Mérési utasítás

Labor 2, Mikrovezérlők

Budai Tamás,

budai.tamas@sze.hu

Automatizálási Tanszék

Széchenyi István Egyetem

2018

v1.1.0

Tartalom

Bevezető

A Labor 2 tárgy mikrovezérlők moduljának célja, hogy rövid betekintést nyújtson a mikrovezérlő programozás világába, az Arduino platformon keresztül.

A laborgyakorlat során az **UnoArduSim** szimulátor szoftverben kell a feladatokat megoldani. Ez a szoftver egy Arduino Uno prototípus-panel és a benne található ATMega 328 mikrovezérlő teljesértékű szimulátora. A program rendkívül felhasználóbarát, valamint számos, a gyakorlatban előforduló perifériát is tartalmaz, így a hallgatók működőképes mikrovezérlős alkalmazások modelljét tudják megalkotni a segítségével. A szimulátor használata azért különösen előnyös, mert így kiköszöbölhetők a hardveres problémák, mint például a kezdők által gyakran elkövetett félrehuzalozás, kontakthibák vagy a hardverelemek hibájából adódó helytelen működés. Így ezen a bevezető kurzuson a hallgatóknak elegendő a szoftveres megvalósításra koncentrálniuk.

A mérési utasítás helyes használata

Jelen mérési utasítás önállóan végezhető laborgyakorlatokat tartalmaz. A gyakorlatok végrehajtásához mindössze egy PC-re, az UnoArduSim szoftverre, valamint az elméleti anyagban található ismeretekre van szükség.

Minden gyakorlat közben szerepelnek kérdések és feladatok amelyek megválaszolása és teljesítése a mérés sikeres teljesítéshez szükségesek. A válaszokat és megoldásokat tartalmazó dokumentumot a mérésvezető utasításai alapján kell elkészíteni és a mérés végén leadni. A mérés teljesítése akkor sikeres, ha a mérésvezető ezt a követelményt igazoló dokumentumot elfogadja.

FIGYELEM! Ez a dokumentum az **UnoArduSim 1.7.3**-as verziójára épül. Bár a feladatok feltehetően helyesen működnek és megoldhatóak a szoftver újabb verzióval is, javasoljuk ennek a verziónak a használatát.

A mérési utasítás felépítése, a használt jelölések

A mérési utasítás több, számozott feladatra tagolódik. Bár a feladatokban van egymásra épülés, mindegyik külön-külön is elvégezhető, a sorrend betartása nem lényeges.

A mérési utasításban a következő jelöléseket alkalmaztuk:

> **FIGYELEM!** ez a figyelmeztetés mindig lényeges információt tartalmaz, általában a gyakran elkövetett hibákra figyelmeztet.

```
kódrészlet
```

forráskód részlet, amely szintaktiailag helyes, azaz bemásolható a szimulátor kódszerkesztőjébe, azonban nem minden esetben futtatható önmagában. A legtöbb esetben a működőképes kódhoz további szerkesztés szükséges.

1. utasítás: A feladat megoldásához szükséges *követendő lépések* és a kérdések *számozott felsorolásként és dőlt betűvel vannak jelölve.*

1. mérés: blinky

A mérés célja a szimulátor alapvető funkcióinak megismerése.

Feladatok

1. *indítsd el a programot, majd készíts egy új mentést a forráskódról a File→ Save as menüpont segítségével. A program nevében ne használj ékezetes karaktereket. A mentés helye szabadon megválaszható, de az egyszerű elérhetőség kedvéért célszerű pl. az Asztal-on létrehozott mappába dolgozni.*

FIGYELEM! Ha új mérésbe kezdesz, alkalmazd a fenti 'Save as' funkciót, nehogy véletlenül egy korábbi programodat írd felül!

2. *kattints duplán a programkód ablakba, a szerkesztő megnyitásához.*

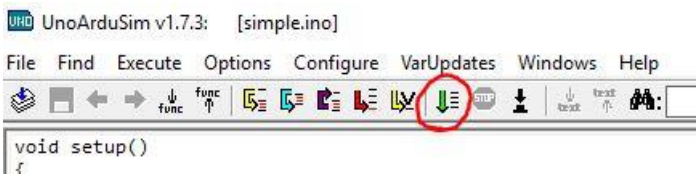
3. *a kódszerkesztőben töröld ki az ott található mintakódot, majd másold be az alábbi forráskódot:*

```
void setup()  // kezdeti beallitasok (egyszer fut le RESET utan)
{
    pinMode(13,OUTPUT);
}
void loop() // 'main loop', folyamatosan fut a kovetkezo RESET-ig.
{
    digitalWrite(13,HIGH);
    delay(500);
    digitalWrite(13,LOW);
    delay(500);
}
```

4. *zárd be a szerkesztőablakot mentéssel (Save gomb, vagy Ctrl+S). Ha a program megkérdezi, hogy felülírod-e az előző mentést, válaszd a 'Yes' lehetőséget.*

5. *nyisd meg újra a szerkesztőablakot, majd az egérrel húzd félre annyira, hogy alatta az eredeti kódablak is látszódjon. Hasonlítsd össze az általad bevitt forráskódot azzal ami a kódablakban van. Mi a különbség? Mi lehet ennek az oka? (segítség: kommentek a kódablakban)*

6. *futtasd a programot a 'run' gombbal:*



1. ábra. run gomb

7. *mi történik a képernyő jobb oldalán az Arduino panelen?*

8. *állítsd le a program futását a 'STOP' gombbal:*

2. ábra. STOP

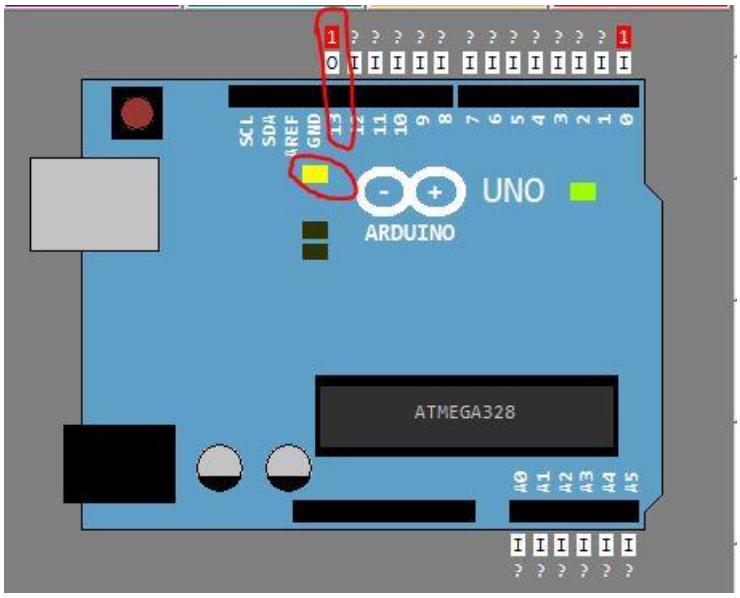
9. *a 'reset' gombbal állítsd alaphelyzetbe a mikrovezérlőt:*

3. ábra. RESET

10. *léptesd a programot lépésenként a 'step into' gombbal (az F4 billentyű is használható):*

4. ábra. step into

11. *figyeld meg hogyan változik a (panelen gyárilag megtalálható) 13. lábra behuzalozott LED állapota miközben a programot lépteted! Melyik forráskódban lévő függvény végrehajtása után villan fel a LED?*



5. ábra. A 13. lábra behuzalozott LED és a 13. láb, mint digitális I/O

12. *Figyeld meg, hogy a léptetés során hogyan viselkedik a* `Loop()` *függvény!*

Mivel ez a `loop()` függvény egy `while(1){}` ún. végtelen ciklusba van beágyazva, a függvény utolsó utasításának végrehajtása után a folyamat kezdődik előről és ismét a loop függvény első sorára kerül a végrehajtás. Ezért van az, hogy bár a programban csak egy be-ki kapcsolás van megvalósítva, a LED folyamatosan villog.

A `delay` **függvény:** A fenti programban a LED 500ms időtartamra ki, majd 500ms időtartamra bekapcsol, azaz egy teljes periódus ideje 500+500=1000ms, azaz 1 másodperc.

13. *módosítsd a fenti programot úgy, hogy a villogás periódusideje 2 másodperc legyen (1mp-ig be, 1mp-ig ki legyen kapcsolva a LED)! A módosított, kész programot (vagy egy képernyőképet a forráskódról) másold be a mérési jegyzőkönyvbe is!*

A mérési jegyzőkönyvnek a következő elemeket kell tartalmaznia:

- Név, NEPTUN-kód
- válasz az 5., 7., 11. pontokban lévő kérdésekre
- a 13. pontban megvalósítandó program forráskódja

2. mérés: digital I/O

A mérés célja az egyszerű digitális be- és kimenetek kezelésének elsajátítása, valamint a szimulátorban használt perifériák bekötésének bemutatása.

Digital I/O:

Az Arduino UNO panelen 14 darab digitális be- és kimeneti láb található.

FIGYELEM! A lábak számozása 0-val kezdődik!

A programban a bemenetekről beolvashatjuk az aktuális értéket, a kimenetekre pedig kiírhatjuk a kívánt értéket. A programban a magas logikai szinthez az `1`, az alacsony logikai szinthez pedig a `0` egész érték tartozik.

Mind a 14 digitális láb használható be- vagy kimenetként, ez azonban azt is jelenti, hogy a lábakat fel kell konfigurálnunk használat előtt!

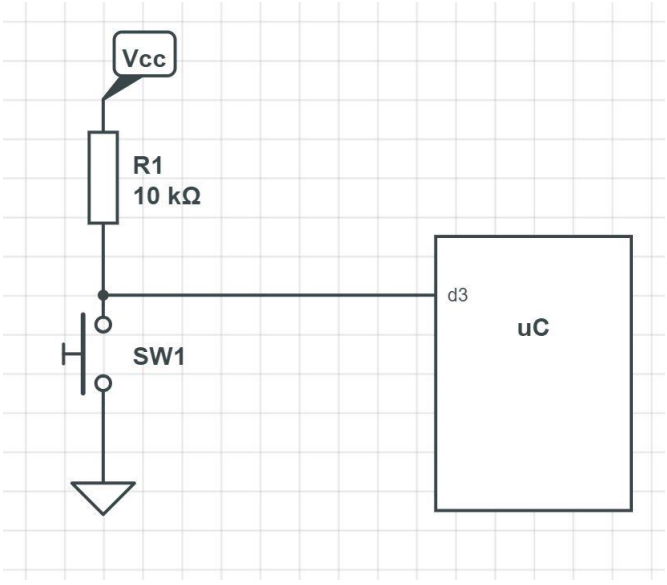
FIGYELEM! A programban nem számít hibának ha kimenetként felkonfigurált láb értéket olvassuk, azonban ez a program helytelen működést eredményezheti!

A lábak felkonfigurálásért a mintaprogram `setup()` függvényében meghívott `pinMode()` függvény a felelős. Első paramétere a digitális láb száma, második pedig az `INPUT` vagy `OUTPUT` konstans.

Megjegyezzük, hogy az ATMega mikrovezérlők esetén minden digitális I/O láb alapértelmezetten bemenetként van felkonfigurálva. Ez azért előnyös, mert nem ad lehetőséget arra, hogy inicializálás nélkül véletlenül magas szintre emeljünk egy olyan lábat amelyet nem szerettünk volna és amelyre más alkatrészek már be vannak huzalozva. Így elkerülhető a mikrovezérlő és a többi alkatrész figyelmetlenségből adódó károsodása.

Nyomógombok

A valóságban a nyomógombokat az alábbi sematikus ábra szerint szokás bekötni:



6. ábra. nyomógomb bekötése

Ebben az elrendezésben a mikrovezérlő lábára magas feszültségszintet kapcsoltunk egy ún. felhúzó ellenálláson (R1) keresztül, majd a gomb megnyomásakor ezt a magas szintet az alacsony szint irányába 'húzzuk'. Erre az ellenállásra azért van szükség, mert különben a nyomógomb megnyomásakor rövidzárlat keletkezne a tápfeszültség és a földpotenciál között.

pushbutton: a szimulátorban található nyomógomb a 6. ábrán látható, zöld gombra kattintással működtethető. A működése alapértelmezetten a csengőkapcsolóhoz és a rugós mikrokapcsolókhoz hasonló, azaz elengedés után automatikusan visszaáll a kezdeti állapotába. Amennyiben normál kapcsolóként szeretnénk használni, a nyomógomb feletti 'latch' bekapcsolásával a működése átállítható. A nyomógomb másik fontos tulajdonsága, hogy benyomáskor milyen szintváltozás történjen. Ezt a zöld nyomógombtól jobbra található választóval állíthatjuk be: alapértelmezetten a nyomógomb lába magas logikai szinten van amelyet megnyomáskor alacsony logikai szintre vált, azaz a nyomógomb 6. ábrán látható kapcsolás szerint működik.

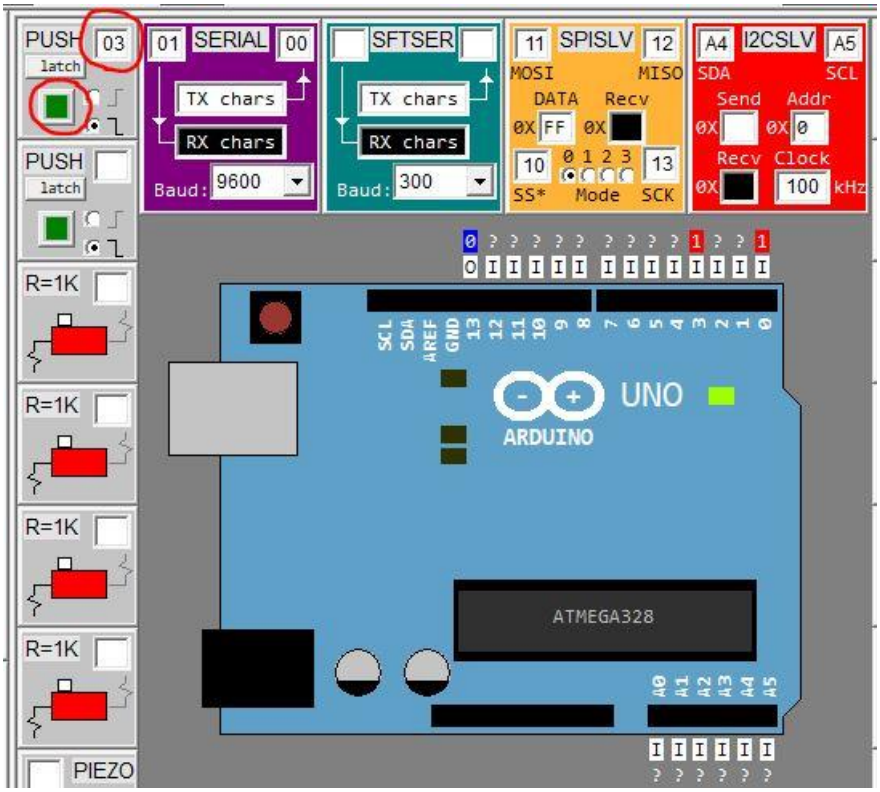
Feladatok

1. *készíts egy mentést a File → Save as menüpont segítségével!*
2. *nyisd meg a kódszerkesztőt és másold bele az alábbi forráskódot, felülírva az ott található kódot, majd mentéssel lépj ki a szerkesztésből:*

```
int button=0;

void setup()

{

        pinMode(13,OUTPUT);

        pinMode(3,INPUT);

}

void loop()

{

        button = digitalRead(3); // nyomogomb a 3-as labon

        if(!button){

                digitalWrite(13,HIGH);

        }

        else{

                digitalWrite(13,LOW);

        }

}
```

3. *huzalozd be a bal oldalon legfelül található nyomógombot a 3-as digitális lábra! ehhez a láb számát írd be a nyomógombon található szövegmezőbe:*

FIGYELEM! Amennyiben a láb száma egyszámjegyű, egy eléírt 0-val ki kell egészíteni kétszámjegyűre!



6. ábra. nyomógomb bekötése

4. *futtasd a programot, majd futás közben próbáld ki a behuzalozott nyomógomb működését! Mikor világít a 13. lábra gyárilag bekötött LED?*
5. *figyeld meg a program bal alsó sarkában található ablakban a* `button` *változó értékének változását a nyomógomb megnyomásakor! Mikor lesz* `0` *és mikor* `1` *az értéke?*
6. *a fenti két megfigyelés alapján a programkód* `if()` *feltételes szerkezetében mit csinálhat a* `button` *előtti felkiáltójel?*
7. *Miket kell módosítani a programban ha a gombot át szeretnénk tenni a 6-os lábra?*

A mérési jegyzőkönyvnek a következő elemeket kell tartalmaznia:

- név, NEPTUN-kód
- 4,5,6,7 kérdésekre adott válaszok

3. mérés: ADC

A mérés célja az analóg-digitális átalakító bemutatása és használatának elsajátítása.

ADC

A mikrovezérlő, mint digitális hálózat önmagában alkalmas a digitális világgal való interfészelésre, egyszerűen összeköthető más digtális jelszinteket használó elemekkel. A gyakorlatban azonban a fizikai valóság egy-egy mérni kívánt paraméterét az adott érzékelő elem a legtöbb esetben analóg áram vagy feszültségjellé alakítja, amelyet a későbbi feldolgozhatóság érdekében digitális jellé kell alakítanunk.

Erre a feladatra alkalmas az ún. ADC, vagy analog to digital converter, amely a bemenetére adott analóg jelet digitálissá alakítja. Az ADC működését itt részletesen nem tárgyaljuk, de felhívjuk a figyelmet két nagyon fontos paraméterre amire figyelnünk kell:

- **felbontás**: arról ad inforációt, hogy az ADC milyen finomsággal tudja a bemeneti jelet felbontani. Az értékét bit-ben szokás megadni. Egy 10bit-es ADC a bemeneti jelet 2^{10} azaz 1024 részre tudja felbontani. Ez azt jelenti, hogy 5V-os rendszer esetén, a bementre érkező 0-5V közti jelből egy 0 és 1023 közti egész számot állít elő. Ebből kiszámolható, hogy a legkisebb feszültségváltozás, amelyet érzékelni tudunk egy ilyen ADC-vel: $\frac{5}{1024} = 0.0048\ V$.
- **sebesség**: a működésből adódóan a fenti átalakítás időbe telik, azaz az analóg jel beérkezése és a számérték előállítása között bizonyos késleltetés lép fel. Bár ez a mai ADC-k esetén még az olcsóbb kivitelekben is jellemzően néhány μs értékű, ez az érték felső korlátot szab az elérhető maximális mintavételi frekvencia értékére.

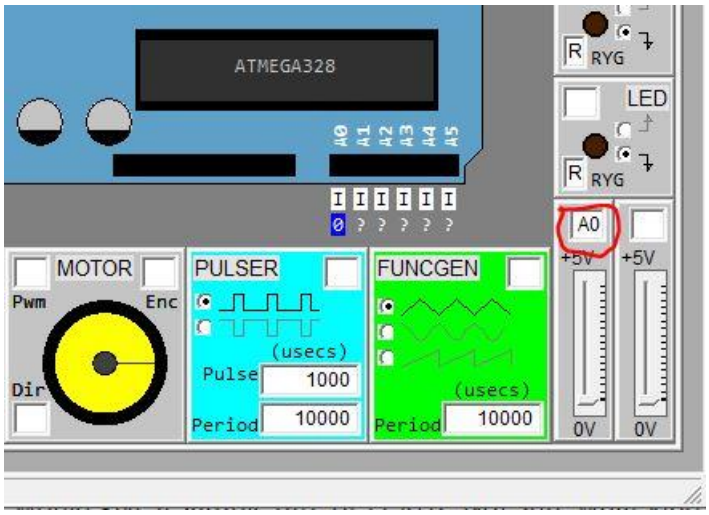
Az Arduino UNO panelen található ATMega mikrovezérlőbe épített 6 darab ADC esetén a felbontás 10bit, a sebesség pedig kb. 100 μs.

Feladatok

1. *hozz lére egy új mentést a szimulátorban, majd a kódszerkesztő ablakba másold be az alábbi kódot:*

```
int analog0=0;

void setup()

{

}

void loop()

{

        analog0 = analogRead(0);

        delay(500);

}
```

2. *Az analóg jel előállítására először a kézzel beállítható feszültségforrást fogjuk használni: huzalozzuk be az első feszültségforrást az A0 analóg lábra:*



7. ábra. A feszültségforrás bekötése

3. *futtasd a programot, majd az egérrel állítsd be különböző jelszintet a csúszka segítségével. Figyeld meg, hogyan változik az analog0 változó értéke!*

4. *huzalozd be valamelyik LED-et a 12-es labra, majd cseréld ki a programot az alábbira:*

```
int analog0=0;

void setup()

{

        pinMode(12,OUTPUT);

}


void loop()

{

        analog0 = analogRead(0);

        if(analog0 > 511){

                digitalWrite(12,HIGH);

        }

        else{

                digitalWrite(12,LOW);

        }

        delay(500);

}
```

5. *futtasd a programot és állítsd a feszültségszint értékét a csúszkával. A program alapján fogalmazd meg mikor kell világítania a LEDnek. Mi történik ha gyorsan változtatod a feszültségszintet? Mi lehet ennek az oka?*

6. *A szemléletesség kedvéért a programban az ADC két mintavétele közti időt mesterségesen megnöveltük 500ms-al. Csökkentsd a programban a delay() függvény paraméterét! Vizsgáld meg, hogy kb. mekkora értéknél van az a határ ahol már nem tudod olyan gyorsan változtatni a feszültségszintet, hogy azt a rendszer ne érzékelje!*

7. *A program az eddigiekben a ADC által szolgáltatott egész számot írta ki. A korábbiakban ismertetett összefüggés alapján számoljuk ezt most vissza feszültség értékre a programban! Módosítsd a programot az alábbiaknak megfelelően:*

```
int analog0=0;

float analogvoltage;

void setup()

{

        pinMode(12,OUTPUT);

}


void loop()

{

        analog0 = analogRead(0);

        analogvoltage = analog0*5/1023.0;

        if(analog0 > 511){

                digitalWrite(12,HIGH);

        }

        else{

                digitalWrite(12,LOW);

        }


}
```

8. *Futtasd a programot, módosítsd a feszültségszintet és nézd meg milyen értékeket vesznek fel a változók.*
9. *Módosítsd a programot úgy, hogy az analogvoltage változó értékét kiszámító képlet végén az 1023.0 számértékből töröld ki a tizedespontot és a 0-t. Ez a szám értékén nem változtat, azonban mégis változást okoz a program futásában. Mi ez a változás? Mi lehet ennek az oka?*

A mérési jegyzőkönyvnek a következő elemeket kell tartalmaznia:

- név, NEPTUN-kód
- 5,6,9 kérdésekre adott válaszok