Attiny13A mikrovezérlő használata C++ szintaktikák oktatásában

Kiss Ádám[1], Kelemen András[2]

[1] kiss.adam@med.u-szeged.hu;
SZTE SZAOK Élettani Intézet
[2] kelemen@jgypk.szte.hu
SZTE JGYPK Informatika Alkalmazásai Tanszék
Szegedi Radnóti Miklós Kísérleti Gimnázium

Absztrakt. Számos tanterv létezik arra, hogy egyszerű algoritmusokkal, egyszerű nyelven, hogy lehet programozási alapokat tanítani. Ezeknek hátránya, hogy vagy a nyelv nem emelhető át teljes mértékben a későbbi tanulmányokba, vagy kevés olyan élményt nyújtanak, aminek a nagyságát vagy okát értik a tanulók. Ebben a dolgozatban olyan eszközöket próbálunk bemutatni, melyek algoritmikus gondolkodás nélkül képesek nyelvi elemeket tanítani. A megszerzett tudással később már az algoritmusok működése tehető középpontba, illetve a nyelvi eszközkészlet élményekre épülő alapjainak bővítése.

Kulcsszavak: C++, élményalapú tanulás, beágyazott rendszerek

Mottó

„Mondd el és elfelejtem; mutasd meg és megjegyzem; engedd, hogy csináljam és megértem."

Konfuciusz

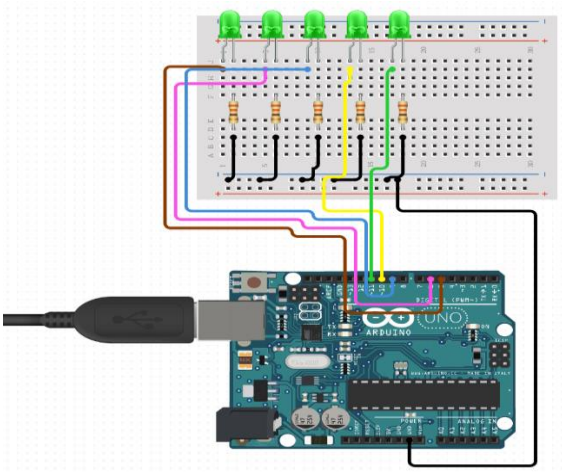
1. Bevezetés

A napjainkban zajló negyedik ipari forradalom erőteljes digitalizációs folyamatokat indított el a gazdasági életben és a társadalomban. Ezen folyamatokra reagálva 2020 szeptemberében bevezetésre került az új NAT, melynek kerettanterveiben az elődjénél hangsúlyosabban szerepel a robotika, valamint az algoritmizálási, programozási ismeretek oktatása [1]. A közoktatásban alacsonyabb évfolyamokon kerül sorra robotika (Lego, Microbit), a blokk programozás (Scratch), valamint a Logo programozási nyelv oktatása. Ezek az eszközök magas szintű vizualitásukkal önálló munkára serkentve a diákot pozitív élménnyé teszik a programozás tanulását. Napjainkban a szoftverek döntő többségét valamilyen szöveges programozási nyelven (C/C++, C#, Java, Python) készítik és a tapasztalatok alapján mind a közoktatásban, mind a felsőoktatásban a kezdők számára a szöveges programozási nyelveken történő programozási ismeretek elsajátítása nagyon nehéz folyamat, annak ellenére, hogy a tanári magyarázatokon, személtetéseken kívül több kiváló magyar nyelvű szakirodalom is rendelkezésre áll [2,3,4,5] Ebben a folyamatban első lépésként a tanulónak az adott programozási nyelv legalapvetőbb kulcs-szavait és szintaktikáját kell elsajátítaniuk, melyre építve jöhetnek az alapvető algoritmusok és az algoritmikus gondolkodási mód elsajátítása. Szintén tapasztalati tény, hogy ebben a tanulási-tanítási szakaszban motiváció felkeltése és folyamatos fenntartása nagyon nehéz feladat egyrészt a rendelkezésre álló idő és az előírt tananyag mennyisége közötti ellentmondás, valamint a diákok eltérő tanulási sebessége miatt. További fusztráló tény a diákok számára, hogy programjuk futásának eredménye az operációs rendszer parancssorában jelenik meg és nem a számítógépen kívül valami látványos felületen vagy jelenségben pl. egy robot mozgásában, vagy LED-ek villogásában.

Ebben a cikkben a fent említett problémák áthidalására mutatunk egy lehetséges utat, mely ugyan azt az élményt nyújtja, mint a fent említett robotok vagy a bélyeg gépek programozása, de sokkal olcsóbb hardverrel megvalósítva. Ráadásul ez módszer a programozási ismerteken túl áramkör tervezési és építési ismerteket is adhat az ez iránt érdeklődők számára.

2. Előzmények az élményalapú programozásban

A magyar közoktatásban az alsóbb évfolyamokon már sok éve használnak grafikus programozási nyelveket (Logo, Scratch) a kezdő lépések megtételére a programozásban. A Logo nyelvet 1967 -ben alkotta meg két matematikus és informatikus Wally Feurzeig és Seymour Paper. Ez a nyelv a Lisp programozási nyelv könnyített, egyszerűsített változata. Többféle implementációja létezik. Magyarországon a közoktatásban az Imagine Logo terjedt el, melyben egy teknőcöt utasításokkal lehet mozgatni a rajzfelületen. Az utasításokat lépésenként is kipróbálhatjuk, de eljárásokba is szervezhetjük. Az eredmény a teknőc által rajzolt ábra formájában azonnal megjelenik. A Scratch a blokk programozási nyelvek családjába tartozó objektumorientált, interpretált, dinamikus és vizuális programozási nyelv, amelyet elsősorban a programozással ismerkedő gyerekek számára fejleszt az MIT. A Scratch-ben a színpadon alakzatokat és szereplőket helyezhetünk el, a szereplőket mozgathatjuk, eseményeket rendelhetünk hozzájuk.



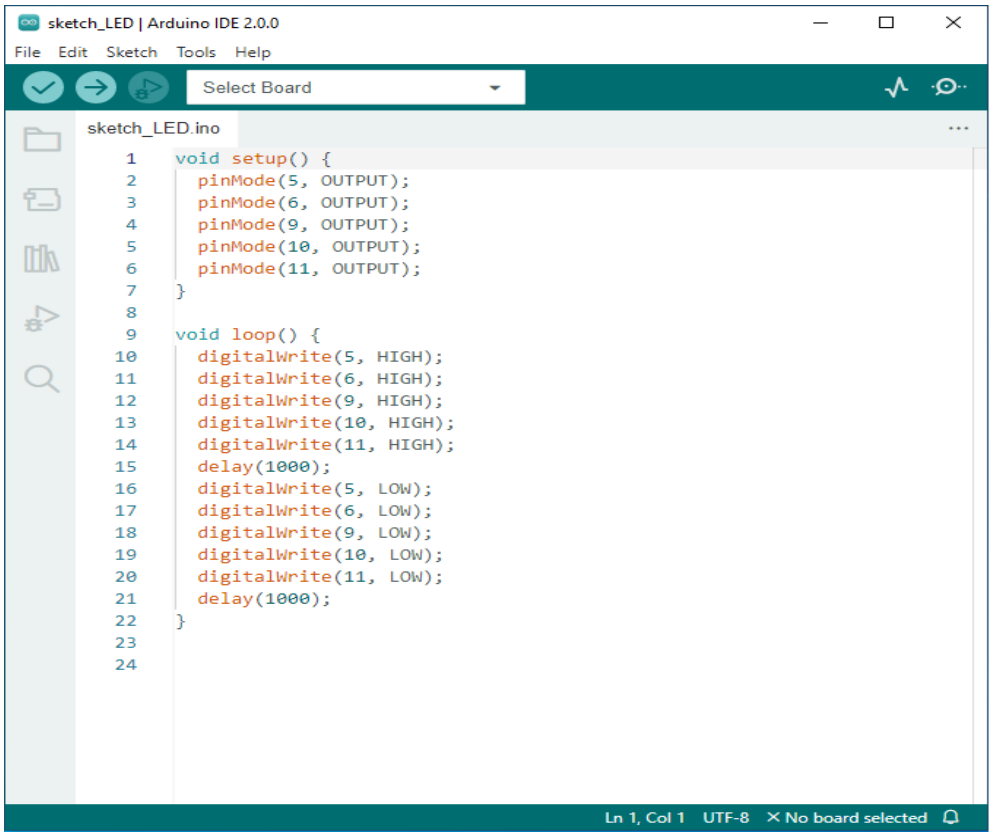
1. ábra: Öt LED bekötése egy Arduino-hoz
(Forrás: Circuit IO)

A LEGO cég 2006 augusztusában mutatta be az első olyan robotépítő készletét [6], mellyel könnyen lehetett játék robotokat építeni és programozni. A robot központi eleme a „tégla”, melybe motorokat és érzékelőket lehet csatlakoztatni. A készletben elegendő mennyiségű passzív alkatrész (tartóelemek, fogaskerekek stb.) is található, melyekkel sokféle robot építhető. A robot programozásához a LEGO saját blokk programozási környezetet ad. Ez a környezet a legújabb verziójában (Robot Inventor) a Scratch-hez hasonló programozói felülettel rendelkezik. Ezzel az innovációval a programozás oktatása kimozdult a számítógép zárt dobozából és valódi élménnyé tette azt, valamint elindította a gyerekbarát hobbielektronika forradalmát.

A LEGO innovációjával párhuzamosan indult az Arduino[7] projekt, melynek célja egy teljes, mégis bővíthető beágyazott fejlesztőkörnyezet biztosítása, amibe a hardverek is beletartoznak az elektronikával hobbi szinten foglakozók számára. A fejlesztőkártyába beépített mikrovezérlő programozását egy magas szintű könyvtárral elfedték. Az Arduinoval való foglalkozáson először egy áramkört kell építeni. Ennek tervezésére és szemléltetésére a Circuit IO nevű webes alkalmazás az egyik legszemléletesebb. Itt a kiválasztott alkatrészekhez a szoftver rögtön egy bekötési ábrát rendel (1. ábra), melynek

segítségével az eszköz könnyen összerakható. Az eszköz programozása a szabadon letölthető Arduino IDE-vel egy a C++-hoz nagyon hasonló nyelven viszonylag egyszerűen megoldható.

Az Arduino programok forráskódja két részből áll. Az inicializáló rész (setup) rész egyszer fut le a bootoláskor. Az ún. loop rész folyamatosan ismétlődve lefut (2. ábra).



2. ábra: Az 1. ábrához tartozó LED-ket villogtató program kódja

3. Célkitűzés

A korábban ismertetett eszközök árban, komplexitásban és újra felhasználhatóság tekintetében legtöbbször nem kielégítőek. Célunk egy olyan eszköz készítése volt, ami megfizethető, valamint transzferálható tudást biztosít, illetve élménypedagógia szempontjából versenyképes a korábban bemutatott Arduino és LEGO termékekkel. Az elkészített eszközt egy nyolc fős csoportban mutattuk be. Célunk volt az élményalapú pedagógia elveit követő demonstráció az utolsó alkalmakon.

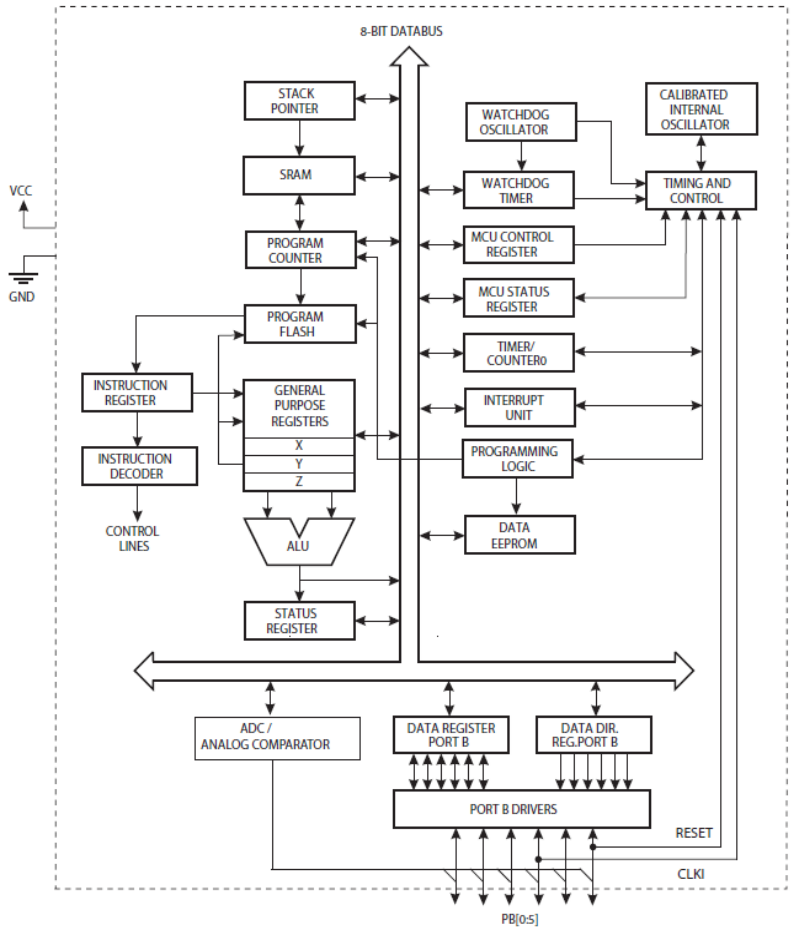
4. Alkalmazott módszer

A kitűzött cél megvalósításához egy Arduino szerű, azonban annál kifizethetőbb, valamint kevesebb hardverismeretet igénylő eszköz tervezését, megépítését, illetve annak programozás oktatásban való kipróbálását tűztük ki.

4.1. Áramköri terv

Munkánkban a korábban felsorolt lehetőséget szerettük volna céltudatosan bővíteni. A LEGO robot térbeli mozgása helyett egyszerű vizuális kimenettel láttuk el az eszközt. Az ehhez szükséges áramkör

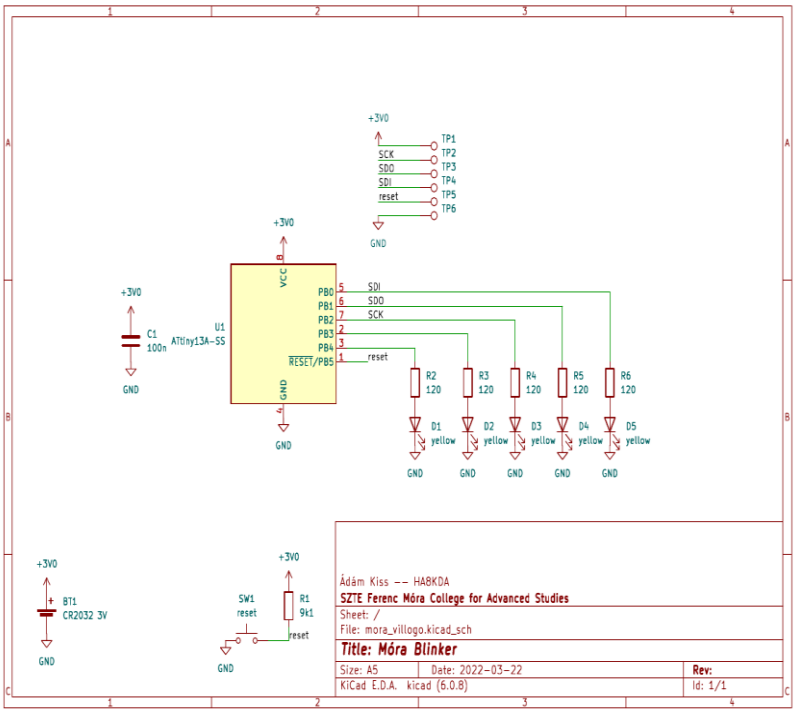
kapcsolási rajzára egy Attiny13A mikrovezérlő (3. ábra) került, ami 2022 tavaszán a legolcsóbb helyben beszerezhető mikrovezérlő volt. A 64 byte operatívmemória, és az 1 kByte flash memóriát elegendőnek ítéltük a célhoz, azaz néhány LED adott utasítássorozat alapján történő kapcsolásához Ezen kívül a szükséges programozó csatlakozók, a LED-ek, illetve egy CR2032-es elemfoglalat kapott helyet a kapcsolási rajzon. A rögzítő furat és a telepes üzem lehetővé teszik, hogy a diákok később díszként használják az elkészült áramköröket.



3. ábra: Az Attiny13 mikrovezérlő blokkvázlata

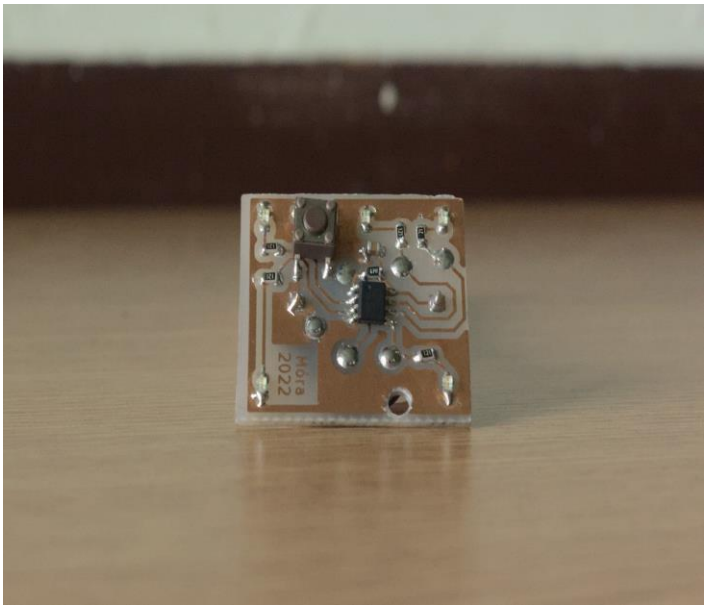
4.2. Megvalósított áramkör

A módszert az SZTE Móra Ferenc Szakkollégiumban próbáltuk ki egy szabadon választható szakkollégiumi kurzus keretében. Az áramköri lapokat az önként jelentkező diákok saját maguknak forrasztották össze a 4. ábrán látható kapcsolási rajz alapján, ami egy élvezetes és különleges élményt adott nekik.



4. ábra: Az elkészült áramkör kapcsolási rajza

Az egyik elkészült példány az 5. ábrán látható.



5. ábra: Egy diák elkészült forrasztása

4.3. Keretrendszer

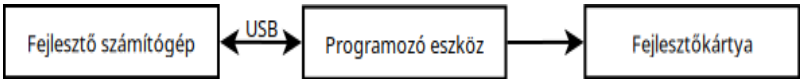
Az oktatási módszer alapgondolata, hogy a nyelvet oktassuk először, és csak később annak használati lehetőségeit. Ennek érdekében a hardver-közeli lépéseket egy keretrendszer biztosítja, maguk a diákok egyszerű kizárólag animációkat írnak egy függvény formájában. A keretrendszert a C++20[4]-as szabványok szerint, illetve annak lehetőségeit használva írtuk. A beépített rutinok egy öt csatornás szoftveres PWM alapú fényerőszabályzást valósítanak meg. A 4. ábra kapcsolási rajzán látható újraindítás gomb megnyomásával a következő animációt tölti be a szoftver.

A diákok a programjukban egy tömbben írnak át számértékeket, melynek eredményeként a számértékkel arányosan változik az ahhoz rendelt LED fényereje. A keretrendszer lehetőséget biztosít mind alacsonyabb szintű, számmal történő, és kvázi-magasszintű asszociatív elérésre is a fényerőket leíró tömbben. Utóbbi fordításidőben kiértékelődik, így nem pazarol erőforrásokat. A keretrendszer tartalmaz továbbá egy késleltető rutint is, melynek segítségével időzíteni is lehet a megjeleníteni kívánt változtatások érvényre jutását.

Magához a program megírásához egyszerű gcc fordítóra van szükség, és GNU Make fordítókörnyezetre. Ezt a párosítást a legtöbb mai fejlesztőkörnyezet támogatja (KDevelop [https://www.kdevelop.org/], Visual Studio Code [https://code.visualstudio.com/], Code::Blocks [https://www.codeblocks.org/] stb.). Egyszerű függvényként kell implementálnia minden diáknak a saját animációját.

5. A fejlesztés folyamatának bemutatása

A fejlesztés folyamán a gazdagépen írjuk a kódot, és egy programozó eszköz segítségével töltjük rá a mikrokontrollerre a 6. ábrán látható módon [8].



6. ábra: Az elkészült program feltöltéséhez szükséges eszközök

A 7. ábrán látható az általunk használt Avr MKII programozó eszköz.

7. ábra: Egy Avr MKII programozó
(Forrás: Microchip)

5.1. A fejlesztőkörnyezet telepítése a gazdagépre

1. Telepítsük az AVR-GCC-t például innen: https://blog.zakkemble.net/avr-gcc-builds/
2. Telepítsünk egy fejlesztőkörnyezetet például innen: https://www.codeblocks.org/
3. Szükséges telepíteni a használt programozó (például 7. ábra) driverét is

5.2. Saját animáció létrehozása

A student mappában hozzunk létre egy hpp fájlt. Ennek az első sorai így nézzenek ki:

```
#pragma once
#include"leds.hpp"
#include"timing.hpp"
```

Definiáljunk egy void(void) típusú függvényt. A LED-ek fényerejét a mappában lévő többi példa alapján lehet állítani egy tömb megfelelő értékeinek átírásával. (Pl.: leds[0]=16) A legmagasabb fényerő értéket a leds.hpp fájl tartalmazza, ami 16. A 0 érték a legkisebb fényerő. A delay_ms függvény segítségével adott millisecundumnyi időkésleltetést vihetünk a vezérlésünkbe.

A main.cpp fájl 25. sorában található tömbbe helyezzük el a saját animációnkat is. Megfelelő animációk közé illesztve bekapcsolás után megtaláljuk majd a sorban a sajátunkat. Amennyiben elfogy a programmemória, úgy néhány animációt törölni kell. Példa az animációk sorrendjét leíró tömbre:

```
static const avr::eeprom_array anim [[gnu::section(".eeprom")]] =
{fade, onoff<500>, rotate<1000>, powerdown, alma, bence, powerdown,
dominik<100>, korte, powerdown};
```

Sikeres fordítás után a használt programozó hardver utasításai alapján égethető a kód a mikrovezérlőbe. Ehhez szükséges a programozó vezetékeinek bekötése is. A bekötési pontok megtalálhatóak a kapcsolási rajzon.

5.3. Tanterv

Az oktatás során az alábbi sorrendben, heti rendszerességgel tartottunk 45 perces alkalmakat:

- Így működik a számítógép (Utasítás-végrehajtási minták)
- Függvény prológusok (ABI-k bemutatása)
- Összetett adattípusok és osztályok memóriaképe
- Fejlesztőkörnyezetek és a GNU Make rendszer
- Fényerőszabályzás alapjai (Impulzusszélesség-moduláció)
- Hogyan tervezzünk animációkat, hogyan valósítsuk meg őket?
- Forrasztás (két alkalom)
- Önálló kódírás

A különböző alkalmak alatt az alapvető programozási tételek [9, 10] (bejárás, elágazás, feltételformálás stb.) is bemutatásra kerültek, gyakorolni csak az utolsó foglalkozásokon volt lehetősége a résztvevőknek. Az utolsó két téma interakciót követelt meg a résztvevők oldaláról, ugyanis ekkor saját kezűleg építették meg a saját eszközeiket. Ekkor már differenciáltan foglalkoztunk az egyes diákokkal, hiszen a különböző készségeik és logikájuk eltértek egymástól.

5.4. Diákmunkák

Az alábbi példát egy pszichológus hallgató írta az órán mutatott módszerek alapján. Teljes mértékben saját maga valósította meg a kódjának első változatát, aholis szekvenciálisan írt utasítások írták le az

állapotok változását. Miután látta, hogy működik, már asszertíven lehetett neki for ciklust mutatni, ami egy egyszerűsítést hozott a kódjába, így az adekvátabb lett.

```
void korte(){
                for(uint8_t i = min_light; i<=max_light; i+=2){
                  leds[BOTTOM_LEFT] = i;
                  leds[BOTTOM_RIGHT] = i;
                  leds[TOP_LEFT] = max_light - i;
                  leds[TOP_RIGHT] = max_light - i;

                  delay_ms(1000);
                }
}
```

Ezen kívül számos egyéb kód született, minden résztvevő legalább egy animációt írt. A diákok kisebb részben az órán mutatott példák alapján, nagyobb részt a saját maguk által kitalált animáció kódolása közben kapott egyéni segítség alapján készítették el az első változatokat. Ezt követően kis csoportokban egymást segítve tanultak új és adekvátabb nyelvi kifejezéseket.

Az elkészült programokat egy tömbben kell elhelyezni, melyek ezen sorrendben jelentek meg magán a hardveren.

5.5. Gyártás

Az eszköz esetleges gyártatása nemzetközi vagy hazai cégekkel is elvégeztethető. Ilyen például az Eurocircuits Kft. vagy a JLCPCB. A függelékben szereplő tervek alapján így rendelhető a hardver. A programozáshoz egy Atmel MKII vagy STK500 programozóra van szükség, amit külön kell vásárolni. A szoftver fordításához GNU környezetre van szükség.

6. Összegzés

Az értékelést a hallgatók személyes, szubjektív benyomása alapján végeztük. A próbakurzuson résztvevő hallgatók túlnyomó többsége tanult már korábban programozást, illetve egyetlen programozni nem tudó, pszichológus egyetemi BA hallgató is jelentkezett az alkalmakra, aki érdeklődve csinálta végig a kurzust. Külön öröm volt neki, hogy az elkészült áramkört hazavihette, ugyanis az alacsony anyagköltségek ezt megengedték, emellett megtanulta az alapvető programozási nyelvi elemeket, illetve az ezek használatához szükséges alapvető gondolkodásmódot. A korábban programozni tanuló résztvevők (7 fő) is tapasztaltak újdonságérzést, valamint a speciális körülmények (virtuális megjelenítés helyett kézzel fogható eredmény) számukra is élvezetessé tették az alkalmakat.

Összegzésként a módszer további vizsgálatra szorul, a kezdeti eredmények biztatóak. A nyelv ismeretével a diákok már tudnak később tisztán mások algoritmusának megértésére koncentrálni. A konkrét példa a modern C++ szabvány miatt összetett vagy bonyolult adatstruktúrák implementálását is megengedik. Hátránya az áramkörnek például egy Arduinoval szemben, hogy ehhez szükséges egy külső programozó áramkör, illetve a LEGO robotokkal szemben, hogy a használt anyagok és a csomagolás hiánya nem teszik kisgyermekbaráttá. Előnye a nagyságrenddel alacsonyabb ár és közvetlen oktatásra kész szoftver.

Meglátásunk, hogy a módszert szélesebb körben is érdemes megvizsgálni. Az észrevételek összegzését az alábbi táblázat tartalmazza:

	LEGO	Arduino	Ez a fejlesztés
Inicializálás	Nem igényel inicializálást	Hardverismeretet igénylő inicializáló rutinok	Nem igényel a beépítettem kívüli inicializálást

	LEGO	Arduino	Ez a fejlesztés
Transzferálható tudás	Nem ültethetők át nyelvi elemek	A megismert nyelvi elemek későbbi munkákban használhatóak	A megismert nyelvi elemek későbbi munkákban használhatóak
Előkészület	Bonyolult hardveres előkészületek	Bonyolult hardveres előkészületek	Egy programozó beszerzése szükséges
Plusz költség	Kb. 150 000 HUF	Kb. 15 000 HUF	Kb 1500 HUF

Irodalom

1. Nemzeti Kerettantervek a Digitális kultúra tantárgy számára https://www.oktatas.hu/kozneveles/kerettantervek/2020_nat
2. Tóth Bertalan: Programozzunk C++ nyelven! Az ANSI C++ tankönyve, ComputerBooks, 2003
3. Juhász Tibor, Kiss Zsolt: Programozási ismeretek, Műszaki Könyvkiadó, 2011, 2015
4. Tamás Ferenc: Visual C# alapismeretek felhasználói szemmel, ISBN 978-963-89484-2-7, Informatika-Számítástechnika Tanárok Egyesülete Budapest 2017
5. Magyary Gyula: Emelt szintű informatika érettségi 2. Python lépésről lépésre, ISBN 978 963 551 028 3, Metropolis Média Group Kft. 2020
6. Lego. NXT Mindstorms. url: https://web.archive.org/web/20060911025537/ http://www.lego.com/eng/info/default.asp?page=pressdetail&contentid=21257&countrycode=2057&yearcode=2006&archive=true. (Utolsó megtekintés: 19.11.2022).
7. Michael Shiloh Massimo Banzi. Getting Started with Arduino. 3. kiad. Make: Community, 2014.
8. Kelemen András: Beágyazott rendszerek programozása C nyelven. [Online educational package (e-learning lesson/topic)] (https://eta.bibl.u-szeged.hu/864/), SZTE Elektronikus Tananyag Archívum, 2011
9. Ivor Horton, Peter Van Weert: Beginning C++20: From Novice to Professional 6th ed. Edition, Apress, 2020
10. Láposi Zoltán, Kiss Tibor: C# programozási segédlet és munkafüzet, Műszaki könyvkiadó Budapest 2019 ISBN 978-963-275-124-5

Függelék

A projekt forráskódja és a részt vevő diákok munkái megtekinthetők online is az alábbi linken: https://github.com/kissadamfkut/mora_blinker

Az áramkör megépítéséhez részletesebb információ a Readme file tartalmaz.

Megjelenés a nemzetközi médiában:

https://hackaday.com/2022/05/14/electronics-and-c-education-with-an-attiny13/