Önálló laboratórium beszámoló BME-TMIT

Készítette: Jámbor László Neptun-kód: L3ATGK
Szak: villamosmérnöki Szakirány: Infokummunikációs rendszerek
Ágazat: Infokommunikációs hálózatok és alkalmazások
E-mail cím: jamborlaszlo2001@gmail.com

Konzulens(ek): Fehér Gábor
Email címe(k): feher@tmit.bme.hu

Tanév: 2022/2023. 2. félév

Téma címe: Wi-Fi vezérlésű redőnymozgató elektronika

Feladat:

A feladatom egy Wi-Fi-n keresztül vezérelhető elektronika fejlesztése volt, amely egy léptetőmotort vezérel. A léptetőmotorral egy redőnyszerkezet forgórészét hajtjuk meg, így olyan redőnymozgató automatikát kapunk, amely az interneten keresztül bármilyen internethozzáféréssel rendelkező eszközzel (PC, okostelefon, tablet) vezérelhető, adott esetben távolról (pl. munkahelyről) is. A feladatnak nem célja valós rendszer installálása, azaz a motor létező redőnyhöz való csatlakoztatása, ehelyett csak a rendszer makett szinten történő fejlesztésére szorítkozunk.

1. A laboratóriumi munka környezetének ismertetése, a munka előzményei és kiindulási állapota

Bevezető/elméleti összefoglaló

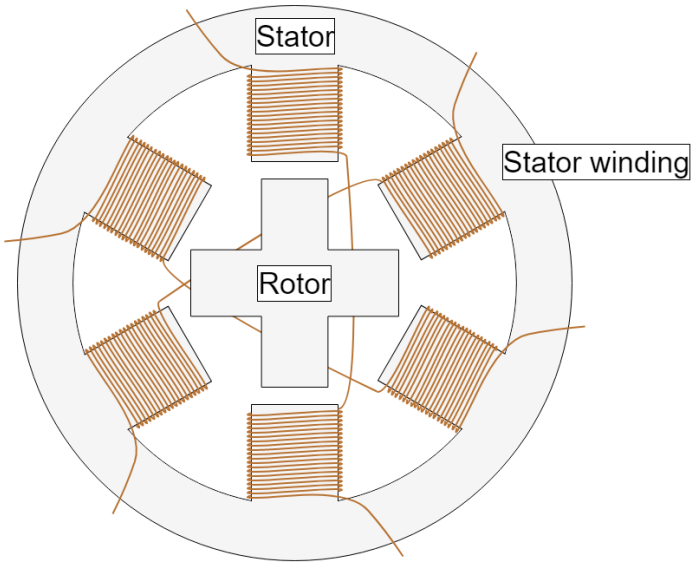
Munkám során 4 fő komponenst használtam: egy léptetőmotort a redőny mozgatásához, egy mikrokontrollert és egy interfészt a motorvezérléséhez valamint a Wi-Fi-n keresztüli vezérlés megteremtéséhez, illetve az Arduino IDE fejlesztőkörnyezetet a mikrokontroller programozásához. A következő részben ezen komponensek ismertetése és elméleti hátterük rövid összefoglalása következik a felsorolásnak megfelelő sorrendben.

Tekintsük át röviden a **léptetőmotorok** tulajdonságait és működési elvét! A léptetőmotor egy olyan szénkefe nélküli motor, amely – a legtöbb alapvető villanymotorral ellentétben – nem csupán forogni kezd tápfeszültség hatására, hanem ennél sokkal precízebben vezérelhető. Egy léptetőmotor egy fordulatot - típustól függően - jellemzően néhány 10 vagy néhány 100 diszkrét lépés egymásutánjaként tesz meg. Könnyen belátható, hogy az, hogy egy motor esetében hány diszkrét lépés szükséges egy teljes fordulat megtételéhez, annak függvénye, hogy a motor forgórésze hány fokot fordul egy lépés alkalmával. Ha tehát például egy léptetőmotor forgórésze egy lépés során 11,25°-ot fordul, akkor 360°/11,25° = 32 lépés fog kitenni egy teljes körbefordulást. Az egy lépés során bekövetkező szögelfordulás nagysága pedig a motor belső szerkezeti kialakításától függ, és ezt az értéket a gyártó az adatlapon mindig feltünteti. A léptetőmotorok felépítésének részletes ismerete nem szükséges a projekt megvalósításához, alapvető működési elvével azonban érdemes megismerkedni.

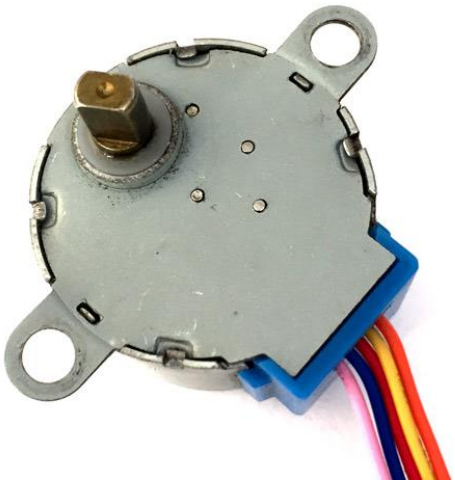
A léptetőmotor állandó mágnesből vagy vasból készült forgórészét az állórészben elhelyezett tekercsek (elektromágnesek) veszik körül a 1. ábrán látható módon[1], melyken a vezérlésnek megfelelően áramot folyatunk. Fontos látni, hogy a szemközti tekercsek összeköttetésben vannak, azaz együtt kapcsolhatók áram alá. A forgórész (rotor) mindig az áramjárta tekercspár által kijelölt vonalba áll be a fellépő mágneses erőhatás miatt. Így lehetséges a motor léptetése. A tekercseket megfelelő sorrendben egymás után áram alá helyezve a rotor forgásba is hozható. Ebből látható, hogy a léptetőmotor egy digitálisan, precízen vezérelhető eszköz. A tekercseken digitális jelek segítségével megfelelő szekvenciában folyatva áramot, a motor léptethető vagy – az adott motor korlátai között – tetszőleges fordulatszámon üzemeltethető. A vezérléshez interfészt szokás használni, mivel a vezérlő elektronika (esetünkben mikrokontroller) jellemzően nem képes motor áramigényét kiszolgálni. Szélesebb körültekintéshez az alábbi oldalt ajánlom: [1]

Munkám során az igen népszerű **28BYJ-48** típusjelzésű léptetőmotort (2. ábra) használtam. Ennek adatlapja az alábbi linken olvasható: [2]
Ez egy egyszerű, olcsó, de sok célra alkalmas léptetőmotor, amely 5-12V-ról üzemeltethető. A motor tengeyle (1/64-es áttételének is köszönhetően) 2048 lépés után tesz meg egy teljes kört, ezért igen finomam vezérelhető. A motorban 4 tekercspár található, azaz négy digitális jelre van szükségünk a motor vezérléséhez. A motor forgatónyomatéka nem feltétlenül elegendő redőny mozgatásához, azonban a makett szinten történő fejlesztéshez tökéletesen megfelel.

[1] A ma használt léptetőmotorok felépítése részben eltérő, működési elvük azonban alapvetően azonos az ábrán látható modellével.



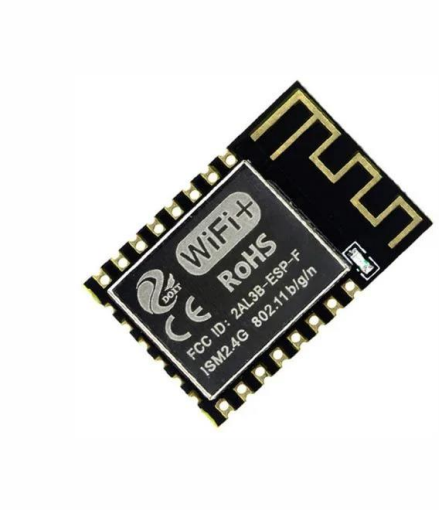
1. ábra: a léptetőmotor sematikus rajza



2. ábra: a 28BYJ-48 léptetőmotor

A **mikrokontrollelrek** lényegében miniatűr számítógépek: processzorral, memóriával, valamint ki- és bemenetekkel rendelkeznek. Igen széles körben használnak mikrokontrollereket beágyazott rendszerekben, a lényeg azonban mindig ugyanaz: a mikrokontroller a rátöltött szoftver segítségével végrehajt valamilyen feladatot: kimeneti jelekkel válaszol a bemeneteire érkező jelekre, kielemzi és tovább küldi a rákötött szenzorok adatait, vezérel egy autó ABS-rendszerét vagy épp léptetőmotorokat mozgat. A mikrokontrollerek általában kis méretűek, kis fogyasztásúak, teljesítményük és memóriájuk korlátozott.

A motor vezérléséhez, illetve a Wi-Fi kapcsolat létrehozásához és az interneten keresztüli eléréshez szükséges weboldal futtatásához az ESP8266 családba tartozó **ESP-12F** jelzésű mikrokontrollert használtam (3. ábra). Ennek adatlapja az alábbi linken található: [3]
Ez egy kis méretű, alacsony fogyasztású mikrokontroller, amely Wi-Fi Access Point-ként is alkalmazható, valamint csatlakozni is képes Wi-Fi hálózatra. A modul 3,0-3,6V-ról üzemeltethető, 80mA átlagos áramfelvétellel.

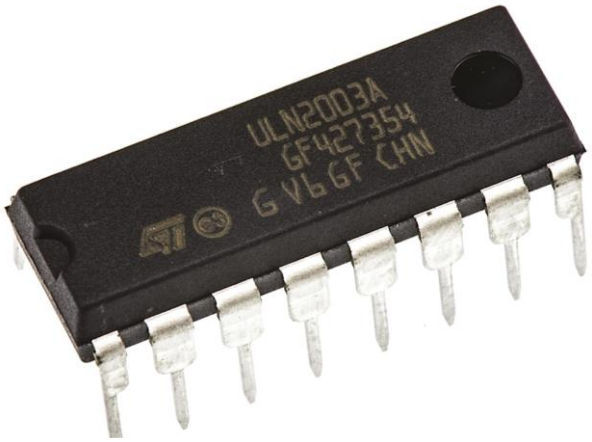


3. ábra: ESP-12F mikrokontroller

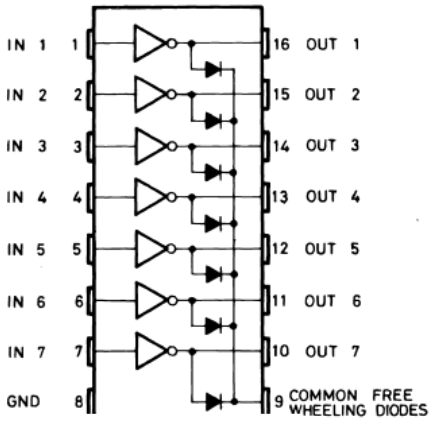
Ahhoz, hogy a mikrokontrollerrel a léptetőmotort vezérelhessük, **interfészre** van szükségünk, mivel a motor által felvett viszonylag nagy áram túl nagy terhelést jelentene a mikrokontroller digitális kimenetei számára.

Erre a célra az **ULN2003A** típusú interfészt használtam (4. ábra). Ennek adatlapja az alábbi link alatt található: [4]

Az IC belső felépítése igen egyszerű. Az 5. ábrán látható módon csak invertereket és visszáram ellen védő diódákat tartalmaz.

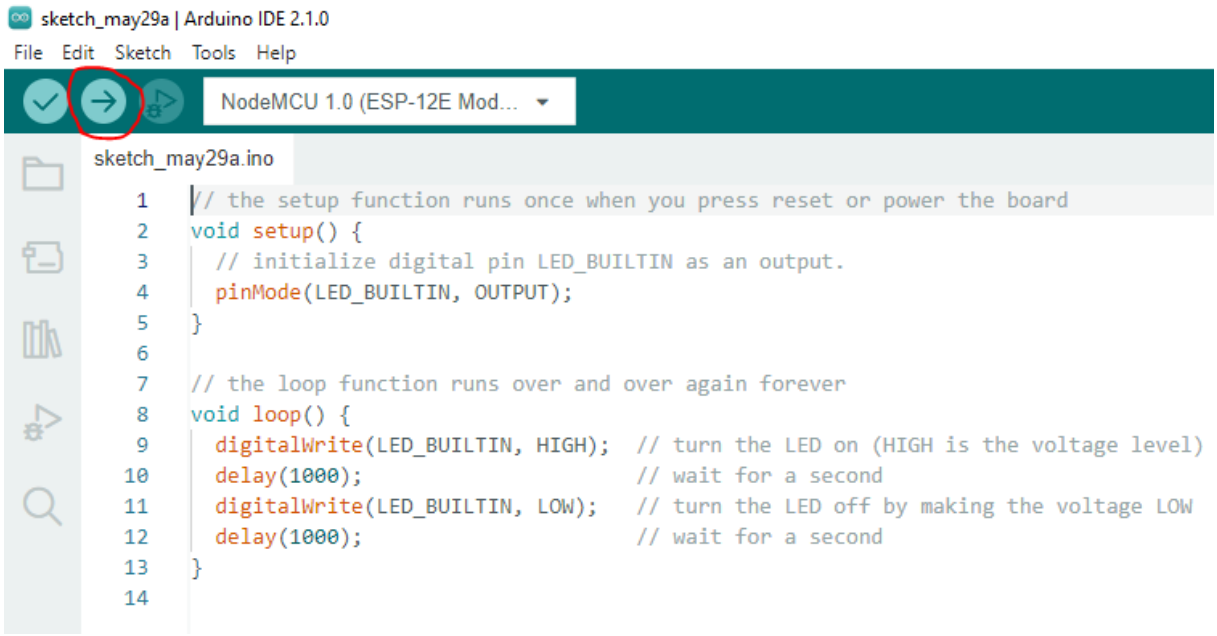


4. ábra: ULN2003A interfész



5. ábra

A mikrokontroller programozásához a népszerű **Arduino IDE**-t használtam. Az IDE kezelése igen egyszerű, felhasználói felülete letisztult és jól átlátható. Az **Arduino** keretrendszerben alapvetően C++ nyelven írhatunk programokat, azonban számos beépített függvény és C++-ban nem megtalálható feature segíti a munkánkat. Ilyen például a ,,pinMode()” függvény, mellyel deklarálhatjuk, hogy a mikrokontrollerünk egy adott pin-jét be- vagy kimenetként kívánjuk használni. Egy Arduino kódban általában található egy ,,void setup(){*}” és egy ,,void loop(){*}” szekció. A mikrokontroller indításakor a ,,void setup(){*}” csak egyszer fut le. Ide kerülnek az eszköz konfigurálásához szükséges kódrészletek, például a fent említett ,,pinMode()” függvény általában itt kerül meghívásra. A ,,void loop(){*}”-ban található kódrészek ellenben, - ahogy ezt a ,,loop” elnevezés is sugallja – folyamatosan újra és újra lefutnak amíg az eszköz üzemel. A 6. ábrán látható rövid kód egy LED-et villogtat. Itt is megfigyelhető, hogy a ,,setup” részben a konfiguráláshoz szükséges kódrészlet került, a ,,loop”-ba pedig a kódnak azon része, amelynek újra és újra meg kell hívodnia ahhoz, hogy a LED folyamatosan villogjon. Az elkészült kódot a fejlesztőkártyánkra a bal felül található, pirossal bekarikázott gomb megnyomásával tölthetjük fel.



6.ábra: az Arduino IDE

A munka állapota, készültségi foka a félév elején

A projekt megvalósításához az alábbi eszközöket kaptam készhez:

- ESP-12E mikrokontroller
- 28BYJ-48 léptetőmotor
- UA2003A interfész
- jumper kábelek

2. Az elvégzett munka és eredmények ismertetése

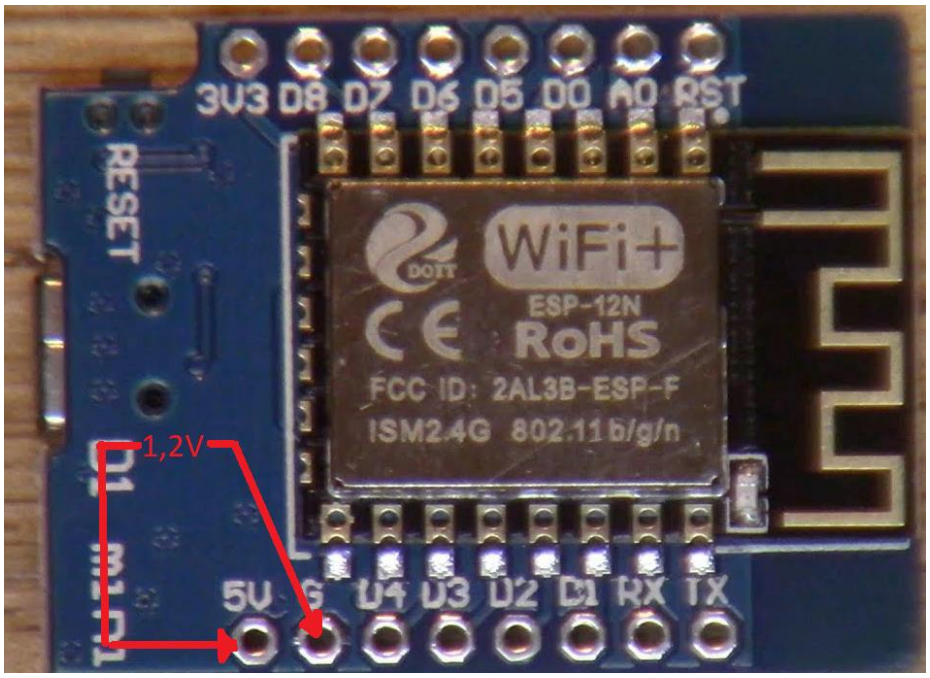
Az általam konkrétan elvégzett munka bemutatása

A félév során végzett munkám a felhasznált komponensek megismerésével és használatuk, programozásuk, felhasználásuk terén történő tájékozódással kezdődött, mivel korábban nem dolgoztam még ezekhez hasonló eszközökkel. Elsősorban YouTube videókból és egyéb internetes forrásokból (gondolok itt elsősorban DIY oldalakra és fórumokra) próbáltam összegyűjteni a munka elkezdéséhez szükséges tudásanyagot. Meg kellett ismerkednem a mikrokontroller tulajdonságaival és az általa kínált lehetőségekkel, az Arduino keretrendszerrel, a léptetőmotorok működésével és felhasználásával. Emellett olyan források után is néztem melyekről úgy gondoltam, segíteni fogják a későbbi munkát.

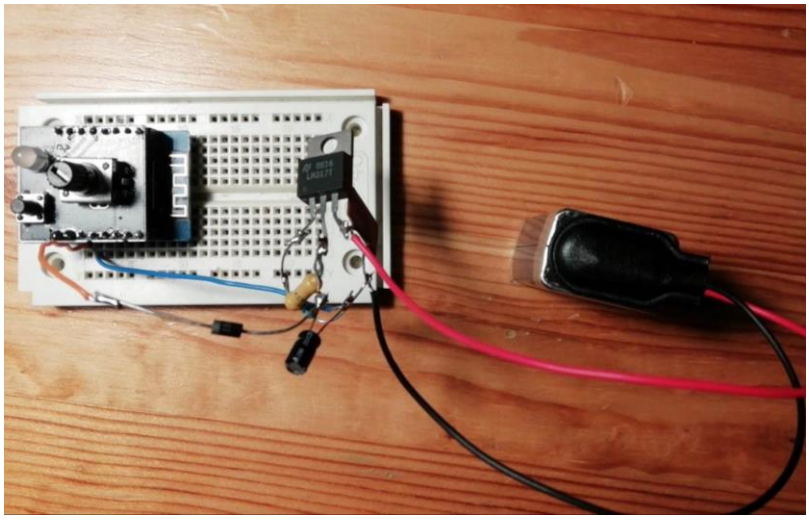
A következő lépés az Arduino IDE telepítése volt. Szerencsére ez a szoftver Windows alatt is használható, így könnyűszerrel telepíteni tudtam a saját rendszeremre. Az Arduino IDE-vel kapcsolatban nagyon jó tapasztalataim vannak: a telepítés zökkenőmentes volt, és a szoftver minden további teendő szüksége nélkül a telepítést követően egyből gördülékenyen üzemelt. Később a munkám során is szívesen dolgoztam vele: a platform rendkívül jól átlátható, bug-októl mentes és kellemes kezelői felülettel rendelkezik.

Az Arduino ide telepítése után a szükséges bővítmények installálásához fogtam. Az alapvető könyvtárakat az alkalmazáson belül le tudtam tölteni, így például a „ESP8266WiFi.h”-t vagy a léptetőmotor vezérléséhez használt „ AccelStepper.h”-t. Néhány szükséges további könyvtárat azonban nem találtam meg az Arduino IDE beépített böngészőjében, ezért például a „ ESPAsyncTCP.h” vagy a „ ESPAsyncWebSrv.h” könyvtárakat GitHub-ról töltöttem le.

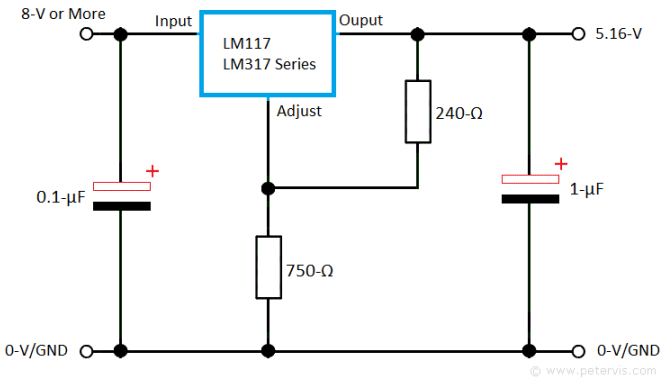
A fejlesztőkörnyezet felállítása után a mikrokontroller számítógéphez történő csatlakoztatása következett. Itt ütköztem az első nehézségbe. A fejlesztőkártyát USB kábelen keresztül csatlakoztattam a számítógéphez, azonban az eszköz semmijen életjelet nem mutatott, és a számítógépen sem jelent meg mint csatlakoztatott USB eszköz. Számos YouTube videót megnéztem a problémával kapcsolatban és fórumokat is olvastam, de hosszas próbálkozás után sem sikerült megoldást találnom. Ezen a ponton elkezdtem gyanakodni arra, hogy esetleg valamilyen hardveres hiba lehet a háttérben. Ezt a gyanúmat erősítette az is, hogy a PC-hez csatlakoztatva a mikrokontrolleren található LED sem világított. Méréssel megállapítottam, hogy a 7. ábrán látható módon a fejlesztőkártya 5V-os pin-jén mindössze kb. 1,2V mérhető, azaz a mikrokontroller nem kap megfelelő tápfeszültséget az USB kábelen keresztül. Ennek tudatában folytatva az internetes információgyűjtést, azt találtam, hogy ez nem egyedi eset: jellemzően a tápvonalban elhelyezett védődióda meghibásodása okoz ilyen jellegű hibát ezeknél a moduloknál. Annak érdekében, hogy a munkát folytatni tudjam, egy LM317 feszültségstabilizátor IC felhasználásával készítettem egy 5V tápfeszültséget szolgáltató kapcsolást, amit egy 9V-os teleppel tápláltam meg. Az áramkörről kép és kapcsolási rajz a 8. és 9. ábrákon látható. Az így előállított 5V-os feszültséget a modulra kapcsoltam. Az eszköz így már tökéletesen működött, az Arduino IDE felismerte, képes voltam programokat tölteni rá, illetve futtatni rajta azokat.



7. ábra: a modul 5V-os pin-je és a föld között kb.1,2V mérhető



8. ábra (felül): az 5V-ot szolgáltató tápegység



9. ábra (balra): az 5V-ot szolgáltató tápegység kapcsolási rajza

Miután úgy éreztem, megismerkedtem a mikrokontroller programozásának alapjaival olyan szinten, hogy tovább tudok lépni, rákötöttem a léptetőmotort a mikrokontrollerre, és elkezdtem ismerkedni az „AccelStepper.h” könyvtárral. Itt a turorial videók és oldalak mellett nagy segítségemre szolgált az alábbi oldal : [5], melyet később is sokat tanulmányoztam. Az „AccelStepper.h” könyvtárban a léptetőmotor precíz vezérléséhez használható függvények sora található. A motor vezérlésénél is először néhány egyszerű feladat megvalósításához fogtam, annak érdekében, hogy megértsem a könyvtár használatának és a motor vezérlésének alapjait. Az internetről kölcsönöztem egy kódot, amely a motor tengelyét ciklikusan egyik, majd másik irányba forgatja, később ezt részben módosítottam, majd programot írtam, ami a motor tengelyén másodpercenként 30°-ot fordít majd megállítja azt, magyarul egy óra másodpercmutatójához hasonló mozgást végez. A munka ezen fázisában, illetve a későbbiekben is a következő néhány oldalon találtam a legtöbb hasznos információt: [6], [7], [8], [9].

Mikor úgy éreztem, hogy a készültségem a szükséges szintet elérte, hozzáfogtam a munka érdemi részének elvégzéséhez, azaz a redőnymozgató fejlesztéséhez. Úgy döntöttem, hogy először a motor vezérlését offline valósítom meg, magyarul az irányítást egy fizikai nyomógombbal oldottam meg. Az „AccelStepper.h” függvényeit használva megalkottam egy első verziót, amibe bedrótoztam az elképzelt mozgatandó redőny hosszát (léptetőmotor)lépésekben, és a rendszer gombnyomásra felhúzta a redőnyt, újabb gombnyomásra pedig leengedte azt.

A lehetőségeim korlátozottak voltak, mivel csak egy nyomógomb állt rendelkezésemre, ezért a redőny mozgatása terén nem tudtam további funkciókat megvalósítani. Ennél fogva elkezdtem a feladat IoT-s felével foglalkozni, annak érdekében, hogy egy megfelelő weboldal segítségével tetszőleges funkciókat megvalósító virtuális gombokat használhassak. Egy internetes forrásból [9] sikerült a mikrokontrollerrel Wi-Fi hálózatra csatlakoznom, és képes voltam egy LED-et az interneten keresztül be és ki kapcsolni. A Wi-Fi kapcsolat használatához és a weboldal futtatásához az „ESP8266WiF.h”, „ESPAsyncTCP.h” és az „ESPAsyncWebSrv.h” könyvtárak függvényeit alkalmaztam. A mikrokontroller csatlakozik a kódba bedrótozott azonosítójú Wi-Fi hálózatra, és futtatja a kódban megírt weboldalt. Ez a weboldal azonban nem volt számomra megfelelő, elsősorban azért, mert az oldalon gombok kattintásra – magnógomb-szerűen –„LOW”-ból „HIGH” állapotba billentek és ott is maradtak, ami nem volt kompatibilis a kódom már korábban megírt, léptetőmotort vezérlő részével, mivel azt korábban egy fizikai nyomógomb használatával alkottam meg, amely értelemszerűn csak addig szolgáltat „HIGH” jelet, amíg lenyomva tartjuk. Tehát a weboldalon is olyan gombokat kellett létrehoznom, amelyek csak a lenyomás idejéig küldenek „HIGH” jelet, felengedéskor visszatérnek „LOW” állapotba. (Egyébként a folyamatos „HIGH” állapot, tehát például a gomb lenyomva tartása sem okoz hibás működést, ilyenkor egyszerűen csak nem mozdul a redőny.) Emellett az oldal megjelenésén és elrendezésén is változtatni akartam. Mindezekhez fel kellett frissítenem a HTML és CSS ismereteimet (ebben az alábbi forrás volt legnagyobb segítségemre: [7]), valamint egy új, szintén internetről származó [6] kódból indultam ki, mivel ezt jobban átláttam és könnyebben a saját igényimre tudtam szabni a fent leírt módon.

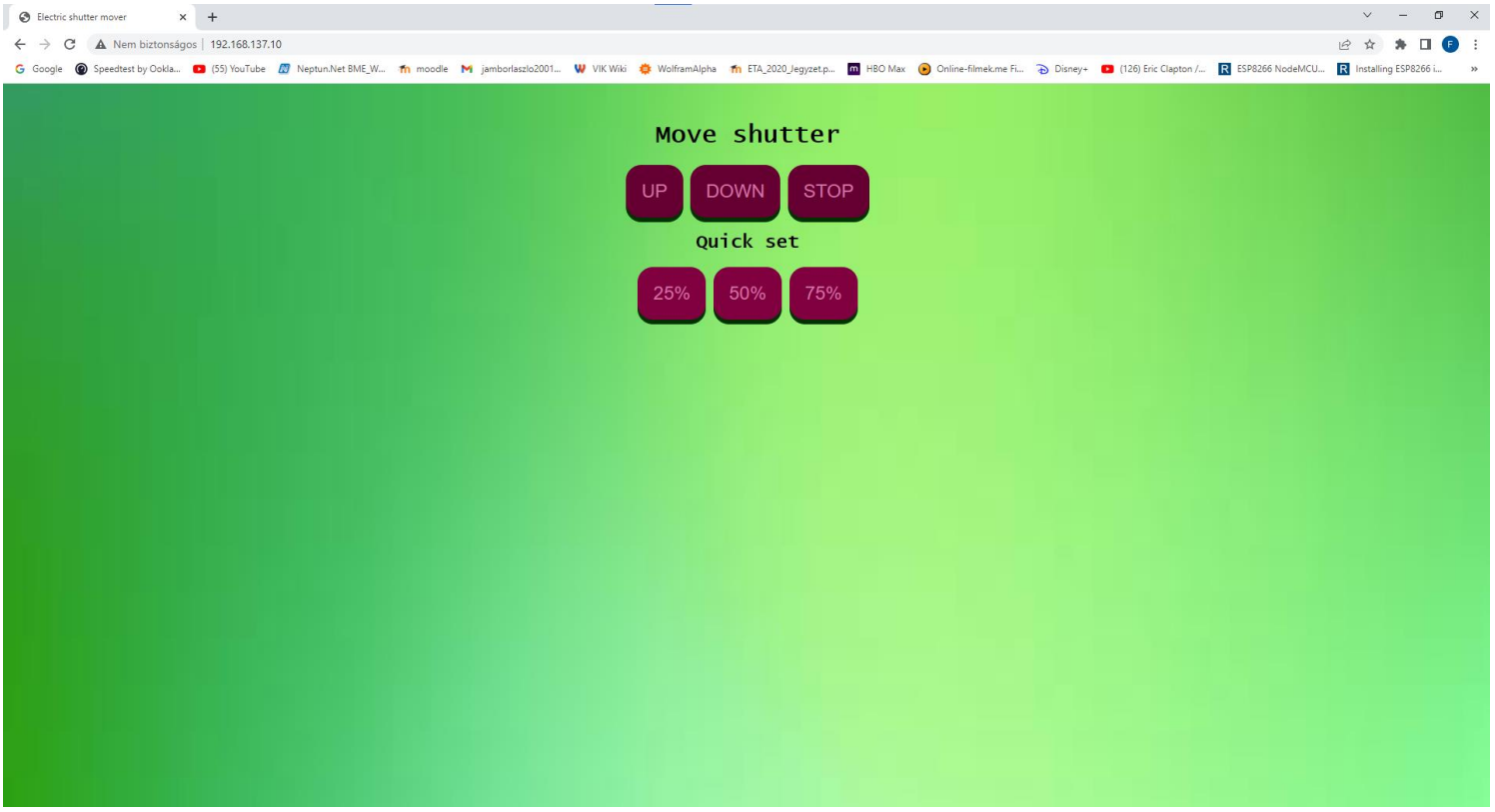
Az interneten keresztüli kommunikáció megvalósítása után egy olyan program implementálásába kezdtem, melyben a redőny tetszőleges módon fel vagy le mozgatható, illetve közbenső helyzetben is megállítható. Ehhez a weboldalon elhelyeztem egy „Up”, egy „Down” és egy „Stop” gombot. Az „Up” vagy „Down” gombok valamelyikének megnyomásakor a program úgy vezérli a léptetőmotort, hogy az a megfelelő irányba pontosan annyi lépést tegyen meg, amennyi az adott (alsó vagy felső) végállásba történő eljutáshoz szükséges. A „Stop” gomb megnyomásakor a program megállítja a léptetőmotort az aktuális

helyzetben. A „Stop” funkció miatt fontos volt egy változó („position”) létrehozása, amelyben a redőny aktuális pozícióját tárolom, annak érdekében, hogy megállítás után, teljes fel- vagy lehúzás esetén a program végálláskapcsoló alkalmazása nélkül is tudja, mikor ér a redőny a végállásba. A program a léptetőmotor minden léptetése után iránytól függően eggyel növeli vagy csökkenti a „position” változó értékét, melynek 0 értéke az alsó, míg a maximális értéke a felső végállást reprezentálja.

Ezután egy újabb funkciót valósítottam meg rendszeren: gyorsgombokat hoztam létre, melyekkel a redőny automatikusan 25, 50 vagy 75%-ig felhúzott állapotba állítható. Ehhez további gombokat hoztam létre a weboldalon. A program megnézi, hogy a kiválasztott pozíció a redőny aktuális helyzete alatt vagy felett található, és ebből meghatározza a léptetőmotor forgásirányát. Ezután megnézi, hány lépés van aktuális helyzete és a célpozíció között, és ennyit léptet a motoron. A weboldalnak erről – az egyelőre végleges – állapotáról készített képernyőfelvétel a 10. ábrán látható.

A rendszert több eszközön (PC, okostelefon, tablet) és több böngészővel (Google Chrome, Microsoft Edge, Mozilla Firefox) is kipróbáltam. A működés az összes fent felsorolt környezetben gördülékeny és hibamentes volt, eltekintve attól, hogy néhányszor a weboldal rövid időre lefagyott.

Terveztem egy ábra elhelyezését is a kezelői felületen, amely grafikus visszajelzést nyújt a redőny aktuális helyzetéről, ennek megvalósítás azonban idő hiányában nem fejeződött be.



10. ábra: a weboldal végleges verziója

Összefoglalás

- Tudásanyagra tettem szert és tapasztalatot szereztem a mikrokontrollerek és az Arduino világában
- Megismerkedtem a léptetőmotorok tulajdonságaival és alapvető tapasztalatra tettem szert a programozásuk terén
- Írtam 250 sor kódot
- C++-ban megírtam a léptetőmotor vezérléséhez szükséges kódot
- HTML-ben létrehoztam a felhasználói felületként szolgáló weboldalt
- Wi-Fi-n keresztül kommunikáltam a mikrokontrollerrel

3. Irodalom, és csatlakozó dokumentumok jegyzéke

A tanulmányozott irodalom jegyzéke:

[1] https://uk.rs-online.com/web/content/discovery/ideas-and-advice/stepper-motors-guide

[2] https://www.mouser.com/datasheet/2/758/stepd-01-data-sheet-1143075.pdf

[3] https://docs.ai-thinker.com/_media/esp8266/docs/esp-12f_product_specification_en.pdf

[4] https://datasheetspdf.com/pdf-file/366764/STMicroelectronics/ULN2003A/1

[5] https://www.airspayce.com/mikem/arduino/AccelStepper/classAccelStepper.html

[6] https://randomnerdtutorials.com/

[7] https://www.w3schools.com/

[8] https://www.instructables.com/

[9] https://www.instructables.com/Internet-Controlled-LED-Using-NodeMCU/