus*, https://github.com/mikalhart/TinyGPSPlus (Utolsó letöltés: 2021. 12.08).
[9] Nick O'Leary: *Arduino Client for MQTT*, https://github.com/knolleary/pubsubclient (Utolsó letöltés: 2021. 12.09).
[10] *MQTT package*, https://www.npmjs.com/package/mqtt (Utolsó letöltés: 2021. 12.09).
[11] *OpenStreetMap*, https://www.openstreetmap.org/ (Utolsó letöltés: 2021. 12.08).
[12] *OSM to GeoJSON*, https://tyrasd.github.io/osmtogeojson/ (Utolsó letöltés: 2021. 12.09).