Matematikai és Informatikai Intézet

Szkenner megvalósítása egér szenzorral

Készítette
Bodnár Máté
Programtervező informatikus BSc

Témavezető
Dr. Geda Gábor
Egyetemi docens

Eger, 2024

Tartalomjegyzék

Bevezetés

A digitalizálás egyre nagyobb szerepet játszik az életünkben, különösen a dokumentumok kezelésében és tárolásában, hiszen a sok ideig tárolt papír dokumentumok elveszhetnek vagy könnyen sérülhetnek.

Manapság a családi fotókat vagy régebbi képeket is célszerű digitalizálni, hogy tovább megmaradjon. A digitalizálással lehetőségünk nyílik arra is hogy rendszerezzünk hivatalos iratokat, így könnyebben tudunk majd hozzájuk férni és akár keresni is köztük.

Bár számos szkenner van a piacon, viszont ezek drágák és nagy helyigényűek. Ezért egy saját készítésű szkenner jobban illeszkedik majd a mi igényeinkhez, valamint sokkal olcsóbb mint egy boltban vett.

Ebben a projektben megmutatom, hogy lehet egy egérszenzort felhasználni hivatalos iratok vagy egyéb más papír alapú dokumentumok szkennelésére. A szenzor két sínnel vízszintesen és függőlegesen fog mozogni, és soronként készíti el a képkockákat. Ezután egy C# alkalmazás feldolgozza azokat és kialakít belőlük egy nagy dokumentumot.

1. fejezet

Bevezető

1.1. Motiváció

Az ötletem mögött több tényező is áll. Elsősorban szerettem volna egy olyan eszközt létrehozni, amely megfizethető alternatívát nyújt a drága szkennerekkel szemben. Az egérszenzorok könnyen beszerezhetők és viszonylag olcsók, ezért jó választásnak tűntek egy saját fejlesztésű szkennerhez. Ez különösen hasznos lehet olyan helyeken, ahol a költségek csökkentése nagyon fontos, például iskolákban vagy kisebb cégeknél.

Mindig is érdekelt, hogyan lehet egy meglévő egyszerű technológiát új és kreatív módon felhasználni. Az egérszenzorokat alapvetően mozgásérzékelésre tervezték, de ebben a projektben megszeretném mutatni, hogy dokumentumok szkennelésére is alkalmasak.

Emellett fontos számomra, hogy egy olyan eszközt alkossak, amelyet egy egyszerű felhasználó is használhat, otthon vagy akár a munkahelyén anélkül, hogy drága berendezésekre kellene költenie.

Valamint kihívást látok ebben a projektben, hogy hogyan is tudom ezt megvalósítani. Izgalmas feladat az, hogy ötvözzem az informatikát az elektronikával. Ez nem csak a szakmai tudásomat fejleszti, hanem egy olyan eredményt ad, amelyre büszke lehetek.

1.2. Célkitűzés

A szakdolgozatom[1] célja egy olyan szkenner létrehozása, amely egy egyszerű egérszenzort használ a dokumentumok fekete-fehér beolvasására. Az eszköz működésének alapja, hogy a szenzor monokróm felvételeket készít a dokumentumról, majd ezeket egy számítógépes programmal feldolgozom és összeállítom egy nagy dokumentummá. Mivel a szenzor csak egy 30 x 30-as képet rögzít így azt interpolációval felnövelem. Ezek az algoritmusok lehetővé teszik, hogy a képet megnöveljük kevés minőség vesztéssel.

A dolgozat eredményeként egy egyszerű és költséghatékony szkennert szeretnék létrehozni, amely hasznos és megkönnyíti az emberek életét.

2. fejezet

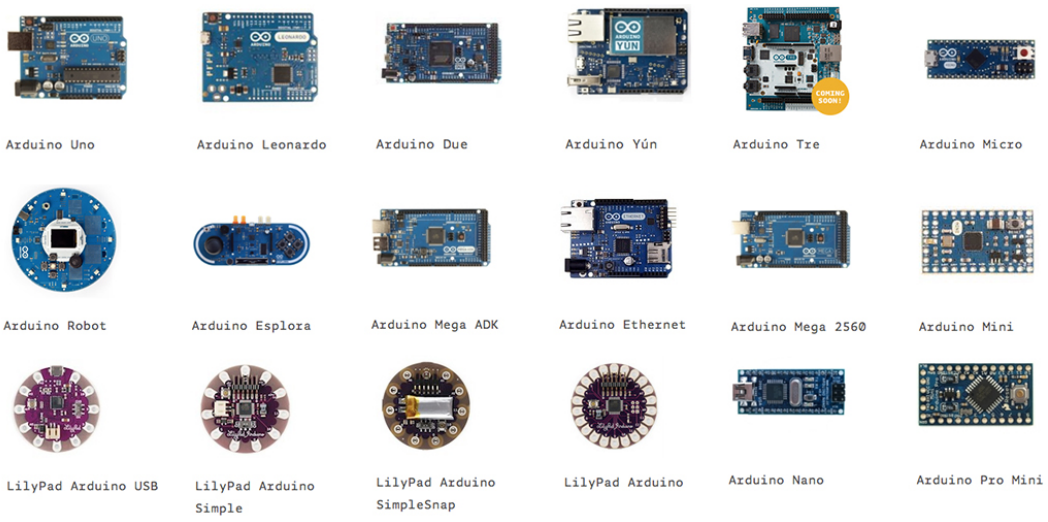
Felhaszánlt technológiák

Ebben a fejezetben a szakdolgozatomban használt technológiákról és azok előnyeiről, fogok beszámolni.

2.1. Arduino

2.1.1. Arduino platform bemutatása

Az Arduino[2][3] egy nyílt forráskódú platform, amiket az elektronikai projektekhez találtak ki, majd bekerült az oktatásba is, oktatási céllal. Sokan használják egyszerűbb feladatok automatizálására vagy akár okos otthon rendszerek kialakítására. Ezek mellett manapság már az ipari alkalmazásuk sem ritka. A működéséhez szükség van egy mikrokontrollerre, valamint egy fejlesztő környezetre az Arduino IDE-re, amivel általában USB kábelen keresztül tudjuk átküldeni a programot a fizikai eszközre.

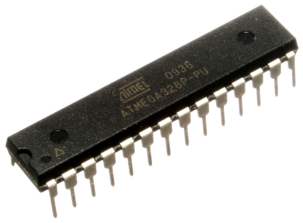


2.1. ábra. Arduino család néhány fajtája

2.1.2. Arduino Nano

Elsősorban UNO-val[4] terveztem elkészíteni ezt a projektet, de helyszűkében kénytelen voltam egy kisebb mikrokontroller után kutatni. Ekkor esett a választásom az Arduino Nano-ra[5]. Az Arduino Nano az Arduino UNO kompaktabb változata, mely ugyanazt az ATmega328 mikrovezérlőt használja, így teljesítményében és képességeiben nagyon hasonló az UNO-hoz. A mikrovezérlő tartalmaz:

- Processzort
- Memóriát
- Perifériákat:
  - Időzítő áramkörök
  - Analóg és digitális be- és kimenetek
  - Kommunikációs interfészek (UART, SPI, I2C) és egyéb funkciók

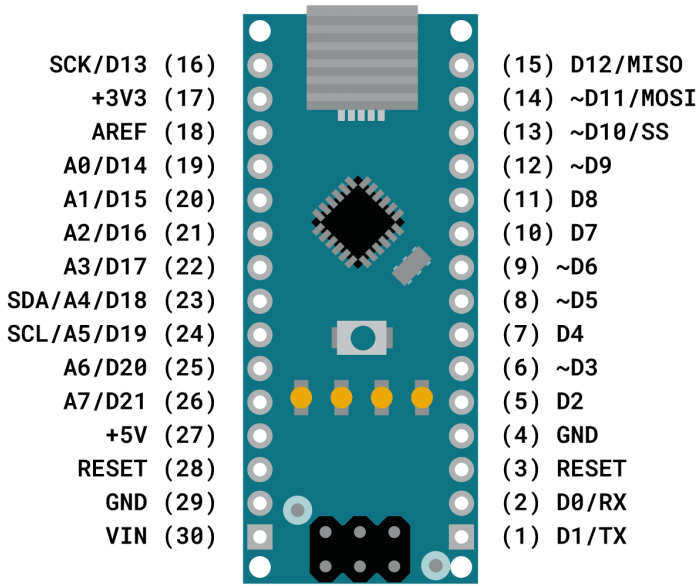


2.2. ábra. ATmega328P mikrovezérlő

Ezek segítségével tudunk szenzorjeleket mérni, nyomógombok vagy más beviteli eszközök állapotát beolvasni. A Nano áramköri lap szerepe az, hogy a mikrovezérlő lábait kivezesse. Így kényelmesebben és egyszerűbben rá tudjuk kötni a különböző eszközöket, amiket vezérelni szeretnénk, vagy értékeket beolvasni róluk.

A mikrokontrollereken általában nem fut operációs rendszer, ezért minden erőforrást a feladatra összpontosít és egy garantált maximális idő alatt képes végre hajtani a feladatokat.

Ahhoz hogy külső eszközöket és áramköröket rátudjunk csatlakoztatni, ismernünk kell a Nano lábkiosztását, amikre a kódból tudunk hivatkozni és vezérelni őket, áramköri lapon fel van tüntetve, hogy melyik lábat melyik számmal érjük el. A Nano-n 14 digitális ki- és bemenet található (D0–D13), valamint 8 analóg bemenet (A0–A7), melyek közül az A0–A5 lábak digitális bemenetként és kimenetként is használhatók. A Nano-n található továbbá fix 3.3,V és 5,V-os kimeneti feszültségű láb is. Ez a lábkiosztás látható a 2.3. számű ábrán.



2.3. ábra. Arduino Nano lábkiosztása

2.1.3. Miért választottam az Arduino Nano-t?

Szakdolgozatom során az Arduino Nano-t választottam, mivel több olyan előnye van, amelyek különösen fontosak a projektem szempontjából:

- **1. Kompakt méret.** A Nano mérete kisebb, mint az UNO-é, így ideális választás volt, mivel helyszűkében voltam.
- **2. Egyszerű használat és támogatottság.** Az Arduino Nano széles körben elterjedt mikrokontroller, amelyhez könnyen elérhetőek könyvtárak és példakódok, ezzel egyszerűsítve a fejlesztést.
- **3. Megfelelő teljesítmény és elegendő számú I/O port.** A Nano szintén ATmega328 mikrovezérlőt használ, amely tökéletesen megfelel az ADNS-9800 szenzor kezelésére, a szenzor mozgatására, valamint az adatok továbbítására.
- **4. Megbízhatóság és stabilitás.** A stabil működés létfontosságú, mivel a projektem dokumentumok folyamatos szkennelését, adatgyűjtést és adatátvitelt igényel hosszabb ideig.
- **5. Költséghatékonyság.** Az Arduino Nano az egyik legkedvezőbb árú fejlesztőeszköz, amely tökéletesen kielégíti a projekt követelményeit.
- **6. Egyszerű programozhatóság és csatlakoztatás.** Az Arduino Nano könnyen programozható az Arduino IDE használatával, programokat egyszerűen USB-n keresztül tölthetünk fel rá.

2.1.4. Az Arduino alkalmazási területei

- **Kezdő projektekhez.** Az Arduino Board-ok tökéletesek a kezdők számára akik az elektronikát és az informatikát szeretnék ötvözni. A fejlesztő környezetet egyszerű kezelni, valamint a könyvtárak és a példa kódok is nagyon sokat segítenek azoknak az embereknek akik elkezdenek érdeklődni az ilyen dolgok iránt.
- **Oktatási platform.** Könnyen kezelhetősége miatt szokták alkalmazni, hogy ezekkel az eszközökkel tanítsák meg az elektronika és az informatika működését.
- **Robotikában.** A nagy vállalatok vezetői is felfedezték ezt a technológiát, és gyakran Arduino Board-okat használnak a robotok megvalósításához és vezérléséhez.
- **Zene és művészet.** Az Arduino Board-okat szokták egyszerű hangszerek létrehozására is alkalmazni, vagy már meglévő hangszerekbe beépíteni elektronikus alkatrészként.
- **IoT - Internet of Things.** Legtöbb esetben okos otthon rendszerekbe szokták beépíteni, mert sok szenzort rá tudunk csatlakoztatni amikkel könnyen vezérelhetjük a saját otthonunkat.
- **Viselhető eszközök. (Wearables)** Kompakt méretük miatt könnyen beépíthetőek ruhákba, akár ékszerekbe vagy más hordozható eszközökbe. Amelyekkel mozgásokra reagálhatunk mérhetünk testhőmérsékletet és még sok mást.

2.2. Visual Studio

A projekt szoftveres részét Visual Studio[6] fejlesztői környezettel készítettem el, mert ez áll hozzám a legközelebb. Itt valósítottam meg az interpolációt a kép minőségvesztés nélküli növelését. Valamint a program irányító felületét is itt programoztam le.

2.3. Github

A verziókövetéshez a Githubot[7] használtam azon belül is a Github Desktop-ot, mellyel könnyen tudtam több platformon is dolgozni a projekten, valamint ha valamit elrontottam könnyen vissza tudtam állítani a projektet egy korábbi verzióra.

3. fejezet

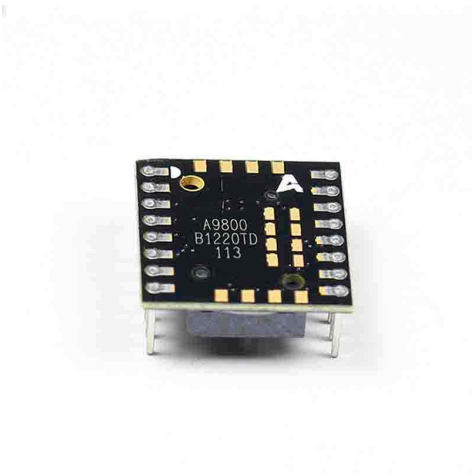
Hardveres megvalósítás

3.1. ADNS-9800 szenzor

Az általam készített szkennernek a lelke a kicsi és olcsó ADNS-9800[8] lézeres optikai érzékelő. Az eredeti felhasználási módja ennek az eszköznek az volt, hogy egerek mozgását kövessék. Ennek a szenzornak az a különlegessége, és azért volt megfelelő számomra, mivel úgy követi a mozgást, hogy folyamatosan képeket készít. Ezáltal fel tudtam használni a projektembe mint egy kis kamerát, valamint gyors képfeldolgozási képességgel rendelkezik, ezért alkalmas dokumentumok beolvasására.

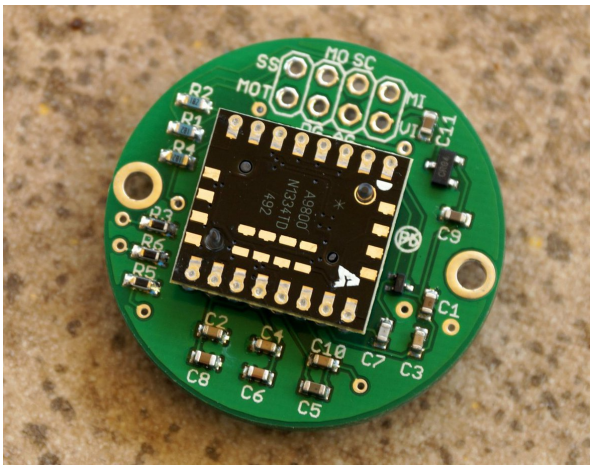
3.1.1. Működése

Egy beépített lézer megvilágítja a szenzor alatt lévő felületet, majd a lencse készít egy képet. A beépített kis processzor dolgozza fel ezeket az adatokat, és számítja ki, hogy az eszköz milyen irányba mozdult el.



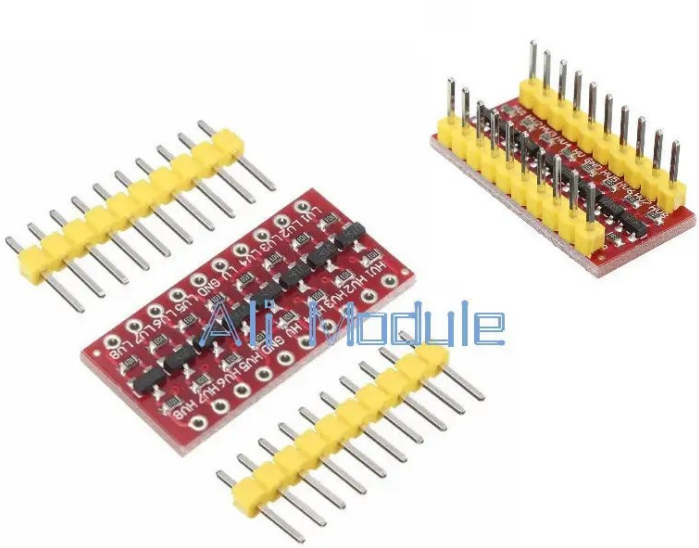
3.1. ábra. ADNS-9800 szenzor

Az általam választott szenzor már nyáklappal volt ellátva így azzal nem volt teendőm. Ezáltal egyszerűbb volt a bekötése és az elhelyezése is.



3.2. ábra. ADNS-9800 szenzor nyák lapra szerelve

Csak egy problémám volt, hogy a nyáklappal szerelt verzió 3.3 V-os logikai szinttel működött viszont az Arduino Nano 5 V-os szinttel. Így be kellett szereznem egy két irányú logikai szintillesztő[12] modult ami átalakította a 3,3 V-os jelet 5 V-ossá. Ehhez a projekthez egy 8 csatornás logikai szintillesztő modult használtam.



3.3. ábra. Két irányú logikai szintillesztő

3.1.2. Adatok beolvasása

Az adatok beolvasása közben a szenzor egy sorokból és egy oszlopokból álló képkockát készít, amely egy 30 x 30-as pixel rács. Ezáltal egy dokumentum szkennelése lehet akár több ezer képkocka is. A működése a következő lépésekből áll:

1. **Feltöltjük a firmware-t a szenzorra.** Erre azért van szükség, mert enélkül a szenzor csak egy hardveres érékelő, amely a lézerrel megvilágított felületről képeket készít. Ahhoz, hogy ezt az információt feldolgozza és továbbítsa szüksége van egy vezérlőprogramra, azaz firmware-re. Ezt a firmware-t minden bekapcsolás után fel kell tölteni, mert a szenzornak nincs állandó memóriája. A firmware feltöltését a 3.1. számú metódus végzi. A regiszterek megfelelő beállítása után a firmwaret soronként továbbítja az SPI buszon keresztül a szenzornak. Ebben a metódusban fontos az időzítés hogy a szenzor helyesen értelmezze az adatokat.

kód 3.1. Firmware feltöltése

```
1   void adns_upload_firmware(){
2     adns_write_reg(REG_Configuration_IV, 0x02);
3     adns_write_reg(REG_SROM_Enable, 0xd);
4     delay(10);
5     adns_write_reg(REG_SROM_Enable, 0x8);
6     adns_com_begin();
7     SPI.transfer(REG_SROM_Load_Burst | 0x80);
8     delayMicroseconds(15);
9     unsigned char c;
10    for(int i = 0; i < firmware_length; i++){
11      c = (unsigned char)pgm_read_byte(firmware_data + i);
12      SPI.transfer(c);
13      delayMicroseconds(15);
14    }
15    adns_com_end();
16  }
```

2. Majd a **sendFrame()** metódust meghívva készítünk egy képet a szenzorral. Amit egy nagy sebességű, full–duplex kommunikációs protokollal (Serial Peripheral Interface – SPI) kiolvasunk és eltároljuk, ahol a kommunikáció egy órajel vezérelt mechanizmussal működik.

kód 3.2. sendFrame metódus

```
1   void sendFrame() {
2     adns_write_reg(REG_Power_Up_Reset, 0x5A);
3     delay(50);
4     adns_write_reg(REG_LASER_CTRL0, 0x00);
5     adns_write_reg(REG_Frame_Capture,0x93);
6     delayMicroseconds(120);
7     adns_write_reg(REG_Frame_Capture,0xc5);
8     delayMicroseconds(120);
9     delay(1);
10    adns_com_begin();
11    delayMicroseconds(100);
12    byte readys = 0;
13    while(readys == 0){
14      SPI.transfer(REG_Motion & 0x7f);
15      delayMicroseconds(100);
16      readys = SPI.transfer(0);
17      readys = readys & 1;
18      delayMicroseconds(20);
19    }
20    SPI.transfer(REG_Pixel_Burst & 0x7f);
21    delayMicroseconds(100);
22    Serial.print("FRAME:");
23    for (int i = 0;i<900;i++){
24      byte pixelValue = SPI.transfer(0);
25      Serial.print(pixelValue);
26      Serial.print("␣");
27    }
28    delayMicroseconds(15);
29    adns_com_end();
30    delayMicroseconds(5);
31    Serial.println();
32  }
```

A metódus működése:

1. **Szenzornak kiadjuk, hogy készítsen egy képet.** Ekkor bele írjuk a megfelelő értéket a REG_Frame_Capture regiszterbe. A szenzor ekkor egy 30 x 30 pixeles képkockát olvas be.
2. **Várunk amíg a szenzor elkészíti a képet.** Az Arduino egy ciklusban folyamatosan ellenőri a szenzor állapotát ezért amíg a szenzor nem jelzi, nem olvassuk ki az értéket.
3. **A szenzor továbbítja a képkockát.** A Pixel Burst regiszteren keresztül történik a képkocka eljuttatása a szenzortól az Arduino-ig SPI kapcsolattal. Az SPI kapcsolat négy vezetéken keresztül történik:
   - **MOSI** (Master Out, Slave In): Az adatok a mikrokontroller felől a szenzor felé mennek.
   - **MISO** (Master In, Slave Out): Az adatok a szenzor felől mennek az Arduino felé.
   - **SCLK** (Serial Clock): Az órajel, amelyet az Arduino generál.
   - **CS** (Chip Select): Az adott szenzort választja ki a kommunikációhoz.
4. **Az Arduino továbbítja az adatokat a számítógépre.** Az adatok soros kommunikáción keresztül kerülnek a számítógépre ahol az feldolgozza ezeket. Minden képkocka "FRAME:" előtaggal kezdődik, amit a szenzor által olvasott színintenzitás értékek követnek. Egy példa frame sor:

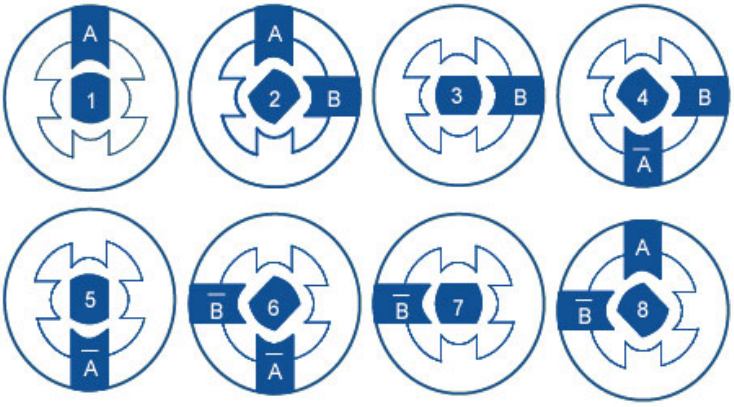
   ```
   FRAME: 124 128 130 122 120 119 118 117 116 ...
   ```

3.1.3. Szenzor mozgása

A szenzort két motor mozgatja az egyik vízszintesen a másik pedig függőlegesen. A projekthez egy bipoláris valamint egy unipoláris léptetőmotort[14] használtam. Mivel egy meglévő szkennerbe építettem bele az egész elektronikát, ezért a függőleges tengely mozgatására szolgáló bipoláris léptetőmotor már adott volt. Viszont a szenzor nagyon kis képet készít ezért finom mozgásokra volt szükségem, ebben segítséget nyújtott a már előre beépített fogaskerekek, amik áttétként funkcionálnak. Ezáltal a motor egy lépése jelentősen kisebb elmozdulás volt fizikailag, így jól tudtam pozicionálni.

3.4. ábra. MITSUMI M42SP-4NP bipoláris motor a függőleges mozgáshoz

Mindkét motornál half-step vezérlési módot használtam, hogy még finomabb és pontosabb mozgást tudjak végezni velük. Ennek az a lényege, hogy a motor kétszer annyi lépésben teszi meg ugyanazt a fordulatot. Ez úgy lehetséges, hogy a négy tekercs közül először egy tekercs kap áramot, majd az első és a második egyszerre, majd csak a második, így a lépés szám a duplájára nő a szögelfordulás pedig a felére csökken.



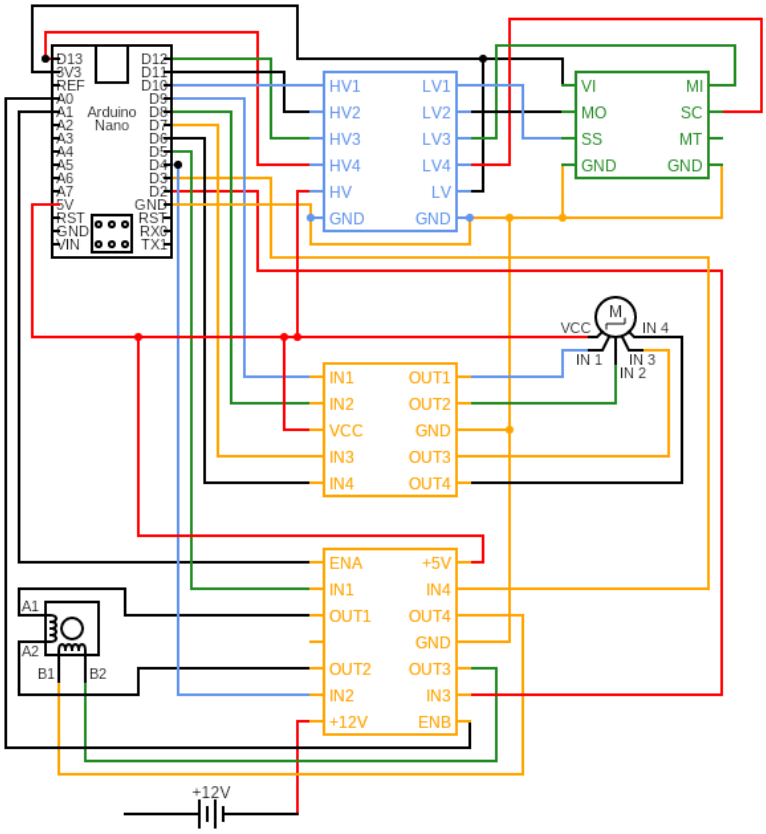
3.5. ábra. Half Step vezérlés működése

A vízszintes mozgáshoz beépítettem egy unipoláris léptetőmotort, amely egy ULN2003 motorvezérlő modullal működik, amit nagyon gyakran használnak ilyen kis apró finom mozgásokhoz, amire nekem is szükségem volt. Mivel ez a motor úgy van felépítve, hogy alapból áttétek vannak benne ezért azzal itt sem kellett foglalkoznom.

3.6. ábra. Unipoláris léptetőmotor a vízszintes mozgáshoz

A motorok egy egy bordás szíjas sínen mozognak, ezáltal a mozgás stabil lesz és csúszásmentes. Mindkettő végén van egy végállás kapcsoló, ha a vízszintes sor a végére ér ezt aktiválja és lentebb megy egy sort, valamint vissza megy a sor elejére a szenzor. Ekkor küldd az Arduino egy ”NEW_ROW” kulcsszót a Serial porton a számítógép felé, ezzel jelezve a feldolgozó szoftvernek, hogy új sor következik az olvasásban. Ha pedig a függőleges végálláskapcsoló válik aktívvá akkor pedig az ”END” kulcsszót küldi el a számítógép felé, jelezve, hogy vége a dokumentumnak és vissza tér a kezdeti állásba.

3.2. Hardveres bekötés



3.7. ábra. Hardveres bekötés

Már kész csak nem biztos hogy jó

4. fejezet

Szoftveres megvalósítás

4.1. Az alap változók

A feldolgozó szoftver Serial porton kapja meg az adatokat az Arduino-tól. Van két darab konstans értékem az egyik a beérkező képkocka szélessége a másik pedig a magassága, mivel a beérkező képkocka 30 x 30 pixel ezért ez a két érték 30. Van egy buffer-em ahol a Serial porton beérkező adatokat tárolom, erre azért van szükség mert nem biztos, hogy egy sor át tud jönni egyszerre. Valamint van egy Bicubic osztály példányom amivel a Bikubik interpolációt végzem. Végül van egy row és egy column változóm ami annak segítségére szolgál, hogy összerakjam a képkockákat egy nagy dokumentummá. Ezt a nagy dokumentumot tartalmazza a listGrid nevű változó, ami egy lista amiben listák vannak amiben egész típusú tömbök vannak.

```
1  private SerialPort serialPort;
2  private const int FrameWidth = 30;
3  private const int FrameHeight = 30;
4  private StringBuilder buffer = new StringBuilder();
5  private Bicubic resizer = new Bicubic(FrameWidth, FrameHeight, 2);
6
7  int row = 0;
8  int column = 0;
9
10 private List<List<int[]>> listGrid = new List<List<int[]>>();
```

4.2. Inicializáló metódusok

Van egy soros portot beállító metódus. Aminek segítségével megnyitom a soros portot és beállítom a megfelelő BaudRate-t. Jelen esetben 115200 bit/másodperc. Hozzárendelem a serialPort változó DataReceived eseményéhez a saját metódusomat, ami azt jelenti, hogy amikor érkezik adat a soros porton akkor meghívja azt a metódust. Ez-

után megnyitom a soros portot hogy tudjam olvasni az adatokat róla, és írok egy ”reset” kulcsszót az Arduino felé ezen a porton keresztül, hogy alaphelyzetbe tegyem a szenzort.

```
1  private void InitializeSerialPort()
2  {
3    serialPort = new SerialPort("COM3", 115200);
4    serialPort.DataReceived += SerialPort_DataReceived;
5    serialPort.Open();
6    serialPort.Write("reset");
7  }
```

5. fejezet

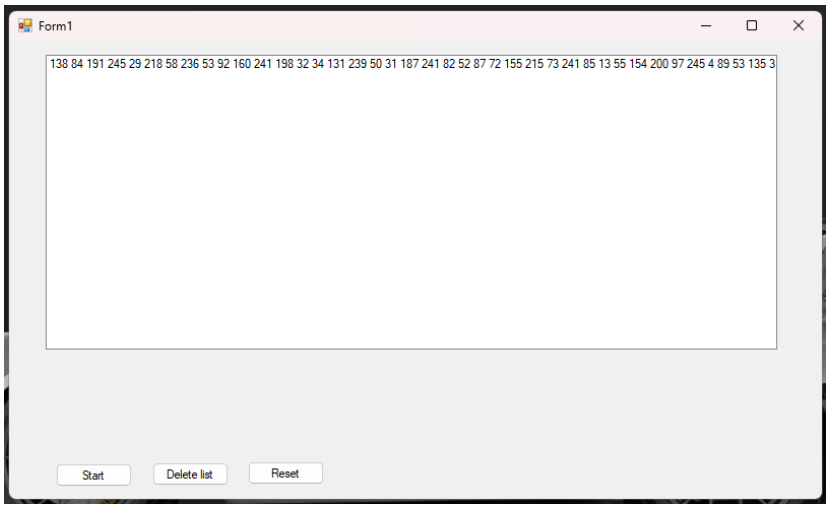
Tesztelés

A programomat az elejétől fogva folyamatosan teszteltem, először a szoftveres részt, majd miután megérkezett hozzám a szenzor elkezdtem azt tesztelni.

A szoftver teszteléséhez úgy fogtam hozzá, hogy egy alap mátrixot beadtam az általam írt algoritmusnak és megnéztem, hogy milyen pontosan számolja ki a pixelek közötti értékeket. Aztán megfogtam egy képet és megnéztem, hogy azzal, hogy viselkedik. Az eredmény az lett hogy a duplájára növelte a kép méretét ami a célom is volt. Ezután összpontosítottam arra, hogy az adatot megkapjam az Arduino-tól, tehát elkezdtem a soros port kommunikációt tesztelni. Fogtam egy Arduino-t és írtam rá egy példa programot aminek csak annyi dolga volt, hogy elküldjön egy 900 elemű tömböt.

```
1  void setup() {
2    Serial.begin(9600);
3  }
4
5  void loop() {
6    for (int i=0;i<900;i++){
7      Serial.print(int(random(0,255))+"␣");
8    }
9    Serial.println();
10   delay(1000);
11 }
```

A szoftverben ehhez megírtam a szükséges metódust ami arra szolgál, hogy fogadja az adatokat. Ezt követően elkezdtem ki alakítani a szoftver felületét, készítettem egy tesztelő Windows Forms alkalmazást ami állt egy listBox-ból valamint egy pár gombból. Egy **Start** gomb amivel elindítottam az adat küldést az Arduino felől, egy **Delete list** amivel törölni tudtam a lista eddigi adatait, valamint egy **Reset** gomb amivel a szenzort tudtam újraindítani.



5.1. ábra. Teszt Windows Forms alkalmazás

Aztán következhetett a szenzor tesztelése. Először egy fehér lapon próbálkoztam amire kisbetűmérettel volt rá írva pár betű, viszont valamiért nem kaptam egy szép képet, ezért elkezdtem mozgatni a szenzort és rájöttem, hogy a fókusz távolság nem volt a megfelelő. Így azt kellett finom hangolnom, egyszerű kártya lapokat használtam, hogy megtaláljam a megfelelő távolságot. Miután sikerült beállítanom a kellő távolságot a papírtól, elkezdtem kinyerni a szenzorból a képeket. Megírtam hozzá a teszt alkalmazást amivel megjelenítem, hogy mit lát a szenzor, és nagyon nagy örömömre sikerült képet kinyernem a szenzorból.

Összegzés

Tapasztalatok amiket szereztem a projekt megvalósítása közben Tovabbfejlesztési gondolatok

színes vagy szürke képet szeretne beolvasni soros porton küldok egy bitet hogy színes vagy szürke legyen a kép a studiobol felbontásra vonatkozóan például feles átfedéssel

Források

- 2.1. ábra: *https://predictabledesigns.com/wp-content/uploads/2017/10/HeroImage.png*
- 2.2. ábra: *https://techfun.hu/wp-content/uploads/2017/09/11.jpg*
- 2.3. ábra: *https://www.makerguides.com/wp-content/uploads/2020/10/arduino-nano-pinout-768x603.png*
- 3.1. ábra: *https://www.sicstock.com/cdn/shop/products/551554.jpg?v=1544514465*
- 3.3. ábra: *https://www.elektrobot.hu/items/3607_1.webp*
- 3.4. ábra: *https://www.picmicrolab.com/wp-content/uploads/2014/04/MITSUMI-Stepping-Motor-M42SP-4NP.jpg*
- 3.5. ábra: *https://www.orientalmotor.com/images/stepper-motors/stepper-motors-half-step.jpg*
- 3.6. ábra: *https://www.hwlibre.com/wp-content/uploads/2024/10/28byj-48-circuito.jpg*

Irodalomjegyzék

[1] Github link a szakdolgozathoz: https://github.com/Bmate2/SZAKDOGA

[2] Arduino: https://www.arduino.cc/en/Guide/Introduction

[3] What is an Arduino: https://learn.sparkfun.com/tutorials/what-is-an-arduino/all

[4] Arduino UNO: http://dx.doi.org/10.6084/m9.figshare.11971794.v1

[5] Arduino Nano: https://docs.arduino.cc/hardware/nano/

[6] Microsoft Visual Studio: https://visualstudio.microsoft.com/

[7] Github: https://github.com/

[8] ADNS-9800 adatlap:https://datasheet.octopart.com/ADNS-9800-Avago-datasheet-10666463.pdf

[9] ADNS-9800 vásárlás: https://www.tindie.com/products/citizenjoe/adns-9800-motion-sensor/

[10] ADNS-9800 tutorial get travel distance: https://www.instructables.com/Arduino-Tutorial-ADNS-9800-Laser-Mouse-Traveled-Di/

[11] ADNS-9800 5 V mód aktiválása: https://forum.arduino.cc/t/using-avago-adns9800/292172/2

[12] Logikai szintillesztő modul: https://www.elektrobot.hu/termek.php?filename=3607.html&i=3607

[13] Soros kommunikáció: https://docs.arduino.cc/language-reference/en/functions/communication/serial/

[14] Léptető motorok: https://docs.arduino.cc/learn/electronics/stepper-motors/

[15] Bicubic Interpolation by Computerphile: https://www.youtube.com/watch?v=poY_nGzEEWM

[16] Bicubic Interpolation from wikipedia: https://en.wikipedia.org/wiki/Bicubic_interpolation

[17] Catmull-Rom Spline: https://en.wikipedia.org/wiki/Centripetal_Catmull%E2%80%93Rom_spline

[18] Kép nagyítás Bikubik interpolációval: https://medium.com/@amanrao032/image-upscaling-using-bicubic-interpolation-ddb37295df0

[19] Windows Forms: https://learn.microsoft.com/hu-hu/dotnet/visual-basic/developing-apps/windows-forms/

[20] Listbox osztály: https://learn.microsoft.com/en-us/dotnet/api/system.windows.forms.listbox?view=windowsdesktop-9.0

[21] LaTeX Arduino stílus fájl: https://github.com/trihedral/ArduinoLatexListing

Nyilatkozat

Alulírott *Bodnár Máté*, büntetőjogi felelősségem tudatában kijelentem, hogy az általam benyújtott, *Szkenner megvalósítása egér szenzorral* című szakdolgozat önálló szellemi termékem. Amennyiben mások munkáját felhasználtam, azokra megfelelően hivatkozom, beleértve a nyomtatott és az internetes forrásokat is.

Aláírásommal igazolom, hogy az elektronikusan feltöltött és a papíralapú szakdolgozatom formai és tartalmi szempontból mindenben megegyezik.

Eger, 2025. március 3.

aláírás