
ESZTERHÁZY KÁROLY KATOLIKUS EGYETEM

Matematikai és Informatikai Intézet

# A Robotika témakör oktatásának új lehetőségei, módszerei, eszközei: Okosotthon-modell kivitelezése micro:bit mikrokontrollerrel

**Készítette**
Szilágyi Debóra
programtervező informatikus BSc

**Témavezető**
Dr. Geda Gábor
egyetemi docens

Eger, 2024

# Tartalomjegyzék

# Bevezetés

Mi volt a célom ezzel a témával? Tehetik fel magukban a kérdést. Nos erre a válaszom igen egyszerű. Szerettem volna egy olyan témával foglalkozni, ami hasznos lehet a későbbi felnövő generáció, vagy azok számára, akiket érdekel a robotika és fejlődni, vagy csak szimplán új dolgokat megismerni szeretnének. Ebbe a körbe pedig én magam is beletartozom. Rengeteg új tapasztalatot lehet szerezni egy ilyen kutatás alatt, így ezt az újonnan szerzett tudást szeretném megosztani ebben a dokumentumban.

Mint jól tudjuk, az idő haladásával az informatika is gyorsan változik. Egyre több lehetőség nyílik meg az emberek előtt, amik még kényelmesebbé tehetik az életünket. Ilyen például az okos otthon is, ami egyre nagyobb népszerűségnek örvend. Egyszerű a használata, sok lehetőség rejlik benne, és ami a téma egyik fontos része, hogy könnyen érthető, letisztult legyen a használata. Amit viszont én ezen kívül még fontosnak tartok, hogy a felhasználóknak legyen lehetőségük mögé látni a rendszernek, amit egyre többen használnak a saját otthonaikban. Hiszen ez által sokkal biztonságosabban használhatják ezeket, illetve néhány alapvető tudást is könnyedén megszerezhetnek ezáltal, ami a javukra válhat a későbbiekben. Ugyanakkor, mivel valószínűleg a jövőben több szerepet kapnak majd a robotok, nem árthat, ha belelátnak a robotika alapjaiba.

Továbbá, a robotikával ki lehet élni a kreativitást, fejleszthetőek a problémamegoldó képességek és összehozza a hasonló érdeklődésű embereket. Jelentős mérföldköveket lehet elérni, vagy csak szimplán örömöt is lehet lelni ezekben a projektekben. Persze nem csak a robotika tanítását tartom az egyetlen fontos résznek, de szerintem jelentős lehet, ha az ember nem csak az általános tárgyakkal ismerkedhet meg.

Mi az egyetemen Arduino UNO mikrokontrollereket [1] használtunk, és különböző szenzorokat kötöttünk hozzá. Ezek során rengeteg újdonságot ismertünk meg és tanultunk meg róluk. Viszont mindig úgy gondoltam, hogy ez a megoldás kezdők részére hirtelen soknak tűnhet. Így személy szerint szeretnék egy olyan megközelítést megmutatni, ami egyszerűen értelmezhető, és akár egy robotikában kevésbé jártas személy is képes lehet az összeállítására. Alapvetően olyan elemekkel próbáltam meg dolgozni, amikkel gyakran találkozik az ember egy ilyen tervezetben, mint például a világítás és hűtés/fűtés kezelése. Ezt mind egy olyan mikrokontrollerrel kiviteleztem, amely vé-

[1] A mikrokontroller, más néven mikrovezérlő, egy olyan IC (integrált áramkör), ami maga is egyfajta számítógép. Van benne processzor, memória és be-kimeneti vezérlők [1].

leményem szerint kevésbé elterjedt még, valamelyest el is tér a többitől, de rengeteg potenciál rejlik benne. Innen született meg az ötlet, hogy használjam ki ezeket az újfajta lehetőségeket. Magáról a micro:bit mikrokontrollerről a 7., az Arduino UNO-ról pedig a 8. oldalon található meg több információ.

# 1. fejezet

# Okosotthon-rendszer

Alapvetően akkor beszélhetünk okosotthon-rendszerről, ha vannak okos eszközeink, amiket nem csak irányíthatunk, hanem maguktól is működnek és a szenzorok képesek kommunikálni egymással, így reakciókat is ki tudnak váltani egymásból. A leggyakrabban okosított funkciók a világításvezérlés, hűtés- és fűtésvezérlés, riasztórendszer, háztartási gépek vezérlése, illetve az árnyékolásvezérlés (redőnyök, függönyök irányítása). Vezérlésükre használhatunk applikációkat, hangvezérlést vagy a szenzorok automatizált működését [2].

## 1.1. A piacon elérhető rendszerek

Manapság egyre több okosotthon-rendszer alakul ki, és folyamatos versengés áll fent, hogy melyek számítanak a legjobbaknak. Egy pár a legismertebbek közül:

Chameleon Smart Home (egyelőre vezetékes) – egy magyar fejlesztés (elérhető magyar nyelvű terméktámogatás), amely arra törekszik, hogy rendszerfüggetlen és könnyedén egységesíthető legyen. Bármilyen tetszőleges vezérlőt, applikációt vagy funkciót szabadon használhatunk hozzá. Egy felületről irányítható, automatizáltan működteti a szenzorokat, de a hangutasításokat is elfogadja. Kompatibilis az iOS és Android alapú rendszerekhez is [3].

Loxone (vezetékes) – osztrák fejlesztés, szintén a különálló eszközök összefogásában segítő, stabil rendszer. Intelligens automatizációval vezérel. Zárt rendszerű, így leginkább a saját termékeire és fejlesztéseire a leghatékonyabb. Ezen előnye, hogy saját fejlesztésű eszközöket is készítenek [4].

Zipato (vezeték nélküli) – lényege megegyezik a fentebb megemlített rendszerekkel, ám ez annyiban másabb rendszer, hogy a hangsúlyt a biztonságba fektették. Energia hatékonyan és komfortosan irányíthatjuk távolról vagy automatizálhatjuk vele eszközeinket. Több okosotthon-rendszer eszközeivel is könnyedén összekapcsolható [5].

# 2. fejezet

# A BBC micro:bit mikrokontroller

Ezt a mikrokontrollert kifejezetten gyerekek oktatási céljaira használják és készítették. Nem csak azért nagyszerű választás, mert könnyű programozni, hanem mert rengeteg lehetőség van benne a saját beépített szenzoraival. Ezáltal könnyedén megtanulható egy-egy szenzor működése és működtetése egyaránt. Illetve ezen felül könnyedén bővíthető, és a beépítetteken felül más, tetszőleges szenzorokat is hozzáköthetünk és irányíthatunk.

Tökéletes választás lehet a kezdők számára, hiszen ideális alapokat nyújt, de haladó szinten is remekül megállja a helyét és kiváló projekteket lehet vele készíteni.

## 2.1. Felépítése

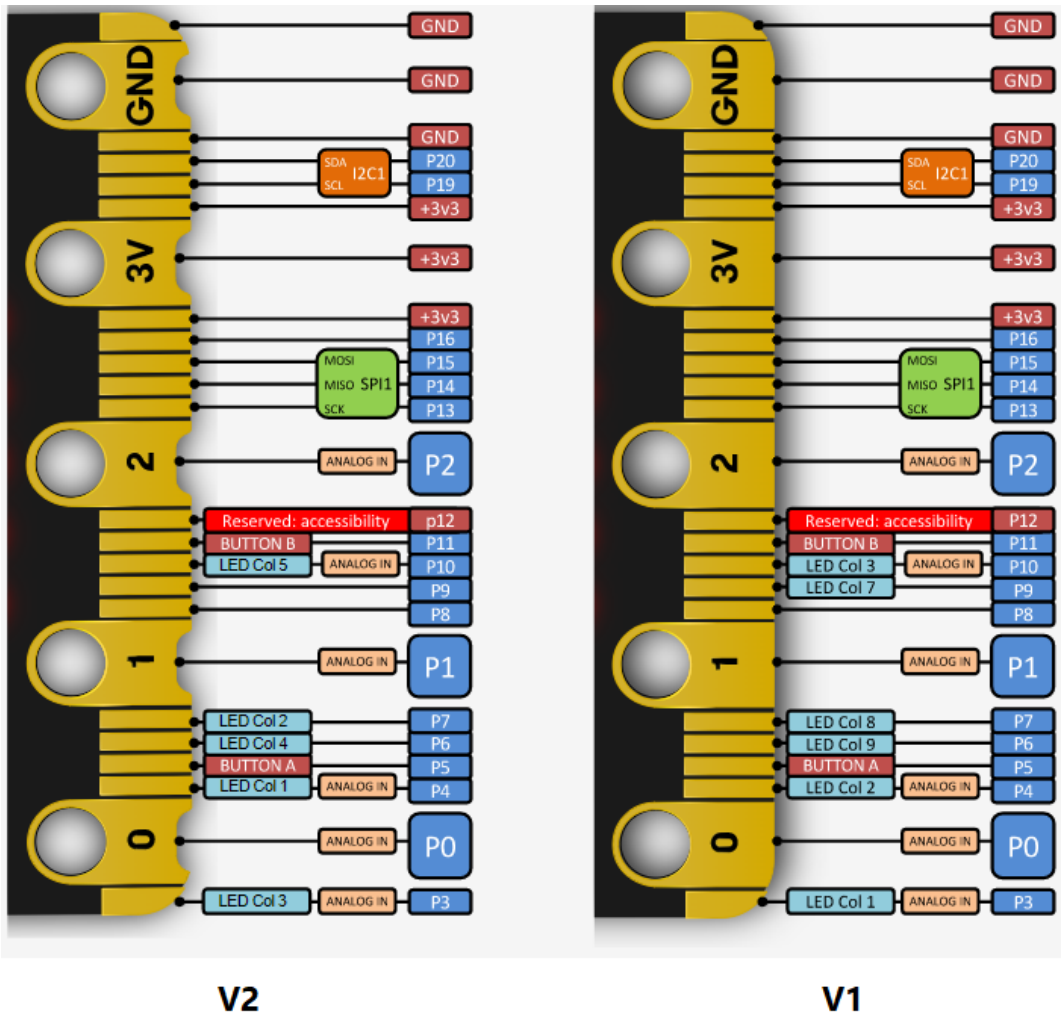
A mikrokontroller felépítését jelen esetben két szempontból szeretném bemutatni: a pinek [1] kiosztását, illetve a mikrokontrollerben megtalálható szenzorokat.

### 2.1.1. v1 és a v2 verziók

A v1 és v2 verziók nem csak a szenzorokban, hanem a pinek kiosztásában is eltérnek. A pinjeinek kiosztása meghatározó egy projekt összerakásánál. Sok esetben, ha plusz szenzorokat szeretnénk felhasználni, és hozzákötni a mikrokontrollerhez, akkor figyelnünk kell arra, hogy melyik pinek milyen feladatokat látnak el.

Szenzorokban leginkább megegyeznek, az eltérés abban van, hogy a v2-es verzió még több szenzorral rendelkezik. A micro:bit v1 tartalmaz LED-eket, gombokat, Bluetooth antennát, gyorsulásérzékelőt és mágneses érzékelőt. A v2 mindezek mellett tartalmaz még érintésérzékeny logót, mikrofont, 3-tengelyű digitális mozgásérzékelőt és hangszórót [6]. A 2.2. és a 2.3. ábrán jól láthatóak a két verzió közötti különbségek.

[1] Más néven élcsatlakozók/lábak.




2.1. ábra. A micro:bit mikrokontroller pinjeinek kiosztása [7].

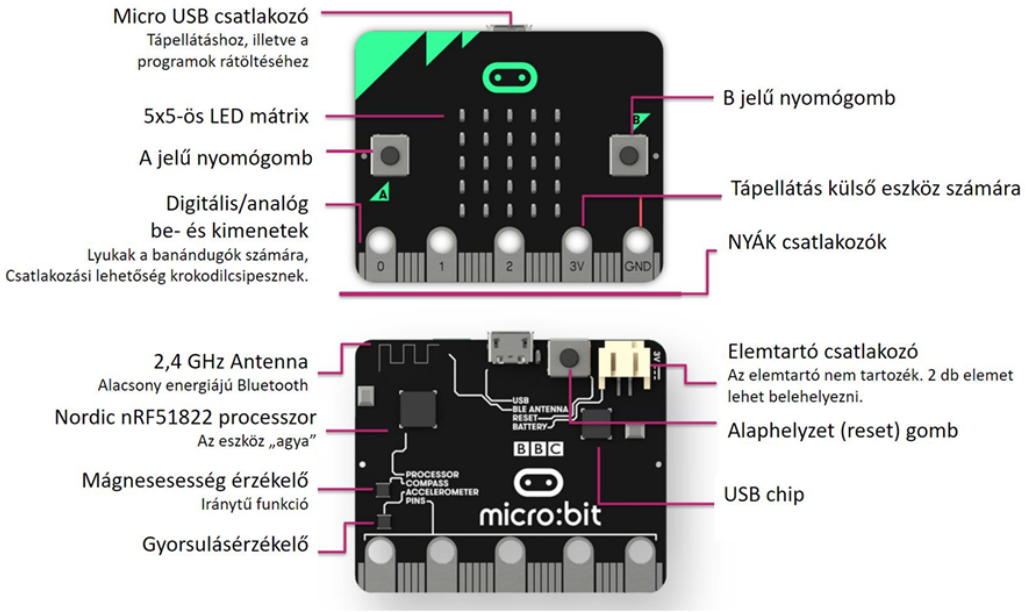
.

### 2.1.2. Összehasonlítva más mikrokontrollerrel

Mint ahogyan fentebb már említettem, mi az egyetemen más mikrokontrollerrel ismerkedtünk meg és annak használata eltérő volt a micro:bitől. Nem csak felépítésében más, de pinjeik kiosztása is eltérő, illetve kódolási lehetőségeikben is mások.

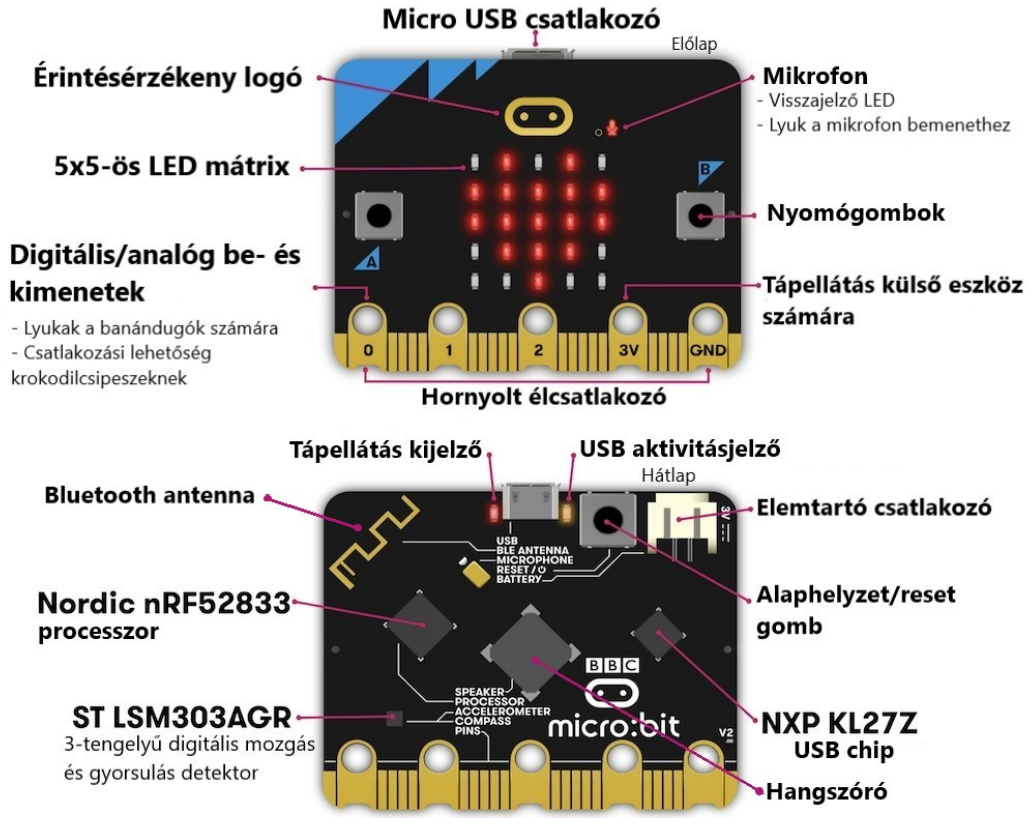
Mi az Arduino UNO-t használtuk, és az Arduino IDE környezetben írtunk rá kódokat C++ nyelvben. Persze nem csak Aruino-t lehet itt programozni, ugyanúgy van lehetőség micro:bitet is, csak le kell tölteni a megfelelő kiterjesztéseket és csomagokat hozzá. Így annyiból megegyezhet a két mikrokontroller, hogy ugyanúgy van lehetőség mindkettőt ugyanazon nyelven és környezetben fejleszteni.

Ugyanakkor a 2.4. ábrát összehasonlítva a 2.2. vagy akár a 2.3. ábrával, látható, hogy felépítésükben eltérőek. Nincs beépített szenzor az Arduinon, de cserébe több szabad port áll a rendelkezésünkre, amikre szabadon köthetünk tetszőleges szenzorokat. Ezek a portok pedig praktikusan vannak kivezetve, így könnyű megoldani a csatlakoztatást. Személy szerint nekem ez a megoldás jobban tetszett, hiszen a micro:bit esetében




2.2. ábra. A BBC micro:bit v1-es verziója [8].

.




2.3. ábra. A BBC micro:bit v2-es verziója.

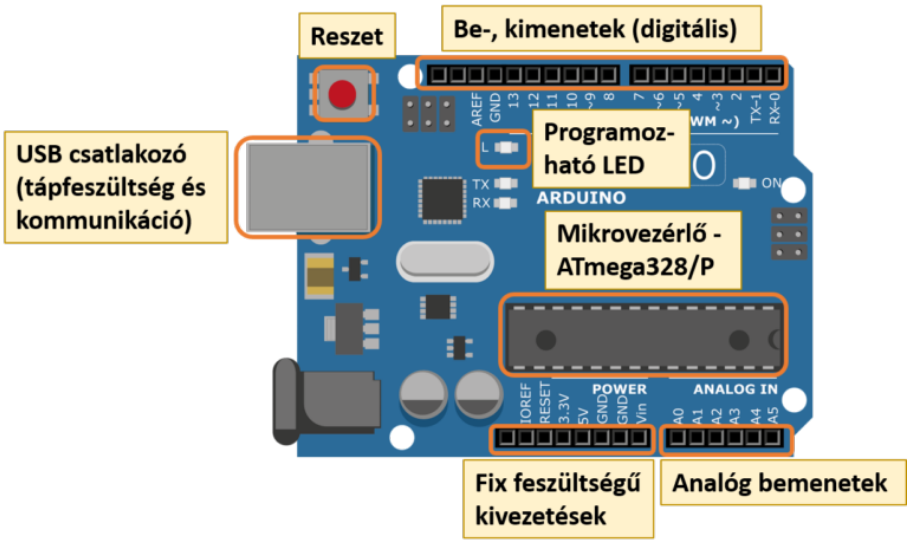
alapból krokodilcsipeszek és banándugók használatára lett kialakítva, ha breadboard-ot szeretnénk használni, kelleni fog egy plusz expansion board, ami hasonlóan vezeti ki a lábait, és csak ezáltal használhatunk hozzá jumper kábelt. Illetve az Arduino

szemlélteti az összes lábán, hogy melyikről van szó (analóg bemenetek, digitális ki- és bemenetek, PWM), míg a micro:bit csak a fő lábait hangsúlyozza ki (P0, P1, P2, 3V, GND), a többi láb kiosztásához pedig már kutakodni kell, hogy melyik barázda melyik láb. Ezt pedig a 2.1. ábra szemlélteti.

Viszont ami előnye a micro:bit-nek az Arduino UNO-val szemben, hogy rendelkezik beépített Bluetooth-al és rádióval is. Ezek által könnyedén lehet kommunikálni és nem kell külön modulokhoz kötni, hogy képes legyen ezekre a funkciókra mint az Arduino esetén [9].

Eltérnek még abban, hogy míg a micro:bit csak 3,3V jelszintet használ, addig az Arduino 5V-t. Ez sok esetben nem okoz problémát, viszont vannak olyan eszközök, amelyek vagy 5V vagy csak 3,3V jelszintet igényelnek, és ilyenkor oda kell figyelni arra, hogy akár egy szintemelő használatával kompatibilissé tegyük őket egymással, nehogy tönkremenjenek [10].

Összességében úgy gondolom, hogy az Arduino UNO inkább kedvező lehet a haladók számára, hiszen igényel némi alaptudást a használata, míg a micro:bit előnyösebb lehet azok számára, akik még nem foglalkoztak robotikával ezelőtt, de szeretnék megismerni, hogy hogyan működnek különböző szenzorok. Magyarán a micro:bit jó választás lehet a tanulmányok elkezdéséhez és a robotikával való megismerkedéshez, illetve remek átvezetés olyan haladó eszközök használatára mint az Arduino mikrokontrollerek.




2.4. ábra. Arduino UNO felépítése [10].

## 2.2. Programozása

A micro:bit programozására meglehetősen sok lehetőség van. Ahogyan már említettem, az az opció is nyitott, hogy akár az Arduino IDE környezetben programozhassuk, de vannak saját környezetek is. Az egyik ilyen – amit én is használtam – a Microsoft

MakeCode editor, amelyben pedig az alább megemlített módokon lehet kódot írni. Továbbá lehetőség van még a Python editort, a Scracth-et, illetve a telefonos alkalmazást is használni [11].

### 2.2.1. Blokkprogramozás

A blokkok összetett kódrészek, ezek lehetnek ciklusok, elágazások, változó implementálások, stb. Ezeket egymáshoz kapcsolásával tudjuk felépíteni a kódot. Roppant egyszerű a használata és a megértése. Ugyanakkor megvan a lehetőség ezeknek a kódoknak a Python-ra vagy JavaScript-re való fordítása, ott pedig megláthatjuk, hogy pontosan miket is tartalmaznak a blokkok [12].

### 2.2.2. JavaScript

Pontosabban TypeScript-ben kódolhatunk, ami statikusan típusos, objektum-orientált script nyelv, a JavaScript bővítése. Egyik előnye, hogy nem kell plusz könyvtárakat és elemeket behivatkozni a kód működtetéséhez. Másik előny ezzel a nyelvvel dolgozni pedig, hogy ugyancsak nyílt forráskódú, illetve operációs rendszer független [13].

### 2.2.3. Python

A Python egy szintén objektum orientált, emellett magas szintű, interaktív és általános célú nyelv, amelyet Guido van Rossum holland programozó kezdeményezett. Egyszerű szintaxisával az egyszerű olvashatóságra törekszik. Ingyen és korlátozás nélkül használható, Windows, Linux és macOS operáció rendszereken egyaránt [14].

## 2.3. Előnyei és hátrányai

A kutatás során előjött néhány eset, amiknél úgy éreztem fontos, hogy meg legyenek említve, hiszen sokat segíthetnek majd a későbbiekben azoknak, akik emellett a mikrokontroller mellett döntenek.

Egyik előnye véleményem szerint az, hogy több nyelven is programozható. A micro:bit már fentebb említett, saját ”MakeCode„ nevezetű platformján könnyedén meg lehet írni rá a kódot és sok kiegészítő funkció jár vele. Másik előnye pedig, a már eleve beépített szenzoraiban rejlik. Sok bekötés és programozás alól menti fel a használóját, ugyanakkor új lehetőségeket nyit meg előtte. Könnyedén lehet olyan projekteket létrehozni, ahol mindenképpen szeretnénk kipróbálni több szenzort is. Gondok nélkül alkalmazhatjuk őket egy időben. A MakeCode platformról a 15. oldalon olvashat többet.

Ugyanakkor hátránya ebből az előnyből ered. Hiszen a mikrokontrollerbe beépített szenzorok már felhasználnak egyes pineket, vagy meghatározott szerepük van (például a P19 és P20 pinek kifejezetten SCL és SDA jelek fogadására/adására kompatibilisek) ezért, ha további szenzorokat szeretnénk hozzákötni, figyelnünk kell arra, hogy hova is kössük be őket. Gyakran előfordul például, hogyha egy szenzort olyan pinre kötünk, amelyre már az egyik LED oszlop van kötve, akkor ha mind a két szenzort (a mikrokontroller saját LED-jét, illetve egy másik tetszőleges szenzort) egyszerre használunk, abban az esetben a LED-ek pislákolni fognak. Személy szerint ezt akkor tapasztaltam, amikor hőmérőt kötöttem egy olyan pinre, amelyhez már egy LED oszlop volt kiosztva, és amikor a mért értéket LED-eken szerettem volna megjeleníteni, pislákoltak a kiíratás közben.

Ennek a hátránynak a megoldása az, hogy megkeressük azokat a pineket, amelyekre nincs az adott, vagy semmilyen beépített szenzor kötve, és azokra kötjük be a kívánt szenzorokat. Amennyiben pedig nem használjuk fel azokat a beépített szenzorokat, amelyek az adott pinekre vannak kötve, akkor a probléma nem is fordul elő. Ezért is fontos ismerni a pinek kiosztását, ami a 2.1. ábrán látható.

# 3. fejezet

# Felhasznált eszközök

## 3.1. Felhasznált hardverelemek

A projekt hardveres részeihez a lentebb említésre kerülő szenzorokon kívül még breadboard-ot (szerelőlap), jumper kábeleket (male és female egyaránt) és expansion board-ot [1] használtam fel.

### 3.1.1. Gomb

Gombok alatt a pluszban bekötött és beépített gombokat is értem. Mivel egy fő mikrokontrollerről irányítom a másik kettőt, így úgy gondoltam, hogy több fog kelleni hozzá, mivel az alap két gomb kevés lenne minden funkció kiszolgálására. Ezért hozzáadtam továbbiakban még két egyszerű nyomógombot. A működési elvük egy és ugyanaz: a gomb megnyomása hatására fog történni valami. Külön kiterjesztést nem igényel, egyszerűen a csatlakozó lábak közül választjuk ki azokat a portokat, amelyekre be lettek kötve. Bekötéséhez pedig egy 10 kΩ-os ellenállás fog még kelleni, ami pedig a 4.1. ábrán látható [15].

### 3.1.2. LCD kijelző

A kijelzőt annak érdekében tettem hozzá a projekthez, hogy ezáltal a vezérlő micro:bit lehet egyfajta panel/felület, amelyről láthatjuk mit csinálunk. Ez jelez vissza a felhasználónak, hogy éppen melyik mikrokontrollerrel állunk kapcsolatban, vagy az éppen kapott információkat nézhetünk meg rajta.

Konkrétabban pedig egy 16x2-es LCD kijelző, I2C modullal felszerelve, ami könnyebbé teszi a kommunikációt a kijelző és mikrokontroller között (csak egy VCC, GND, SDA

[1] Bővítőkártya, amely lehetővé teszi a mikrokontroller összes csatlakozójához való hozzáférést. Kétféle módon: tüskesor vagy hüvelysor segítségével, amik male vagy female jumper kábelekkel kompatibilisek [16].

és SCL port szükséges) [17]. Erre a kijelzőre pedig található kiterjesztés a micro:bit felületén a ”I2C„ kulcsszót használva.

Sok esetben a beépített LED-eket használják információk kiíratására, de mivel ott több vagy hosszabb szövegeket már sok idő megjeleníteni, így jobb ötletnek láttam egy külön kijelző bevezetését erre a célra. A 4.1. ábrán pedig látható, hogyan kötöttem be [18].

### 3.1.3. Hőmérő

Hőmérő ugyan megtalálható a beépített szenzorai között a micro:bitnek, de sajnos tapasztalataim alapján, mivel közel helyezkedik el az USB csatlakozóhoz, így az onnan érkező áram által termelt valamennyi hő befolyásolja a hőmérő mérésének értékét.

Ezért én inkább egy külső hőmérő szenzort használtam fel. Egész pontosan egy Dallas DS18B20, 1 vezetékes digitális hőmérő-érzékelőt. Ennek hőmérséklet tartománya -55°C és +125°C fok közé esik [19]. Ezáltal tapasztaltan jobb eredményeket kaptam vissza a mérések során, ami egy okos otthon környezetben elengedhetetlen feltétel lehet.

Ehhez a hőmérőhöz pedig a micro:bit rendelkezik kiterjesztéssel, amelyet a kódnál a ”Kiterjesztések„ fül megnyomása után, a ”temperature„/”temp„, ”dallas„ vagy ”dstemp„ kulcsszavak keresésével találhatunk meg és adhatunk hozzá. Ezek után pedig problémamentesen megírhatjuk rá a kódot. Bekötéséhez ide is szükség lesz egy 10 kΩ-os ellenállásra, ami pedig a 4.2. ábrán látható [20].

Kezdők részére persze tökéletes lehet a beépített szenzor, viszont a haladók számára, vagy akik egy pontosabb mérést szeretnének szerintem ez a fajta szenzor jó választás lehet.

### 3.1.4. RGB LED

A RGB LED-eknél az egyik legfontosabb dolog az, hogy tudjuk pontosan milyen LED-et is használunk. Ugyanis, vannak közös anódos (+) és közös katódosok (-). Ezeknél nem mindegy, hogy hogyan kötjük be őket, hiszen, ha a közös láb anód, akkor a VCC-re kell bekötnünk az adott lábat, viszont ha katód, akkor pedig a GND-re, a többi lábat pedig 220 Ω-os ellenállással az analóg portokra (PWM kimenet). Illetve a kódolásban is eltér a kettő, hiszen közös anód esetén a 0 érték lesz az, ami a maximális világítást jelenti, és 1023, ami pedig a kikapcsolt állapot, közös katód esetében pedig pont az ellenkezője. Ebben a projektben én egy közös anódosat használtam, aminek a bekötése ugyancsak a 4.2. ábrán látható [21].

### 3.1.5. Szilárdtest relé

A szilárdtest relé egy olyan eszköz, amely segítségével elektronikus áramot tudunk vezérelni. Nincs mozgó alkatrésze, ezért tartósabbak és gyorsabbak, mint az elektromechanikus társaik. Ezen előnyök lévén képesek a gyors kapcsolásokra, amelyeket az áram áramoltatása és blokkolására szánt vezérlőjelek alkalmazásával vagy eltüntetésével vagyunk képesek végrehajtani. Másik fontos előnye még, hogy képes nagy teljesítményű áramot és feszültséget kapcsolni. Ezért alkalmazása tökéletes világításhoz, illetve hőszabályozáshoz is egyaránt [22].

Ebben a projektben, én egy ANLY ASR-03DA típusú szilárdtest relét használtam és egy hagyományos villanykörtéhez és – az azt működtető – áramellátáshoz kötöttem be. Ezzel szemléltetve azt, hogy a világítás remekül elkészíthető és kezelhető ebben a formában. De bármilyen eszközhöz felhasználható, amely 220V - 230V-os árammal működik (ezáltal a hőszabályozáshoz is felhasználhatóak hűtésre és fűtésre alkalmas eszközök). Ennek bekötése a 4.3. ábrán látható.

### 3.1.6. Fotoellenállás

Csakugyan, mint ahogyan a hőmérőnél is, itt is külsőleg hozzáadott, két kivezetéssel rendelkező fotoellenállást használtam, annak ellenére, hogy a micro:bit szintén rendelkezik beépített fénymérési funkcióval és szenzorral. Itt is egy biztosabb mérést kaptam vissza a beérkező fényről. Kiterjesztés ehhez sem szükséges, hiszen szintén irányítható bármely olvasható portról (a legalkalmasabb olvasható portok a P0, P1 és P2). Ennek bekötéséhez szintén egy 10 kΩ-os ellenállásra is szükség lesz, ami a 4.3 árbán látható [23].

## 3.2. Felhasznált szoftverek

### 3.2.1. Microsoft MakeCode

A ”Microsoft MakeCode„, ami egy nyílt forráskódú [2] szerkesztő. Ennek egyik legjobb tulajdonsága, hogy 3 nyelven is lehet kódot írni rá. Legelterjedtebb és legegyszerűbb a Blokk, de használhatóak még Python és JavaScript nyelvek. Az oldalon könnyedén meg lehet találni mindent amire szükség lehet: kód blokkok, kiterjesztések, szimulátor. Ezek által egyszerűen lehet kódot írni, bővíteni azt a hozzákötött szenzorok csomagjaival, majd az elkészült projektet le is szimulálja nekünk az oldal, így látjuk mi történik pontosan, mielőtt azt feltöltenénk a micro:bitre [24].

[2] A nyílt forráskódok (open source) olyan szoftverek, amelyeket szabadon fel lehet használni, másolni, módosítani, terjeszteni, stb. [25].

### 3.2.2. Fritzing

A Fritzing is egy szintén nyílt forráskódú szoftver [3]. Eredetileg a németországi potsdami Alkalmazott Tudományok Egyetemének kutatói fejlesztették ki, de azóta bárki szabadon felhasználhatja. A szoftver legújabb verziói fizetősek (adományként kérik), de korábbi verziók ingyenesen is letölthetőek. Ugyanakkor több platformos, egyazon által elérhető macOS, Windows és Linux operációs rendszereken is.

Lényege, hogy áramkör prototípusokat, hardveres projekteket lehet a segítségével megtervezni. Rengeteg fajtájú és típusú szenzor, mikrokontroller és egyéb hasznos eszköz megtalálható a felületen, amikre itt is lehetőség van kódot írni, illetve szimulálni azt. Egyik előnye, hogy bővíthető, hiszen a nyílt forráskód bárki számára lehetővé teszi, hogy további eszközöket adjon hozzá, amik esetleg nincsenek benne az alap kínálatban. A nagy fejlesztői közösség miatt, pedig biztosítva van az, hogy naprakész maradjon a szoftver és bekerüljenek az újabb eszközök is. Másik előnye még, hogy a tervezésnél több opciónk is van, készíthetünk akár sematikus [4] vagy CAD [5] ábrákat is, amelyek által jól átlátható dokumentációkat tudunk készíteni hozzá [26, 27].

Remekül alkalmazható akár tanórákhoz, hiszen nem feltétlenül szükséges, hogy minden hardver a rendelkezésünkre álljon, illetve teljes mértékben lecsökkenti az esetleges baleseteket, amik keletkezhetnének azáltal, hogy rosszul kötünk be valamit, illetve, ha rosszul írtunk volna meg valamilyen kódot, akkor ebben az esetben a szenzorok nem fognak hibásan működni, hanem visszajelzést ad a szimuláció.

### 3.2.3. PlantUml

A PlantUML egy hatékony, alkalmazkodó, ingyenes és szintén nyílt forráskódú web szerver. Egyszerűen és gyorsan lehet a segítségével UML [6] diagramot készíteni szöveges leírással. Inkább rajzeszközként szolgál mintsem modellezőként. A diagramokat a megadott szöveg alapján automatizáltan generálja le az intelligens elrendezési algoritmusait felhasználva. Ezáltal tetszetős elrendezést nyújt és könnyedén lehet szerkeszteni, frissíteni kézi igazítások nélkül. Előnye még, hogy magas fokon testre szabhatóak a vizuális elrendezések [28]. A 4.4. ábrához pedig ezt az eszközt használtam fel.

[3] Más néven alkalmazás, program.

[4] Csak főbb vonalaiban ábrázolt [29].

[5] Az angol computer-aided design rövidítése. Számítógéppel tervezett fizikai objektum grafikus ábrázolása [30].

[6] Az UML (Unified Modelling Language) egy vizuális modellező nyelv [31].

# 4. fejezet

# Az összetett projekt

## 4.1. Módszer leírása

Az elképzelés az volt, hogy ne csak egy mikrokontrollert használjak fel a projekthez, hanem többet is. Hiszen ha mindent egy helyről akarnék irányítani és kezelni, egy idő után bonyolulttá válhat akár a megértése, akár az elkészítése. Ez a kód megírásának szempontjából is barátságosabb megoldás, hiszen nem több száz soros kódot kell írni és majd elolvasni hozzá, hanem könnyebben lehet részekre osztani a modulokat [1].

Így született meg a jelenleg 3 mikrokontrollerből álló projekt. Mindegyik külön feladatokat lát el:

Vezérlő – a szerkezet közepe, ez irányítja a többi hozzákapcsolt mikrokontrollert, és ehhez térnek vissza a mért adatok, illetve innen kapcsolhatjuk automata vagy manuális üzemmódba a mikrokontrollereket.

Hőszabályozó – feladata, hogy automatizált állapotban mérje a környezetének hőmérsékletét és ennek megfelelően kapcsoljon be vagy éppen ki, fűtést vagy hűtést.

Világítás kezelő – automata üzemmódban az a feladata, hogy mérje környezetében a fényerősséget és annak megfelelően kapcsolja fel, vagy éppen le a lámpákat amik hozzá vannak kötve.

### 4.1.1. Bővíthetőség

A fentiekben leírt módszer pedig könnyedén elérhetővé teszi a bővítés lehetőségét. Hiszen minden mikrokontroller külön feladatot lát el, így ha plusz funkciókat szeretnénk hozzáadni a projekthez, csak egy új mikrokontroller és az ahhoz kellő szenzorok és osztályok hozzáadására lesz csak szükség. Ez által nem kell nagy átalakításokat végrehajtani a kódokban sem, hiszen egyedül a vezérlő kommunikál majd az újonnan hozzáadott elemmel.

[1] A program egy önálló része, ami valamilyen feladatot hajt végre [32].

A jelenlegi hűtés/fűtés és világítás kezelés mellé praktikus lehet akár egy csengő funkció beszerelése, ami kifejezetten egyszerűen megvalósítható, hiszen plusz szenzorok hozzáadására sincs szükség (beépített gombok, illetve a micro:bit v2 verzió hangszórót is tartalmaz). Illetve, ha még kényelmesebbé szeretnénk tenni otthonunkat, fejleszthetünk rá telefonos alkalmazást is, ami a micro:bit Bluetooth kapcsolatának segítségével valósulhat meg.

## 4.2. Működési elv leírása

Mint ahogyan fentebb már említésre került, illetve a bekötési ábrák és kódok alapján látható, hogy 3 mikrokontroller működik egyszerre és önállóan. Mindegyiknek megvan a saját feladata, amit automatikusan végeznek, de ha kedvünk tartja mi magunk is irányíthatjuk őket a vezérlő mikrokontroller segítségével.

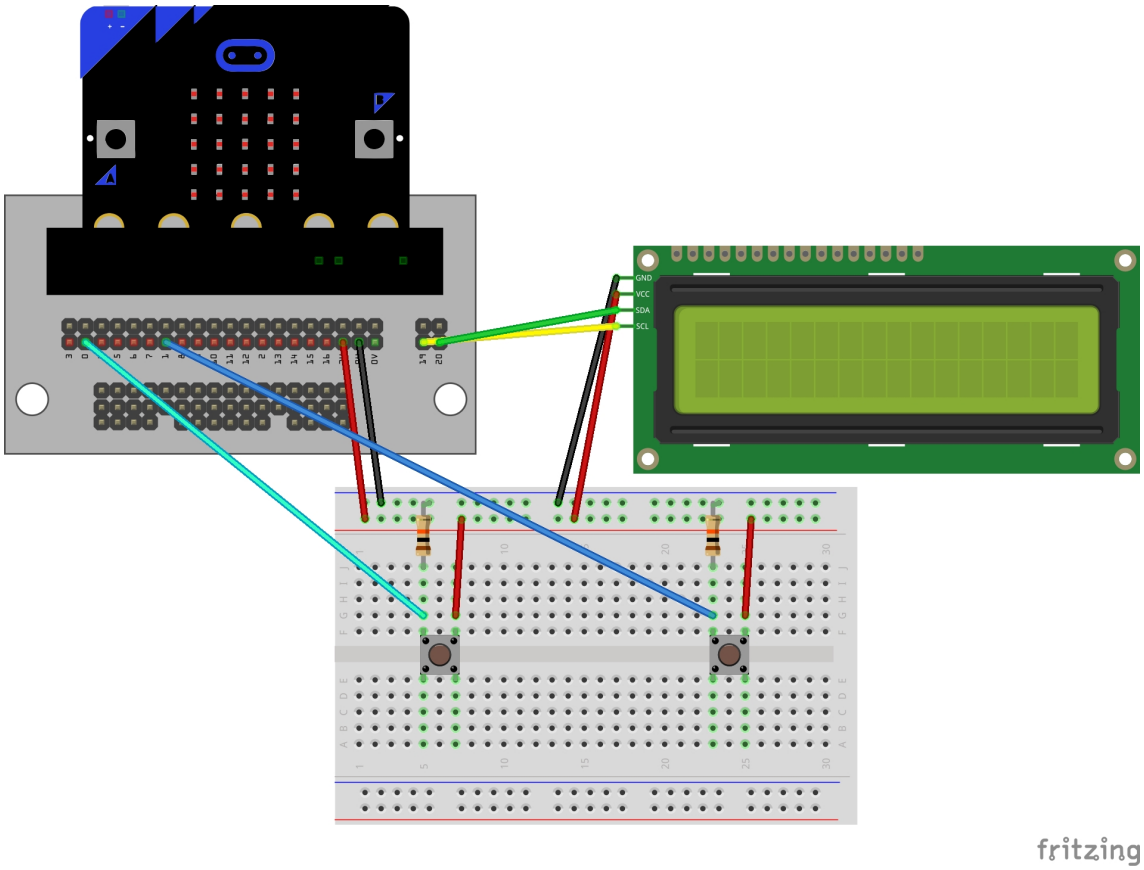
A 3 felület között rádiós kapcsolat teszi lehetővé a kommunikációt. Közös rádiócsoportot használnak, amelyen keresztül fogadják és küldik egymásnak az adatokat. Az, hogy mit és milyen esetben küldjenek egymásnak a rádión keresztül, a műveleti, illetve a vezérlés osztályok kódjaiban szemlélhetünk meg a 19. oldaltól kezdődően.

### 4.2.1. Vezérlő mikrokontroller

A vezérlő mikrokontroller alap helyzetben egy ”Udvozollek„ feliratot jelenít meg a kijelzőn, arra várva, hogy mit szeretnénk csinálni a továbbiakban (a 4.1. kód 32. sor). A P0 vagy P1 lábára bekötött gombot megnyomva egy ”radio_jel„ nevű segédváltozó értékét beállítjuk 1-re vagy éppen 2-re. Ezáltal követni tudjuk majd, hogy éppen melyik mikrokontrollerrel veszi fel a kapcsolatot. Ez azért fontos, mivel közös rádiócsoportot használnak (ha különböző csoportot használnának a vezérlő addig nem tudna jelet kapni más mikrokontrollertől, ameddig benne van már egy meglévő kommunikációba, pl.: nem fogadná a csengőtől érkező jelet, mert éppen a hőszabályozóval van kapcsolatban), így nem zavar be semmilyen elküldendő vagy fogadandó adat a kommunikációba.

A P0 lábon lévő gombbal a hőszabályozóval tudjuk felvenni a kapcsolatot. A gomb megnyomása után küldeni fog egy jelet, amit a hőszabályozó feldolgoz, és visszaküldi az ott mért aktuális °C fokot, amit az LCD kijelzőn meg is jelenít. A manuális vezérlésére a mikrokontroller beépített ”A„ és ”B„ gombjával van lehetőség. Az ”A„ gomb megnyomásának hatására a fűtést tudjuk ki- és bekapcsolni. A ”B„ gombbal pedig a hűtést. A két gomb együttes megnyomásával pedig visszaállítjuk manuális vezérlésről automata üzemmódba.

A P1 lábhoz tartozó gomb pedig a lámpa irányításáért felel. Az elv ugyanúgy működik itt is. A rádiós kapcsolat létrejötte során visszakapjuk azt, hogy sötét vagy éppen világos van-e. Az ”A„ gomb megnyomásának hatására felkapcsoljuk a lámpát, a ”B„




4.1. ábra. Vezérlő mikrokontroller bekötési ábrája [33, 34].

gombbal pedig lekapcsoljuk. A két gomb együttes megnyomása itt is az automatikus működés visszaállítására szolgál. Ezek a manuális irányítások pedig addig maradnak érvényesek ameddig automata állapotra nem váltjuk.

Mivel vezérlőnek v2-es verziójú mikrokontrollert használtam, így ennek alaphelyzetbe állításához a logó megérintése beépített funkciót használtam fel. Azaz, ha már nem akarunk semmit sem irányítani, akkor a logó megérintésével visszaválthatunk a várakozó állapothoz.

kód 4.1. A vezérlés osztály kódja.

```
1  class Vezerles {
2      // osztályok és segédváltozó implementálása
3      private v: Vilagitas
4      private ho: FutesHutes
5      private radio_jel: number
6
7      // a Vilagitas és FutesHutes osztály átadása a
           ↪ konstruktorban
8      constructor(vilagitas: Vilagitas, ho: FutesHutes)
           ↪ {
9          this.v = vilagitas
10         this.ho = ho
```

```
11          this.radio_jel = 0
12      }
13
14      // getterek és setterek
15      getRadio() {
16          return this.radio_jel
17      }
18
19      setRadio(ertek: number) {
20          this.radio_jel = ertek
21      }
22
23      // saját metódusok kifejtése
24      auto(): void {
25          I2C_LCD1602.clear()
26          I2C_LCD1602.ShowString("Udvozollek", 0, 0)
27      }
28
29      Start(): void {
30          // beépített és saját eljárások felhasználása
31          I2C_LCD1602.LcdInit(39)
32          radio.setGroup(1)
33          this.auto()
34
35          input.onPinPressed(TouchPin.P0, function () {
36              I2C_LCD1602.clear()
37              I2C_LCD1602.ShowString("Homerseklet", 0,
                    ↪ 0)
38              radio.sendString("A")
39              this.setRadio(1)
40          })
41
42          input.onPinPressed(TouchPin.P1, function () {
43              I2C_LCD1602.clear()
44              I2C_LCD1602.ShowString("Feny", 0, 0)
45              radio.sendString("B")
46              this.setRadio(2)
47          })
48
49          input.onButtonPressed(Button.A, function () {
50              if (this.getRadio() == 1) {
51                  this.ho.buttonA()
52              } else if (this.getRadio() == 2) {
53                  this.v.be()
54              }
55          })
56
```

```
57          input.onButtonPressed(Button.B, function () {
58              if (this.getRadio() == 1) {
59                  this.ho.buttonB()
60              } else if (this.getRadio() == 2) {
61                  this.v.ki()
62              }
63          })
64
65          input.onButtonPressed(Button.AB, function () {
66              if (this.getRadio() == 1) {
67                  this.ho.auto()
68              } else if (this.getRadio() == 2) {
69                  this.v.auto()
70              }
71          })
72
73          radio.onReceivedValue(function (name, value) {
74              if (name == "ho") {
75                  I2C_LCD1602.ShowNumber(value, 0, 1)
76                  I2C_LCD1602.ShowString("Celsius", 5,
                        ↪ 1)
77              } else if (name == "feny") {
78                  if (value < 300) {
79                      I2C_LCD1602.ShowString("Sotet van
                            ↪ ", 0, 1)
80                  } else {
81                      I2C_LCD1602.ShowString("Vilagos
                            ↪ van", 0, 1)
82                  }
83              }
84          })
85
86          input.onLogoEvent(TouchButtonEvent.Pressed,
                ↪ function () {
87              this.auto()
88          })
89      }
90  }
```

Az egyszerűbb olvashatósághoz és megértéshez úgy készítettem el a kódot, hogy több osztályba szerveztem ki a metódusokat és változókat, illetve az ismétlődő eljárásokat külön interface-be szerveztem, amit csak elég felülírni az aktuális osztályban, így könnyebb megoldani a bővítési kérdést is.

kód 4.2. A műveletek interface kódja.

```
1  interface Muveletek {
2      // a többször felhasznált metódusokat kigyűjtjük
```

```
        ↪ interface-be
    be(): void
    ki(): void
    auto(): void
}
```

kód 4.3. A fűtés-hűtés osztály kódja.

```
class FutesHutes implements Muveletek{
    // segédváltozók a hűtés és fűtés, illetve be- és
        ↪ kikapcsolt állapotok szétválasztására
    private be_ki: boolean
    private fut_hut: string

    constructor() {
        this.be_ki = false
        this.fut_hut = ""
    }

    // getterek és setterek
    getBeKi() {
        return this.be_ki
    }

    setBeKi(beki: boolean) {
        this.be_ki = beki
    }

    getFut_Hut(): string {
        return this.fut_hut
    }

    setFut_Hut(ertek: string): void {
        this.fut_hut = ertek
    }

     // kifejtjük a Muveletek interface, illetve saját
        ↪ metódusait
    be(): void {
        if (this.getFut_Hut() == "FUT") {
            this.setBeKi(true)
            radio.sendString("FUT")
            I2C_LCD1602.clear()
            I2C_LCD1602.ShowString("Futes be", 0, 0)
        } else if (this.getFut_Hut() == "HUT") {
            this.setBeKi(true)
            radio.sendString("HUT")
            I2C_LCD1602.clear()
```

```
39              I2C_LCD1602.ShowString("Hutes be", 0, 0)
40          }
41      }
42
43      ki(): void {
44          this.setBeKi(false)
45          radio.sendString("NEM_FUT_HUT")
46          I2C_LCD1602.clear()
47          I2C_LCD1602.ShowString("Futes/Hutes ki", 0, 0)
48      }
49
50      auto(): void {
51          this.setBeKi(false)
52          radio.sendString("HO_AUTO")
53          I2C_LCD1602.clear()
54          I2C_LCD1602.ShowString("Futes/hutes", 0, 0)
55          I2C_LCD1602.ShowString("auto", 0, 1)
56      }
57
58      buttonA(): void {
59          if (this.getBeKi() == false) {
60              this.setFut_Hut("FUT")
61              this.be()
62          } else if (this.getBeKi() == true) {
63              this.ki()
64          }
65      }
66
67      buttonB(): void {
68          if (this.getBeKi() == false) {
69              this.setFut_Hut("HUT")
70              this.be()
71          } else if (this.getBeKi() == true) {
72              this.ki()
73          }
74      }
75  }
```

kód 4.4. A vilagitas osztály kódja.

```
1   class Vilagitas implements Muveletek{
2       // kifejtjük a Muveletek interface metódusait
3       be(): void {
4           radio.sendString("FEL")
5           I2C_LCD1602.clear()
6           I2C_LCD1602.ShowString("Lampa fel", 0, 0)
7       }
8
```

```
9       ki(): void {
10          radio.sendString("LE")
11          I2C_LCD1602.clear()
12          I2C_LCD1602.ShowString("Lampa le", 0, 0)
13      }
14
15      auto(): void {
16          radio.sendString("FENY_AUTO")
17          I2C_LCD1602.clear()
18          I2C_LCD1602.ShowString("Lampa auto", 0, 0)
19      }
20  }
```

kód 4.5. A main osztály kódja.

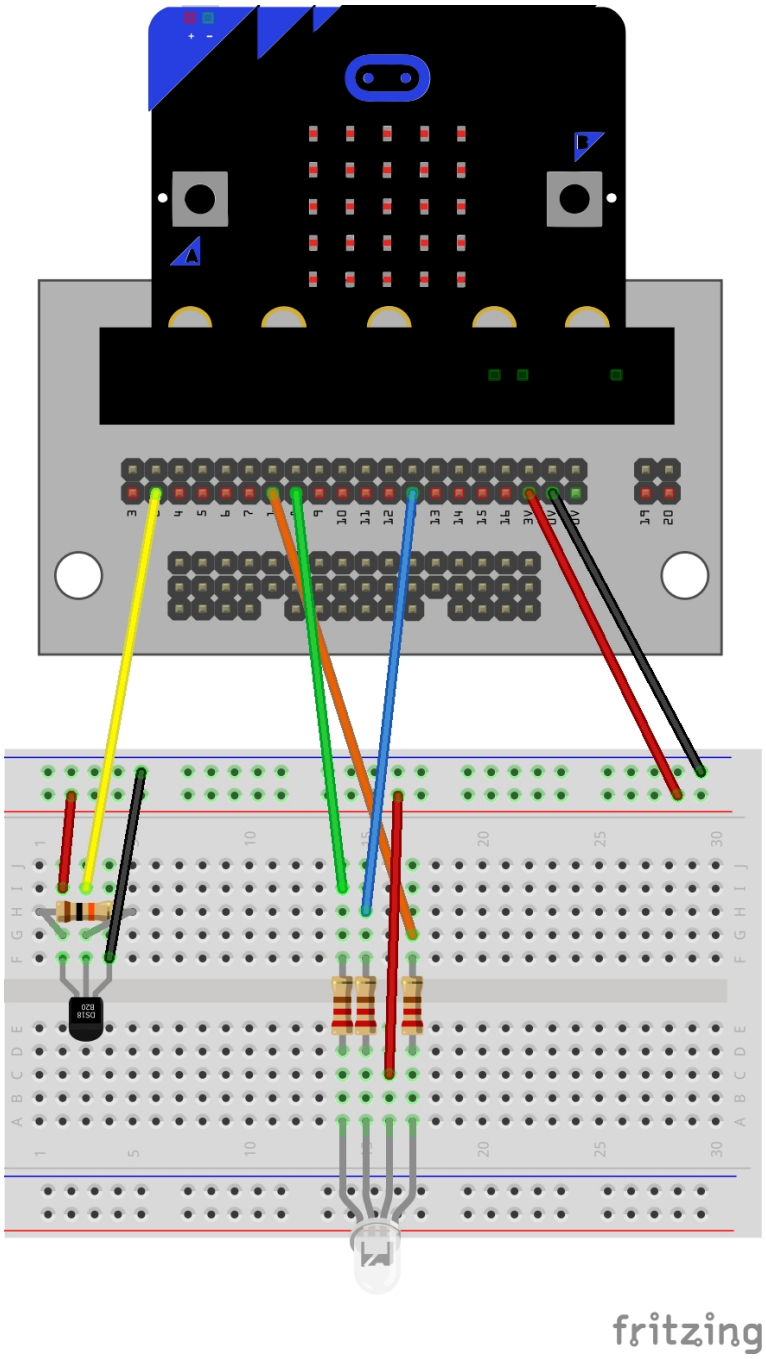
```
1  let vil = new Vilagitas()
2  let ho = new FutesHutes()
3  let vez = new Vezerles(vil, ho)
4  vez.Start()
```

A main oldalon pedig összesen ennyi dologra lesz szükségünk a beüzemeltetéshez. Példányosítjuk az osztályokat, átadjuk a paramétereket és meghívjuk a Start() metódust.

### 4.2.2. Hőszabályozó mikrokontroller

Alapjáraton a feladata, hogy mérje a környezete hőmérsékletét, és kiírja annak értékét saját beépített LED-jein, a bekötött LED pedig színekkel jelezze milyen funkció zajlik éppen. Mivel ehhez a modulhoz egyelőre csak RGB LED-et használtam a fűtés és hűtés prezentálásához, így azokat fogja irányítani. Amennyiben a hőmérséklet 21°C fok alá esik, bekapcsolja a ”fűtést„, ami az RGB LED piros színnel való világítása jelzi. 27°C fok elérését követően kékre vált a LED, jelezve hogy bekapcsolt a ”hűtés„ funkció. Amennyiben a hőmérséklet a két határ közé esik, úgy a LED zölden világít, jelezve, hogy automata állapotban működik, és kellemes az idő a lakásban.

Amint a rádión keresztül érkezik a jel, hogy manuálisan kívánjuk irányítani, kialszanak mind a beépített, mind a bekötött LED-ek. A kapott kulcsszó fogja eldönteni, milyen eljárás is menjen végbe, egészen addig, amíg más utasítás nem érkezik. Amennyiben a ”FUT„ jel érkezik, a LED narancssárgás fénnyel fog világítani, jelezve, hogy ez már a manuálisan bekapcsolt fűtés folyamata lesz. ”HUT„ kulcsszó esetén pedig egy kékes-zöldes fény fog égni a rendes kék helyett, ugyancsak a manuális irányítást jelezve. Ezen kulcsszavak fogadásakor egy ”hut_fut_auto„ segédváltozó értékét módosítja manuálisra. ”HO_AUTO„ szó fogadásánál az egyetlen amit tesz, hogy az előbb említett segédváltozót automatára állítja, ezzel visszaállítva őt az automata érzékelésre és működésre.




4.2. ábra. A hőmérő mikrokontroller bekötési ábrája [33].

kód 4.6. A műveletek osztály kódja.

```
1  class Muveletek {
2      // szükséges osztály implementálása
3      private h: Homero
4
5      // a konstruktor bemenete az adott Hőmérő osztály
           ↪ lesz
6      constructor(homero: Homero) {
7          this.h = homero
```

```
8       }
9
10      // metódusok kifejtése
11      fut(): void {
12          basic.pause(100)
13          pins.analogWritePin(AnalogPin.P1, 100)
14          pins.analogWritePin(AnalogPin.P2, 1023)
15          pins.analogWritePin(AnalogPin.P8, 900)
16      }
17
18      nem_fut_hut(): void {
19          basic.pause(100)
20          pins.analogWritePin(AnalogPin.P1, 1023)
21          pins.analogWritePin(AnalogPin.P2, 1023)
22          pins.analogWritePin(AnalogPin.P8, 1023)
23      }
24
25      hut(): void {
26          basic.pause(100)
27          pins.analogWritePin(AnalogPin.P1, 1023)
28          pins.analogWritePin(AnalogPin.P2, 100)
29          pins.analogWritePin(AnalogPin.P8, 900)
30      }
31
32      auto(): void {
33          this.h.setHo(dstemp.celsius(DigitalPin.P0))
34          if (this.h.getHo() > 27) {
35              pins.analogWritePin(AnalogPin.P1, 1023)
36              pins.analogWritePin(AnalogPin.P2, 0)
37              pins.analogWritePin(AnalogPin.P8, 1023)
38          } else if (this.h.getHo() < 21) {
39              pins.analogWritePin(AnalogPin.P1, 0)
40              pins.analogWritePin(AnalogPin.P2, 1023)
41              pins.analogWritePin(AnalogPin.P8, 1023)
42          } else {
43              pins.analogWritePin(AnalogPin.P1, 1023)
44              pins.analogWritePin(AnalogPin.P2, 1023)
45              pins.analogWritePin(AnalogPin.P8, 0)
46          }
47          basic.showNumber(this.h.getHo())
48      }
49
50      Start(): void {
51          // saját és beépített metódusok felhasználása
52          radio.setGroup(1)
53
54          radio.onReceivedString(function (
```

```
            ↪ receivedString) {
55              if (receivedString == "A") {
56                  radio.sendValue("ho", this.h.getHo())
57              }
58              else if (receivedString == "FUT") {
59                  this.h.setAuto("MANUAL")
60                  this.fut()
61              } else if (receivedString == "NEM_FUT_HUT
                    ↪ ") {
62                  this.h.setAuto("MANUAL")
63                  this.nem_fut_hut()
64              } else if (receivedString == "HUT") {
65                  this.h.setAuto("MANUAL")
66                  this.hut()
67              }
68              else if (receivedString == "HO_AUTO") {
69                  this.h.setAuto("AUTO")
70              }
71          })
72
73          basic.forever(function () {
74              if (this.h.getAuto() == "AUTO") {
75                  this.auto()
76              }
77          })
78      }
79  }
```

A változókat egyszerűsítésképpen itt is kirendeztem külön osztályba.

kód 4.7. A hőmérő osztály kódja.

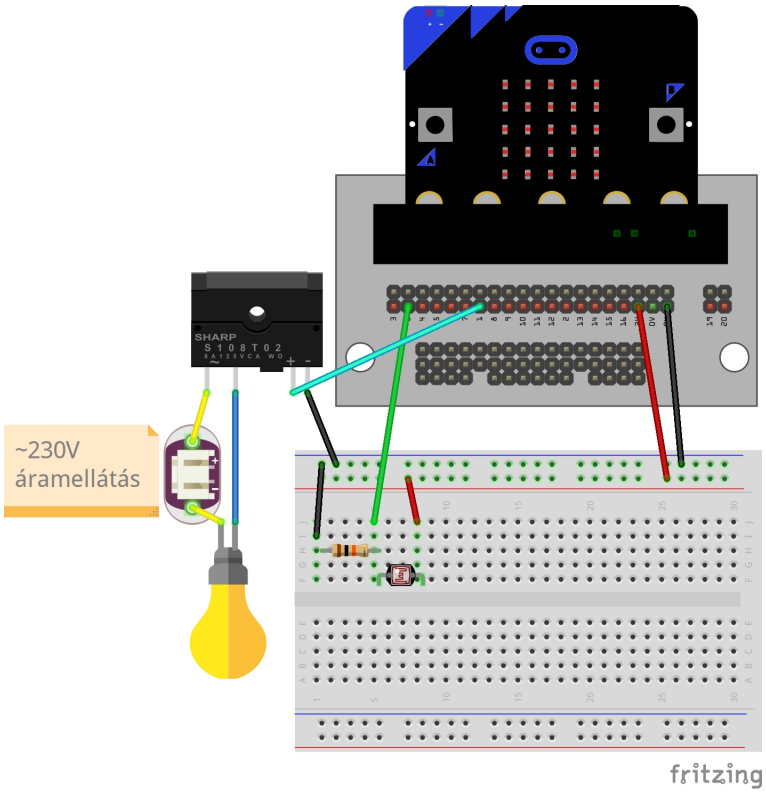
```
1   class Homero {
2       // segédváltozók implementálása
3       private ho: number
4       private hut_fut_auto: string
5
6       // a konstruktorba beállítjuk az alap értékeket
7       constructor() {
8           this.hut_fut_auto = "AUTO"
9           this.ho = 0
10      }
11
12      // getterek és setterek
13
14      getAuto(): string {
15          return this.hut_fut_auto
16      }
```

```
17
18      setAuto(ertek: string): void {
19          this.hut_fut_auto = ertek
20      }
21
22      getHo(): number {
23          return this.ho
24      }
25
26      setHo(szam: number): void {
27          this.ho = szam
28      }
29  }
```

kód 4.8. A main felület kódja.

```
1  let h = new Homero()
2  let m = new Muveletek(h)
3  m.Start()
```

### 4.2.3. Világítást kezelő mikrokontroller




4.3. ábra. A mikrokontrollerrel vezésrelt világítás bekötési ábrája [33, 35].

Az ő feladata, hogy mérje a beérkező fényt, és annak értékét írja ki a beépített LED-jeire, majd ennek megfelelően kapcsolja be vagy éppen ki a lámpát. Ennek eldöntésére

én egy egyszerű határérték vizsgálatot használtam a kódban, hiszen az könnyedén állítható tapasztalat és kívánság szerint. Így jelen állás szerint a lámpa 300 mért érték alatt fog felkapcsolni – úgy gondolom az megfelelően sötét – a felett pedig lekapcsolt állapotban marad.

Szintén a rádió által küldött jelek alapján vált automata és manuális állapot között. Ehhez pedig szintén társul egy ”le_fel_auto„ segédváltozó, ami segít az állapotok kezelésében. ”FEL„ jel esetén szintén kialszanak a beépített LED-ek, amiken addig a mért érték jelent meg és felkapcsolódik (vagy felkapcsolva marad, ha már eleve abban az állapotban volt) a lámpa, ”LE„ szó esetén pedig lekapcsolódik a lámpa. Az ”AUTO„ szó fogadásakor pedig szintúgy visszaváltó automata működésbe, és folytatja a fény mérését és annak kiíratását.

kód 4.9. A műveletek osztály kódja.

```
1  class Muveletek {
2      // szükséges osztály implementálása
3      private f: Fenymero
4
5      // a konstruktor bemenete az adott Fénymérő
           ↪ osztály lesz
6      constructor(fenymero: Fenymero) {
7          this.f = fenymero
8      }
9
10     // metódusok kifejtése
11     fel(): void {
12         basic.pause(100)
13         pins.digitalWritePin(DigitalPin.P1, 1)
14     }
15
16     le(): void {
17         basic.pause(100)
18         pins.digitalWritePin(DigitalPin.P1, 0)
19     }
20
21     auto(): void {
22         this.f.setFeny(pins.analogReadPin(AnalogPin.P0
               ↪ ))
23         pins.digitalWritePin(DigitalPin.P1, 0)
24         if (this.f.getFeny() < 300) {
25             pins.digitalWritePin(DigitalPin.P1, 1)
26         }
27         basic.showNumber(this.f.getFeny())
28     }
29
30     Start(): void {
```

```
31          // saját és beépített metódusok felhasználása
32          radio.setGroup(1)
33
34          radio.onReceivedString(function (
                ↪ receivedString) {
35              if (receivedString == "B") {
36                  radio.sendValue("feny", this.f.getFeny
                        ↪ ())
37              } else if (receivedString == "FEL") {
38                  this.f.setAuto("FEL")
39                  this.fel()
40              } else if (receivedString == "LE") {
41                  this.f.setAuto("LE")
42                  this.le()
43              } else if (receivedString == "FENY_AUTO")
                    ↪ {
44                  this.f.setAuto("AUTO")
45              }
46          })
47
48          basic.forever(function () {
49              if (this.f.getAuto() == "AUTO") {
50                  this.auto()
51              }
52          })
53      }
54  }
```

A kód felépítésében pedig az eddigi mintát követve, itt is külön osztály van a változók számára, illetve egy rövid és egyszerű main felület.

kód 4.10. A fénymérő osztály kódja.

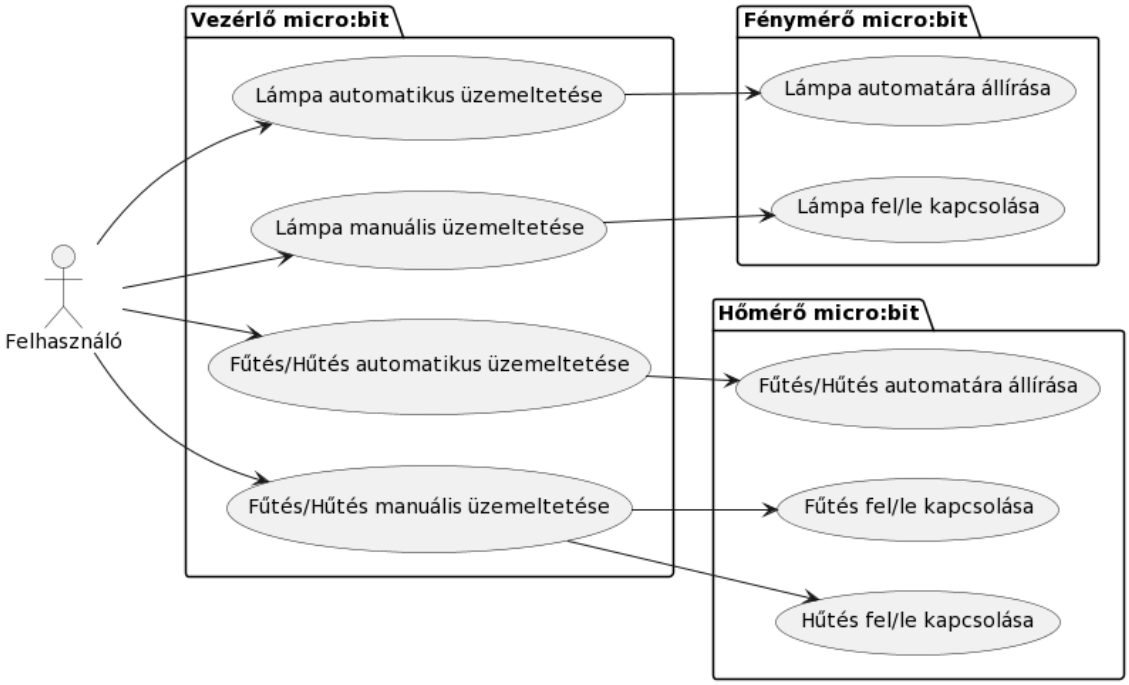
```
1   class Fenymero {
2       // segédváltozók implementálása
3       private feny: number
4       private le_fel_auto: string
5
6       // a konstruktorba beállítjuk az alap értékeket
7       constructor() {
8           this.le_fel_auto = "AUTO"
9           this.feny = 0
10      }
11
12      // getterek és setterek
13
14      getAuto(): string {
15          return this.le_fel_auto
```

```
16      }
17
18      setAuto(ertek: string): void {
19          this.le_fel_auto = ertek
20      }
21
22      getFeny(): number {
23          return this.feny
24      }
25
26      setFeny(szam: number): void {
27          this.feny = szam
28      }
29  }
```

kód 4.11. A main felülete kódja.

```
1  let f = new Fenymero()
2  let m = new Muveletek(f)
3  m.Start()
```

## 4.3. Felépítés




4.4. ábra. A rendszer működésének felépítése.

A felhasználó a vezérlő micro:bit beépített, illetve külsőleg hozzákötött gombjait használva felveszi a kapcsolatot, továbbá irányítja a többi micro:bitet. A vezérlő pedig

a lenyomott gombok alapján feldolgozza, hogy melyik funkciót kívánja használni a felhasználó. Ezt követően küldi ki a megfelelő rádiójelet, amelyeket a hőmérő és fénymérő micro:bit dolgoz fel, és attól függően működnek a továbbiakban.

# 5. fejezet

# Tesztelés

## 5.1. Manuális tesztelés

A különböző használtai eseteket egy táblázatban szedtem össze, és összehasonlítottam, hogy az elvárt működéshez képest hogyan is működik a rendszer. Az eredmények és leírások az 5.1. ábrán láthatóak.

## 5.2. Egység tesztek

A szenzorok és rádiós kapcsolat teszteléséhez egyszerű Blokk kódolást [36] alkalmaztam, mivel ezzel a módszerrel lehet a legkönnyebben kipróbálni és prezentálni a szenzorok helyes működését. Ezeket a kis teszteket pedig érdemes végig csinálni – sőt ezekkel a lépésekkel célszerű elkezdeni a projektet –, hiszen ezáltal meggyőződhetünk róla, hogy szenzoraink jól működnek és már az elején kizárhatunk rengeteg hibát.

### 5.2.1. Fotorezisztor

A fotorezisztor mérési értékét könnyen ellenőrizhetjük. Egyszerűen csak létrehozunk induláskor egy változót (amit a ”Változók„ fülnél tehetünk meg), majd ebbe tároljuk el az értéket amit a P0 portról – jelen esetben oda a fotorezisztor van bekötve – olvas be mikrokontroller. Ezután pedig a szám kiírása blokkba beletesszük a korábban létrehozott változót, így a micro:bit beépített LED-jei kiírják nekünk a kapott mérési értéket. Az ehhez tartozó kódot az 5.2. ábra mutatja.

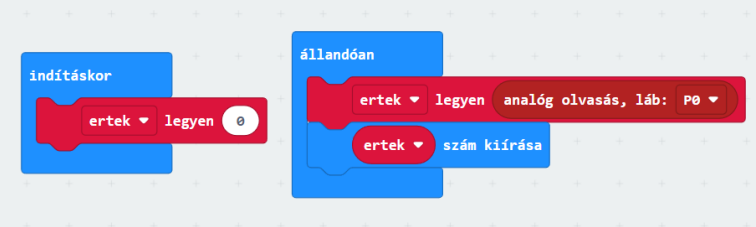
Ha pedig nem a konkrét mérési értéket szeretnénk látni, hanem például azt hogy sötét vagy világos van – ami az okos otthon szempontjából egy értelmesebb megoldás lehet –, akkor az előző kódhoz hozzáadunk egy elágazást (a ”Feltételek„ fülre kattintva lesznek elérhetőek az elágazások és összehasonlító blokkok). Ebben beállíthatunk egy értéket – én ebben a projektben a tapasztalataim alapján a 300-as határértéket reálisnak tartottam, de ez szabadon megváltoztatható–, amely alatti mérés során a ”sotet„

| Teszteset | Rövid leírás | Elvárt eredmény | Eredmény |
|---|---|---|---|
| 1. | A vezérlő automata működése. | Jelenjen meg az "Udvozollek„ szöveg a kijelzőn. | Megjelenik az elvárt szöveg a kijelzőn. |
| 2. | A hőszabályozó automata működése. | Jelenjen meg a mért érték a beépített LED-eken és az RGB LED zöld színnel világítson megfelelő hőmérsékletnél (21-27°C), pirossal hideg esetén (<21°C) és kékkel meleg esetén (>27°C). | Megjelenik a mért érték a beépített LED-eken és az RGB LED zöld színnel világít megfelelő hőmérsékletnél (21-27°C), pirossal hideg esetén (<21°C) és kékkel meleg esetén (>27°C). |
| 3. | A világítás irányító automata működése. | Jelenjen meg a mért érték a beépített LED-eken és a villanykörte sötét esetén (300) világítson, világos esetén kikapcsolt állapotban legyen. | Megjelenik a mért érték a beépített LED-eken és a villanykörte sötét esetén (300) világít, világos esetén kikapcsolt állapotban van. |
| 4. | P0 gomb megnyomására rádiós kapcsolat kialakítása a hőszabályozó mikrokontrollerrel. | Sikeres kapcsolatfelvétel esetén a hőszabályozó által mért eredmény fogadása és megjelenítése a kijelzőn. | Megjelenik a hőszabályozón mért eredmény a kijelzőn. |
| 5. | P1 gomb megnyomására rádiós kapcsolat kialakítása a világítást irányító mikrokontrollerrel. | Sikeres kapcsolatfelvétel esetén a világítás irányító által mért eredmény fogadása és megjelenítése a kijelzőn. | Megjelenik a világítás irányítón mért eredmény a kijelzőn. |
| 6. | A gomb megnyomása a hőszabályozó irányítására a vezérlőn. | Jelenjen meg a vezérlő kijelzőjén a "Futes be„ vagy "Futes/Hutes ki„ szöveg. A hőszabályozó RGB LED-je pedig narancssárgán világítson vagy kikapcsolt állapotban legyen. | Megjelenik a vezérlő kijelzőjén a "Futes be„ vagy "Futes/Hutes ki„ szöveg. A hőszabályozó RGB LED-je pedig narancssárgán világít vagy kikapcsolt állapotban van. |
| 7. | B gomb megnyomása a hőszabályozó irányítására a vezérlőn. | Jelenjen meg a vezérlő kijelzőjén a "Hutes be„ vagy "Futes/Hutes ki„ szöveg. A hőszabályozó RGB LED-je pedig zöldes-kéken világítson vagy kikapcsolt állapotban legyen. | Megjelenik a vezérlő kijelzőjén a "Hutes be„ vagy "Futes/Hutes ki„ szöveg. A hőszabályozó RGB LED-je pedig zöldes-kéken világít vagy kikapcsolt állapotban van. |
| 8. | A+B gomb megnyomása a hőszabályozó irányítására a vezérlőn. | Jelenjen meg a vezérlő kijelzőjén a "Futes/Hutes auto„ szöveg. A hőszabályozó RGB LED-je pedig zölden világítson megfelelő hőmérséklet esetén (21°C - 27°C), pirossal hideg esetén (> 21°C) vagy kékkel meleg esetén (< 27°C). | Megjelenik a vezérlő kijelzőjén a "Futes/Hutes auto„ szöveg. A hőszabályozó RGB LED-je pedig zölden világít megfelelő hőmérséklet esetén (21°C - 27°C), pirossal hideg esetén (> 21°C) vagy kékkel meleg esetén (< 27°C). |
| 9. | A gomb megnyomása a világítás irányítására a vezérlőn. | Jelenjen meg a vezérlő kijelzőjén a "Lampa fel„ szöveg. A világítás irányítóra kötött villanykörte pedig világítson. | Megjelenik a vezérlő kijelzőjén a "Lampa fel„ szöveg. A világítás irányítóra kötött villanykörte pedig világít. |
| 10. | B gomb megnyomása a világítás irányítására a vezérlőn. | Jelenjen meg a vezérlő kijelzőjén a "Lampa le„ szöveg. A világítás irányítóra kötött villanykörte pedig ne világítson. | Megjelenik a vezérlő kijelzőjén a "Lampa le„ szöveg. A világítás irányítóra kötött villanykörte pedig nem világít. |
| 11. | A+B gomb megnyomása a világítás irányítására a vezérlőn. | Jelenjen meg a vezérlő kijelzőjén a "Lampa auto„ szöveg. A világítás irányítóra kötött villanykörte sötét esetén (< 300) világítson, világos esetén (> 300) pedig ne. | Megjelenik a vezérlő kijelzőjén a "Lampa auto„ szöveg. A világítás irányítóra kötött villanykörte sötét esetén (< 300) világít, világos esetén (> 300) pedig nem. |
| 12. | A logó megérintése a vezérlőn. | A vezérlő álljon vissza automata működésbe. | A vezérlő visszaáll automata működésbe. |

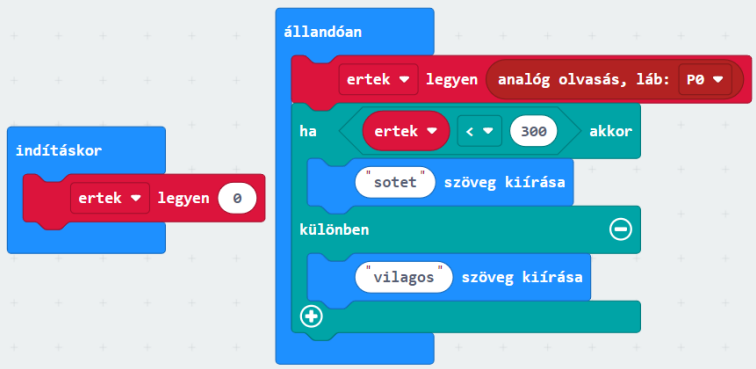
5.1. ábra. Manuális tesztelés táblázata.

vagy afelettinél pedig a ”vilagos„ szöveget jelenítjük meg (fontos tudni, hogy a mic-

ro:bit nem ismeri az ékezetes betűket, ezért nem fogja tudni megjeleníteni őket). Ezt a kódot pedig az 5.3. ábra prezentálja.




5.2. ábra. Egyszerű mérés kódja.




5.3. ábra. A fénymérés alternatív változata.

### 5.2.2. Gomb

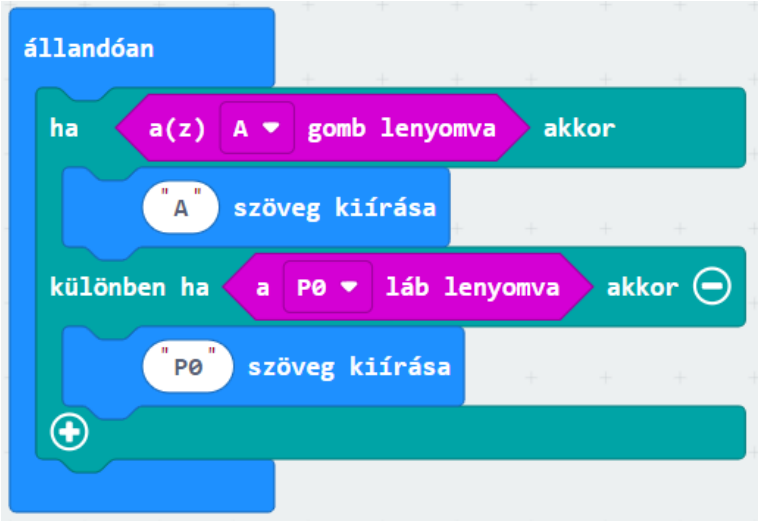
A gomboknál két esetet vizsgáltam meg: a beépített és a külsőleg hozzákötött gombok alkalmazása, illetve a kódblokkok felhasználásai közötti különbségeket. Végeredményben pedig szinte megegyezik a két-két eset, de kifejtem miben is különböznek valójában.

Az első esetben (5.4. ábra) a különböző gombok lenyomásának hatására (a gombokhoz felhasználható kódblokkok a ”Bemenet„ fülnél találhatóak meg) fog csinálni valamit a micro:bit – jelen esetben különféle üzeneteket jelenít meg. – A különbség a kódblokkok között, hogy az egyik fajtával a mikrokontroller beépített ”A„ és ”B„ gombját, illetve azokat együttesen lehet kiválasztani. A másik kódblokkban pedig a P0, P1 és P2 láb között lehet választani.

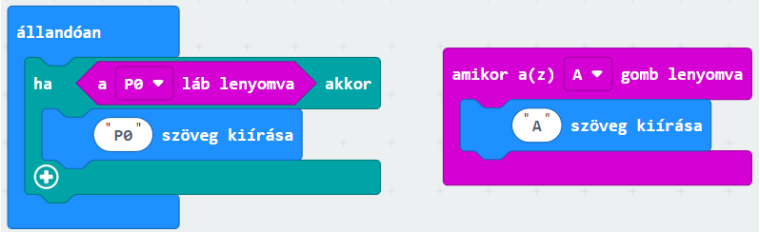
A másik eset (5.5. ábra) ugyan ezt fogja csinálni, viszont máshogy közelíti meg a gomb lenyomásának értelmezését. A két eset közötti különbség csupán annyi, hogy míg az első esetben a folyamat akkor hajtódik végre, amikor az adott gombot lenyomjuk, addig a másik esetben a gomb lenyomása egy igaz/hamis értéket ad vissza, amit az 5.6. ábra prezentál. Ebben az esetben is meg vannak különböztetve a beépített gombok, illetve a portokhoz köthetőek. A két eset összehasonlítására pedig egy JavaScript példát is mutatok a könnyebb megértésért, amit az 5.7. ábra mutat be, amiben sárga és zöld színnel emeltem ki a két metódust.

Röviden tehát, az első esetben a kódblokk egy eljárásként szerepel, hiszen nincs visszatérési értéke (void), a második esetben pedig egy függvény, ami igaz/hamis értékkel tér vissza (boolean).


5.4. ábra. A különböző gombok tesztelésének egyik módja.




5.5. ábra. A különböző gombok tesztelésének másik módja.




5.6. ábra. A gombnyomás két változatának szemléltetése Blokk kódnyelven.

### 5.2.3. I2C LCD

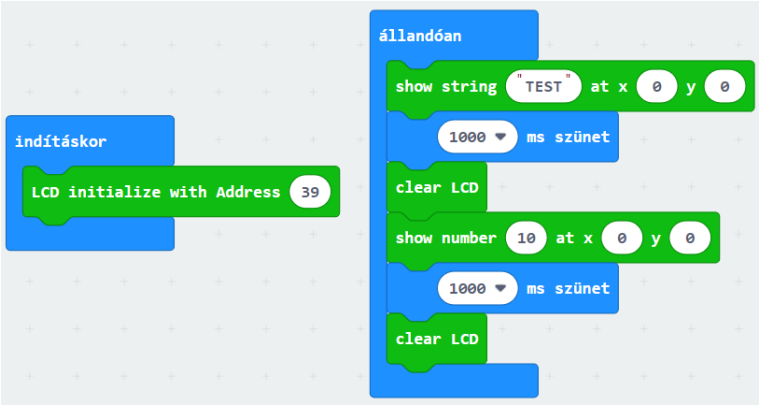
Az LCD kijelző tesztelése nem kifejezetten bonyolult, csupán tudni kell róla pár alapvető dolgot. Például a címezése jelen esetben 39 (0x27) lesz [37]. A kiíratás pedig X (16

```
/* ez a gomb lenyomásához tartozó kódblokk,
amely lenyomásának hatására csinál valamit
a micro:bit */

input.onButtonPressed(Button.A, function () {
    basic.showString("A")
})

/* itt pedig azt vizsgálja, hogy
a gomb le van-e nyomva vagy sem és
annak az értékét adjuk át az elágazásban*/

basic.forever(function () {
    if (input.pinIsPressed(TouchPin.P0)) {
        basic.showString("P0")
    }
})
```

5.7. ábra. A gombnyomás összehasonlításának szemléltetése JavaScript nyelven.

karakter/oszlop) és Y (2 sor) koordináták szerint állíthatóak. Kiterjesztésre itt viszont már szükség lesz, hiszen ez egy pluszban hozzáadott szenzor. Erről pedig a 13. oldalon lehet tudakozódni.

A kódban (5.8. ábra) az egyszerű felhasználását mutatom be, ahol szöveget és számot egyaránt megjelenítek rajta. A szövegek egy másodpercenként váltakoznak a ”clear LCD„ segítségével, amely letörli a képernyőre kiírtakat.




5.8. ábra. Az I2C LCD kijelző tesztelésének kódja.

### 5.2.4. RGB LED

Az RGB LED színeit, ki - és bekapcsolására egy olyan kódot készítettem ami ezeket az eseteket kiválóan prezentálja. Mint ahogyan azt már korábban említettem a LED bemutatásánál a 14. oldalon, fontos különbség van a közös anódos vagy katódos kialakítások között. Ez itt az alábbi kódban úgy nyilvánul meg, hogy a 0 érték a maximum

és az 1023 a minimum érték. Felmerülhet a kérdés: ”de hát nem 255 és 0 közötti értékekkel kellene működnie?„. Viszont a tapasztalataim alapján ameddig csak egy színt szeretnénk kipróbálni működik a 0 - 255 tartomány, viszont amennyiben mind a három színt fel szeretnék használni, vagy váltogatni akarjuk, akkor akármelyik színt állítjuk 255-ös értékre továbbra is világítani fog, de 1023-as értékkel viszont kikapcsolódik.

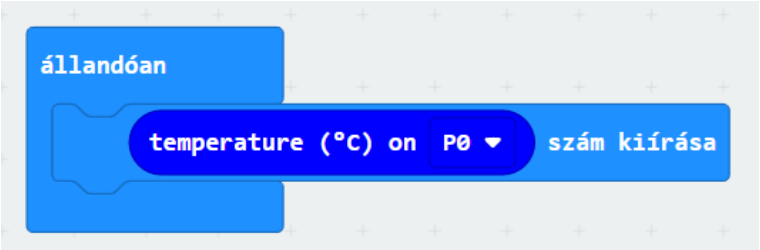
Ezek tudatában hoztam létre az alábbi kódot, amelyben a kezdő állapot az, hogy minden szín ki van kapcsolva, majd adott gombok megnyomása vagy rázás esetén változnak meg a színek. Azaz A gomb: kék, B gomb: piros, rázás: zöld és A+B gomb: fehér fény.

A kódot lehet egyszerűsíteni azzal, hogy nem hozunk létre külön változókat a színeknek, hanem az állandó blokkban lévő analóg írásokat helyezzük közvetlenül be a gombokhoz és rázáshoz. Viszont ezzel a megoldással én csak a könnyebb megérthetőséget szerettem volna elérni.


5.9. ábra. Az RGB LED tesztelésének kódja.

### 5.2.5. Hőmérő

A hőmérő általános tesztelése talán az egyik legegyszerűbb, hiszen egyedül a hozzáadott hőmérő kiterjesztését kell hozzáadnunk a kódhoz, amiről a 14. oldalon volt szó. Ennek a kiterjesztésnek a kódblokkja egy számmal tér vissza (number), amely már °C-ba van átkonvertálva. Ezt pedig legegyszerűbben egy egyszerű kiíratással lehet leolvasni.




5.10. ábra. A hőmérséklet leolvasásának teszt kódja.

### 5.2.6. Szilárdtest relé villanykörtével

Ennek tesztelése is igazán egyszerű és kevés kód kell a megvalósításához. Hiszen a szilárdtest relé digitális jeleket (0 = nem folyik áram/1 = folyik áram) fogad és azokat alakítja át elektromos árammá, amely pedig a villanykörtét üzemelteti, azaz egyedül a reléhez kell kódot írni, a feladat nagy részét az végzi.

Ebben a kódban a kezdő állapot szerint lekapcsolva lesz a villanykörte. Az A gomb megnyomásának hatására pedig digitális jelet küldünk a bekötött portra. Ennek következtében a villanykörte fel fog kapcsolódni. B gomb megnyomásánál pedig visszaállítjuk a digitális jelet, így a lámpa újra lekapcsolt állapotban lesz.

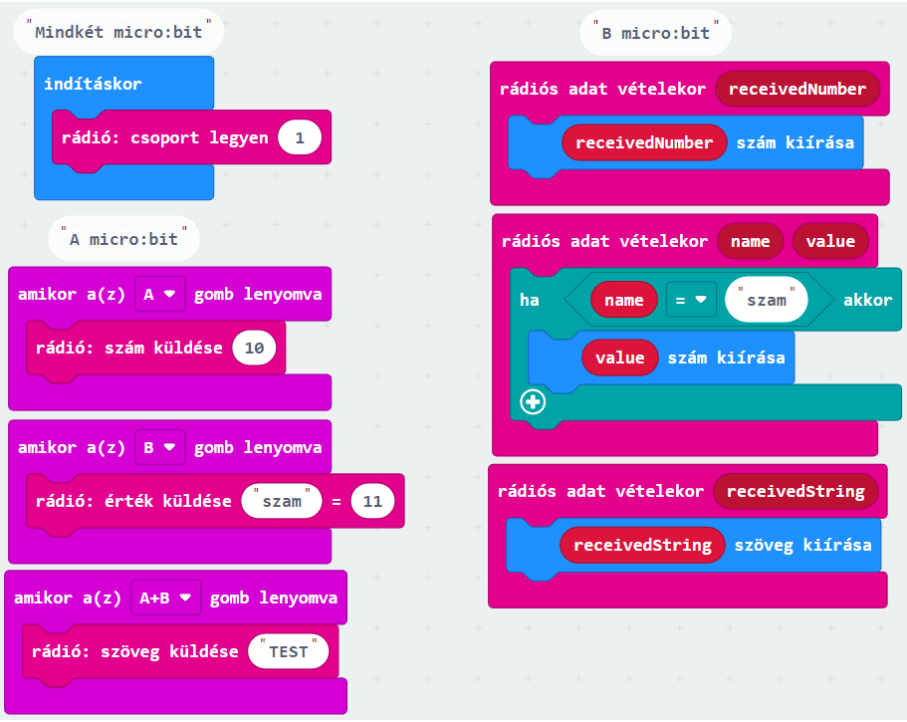

5.11. ábra. A szilárdtest reléhez kötött lámpa tesztelési kódja.

### 5.2.7. Rádiós kapcsolat

A rádiós kapcsolat kialakítására szerencsére vannak beépített kódblokkok, így egyszerű a kivitelezése. Egyedüli hátulütője, hogy egyetlen egy micro:biten ezt nem lehet kipróbálni, hiszen nem látnánk az eredményét. Ezért többre is szükségünk lesz, ahol pedig egy kontrollerre csak a kód ”fele„ kerül, a másik fele pedig egy másikra. Ez azért fontos, mivel többféle opció is van a kommunikációra, így összehangoltan kell megírni a kódot mindkét micro:biten. Az alábbi kód pedig bemutatja, hogyan néznek ki az egymáshoz tartozó kódpárok mindkét oldalon.

Az indításkor kihagyhatatlan beállítani a rádiócsoportot minden micro:bit számára. Ez által fogják tudni, hogy mely kontrollerek tartoznak egy csoportba (ez leginkább akkor fontos, ha több csoport is van). Majd az ”A micro:bit„-nél felhasználtam az összes rádiós jel küldésére szolgáló lehetőséget, a ”B micro:bit„-nél pedig a fogadásra használhatóakat.

Így amikor az ”A micro:bit„ A gombját használjuk, akkor egy számot fog küldeni. B gombnál valamilyen értéket egy névvel együtt (az érték szabadon változtatható, nem csak szám lehet), a két gomb közös lenyomásánál pedig egy szöveget (string) fog elküldeni. A ”B micro:bit„ pedig attól függően, hogy milyen rádiós adatot kap, feldolgozza azt, majd jelen esetben kiírja a LED-jein a kapott adatokat. Tehát szám esetén a kapott számot, szöveg esetén a kapott szöveget, név és érték esetén pedig akár tudjuk a folyamatokat a névhez kötni és úgy dolgozni annak értékével vagy fordítva, de akár magát a kapott nevet is kiírathatjuk az érték mellett/helyett.




5.12. ábra. A rádiós kapcsolat tesztelésének kódja.

# Összegzés

Összességében örülök neki, hogy egy ilyen projektet választottam szakdolgozatként, hiszen mindig is érdekeltek a robotikás témák, hogy milyen lehetőségeket lehet kihozni még belőle és, hogy miféle újdonságokat tanulhatok meg azon felül, mint amiket eddig tanultam. Illetve, hogy hogyan épülnek fel és miként működnek az okos otthonok. Egy ilyen projekt összeállítása és kutatása pedig remek alkalmat adott számomra, hogy jobban belemélyedhessek ebbe a témakörbe és megismerkedhessek egy számomra addig még ismeretlen mikrokontrollerrel és olyan megoldásokkal, amiket eddig még nem próbáltam.

Korábban volt szerencsém LEGO robotokkal is dolgozni, illetve itt az egyetemen pedig már egy ismertebb mikrokontrollerrel, a már fentebb említett Arduino UNO-val. Valahogy a micro:bittel való munkám mind a kettővel való élményt és tapasztalataimat előhozta. Ezért is volt számomra nosztalgikus amikor egyaránt használtam a Blokk és JavaScript kódokat, olyan érzés volt mintha mind a két – LEGO és Arduino – felületet programoztam volna. Úgy gondolom, hogy a micro:bit valahol a kettő között áll és tökéletes oktató eszköz arra, hogy valakit a kezdő szintről egy haladóba emeljen át, hiszen rengeteg lehetőséggel ismerteti meg az embert.

Továbbá igazán remélem, hogy az ebbe fektetett munkám és időm megtérül majd azáltal, hogy másoknak a segítségére lehet. Igyekeztem egy másabb megközelítést alkalmazni a megszokottól. Próbáltam olyan kódokat írni, amik mind a kezdő és haladó programozók számára érthetőek és követhetőek lehetnek. Örülnék neki, ha az emberek ezt elolvasva inspirálva éreznék magukat és új ötletekkel állnának elő ebben a témakörben.

# Irodalomjegyzék


[1] Digitális Mágia: *Mikrokontroller*
URL: http://dongo.biz/dm/def/term/microcontroller, Hozzáférés: 2024.03.14.

[2] SMARTBUILD: *Melyik a legjobb okosotthon-rendszer?*, SmartBuild Kft., 2024
URL: https://smartbuild.hu/legjobb-okosotthon/, Hozzáférés: 2024.04.14.

[3] SMARTBUILD: *Chameleon Smart Home*, SmartBuild Kft., 2024
URL: https://smartbuild.hu/chameleon-smart-home/, Hozzáférés: 2024.04.14.

[4] SMARTBUILD: *Loxone okosotthon-rendszer*, SmartBuild Kft., 2024
URL: https://smartbuild.hu/loxone-okosotthon-rendszer/, Hozzáférés: 2024.04.14.

[5] SMARTBUILD: *Zipato okosotthon-rendszer*, SmartBuild Kft., 2024
URL: https://smartbuild.hu/zipato-okosotthon/, Hozzáférés: 2024.04.14.

[6] PI-SHOP: *BBC micro:bit v2*
URL: https://www.pi-shop.hu/bbc-microbit-v2, Hozzáférés: 2024.03.15.

[7] Microsoft | MakeCode: *micro:bit pins*, 2022.
URL: https://makecode.microbit.org/device/pins, Hozzáférés: 2024.03.14.

[8] Dr. Abonyi-Tóth Andor: *Programozzunk micro:biteket!*, 2017.
URL: http://tehetseg.inf.elte.hu/jegyzetek/AbonyiToth_Programozzunk-microbiteket-2018.pdf, Hozzáférés: 2024.03.14

[9] Kushagra Keshari: *Does Arduino Uno have Bluetooth and WiFi?*, Quora, 2021
URL: https://www.quora.com/Does-Arduino-Uno-have-Bluetooth-and-WiFi, Hozzáférés: 2024.04.06.

[10] Makan Gergely: *Az Arduino bemutatása*, Szegedi Tudományegyetem, 2016.
URL: https://www.inf.u-szeged.hu/~makan/tananyagok/elektronikai-alapok-programozoknak/az-arduino-bemutatasa/, Hozzáférés: 2024.03.16.

[11] MICRO:BIT: *Let's code*, Micro:bit Educational Foundation.
URL: https://microbit.org/code/, Hozzáférés: 2024.04.04.

[12] MAKECODE: *Blokk nyelv*, Microsoft, 2022.
URL: https://makecode.microbit.org/blocks, Hozzáférés: 2024.03.25.

[13] EÖTVÖS LORÁND TUDOMÁNYEGYETEM: *A TypeScript programozási nyelv*, ELTE IK, Programozási Nyelvek és Fordítóprogramok Tanszék, 2010.
URL: http://nyelvek.inf.elte.hu/leirasok/TypeScript/index.php?chapter=1, Hozzáférés: 2024.03.25.

[14] K ANDRÁS: *A Python programozási nyelv – 1.*, MálnaSuli, 2020.
URL: https://www.malnasuli.hu/oktatas/a-python-programozasi-nyelv-1/, Hozzáférés: 2024.03.25.

[15] MULTIWINGSPAN: *BBC micro:bit Adding More Buttons*.
URL: http://multiwingspan.co.uk/micro.php?page=push, Hozzáférés: 2024.03.29.

[16] MÁLNA PC: *PICO GPIO Expansion Board*, Revolt Kft., 2024.
URL: https://malnapc.hu/pico-gpio-expansion-board, Hozzáférés: 2024.04.04.

[17] EMAG: *Arduino LCD 1602 IIC/I2C képernyő*, 2024.
URL: https://www.emag.hu/arduino-lcd-1602-iic-i2c-kepernyo-cl339/pd/DVDLFSBBM/, Hozzáférés: 2024.03.29.

[18] ROBOTIQUE: *Display text on 1602A LCD I2C display with Micro:bit*, 2021.
URL: https://www.robotique.tech/robotics/display-text-on-1602a-lcd-i2c-display-with-microbit/#, Hozzáférés: 2024.03.29.

[19] RPIBOLT: *DS18B20 1 vezetékes digitális hőmérő szenzor*.
URL: https://www.rpibolt.hu/termek/ds18b20_1_vezetekes_digitalis_homero_szenzor_raspberry_pi-hez.html, Hozzáférés: 2024.03.23.

[20] BLOGMYWIKI: *Connect DS18B20 temperature sensor to micro:bit*, 2019.
URL: http://www.suppertime.co.uk/blogmywiki/2019/04/microbit-ds18b20-temperature-sensor/, Hozzáférés: 2024.04.13.

[21] MICROBIT LEARNING: *micro:bit basic RGB led example*.
URL: https://www.microbitlearning.com/code/arduino/microbit-basic-rgb-led-example.php, Hozzáférés: 2024.03.29.

[22] Polaridad.es: *Szilárdtestrelé: mi ez és hogyan működik*, 2024.
URL: https://polaridad.es/hu/relevador-estado-solido/, Hozzáférés: 2024.04.04.

[23] Simon Monk: *micro:bit Lesson 4. Sensing Light*, adafruit, 2018.
URL: https://learn.adafruit.com/micro-bit-lesson-4-sensing-light-and-temperature?view=all, Hozzáférés: 2024.03.29.

[24] micro:bit: *Information about the MakeCode editor for the BBC micro:bit*, 2023.
URL: https://tech.microbit.org/software/makecode/, Hozzáférés: 2024.03.14.

[25] oander: *A nyílt forráskód előnyeiről dióhéjban*, 2023.
URL: https://oander.hu/blog/a-nyilt-forraskod-elonyeirol-diohejban/, Hozzáférés: 2024.03.14.

[26] Isaac: *Fritzing: szoftver gyártóknak és elektronikai cikkeknek (és alternatíváknak)*, hardverfont, 2022.
URL: https://www.hwlibre.com/hu/fritzing/, Hozzáférés: 2024.03.16.

[27] Solvusoft: *Fritzing*, 2023.
URL: https://www.solvusoft.com/hu/file-extensions/software/open-source/fritzing/, Hozzáférés: 2024.03.16.

[28] PlantUML: *Understanding the Purpose of PlantUML.*
URL: https://plantuml.com/faq, Hozzáférés: 2024.04.06.

[29] Idegen szavak gyűjteménye: *Sematikus.*
URL: https://idegen-szavak.hu/sematikus, Hozzáférés: 2024.04.06.

[30] TANGENS: *Mi köze a kanadai dollárnak a mérnöki tervezéshez?*, 2020.
URL: https://tangens.hu/cad-szoftver-mi-koze-a-kanadai-dollarnak-a-mernoki-tervezeshez/, Hozzáférés: 2024.03.16.

[31] Smith Péter: *Szekvencia-diagramok készítése PlantUML nyelven*, PSPROG, 2022.
URL: https://psprog.hu/article/szekvencia-diagramok-keszitese-plant-uml-nyelven, Hozzáférés: 2024.04.06.

[32] WikiSzótár.hu: *Modul szó jelentése*, TRIANITY Kft., 2024.
URL: https://wikiszotar.hu/ertelmezo-szotar/Modul, Hozzáférés: 2024.04.08.

[33] Peter Van Epp: *Improved Micro:bit part*, fritzing Forum, 2019.
URL: https://forum.fritzing.org/t/improved-micro-bit-part/7288/8, Hozzáférés: 2024.04.06.

[34] Steve Lilley: *16x2 I2C LCD Part*, fritzing Forum, 2016.
URL: https://forum.fritzing.org/t/16x2-i2c-lcd-part/2041/4, Hozzáférés: 2024.04.06.

[35] Old Grey: *Simple 12v Bulb Part?*, fritzing Forum, 2020.
URL: https://forum.fritzing.org/t/simple-12v-bulb-part/10298/2, Hozzáférés: 2024.04.06.

[36] micro:bit: *micro:bit*, Microsoft, 2022.
URL: https://makecode.microbit.org/#editor, Hozzáférés: 2024.03.23.

[37] Osoyoo: *Micro bit Lesson – Using the I2C LCD 1602 Display*, 2018.
URL: https://osoyoo.com/2018/09/19/micro-bit-lesson-using-the-i2c-lcd-1602-display/, Hozzáférés: 2024.04.08.

NYILATKOZAT

Alulírott ..... SZILÁGYI DEBÓRA ....., büntetőjogi felelősségem tudatában kijelentem, hogy az általam benyújtott, ..... OKOSOTTHON - MODELL KIVITELEZÉSE MICRO:BIT MIKROKONTROLLERREL ..... című szakdolgozat (diplomamunka) önálló szellemi termékem. Amennyiben mások munkáját felhasználtam, azokra megfelelően hivatkozom, beleértve a nyomtatott és az internetes forrásokat is.

Tudomásul veszem, hogy a szakdolgozat elektronikus példánya a védés után az Eszterházy Károly Katolikus Egyetem könyvtárába kerül elhelyezésre, ahol a könyvtár olvasói hozzájuthatnak.

Kelt: ..... Balmazújváros ....., ..... 2024 ..... év ..... április ..... hó ..... 15 ..... nap.

..... Szilágyi Debóra .....

aláírás