MTA-SZTE Műszaki Informatika
Szakmódszertani Kutatócsoport

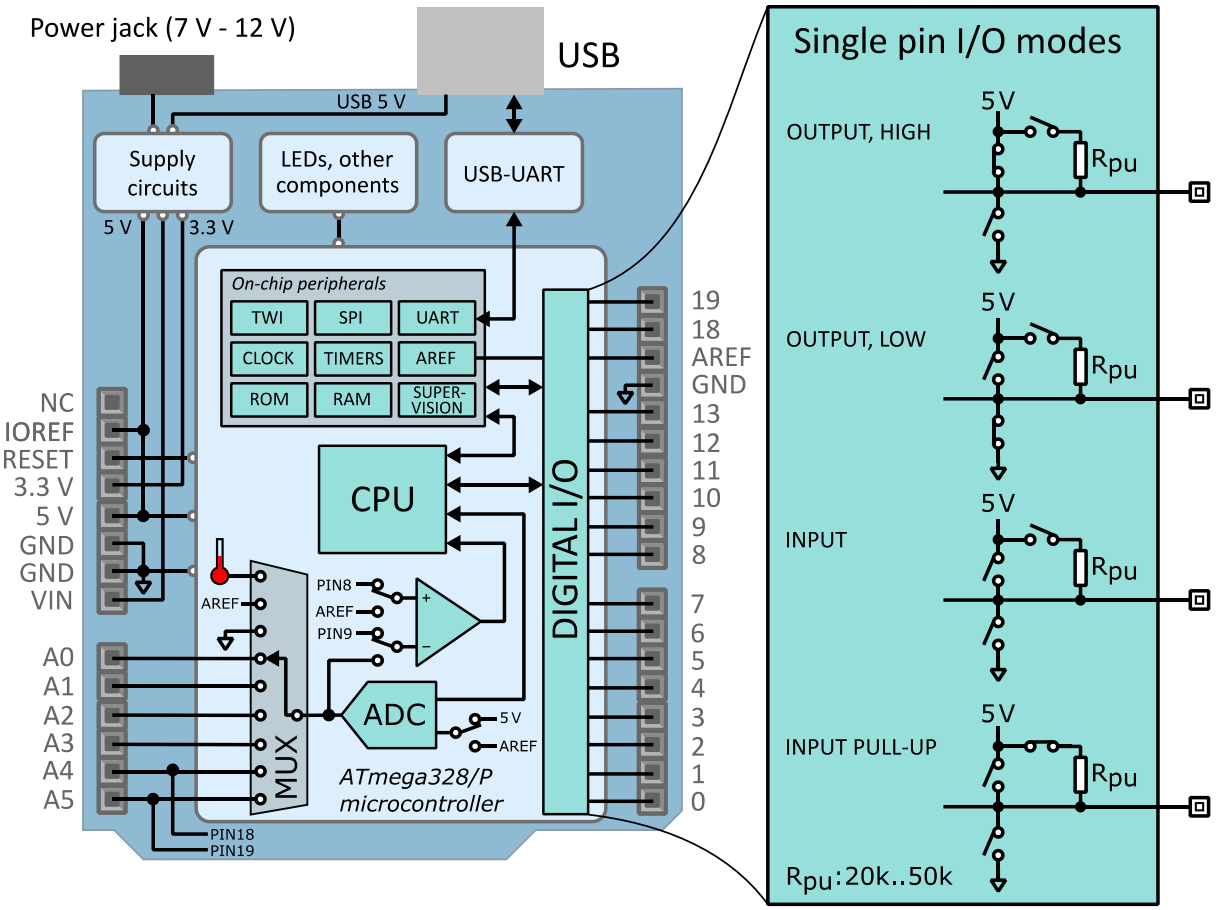
Az Arduino UNO platform

Makan Gergely, Gingl Zoltán

Az Arduino egy mikrovezérlő programozását lehetővé tévő hardverből és szoftverből álló platform. Az Arduino UNO esetében a hardver egy ATmega328/P típusú mikrovezérlőre épül, aminek a lábai ki vannak vezetve egy áramköri lap széleire, így könnyen csatlakoztathatók hozzá külső elektronikai alkatrészek. A mikrovezérlő működéséhez és a programozásához szükséges alkatrészek szintén megtalálhatók az áramköri lapon. A szabadon letölthető szoftverfejlesztő-környezet (Arduino IDE, integrated development environment) tartalmazza a kód megírásához szükséges alapvető elemeket: a szövegszerkesztőt és a fordítót, valamint az Arduino használatát jelentősen megkönnyítő függvénykönyvtárat. A fejlesztőkörnyezet segítségével a lefordított kódot le lehet tölteni a mikrovezérlőre és a beépített szoftvereszközökkel soros adatátviteli protokollon keresztül kommunikálni lehet a letöltött kódot futtató mikrovezérlővel. Így az Arduinótól érkező üzeneteket meg lehet jeleníteni a számítógép képernyőjén, valamint üzeneteket lehet küldeni az Arduinónak is.

Az Arduino UNO felépítése

Az Arduino UNO hardverfelépítését az 1. ábra mutatja. Az Arduino UNO USB porton keresztül csatlakozik a számítógéphez, ezen keresztül lehet felprogramozni és a mikrovezérlővel kommunikálni. Az áramkör a tápfeszültséget is innen kapja alapesetben, de a *Power jack* csatlakozón keresztül DC kimenetű, 7 V - 12 V-os külső tápegységről (pl. fali adapterről) is megtáplálhatjuk az Arduinót. Ebben az esetben automatikusan átvált a külső egységről való táplálásra és az USB-s tápforrást lekapcsolja a mikrovezérlőről. A külső tápfeszültség előnye az USB-hez képest a nagyobb elérhető áramerősség és a stabilabb feszültségszint is, ami analóg méréseknél lehet különösen hasznos. A mikrovezérlő számítógép nélkül is képes a rátöltött programok futtatására. A mikrovezérlőre töltött program a tápfeszültség nélkül is a chip permanens, úgynevezett flash memóriájában marad, és rögtön elindul, amint a mikrovezérlő tápfeszültséget kap valamelyik forrásból. Az Arduino UNO panelen lévő *Supply circuits* nevű blokkba tartozó áramkör biztosítja többek között a megfelelő értékű tápfeszültség előállítását és a tápfeszültség átkapcsolását a külső forrásra, ha az csatlakoztatva van. A *LEDs, other components* blokk az áramköri lemezen lévő 4 db LED-et foglalja magába, valamint egy reset gombot is tartalmaz, amivel újraindíthatjuk a mikrovezérlőt. A LED-ek funkciói a következők: 1 db tápfeszültség meglétét jelző LED, 1 db szoftveresen vezérelhető LED, 2 db soros kommunikációt jelző LED. Az *USB-UART* blokk a mikrovezérlő programozásáért és számítógéppel való kommunikációért felelős áramköröket tartalmazza.



1. ábra: Az ábra bal oldalán az Arduino UNO áramköri lapon lévő komponensek, azon belül részletesen az Atmega328/P mikrovezérlő belső felépítésének blokkvázlata látható. Az ábra jobb oldali része az egyes digitális kivezetésekhez tartozó beállítási módokat mutatja.

Az *ATmega328/P* blokk részletesen mutatja a mikrovezérlő belső felépítését. A processzor (*CPU*) végzi el a program végrehajtásához szükséges műveleteket. A digitális ki- és bemenet modul (*DIGITAL I/O*) logikai alacsony kimeneti szint esetén 0 V-ot, logikai magas esetén a tápfeszültséget (5 V-ot) kapcsolja a kimenetre. Ha bemenetként használjuk, akkor a bemenetre kötött 0 V-hoz közeli feszültség logikai alacsony értéknek (0) felel meg, az 5 V-hoz közeli feszültséget pedig logikai magas értéknek (1) fogja beolvasni a mikrovezérlő. Az analóg-digitális-átalakító (analogue-to-digital converter, *ADC*) feszültség mérését teszi lehetővé, feszültséget tud egész számokká konvertálni, amit a kódban fel tudunk használni. Az ADC hat külső kivezetésre kötött jelet és további három, a chipen belüli jelet (hőmérsékletszenzor, áramköri földpont, referenciafeszültség) képes mérni úgy, hogy egy multiplexer (MUX) egyenként kapcsolja a mérendő csatornákat az ADC bemenetére. Az egyéb beépített perifériák blokk (*On-chip peripherals*) tartalmazza a kommunikációs áramköröket (*TWI, SPI, UART*), az órajelet adó perifériát (*CLOCK*), az időzítőket (*TIMERS*), az analóg mérésekhez szükséges referencia áramkört (*AREF*), a memóriákat (*RAM, ROM*) és a helyes működésért felelős perifériát (*SUPERVISION*). A CPU és az ADC között található háromszög alakú komponens egy komparátor, ami két feszültséget tud összehasonlítani és logikai magas kimenetet ad, ha a pozitív bemenetére kapcsolt feszültség nagyobb a negatív bemenetére kapcsolt feszültségnél. Ezzel például figyelni tudjuk, hogy egy a pozitív bemenetre kötött analóg jel mikor metszi a negatív bemenetére kapcsolt feszültségszintet.

Az ábrán a nyilak jelzik a kommunikáció irányát. Láthatjuk, hogy a CPU minden komponenssel tud kommunikálni, a digitális perifériák (pl. kommunikációs és időzítő perifériák) a digitális ki- és bemenetekkel is összeköttetésben vannak. Az UART periféria az USB-UART blokkal is össze van kötve, ami kétirányú kommunikációt tesz lehetővé a számítógéppel az USB csatlakozón keresztül.

Az Arduino UNO áramköri lap két oldalán tüskesor aljzat csatlakozók találhatók, amire a mikrovezérlő lábai és a tápfeszültségek vannak kivezetve. Az Arduinóhoz nagyon sok kiegészítő áramköri lap, úgynevezett shield kapható, amelyek egyszerűen és gyorsan ráhelyezhetők az Arduinóra. A shield két oldalán lévő tüskesor az Arduinón lévő aljzatba helyezhető, így mechanikailag is stabil és kompakt marad az áramkör, ahogy 2. ábrán is látható.

2. ábra: Az Arduinóhoz kényelmesen hozzáilleszthető feltét áramkör, ún. Arduino shield.

A feltétek, shieldek mellett rengeteg kisebb, Arduinóhoz illeszthető modul is kapható, amik használatához szükséges Arduino kód szintén szabadon letölthető, így egy adott projekt részfeladatai gyorsan megvalósíthatók. Fontos, hogy mindig figyeljünk a működési feszültségekre, mert az eltérő lehet az Arduino és a shield esetében. Ha például egy 3.3 V-os shieldet egy 5 V-os Arduinóhoz kötünk, akkor az tönkreteheti a shieldet, de akár az Arduino is károsodhat, a működés sem lesz megfelelő. Ha mindenképpen össze akarunk kötni két eltérő működési feszültségű áramkört, akkor gondoskodnunk kell a megfelelő szintillesztésről és az áramkörök védelméről.

Az Arduino csatlakozóin az alábbi kivezetések érhetők el:

- **NC** – nincs bekötve.
- **IOREF** – az 5 V-os tápfeszültséggel van összekötve, egy külső áramkör ezzel detektálni tudja, hogy mekkora az Arduino tápfeszültsége.
- **RESET** – ezt a bemenetet 0 V-ra kötve újraindul a mikrovezérlő, az Arduinón lévő reset nyomógomb is erre a kivezetésre van kötve.
- **3.3 V** – 3,3 V-os tápfeszültség kimenet, a maximális kivehető áram: 50 mA.
- **5 V** – 5 V-os tápfeszültség kimenet, a maximális kivehető áram: ~400 mA USB tápforrás esetén, ~650 mA külső (Power jack csatlakozó) tápforrás esetén.
- **GND** – az áramkör földpontja. Külső tápfeszültség használata esetén annak negatívabb feszültségű pontja, USB csatlakozáskor pedig a számítógép földpontja.
- **VIN** – tápfeszültség bemenet, a Power jack csatlakozóhoz hasonlóan innen is megtáplálható az áramkör.

- **A0-A5** – analóg bemenetek, az ADC multiplexerére vannak bekötve, feszültség mérésére alkalmasak. Digitális ki- és bemenetként is használhatók.
- **AREF** – a belső referenciafeszültség értéke jelenik itt meg. Külső referenciafeszültség is köthető ide, amit az ADC kaphat meg referenciaként. Ezt csak akkor tehetjük meg, ha az Arduinót előzőleg már külső referencia használatára programoztuk, ellenkező esetben az áramkörök károsodhatnak.
- **0-13, 18, 19** – általános célú digitális be- és kimenetek
    - ezek közül az alábbiak perifériák kivezetéseiként is használtatók:
        - **0** – UART RX (UART kommunikáció adat bemenete)
        - **1** – UART TX (UART kommunikáció adat kimenete)
        - **2, 3** – INT0, INT1 (külső megszakítás bemenetek)
        - **3, 5, 6, 9, 10, 11** – PWM kimenetek
        - **11, 12, 13** – SPI kommunikációs kivezetések
        - **13** – beépített LED
        - **18, 19** – I^2C (TWI) kommunikációs kivezetések

Az Arduino UNO főbb paraméterei:

- Működési feszültség: **5 V**
- Digitális ki-, bemenetek száma: **14**
    - ebből a PWM digitális ki-, bemenetek száma: **6**
- Analóg bemenetek száma: **6**
- Maximális kivehető/elnyelhető áram egy digitális kimenetből: **20 mA**
- Maximális kivehető áram a 3,3 V-os kimenetből: **50 mA**
- A flash (programot tároló ROM) memória mérete: **32 KB**
- Belső RAM memória adatok, változók számára (static RAM, SRAM): **2 KB**
- Belső csak olvasható adatmemória (EEPROM): **1 KB**
- Órajelfrekvencia: **16 MHz**

Az Arduino használatához szükséges további részletek a következő honlapon a „DOCUMENTATION” fülön találhatók: https://store.arduino.cc/arduinouno-rev3

Digitális be- és kimenetek

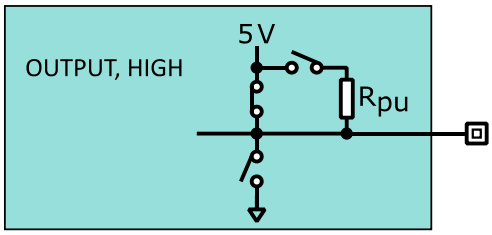
Az általános célú be- és kimenetek (general purpose input/output, GPIO), más néven portok felépítése három, szoftveresen beállítható kapcsoló, egy úgynevezett felhúzó ellenállás, az 5 V-os tápfeszültség és a földpont használatára alapul. Ezek segítségével többféle üzemmód valósítható meg a következőknek megfelelően.

Kimeneti (output) módok

A kimeneti módot egy adott kivezetés esetén a `pinMode(pinNumber, OUTPUT)` függvényhívással állíthatjuk be.

Kimenet, magas szint

Az „OUTPUT, HIGH” esetén csak a tápfeszültséget a kimenettel összekötő kapcsoló (egy belső tranzisztor) van zárt állásban, a többi nyitott (3. ábra). Ez tehát a kimenetre köti a tápfeszültséget, azaz 5 V-ot.

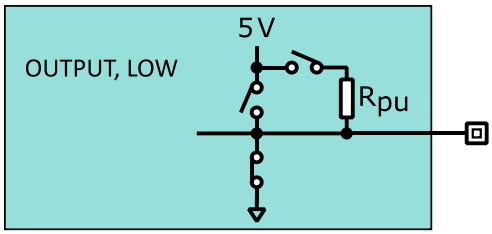


3. ábra: Logikai magas kimenetet biztosító beállítás

Ezt a `digitalWrite(pinNumber, HIGH)` függvényhívás biztosítja.

Kimenet, alacsony szint

Az „OUTPUT, LOW” is kimeneti beállítás, a logikai alacsony kimenet esetén a földpontot kapcsolja a mikrovezérlő a kimenetre (4. ábra).



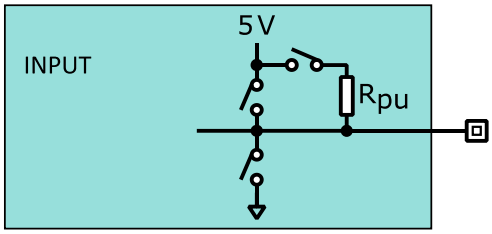
4. ábra: Logikai alacsony kimenetet biztosító beállítás

A kapcsolódó forráskód: `digitalWrite(pinNumber, LOW)`.

Bemeneti (input) módok

Nagy impedanciás bemenet

A bemeneti módot egy adott kivezetés esetén a `pinMode(pinNumber, INPUT)` függvényhívással állíthatjuk be. Ekkor az összes kapcsoló nyitva van, az adott kivezetésre kötött külső feszültségjel zavartalanul és közvetlenül jut a mikrovezérlő belső perifériáihoz (5. ábra).

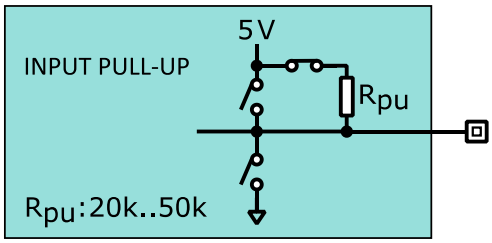


5. ábra: A nagyimpedanciás digitális bemenethez tartozó beállítás

A jel állapotát a `digitalRead(pinNumber)` függvénnyel kérdezhetjük le.

Bemenet felhúzó ellenállással

Az „INPUT PULL_UP” bemeneti mód esetében csak az R_{pu} felhúzóellenállásnál lévő kapcsoló van zárva, amit a `pinMode(pinNumber, INPUT_PULLUP)` függvényhívással érhetünk el. Ekkor a kivezetésre és a belső áramkörökhöz is 5 V feszültség jut, de csak a gyenge felhúzóellenálláson keresztül, amit egy külső jel könnyen megváltoztathat (6. ábra).



6. ábra: Digitális bemenet bekapcsolt felhúzóellenállással

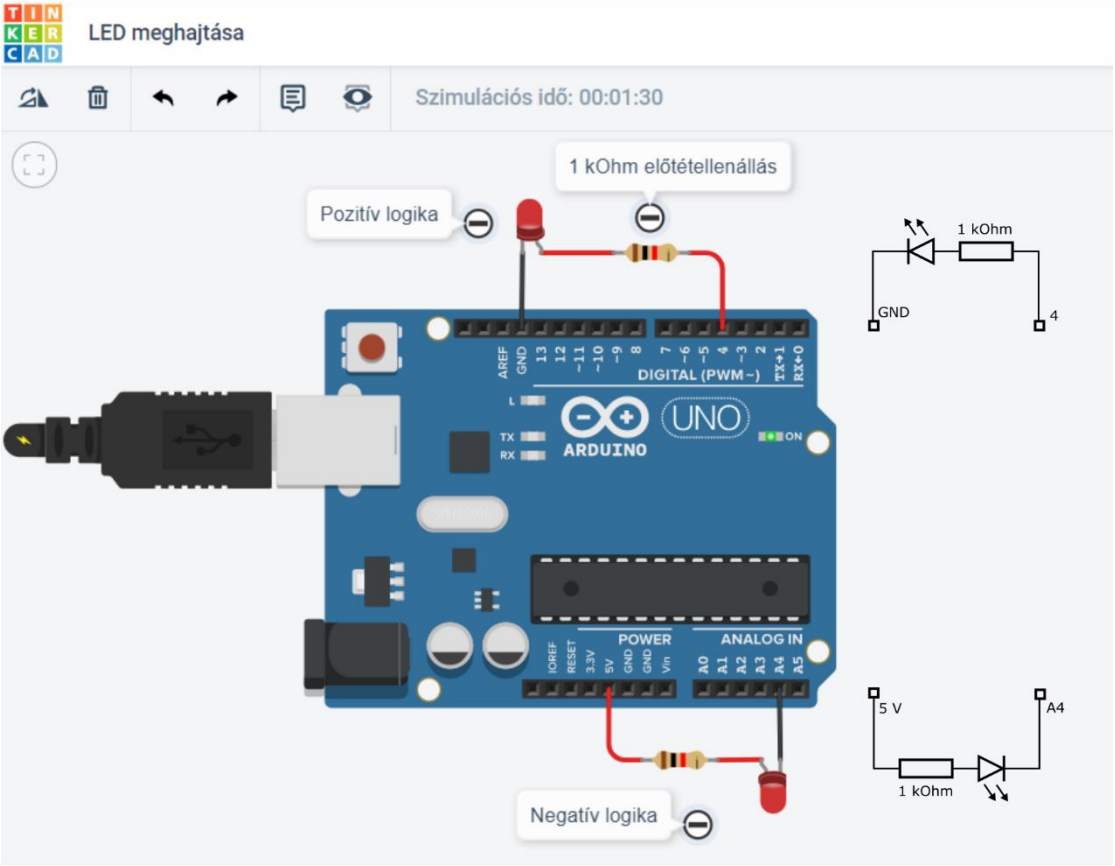
A felhúzóellenállás értéke tipikusan a 20 kOhm..50 kOhm tartományba esik, fő feladata stabil logikai értéken tartani a jelszintet külső meghajtás hiányában is. A későbbiekben látunk egy példát ennek alkalmazására. A jel állapotát ebben az esetben is a `digitalRead(pinNumber)` függvény adja meg.

Alternatív funkciók

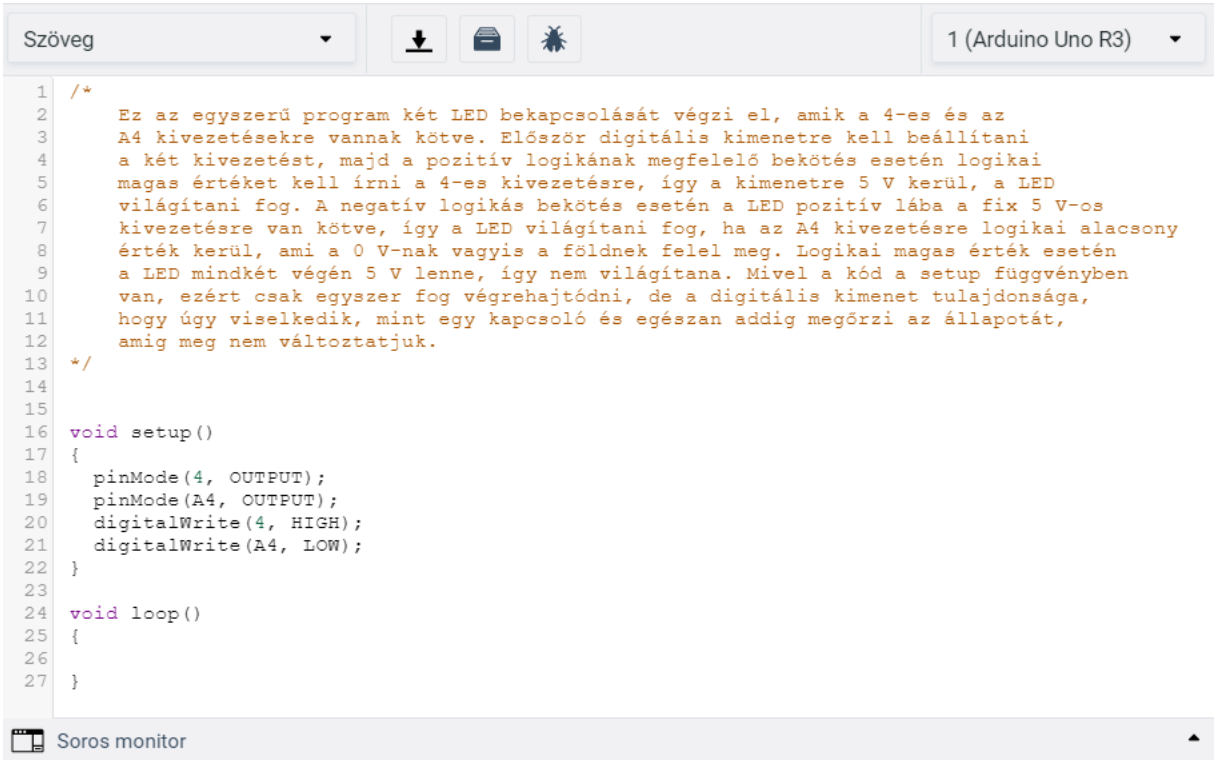
A digitális portok egy része a mikrovezérlő chipre integrált perifériákhoz is rendelhető. Ekkor is be kell állítani ezeket be- vagy kimenetként a feladatuknak megfelelően, de a működtetésüket a továbbiakban a hozzájuk rendelt periféria végzi. A **0** és **1** sorszámú kivezetéseket az Arduino alapértelmezetten az UART áramkörhöz rendeli, ami a számítógéppel való kommunikációhoz szükséges, így ezeket nem célszerű saját célra használni.

LED vezérlése

A 7. ábra egy a LED-ek vezérlésére alkalmas kétféle elrendezéseket mutatja. A kód működésének leírása a forráskód felső részében olvasható (8. ábra). A projekt megnyitható a Tinkercad oldalon.



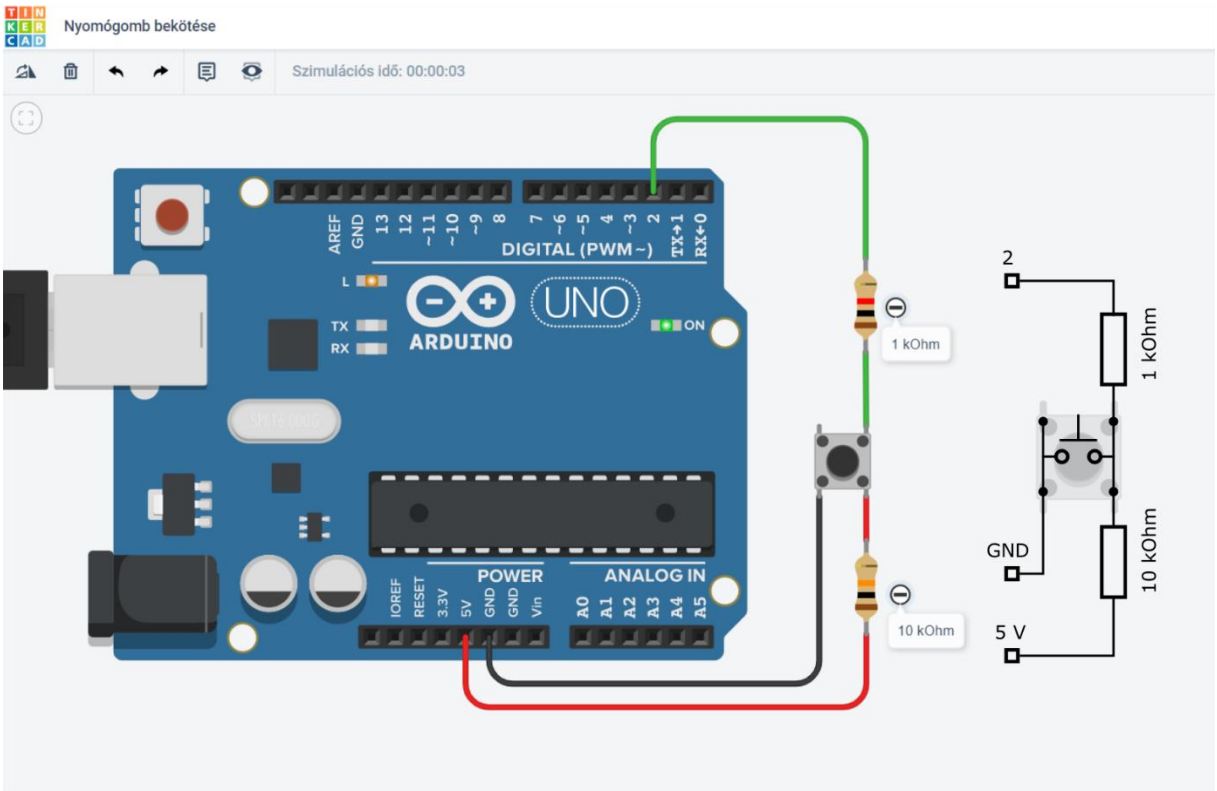
7. ábra: LED-ek helyes bekötése előtétellenállással.



8. ábra: A LED-ek bekapcsolását megvalósító kód

Nyomógomb állapotának beolvasása

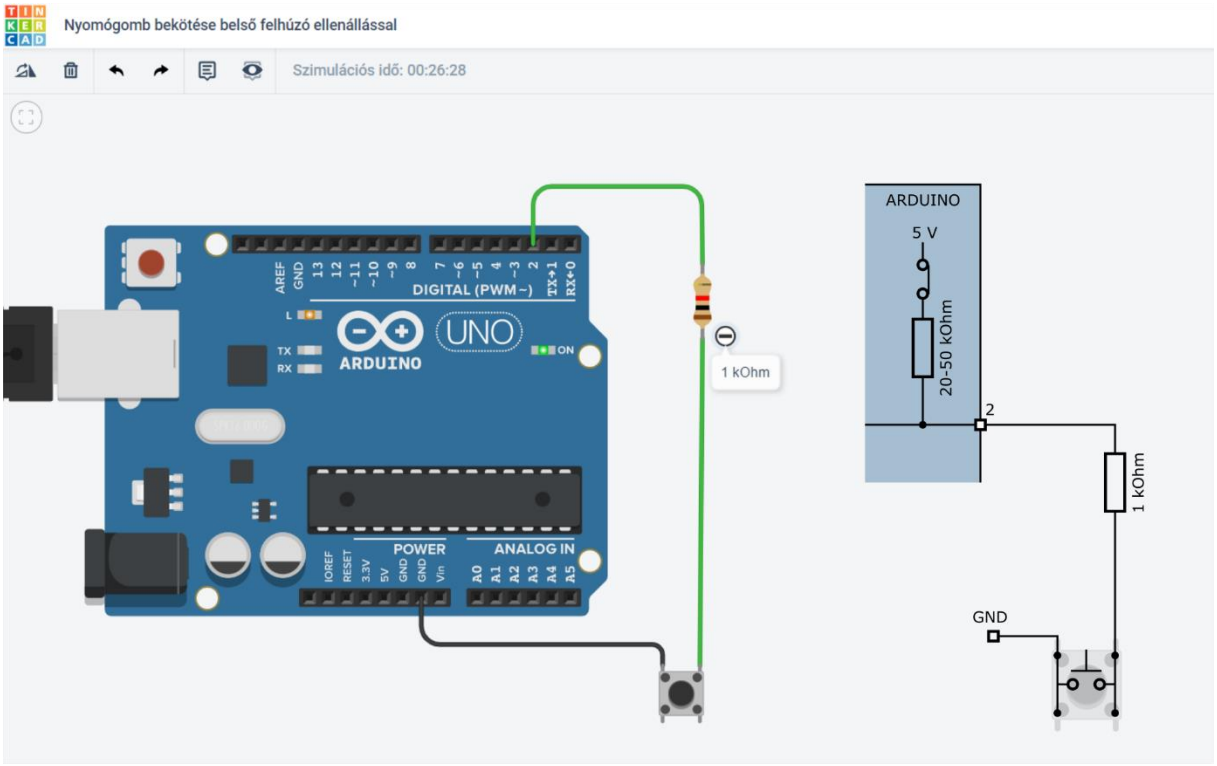
A 9. ábra egy nyomógomb beolvasására alkalmas elrendezést mutat. A működés rövid leírása a forráskód felső részében olvasható (10. ábra). A projekt megnyitható a Tinkercad oldalon.



9. ábra: Nyomógomb bekötése felhúzó- és védőellenállással.

10. ábra: Egy digitális bemeneten lévő feszültség logikai jelszintjének beolvasására alkalmas kód

Az „INPUT PULL-UP” konfigurációt még egyszerűbben használhatjuk egy nyomógomb állapotának beolvasására. Ebben az esetben a belső felhúzóellenállás ellenállás veszi át a fentebbi kapcsolásban látható 10 kOhm értékű ellenállás szerepét. Az 11. és 12. ábrán látható megoldás ezt a módszert alkalmazza. A projekt megnyitható a Tinkercad oldalon.



11. ábra: Nyomógomb bekötése belső felhúzóellenállás bekapcsolásával

Kód | Szimuláció indítása | Exportálás | Megosztás

Szöveg | 1 (Arduino Uno R3)

```
/*
    Ebben a példában egy nyomógom bekötése látható belső felhúzó ellenállással.
    Ez biztosítja, hogy a nyomógomb felengedett állapotában is határozott
    potencálon (5 V) legyen. A belső felhúzó ellenállás értéke 20-50 kOhm.
    Az 1 kOhm-os ellenállás egy áramkorlátozó ellenállás, ami rövidzár esetén véd.
    Ebben az esetben a nyomógomb negatív logikában működik. Ha a nyomógomb nincs lenyomva,
    logikai magas, ha le van nyomva, akkor logikai alacsony értéket olvashatunk
    be. A beolvasott logikai értékkel vezéreljük az "L" felíratú LED-et.

*/

bool buttonState = 0;

void setup()
{
  pinMode(2, INPUT_PULLUP);
  pinMode(13, OUTPUT);
}

void loop()
{
  buttonState = digitalRead(2); //a 2-es kivezetésre kötött digitális jel beolvasása
  digitalWrite(13, buttonState); // a 13-as kivezetésre kötött "L" felíratú LED vezérlése
  delay(10);
}
```

Soros monitor

12. ábra: Belső felhúzóellenállás bekapcsolása digitális bemenet állapotának beolvasása

A digitális be- és kimenetek paraméterei

A következő táblázat összefoglalja a feszültségtartományokat, melyek megfelelő működést biztosítanak a logikai jelek kezeléséhez (5 V tápfeszültség esetén):

Mód / Állapot	Feszültség
Digitális kimenet / alacsony (0, LOW), terhelés < 20 mA	< 1 V
Digitális kimenet / magas (1, HIGH), terhelés < 20 mA	> 4,1 V
Digitális bemenet / alacsony (külső forrásból)	< 1,5 V
Digitális bemenet / magas (külső forrásból)	> 3,0 V

A táblázatból kiolvasható, hogy például logikai alacsony kimeneti beállítás esetén a kimeneti feszültség garantáltan kisebb, mint 1 V, ha a kimeneti áram az előírt 20 mA határértéken belül van. Hasonlóképp, bemenetként garantáltan logikai magasnak fogja a mikrovezérlő a jelet detektálni, ha a feszültség nagyobb, mint 3 V. Nem célszerű ezekhez a határokhoz túl közel kerülni a kiszámítható működés érdekében.

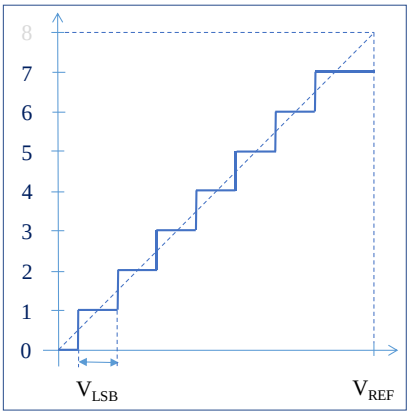
Az ATmega328/P adatlapjában található elektromos szintekre vonatkozó abszolút maximális határokat is be kell tartanunk, hogy ne károsodjon az Arduino áramkör és megbízható maradjon a működése. A legfontosabbak a következő listában láthatók. Ezek már nem feltétlen jelentenek megfelelő működést, de betartásuk esetén az áramkör nem károsodik. **A megfelelő és biztonságos működés érdekében nem szabad ezekhez az értékekhez túl közel kerülni!** A helyes működés adatai a fentebbi részekben láthatók.

Bemenetre köthető feszültségek határértékei:

    - -0,5 V és a mikrovezérlő tápfeszültség+0,5 V között
- Maximális áramok:
    - 1 digitális kimenet esetén: **40 mA** ki- és befolyó áram
    - Az összes digitális kimenetet figyelembe véve: **180 mA** ki- és befolyó áram
    - A0-A5, 0-4, RESET csoportra véve összesen: **145 mA** kifolyó áram
    - 5-13 csoportra véve összesen: **149 mA** kifolyó áram
    - A0-A5 csoportra véve összesen: **99 mA** befolyó áram
    - 5-13 csoportra véve összesen: **99 mA** befolyó áram
    - 0-4, RESET csoportra véve összesen: **95 mA** befolyó áram

Analóg-digitál-átalakító (ADC):

Az analóg-digitál-átalakító (A/D konverter, ADC) az analóg jelet, a bemeneti V_{in} feszültséget x digitális adattá, tulajdonképpen egész számmá konvertálja. Működéséhez egy V_{ref} referenciafeszültség szükséges, melyet N darab egyforma $V_{LSB} = V_{ref}/N$ szélességű intervallumra bont fel az áramkör (13. ábra). Az LSB a legkisebb helyiértékű (least significant) bithez tartozó értéket jelenti. A bemeneti jeltartomány tehát 0 V-tól V_{ref}-ig terjed, mely N egyforma részre van felosztva, a vonalzó a milliéterbeosztásához hasonlóan. A digitalizálás során a konverter megkeresi azt x-edik intervallumot, ami a mérendő feszültséget tartalmazza. A mért feszültség így jó közelítéssel a következő:



13. ábra: Az ADC N db részre bontja fel a referenciafeszültséget.

$$V_{in} \approx x \cdot V_{LSB} = x \cdot \frac{V_{ref}}{N}$$

Az ADC felbontását leggyakrabban bit egységben adjuk meg. Az ATmega328/P-ben lévő ADC 10-bites, ami azt jelenti, hogy a referenciafeszültséget N=2^{10} (1024) részre bontja fel. Az ADC-nek a referenciafeszültsége lehet a belső áramkörök által biztosított 5 V tápfeszültség vagy 1,1 V és akár külső forrásból is érkezhet. A kívánt referenciafeszültséget az analogReference() függvénnyel választhatjuk ki az alábbi módon:

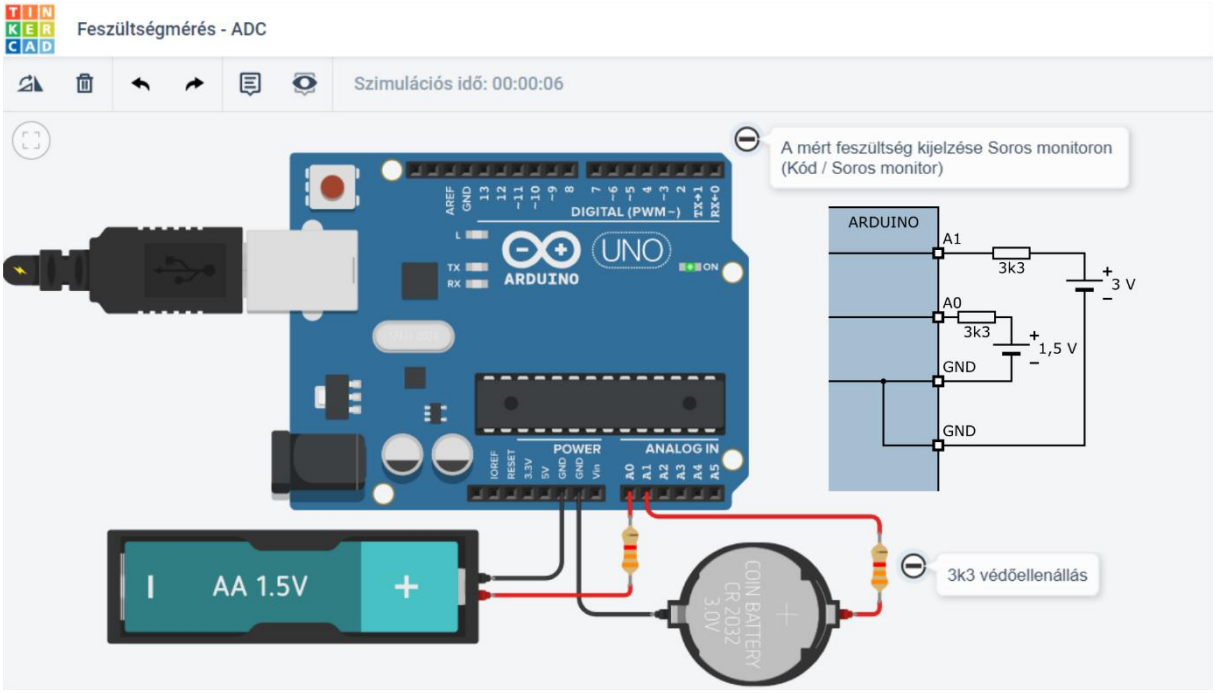
- V_{ref}=5 V: analogReference (DEFAULT);
- V_{ref}=1.1 V: analogReference (INTERNAL);
- V_{ref} külső jel: analogReference (EXTERNAL);

A következő példakód az A0 kivezetésre kötött feszültség meghatározását mutatja 5 V referenciafeszültség használata esetére:

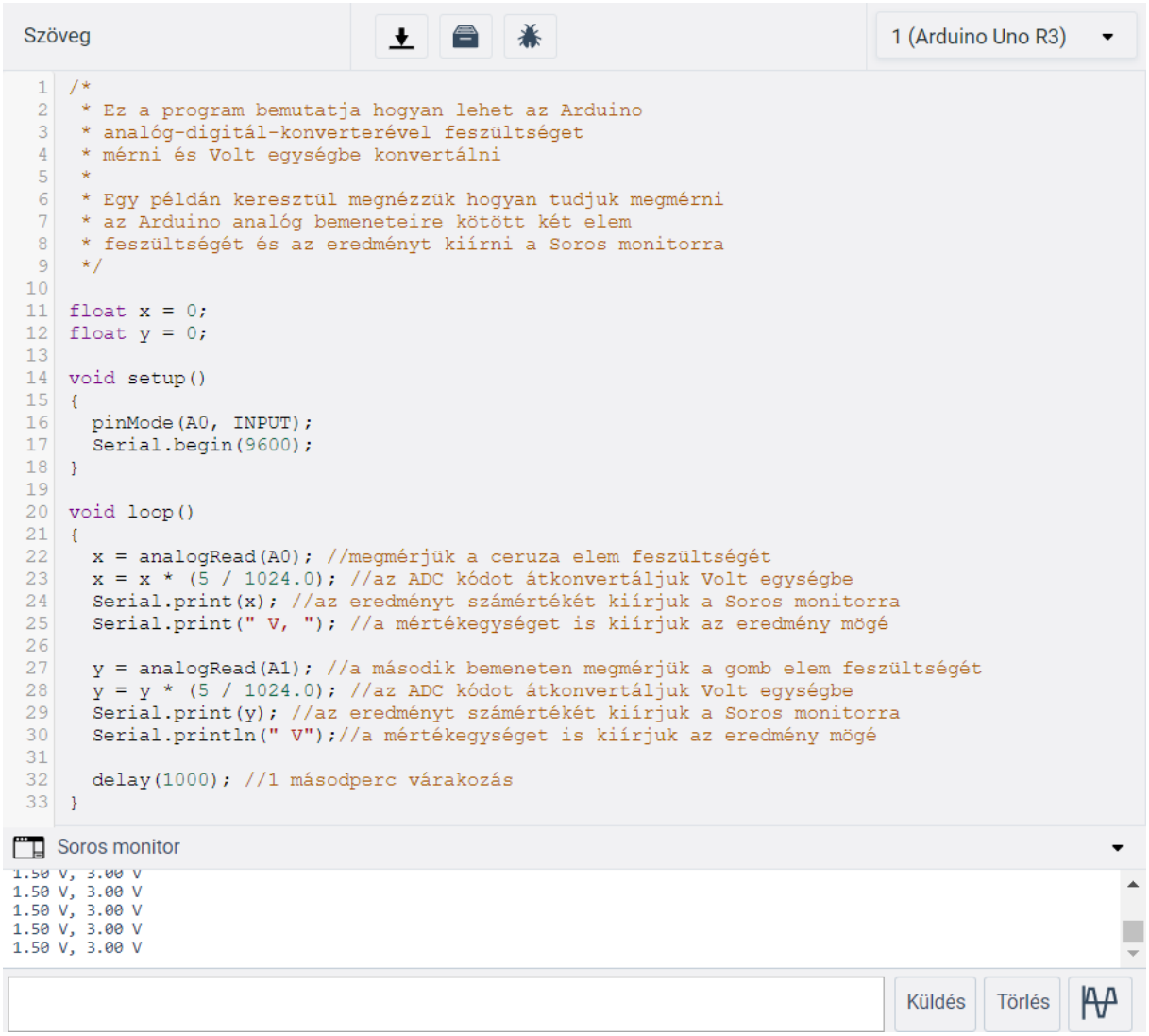
```
voltage = analogRead(A0)*(5.0/1024);
```

Az ATmega328/P mikrovezérlőben lévő ADC-ről részletesen a mikrovezérlő adatlapjában lehet olvasni a 24. fejezetben.

A Tinkercad online szimulációs oldalon is található egy, az ADC működését bemutató szimulációs példa (14. és 15. ábra).



14. ábra: Analóg feszültség mérésére alkalmas kapcsolás 3,3 kOhm-os védőellenállással, két csatorna egyidejű mérésére. A soros ellenállások védenek az esetlege fordított polaritás, hibás kivezetéskonfigurálás és az Arduino áramtalanítása esetén.



15. ábra: Analóg mérés beolvasása, ADC adat konvertálása V egységbe, az értékek felküldése és megjelenítés a számítógép képernyőjén

Referenciafeszültségnek az 5 V alapértelmezetten jól használható. Kis feszültségek esetén lehet megfontolandó a belső 1,1 V-os referenciafeszültség használata. Ha például a mérendő jel 0 V és 1 V között ingadozik, akkor célszerű az 1,1 V-os referenciát beállítani. Ha a mérendő jel nem használja ki a méréstartomány nagy részét, akkor az olyan, mintha az ADC felbontása jelentősen csökkenne. Jegyezzük meg, hogy az 5 V referenciafeszültség tipikusan 5 % pontosságú, míg a belső 1,1 V-os referencia 1 V-tól 1,2 V-ig terjedhet, azaz közel 10 % lehet a hibája. Multiméterrel ellenőrizhetjük ezeknek a feszültségeknek a pontosságát.

Az ATmega328/P analóg-digitál konverterének főbb paraméterei:

- 10-bit felbontás
- 0,5 LSB nemlinearitási hiba
- ± 2 LSB abszolút pontosság
- 15 kHz maximális mintavételi gyakoriság (10 bites felbontás mellett)
- 6 külső csatorna mérési lehetősége
- Belső hőmérsékletszenzor csatorna

- 0 V - 5 V bemeneti feszültségtartomány
- a referenciafeszültség lehet a
    - tápfeszültség (5 V)
    - 1,1 V-os belső referenciafeszültség
    - külső feszültségforrás

A fentebbi listából nem volt még szó az a 0,5 LSB nemlinearitásról és a ± 2 LSB abszolút pontosságról. Az LSB a legkisebb helyiértékű bithez tartozó feszültség nagysága, azaz Vref/N. Minden ADC-nek különböző típusú hibái vannak, amik közül némelyet kalibrálással csökkenteni lehet. A kalibrálás nem egyszerű és nem is nagyon van rá szükség oktatási alkalmazásoknál a legtöbb esetben, a fontos inkább az, hogy tudjunk ezek létéről és nagyságáról. A nemlinearitás az ADC-kód-feszültség függvénynek az ideálistól való legnagyobb eltérése. Az abszolút pontosság az ADC-kód-feszültség függvénynek a legnagyobb eltérése az ideálistól az ADC összes hibáját figyelembe véve. Erre az adatlap ±2 LSB tipikus értéket ad meg, ami 5 V referenciafeszültség esetén ±10 mV körüli érték, 1 V mérésekor tehát ±1 %. A tipikus érték azt jelenti, hogy a legtöbb ATmega328/P mikrovezérlő ekkora legnagyobb hibával rendelkezik. További részletek a hibákról: ATmega328/P Datasheet 24.6.3 ADC Accuracy Definitions.

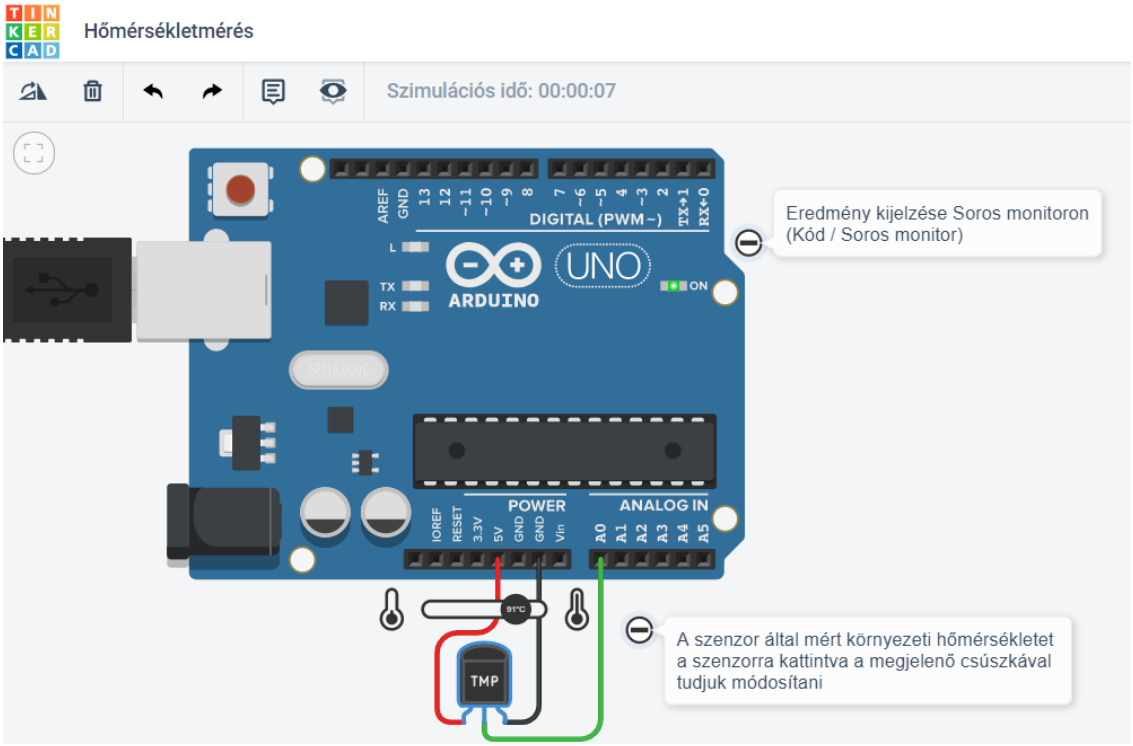
Digitál-analóg-átalakító (DAC):

A digitál-analóg-konverter (DAC) az ADC fordítottja, azaz számokból állít elő feszültséget. Ilyen periféria az Arduino UNO-n nincs, helyette a PWM kimenet használható bizonyos megkötésekkel, vagy kiegészítő áramkörként kapcsolható hozzá külső DAC áramkör is.

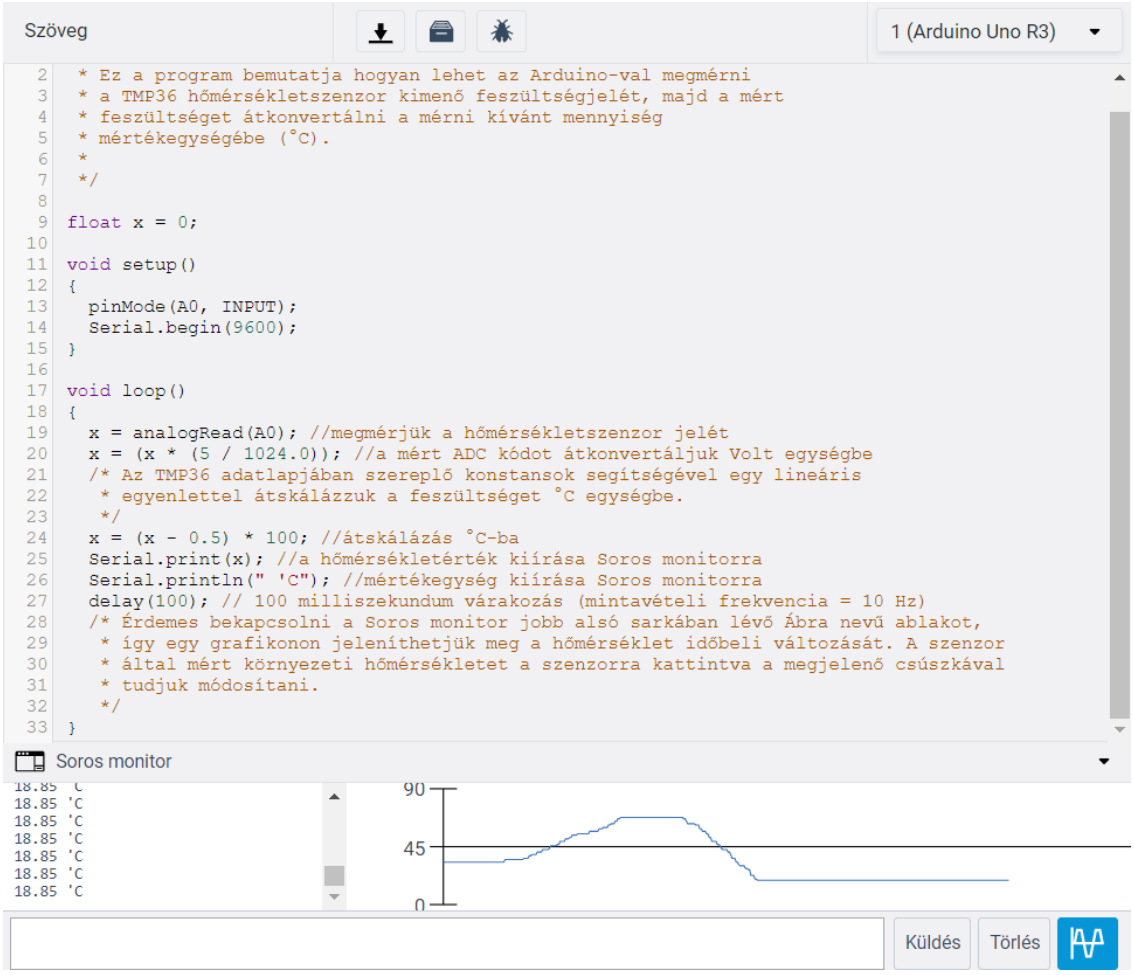
A PWM egy periodikus kétállapotú jel, melynél megadhatjuk, hogy a jel a periódus hányad részében legyen a logikai magas értéknek megfelelő feszültségszinten. 5 V tápfeszültség esetén tehát egy 0 V és 5 V között kapcsolgatott jelet kapunk. Így elérhetjük, hogy egy LED fényerejét vagy motor fordulatszámát ne csak kétféle érték közt tudjuk változtatni, hanem jóval finomabban. Természeteseten ez csak akkor működik jól, ha a jel átlaga fejti ki a hatását. LED esetén a szem nem képes követni a gyors változásokat, így egy átlagos fényerőt látunk, motor esetén pedig a mechanikai tehetetlenség jelenti az átlagérték képzését.

Szenzorok jelének digitalizálása

Ha egy szenzorral meg szeretnénk mérni valamilyen fizikai mennyiséget, akkor a szenzorra jellemző skálázásra is szükségünk lesz, hogy mértékegységhelyesen kapjuk meg mérési eredményt. Feszültségkimenetű szenzorok esetében gyakran lineáris egyenlet (egy szorzás és egy összeadás) jellemzi a feszültség és a fizikai mennyiség összefüggését, de lehet a kapcsolat nemlineáris is. Ellenálláskimenetű szenzorok is léteznek, amik méréséhez még valamilyen kapcsolás is szükséges (legegyszerűbben egy feszültségosztó), hiszen közvetlenül csak feszültség mérhető. A Tinkercad oldalon elérhető egy feszültségkimenetű hőmérsékletszenzor (TMP36) szimulációs példája, ahol megnézhetjük a szükséges skálázás megvalósítását a programban (16. és 17. ábra).



16. ábra: A TMP36 hőmérsékletszenzor egy lehetséges bekötése




```
 * Ez a program bemutatja hogyan lehet az Arduino-val megmérni
 * a TMP36 hőmérsékletszenzor kimenő feszültségjelét, majd a mért
 * feszültséget átkonvertálni a mérni kívánt mennyiség
 * mértékegységébe (°C).
 *
 */

float x = 0;

void setup()
{
  pinMode(A0, INPUT);
  Serial.begin(9600);
}

void loop()
{
  x = analogRead(A0); //megmérjük a hőmérsékletszenzor jelét
  x = (x * (5 / 1024.0)); //a mért ADC kódot átkonvertáljuk Volt egységbe
  /* Az TMP36 adatlapjában szereplő konstansok segítségével egy lineáris
   * egyenlettel átskálázzuk a feszültséget °C egységbe.
   */
  x = (x - 0.5) * 100; //átskálázás °C-ba
  Serial.print(x); //a hőmérséklet érték kiírása Soros monitorra
  Serial.println(" 'C"); //mértékegység kiírása Soros monitorra
  delay(100); // 100 milliszekundum várakozás (mintavételi frekvencia = 10 Hz)
  /* Érdemes bekapcsolni a Soros monitor jobb alsó sarkában lévő Ábra nevű ablakot,
   * így egy grafikonon jeleníthetjük meg a hőmérséklet időbeli változását. A szenzor
   * által mért környezeti hőmérsékletet a szenzorra kattintva a megjelenő csúszkával
   * tudjuk módosítani.
   */
}
```

```
18.85 'C
18.85 'C
18.85 'C
18.85 'C
18.85 'C
18.85 'C
18.85 'C
```

17. ábra: A hőmérsékletszenzor kimenetén megmért feszültség átalakítása hőmérsékletbe, az értékek küldése és megjelenítése a monitoron

Az Arduino platform használata

Az Arduino használatával kapcsolatban néhány fontos dologra hívjuk fel a figyelmet a következőkben.

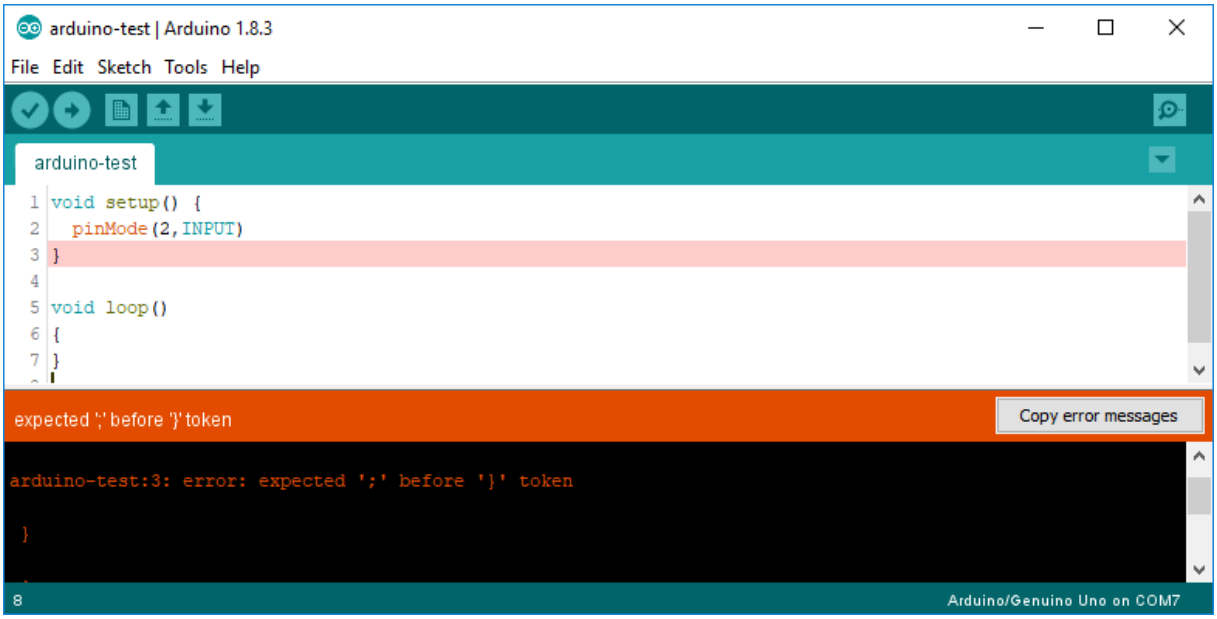
Az integrált fejlesztőkörnyezet használata

Az IDE segítségével tudjuk megírni a programszöveget, melyet le is tudunk fordítani az ATmega328/P mikrovezérlő által végrehajtható kódra. Ez lényegében azt jelenti, hogy a programban szereplő utasításoknak megfelelő gépi kódot (bájtok sorozatát) állítjuk elő. Ezt aztán le tudjuk tölteni a mikrovezérlő programmemóriájába, ami egy pendrive-hoz hasonlóan akkor is megőrzi a tartalmát, ha áramtalanítjuk.

Hibák elkerülése, keresése

A programozás során még gyakorlottabb felhasználók is vétenek néha hibákat. A fő cél természetesen ezek elkerülése gondos tervezéssel és igényes programozási stílus alkalmazásával. A másik fontos cél, hogy az esetleges hibákat minél hamarabb megtaláljuk, kijavítsuk. Ez nem könnyű feladat, mivel a hiba lehet az elgondolásban, a tervezésben, a kódban, de akár a hardveres megvalósításban is. Fontos ezért, hogy jól ismerjük és begyakoroljuk a hibaelkerülési és hibakeresési lehetőségeket is.

A fordítás során hibaüzenetet kapunk, ha a megírt program nem felel meg a programozási nyelv szabályainak, úgynevezett szintaktikai hiba van benne. Ilyen esetben a kódot nem tudjuk lefordítani és a mikrovezérlőre tölteni. Az IDE megmutatja a hiba helyét és fajtáját, ezzel segít a megoldásban, ahogy a 18. ábra mutatja, ahol egy pontosvessző hiányzik az utasítás végén:



18. ábra: Az Arduino IDE kiírja a fordítás során észlelt esetleges hibákat az alsó terminálba ablakba

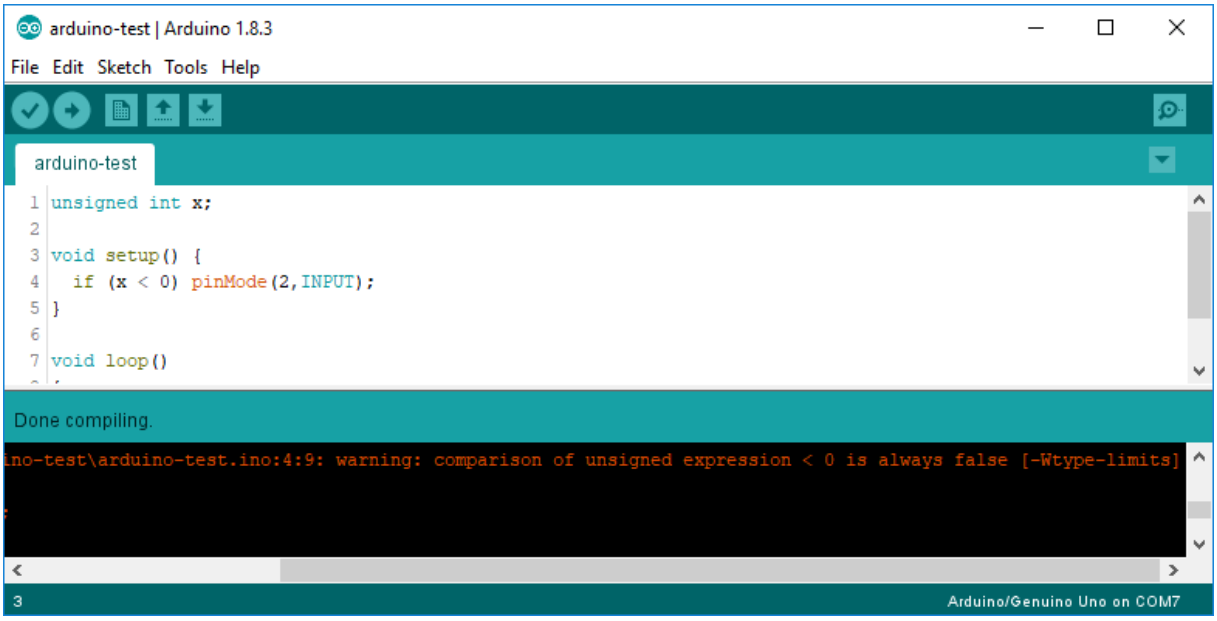
Olyan is előfordulhat, hogy bár a program lefordítható, megfelel a nyelv szabályainak, mégsem fog az elvárásoknak megfelelően működni. Lehetséges például, hogy egy LED-et szeretnénk vezérelni, de a hozzá tartozó kivezetést nem állítjuk kimeneti módba. Az ilyen hibát csak működés közben vehetjük észre, az IDE nem tudja ezeket érzékelni.

Lehetnek olyan részei is a programnak, amelyekről még futtatás előtt sejthető, hogy nem fognak jól működni, mert figyelmetlenül használjuk a lehetőségeket. Ha például az `x` változó típusa `unsigned int`, akkor a következő programsor okozhat problémát:

```
if (x < 0) pinMode(2,INPUT);
```

Nyilvánvaló, hogy a feltétel sosem teljesülhet, azaz valószínűleg hibázott a programozó.

Ilyen esetekben a fordító figyelmeztetést (warning) adhat, ahogy a következő 19. ábrán látható:



19. ábra: A program működésében jelentkező esetleges hibákat segít megelőzni, ha figyelünk a fordító figyelmeztetéseire

Ezzel kapcsolatban egy fontos problémára hívjuk fel a figyelmet, amit az Arduino közösségben sajnos elterjedten hagynak figyelmen kívül. A fordítóprogramoknál beállítható, hogy milyen szigorúan adjanak figyelmeztetéseket, akár ki is kapcsolhatjuk ezek megjelenítését. Az Arduino IDE telepítése után ez utóbbi az alapértelmezés, így semmilyen gyanús kódrészletről sem kapunk figyelmeztetést. Különösen kezdő programozók esetén ajánlott a legerősebb figyelmeztetési szint beállítása és a fordító üzeneteinek figyelése.

A hardver használata

Az Arduino mikrovezérlőjére töltött program azonnal elindul, ha tápfeszültséget kap, akár az USB portra csatlakozáskor. Ha új programot akarunk tölteni rá, akkor szükséges csatlakoztatni az eszközt, ezért az előző kód el fog indulni. Érdemes ezért a következőket figyelembe venni.

Az Arduino elektronikai eszköz, amihez gyakran külső áramköröket csatlakoztathatunk. Alapelv az elektronikában, hogy az áramkörök összeállítását mindig áramtalanított állapotban végezzük. Ez azt jelenti, hogy az Arduinót nem csatlakoztathatjuk addig az USB portra sem, amíg nem állítottuk össze a teljes kapcsolást. Fontos, hogy ellenőrizzük a kapcsolás helyességét, ellenkező esetben a hardver akár károsodhat is. Különösen ügyelni kell a rövidzárak és túlterhelés elkerülésére. Ha aktív áramköröket csatlakoztatunk, akkor még arra is figyelnünk kell, kompatibilisek-e az 5V-os tápfeszültség- és jelszintekkel. Mindig célszerű áramköri rajzot készíteni, ami alapján a kapcsolást elkészítjük.

Azt is láttuk fentebb, hogy a kivezetéseket programból állíthatjuk be kimentként vagy bemenetként. Ha megváltoztatjuk az áramköri környezetet, de a program még az előzőhöz tartozik, akár kimenetek kerülhetnek rövidzárba az eszköz USB porta csatlakoztatásakor. Ezért célszerű legalább az alapbeállításokat elvégző programrészt megírni és letölteni, mielőtt új kapcsolást hozunk létre. A következő lépéseket érdemes tehát követni.

1. Az áramkör összeállítását csak áramtalanított esetben szabad elvégezni.
2. Ha új áramköri kapcsolást akarunk létrehozni, akkor
    a. Áramtalanított esetben távolítsunk el minden külső áramköri komponenst az Arduinóról.
    b. Gondosan tervezzük meg és írjuk meg az új kapcsolást kezelő programnak legalább azt a részét, ami a hardveres beállításokat megfelelően végzi el.
    c. Csatlakoztassuk az Arduino áramkört külső komponensek nélkül a számítógéphez és töltsük le az elkészített programot.
    d. Áramtalanítsuk az Arduinót (húzzuk ki az USB portból) és készítsük el az új kapcsolást.
    e. Csatlakoztassuk az Arduinót a számítógéphez és szükség esetén folytassuk a program fejlesztését.

Érdemes arra is figyelni, hogy ha az Arduino áramtalanított, akkor nem szabad külső feszültségforrás jelét a kivezetéseire kapcsolni. Szerencsés ezért soros ellenállást használni, ami ilyen esetben 1mA környékére korlátozza az áramot, de a normális működést nem befolyásolja. Ilyen megoldás látható a 14. ábrán.

Ha ezeket a szabályokat betartjuk, akkor sok problémát, fáradságos és hosszas hibakeresést, akár áramkörök károsodását kerülhetjük el. Az Arduino mikrovezérlőre és más professzionális elektronikai komponensekre épül, melyeket villamosmérnökök is alkalmaznak. Ennek megfelelő hozzáállással és gondossággal illik használni.

Köszönetnyilvánítás

A tanulmány elkészítését a Magyar Tudományos Akadémia Tantárgypedagógiai Kutatási Programja támogatta.

Referenciák

1. Arduino főoldal, https://www.arduino.cc/
2. Tinkedcad áramkörök, https://www.tinkercad.com/learn/circuits
3. További oktatási anyagok, http://www.inf.u-szeged.hu/miszak/arduino-alkalmazasa-a-fizika-es-az-informatika-oktatasaban/